SHIH YUNG CHIN

Utilização da modelagem e simulação para
fins de análise comparativa de desempenho
entre o arranjo físico funcional e o arranjo
físico distribuído

São Carlos
2010

SHIH YUNG CHIN

Utilização da modelagem e simulação para fins de análise comparativa de desempenho entre o arranjo físico funcional e o arranjo físico distribuído

Tese apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia Mecânica

Área de Concentração:

Manufatura

Orientador:

Prof. Dr. Eduardo Vila Gonçalves Filho

São Carlos
2010

AGRADECIMENTOS

Em primeiro lugar, não posso deixar de agradecer a Deus por ter me dado sabedoria para tornar possível a realização deste trabalho. Em muitos momentos, tive que realmente usar de muita criatividade para a solução de diversos problemas. Este trabalho, com certeza, foi um tremendo desafio.

Aos meus pais que me deram muita força. Em muitos momentos, fui perdendo a esperança de concretizar este trabalho, mas não desanimei.

Ao orientador Prof. Eduardo Vila Gonçalves Filho pela sua paciência durante todos esses anos, acreditando no meu potencial. Agradeço pelo seu rigor e, principalmente, pelos seus constantes questionamentos. Aprendi muito com ele.

Ao Prof. e ex-orientador de mestrado Arthur José Vieira Porto pelas suas valiosas sugestões, que foram fundamentais para o desenvolvimento deste trabalho. Agradeço também à sua família que sempre nos recebeu de braços abertos em sua casa para confraternização entre os colegas do laboratório.

Aos colegas do Laboratório de Simulação de Sistemas de Eventos Discretos que foram importantes nos momentos de descontração. Agradeço, em particular, ao doutorando Hilano José de Rocha Carvalho pelo seu apoio no desenvolvimento deste trabalho.

À secretaria de Pós graduação que sempre me atendeu bem, principalmente nos momentos que eu mais precisei.

À CAPES pela bolsa concedida, que foi fundamental para que eu me dedicasse exclusivamente ao trabalho e à publicação de artigos científicos.

RESUMO

Shih, Y. C. **Utilização da modelagem e simulação para fins de análise comparativa de desempenho entre o arranjo físico funcional e o arranjo físico distribuído.** 2010. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

Os sistemas produtivos tradicionais encontram dificuldades para atender eficientemente aos novos perfis de mercado (caracterizados principalmente pela alta variedade e pela alta variabilidade na demanda), por isso novos tipos de arranjos físicos estão sendo propostos na literatura e dentre eles, estão os arranjos físicos distribuídos. Nos arranjos físicos distribuídos, as máquinas são espalhadas no chão de fábrica, possibilitando assim formar células virtuais mais eficientes. Verifica-se que os trabalhos da literatura utilizam-se de formulação matemática sem levar em consideração concomitância de uso dos recursos produtivos. Existem também trabalhos de pesquisa que utilizam-se das ferramentas de modelagem e simulação, porém a célula virtual abordada difere-se do conceito teórico. O objetivo principal deste trabalho é fazer uma análise comparativa de desempenho (em termos de tempo e custo de consolidação dos pedidos) entre o arranjo físico funcional e o arranjo físico distribuído. Para obter os resultados, este trabalho primeiramente realiza a construção de um modelo de simulação (em Rede de Petri Colorida) referente à empresa fictícia estudada. Em seguida, realiza a simulação deste modelo para diversas situações de operação. Outra importante contribuição deste estudo é poder fazer a simulação para várias amostras de qualquer um dos dois arranjos físicos em análise (podendo conter, em cada amostra, qualquer quantidade de máquinas) sem a necessidade de fazer alteração na estrutura lógica.

Palavras-chave: Modelagem e simulação. Arranjo físico. Análise de desempenho.

ABSTRACT

Shih, Y. C. **The using of modeling and simulation for the performance comparative analysis between functional layout and distributed layout.** 2010. Thesis (Doctoral) – School of Engineering of São Carlos, University of São Paulo, São Carlos, 2010.

Recently researches related to new production systems have been proposed in the literature since traditional ones can not attend efficiently to the new profiles of market (especially characterized by higher variety and variability of demand). In the distributed layout, one of the proposed in the literature, machines are scattered on the floor to make possible an efficient virtual cellular formation. Many researches in the literature only consider mathematical formulation but any other concomitancy issue has not been considered. There are also research works employing modeling and simulation techniques, but the virtual cellular (its formation and how it should work) is far from the original idea previously proposed. The main goal of this research is to perform a comparative study (time and costs for the order consolidation) between functional layout and distributed layout. Results are collected from the simulation model (constructed in Colored Petri Net), which represents a fictitious company. Following, the simulation is conducted by subjecting the constructed model under several conditions of operation. Other contribution of this research is the possibility to simulate several samples for both types of layouts (for any quantity of machines in each sample), without changing the logical structure of the simulation model.

Keywords: Modeling and simulation. Layout. Performance analysis.

LISTA DE FIGURAS

Figura 2.1 - Representação genérica de um sistema de produção	6
Figura 2.2 - Exemplo de representação de um sistema de manufatura	7
Figura 2.3 - Proposta de sistematização dos estudos de simulação	11
Figura 2.4 - Exemplo de Redes de Petri Coloridas	16
Figura 2.5 - Exemplo de uma Rede de Petri colorida.....	18
Figura 2.6 - Exemplo de um arranjo físico funcional	23
Figura 2.7 - Exemplo de um arranjo físico celular	23
Figura 2.8 - Exemplo de um arranjo físico distribuído	24
Figura 2.9 - Vista superior de um arranjo físico distribuído	25
Figura 2.10 - Alguns exemplos de possíveis células virtuais a serem formadas para a produção da peça requisitada	25
Figura 2.11 - Exemplo de quebra de lote grande em lote pequeno.....	26
Figura 2.12 - Vistas superiores: dois exemplos	28
Figura 3.1 - Vista superior do chão de fábrica	51
Figura 3.2 - Procedimento para a geração de perfil de peças.....	53
Figura 3.3 - Procedimento para a geração do perfil de demanda para todos os clientes	55
Figura 3.4 - Procedimento para o cálculo do número de réplicas de cada máquina	56
Figura 3.5 - Procedimento para a obtenção de todos os possíveis formatos de chão de fábrica	58
Figura 3.6 - Procedimento para a obtenção das distâncias entre os locais (De-Para)	58
Figura 3.7 - Procedimento para a geração do arranjo físico funcional	60
Figura 3.8 - Procedimento para a obtenção do arranjo físico aleatoriamente distribuído	61
Figura 3.9 - Procedimento para o cálculo do grau de distribuição para um arranjo físico maximamente distribuído - parte 1	63
Figura 3.10 - Procedimento para o cálculo do grau de distribuição para um arranjo físico maximamente distribuído - parte 2	64
Figura 3.11 - Possíveis posições do vão vazio	65
Figura 3.12 - Exemplos de arranjo físico funcional com separação ou não de máquinas do mesmo tipo inserindo o vão vazio “0”	66
Figura 3.13 - Procedimento para a geração das diversas posições do vão vazio para o arranjo físico aleatoriamente distribuído e para o arranjo físico maximamente distribuído.....	66
Figura 3.14 - Procedimento para a geração das possíveis posições do vão vazio para o arranjo físico funcional e para o arranjo físico parcialmente distribuído, onde a é posição_nula[iii]	67

Figura 3.15 - Exemplo de alocação de máquinas	69
Figura 3.16 - Procedimento para o preenchimento das máquinas nos locais	70
Figura 3.17 - Procedimento para a separação da matéria-prima por carga da empilhadeira	73
Figura 3.18 - Procedimento para a geração das posições iniciais dos funcionários tipo 3	74
Figura 3.19 - Abordagem por trocas simultâneas	77
Figura 3.20 - Arranjo físico	80
Figura 3.21 - Árvore de rotas	81
Figura 3.22 - Exemplo da dinâmica de funcionamento usando critério baseado em primeira máquina disponível	83
Figura 3.23 - Exemplo da dinâmica de funcionamento usando critério baseado em menor distância.....	84
Figura 3.24 - Exemplo da dinâmica de funcionamento usando critério baseado em menor fila	85
Figura 3.25 - Exemplo da dinâmica de funcionamento usando critério baseado em maior fila e menor distância.....	86
Figura 3.26 - Exemplo da dinâmica de funcionamento usando critério baseado em menor distância e maior fila.....	87
Figura 4.1 - Tela de execução do código de programação para o dimensionamento das réplicas de cada tipo de máquina	90
Figura 4.2 - Tela de execução do código de programação para a obtenção dos possíveis formatos dos locais	91
Figura 4.3 - Tela de execução do código de programação para a obtenção das distâncias da matriz de distância De-Para de uma das configurações	92
Figura 4.4 - Tela de execução do código de programação para a obtenção das distâncias da entrada aos locais e dos locais à saída.....	93
Figura 4.5 - Tela de execução do código de programação para a obtenção do arranjo físico funcional.....	94
Figura 4.6 - Término da execução do código de programação do procedimento proposto para a geração do arranjo físico funcional.....	94
Figura 4.7 - Tela de execução do código de programação responsável pela alocação dos transportadores intermediários nos locais	95
Figura 4.8 - Término da execução do código de programação do procedimento proposto para a geração do arranjo físico aleatoriamente distribuído	96
Figura 4.9 - Término da execução do código de programação do procedimento proposto para a geração do arranjo físico parcialmente distribuído.....	97

Figura 4.10 - Tela de execução do código de programação para a obtenção do arranjo físico maximamente distribuído	97
Figura 4.11 - Término da execução do código de programação para a obtenção do arranjo físico maximamente distribuído	99
Figura 4.12 - Informações gravadas no arquivo externo referentes ao número de amostras de arranjo físico funcional.....	106
Figura 4.13 - Informações gravadas no arquivo externo referentes à matriz de distância De-Para, cujo o formato dos locais é 4x6.....	111
 Figura A1.1 - Interface do CPN tools e a estrutura hierárquica de páginas.....	163
Figura A1.2 - Página hierárquica principal e suas ramificações	164
Figura A1.3 - Subpágina <i>ResourcesAllocation&OrdersInitialization&Processing</i> e suas ramificações	165
Figura A1.4 - Submodelo da subpágina <i>OrdersInput</i>	166
Figura A1.5 - Submodelo da subpágina <i>ProcessesDefinition</i>	168
Figura A1.6 - Subpágina <i>ToolMachSetup</i> e suas ramificações	170
Figura A1.7 - Submodelo da subpágina <i>ToolMachSetup</i>	171
Figura A1.8 - Submodelo da subpágina <i>CheckToolMach</i>	172
Figura A1.9 - Submodelo da subpágina <i>CheckMachToolList</i>	172
Figura A1.10 - Submodelo da subpágina <i>MachQueueSubQueue</i> e a função <i>TimeProdOPT</i>	174
Figura A1.11 - Subpágina <i>MachOutQueueSemiProdInAdd</i> e suas ramificações	174
Figura A1.12 - Submodelo da subpágina <i>UpdateLayoutPosAdd</i>	176
Figura A1.13 - Submodelo da subpágina <i>UpdateLayoutDist</i>	177
Figura A1.14 - Subpágina <i>Transportation</i> e suas ramificações	178
Figura A1.15 - Subpágina <i>TranspInitialMotionControl</i> e suas ramificações.....	179
Figura A1.16 - Subpágina <i>VirtCellDist</i> e suas ramificações	179
Figura A1.17 - Submodelo da subpágina <i>VirtCellDist</i>	180
Figura A1.18 - Submodelo da subpágina <i>CheckListLocal</i>	181
Figura A1.19 - Submodelo da subpágina <i>VirtualCellLaytPosSub</i>	182
Figura A1.20 - Submodelo da subpágina <i>VirtualCellLaytPosSub</i> (continuação)	182
Figura A1.21 - Subpágina <i>ToolMachSetupVirt</i> e suas ramificações.....	183
Figura A1.22 - Subpágina <i>TranspInitLayout</i> e suas ramificações	184
Figura A1.23 - Subpágina <i>TranspFromToInitVirtDist</i> e suas ramificações	184
Figura A1.24 - Submodelo da subpágina <i>TranspFromToInitVirtDist</i>	185

Figura A1.25 - Submodelo da subpágina <i>TranspFromToInitVirtDist</i> (continuação)	187
Figura A1.26 - Subpágina <i>TranspFromToInitDisp</i> e suas ramificações.....	188
Figura A1.27 - Submodelo da subpágina <i>TranspFromToInitDisp</i>	189
Figura A1.28 - Subpágina <i>TranspFromToDISTCrInit</i> e suas ramificações.....	192
Figura A1.29 - Subpágina <i>TranspFromToQueueCrInit</i> e suas ramificações	192
Figura A1.30 - Submodelo da subpágina <i>TranspFromToQueueCrInit</i>	193
Figura A1.31 - Subpágina <i>TranspInterMotionControl</i> e suas ramificações.....	195
Figura A1.32 - Subpágina <i>TranspInterFromToInterDisp</i> e suas ramificações	195
Figura A1.33 - Subpágina <i>TranspFromToInterDistCr</i> e suas ramificações.....	196
Figura A1.34 - Subpágina <i>TranspFromToQueueCrInter</i> e suas ramificações	197
Figura A1.35 - Subpágina <i>TranspExitMotionControl</i> e suas ramificações.....	198
Figura A1.36 - Subpágina <i>TranspFromToExitDistQueueCr</i> e suas ramificações	199
Figura A1.37 - Subpágina <i>TranspFromToExitQueueDistCr</i> e suas ramificações	199
Figura A1.38 - Subpágina <i>ModelLogicControl</i> e suas ramificações	200
Figura A1.39 - Subpágina <i>DistQueueCriterion</i> e suas ramificações.....	201
Figura A1.40 - Subpágina <i>DefinitionVirtualCellsLayout</i> e suas ramificações.....	202
Figura A1.41 - Submodelo da subpágina <i>DefInitMachOrdersVC</i>	203
Figura A1.42 - Submodelo da subpágina <i>DefinitionMachPosVC</i>	204
Figura A1.43 - Submodelo da subpágina <i>TranspFromToDistVC</i>	206
Figura A1.44 - Subpágina <i>TranspFromToDistVC</i> e sua ramificação	207
Figura A1.45 - Subpágina <i>DefineVirtCellPiece</i> e sua ramificação	208
Figura A1.46 - Submodelo da subpágina <i>DefineVirtCellPiece</i>	208
 Figura A3.1 - Tela de execução do código de programação para a geração dos perfis das peças	212
 Figura A4.1 - Separação da matéria-prima por carga da empilhadeira	217
Figura A4.2 - Informações impressas pelo código de programação para a separação da matéria-prima por carga da empilhadeira	220
 Figura A5.1 - Tela de execução do código de programação para a geração das possíveis posições do vão vazio	222
Figura A5.2 - Representação das vistas superiores dos arranjos físicos aleatoriamente distribuídos com as posições dos vãos vazios.....	223

Figura A5.2 - Representação das vistas superiores dos arranjos físicos aleatoriamente distribuídos com as posições dos vãos vazios – continuação.....	224
Figura A5.3 - Informações impressas ao arquivo externo das diversas posições do vão vazio para o modelo de simulação.....	225
Figura A5.4 - Representação da vista superior do melhor arranjo físico obtido pelo modelo de simulação	226
Figura A5.5 - Tela de execução do código de programação para a geração das possíveis posições do vão vazio no arranjo físico funcional	227
Figura A5.6 - Alocação invertida do arranjo físico funcional.....	227
Figura A5.7 - Informações sobre novos arranjos físicos funcionais impressas ao arquivo externo para serem lidas pelo modelo de simulação.....	228
 Figura A6.1 - Informações contidas no arquivo CrFunc.txt	229
Figura A6.2 - Informações contidas no arquivo CrMax.txt.....	231
 Figura A7.1 - Tempo de consolidação do pedido - arranjo físico funcional	235
 Figura A8.1 - Custo de consolidação do pedido - arranjo físico funcional	249
 Figura A9.1 - Tempo de consolidação do pedido – critério menor distância maior fila	250
Figura A9.2 - Tempo de consolidação do pedido - critério maior fila menor distância.....	251
 Figura A10.1 - Custo de consolidação do pedido - Critério menor distância maior fila.....	253
Figura A10.2 - Custo de consolidação do pedido - Critério maior fila menor distância.....	253
 Figura A11.1 - Tempo de consolidação do pedido - critério menor distância maior fila.....	254
Figura A11.2 - Tempo de consolidação do pedido - critério maior fila menor distância.....	254
 Figura A12.1 - Custo de consolidação do pedido - critério menor distância maior fila.....	255
Figura A12.2 - Custo de consolidação do pedido - critério maior fila menor distância.....	256
 Figura A13.1 - Tempo de consolidação do pedido – critério menor distância maior fila e maior fila menor distância.....	257

Figura A14.1 - Custo de consolidação do pedido – critério menor distância maior fila	258
Figura A14.2 - Custo de consolidação do pedido – critério maior fila menor distância	258
Figura A15.1 - Informações contidas no arquivo externo IN_CPN.sml.....	286
Figura A16.1 - Informações gravadas no arquivo externo referentes ao número de amostras do arranjo físico funcional.....	288
Figura A17.1 - Informações gravadas no arquivo externo referentes ao número de amostras do arranjo físico aleatoriamente distribuído	290
Figura A18.1 - Informações gravadas no arquivo externo referentes ao número de amostras do arranjo físico parcialmente distribuído	293
Figura A19.1 - Impressão do arranjo físico maximamente distribuído em arquivo externo.....	294
Figura A20.1 - Arquivo externo informando a posição alocada do transportador intermediário.....	295
Figura A21.1 - Ilustração do método roleta russa de seleção	297
Figura A21.2 – Exemplo de cruzamento entre os indivíduos pais para transmissão de genes aos indivíduos filhos	298
Figura A21.3 – Exemplo de mutação nos indivíduos filhos	299

LISTA DE TABELAS

Tabela 2.1 - Características de cada tipo de arranjo físico	22
Tabela 2.2 - Exemplo de depreciação linear.....	39
Tabela 2.3 - Exemplo de depreciação pela soma de dígitos.....	40
Tabela 2.4 - Exemplo de depreciação por horas trabalhadas	41
Tabela 2.5 - Resultados do ensaio de dureza para o exemplo considerado	45
Tabela 2.6 - Fatores e quantidade de níveis selecionados para os experimentos	46
Tabela 2.7 - Os três tipos adotados de variação de demanda anual de peças	47
Tabela 2.8 - Níveis de fatores.....	48
Tabela 2.9 - Fatores adotados para este estudo	49
Tabela 3.1 - Exemplo de representação das operações e ferramentais por tipo de peça	52
Tabela 3.2 - Possíveis combinações dos critérios adotados neste trabalho (quando as máquinas estão ocupadas) para o funcional	88
Tabela 3.3 - Possíveis combinações dos critérios adotados neste trabalho para o distribuído	88
Tabela 4.1 - Posições das réplicas das máquinas para a fabricação da peça tipo 78.....	128
Tabela 4.2 - Distâncias das oito possíveis rotas	129
Tabela 5.1 - Tempo de consolidação do pedido para 24 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote.....	132
Tabela 5.2 - Custo de consolidação do pedido para 24 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote.....	133
Tabela 5.3 - Tempo de consolidação do pedido para 24 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote	134
Tabela 5.4 - Custo de consolidação do pedido para 24 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote	135
Tabela 5.5 - Tempo de consolidação do pedido para 24 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote.....	137
Tabela 5.6 - Custo de consolidação do pedido para 24 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote	138
Tabela 5.7 - Tempo de consolidação do pedido para 24 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote	138
Tabela 5.8 - Custo de consolidação do pedido para 24 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote	138

Tabela 5.9 - Tempo de consolidação do pedido para 56 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote.....	140
Tabela 5.10 - Custo de consolidação do pedido para 56 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote.....	140
Tabela 5.11 - Tempo de consolidação do pedido para 56 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote.....	141
Tabela 5.12 - Custo de consolidação do pedido para 56 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote.....	141
Tabela 5.13 - Tempo de consolidação do pedido para 56 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote.....	142
Tabela 5.14 - Custo de consolidação do pedido para 56 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote.....	142
Tabela 5.15 - Tempo de consolidação do pedido para 56 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote	143
Tabela 5.16 - Custo de consolidação do pedido para 56 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote	143
Tabela 5.17 - Tempo de consolidação do pedido para 150 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote.....	144
Tabela 5.18 - Custo de consolidação do pedido para 150 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote.....	145
Tabela 5.19 - Tempo de consolidação do pedido para 150 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote.....	146
Tabela 5.20 - Custo de consolidação do pedido para 150 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote.....	146
Tabela 5.21 - Tempo de consolidação do pedido para 150 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote.....	147
Tabela 5.22 - Custo de consolidação do pedido para 150 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote.....	147
Tabela 5.23 - Tempo de consolidação do pedido para 150 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote.....	148
Tabela 5.24 - Custo de consolidação do pedido para 150 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote	148

Tabela A3.1 - Informações dos perfis das peças impressas pelo código de programação (primeiro arquivo)	213
Tabela A3.2 - Informações dos perfis das peças impressas pelo código de programação (segundo arquivo)	214
Tabela A21.1 - Exemplo de indivíduos de uma população com suas respectivas seqüências genéticas(string) e resistências (fitness)	296

LISTA DE ABREVIATURAS E SIGLAS

CPFL	Companhia Paulista de Força e Luz
CPN Tools	Colored Petri Net Tools
DEM	Distância da Entrada a uma Máquina qualquer
DL1E	Distância do Local 1 até a Entrada
DM's	Distância entre as Máquinas
DMS	Distância entre uma Máquina qualquer a Saída
DSL2	Distância da Saída até o Local 2
FCFS	First Come First Serve (Atendimento conforme a ordem de chegada)
HCPN	Hierarchical Colored Petri Nets (Redes de Petri coloridas hierárquicas)
L1	Local 1
L2	Local 2
LCFS	Last Come First Serve (Atendimento conforme o último que chegou)
LIC	Linha Imaginária Central
LPT	Longest Processing Time (Escolha do maior tempo de processamento)
LREM	Lowest Remaining Time (Menor tempo restante para finalizar o atendimento)
LTOP	Lowest Processing Total Time (Menor tempo total de processamento)
MD	Primeira Máquina Disponível
MDMF	Menor Distância e se houver empate, Maior Fila
MF	Menor Fila
MFMD	Maior Fila e se houver empate, Menor Distância
Minmax	Minimizar a Maior Distância entre os Diferentes Tipos de Máquinas
MP	Mais Próxima ou Menor Distância
MREM	Longest Remaining Time (Maior tempo restante para finalizar o atendimento)
MTOP	Longest Processing Total Time (Maior tempo total de processamento)
RP	Rede(s) de Petri
RPC	Rede(s) de Petri Colorida(s)
QAP	Quadratic Assignment Problem (Problema de designação quadrática)
SPT	Shortest Processing Time (Escolha do menor tempo de processamento)
WIP	Work In Process (Material em processo)

Sumário

1 INTRODUÇÃO	1
1.1 DISCUSSÕES GERAIS	1
1.2 OBJETIVOS.....	2
1.3 METODOLOGIA DE PESQUISA	2
1.3.1 Definição, classificações e tipos de dados	2
1.3.2 Metodologia adotada neste trabalho	3
1.4 ESTRUTURA DO TRABALHO.....	4
2 REVISÃO DA LITERATURA.....	5
2.1 SISTEMAS DE PRODUÇÃO	5
2.2 CLASSIFICAÇÃO DE MODELOS.....	7
2.3 SIMULAÇÃO COMPUTACIONAL.....	8
2.4 REDES DE PETRI COLORIDAS	13
2.5 DEFINIÇÕES E TIPOS DE ARRANJOS FÍSICOS.....	19
2.5.1 Arranjo físico por produto	20
2.5.2 Arranjo físico posicional	20
2.5.3 Arranjo físico funcional	21
2.5.4 Arranjo físico celular.....	21
2.5.5 Arranjo físico distribuído	22
2.5.6 Formulação matemática para obtenção do fluxo	27
2.5.7 Extensões do QAP.....	31
2.6 CUSTEIO DE PRODUÇÃO	35
2.7 CRITÉRIOS DE SELEÇÃO DE PEÇAS E MÁQUINAS.....	43
2.8 ORGANIZAÇÃO DOS DADOS PARA A REALIZAÇÃO DOS EXPERIMENTOS	44
3 PROCEDIMENTOS ADOTADOS PARA A DEFINIÇÃO DA DINÂMICA DA FÁBRICA.....	50
3.1 LOCAIS DE TRABALHO.....	51
3.2 GERAÇÃO DOS TIPOS DE PEÇAS.....	52
3.3 GERAÇÃO DE DEMANDA	54
3.4 DIMENSIONAMENTO DO NÚMERO DE RÉPLICAS DE CADA TIPO DE MÁQUINA.....	55
3.5 POSSÍVEIS FORMATOS DOS LOCAIS NO CHÃO DE FÁBRICA E GERAÇÃO DE UMA MATRIZ DE DISTÂNCIAS DE-PARA PARA CADA FORMATO.....	57
3.6 GERAÇÃO DOS ARRANJOS FÍSICOS.....	59
3.6.1 Arranjo físico funcional	59

3.6.2 Arranjo físico aleatoriamente distribuído	60
3.6.3 Arranjo físico parcialmente distribuído.....	61
3.6.4 Arranjo físico maximamente distribuído.....	62
3.7 VÃO VAZIO (DUMMY DEPARTMENT)	64
3.8 ALOCAÇÃO DE MÁQUINAS	69
3.9 RECURSOS ADOTADOS NA FABRICAÇÃO DE PEÇAS	70
3.9.1 Recursos usados na fabricação	71
3.9.2 Recursos de apoio à fabricação	71
3.10 CRITÉRIOS DE SELEÇÃO DE CAIXAS	77
3.11 CRITÉRIOS DE SELEÇÃO DE MÁQUINAS	78
3.11.1 Local 1 (L1) até as máquinas.....	79
3.11.2 Entre as máquinas.....	80
3.11.3 Chamar o transportador final do Local 2 (L2)	81
3.11.4 Máquinas até o Local 2 (L2).....	81
4 DADOS DE ENTRADA AO MODELO DE SIMULAÇÃO.....	89
4.1 CARACTERIZAÇÃO DA FÁBRICA FICTÍCIA NO INSTANTE 0	89
4.2 INSTABILIDADE DE MERCADO NO INSTANTE t	99
4.3 DESCRIÇÃO DA DINÂMICA DA SIMULAÇÃO DO MODELO	103
5 ANÁLISE DOS RESULTADOS DO MODELO DE SIMULAÇÃO.....	132
6 CONCLUSÕES E TRABALHOS FUTUROS.....	150
REFERÊNCIA BIBLIOGRÁFICA	153
BIBLIOGRAFIA CONSULTADA	161
APÊNDICE 1 DESCRIÇÃO DO MODELO DE SIMULAÇÃO DA FÁBRICA FICTÍCIA	163
A1.1 ESTRUTURA HIERÁRQUICA DO MODELO DE SIMULAÇÃO: A DEFINIÇÃO DA PÁGINA PRINCIPAL.....	164
A1.2 MODELAGEM DA ENTRADA, DA ALOCAÇÃO E DA DETERMINAÇÃO DOS PROCESSOS DE TRANSFORMAÇÃO DOS PEDIDOS DE PEÇAS.....	165
A1.3 MODELAGEM DO TRANSPORTE.....	177
A1.3.1 Modelagem do transporte inicial.....	178
A1.3.2 Modelagem do transporte intermediário.....	194
A1.3.3 Modelagem do transporte final	198
A1.4 MODELAGEM DA TOMADA DE DECISÃO GERAL	200
APÊNDICE 2 VERIFICAÇÃO E VALIDAÇÃO DO MODELO DE SIMULAÇÃO.....	210

APÊNDICE 3 GERAÇÃO DOS PERFIS DAS PEÇAS	212
APÊNDICE 4 SEPARAÇÃO DA MATÉRIA-PRIMA POR CARGA DA EMPILHADEIRA	215
APÊNDICE 5 APLICAÇÃO DO CÓDIGO DE PROGRAMAÇÃO PARA A OBTENÇÃO DAS POSIÇÕES DO VÃO VAZIO.....	221
APÊNDICE 6 OUTRAS INFORMAÇÕES RELEVANTES.....	229
APÊNDICE 7 TEMPO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO FUNCIONAL (20 AMOSTRAS).....	232
APÊNDICE 8 CUSTO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO FUNCIONAL (20 AMOSTRAS).....	236
APÊNDICE 9 TEMPO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO ALEATORIAMENTE DISTRIBUÍDO (20 AMOSTRAS)	250
APÊNDICE 10 CUSTO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO ALEATORIAMENTE DISTRIBUÍDO (20 AMOSTRAS)	252
APÊNDICE 11 TEMPO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO PARCIALMENTE DISTRIBUÍDO (20 AMOSTRAS)	254
APÊNDICE 12 CUSTO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO PARCIALMENTE DISTRIBUÍDO (20 AMOSTRAS)	255
APÊNDICE 13 TEMPO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO MAXIMAMENTE DISTRIBUÍDO	257
APÊNDICE 14 CUSTO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO MAXIMAMENTE DISTRIBUÍDO	258
APÊNDICE 15 INFORMAÇÕES CONTIDAS NO ARQUIVO IN_CPN.sml.....	259
APÊNDICE 16 IMPRESSÃO NO ARQUIVO EXTERNO DAS 20 AMOSTRAS DO ARRANJO FÍSICO FUNCIONAL (24 MÁQUINAS)	287
APÊNDICE 17 IMPRESSÃO NO ARQUIVO EXTERNO DAS 20 AMOSTRAS DO ARRANJO FÍSICO ALEATORIAMENTE DISTRIBUÍDO (24 MÁQUINAS)	289
APÊNDICE 18 IMPRESSÃO NO ARQUIVO EXTERNO DAS 20 AMOSTRAS DO ARRANJO FÍSICO PARCIALMENTE DISTRIBUÍDO (24 MÁQUINAS).....	291
APÊNDICE 19 IMPRESSÃO NO ARQUIVO EXTERNO DA AMOSTRA DO ARRANJO FÍSICO FUNCIONAL (24 MÁQUINAS)	294
APÊNDICE 20 ESPALHAMENTO DO TRANSPORTADOR INTERMEDIÁRIO	295
APÊNDICE 21 ALGORITMO GENÉTICO.....	296

1 INTRODUÇÃO

1.1 DISCUSSÕES GERAIS

Hoje já se pode afirmar, de fato, que as empresas de manufatura estão vivenciando um ambiente de extrema competitividade, ou seja, existe disputa acirrada pela clientela de modo que possam aumentar as suas vendas e assim aumentarem seus lucros. Em decorrência disso, os clientes têm muitas opções de compra e naturalmente tendem a rejeitar empresas que forneçam produtos com defeitos, com preços altos de venda, com atraso nas entregas etc. Enfim, toda esta adversidade de mercado, em termos técnicos, pode ser entendida como sendo alta variação na demanda; demanda total ou parcialmente estocástica; frequentes mudanças no conjunto de produtos; forte competição; alta variação no tamanho do lote de produção; alta variação nos tempos de processamento; alta variação nos tempos de preparação de ferramental e sequências de produção variáveis (RHEAULT et al., 1995).

O arranjo físico funcional, de certa forma, atua bem em um ambiente de instabilidade pois permite dar grande flexibilidade ao sistema de produção. Porém, segundo Montreuil et al. (1999), este arranjo físico apresenta algumas desvantagens, tais como elevado percurso entre departamentos, longas esperas em fila e longos tempos de troca de ferramental, o que naturalmente acabam refletindo em aumento no custo final das peças.

Por causa das ineficiências do arranjo físico funcional, Drolet (1989), Montreuil et al. (1991) e Benjaafar & Sheikhzadeh (2000) propõem um novo arranjo físico, denominado de arranjo físico distribuído, em que as máquinas de vários tipos encontram-se espalhadas no chão de fábrica justamente para aproximá-las entre si e, com isso, incorporar nele a forma de trabalho do tradicional arranjo físico celular.

Sendo considerado, portanto, como um novo arranjo físico, é necessário fazer estudos sobre o desempenho deste novo sistema produtivo. Uma das formas é por meio do uso das ferramentas de modelagem e simulação cuja finalidade é verificar se o desempenho do que está sendo proposto é melhor ao que é ainda usado na prática, ou seja, o arranjo físico funcional.

Verifica-se na literatura que os trabalhos de pesquisa abordam as células virtuais (tanto a formação quanto a maneira de se trabalhar com elas) diferentemente do conceito teórico para avaliação de desempenho de um arranjo físico distribuído. Isso porque existe muita dificuldade de identificar as máquinas e assim fazer a reserva delas para o encadeamento de processamento do lote de uma determinada peça.

Este trabalho, portanto, apresenta um estudo comparativo de desempenho entre o arranjos físicos funcional e distribuído usando as ferramentas de modelagem e simulação.

1.2 OBJETIVOS

Conforme dito anteriormente, o objetivo principal deste trabalho de pesquisa é, então, fazer um estudo comparativo entre o arranjo físico funcional e o arranjo físico distribuído usando as ferramentas de modelagem e simulação, cujos parâmetros medidores são tempo e custo para a consolidação dos pedidos.

Existem também vários objetivos secundários, conforme apresentados a seguir:

- a) Definir as formas de trabalho, em detalhes, de cada arranjo físico a ser comparado. Ou seja, como os lotes chegam; como as peças podem ser conduzidas até as máquinas responsáveis pelas operações seguintes; como os operadores estão posicionados; quais as características das máquinas usadas; quais os critérios de seleção de recursos a serem usados etc.;
- b) Construir o modelo de simulação em si. Tal modelo deve ser capaz de analisar a disponibilidade de máquinas; de requisitar transportadores; de tomar decisões caso houver disputa pelo uso de recursos; de fazer a identificação da máquina requisitada pela peça, de verificar a necessidade de realizar a troca de ferramental etc. Este modelo também deve ser capaz para avaliar o desempenho dos dois tipos de arranjos físicos que contenham qualquer quantidade de máquinas;
- c) Apresentar detalhadamente os procedimentos adotados que auxiliem na caracterização da fábrica fictícia adotada neste estudo. A partir destes procedimentos é que a lógica pode ser definida para iniciar a construção do modelo de simulação. Dentre os procedimentos a serem abordados, estão as possibilidades de formação de vão vazio; as diferentes formas de alocação de máquinas; as diversas formatações possíveis que os locais podem formar no chão de fábrica etc.
- d) Após a geração de dados do item c, definir os parâmetros de interesse que desejam ser testados no modelo de simulação, principalmente os que possam representar bem as variabilidades de mercado que se deseja testar.

1.3 METODOLOGIA DE PESQUISA

1.3.1 Definição, classificações e tipos de dados

Este tópico tem por objetivo apresentar algumas definições com relação à metodologia de pesquisa. Cervo & Bervian (1996) definem metodologia como sendo a ordem que deve ser imposta aos processos, os quais são necessários para atingir um fim dado ou um resultado desejado. Entende-se

por método o conjunto de processos que o espírito humano deve empregar na investigação e demonstração da verdade. Para Hegenberg (1976), o método (do grego *méthodos*, caminho para chegar a um fim) pode ser entendido como caminho para se chegar a um determinado resultado; é uma forma de selecionar técnicas; forma de avaliar alternativas para a ação (científica).

Ruiz (2006) classifica a pesquisa como sendo:

- Exploratória: consiste na caracterização inicial do problema, de sua classificação e de sua definição. Constitui, pois, o primeiro estágio de toda pesquisa científica. Este não tem por objetivo resolver, de imediato, um problema, mas tão somente caracterizá-lo;
- Teórica: tem por objetivo ampliar generalizações, definir leis mais amplas, estruturar sistemas e modelos teóricos;
- Aplicada: toma como certas leis ou teorias mais amplas como ponto de partida, e tem por objetivo investigar, comprovar ou rejeitar hipóteses sugeridas pelos modelos teóricos;
- Pesquisa de campo: consiste na observação dos fatos e registro de variáveis. Isso pode ser feito por meio de pesquisa de campo (entrevistas, questionários e formulários). Ao coletar dados, deve-se fazer análises;
- Pesquisa de laboratório: este permite que o pesquisador reitere, provoque e produza fenômenos em condições de controle. A principal característica deste item é que o pesquisador pode manipular e controlar as variáveis, uma a uma, tanto quanto possível com o objetivo de determinar qual ou quais delas são a causa. Neste ambiente de trabalho, pode-se fazer observações, experimentações etc.

Com relação à coleta de dados, Chizzotti (1991) afirma que existem basicamente dois tipos de dados: quantitativo (mensuração das variáveis pré-estabelecidas, procurando verificar e explicar sua influência sobre outras variáveis) e qualitativo (fundamenta-se em dados coligidos nas interações interpessoais, na co-participação das situações dos informantes, analisadas a partir da significação que estes dão aos seus atos).

1.3.2 Metodologia adotada neste trabalho

Conforme a apresentação da metodologia científica, este trabalho adota o tipo de pesquisa de natureza exploratória-bibliográfica, fazendo revisão da literatura referente ao arranjo físico distribuído, as técnicas comumente empregadas, as dificuldades encontradas e assim melhor caracterizar o problema a ser abordado neste trabalho. O método adotado é modelagem em computador, onde o sistema modelado representa uma fábrica fictícia. Os dados de entrada deste são gerados por códigos de

programação. Ao fazer a simulação deste modelo, é possível fazer experimentos conforme desejado, analisando os possíveis comportamentos do sistema produtivo modelado. Os dados coletados são de natureza quantitativa, onde os parâmetros a serem medidos são tempo e custo de consolidação dos pedidos.

1.4 ESTRUTURA DO TRABALHO

Ao todo, esta tese está dividida em seis tópicos, mais as referências bibliográficas e os apêndices. O Tópico 2 apresenta a revisão da literatura iniciando com a descrição dos itens relacionados às ferramentas usadas neste trabalho, que é modelagem e simulação. Portanto, a revisão é iniciada com a descrição geral sobre os sistemas de produção e depois, sobre a classificação de modelos etc. No Tópico 3 descrevem-se os procedimentos adotados para auxiliar na caracterização da dinâmica da fábrica fictícia. É com esta dinâmica que a lógica passa a estar definida para ser usada na construção do modelo de simulação. Depois, o Tópico 4 tem a finalidade de apresentar como é que se faz a geração e a definição dos dados para a simulação do modelo, incluindo os que representam o perfil instável de mercado. Este tópico também apresenta a explicação de como o modelo de simulação realiza a atualização das informações para a consolidação do pedido. O Tópico 5 apresenta a síntese dos resultados da simulação, que foram obtidos sujeitando-se o modelo para atuar sob diversas condições de instabilidade de mercado, bem como análise destes. Em seguida, as conclusões e propostas para futuros trabalhos podem ser vistas no Tópico 6. Por fim, estão apresentadas nos apêndices descrições mais aprofundadas sobre o modelo de simulação, sobre a interpretação dos resultados (obtidos diretamente deste modelo), sobre a geração dos arranjos físicos com problema do vão vazio e, principalmente, sobre as informações contidas no arquivos externos, lidas e atualizadas pelo modelo de simulação.

2 REVISÃO DA LITERATURA

2.1 SISTEMAS DE PRODUÇÃO

Este trabalho está focado no estudo comparativo de desempenho entre o arranjo físico funcional e o arranjo físico distribuído através do uso das ferramentas de modelagem e simulação de sistemas. Por isso, este tópico, inicialmente, tem por finalidade fazer revisão bibliográfica referente a essas ferramentas.

Um sistema de produção pode ser definido como sendo toda e qualquer empresa que fornece algum tipo de produto, ou serviço, ao cliente. De uma forma genérica, um sistema é composto por uma entrada, por uma saída e por um processo de transformação.

A entrada, também chamada de *input*, representa a entrada efetiva de recursos a serem transformados bem como de recursos transformadores. Já a saída, também chamada de *output*, representa o resultado da interação dos recursos. O processo de transformação define como os recursos de entrada são usados para se atingir à saída desejada. Todos esses componentes dos sistemas de produção influenciam, ou são influenciados por, um ambiente externo, onde tanto os concorrentes quanto os clientes estão localizados.

Uma representação genérica de um sistema de produção, então, pode ser vista na Figura 2.1. A parte superior desta figura ilustra o ambiente externo, onde o sistema de produção está hipoteticamente inserido. As setas localizadas sobre a fronteira representam as influências exercidas pelo sistema de produção e pelo ambiente externo. A fronteira desta figura representa a separação entre o sistema de produção e o ambiente externo.

O processo de transformação é comumente representado por um retângulo, e junto a este, existem duas setas. A primeira seta, localizada no lado esquerdo da Figura 2.1, representa a entrada dos recursos usados pelo processo de transformação, enquanto a segunda, localizada no lado direito, representa a saída de um produto ou serviço.

De acordo com Black (1998), um sistema de produção de sucesso é aquele que consegue responder adequadamente às influências do ambiente externo e que consegue também manter a capacidade de organizar eficientemente o processo de transformação ao longo do tempo. Porém, dependendo do sistema de produção em análise, a forma de interação entre os recursos dentro do processo de transformação pode se tornar bastante complexa, o que torna o processo de organização dos recursos extremamente difícil. Deve-se, portanto, desenvolver técnicas capazes de representar as interações dos recursos e assim definir como os recursos devem estar organizados para melhor responder às influências externas. Uma das técnicas é trabalhar com modelos.

Figura 2.1 - Representação genérica de um sistema de produção (Fonte: autor)

O objetivo principal de se trabalhar com modelos, é poder representar, de forma simplificada, um sistema real em estudo por meio de expressões matemáticas, por meio de figuras geométricas etc. Segundo Gordon (1969) e Carson (2003), devido à complexidade dos sistemas reais, a construção de um modelo para fins de estudo é uma alternativa para se compreender o comportamento de um sistema real.

Para que um modelo possa representar adequadamente um sistema real em estudo, alguns termos do mesmo, conforme apresentados a seguir, devem estar bem definidos (BANKS et al., 1984 e BANKS, 1998):

- Entidade: objeto de interesse do sistema. Este pode ser tanto dinâmico (por exemplo, peças que se movem pela fábrica, clientes saindo ou entrando em uma fila de supermercado etc.) quanto estático (máquina, caixa de um banco etc.);
- Atributo: propriedade de uma entidade, isto é, as características que definem uma entidade. Por exemplo, considerando uma peça como sendo uma entidade, um atributo seria o código da peça, o peso dela etc.;
- Atividade: tarefa sendo feita pela entidade fixa ou recurso num período de tempo;
- Recurso: é uma entidade fixa que executa uma determinada atividade para a entidade dinâmica;
- Estado: variáveis necessárias para descrever o sistema num instante qualquer;
- Evento: ocorrências que poderão alterar o estado do sistema;
- Endógeno: relativo a atividades e eventos que ocorrem dentro do sistema;
- Exógeno: relativo a atividades e eventos no meio que afetam o sistema.

Para entender melhor sobre a importância de definir tais termos durante a construção de um modelo, considere o exemplo apresentado na Figura 2.2. Esta figura mostra a vista superior de uma representação de um sistema de produção (manufatura) a ser modelado e dentro deste há várias máquinas (representadas em formato quadrado). Naturalmente, os números desta figura representam então os tipos de máquinas. Assim, o número 1 representa a máquina tipo 1, e assim por diante.

De acordo com o exemplo, uma entidade, no caso poderia ser uma determinada peça, terminou uma das operações na máquina tipo 1 e necessita ser movimentada à máquina para a operação seguinte, mais especificamente a máquina tipo 3. Se a definição do atributo desta entidade não estiver correta, em vez da peça ser encaminhada para a máquina 3, pode então ser encaminhada para a máquina tipo 4. Em consequência disso, a peça recebe operações diferentes daquela que foi designada em projeto.

Figura 2.2 - Exemplo de representação de um sistema de manufatura (Fonte: autor)

2.2 CLASSIFICAÇÃO DE MODELOS

As ferramentas a serem escolhidas para a realização de um trabalho dependem do grau de complexidade do sistema real e dependem também das técnicas empregadas no estudo. Por exemplo, se o objetivo é apresentar ao cliente um determinado imóvel, então uma alternativa seria usar a técnica da modelagem em programa de computador para ilustrar tal imóvel. Neste caso, o imóvel (sistema real) poderia ser representado em formato de modelo fictício (técnica), no qual foi construído usando um programa computacional (ferramenta). Se o objetivo é avaliar o dano estrutural causado pelo vento nas construções civis, então uma opção seria usar a técnica de construir um modelo tangível (maquete de papelão) usando como ferramenta técnicos, alunos etc. Note que neste segundo exemplo, existe um

parâmetro a mais para ser avaliado em comparação ao primeiro exemplo, que é o vento, o qual muda de intensidade ao longo do tempo. Portanto, dependendo do modelo e dependendo de quais parâmetros que se deseja analisar, pode afetar na escolha da ferramenta.

Sendo assim, é importante que se conheçam as classificações de modelos. De acordo com Banks et al. (1984), os modelos podem ser classificados em modelos matemáticos ou físicos. O modelo matemático utiliza notações e equações matemáticas para representar um sistema. Já o modelo físico (tangível) utiliza maquetes para permitir melhor visualização, ou utiliza protótipos para permitir testes ou ensaios antes de construir um sistema real (tal como avião, plataformas de petróleo, barragens etc.).

Os modelos de simulação, por fazerem parte do grupo de modelos matemáticos, podem ser classificados como: estáticos ou dinâmicos, determinísticos ou estocásticos e discretos ou contínuos.

- Modelo estático: representa o sistema num instante de tempo apenas. Não é levado em consideração a variável tempo no modelo;
- Modelo dinâmico: diferente do estático, este considera a variação do sistema com o tempo;
- Modelo determinístico: quando não existem variáveis aleatórias de entrada no modelo;
- Modelo estocástico: quando existe uma ou mais variáveis aleatórias de entrada no modelo;
- Modelo discreto: considera o comportamento das variáveis de estado variando conforme períodos de tempos separados adequadamente, diferentes do instantâneo;
- Modelo contínuo: considera o comportamento instantâneo da variável de estado em relação a cada fração de tempo.

2.3 SIMULAÇÃO COMPUTACIONAL

A simulação é também conhecida como uma técnica de avaliar o comportamento de um modelo, podendo considerar neste último variações dos parâmetros (alterações das entidades ao longo do tempo, por exemplo) na análise. Ou seja, as entidades apresentam variações de estado e essas variações exercem influências no comportamento do modelo.

A simulação computacional é, então, uma maneira de realizar a simulação usando uma ferramenta computacional (*hardware* e *software*), podendo inclusive até realizar a modelagem. Os modelos construídos no computador apresentam toda uma estrutura lógica, incluindo decisões para direcionar as entidades, atualizações das informações dos estados das entidades etc. e é justamente esta estrutura lógica do modelo que torna a modelagem computacional mais demorada. Porém, os resultados podem ser obtidos em curto espaço de tempo.

Algumas inferências com relação ao sistema real em estudo podem ser detectadas com o uso da simulação computacional, tais como (LOBÃO & PORTO, 1997):

- Comparação de desempenho com outros sistemas;
- Estudos sobre utilização de capacidade instalada, nível de inventário, lógica de controle, refinamento de projeto, integração, sequenciamento, gargalo de sistema, arranjo físico, índice de produtividade.

Observado o modo como o comportamento de cada parâmetro muda com o tempo, muitas “corridas” de simulação devem ser realizadas para a compreensão das relações envolvidas no sistema (YAMADA, 2004).

A simulação computacional hoje é, então, considerada como uma importante ferramenta de suporte às decisões testando e avaliando o desempenho do modelo representativo do sistema real. Com o uso dela é possível definir a melhor forma de organização dos recursos usados na produção (bens ou serviços). Da mesma forma, para Ulgen & Gunal (1998), é importante que as técnicas de simulação sejam utilizadas para se gerar informações antes que qualquer decisão seja tomada no sistema real.

Dentre os exemplos de trabalhos da literatura que realizam a modelagem de um sistema de manufatura, pode-se citar Maggio (2005). Este autor realizou a modelagem de um protótipo (de um sistema de manufatura) contendo seis máquinas (seis estações), três veículos auto-guiados, estações de carga e descarga e uma de manutenção de veículos.

Prata (2006) apresenta um estudo usando a modelagem e simulação para fins de avaliação de desempenho de um porto localizado no estado do Ceará.

Pode-se citar também o trabalho de Zagonel (2006). Este autor apresenta um estudo de desempenho, também via modelagem e simulação, de uma fábrica que contém apenas seis máquinas, sendo que todas as máquinas são fixas nos locais.

Shih (2005) realiza a modelagem usando Arena[®] 5.0, cujo o sistema modelado é um centro de distribuição de peças.

As técnicas de modelagem e simulação também são usadas na área agrícola. Carvalho (2006) apresenta um modelo de um sistema responsável pelo controle do comportamento de um robô móvel.

Infelizmente, quanto maior a complexidade do sistema analisado, maior o tempo demandado para a realização da modelagem por causa, principalmente, da estrutura lógica. Devido ao longo tempo consumido para realizar a modelagem, é fundamental que se construa um modelo capaz de ser reaproveitado para realizar simulações diversas. Por exemplo, em se tratando de um sistema de manufatura, o modelo de simulação deve ser capaz de avaliar o desempenho do sistema (sem alterações na estrutura lógica) acrescentando-se/removendo-se as máquinas no chão de fábrica, sendo que estas máquinas podem ocupar qualquer local do chão de fábrica. Tradicionalmente, um modelo no qual foi construído para simular uma determinada condição de operação de um sistema de manufatura

não pode ser reaproveitado caso sejam feitas estas alterações nos recursos produtivos, sendo necessário, portanto, realizar alterações na estrutura lógica.

Um dos simuladores computacionais, denominado *Colored Petri Net Tools* (ou CPN tools), permite fazer simulações sujeitando o modelo para atuar em várias condições de operação e para qualquer tamanho do sistema analisado, sem a necessidade de efetuar mudanças na estrutura lógica, sendo, portanto, o escolhido para ser usado neste trabalho. A versão escolhida para ser usada neste estudo é versão acadêmica 2.2.0, que pode ser obtida do CPN Tools (2009).

Para conduzir um bom estudo de simulação, Lobão & Porto (1997) apresentam uma proposta de sistematização de estudos de simulação, como se pode ver na Figura 2.3. Mas estes mesmos autores, enfatizam que a utilização completa ou parcial dos passos depende do grau de complexidade do projeto de estudo de simulação a ser desenvolvido.

Esses passos, para Yamada (2004), podem ser entendidos descrevendo o significado de cada um, conforme mostrados a seguir:

1) Definição do problema e dos objetivos do estudo

Engloba a compreensão do funcionamento do sistema, identificando-se o propósito do estudo de simulação, as entradas necessárias, as saídas desejadas do modelo e o comportamento desejado do sistema a ser modelado.

2) Elaboração de um esboço do modelo

Um esboço inicial do modelo do sistema deve ser elaborado com a realização dos primeiros estudos sobre o fluxo de informações, a disposição física dos componentes, a hierarquização dos módulos etc. Dependendo da abordagem escolhida para a construção do modelo, algumas ferramentas e metodologias podem ser aplicadas, tais como: diagrama de blocos, diagrama de objetos ou diagramas de fluxo de dados.

3) Aquisição de dados

Com o esboço do modelo, podem ser identificadas as entradas e saídas mais relevantes, e verificarem-se quais são controláveis e quais são incontroláveis, para propor soluções para eliminar ou ao menos atenuar os problemas. A aquisição de dados pode ser realizada de diversas formas: dados históricos, medições “in loco”, catálogos de fabricantes, entrevistas com operadores e projetistas do próprio sistema ou sistema similar. Nesta etapa é importante a determinação do tipo de distribuição da probabilidade mais semelhante a cada conjunto de dados, através de análise de resíduos da distribuição, construção de histogramas e comparação com curvas de distribuição padronizadas, testes de aderência etc.

Figura 2.3 - Proposta de sistematização dos estudos de simulação (Fonte: Lobão & Porto, 1997)

4) Validação dos dados

A verificação da validade dos dados coletados pode ser realizada através de:

- Coleta de dados de diferentes fontes para verificação de semelhanças entre os dados dessas fontes com o conjunto de dados do estudo;

- Construção de histograma para verificação de semelhança entre os dados coletados e os padronizados;
- Corrida de simulação para verificação de semelhança entre os resultados obtidos com os reais;

5) Construção de um modelo para realização do estudo de simulação

Nesta etapa, um fator importante para o sucesso da simulação é a escolha do *software* de simulação, cujas gerações mais recentes contam com módulos gráficos e inteligência artificial, permitindo a realização de estudos de simulação por usuários sem conhecimentos profundos nas diversas áreas envolvidas, tais como modelagem, simulação, processo a ser estudado, estatística e programação de computadores, entre outros.

Deve ser observado também o nível de detalhamento do modelo. Quanto maior o número de variáveis consideradas durante a modelagem, maior é a proximidade do modelo de representar o sistema real modelado. Entretanto, isso resulta em superdimensionamento do modelo, em aumento dos custos e em diminuição de sua confiabilidade. Por outro lado, quanto menor o número de variáveis consideradas durante a modelagem, menores serão as possibilidades de se obter um modelo consistente.

6) Validação do modelo

Consiste em verificar o quão fiel o modelo representa o sistema real. A validação dos modelos pode ter diversas abordagens:

- Interatividade com o cliente, no sentido de verificar a correção das respostas desejadas, a adequação na eliminação de variáveis e aproximações realizadas na construção do modelo;
- Identificação, através da animação do modelo, da ocorrência de impossibilidades físicas e/ou inconsistências;
- Realização de corrida de simulação por outro modelador ou operador familiarizado com o estudo para verificação de problemas e revisão do modelo;
- “Teste de Turing”, que é a análise, por especialistas, de relatórios com resultados reais do sistema e resultados obtidos pela simulação do modelo, visando identificar diferenças entre os dois grupos de dados.

7) Projeto de experimento

É a fase de proposição de procedimentos e testes para analisar e comparar as alternativas para a solução do problema. Para cada configuração experimental, é importante a realização de uma nova bateria de testes de validação para garantir-se a confiabilidade dos resultados. Dessa forma, é muito

importante a determinação do tamanho adequado da amostra de dados para produzir resultados dentro do intervalo de confiança desejado.

8) Execução do experimento e análise de resultados

Os dados utilizados e obtidos na execução do experimento devem ser organizados em forma de tabelas e gráficos e armazenados para posterior análise final.

9) Refino do projeto de experimento

Este passo deve ser realizado caso os resultados obtidos não sejam satisfatórios para a resolução do problema.

10) Análise final dos resultados e documentação do processo

A análise final dos resultados e a finalização do projeto devem ser organizados sob alguns aspectos:

- Produção de documentação final que permita o rápido e claro entendimento do sistema por outros usuários quando da necessidade de ampliação ou manutenção do modelo;
- Acesso restrito a documentação, de forma a evitar alteração involuntária da configuração do sistema, através de senhas, chaves de *hardware* e outros dispositivos.

2.4 REDES DE PETRI COLORIDAS

O *software* escolhido para este trabalho de pesquisa é o *software* CPN tools versão 2.2.0. De acordo com CPN Tools (2009), este é composto por dois componentes: *Graphical User Interface* (GUI) ou interface usuário gráfico e CPN ML (o CPN ML é uma linguagem de programação usando compilador SML/NJ).

Como o *software* CPN tools é fundamentado em Redes de Petri Coloridas, este tópico, então, tem por finalidade apresentar os componentes de uma Rede de Petri Colorida (RPC), importantes para a realização da modelagem do arranjo físico funcional e do arranjo físico distribuído proposta neste trabalho de pesquisa.

Os componentes de uma RPC são: elementos e formalismos.

- Elementos de uma Rede de Petri Colorida (ou RPC): uma das principais vantagens da RPC é poder realizar modelagens de sistemas de grande complexidade. Pela abordagem tradicional, ou seja, pela RP (Rede de Petri), a modelagem de sistemas complexos exigia grandes quantidades

de lugares e transições. Naturalmente, surgiu a necessidade de se criar um procedimento de redução do número de lugares e redução do número de transições, originando assim a RPC. Em função da redução do número de lugares e da redução do número de transições, o modelo também se torna mais enxuto na sua forma de apresentação, porém, em consequência da redução do número de lugares e do número de transições, mudam-se também os perfis das fichas (tradicionalmente conhecidas como *tokens*). Em vez de trabalhar com fichas, uma RPC trabalha com um conjunto de cores. Cor é apenas um nome dado pela literatura para separar grupos de informações.

Enfim, de acordo com Desrochers & Al-Jaar (1995), um modelo simples de RPC é constituído por grafos direcionados bipartidos representados pela quintupla $RPC=(P,T,C,I,O)$, em que:

- P é um conjunto finito de lugares. Cada lugar é definido por uma dimensão n, ou seja, se o lugar apresentar uma cor, então a dimensão n é 1; se o lugar apresentar duas cores, então a dimensão n é 2 etc.;
- T é um conjunto finito de transições. Da mesma forma se a transição considerar uma cor, então a dimensão m é apenas 1; se a transição considerar duas cores, então m é 2 etc.
- C são os conjuntos de cores:
 - Se o lugar apresentar uma cor, então a dimensão n é 1; se o lugar apresentar duas cores, então a dimensão n é 2 etc. De uma forma geral, portanto, as cores de um lugar p podem ser escritas da seguinte forma: $C(p_1), C(p_2), \dots, C(p_i)$, com $i=1, \dots, n$;
 Como cada cor $C(p_i)$ pode ter várias unidades k, então uma alternativa é representá-la por $aC(p_i)$, com a variando de 1 até k. Por exemplo, se existem duas unidades da cor $C(p_3)$, então tem-se $k=2$. Portanto, a representação para o conjunto de unidades da cor $C(p_3)$ seria: $aC(p_3)=\{1C(p_3), 2C(p_3)\}$ com $a=1,2$;
 - Se a transição lidar com uma cor, então a dimensão m é 1; se a transição lidar com duas cores, então a dimensão m é 2 etc. De uma forma geral, portanto, as cores consideradas em uma transição t podem ser escritas da seguinte forma: $C(t_1), C(t_2), \dots, C(t_j)$, com $j=1, \dots, m$;
- I=é arco de entrada relacionado ao conjunto cores direcionados de P para T;
- O=é arco de saída relacionado ao conjunto de cores direcionados de T para P;

- Formalismos de uma Rede de Petri Colorida: em virtude da mudança de se considerar cores em vez de fichas, mudam-se também a maneira de se trabalhar com elas etc. Deve-se, portanto, criar novos formalismos.

A Figura 2.4 mostra um exemplo em RPC visando ilustrar o formalismo citado. O exemplo já está apresentado no *software* que foi usado neste trabalho de pesquisa. Para este exemplo, um dos formalismos é:

```
colset culoare = with C1|C2;
var i: culoare;
```

O formalismo apresentado significa que o modelo trabalha com um grupo de cores ou conjunto de cores (colset), denominado culoare, e neste grupo existem dois tipos de cores, denominados C1 e C2. Para melhor compreender esta representação numa situação prática de modelagem, significa, por exemplo, que o modelo lida apenas com dois tipos de carros, os quais pertencem a apenas um grupo, denominado de carros.

```
colset carros=with vermelho|preto;
var i: carros;
```

Enfim, de acordo com o exemplo da Figura 2.4, os lugares podem conter duas possíveis cores C1 e C2, as quais fazem parte do conjunto de cores (colset) chamado de culoare. Diferente das RP de baixo nível, as fichas no mesmo lugar não tinham diferenciação uma das outras. Agora, um mesmo lugar pode ter cores diferentes. Após a habilitação da transição, a variável i é que define quantas unidades de cada cor devem existir num determinado lugar.

Ainda de acordo com a Figura 2.4, o lugar A, representado pelo círculo localizado na parte superior da Figura 2.4, contém duas cores. A primeira cor deste lugar A é 1`C1++. O número 1` representa uma unidade da cor C1, e de forma similar 1`C2 representa uma unidade da cor C2. O número 2 representado no lugar A, além de indicar o resultado da soma de uma unidade da cor C1 mais uma unidade da cor C2, ele indica também fichas (similares ao RP), ou seja, existem duas fichas. A diferença é que na RPC, existe distinção entre as fichas. A representação ++ significa que o lugar, neste instante, contém mais de um tipo de cor.

Com relação aos lugares, pode-se notar na Figura 2.4 que em todos eles possuem a representação culoare, o que significa que estes podem conter tanto a cor C1 quanto a cor C2.

Em seguida, pode-se notar uma seta (arco) “saindo” do lugar A. O arco, denominado de arco de entrada a transição T1, é representada por uma variável (denominada i). Isso significa que a variável i pode conter as informações de uma das cores do conjunto culoare. O arco tem a

característica de dar condições para uma determinada transição realizar o disparo, bem como “remover” as unidades das cores de cada lugar. Também é possível que o arco defina novas unidades de cores C1 ou C2 para os lugares. Enfim, mais detalhes podem ser compreendidas ao longo do texto.

Figura 2.4 - Exemplo de Redes de Petri Coloridas (Fonte: Blaga et al., 2008)

Saindo deste lugar A, pode-se notar que a transição T1 possui dois arcos de entrada, oriundos do lugar A e do lugar E. A condição para que T1 dispare é que no lugar A e no lugar E tenham, ao mesmo tempo, alguma unidade do mesmo tipo de cor. Também não ocorre o disparo se cada lugar contiver uma unidade de cores diferentes.

De acordo com o exemplo, a transição T1 está habilitada para realizar o disparo porque existe uma unidade da cor C1 no lugar A e uma unidade da cor C1 no lugar E. Também ocorre disparo da transição T1 caso contivesse no lugar A $3 \cdot C1 + 1 \cdot C2$ e $2 \cdot C1$ no lugar E, porque exige-se apenas uma unidade de cada tipo de cor porque, neste caso, a variável i representa uma unidade da cor.

Um lugar também pode, inicialmente, ter as unidades de cada uma das cores C1 ou C2 definidas. Por exemplo, no caso do lugar E, inicialmente, contém uma unidade C1, daí representação $1 \cdot C1$; inicialmente o lugar F contém apenas uma unidade da cor C1, daí representação 1 ou $1 \cdot C1$; ou $1 \cdot C1 + 1 \cdot C2$ no lugar A para representar que inicialmente o lugar A contém uma unidade de cor C1 e uma unidade de cor C2.

Da mesma forma, um lugar pode conter, também, unidades de cada cor ainda a serem definidas por um arco de saída. Esta definição é feita após o disparo. Então, como a transição T1 está habilitada, é definido para o lugar B apenas uma unidade de $1 \cdot C1$ porque i é unitário. Caso seja necessário definir para o lugar B várias unidades da cor C1, uma alternativa seria trabalhar com *if-then-else*. Considere o arco que conecta T2 ao lugar E. Pode-se notar que neste arco existe uma representação *if $i=C1$ then $1 \cdot C2$ else $1 \cdot C1$* . Esta representação apresenta algumas informações extremamente importantes:

- Se no lugar B estiver contido uma unidade C1, então a transição T2 dispara e a representação define que sendo i igual a 1, então uma unidade C2 é definida no lugar E;
- Se no lugar B estiver contido uma unidade C2, então T2 também dispara, porém a unidade definida no lugar E é da cor C1;
- Se deseja-se definir novas unidades da cor ao lugar E, uma alternativa de representação para o arco seria *if $i=C1$ then $3 \cdot C2$ else $2 \cdot C1$* . Isso quer dizer que, assim que o lugar B contiver uma unidade da cor C1, é feito o disparo da transição T2, e o arco define 3 unidades da cor C2 para o lugar E.

A linguagem *if-then-else* usada é parte da linguagem de programação usada pelo CPN ML, cujo compilador SML/NJ já está embutido no *software* CPN tools.

Enfim, a interpretação para as demais partes do exemplo da Figura 2.4 segue o mesmo raciocínio.

O objetivo da Figura 2.5 é mostrar mais um exemplo em que outros formalismos são usados com frequência neste trabalho de pesquisa. Dentre eles, estão a guarda, a função, a outra estrutura do tipo *if-then-else* do CPN ML e os conjuntos de cores INT e STRING.

Figura 2.5 - Exemplo de uma Rede de Petri colorida (Fonte: autor)

Pode-se notar nessa Figura 2.5 o estado inicial (ou seja, definições das unidades de cada cor nos lugares) de um exemplo de modelo em RPC, sendo que no lugar P1 estão contidos três fichas. As unidades e os tipos de fichas são definidos diretamente neste lugar P1 (então, de acordo com o exemplo, tem-se uma unidade da cor "5" e duas unidades da cor "10", totalizando um total de três fichas no lugar P1).

Em uma RPC, como visto, também é permitido trabalhar com cores que sejam de outros formatos, tipo letras combinadas com números, tais como C1 e C2. A definição do grupo de cores vai depender do que cada um vai representar durante a modelagem e escolher a que for mais conveniente. O formalismo usado para definir o conjunto de cores do tipo inteiro é do tipo INT. Pode-se notar, então, que no lugar P1 existe uma representação INT, o que significa que o conjunto de cores contidos no lugar P1 é do tipo inteiro.

Ainda com relação à Figura 2.5, existe um arco de entrada, denominado X, e é uma variável que contém as informações do conjunto de cores do tipo inteiro. A sua representação é dada por Val X: INT. Esta variável contém as informações dos tipos de cores e não as unidades dos tipos de cores.

Pode-se notar que a transição T1 está definida com uma guarda do tipo $[X > 0]$. A guarda de uma transição, representada por $[X > 0]$, é uma restrição a mais dada à transição para que o disparo aconteça. Para este exemplo, a transição T1 está habilitada porque a variável X requisita apenas uma unidade de qualquer uma das duas cores. Como a guarda da transição T1 está definida, quer dizer que ela não só depende das condições do lugar P1 e do arco X, mas também da restrição imposta pela guarda. Como ambas as cores 5 e 10 são positivas, então satisfazem a restrição imposta pela guarda e dessa forma o disparo da transição T1 ocorre.

Ao realizar o disparo, a função do arco (denominada FUN(X)) é ativada e este coleta a informação da variável X e a transforma em um novo conjunto de cores denominado STRING do tipo string (variáveis textuais). As cores que fazem parte do conjunto de cores STRING são YES e NO. O exemplo ilustrado na Figura 2.5, portanto, é encerrado gerando-se duas novas cores no lugar P2, denominadas de YES e NO, cujas unidades geradas de cada cor são $1 \cdot \text{YES} + 2 \cdot \text{NO}$.

Enfim, existem também outros formalismos, porém menos usados neste trabalho de pesquisa, que podem ser vistos em Jensen (1991).

Para o desenvolvimento deste trabalho de pesquisa, também foram utilizadas as RPC hierárquicas (HCPN – Hierarchical Colored Petri Nets). A diferença destas para com as redes de Petri coloridas convencionais, ou não-hierárquicas, é a possibilidade de subdividir modelos complexos em submodelos mais simples e de integrá-los através de técnicas de modularização: transições de substituição (substitution transitions) e lugares de fusão (fusion places).

Para uma HCPN, define-se uma página principal onde se encontra o modelo principal. A partir deste modelo principal, podem-se associar outras páginas ou subpáginas dentro das quais submodelos são implementados. Tais submodelos são definidos a partir de transições de substituição. Portanto, uma transição de substituição é uma transição a partir da qual uma página é definida e onde um submodelo pode ser construído. O disparo das transições de substituição decorre dos processos que ocorrem nos submodelos definidos.

Os lugares de fusão constituem um recurso muito importante de integração entre submodelos que não estejam necessariamente e diretamente associados hierarquicamente.

A teoria das HCPN reserva uma página diversa e sem qualquer relação funcional com relação à modelagem, onde o modelo final com todas as suas ramificações modulares pode ser analisado hierarquicamente: a página hierárquica (hierarchy page).

2.5 DEFINIÇÕES E TIPOS DE ARRANJOS FÍSICOS

Após a descrição das ferramentas de modelagem e simulação, a próxima etapa da revisão tem por finalidade apresentar o tipo de sistema de produção onde essas ferramentas serão usadas neste estudo.

Um arranjo físico é definido como sendo a disposição física de vários elementos ou recursos produtivos no chão de fábrica, usados na produção de um bem ou serviço, tais como máquinas, equipamentos, instalações e pessoal. Portanto, quando se estuda (propõe) um arranjo físico, é necessário analisar qual o melhor relacionamento entre as atividades (tais como materiais, informações) para que as unidades produtivas possam ser instaladas adequadamente.

Para se ter ideia sobre a importância desta questão de posicionamento de recursos, Sule (1994) e Islier (1998), por exemplo, afirmam que aproximadamente 30-75% do custo de um produto pode ser

atribuído à manipulação de material e transporte. Então, uma opção de estudo para minimizar este custo seria analisar como se deve disponibilizar máquinas, equipamentos e serviços de suporte em uma determinada área.

Dependendo da forma de como os recursos produtivos estão posicionados em um chão de fábrica, os arranjos físicos então recebem diferentes nomes: arranjo físico por produto, arranjo físico posicional, arranjo físico funcional e arranjo físico celular. Estes arranjos físicos também são conhecidos como arranjos físicos tradicionais.

Posteriormente, com o perfil de demanda cada vez mais instável, foram surgindo novos tipos de arranjos físicos. Dentre eles, pode-se citar o arranjo físico distribuído.

2.5.1 Arranjo físico por produto

Este arranjo físico é caracterizado pelo posicionamento de equipamentos de acordo com a sequência de produção (OLIVÉRIO, 1985). É o melhor arranjo físico quando se deseja atingir grande volume de produção e pequena variedade, conseguindo assim economia de escala. Uma grande desvantagem é a possibilidade de repetição de equipamentos (máquinas idênticas posicionadas em pontos diferentes dentro deste arranjo físico). Isso acontece porque pode existir repetição de operação em uma etapa posterior descrita pela folha de processo e, para evitar o retorno de peça (fluxo reverso), duplica-se, ou triplica-se etc. a mesma máquina. Além disso, o tempo de troca de ferramental é longo, o que justifica o fato de ter que produzir grandes volumes para compensar os custos referentes à essa troca.

2.5.2 Arranjo físico posicional

O arranjo físico recebe este nome quando existem diversos tipos de equipamentos (recursos transformadores) localizados um próximo ao outro para serem usados na produção de bens ou serviços. No caso de um sistema de manufatura, a peça a ser processada não necessita ser movimentada. Em vez de materiais, informações ou clientes fluírem através de uma operação, quem sofre o processamento fica estacionado, enquanto equipamento, maquinário, instalações e pessoas é que se movem ou são movidos.

Segundo Slack et al. (1999), a razão para implementar esse tipo de arranjo físico, geralmente, é que o produto costuma ser de grande valor ou pela dificuldade em movê-lo de forma conveniente.

2.5.3 Arranjo físico funcional

Segundo Costa (2004), as máquinas são agrupadas de acordo com o tipo geral de processo de manufatura: tornos em um departamento, furadeiras em outro e assim por diante. A vantagem deste arranjo físico é a capacidade de produzir uma grande variedade de produtos. Tompkins et al. (1996) ressaltam que neste tipo arranjo físico, o fluxo de peças é pequeno dentro de cada departamento (também chamado de intradepartamental) e grande entre eles (também chamado de interdepartamental). Além disso, existem ainda outras desvantagens, tais como longos tempos de trocas de ferramental e longas esperas em fila.

2.5.4 Arranjo físico celular

No arranjo físico celular, grupos de diferentes tipos de máquinas são formados e alocados em determinadas seções no chão de fábrica. As peças que têm alguma semelhança em seu processamento são também agrupadas e então processadas em um dos grupos de máquinas. A separação de cada célula é visível fisicamente. Segundo Irani (1999), manufatura celular é uma estratégia que divide o sistema de produção em vários grupos (ou células) de modo que cada um deles seja capaz de produzir uma família de peças.

De uma forma ainda mais genérica, Black (1998) descreve algumas características chaves:

- As máquinas são dispostas na sequência do processo;
- Normalmente a célula é projetada em forma de U;
- Uma peça (ou lotes pequenos) é que flui na célula;
- Os trabalhadores são treinados para lidar com mais de um tipo de processo;
- Os operadores trabalham em pé e caminhando;
- São usadas máquinas mais lentas e específicas, que são menores e mais baratas.

Como resultado da implementação de células de manufatura, Gorgulho Júnior (2006) cita:

- Alta utilização de máquinas;
- Atmosfera de equipe;
- Boa flexibilidade e produtividade;
- Equipamentos de uso geral com eventuais máquinas específicas;
- Operadores executam mais de uma atividade ao mesmo tempo;
- Redução do material em processo;

- Redução do manuseio de material;
- Redução dos tempos de preparação de ferramental;
- O movimento de peças intercelular também é bastante reduzido.

Em resumo, Morábito (1998) apresenta outras características referentes aos três tipos de arranjos físicos, conforme mostradas na Tabela 2.1:

Tabela 2.1 - Características de cada tipo de arranjo físico (Fonte: Morábito, 1998)

Característica	Processo	Produto	Celular
Leadtime de produto	alto	baixo	Baixo
WIP	alto	baixo	Baixo
Flexibilidade de produto	alta	baixa	Média
Flexibilidade de demanda	alta	média	Média
Utilização de máquinas	baixa	alta	Média
Custo unitário de produção	alto	baixo	médio

2.5.5 Arranjo físico distribuído

O mercado está se tornando mais competitivo. Esta competitividade, de acordo com Rheault et al. (1995), pode ser representada por alta variação na demanda; demanda total ou parcialmente estocástica; frequentes mudanças no conjunto de produtos; alta variação no tamanho do lote de produção; alta variação nos tempos de processamento; alta variação nos tempos de preparação de ferramental e sequências de produção variáveis.

Segundo Montreuil et al. (1999), quando o tradicional arranjo físico funcional é implementado em um ambiente de instabilidade, este permite dar grande flexibilidade a produção, porém sofre com algumas desvantagens, tais como longos deslocamentos, longas esperas em fila e longos tempos de trocas de ferramental.

Enfim, os arranjos físicos tradicionais já não atendem aos novos requisitos de mercado e por isso novos tipos de arranjos físicos têm surgido na literatura, tais como arranjo físico modular, arranjo físico

fractal e arranjo físico distribuído. Em particular, este trabalho vai focar apenas neste último, ou seja, no arranjo físico distribuído.

Montreuil et al. (1991), Drolet (1989) e Benjaafar & Sheikhzadeh (2000), propõem espalhar as máquinas no chão de fábrica para satisfazer as condições não alcançadas pelo arranjo físico funcional. Ao realizar o espalhamento das máquinas, o arranjo físico é então denominado de arranjo físico distribuído.

As Figuras 2.6, 2.7 e 2.8 mostram três exemplos de arranjos físicos: funcional, celular e distribuído, respectivamente. Note que no arranjo físico funcional, as máquinas ficam concentradas em certas áreas no chão de fábrica. Já no arranjo físico celular, as máquinas são agrupadas de acordo com as operações das peças. E por fim, no arranjo físico distribuído, as máquinas são espalhadas no chão de fábrica.

Figura 2.6 - Exemplo de um arranjo físico funcional (Fonte: Benjaafar & Sheikhzadeh, 1996)

Figura 2.7 - Exemplo de um arranjo físico celular (Fonte: Benjaafar & Sheikhzadeh, 1996)

Figura 2.8 - Exemplo de um arranjo físico distribuído (Fonte: Benjaafar & Sheikhzadeh, 1996)

Existem também vários tipos de arranjos físicos distribuídos. Dependendo da maneira de como é feito o espalhamento de máquinas no chão de fábrica, um arranjo físico distribuído pode, então, receber nomes diferentes, conforme apresentados a seguir:

- Arranjo físico aleatoriamente distribuído: as máquinas são selecionadas aleatoriamente e, então, espalhadas no chão de fábrica;
- Arranjo físico parcialmente distribuído: dois ou três grupos do mesmo tipo de máquina são formados e alocados no chão de fábrica;
- Arranjo físico maximamente distribuído: as máquinas de mesmo tipo são alocadas nas posições mais distantes umas das outras. A equação Minmax (minimizar a maior distância entre os diferentes tipos de máquinas), de Francis & White (1974), ajuda a obter um bom espalhamento de máquinas no chão de fábrica.

É importante também apresentar, com detalhes, como um arranjo físico distribuído trabalha, o que naturalmente influencia na forma de definir a estrutura lógica do modelo de simulação em computador. Considere o exemplo mostrado na Figura 2.9. Esta figura apresenta a vista superior de uma fábrica fictícia. Os números representam os tipos de máquinas e, portanto, existem três réplicas da máquina tipo 1, quatro máquinas tipo 2, e assim por diante. Note que os diferentes tipos de máquinas estão espalhadas no chão de fábrica.

A ideia no arranjo físico distribuído é poder encadear as operações para processar uma determinada peça, ou seja, identificar as máquinas que farão parte do processamento do lote da peça e não permitir mais que nenhum outro tipo de peça utilize essas máquinas até que o lote termine de receber todo o processamento. Esta separação do conjunto de máquinas que fazem parte do encadeamento da peça é denominada de célula virtual.

1	5	2	4
5	5	1	2
2	3	5	5
5	2	4	1

Figura 2.9 - Vista superior de um arranjo físico distribuído (Fonte: autor)

Considere um exemplo em que o arranjo físico distribuído, representado pela Figura 2.9, deva produzir uma determinada peça, denominada X, cujas operações sejam $1 \rightarrow 4 \rightarrow 5$. O passo seguinte é formar células virtuais que sejam constituídas por essas máquinas (ou seja, máquinas 1, 4 e 5). Note que existem várias réplicas de cada tipo de máquina requisitada pela peça X e, portanto, também muitas possibilidades para se formar células virtuais, como ilustradas na Figura 2.10.

1	5	2	4
5	5	1	2
2	3	5	5
5	2	4	1

1	5	2	4
5	5	1	2
2	3	5	5
5	2	4	1

1	5	2	4
5	5	1	2
2	3	5	5
5	2	4	1

1	5	2	4
5	5	1	2
2	3	5	5
5	2	4	1

Figura 2.10 - Alguns exemplos de possíveis células virtuais a serem formadas para a produção da peça requisitada (Fonte: autor)

Como existem várias possibilidades para se formar células virtuais, recomenda-se escolher sempre aquela onde as réplicas estejam posicionadas de forma que a soma das distâncias das máquinas seja a menor possível. Portanto, de acordo com o exemplo, considerando que todas as máquinas estejam livres, a possibilidade mostrada no canto superior esquerdo é a melhor porque a distância é de apenas duas unidades de distância. Se alguma máquina da célula virtual estiver ocupada, simplesmente o lote nem chega no chão de fábrica.

Assim que o lote de peças X chega na célula virtual escolhida, todo lote é processado nela até que todas as operações sejam realizadas. Durante o processamento deste lote, nenhuma máquina deve ser usada para processar outro tipo de peça.

Mas o mais importante em trabalhar com células virtuais é poder “quebrar” lotes grandes em lotes pequenos (se possível unitário). A Figura 2.11 mostra duas alternativas possíveis a serem adotadas numa fábrica para a produção de um determinado tipo de peça, cujo tamanho do lote é de 4 unidades. Além disso, essa peça deve ser submetida a três operações. O exemplo, ilustrado na parte superior dessa Figura 2.11, mostra a produção em lotes, típica em um tradicional arranjo físico funcional. Já o exemplo, ilustrado na parte inferior da Figura 2.11, mostra a “quebra” do lote de 4 unidades em lotes menores. No caso do exemplo, o lote menor representado é de apenas 1 unidade. Note que o tempo total de produção do lote de peça reduz bastante no segundo caso. Esta segunda forma de trabalho é justamente como arranjos físicos celulares (os tradicionais) trabalham. Portanto, a ideia de formar células virtuais (em arranjos físicos distribuídos) é poder formar vários arranjos físicos celulares e incorporar a forma de trabalho destes.

Figura 2.11 - Exemplo de quebra de lote grande em lote pequeno (Fonte: autor)

2.5.6 Formulação matemática para obtenção do fluxo

Após a descrição de como cada tipo de arranjo físico em análise trabalha (arranjo físico funcional e arranjo físico distribuído), deve-se então obter a distância percorrida pela peça para que o modelo de simulação possa realizar a contabilização do tempo de deslocamento das peças para fins de avaliação de desempenho entre os dois tipos de arranjos físicos. Naturalmente, em função da obtenção da distância percorrida, é possível obter o tempo de deslocamento.

Enfim, a formulação matemática comumente usada para a obtenção da distância percorrida pela peça é QAP (Quadratic Assignment Problem). Esta formulação serve tanto para um arranjo físico distribuído quanto para um arranjo físico funcional. QAP considera as máquinas como sendo de mesmo tamanho e quadrada/retangular. O chão de fábrica é dividido em linhas e colunas, ou seja, máquinas posicionadas na horizontal e na vertical, respectivamente. Após conhecer a posição de cada tipo de máquina, o lote de fabricação bem como as operações de cada peça, é possível obter a distância percorrida pelas peças entre as máquinas. Pela lógica, quanto menor distância percorrida, menor é o custo associado à movimentação de materiais (KUSIAK & HERAGU, 1987 e MONTREUIL et al., 1993).

A função-objetivo que minimiza o fluxo entre departamentos é dada por Benjaafar (1994):

$$MinZ = \sum_{i=1}^N \sum_{j=1}^N \sum_{n_i=1}^{N_j} \sum_{n_j=1}^{N_j} \sum_{k=1}^K \sum_{l=1}^K V_{n_i.m_j} \cdot d_{kl} \cdot X_{n_jk} \cdot X_{m_jl}$$

Sujeito a:

$$\sum_{k=1}^K X_{ik} = 1 \quad \text{para todo } n_i \text{ e } i;$$

$$\sum_{i=1}^N \sum_{n_i=1}^{N_i} X_{n_i,k} = 1 \quad \text{para todo } k;$$

$$\sum_{i=1}^N \sum_{n_i=1}^{N_i} V_{n_i.m_j} \cdot T_{m_j} \leq C_{m_j} \quad \text{para todo } m_j \text{ e } j;$$

$$\sum_{j=1}^N \sum_{m_j=1}^{N_j} V_{n_i.m_j} \cdot T_{n_i} \leq C_{n_i} \quad \text{para todo } n_i \text{ e } i;$$

$$\sum_{n_i=1}^{N_i} \sum_{m_j=1}^{N_j} V_{n_i.m_j} = V_{ij} \quad \text{para todo } i \text{ e } j;$$

$$X_{n_jk} = 0,1 \quad \text{para todo } n_i \text{ e } k.$$

Onde:

$$X_{n_i,k} = \begin{cases} 1 & \text{se o } n\text{-ésimo departamento do tipo } i \text{ for atribuído à localização } k \\ 0 & \text{caso contrário} \end{cases}$$

V_{n_i,m_j} : volume de fluxo entre o n-ésimo departamento do tipo i e m-enésimo departamento do tipo j;

d_{kl} : distância entre a localização k e a localização l;

V_{ij} : volume de fluxo entre os departamentos do tipo i e tipo j;

T_{n_i} : tempo de processamento médio equivalente por unidade de carga transferida no departamento ni;

C_{n_i} : capacidade do departamento ni (tempo disponível de operação);

n_i : número de máquinas do tipo i;

N: número total de tipos de máquinas;

K: número total de localizações.

Esta formulação matemática pode ser melhor entendida criando-se dois exemplos de arranjos físicos, com a representação apenas dos locais (onde serão alocadas as máquinas), conforme mostrados na Figura 2.12. Ambos são representados por variáveis que fazem parte do cálculo de distância percorrida pela peça.

Figura 2.12 - Vistas superiores: dois exemplos (Fonte: autor)

De acordo com a Figura 2.12, as variáveis que compõem o chão de fábrica são:

DL1E: Distância do local 1 até a entrada;
 DEM: Distância da entrada a uma máquina qualquer;
 DM's: Distância entre as máquinas;
 DMS: Distância entre uma Máquina qualquer a saída;
 DSL2: Distância da saída ao local 2.

As coordenadas x e y são usadas para definir as coordenadas de cada local (onde as máquinas serão alocadas). Por convenção, a coordenada x é usada para o cálculo da distância a ser deslocada pela peça na vertical e a coordenada y é usada para o cálculo da distância a ser deslocada pela peça na horizontal. Pode-se notar que existe, em cada Figura, uma linha imaginária, denominada LIC. Esta linha será explicada mais adiante.

Note ainda, na Figura 2.12, que estão ilustradas duas rotas “A” e “B”. Estas duas mostram percursos diferentes para se atingir o mesmo destino. A rota “B” mostra um exemplo de deslocamento em uma situação real. O círculo representa o local de origem ou de destino da peça. Enquanto a rota “A”, é a forma adotada para a obtenção da distância percorrida pela peça neste trabalho de pesquisa. Note que os deslocamentos obtidos, tanto em “A” quanto em “B”, são iguais.

A distância percorrida pela peça entre máquinas, ou distância entre máquinas, ou simplesmente DM's, é também conhecida como a distância Hamming, conforme mostrada na expressão (1):

$$\text{Distância entre máquinas} = |x_{\text{máq. origem}} - x_{\text{máq. destino}}| + |y_{\text{máq. origem}} - y_{\text{máq. destino}}| \quad (1)$$

Assim, o cálculo da distância percorrida pela peça entre máquinas é, na realidade, apenas a soma dos DM's entre duas máquinas requisitadas.

O fato é que o esquema apresentado na Figura 2.12 pode ser usado tanto para o arranjo físico funcional quanto para o arranjo físico distribuído, porém com algumas particularidades.

1) Lote unitário:

- Arranjo físico funcional e arranjo físico distribuído: a distância percorrida é o próprio DM's.

2) Lote diferente de unitário:

- Arranjo físico funcional: considerando que a empresa utilize carrinhos manuais para o transporte de caixas (onde contém peças semi-acabadas), o fluxo passa a depender da capacidade destes.

Por exemplo, considerando que não ocorra mistura de tipos de peças nem na caixa, nem no carrinho, o cálculo de “n_vezes” da expressão (3), significa o número de vezes em que ocorre o deslocamento entre máquinas. Trunc significa truncar,

sendo que este tem a finalidade de arredondar para cima. Para usar a expressão (3) é preciso ter calculado a expressão (2).

A primeira incógnita da expressão (2), denominada peso peça[kg], representa o peso unitário de um determinado tipo de peça, em kg. A segunda incógnita da expressão (2), denominada tamanho lote, representa a quantidade de peças a serem produzidas. A terceira incógnita da expressão (2), denominada capacidade admissível, representa a capacidade de cada caixa de armazenamento. Assim, pode-se obter, com auxílio da expressão (2), o número de caixas usadas para o armazenamento de um determinado tipo de peça.

A equação do numerador da expressão (3) representa o cálculo da soma dos pesos de todas as caixas, enquanto a incógnita do denominador da expressão (3) representa a capacidade admissível do carrinho.

$$n_caixas = trunc\left(\frac{peso\ peça[kg] * tamanho\ lote}{capacidade\ admissível[kg]}\right) \quad (2)$$

$$n_vezes = trunc\left(\frac{\sum_{i=1}^{n_caixas} (peso\ peça_{(i)}[kg] * tamanho\ lote)}{capacidade\ admissível\ carrinho[kg]}\right) \quad (3)$$

A distância percorrida entre máquinas, neste caso, então é obtido multiplicando “n_vezes” e “DM’s”.

- Arranjo físico distribuído: como a forma de trabalho deste é quebrar lotes grandes em lotes pequenos, não se pode contabilizar as distâncias percorridas por cada peça do lote. Se assim for feito, o deslocamento no arranjo físico distribuído será bem superior ao do arranjo físico funcional. A explicação é que no arranjo físico funcional, o lote todo é que faz o percurso e no arranjo físico distribuído, é peça por peça deste lote. Então, considerando que um lote tem 10 peças e a distância entre duas máquinas de operações consecutivas seja de 2m, tem-se no arranjo físico funcional apenas 2m de deslocamento, enquanto que no arranjo físico distribuído, tem-se 20m. Portanto, não se deve abordar o problema do arranjo físico distribuído contabilizando diretamente a expressão (1).

A expressão (1), apesar de não poder ser usada diretamente para contabilizar o fluxo no arranjo físico distribuído, deve ser usada para identificar máquinas para a formação da célula virtual.

Existe também a possibilidade de se fazer adaptação na formulação matemática QAP, tal como considerar as distâncias das máquinas até às áreas externas. Para isso, é necessário considerar uma linha imaginária central, denominada LIC, tal como mostrada em cada Figura 2.12.

Esta linha divide o chão de fábrica exatamente em duas partes iguais. O propósito principal de ilustrar estas duas figuras é mostrar que, dependendo dos locais (onde as máquinas serão alocadas) em relação a LIC, a distância a ser percorrida pela peça (da entrada a uma máquina qualquer ou de uma máquina qualquer a saída) no eixo y é a soma de vários DM's ou é a soma de vários DM's mais DM's/2.

Apenas Montreuil et al. (1991) comentam sobre a importância de computar as variáveis referentes ao local de saída e de entrada, porém também não chegaram a formulá-las matematicamente.

Enfim, o cálculo da distância referente à entrada até a máquina escolhida, e o cálculo da distância da máquina (onde foi realizada a última operação) até a saída podem ser realizados usando-se as expressões (4) a (7). A soma dos módulos das expressões (4) e (5) calcula a distância percorrida pelas peças da entrada até a máquina escolhida. Da mesma forma, das expressões (6) e (7) pode-se obter a distância da máquina até a saída.

$$\text{Deslocamento horizontal} = |Y_{\text{máq. escolhida}} - Y_{\text{entrada}}| \quad (4)$$

$$\text{Deslocamento vertical} = |X_{DL1E} - X_{\text{máq. escolhida}}| \quad (5)$$

$$\text{Deslocamento horizontal} = |Y_{\text{última máquina}} - Y_{\text{saída}}| \quad (6)$$

$$\text{Deslocamento vertical} = |X_{DSL2} - X_{\text{última máquina}}| \quad (7)$$

2.5.7 Extensões do QAP

Conforme o sistema real em estudo se torna complexo, as dificuldades para se obter uma solução ótima pelo QAP também aumentam. Sendo assim, uma alternativa é, então, apelar para métodos heurísticos. Recentemente, diversos métodos heurísticos foram desenvolvidos e aplicados em diversos trabalhos relacionados aos estudos de arranjos físicos sujeitos a um ambiente instável de mercado. Para melhor contextualizar a relação entre os métodos heurísticos desenvolvidos e os trabalhos da literatura, pode-se dividir esses trabalhos em duas categorias:

- Arranjo físico não-fixo (dinâmico) para um ambiente instável: consiste em alterar as máquinas de posição, buscando sempre adequá-las aos novos perfis de demanda. De acordo com Montreuil & Laforge (1992), a existência de um ambiente instável não é problema para se projetar um arranjo físico se as máquinas forem facilmente movidas de um lugar ao outro. Esta categoria de trabalho, então, leva em consideração troca de equipamentos de posição bem como custos associados à realocação.

O trabalho de Rosenblatt (1986) apresenta um modelo matemático com enfoque em custo de realocação. Obviamente, quanto menor o valor de realocação, menor é a mudança de equipamentos de sua posição.

Erel et al. (2003) propõem uma heurística, baseada em custo de fluxo, para a geração de arranjo físico. Já Conway & Venkataramanan (1994) fazem do uso de algoritmo genético para auxiliar na geração de novas populações de arranjos físicos para cada instante de tempo t .

- Arranjo físico fixo (robusto) para um ambiente instável: consiste em definir um arranjo físico que seja adequado para todos os instantes de tempo, ou seja, uma estrutura única e fixa. Benjaafar et al. (2002) comentam que realizar realocações periódicas de máquinas não é uma solução “inteligente”. É muito mais atrativo elaborar um arranjo físico que opere adequadamente ao longo do tempo. Alguns autores, tais como Webster & Tyberghein (1980), chamam esta robustez de “flexibilidade de arranjo físico”, que é a capacidade/habilidade do arranjo físico em absorver as flutuações de demanda.

Uma forma de obter este tipo de arranjo físico é levantar as informações sobre o comportamento estocástico do perfil da demanda. Um dos trabalhos que abordam o problema dessa forma é o de Palekar et al. (1992).

Benjaafar & Sheikhzadeh (2000) apresentam um procedimento matemático, onde um arranjo físico robusto é obtido a partir das informações de mercado. O procedimento adotado pelos autores realiza a contabilização da distância percorrida pelo lote unitário da peça entre as máquinas. A solução pode ser melhorada aplicando-se a técnica *pairwise exchange* (pares de máquinas são mudadas dos seus respectivos locais) e assim o desempenho é novamente avaliado. A melhor solução é então armazenada pelo programa computacional. Este tipo de arranjo físico robusto é o mesmo que Montreuil et al. (1991), na realidade, já vinham propondo, que é denominado de arranjo físico distribuído. Infelizmente estes autores não comentam como o arranjo físico trabalha, apenas apresentam o desempenho. Isso dificulta o entendimento dos resultados.

Heragu (1990) apresenta um “QAP modificado” para a obtenção dos custos relacionados à movimentação de materiais entre máquinas. Tate & Smith (1995) apresentam um

procedimento heurístico para obter uma solução deste QAP modificado. O procedimento apresentado é uma adaptação do algoritmo genético.

Existem também trabalhos que fazem comparações de desempenho entre os arranjos físicos distribuídos ainda usando a formulação matemática QAP. Esta comparação é importante para mostrar situações ou em que condições de operação um arranjo físico distribuído apresenta melhores resultados que o tradicional arranjo físico funcional. Lahmar & Benjaafar (2005) apresentam resultados computacionais (custos de rearranjo) dos arranjos físicos (maximamente, parcialmente, aleatoriamente e funcional) avaliando, por exemplo, o impacto do número de tipos de peças no custo operacional de fabricação. Estes autores avaliam também a relação do nível de robustez de cada tipo de arranjo físico em relação à variação do número de réplicas de máquinas.

Gorgulho Júnior & Gonçalves Filho (2007) fazem um estudo comparativo de desempenho entre os arranjos físicos distribuídos em relação ao arranjo físico funcional considerando a flexibilidade de processamento de peças e verificaram que o arranjo físico maximamente distribuído é o que resulta em menor deslocamento total de peças entre máquinas. Para a realização do estudo, estes autores usaram apenas códigos de programação. Os autores afirmam que a ordem de desempenho, do pior para o melhor, é: arranjo físico funcional, arranjo físico parcialmente distribuído, arranjo físico aleatoriamente distribuído e arranjo físico maximamente distribuído.

Nota-se, com base nos artigos consultados, que cada tipo de peça a ser submetida ao processo de fabricação chega ao chão de fábrica de forma unitária e não em lotes. Ao que tudo indica, é fundamental que estudos mostrem a real vantagem de se adotar um arranjo físico distribuído em relação aos arranjos físicos tradicionais levando-se em consideração outros parâmetros, tais como chegada de lotes e “quebra” destes em lotes menores.

Dentre os trabalhos que usam as técnicas de modelagem e simulação computacional para se estudar um arranjo físico distribuído, pode-se citar Pitombeira Neto et al. (2007). Estes autores apresentam os resultados do simulador Promodel® avaliando a relação da variação WIP (Work-In-Process) e velocidade do transportador. Neste trabalho, verifica-se que quanto maior a velocidade do transportador, menor é o WIP. Os autores consideram um número fixo de 12 máquinas (3 linhas e 4 colunas) não podendo ser estendido para avaliação de desempenho com uma quantidade diferente de máquinas. Além disso, os autores abordam o arranjo físico como sendo puramente de transporte, sem levar em consideração a formação de células virtuais.

Pitombeira Neto (2007) também faz do uso das técnicas de modelagem e simulação para avaliar o desempenho de um arranjo físico distribuído, porém o autor aborda o arranjo físico distribuído como sendo simplesmente o espalhamento das máquinas no chão de fábrica, sem lidar com a formação das células virtuais. Além disso, o autor considera que o lote todo é movimentado entre as máquinas por um transportador, como se fosse num arranjo físico funcional, sem considerar a “quebra” dos lotes.

Askin et al. (1999) apresentam um estudo comparativo de desempenho, usando simulação, entre o arranjo físico distribuído, arranjo físico funcional e arranjo físico fractal. Estes autores verificaram que o desempenho do arranjo físico distribuído se situa entre o arranjo físico funcional e o arranjo físico fractal. Entretanto, realizam o estudo para apenas um único tipo de arranjo físico distribuído, sem especificar qual dos três tipos que foi adotado em estudo. Além disso, usou-se apenas códigos de programação para realizar a simulação.

Reaes (2008) faz um estudo comparativo de desempenho de um arranjo físico distribuído usando o simulador ARENA® versão 9.0. De acordo com o sistema de manufatura modelado, este apresenta dez departamentos, sendo que em cada um está contida uma quantidade fixa de máquinas. A separação dos departamentos é visível a olho nu e de acordo com o autor, cada departamento é chamado de célula virtual. Além disso, o autor rigidifica o número de máquinas em cada departamento pois não considera a identificação instantânea de máquinas para a formação da célula virtual; o modelo construído pelo autor não pode ser usado para efetuar simulação para acima de dez departamentos; ainda com relação ao modelo construído pelo autor, as máquinas não são aproveitadas para a formação de novas células virtuais e, por fim, as máquinas de cada departamento só são usadas para a produção de uma família de peças. Conforme estas características apresentadas pelo autor, o sistema de manufatura modelado, então, aborda o conceito de formação de célula virtual diferentemente do que é proposto na literatura.

Verifica-se que, de acordo com os trabalhos citados anteriormente, ou não consideram a formação de células virtuais; ou restringem a simulação para apenas uma amostra de arranjo físico; ou usam apenas códigos de programação para avaliar o percurso da peça entre as máquinas (desconsiderando as questões de concomitâncias no uso de recursos produtivos); ou cada tipo de peça a ser submetida a fabricação chega ao chão de fábrica de forma unitária (diferindo de fato com o que acontece na prática, que chega em lote).

Baseado no conceito teórico referente à formação de célula virtual apresentado na literatura, pode-se deduzir alguns passos, conforme apresentados a seguir, para auxiliar na formação de células virtuais em um arranjo físico distribuído, que são os procedimentos adotados neste estudo:

- Assim que uma determinada peça for requisitada, a fábrica deve imediatamente ser capaz de visualizar todas as operações (necessariamente todas) referentes à essa peça requisitada;
- Identificar instantaneamente, no chão de fábrica, as máquinas que fazem parte da operação da peça requisitada e que estejam necessariamente livres;
- Reservar este conjunto de máquinas (célula virtual);
- Garantir que nenhum outro tipo de peça use essas máquinas;
- Transferir a peça do lote nas máquinas da célula virtual em lotes pequenos, se possível lote unitário (como num arranjo físico celular);

- Após o término de processamento (ou seja, última operação da última peça do lote), disponibilizar as máquinas desta célula virtual para que outros tipos de peças possam formar células virtuais com essas máquinas recém liberadas.

Se a abordagem da célula virtual não for feita desta forma (conforme apresentados nos seis itens), então o estudo de um arranjo físico distribuído não está conceitualmente correto. De acordo com Montreuil et al. (1993), é justamente esta a dificuldade de identificar instantaneamente no chão de fábrica máquinas que irão formar a célula virtual, pois é necessário um sistema de controle em tempo-real para a escolha da melhor programação para os lotes. E justamente o que este trabalho de pesquisa propõe abordar, através do modelo de simulação.

2.6 CUSTEIO DE PRODUÇÃO

A aplicação da formulação matemática de QAP em estudos permite a obtenção dos tempos de deslocamentos de peças entre máquinas. Ao implementar esta formulação matemática QAP em modelos de simulação, é possível obter também o tempo de consolidação dos pedidos.

Como um dos objetivos principais deste trabalho é avaliar também o custo de consolidação dos pedidos solicitados pelos clientes, então este tópico tem a finalidade de apresentar as formulações usadas na literatura para obter os custos de fabricação das peças. Não é objetivo deste trabalho desenvolver um novo método de custeio, mas sim de analisar o *trade-off* entre o tempo e custo. Por isso, o custo aqui revisado é usado apenas como uma ferramenta para fins de comparação.

Segundo a *SOCIETY OF MANUFACTURING ENGINEERS* (1965), o custo real de produção das peças pode ser obtido usando-se a expressão (8), a qual é composta por outros quatro componentes de custo:

$$C_{\text{real}} = C_{\text{manufatura}} + C_{\text{engenharia}} + C_{\text{distribuição}} + C_{\text{administrativo}} \quad (8)$$

Em que:

- $C_{\text{manufatura}}$ representa o custo associado às transformações da peça;
- $C_{\text{engenharia}}$ representa o custo associado ao desenvolvimento da peça antes de iniciar a transformação;
- $C_{\text{distribuição}}$ representa o custo associado às atividades relacionadas às entregas da peça ao cliente;
- $C_{\text{administrativo}}$ representa o custo associado às atividades relacionadas ao processamento de pedido da peça, às atividades relacionadas ao custo de pessoal da área de administração etc.

Será considerado neste trabalho apenas o componente de custo $C_{\text{manufatura}}$, porque é no sistema de manufatura que ocorre a transformação física da peça.

O custo de manufatura (ou $C_{\text{manufatura}}$) é composto por outros três componentes de custo, tal como mostrado na expressão (9):

$$C_{\text{manufatura}} = C_{\text{fixo produção}} + C_{\text{inventário produção}} + C_{\text{efetivo produção}} \quad (9)$$

O custo fixo de produção (ou $C_{\text{fixo produção}}$) está associado aos aluguéis do local onde ocorre a transformação física da peça etc. Este será desconsiderado, pois não se altera em função das melhorias implementadas no sistema produtivo. Ou seja, independente de como é realizada a produção, o custo fixo de produção será o mesmo.

O custo de inventário (ou $C_{\text{inventário produção}}$) está associado ao custo de todo material que permanece no sistema de manufatura, independentemente se já está concluído ou não. O material referido pode ser peças em estoque, peças semi-acabadas, peças em transformação ou peças em fila aguardando processamento na máquina.

O custo efetivo de produção (ou $C_{\text{efetivo produção}}$) está associado ao custo de produção (ou $C_{\text{produção}}$) e associada à eficiência de produção (ou $\eta_{\text{produção}}$), conforme apresentado na expressão (10):

$$C_{\text{efetivo produção}} = \left(\frac{C_{\text{produção}}}{\eta_{\text{produção}}} \right) \quad (10)$$

A eficiência de produção calcula a relação entre a carga horária requisitada pelas peças nas máquinas e a carga horária disponível nas máquinas, conforme mostrada na expressão (11). Já o custo de produção, representado na expressão (12), calcula a soma de outros componentes de custo. Enfim, para calcular o custo efetivo de produção ($C_{\text{efetivo produção}}$), é necessário então ter obtido os valores das expressões (11) e (12).

$$\eta_{\text{produção}} = \left(\frac{\sum t_p}{\sum td_i} \right) = \frac{\text{Somatório dos tempos padrões produzidos no mês}[h]}{\text{Somatório dos tempos padrões disponíveis no mês}[h]} \quad (11)$$

$$C_{\text{produção}} = C_{\text{montagem}} + C_{\text{ajustagem}} + C_{\text{contqual}} + C_{\text{embalagem}} + C_{\text{despacho}} + C_{\text{variável}} \quad (12)$$

Um dos componentes da expressão (12), denominado de custo de montagem (ou $C_{montagem}$), está associado à fixação da peça na máquina para a realização de uma determinada transformação (ou processamento). No caso deste trabalho, este custo não deve ser contabilizado porque já está embutido no custo de mão-de-obra (conforme será discutido mais adiante), uma vez que o próprio operador da máquina já faz a montagem da mesma.

O custo de ajustagem (ou $C_{ajustagem}$) está associado à fixação da ferramenta na máquina. Este custo também será desconsiderado porque o próprio operador é quem faz esta fixação de ferramental na máquina. Portanto, este custo também já está embutido no custo de mão-de-obra.

O custo de controle da qualidade (ou $C_{contqual}$) está associado às atividades do departamento de controle de qualidade das peças. Entretanto, este custo será desconsiderado por não ser o enfoque desse trabalho de pesquisa.

O custo de embalagem (ou $C_{embalagem}$) está associado ao processo de embalar as peças. Este custo de embalagem também será desconsiderado porque este ocorre após o processo de fabricação.

O custo de despacho (ou $C_{despacho}$) está associado ao processo de envio de peças ao cliente. Este também será desconsiderado, pois este trabalho está focando apenas no processo de fabricação.

Assim, para este trabalho, o cálculo do custo de produção (ou $C_{produção}$), então, se resume à expressão (13):

$$C_{produção} = C_{variável} \quad (13)$$

O custo variável (ou $C_{variável}$) está associado a todos os recursos que participam efetivamente do processo de transformação física da peça que ocorre em um sistema de manufatura. Este custo variável pode ser visto na expressão (14):

$$C_{variável} = C_{máquina} + C_{energia} + C_{mão-de-obra} + C_{ferramental} + C_{material} \quad (14)$$

O componente denominado de custo de máquina (ou $C_{máquina}$) da expressão (14) está associado à aquisição de máquinas. Além disso, este custo de máquina está associado ao funcionamento das máquinas no chão-de-fábrica. Este custo de máquina, então, pode ser visto na expressão (15):

$$C_{máquina} = C_{juro} + d_{depreciação} + C_{manutenção} \quad (15)$$

O custo de juro (ou C_{juro}) está associado ao pagamento realizado pela empresa para adquirir as máquinas. Este trabalho irá considerar que todas as máquinas já foram previamente adquiridas e, portanto, este componente de custo de juro não será avaliado.

O custo de depreciação (ou $d_{\text{depreciação}}$) está associado à desvalorização de máquinas, em valor monetário, em decorrência do uso. Existem várias maneiras para a realização do cálculo de depreciação de um bem. Segundo Ehrlich (1989), existem três maneiras consideradas as mais importantes para a realização do cálculo de depreciação de um equipamento, conforme apresentadas a seguir:

1) Depreciação linear

Considere um equipamento a ser depreciado em n períodos. O valor inicial do equipamento, denominado V_o , é o valor inicialmente pago pela empresa para adquirir o equipamento. Já o valor residual, V_R , é um valor contábil residual que se espera atribuir ao equipamento após n períodos de uso. Para obter o valor de depreciação do equipamento após cada período de uso, representado por d , até o período n desejado, é necessário subtrair V_R do valor inicial V_o e dividir o resultado desta subtração por n períodos, conforme apresentada na expressão (16):

$$d = \frac{V_o - V_R}{n} \quad (16)$$

Para entender melhor este método de cálculo a fim de determinar a depreciação de um equipamento, considere o seguinte exemplo:

Seja um bem comprado por R\$100.000,00. Deseja-se saber qual o valor atribuído ao bem após 5 anos de uso (então, vida útil de 5 anos), já considerando a depreciação. Se o valor residual for diferente de zero (por exemplo, R\$10.000,00), então a depreciação anual será (17):

$$d = \frac{100.000 - 10.000}{5} = \frac{90.000}{5} = 18.000 \quad (17)$$

Neste caso, durante os 5 anos, tem-se os seguintes valores atribuídos ao bem em análise, por ano, já considerando a depreciação anual, ver Tabela 2.2:

Tabela 2.2 - Exemplo de depreciação linear (Fonte: adaptado de Ehrlich, 1989)

Ano	Depreciação d (em R\$)	Valor do bem já depreciado (em R\$)
1	18.000	100.000 - 18.000 = 82.000
2	18.000	82.000 - 18.000 = 64.000
3	18.000	64.000 - 18.000 = 46.000
4	18.000	46.000 - 18.000 = 28.000
5	18.000	28.000 - 18.000 = 10.000

2) Depreciação exponencial

Na prática, existem diversos tipos de equipamentos que não se depreciam de forma linear. Isto é, a depreciação varia conforme a eficiência do equipamento em função do tempo. Este tipo de depreciação, também é chamado de depreciação exponencial. Devido à complexidade para realizar o cálculo deste tipo de depreciação, existe um método simplificado, denominado de depreciação pela soma dos dígitos.

Para compreender o cálculo desta depreciação pela soma de dígitos, considere que n seja o número de períodos (anos), onde n refere-se à vida útil de um equipamento. O valor da depreciação pela soma de dígitos pode ser obtido usando-se a expressão (18). A incógnita d_t da expressão (18) é a depreciação pela soma dos dígitos de cada período t ; a incógnita V_o é o valor inicial do equipamento em análise a ser depreciado; a incógnita V_R é o valor residual contábil que se deseja atribuir ao equipamento após n períodos; a incógnita n é a vida útil do equipamento; a incógnita t é o período de interesse que se deseja conhecer sobre o valor do bem em depreciação e, por fim, a incógnita N é a soma aritmética do período 1 até o período n .

$$d_t = \frac{V_o - V_R}{N} (n - t + 1) \quad (18)$$

A incógnita N da expressão (18), denominada de soma dos dígitos, pode ser obtida usando-se a expressão (19):

$$N = \sum_{k=1}^n k \quad (19)$$

Para entender melhor este método de cálculo para determinar a depreciação de um equipamento, considere o seguinte exemplo:

Seja um bem comprado por R\$100.000,00. Deseja-se saber qual o valor atribuído ao bem após 10 anos de uso (então, vida útil de 10 anos), já considerando a depreciação. Como o período n é 10 anos, então a soma aritmética de N é igual a 55 (ou seja, $1+2+3+\dots+10$). Se o valor residual for R\$10.000,00, pode-se calcular a depreciação de cada período t (e, portanto, também pode-se calcular o valor do bem já depreciado em cada período) até o período n desejado, conforme mostrados nos resultados da Tabela 2.3:

Tabela 2.3 - Exemplo de depreciação pela soma de dígitos (Fonte: autor)

Período (ano)	Depreciação d_t (em R\$)	Valor do bem já depreciado (em R\$)
1	$d_1 = \frac{100.000 - 10.000}{55}(10 - 1 + 1) = 16.363,64$	$100.000 - 16.363,64 = 83.636,36$
2	$d_2 = \frac{100.000 - 10.000}{55}(10 - 2 + 1) = 14.727,27$	$83.636,36 - 14.727,27 = 68.909,09$
3	$d_3 = \frac{100.000 - 10.000}{55}(10 - 3 + 1) = 13.090,91$	$68.909,09 - 13.090,91 = 55.818,18$
4	$d_4 = \frac{100.000 - 10.000}{55}(10 - 4 + 1) = 11.454,55$	$55.818,18 - 11.454,55 = 44.363,63$
5	$d_5 = \frac{100.000 - 10.000}{55}(10 - 5 + 1) = 9.818,18$	$44.363,63 - 9.818,18 = 34.545,45$
6	$d_6 = \frac{100.000 - 10.000}{55}(10 - 6 + 1) = 8.181,81$	$34.545,45 - 8.181,81 = 26.363,64$
7	$d_7 = \frac{100.000 - 10.000}{55}(10 - 7 + 1) = 6.545,45$	$26.363,64 - 6.545,45 = 19.818,19$
8	$d_8 = \frac{100.000 - 10.000}{55}(10 - 8 + 1) = 4.909,09$	$19.818,19 - 4.909,09 = 14.909,10$
9	$d_9 = \frac{10.000 - 1.000}{55}(10 - 9 + 1) = 3.272,73$	$14.909,10 - 3.272,73 = 11.636,37$
10	$d_{10} = \frac{10.000 - 1.000}{55}(10 - 10 + 1) = 1.636,36$	$11.636,37 - 1.636,36 = 10.000,01$

Assim, de acordo com o exemplo, após 10 anos, o valor do bem que era de R\$100.000,00 passa a ser R\$10.000,01.

3) Depreciação por horas trabalhadas

Existe também um método de cálculo considerando a depreciação de um bem em função das horas trabalhadas, ou seja, depreciação por cotas proporcionais às horas de trabalho.

Suponha que a previsão de capacidade de trabalho de um equipamento seja de 15.000 horas com preço inicial de R\$100.000,00. Se no fim do primeiro ano houver tido uma duração de trabalho igual a 3.200 horas, então a depreciação para este primeiro ano, denominado d_t onde t é igual a 1, seria de R\$21.333,00, como mostrada na Tabela 2.4:

Tabela 2.4 - Exemplo de depreciação por horas trabalhadas (Fonte: adaptado de Ehrlich, 1989)

Ano	Depreciação d_t (em R\$)	Valor do bem já depreciado (em R\$)
1	$d_1 = \frac{3.200}{15.000} * 100.000 = 21.333,00$	$100.000 - 21.333 = 78.667,00$

Pode-se notar, de acordo com o exemplo ilustrado na Tabela 2.4, que inicialmente o valor de um bem, que era de R\$100.000,00, passou a ser de R\$78.667,00 após um ano de uso. Para este trabalho de pesquisa, adotou-se este terceiro método de cálculo de depreciação, ou seja, depreciação por horas trabalhadas.

Ainda com relação à expressão (15), o componente de custo denominado de custo de manutenção (ou $C_{\text{manutenção}}$) está associado ao uso de serviços especializados solicitados pela empresa para manter as máquinas em plenas condições de funcionamento, o que inclui serviços prestados pelas empresas especializadas responsáveis pela manutenção etc. Assumindo a hipótese de que não houve nenhuma manutenção ocorrida durante o período em análise, este custo de manutenção então será desconsiderado neste trabalho.

Dando continuidade na análise dos componentes de custo variável da expressão (14), o componente denominado custo de energia (ou C_{energia}) está associado ao consumo de energia ocorrido nas máquinas. A expressão (20) permite a obtenção do custo de energia em máquinas, sendo que as máquinas em análise podem ser máquinas elétricas, máquinas hidráulicas etc.

$$C_{\text{energia}} = \text{Pot} * \text{tempo} * C_{\text{energia/hora}} \quad (20)$$

A incógnita da expressão (20) denominada Pot (cuja unidade é dada em KW) refere-se à potência consumida pelas máquinas. A incógnita denominada tempo refere-se ao tempo de consumo de energia em horas, contabilizando apenas os instantes de tempo em que as máquinas fazem processamento das peças. E por fim, a incógnita, denominada $C_{\text{energia/hora}}$, é o valor (em R\$) a ser pago pela empresa pelo consumo de energia por hora. A unidade de medida da incógnita $C_{\text{energia/hora}}$ é em R\$/KWh e este valor ($C_{\text{energia/hora}}$) pode ser facilmente obtido em contas de energia elétrica. Por exemplo, no dia 08/05/2007, o valor estabelecido pela empresa CPFL, prestadora de serviços energéticos do interior do estado de São Paulo, foi de R\$0,354838/KWh. Baseado neste valor, os custos de consumo de energia de todas as máquinas podem então ser obtidos.

O componente de custo da expressão (14), denominado de custo de mão-de-obra (ou $C_{\text{mão-de-obra}}$), está associado ao uso de funcionários nas operações de fabricação. O resultado do custo de mão-de-obra pode ser obtido usando-se a expressão (21), porém note que este é composto por mais dois componentes de custo, denominados de custo dos assalariados (ou $C_{\text{salário}}$) e outros custos adicionais (ou $C_{\text{adicionais}}$). O primeiro componente da expressão (21), custo dos assalariados, está associado a todos os funcionários que recebem um valor de salário mensal, independente das atividades realizadas pelos funcionários. Enquanto o segundo componente da expressão (21), denominado de custos adicionais, está associado a todos os funcionários que recebem proporcionalmente ao tempo trabalhado, geralmente dados em R\$/h.

$$C_{\text{mão-de-obra}} = C_{\text{salário}} + C_{\text{adicionais}} \quad (21)$$

Ainda dando continuidade na análise dos componentes de custo variável da expressão (14), o custo de ferramental (ou $C_{\text{ferramental}}$) está associado às ferramentas usadas na fabricação das peças. O custo do ferramental é, então, obtido usando-se a expressão (22).

$$C_{\text{ferramental}} = d_{\text{depreciação}} + C_{\text{troca}} + C_{\text{afiação}} \quad (22)$$

O componente de custo da expressão (22) denominado de custo de depreciação (ou $d_{\text{depreciação}}$) está associado à desvalorização das ferramentas, em valor monetário, em decorrência do uso. Para obter o resultado do custo de depreciação, pode-se trabalhar com a hipótese de depreciação por quantidade de peças produzidas, ou com a hipótese de depreciação pelo tempo de processamento, ou com a hipótese de depreciação que varie em função da intensidade de força aplicada pelo ferramental

na peça. Para este trabalho, considera-se o custo de depreciação pelo tempo de processamento do ferramental.

O segundo componente da expressão (22) denominada de custo de troca (ou C_{troca}) está associado às aquisições de novos ferramentais. Quanto maior o uso do ferramental, maior o desgaste e consequentemente maior a necessidade de trocas de ferramental (troca de um ferramental desgastado por um ferramental novo). Portanto, maior o custo de troca.

E por fim, o componente de custo da expressão (22) denominado de custo de afiação (ou $C_{afiação}$) está associado aos possíveis ajustes a serem feitos nos ferramentais para que estes possam, durante a operação, executar a suas funções corretamente. Para este trabalho, este componente será desconsiderado, porque considera-se que as ferramentas adquiridas já estão conforme às normas e prontas para serem usadas. Além disso, adota-se a hipótese de que em pleno funcionamento, a ferramenta não necessite de afiações. O cálculo de custo de ferramental, pode ser realizado usando-se a expressão (22):

Finalmente, o último componente de custo da expressão (14), denominado de custo de material (ou $C_{material}$) está associado aos materiais brutos a serem usados em fabricação. As unidades comumente usadas para o custo de material são dadas por R\$/kg ou R\$/unidade. Para este trabalho, adota-se R\$/unidade.

2.7 CRITÉRIOS DE SELEÇÃO DE PEÇAS E MÁQUINAS

As ferramentas de modelagem e simulação usadas neste trabalho permitem explorar as possíveis concomitâncias (requisições simultâneas de recursos produtivos) que ocorrem com muita frequência em um sistema real. Por esse motivo, este tópico tem a finalidade de apresentar os critérios de seleção de recursos produtivos abordados na literatura que podem ser usados neste trabalho.

Porém, ainda não é enfoque deste tópico selecionar os critérios de seleção de recursos produtivos a serem usados neste estudo, porque depende da dinâmica da empresa fictícia adotada neste trabalho de pesquisa. Mais adiante, com a descrição desta dinâmica, os critérios de seleção de recursos produtivos podem, então, ser definidos.

De uma forma geral, os critérios de seleção de recursos produtivos abordados na literatura, referentes aos sistemas de manufatura, podem ser separados em dois grupos: critérios de seleção de peças e critérios de seleção de máquinas.

- Critérios de seleção de peças: este grupo de critérios de seleção define a ordem em que as peças são processadas nas máquinas. Gupta (1999) aponta que ao todo existem oito tipos de seleção de peças. São: FCFS (O que chega primeiro é atendido primeiro, conforme a ordem de chegada), LCFS (O ultimo que chega tem a prioridade de ser atendido), SPT (escolha da peça

com o menor tempo de processamento), LPT (escolha da peça com o maior tempo de processamento), LREM (Menor tempo que ainda resta para finalizar a peça), MREM (Maior tempo que ainda resta para finalizar a peça), LTOP (Menor tempo total de processamento), MTOP (Maior tempo total de processamento).

Subramanian et al. (2000) apontam que, além dos citados anteriormente, existe a seleção aleatória de peças e também seleção de peça baseada em prazo de entrega;

- Critérios de seleção de máquinas: este grupo de critérios de seleção define o direcionamento que deve ser dado às peças para a escolha de máquinas. Sawik (1995) apresenta sete critérios de escolha de máquinas que podem ser usados, geralmente, em um sistema de manufatura. Os critérios de seleção de máquinas citados por Sawik (1995) são: aleatoriamente, menor/maior tempo de transporte (ou máquina mais próxima/distante), menor/maior tamanho de fila (ou menor/maior tempo de espera em fila), maior/menor carga de trabalho (máquina mais/menos ocupada).

2.8 ORGANIZAÇÃO DOS DADOS PARA A REALIZAÇÃO DOS EXPERIMENTOS

Como o objetivo deste trabalho de pesquisa é avaliar o desempenho entre o arranjo físico funcional e o arranjo físico distribuído, tem-se então, ao todo, quatro tipos de arranjos físicos, porque existem três tipos de arranjos físicos distribuídos mais o arranjo físico funcional.

Conforme Rheault et al. (1995), o perfil de mercado instável é caracterizado por alta variação na demanda; frequentes mudanças no conjunto de produtos; alta variação no tamanho do lote de produção; alta variação nos tempos de operação; alta variação nos tempos de preparação de ferramental e sequências de produção variáveis. Isso quer dizer que deve-se sujeitar cada tipo de arranjo físico às condições expostas por esses autores para identificar, assim, qual o melhor arranjo físico para atuar neste mercado de extrema instabilidade. Estas condições também são denominadas de fatores de experimentação.

Entretanto, em cada fator de experimentação, existem também faixas de valores. Por exemplo, se o fator de experimentação a ser submetido ao modelo de simulação é variação do tempo de uma operação, então um dos valores possíveis a ser inserido ao modelo de simulação para fins de experimento é que o tempo escolhido esteja entre 20s e 50s. Também existe a possibilidade do tempo de cada operação estar entre 15s e 40s. Enfim, essas possibilidades também são chamadas de níveis de fatores.

Existe uma alternativa para melhor representar os fatores de experimentação e seus respectivos níveis, que é por uma tabela. Por exemplo, Montgomery (1997) apresenta um experimento no qual

deseja-se saber qual a influência da temperatura e a influência do percentual de carbono na dureza de um determinado material. Para este exemplo, tanto temperatura quanto percentual de carbono são os fatores de experimentação. Entretanto, ainda existe, para cada fator de experimentação, níveis. Então, ainda de acordo com o exemplo de Montgomery (1997), as temperaturas a serem analisadas são 900° C e 1100° C (níveis 1 e 2, respectivamente), e os percentuais de carbono a serem analisados são 5% e 15% (níveis 1 e 2, respectivamente). São feitos dois ensaios para cada combinação de temperatura e percentual de carbono. A Tabela 2.5 apresenta os resultados obtidos neste ensaio, com todos experimentos possíveis. Estes resultados também são chamados de respostas do experimento.

Tabela 2.5 - Resultados do ensaio de dureza para o exemplo considerado (Fonte: adaptado de Montgomery, 1997)

Experimento	Temperatura (Fator)	% Carbono (Fator)	Dureza - Média dos ensaios
1	Nível 1	Nível 1	19,0
2	Nível 1	Nível 2	42,5
3	Nível 2	Nível 1	31,0
4	Nível 2	Nível 2	52,0

Pode-se notar, também, que quanto maior o número de fatores de experimentação analisado, bem como quanto maior o número de níveis de cada fator, aumentam-se as combinações entre os níveis e, conseqüentemente, aumentam-se também os números de experimentos a serem realizados. Para se ter idéia, se um determinado estudo de simulação tiverem seis fatores importantes a serem analisados e em cada fator existirem quatro níveis, isso implica em $4 \times 4 \times 4 \times 4 \times 4 \times 4 = 4096$ experimentos.

Por questões de viabilidade, trabalhos da literatura relacionados aos arranjos físicos reduzem, ou o número de fatores, ou o número de níveis para a realização dos experimentos. Por exemplo, Reaes (2008) apresenta um estudo comparativo de desempenho entre o arranjo físico funcional, o arranjo físico celular e o arranjo físico celular virtual. Para conduzir os seus experimentos em modelo de simulação, este autor considerou seis fatores de experimentação: o primeiro fator de experimentação é o tipo de sistema, o segundo fator de experimentação é o nível de redução de troca de ferramental; o terceiro é o tamanho do lote de peças; o quarto é a flexibilidade de operadores; o quinto é a variabilidade da demanda; e por fim, o sexto, é a variação nos intervalos entre chegadas de peças a serem fabricadas. O número de níveis de cada fator de experimentação que o autor considerou no seu estudo pode ser visto na Tabela 2.6.

Tabela 2.6 - Fatores e quantidade de níveis selecionados para os experimentos (Fonte: Reaes, 2008)

Tipo de sistema	Nível de redução de troca de ferramenta	Tamanho do lote de peças	Flexibilidade dos operadores	Variabilidade da demanda	Variação nos intervalos entre chegadas de peças
3	2	4	3	3	2

A Tabela 2.6 mostra seis colunas, que correspondem aos fatores adotados pelo autor no seu estudo. A primeira coluna mostra o primeiro fator denominado Tipo de sistema e este é composto por três níveis. O número 3 da primeira coluna representa os três tipos de arranjos físicos (arranjo físico funcional, arranjo físico celular, arranjo físico celular virtual). Para cada nível de arranjo físico, o autor considera apenas uma amostra.

Na segunda coluna, tem-se os níveis relacionados às trocas de ferramentas. O autor considera dois níveis. Para obter estes níveis, deve-se primeiramente definir um tempo padrão de troca de ferramental comum para as máquinas. No seu estudo, este autor considera como sendo uma exponencial de 3h, ou $EXPO(3)h$. Em seguida para obter os níveis relacionados aos tempos de trocas de ferramentais, considera-se um fator de redução, representado pelo símbolo δ . Por exemplo, este autor considerou que a empresa é capaz de reduzir o tempo de troca de ferramental para 40%, então o primeiro nível referente à segunda coluna da Tabela 2.6 será $Expo(3 \times 0,60)h$. Da mesma forma, para o segundo nível, se houve redução no tempo de troca de ferramental de 75%, quer dizer que o tempo de troca é de $Expo(3 \times 0,25)h$.

A terceira coluna refere-se ao tamanho do lote. Foram adotados quatro níveis. Os tamanhos dos lotes considerados por Reaes (2008) foram 30, 40, 50 e 60 peças. Outros tamanhos de lotes podem ser vistos em Suresh & Slomp (2005). Por exemplo, estes autores consideram doze níveis: 5, 10, 15, 20, 25, 30, 35, 40, 50, 60, 80 e 100 peças.

Ainda com relação à Tabela 2.6, a quarta coluna refere-se às habilidades dos operadores de controlar várias máquinas. Reaes (2008) denomina η como sendo a flexibilidade do operador. Os níveis adotados foram $\eta = 1$ o que significa um operador responsável pelo controle de apenas um tipo de máquina; $\eta = 2$ o que significa um operador responsável pelo controle de dois tipos de máquinas; e $\eta = 3$ o que significa um operador responsável pelo controle de três tipos de máquinas.

A quinta coluna da Tabela 2.6 refere-se à variabilidade da demanda. De acordo com o autor, a variação da demanda depende da família de peça. Por isso, este autor divide a demanda em três

famílias, dados por famílias 1 e 5, família 2 e família 4. Cada família apresenta três níveis: os quais são denominados de: nula (demanda constante), um quarto da plena (pouca variação na demanda) e plena (alta variação). A representação “Unif” significa distribuição uniforme. Estes níveis de variação da demanda por família de peças podem ser vistos na Tabela 2.7.

Tabela 2.7 - Os três tipos adotados de variação de demanda anual de peças (Fonte: Reaes, 2008)

Famílias	Nula	1/4	Plena
1 e 5	1240	Unif (992-1488)	Unif (248-2232)
2	1550	Unif (1240-1860)	Unif (310-2790)
4	1975	Unif (1580-2370)	Unif (395-3555)

E por fim, a sexta coluna da Tabela 2.6 refere-se aos níveis de variação nos intervalos entre chegadas de peças. Os dois níveis são: nula (chegadas a intervalos constantes) e alta (segundo uma função exponencial). Para obter o valor do primeiro nível, o tempo é gerado pela expressão $(q/d_i) \cdot 3600$ horas, onde q é o tamanho do lote, d_i é a demanda gerada para a peça i e 3600 são as horas de simulação (aproximadamente um ano). Para obter o valor do segundo nível, o tempo é dado pela expressão $EXPO(q/d_i) \cdot 3600$ horas, onde, mais uma vez, q é o tamanho do lote, d_i é a demanda gerada para a peça e 3600 são as horas de simulação.

Então, de acordo com a Tabela 2.6, cada arranjo físico é submetido aos cinco fatores. Para a realização do experimento, define-se a combinação dos níveis entre fatores diferentes. Após a realização dos experimentos para todas as combinações possíveis, pode-se obter, então, uma resposta do experimento realizado no modelo de simulação para fins de comparação de desempenho entre o arranjo físico funcional, arranjo físico celular e arranjo físico celular virtual.

Askin et al. (1999) apresentam um estudo comparativo de desempenho entre o arranjo físico distribuído, o arranjo físico funcional e o arranjo físico fractal. Os autores consideraram quatro fatores de experimentação, os quais são: o tempo de cada operação, o número total de máquinas, o número de operações e a utilização de máquinas. Pode-se notar nas colunas da Tabela 2.8, então, os quatro fatores de experimentação adotados pelos autores. Cada fator considerado tem-se dois níveis.

A primeira coluna da Tabela 2.8 refere-se ao número de experimentos realizados, que também pode ser compreendida como sendo o número de combinações possíveis entre os níveis de todos os quatro fatores.

Ainda com relação à Tabela 2.8, a segunda coluna refere-se ao tempo de cada operação executada pela máquina na peça. Os autores adotaram, como sendo o primeiro nível, um tempo de operação constante e, como segundo nível um tempo de operação exponencial.

A terceira coluna da Tabela 2.8, refere-se ao fator denominado de número total de máquinas no chão de fábrica e este tem dois níveis, sendo que para o primeiro nível, considera-se 30 máquinas e para o segundo nível, considera-se 60 máquinas.

A quarta coluna da Tabela 2.8 refere-se ao fator denominado de número de operações, ou seja, o número de operações a serem executadas em cada peça. No primeiro nível, exige-se de um a cinco operações, enquanto no segundo nível, exige-se de um a 15 operações.

E por fim, na quinta coluna da Tabela 2.8, este refere-se ao fator referente à utilização de máquinas, ou seja, o nível de ocupação de cada máquina. Os autores consideram que cada máquina pode assumir então dois níveis, sendo o primeiro nível como 75% e o segundo nível como 85%.

Portanto, sendo quatro fatores e dois níveis cada um, tem-se $2 \times 2 \times 2 \times 2 = 16$ combinações possíveis ou 16 experimentos possíveis e é justamente como está representado na Tabela 2.8.

Tabela 2.8 - Níveis de fatores (Fonte: Askin et al., 1999)

Experimento	Tempo de operação	Número total de máquinas	Número de operações	Utilização
1	Fixo	30	5	75%
2	Fixo	30	5	85%
3	Fixo	30	15	75%
4	Fixo	30	15	85%
5	Fixo	60	5	75%
6	Fixo	60	5	85%
7	Fixo	60	15	75%
8	Fixo	60	15	85%
9	Exponencial	30	5	75%
10	Exponencial	30	5	85%
11	Exponencial	30	15	75%
12	Exponencial	30	15	85%
13	Exponencial	60	5	75%
14	Exponencial	60	5	85%
15	Exponencial	60	15	75%
16	Exponencial	60	15	85%

Em função das restrições computacionais (de *hardware* e de *software*), este trabalho de pesquisa considera cinco fatores de experimentação para avaliar o desempenho do arranjo físico

funcional e do arranjo físico distribuído. Estes cinco fatores são: número de operações, quantidade de cada tipo de peça demandada por cliente, tempo de cada operação, quantidade total de máquinas e tipo de arranjo físico. Estes cinco fatores estão organizados conforme apresentada na Tabela 2.9.

Tabela 2.9 - Fatores adotados para este estudo (Fonte: autor)

Fatores				
Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Quantidade total de máquinas	Tipo de arranjo físico

Ainda não é enfoque deste tópico descrever os níveis de cada fator a serem adotados neste estudo. Os referidos níveis de cada fator serão discutidos mais adiante, juntamente com outros dados a serem usados na simulação do modelo.

3 PROCEDIMENTOS ADOTADOS PARA A DEFINIÇÃO DA DINÂMICA DA FÁBRICA

O objetivo deste trabalho de pesquisa é realizar um estudo comparativo de desempenho entre o arranjo físico funcional e o arranjo físico distribuído através da simulação do modelo a ser construído. É necessário, então, definir o dinamismo da empresa em análise para que se possa dar início à modelagem em computador.

Pode-se notar, ao longo do texto, que diversos procedimentos são propostos (ou discutidos) com a finalidade de auxiliar nas etapas de caracterização da fábrica fictícia, importantes para a definição da estrutura lógica do modelo de simulação.

A principal característica da fábrica fictícia é que esta pode assumir diferentes tamanhos (quantidade total de máquinas), pois depende do instante t que se inicia o estudo comparativo. Independente disso, uma vez que as etapas que caracterizem a fábrica fictícia estiverem definidas, o modelo de simulação pode ser construído e a simulação do modelo pode ser conduzida sem alterações posteriores.

Por isso, este tópico enfatiza nos procedimentos adotados, sem dar ênfase ainda quanto à quantidade total de máquinas que a fábrica fictícia pode conter.

Em resumo, para este tópico devem ser estabelecidos:

- Os locais de trabalho;
- O número de tipos de peças;
- A demanda de peças;
- O número de máquinas;
- As possíveis configurações da fábrica;
- A matriz de distância de-para;
- Os arranjos físicos funcional e distribuído;
- As posições de vão vazio;
- A maneira de alocação de máquinas;
- Os recursos adotados na fabricação de peças;
- Os critérios de seleção de caixas;
- Os critérios de seleção de máquinas.

3.1 LOCAIS DE TRABALHO

Este tópico tem a finalidade de apresentar os locais de trabalho adotados neste estudo. A fábrica fictícia possui dois locais externos, denominados de Local 1 e Local 2, e uma área denominada chão de fábrica, onde as máquinas serão instaladas, como mostrada na Figura 3.1. As posições são usadas como artifício para identificar os locais onde as máquinas serão alocadas. Considera-se neste trabalho de pesquisa a hipótese de que todas as máquinas possuem o mesmo tamanho e de formato quadrado. Todos os arranjos físicos funcionais e os arranjos físicos distribuídos gerados para este estudo possuem, então, esses dois locais externos, denominados de Local 1 (ou L1) e Local 2 (ou L2). Estes dois locais encontram-se opostos entre si e encontram-se sobre a linha central imaginária (LIC).

Figura 3.1 - Vista superior do chão de fábrica (Fonte: autor)

Considera-se neste estudo que a demanda de mercado é representado por pedidos de clientes, e estes pedidos são recebidos no L1 da fábrica fictícia. Neste local (ou seja, no L1), também estão armazenadas matérias-primas das peças a serem usadas em processo de fabricação. Considera-se que existe disponibilidade suficiente de matéria-prima da peça requisitada pela demanda.

O L2 é o local de destino para as peças acabadas. Neste local, as peças acabadas ficam armazenadas até que sejam embaladas e despachadas. Para este trabalho de pesquisa, foca-se apenas até o momento em que as peças acabadas chegam ao L2, não se importando com as atividades posteriores que ocorrem neste local.

Após a definição desses dois locais externos no chão de fábrica, é possível obter as distâncias de cada posição até eles. Essas distâncias são importantes de serem levadas em consideração porque fazem parte do trajeto percorrido pelas peças. Para fins de determinação dessas distâncias, é necessário conhecer a distância entre duas máquinas adjacentes, que neste trabalho de pesquisa será adotada como sendo um metro para ambos os tipos de arranjos físicos (arranjo físico funcional e arranjo físico distribuído).

O procedimento a ser usado para obter as distâncias de cada local até um desses dois locais externos é computar a distância, na horizontal, de cada posição até o centro do chão de fábrica mais a distância, na vertical, do centro do chão de fábrica até o L1 ou L2. Este procedimento pode ser implementado em linguagem de programação para a obtenção da distância de cada posição até esses dois locais externos. O procedimento solicita que o usuário insira o número de posições, o número de linhas e o número de colunas que compõem o chão de fábrica. Com isso, as distâncias de cada posição até os dois locais externos são geradas.

3.2 GERAÇÃO DOS TIPOS DE PEÇAS

Este tópico tem por finalidade definir os perfis das peças a serem fabricadas na fábrica fictícia.

Os perfis de cada peça, importantes para este trabalho de pesquisa, são: o tipo de peça, o peso unitário, o número de operações, as operações executadas nas peças, os ferramentais usados e os tempos de cada operação.

Para melhor compreender a representação do perfil de peça adotado neste trabalho de pesquisa, considere o exemplo simplificado da Tabela 3.1. Nesta tabela, existem três colunas referentes aos perfis das peças denominadas de tipos de peças, número de processos e operações/ferramentais.

Tabela 3.1 - Exemplo de representação das operações e ferramentais por tipo de peça (Fonte: autor)

Tipos de peças	Número de Processos	Operações / ferramentais							
1	2	58	21						
2	8	47	24	62	49	62	43	61	62
3	1	81							
4	6	73	17	38	36	18	92		

Ainda com relação ao exemplo da Tabela 3.1, considere a representação do perfil da peça tipo 3. A coluna “número de processos” referente à esta peça tipo 3 informa que o número de processo é unitário, o que significa que a peça tipo 3 tem apenas uma operação. A coluna “operações/ferramentais” referente à esta peça tipo 3 é 81, o que significa que esta peça sofre operação em uma determinada máquina tipo 8 cuja representação é 8, usando um determinado ferramental tipo 1 representado por 1.

É importante ressaltar que peças com operações consecutivas idênticas, tais como 61 e 62 da peça tipo 2, são, então, realizadas na mesma máquina sem a necessidade de serem movimentadas de uma máquina à outra. Para este caso particular de operações consecutivas idênticas, muda-se a ferramenta de 1 para o ferramental 2. Isso é desejável para minimizar o erro esporádico de fabricação em decorrência da fixação da peça na máquina.

Após a definição das representações dos perfis de peças adotadas neste trabalho de pesquisa, é possível definir um procedimento que pode ser usado para gerar todos os perfis de peças. Este procedimento está apresentado na Figura 3.2.

Figura 3.2 - Procedimento para a geração de perfil de peças (Fonte: autor)

A ideia deste procedimento é solicitar que o usuário insira: o número de tipos de peças; o número de tipos de máquinas e o número de tipos de ferramentas e o número de operações (processos). Ao executar o procedimento, este inicia com a definição do perfil da primeira peça, escolhendo um número aleatório localizado entre uma operação até o número de operações que o usuário definiu. Após a definição do número de operações para a primeira peça, inicia-se com a caracterização das operações da primeira peça. Para a primeira operação, escolhe-se uma das máquinas, uma das ferramentas bem como o tempo de operação. Repete este mesmo procedimento de definição ocorrido na primeira operação para as demais operações restantes. Deve-se garantir que o ferramental usado em operações consecutivas realizadas no mesmo tipo de máquina não seja o mesmo. Antes de partir para a definição do perfil da segunda peça, pode-se também definir o seu peso.

Enfim, ao executar o procedimento para todas as peças, tem-se os perfis de todas elas.

O procedimento apresentado pode ser implementado e executado em linguagem de programação de modo a gerar dados referentes aos perfis de peças para fins de estudo.

3.3 GERAÇÃO DE DEMANDA

A etapa seguinte é gerar dados referentes às demandas das peças, fundamentais para definir o tamanho da fábrica fictícia (quantidade total de máquinas).

Neste trabalho de pesquisa, a demanda pode ser interpretada como sendo um conjunto de listas de peças enviadas pelos clientes ao longo do tempo. Estas listas, também são chamadas de pedidos.

Basicamente, estes pedidos contêm informações dos tipos de peça e das quantidades de cada tipo. Considerando, por exemplo, que no Tópico 3.2 tenha-se gerado 100 peças (isto é, a empresa só é capaz de produzir 100 tipos de peças), a geração da lista de demanda é, na realidade, uma escolha aleatória de vários tipos de peças de um conjunto de 100.

O procedimento apresentado na Figura 3.3 permite a escolha de qualquer tipo de peça para a composição do pedido de cada cliente. Cada pedido representa a demanda de um cliente. Assim, se a hipótese é trabalhar com n clientes, significa que o *loop* deverá ser realizado n vezes. O procedimento também permite definir os lotes para cada tipo de peça, que de acordo com a Figura 3.3, variam de um a três unidades.

As informações dos pedidos que chegam no L1, permitem, com auxílio do computador, identificar os pesos das peças, e as suas respectivas operações. Pode-se também identificar o tipo de matéria-prima a ser usada na fabricação de peças demandadas nesses pedidos.

Com as informações do tamanho dos lotes, bem como os tempos de cada operação, pode-se definir as réplicas de cada tipo de máquina, como mostrado no Tópico 3.4.

Figura 3.3 - Procedimento para a geração do perfil de demanda para todos os clientes (Fonte: autor)

3.4 DIMENSIONAMENTO DO NÚMERO DE RÉPLICAS DE CADA TIPO DE MÁQUINA

Este tópico apresenta um procedimento para a realização do dimensionamento do número de réplicas de cada tipo de máquina. Basicamente, os passos necessários para a realização do dimensionamento são: definir a demanda de cada tipo de peça; identificar as máquinas responsáveis por cada operação e seus respectivos tempos de operação; definir turnos de trabalho (tempo disponível das máquinas).

A expressão matemática (23) apresenta uma forma para o cálculo do número de réplicas para cada tipo de máquina, onde o numerador da expressão contabiliza a soma dos tempos de operação, exigido pelas peças, por cada tipo de máquina.

$$n.maq_{(m)} = trunc\left(\frac{\sum_{i=1}^J tempoprocesso maq_{(m)}}{tempo disponivel máq_{(m)}}\right) \quad (23)$$

Em que:

m: tipo de máquina;

i: contador do total de peças da demanda;

J: Total de peças da demanda.

Ainda com relação à expressão (23), a variável “tempo disponível máq_(m)” representa o tempo disponível da máquina tipo m. Se houvessem 15 tipos de máquinas, m então varia de 1 até 15, o que significa que deve-se usar a expressão (23) 15 vezes. É necessário que haja arredondamento para cima, daí trunc, pois a quantidade de réplicas de cada tipo de máquina não deve ser decimal.

A partir da expressão (23), pode-se definir um procedimento para a realização do dimensionamento do número de réplicas de cada tipo de máquina m. Este procedimento, então, pode ser visto na Figura 3.4, onde a variável i do procedimento representa o contador do total de peças da demanda. Assim, o *loop* deve ser executado J-1 vezes.

Figura 3.4 - Procedimento para o cálculo do número de réplicas de cada máquina (Fonte: autor)

Em resumo, a ideia do procedimento ilustrado na Figura 3.4 é identificar os tipos de máquinas responsáveis pelas operações de cada peça demandada bem como os tempos de cada operação. Ao efetuar este procedimento de cálculo para todas as peças da demanda, tem-se então o tempo total de operação de cada tipo de máquina usada na fabricação das peças. Ao informar ao procedimento o tempo disponível de cada tipo de máquina, tem-se então o número de réplicas de cada uma delas.

Para melhor compreender, considere que a demanda tenha solicitado 50 unidades da peça tipo A e 10 unidades da peça tipo B. Considere também que ambos os tipos de peças exijam até 15 tipos de

máquinas para a execução de operações. Para cada tipo de máquina, o contador i deve iniciar a contagem de 1 até 60, sendo que $J=60$. No fim, tem-se a soma de todos os tempos requisitados pelas peças por cada tipo de máquina m . Em seguida, deve-se dividir o resultado da soma (exigidos em cada máquina m) pelo tempo disponível da máquina tipo m .

Este procedimento também pode ser implementado em uma linguagem de programação para obtenção das réplicas de cada tipo de máquina.

3.5 POSSÍVEIS FORMATOS DOS LOCAIS NO CHÃO DE FÁBRICA E GERAÇÃO DE UMA MATRIZ DE DISTÂNCIAS DE-PARA PARA CADA FORMATO

A soma das réplicas de todos tipos de máquinas define também os locais onde as máquinas serão alocadas. Deve-se, agora, definir como será a configuração dos locais no chão de fábrica. O procedimento para a geração de todos os possíveis formatos está apresentado na Figura 3.5, onde x é o número de linhas e y é o número de colunas dos locais no chão de fábrica.

Basicamente, o procedimento identifica se a soma das réplicas de todos os tipos de máquinas é par ou ímpar. Se a soma for par, quer dizer que é possível obter os formatos dos locais em formato quadrado ou em formato retangular. Se a soma for ímpar, também é possível obter os locais em formato quadrado ou em formato retangular, desde que x e y sejam iguais. Caso a soma for ímpar, tem-se o problema da formação do vão vazio, conforme será discutido mais adiante.

Enfim, considerando que a soma das réplicas de todos os tipos de máquinas seja par, ou a soma das réplicas de todos os tipos de máquinas seja ímpar (com x e y iguais), o procedimento inicia fazendo a divisão da soma das réplicas pela variável i , sendo i começando por 1. Se o resto da divisão for zero, significa que existe um formato retangular. Então x armazena o número de linhas da configuração correspondente a i e y armazena o resultado da divisão da soma das réplicas pela x , que é o número de colunas da configuração dos locais. A cada *loop* no procedimento, soma-se 1 à variável i até que ela seja igual à soma das réplicas de cada tipo de máquina. Com isso, tem-se todos os formatos possíveis para os locais formados no chão de fábrica.

A partir deste momento, também podem ser geradas as matrizes de distâncias De-Para para cada formato dos locais no chão de fábrica, usando também um procedimento, como mostrado na Figura 3.6. Cada matriz de distância obtida contém valores de distância entre os locais (onde as máquinas serão alocadas) para apenas um único formato. Assim, se houverem cinco formatos possíveis, significa que existem cinco matrizes de distância De-Para.

Figura 3.5 - Procedimento para a obtenção de todos os possíveis formatos de chão de fábrica (Fonte: autor)

Figura 3.6 - Procedimento para a obtenção das distâncias entre os locais (De-Para) (Fonte: autor)

O procedimento apresentado na Figura 3.6 é relativamente complexo de ser explicado, por lidar com diversos contadores. Uma maneira simples de explicá-lo é que este procedimento inicialmente faz

a leitura de x e de y obtidos no procedimento anterior, por contador. Depois disso, para cada formato, definido por x e por y, inicia-se a criação das variáveis kk e ll. As variáveis kk e ll são responsáveis por definir a localidade da matriz De-Para onde será armazenada a distância entre o local da posição ll do chão de fábrica e o local da posição kk do chão de fábrica.

Para obter o resultado de cada distância que compõe a matriz de distância De-Para de kk e ll, deve-se usar outros contadores denominados i,ii,j e jj. Cada distância da matriz De-Para de kk e ll é, então, obtida através de $|i-ii|+|j-jj|$.

3.6 GERAÇÃO DOS ARRANJOS FÍSICOS

Após a realização do dimensionamento do número de réplicas de cada tipo de máquina, pode-se então gerar amostras para cada tipo de arranjo físico. No caso deste trabalho de pesquisa, os arranjos físicos a serem gerados para fins de comparação de desempenho são: arranjo físico funcional e arranjo físico distribuído. O objetivo deste tópico é, então, gerar apenas amostras de cada tipo de arranjo físico, sem se preocupar ainda com a forma correta de alocação das máquinas nos locais.

3.6.1 Arranjo físico funcional

A característica principal deste arranjo físico é a alocação de máquinas do mesmo tipo em uma única seção ou departamento. A Figura 3.7 apresenta um procedimento capaz de gerar um arranjo físico funcional, onde M representa o total de tipos de máquinas, N_m representa o número de réplicas de cada tipo de máquina denominada m e $\sum_{m=1}^M N_m$ representa a soma das réplicas de todos os tipos de máquinas do arranjo físico.

A ideia do procedimento ilustrado na Figura 3.7 é inicialmente solicitar que o usuário informe o número de tipos de máquinas. Com essa informação, é possível criar um contador para ler as réplicas de cada tipo de máquina até que este contador atinja o valor informado pelo usuário.

Em seguida, o procedimento define aleatoriamente qual tipo de máquina a ser alocada cujo número de réplicas do tipo de máquina selecionado seja diferente de zero. Caso for zero, deve-se escolher novamente outro tipo de máquina. Enfim, caso as réplicas do tipo selecionado de máquina seja diferente de zero, faz-se a alocação na primeira casa do vetor Layout[i]. Devido à alocação de uma máquina, reduz-se então uma réplica da máquina selecionada. A alocação deste tipo de máquina deve continuar até que as réplicas sejam iguais a zero. Naturalmente, as casas do vetor Layout[i] estarão sendo incrementadas na medida em que ocorre a alocação das máquinas em cada casa. Seleciona-se

o segundo tipo de máquina e assim sucessivamente até que as réplicas de todos os tipos de máquinas sejam iguais a zero.

Figura 3.7 - Procedimento para a geração do arranjo físico funcional (Fonte: autor)

3.6.2 Arranjo físico aleatoriamente distribuído

Como o próprio nome já sugere, este é obtido aleatoriamente, ou seja, as máquinas são selecionadas sem uma ordem especial até que não reste mais nenhuma máquina. A Figura 3.8 apresenta um procedimento para a obtenção deste tipo de arranjo físico.

O procedimento ilustrado na Figura 3.8 solicita que o usuário insira o número de tipos de máquinas, denominado de M . Depois disso, cria-se uma variável m e outra variável N_m , sendo que m é contador que varia de 1 até M e N_m para armazenar as réplicas de cada tipo de máquina da variável m . Estas réplicas podem ser tanto inseridas manualmente quanto lidas de um arquivo externo. A soma das

réplicas de todos os tipos de máquinas $\sum_{m=1}^M N_m$ define “total máquinas”.

Figura 3.8 - Procedimento para a obtenção do arranjo físico aleatoriamente distribuído (Fonte: autor)

Dando continuidade ao procedimento, este então seleciona um dos tipos de máquinas de M . A máquina escolhida, se possuir réplica diferente de zero, é selecionada para ocupar a casa “1” de “total máquinas”. Se a máquina escolhida não possuir réplica, então outro tipo de máquina é selecionada novamente. A máquina escolhida, então, tem a sua réplica reduzida em uma unidade. O segundo tipo de máquina agora é escolhido aleatoriamente para ocupar a casa “2” e assim sucessivamente. Este procedimento só termina quando a variável “total máquinas” se iguala a $\sum_{m=1}^M N_m$. As posições representam as diversas localidades do chão de fábrica, onde as máquinas serão alocadas.

3.6.3 Arranjo físico parcialmente distribuído

Benjaafar & Sheikhzadeh (2000) apresentam dois graus de desagregação de máquinas do mesmo tipo. Quando é de primeiro grau, significa que são formados no chão de fábrica dois grupos de mesmo tipo de máquina. A mesma análise serve para o de segundo grau (formação de três grupos em três localidades diferentes do mesmo tipo de máquina). Em comparação ao arranjo físico aleatoriamente distribuído, o arranjo físico parcialmente distribuído é mais complexo de ser representado em fluxograma. Assim, os passos descritos abaixo irão auxiliar na obtenção de um arranjo físico parcialmente distribuído.

Passo 1: Leia os tipos de máquinas (M) e réplicas de cada uma delas (N_m);

Passo 2: Escolha aleatoriamente uma máquina. Entre também com o número de grupos desejados, por exemplo ($k=2$ ou $k=3$). Divida N_m por k . Arredonde o resultado da divisão caso seja um número decimal. Fazendo isso, é possível conhecer quantas réplicas da mesma máquina em cada grupo.

Passo 3) Selecione os grupos de máquinas obtidos no passo 2 e aloque-os como se fosse na horizontal. É necessário fazer com que o código de programação garanta que dois grupos ou três grupos de mesmo tipo de máquina não sejam selecionados sucessivamente. Caso isso ocorra, será obtido um arranjo físico funcional e não um arranjo físico parcialmente distribuído.

3.6.4 Arranjo físico maximamente distribuído

A geração deste tipo de arranjo físico pode ser conseguida usando-se Minmax, proposto por Francis & White (1974), que é uma função-objetivo que minimiza a maior distância entre máquinas distintas. Este cálculo pode ser representado por um procedimento, e este, quando usado, permite calcular o grau de distribuição de cada arranjo físico gerado aleatoriamente. Grau de distribuição é um parâmetro adaptado de Minmax que avalia o nível de distribuição de máquinas. O arranjo físico que apresentar o menor grau de distribuição é também chamado de arranjo físico maximamente distribuído. Assim, se tiver uma amostra de tamanho n , deve-se usar o fluxograma apresentado n vezes.

O procedimento citado pode ser melhor entendido se separado em duas partes, sendo a primeira parte responsável pelo armazenamento das posições das réplicas de cada tipo de máquina (Figura 3.9). Já a segunda (Figura 3.10), continuação da parte 1, mostra como obter o valor do grau de distribuição, onde a variável “Distância[k]” recebe inicialmente um valor suficientemente grande. Este segundo identifica todas as menores distâncias das réplicas de cada tipo de máquina obtidas na parte 1. O valor da soma de todas estas menores distâncias de cada máquina dividido pelo número de tipos de máquinas define o grau de distribuição para o arranjo físico gerado. Novas populações (amostras) de arranjo físico podem ser geradas usando algoritmo genético. Realizando todas estas etapas para n (muito grande) de amostras, o menor deles é chamado de arranjo físico maximamente distribuído.

Até agora tanto os locais, o número de tipos de máquinas e suas respectivas réplicas são conhecidos. Também os arranjos físicos (em formato horizontal) foram gerados. Antes de fazer a alocação em si, é preciso compreender alguns detalhes. Ainda com relação ao Tópico 3.5, se houvessem 23 máquinas (número ímpar), seriam necessários 23 locais. Pela lógica, nenhuma empresa adotaria um conjunto enfileirado de 23 máquinas, mas sim, uma construção retangular. Sendo assim, deve-se somar uma unidade ao total de máquinas para que o formato retangular possa ser obtido. Portanto, com 24 locais, as possibilidades são 2x12, 4x6, 8x3, 24x1, 1x24, 3x8, 6x4, 12x2. Por causa da soma de uma unidade, sempre irá existir um vão vazio, que é denominado *dummy department* na literatura.

Figura 3.9 - Procedimento para o cálculo do grau de distribuição para um arranjo físico maximamente distribuído - parte 1 (Fonte: autor)

Figura 3.10 - Procedimento para o cálculo do grau de distribuição para um arranjo físico maximamente distribuído - parte 2 (Fonte: autor)

3.7 VÃO VAZIO (*DUMMY DEPARTMENT*)

Dentre os trabalhos sobre arranjo físico distribuído, apenas Benjaafar & Sheikhzadeh (2000) comentam sobre o vão vazio, porém não chegam a lidar com esta questão. Isso porque em sua formulação matemática QAP, uma das restrições é que os locais sejam iguais ao número de máquinas, portanto, ignorando a possibilidade de vão vazio.

A Figura 3.11 mostra uma situação em que existem seis locais/espacos e cinco tipos de máquinas (a numeração representa o tipo de máquina). Além das seis posições que o vão vazio pode ocupar, existe também a possibilidade de que as próprias máquinas troquem de posição entre si. É necessário avaliar o desempenho em cada situação e identificar em que posição o vão vazio deve estar e qual máquina deve ocupá-la. Assim, se houverem n locais e $n-1$ máquinas, ao todo existem $n \cdot (n-1)!$ possibilidades que é o próprio $n!$. Em resumo, sob o ponto de vista matemático, vão vazio não deixa de ser um tipo de máquina.

Uma outra forma de representação é deixar todas as máquinas na mesma linha sem vínculo com a configuração do arranjo físico. Por exemplo, no primeiro arranjo da Figura 3.11, a representação seria

“123450”, no segundo seria “123405”, e assim por diante, onde o vão vazio é representado pelo número “0”.

Figura 3.11 - Possíveis posições do vão vazio (Fonte: autor)

Dependendo dos tipos de arranjos físicos abordados neste trabalho de pesquisa, não se pode inserir um vão vazio “0” em qualquer posição porque isso tiraria a característica do próprio arranjo físico. Por exemplo, no arranjo físico funcional e no arranjo físico parcialmente distribuído, a característica principal é a existência de agrupamento de mesmos tipos de máquinas. Quando ocorre a inserção deste valor nulo em qualquer local, pode resultar na separação dos mesmos tipos de máquinas.

A Figura 3.12 mostra dois exemplos de arranjos físicos funcionais com dois tipos de máquinas. Foi escolhido o arranjo físico funcional para demonstração porque é mais fácil de visualizar. Mas, conceitualmente, a forma de geração é também a mesma para o arranjo físico parcialmente distribuído. O primeiro exemplo da Figura 3.12 mostra a primeira maneira de inserção do vão vazio “0”, que acontece após identificar os grupos de mesmos tipos de máquinas. O segundo exemplo da Figura 3.12 mostra a segunda maneira de inserção do vão vazio, que é feita aleatoriamente. Nota-se que no primeiro exemplo, o arranjo físico obtido ainda continua sendo arranjo físico funcional, porém já no segundo, uma das réplicas da máquina 2 está separada de seu grupo e, portanto, o arranjo físico obtido já não é denominado de arranjo físico funcional.

Já nos casos do arranjo físico aleatoriamente distribuído e do arranjo físico maximamente distribuído, o vão vazio pode estar em qualquer local, sem restrições como acontecia no arranjo físico funcional e no arranjo físico parcialmente distribuído.

Figura 3.12 - Exemplos de arranjo físico funcional com separação ou não de máquinas do mesmo tipo inserindo o vão vazio “0” (Fonte: autor)

É possível definir um procedimento para a inserção do vão vazio “0” para cada arranjo físico analisado neste trabalho (funcional, aleatoriamente distribuído, parcialmente distribuído e maximamente distribuído). O procedimento ilustrado na Figura 3.13 define as possíveis posições do vão vazio para o arranjo físico aleatoriamente distribuído e para o arranjo físico maximamente distribuído. Pode-se também definir as posições do vão vazio para o arranjo físico funcional e para o arranjo físico parcialmente distribuído, como mostrado na Figura 3.14.

Figura 3.13 - Procedimento para a geração das diversas posições do vão vazio para o arranjo físico aleatoriamente distribuído e para o arranjo físico maximamente distribuído (Fonte: autor)

- Se este “novo contador” estiver em uma casa diferente da primeira casa e diferente da casa “contador+1” do “novo_layout”, então basta ler as máquinas do “layout[i]” da primeira casa até a casa “novo contador-1” e copiá-las da primeira casa até “novo contador-1” do “novo_layout”. Preencher a casa “novo contador” do “novo_layout” com o valor nulo. Em seguida, ler as máquinas do “layout[i]” partindo da casa “novo contador” até “contador” e copiá-las da casa “novo contador+1” até “contador+1” do “novo_layout”. O número de vezes em que deve ser realizado o *loop* é exatamente “contador-1”.
- Se o “novo contador” estiver na última casa do vetor “novo_layout” basta ler as máquinas da casa 1 até “contador” do “layout[i]” e copiá-las da primeira casa até “contador” do “novo_layout”. Preencher a última casa do “novo_layout” com o valor nulo.

O procedimento ilustrado na Figura 3.14 faz inicialmente a leitura dos tipos de máquinas do vetor “layout[i]” do arranjo físico funcional e do arranjo físico parcialmente distribuído. Ao mesmo tempo, o “contador” faz a contabilização das casas deste “layout[i]”. Toda vez que existir uma mudança no tipo de máquina, então cria-se um “marcador”, e este marcador recebe o valor da casa onde ocorreu a mudança no tipo de máquina. Como são várias transições por causa dos diferentes tipos de máquinas, tem-se então vários “marcadores”. Cria-se um vetor “novo_layout” com “contador+1” casas. Cria-se também um “novo contador” para percorrer o “novo_layout”.

Para preencher as máquinas o arranjo físico denominado “novo_layout”, existem também três situações.

- Se este “novo contador” estiver na primeira casa do “novo_layout”, então basta ler as máquinas do “layout[i]” iniciando pela casa 1 até “contador” e copiá-las da casa 2 até “contador+1” do “novo_layout”; Preencher a primeira casa do “novo_layout” com o valor nulo;
- Se este “novo contador” estiver em uma casa diferente da primeira casa e diferente da casa “contador+1” do “novo_layout” e que seja igual a um dos “marcadores”, então basta ler as máquinas do “layout[i]” da primeira casa até a casa “novo contador-1” e copiá-las da primeira casa até “novo contador-1” do novo_layout”. Preencher a casa “novo contador” do “novo_layout” com o valor nulo. Em seguida, ler as máquinas do layout[i] partindo da casa “novo contador” até “contador” e copiá-las da casa “novo contador+1” até “contador+1” do “novo_layout”; O número de vezes em que deve ser realizado o *loop* é exatamente igual a “marcadores”.
- Se o “novo contador” estiver na última casa do vetor “novo_layout” basta ler as máquinas da casa 1 até “contador” do “layout[i]” e copiá-las da primeira casa até “contador” do “novo_layout”. Preencher a última casa do “novo_layout” com o valor nulo.

3.8 ALOCAÇÃO DE MÁQUINAS

Como o arranjo físico gerado está no formato horizontal, a etapa seguinte é alocar máquinas nos diversos locais disponíveis no chão de fábrica. A Figura 3.15 mostra a ordem de preenchimento (da esquerda para direita e de cima para baixo) e é justamente como a linguagem de programação normalmente trabalha. Considere que o arranjo físico gerado pelo código seja “455326165”.

A primeira máquina a ser alocada é a máquina do tipo 4 e, em seguida, uma outra máquina 5 e assim sucessivamente.

Figura 3.15 - Exemplo de alocação de máquinas (Fonte: autor)

No entanto, este método de alocação já não pode ser aplicado quando se trabalha com arranjo físico funcional e arranjo físico parcialmente distribuído.

Sagan (1991) propõe um método, denominado de preenchimento espacial, que pode ser usado para alocar corretamente as máquinas no chão de fábrica. Este método de preenchimento espacial pode ser obtido através do procedimento ilustrado na Figura 3.16.

O procedimento ilustrado na Figura 3.16 é implementado juntamente com o procedimento que gera os arranjos físicos funcionais e arranjos físicos parcialmente distribuídos.

Enfim, este procedimento deve ser informado sobre a configuração dos locais no chão de fábrica, dados em x e em y. Por exemplo, se x for 4 e y for 6, significa que a configuração formada pelos locais é quatro linhas e seis colunas. Em seguida, este procedimento verifica se a linha é par ou ímpar, considerando que a contagem é feita da parte superior para a parte inferior. Se for ímpar, como no caso da primeira linha e no caso da terceira linha, o preenchimento das máquinas nos locais segue normalmente, ou seja, da esquerda para a direita. Se for par, como no caso da segunda linha e no caso da quarta linha, então o preenchimento deve partir da direita para a esquerda. Se for novamente ímpar, então o preenchimento volta a ser da esquerda para a direita e assim sucessivamente.

Figura 3.16 - Procedimento para o preenchimento das máquinas nos locais (Fonte: adaptado de Sagan, 1991)

3.9 RECURSOS ADOTADOS NA FABRICAÇÃO DE PEÇAS

Além da caracterização da fábrica fictícia em estudo, existem também outros recursos complementares usados na fabricação de peças. Este tópico, então, define os recursos necessários (responsáveis tanto pelo processamento quanto pela movimentação de peças), importantes para a definição da dinâmica da fábrica fictícia em estudo.

Os recursos complementares podem ser classificados em dois grupos:

- Recursos usados na fabricação;
- Recursos de apoio à fabricação.

3.9.1 Recursos usados na fabricação

Este tópico descreve os recursos usados na fabricação das peças na fábrica fictícia adotada neste estudo. Como existem vários recursos usados na fabricação de peças, este, então, descreve apenas o que fazem parte da estrutura lógica do modelo de simulação.

Basicamente para este tópico, existem dois recursos: recurso máquina e recurso ferramenta. O recurso máquina é usado para realizar a operação nas peças. Em cada máquina, adota-se a hipótese de que ela é capaz de acoplar um conjunto de ferramentas. Assim, se uma determinada peça requisitar um determinado ferramental que já está acoplado neste conjunto de ferramentas, não é necessário realizar trocas. Adota-se, neste trabalho de pesquisa, que as máquinas podem acoplar até dois tipos de ferramentas ao mesmo tempo.

3.9.2 Recursos de apoio à fabricação

Existem também vários recursos de apoio à fabricação das peças. Os recursos de apoio à fabricação dependem do tipo de arranjo físico abordado (arranjo físico funcional ou arranjo físico distribuído):

- 1) Arranjo físico funcional: neste trabalho de pesquisa, os recursos de apoio à fabricação de peças no arranjo físico funcional são: caixas, empilhadeiras e funcionários.

- Caixas: as caixas são usadas para o armazenamento de peças (matéria-prima, semi-acabadas ou acabadas) e todas possuem a mesma capacidade de armazenamento. Uma caixa armazena apenas um único tipo de peça. Em qualquer parte do chão de fábrica fictícia, onde houver peças, então existe caixa as acompanhando.

Ainda com relação às caixas, adota-se a hipótese de que não existe restrição com relação à quantidade de caixas disponíveis para uso. Outra hipótese a considerar é que a quantidade demandada de cada tipo de peça, por cliente, não deve ultrapassar a capacidade da caixa;

- Empilhadeiras: para este trabalho de pesquisa, considera-se também a possibilidade de usar empilhadeiras para a movimentação de caixas. Todas as empilhadeiras são idênticas em termos de capacidade de carregamento, exercendo apenas funções diferentes. Existem três funções exercidas pelas empilhadeiras na fábrica fictícia: a primeira é usada na movimentação de matéria-prima até as máquinas. A segunda é usada na movimentação de peças semi-acabadas entre as máquinas e a terceira, é usada na movimentação de peças acabadas.

Adota-se a hipótese de que, quando uma determinada empilhadeira for designada para exercer uma das funções, ela simplesmente não realiza mais nenhuma outra.

Também adotam-se hipóteses para as quantidades de empilhadeiras usadas neste trabalho de pesquisa. Exceto as empilhadeiras que fazem a movimentação de peças semi-acabadas (no caso deste estudo considera-se apenas uma empilhadeira), não existem problemas de disponibilidade de empilhadeiras;

- Funcionários: é o último recurso de apoio à fabricação considerado neste trabalho de pesquisa. Para este estudo, pode-se separar os funcionários em quatro grupos:
 - Funcionários tipo 1: estes funcionários são responsáveis pelo carregamento de matéria-prima nas caixas. Esta atividade de carregamento de matéria-prima nas caixas acontece no L1.

Em seguida, estes funcionários utilizam-se de empilhadeiras de modo que as caixas (onde estão contidas as matérias-primas) possam ser movimentadas no chão de fábrica. Mas para que isso aconteça, deve-se fazer o carregamento dessas caixas nas empilhadeiras. Após o carregamento de caixa(s), estes funcionários conduzem estas empilhadeiras para que a movimentação das caixas no chão de fábrica possa acontecer com a finalidade de efetuar o descarregamento de caixas nas máquinas. Assim que todas as caixas da empilhadeira são descarregadas no chão de fábrica, ela retorna ao L1.

Para este trabalho de pesquisa, em particular, é importante definir um procedimento que realize a divisão da matéria-prima por carga da empilhadeira. É com este procedimento que define-se a frequência de partidas das empilhadeiras do L1 e é justamente o que ativa o modelo de simulação. Este procedimento pode ser visto na Figura 3.17. Além disso, este procedimento não espera que a demanda se acumule para formar o lote de fabricação (batching orders). Isso torna este estudo importante, porque esta estocasticidade é mantida e pode ser avaliada.

A ideia do procedimento da Figura 3.17 é unir as informações da demanda com o peso das peças. Como já se sabe o tamanho do lote de peças que devem ser fabricadas, é possível então conhecer os pesos das matérias-primas. Em virtude do recurso de apoio usado ser a empilhadeira, pode-se identificar e separar as matérias-primas das peças de acordo com a carga da empilhadeira;

Figura 3.17 - Procedimento para a separação da matéria-prima por carga da empilhadeira (Fonte: autor)

- Funcionários tipo 2: estes funcionários são operadores de máquinas, exercendo atividades tais como troca de ferramentais, fixação de peças nas máquinas etc. Cada funcionário fica responsável por apenas uma única máquina.

Após a chegada da caixa (onde as matérias-primas ou as peças semi-acabadas estão armazenadas), o funcionário tipo 2 inicialmente faz a checagem do ferramental exigido.

Após esta etapa de checagem de ferramental (e troca de ferramental, caso necessário), este funcionário faz a fixação da peça na máquina para que ela inicie a operação, e assim sucessivamente até que todo lote de peças da caixa tenham sido processadas na máquina. Para este estudo, considera-se que cada peça, assim que receber a operação na máquina, é posta novamente na caixa e aguarda o término das operações de todo o lote que devem ser realizadas na máquina;

- Funcionários tipo 3: Estes funcionários são responsáveis pela movimentação de caixas entre as máquinas de diferentes departamentos. Dentro dessas caixas estão contidas peças semi-acabadas. Para isso, eles usam empilhadeiras para auxiliá-los na atividade de movimentação de peças semi-acabadas. Sendo assim,

pode-se dizer que o número de funcionário tipo 3 alocado na tarefa é igual ao número da empilhadeira usada. E portanto, a posição do funcionário tipo 3 é igual à posição da empilhadeira que ele controla.

Essas posições iniciais dos transportadores (empilhadeiras) podem ser geradas usando um procedimento, conforme mostrado na Figura 3.18. Inicialmente, deve-se informar ao procedimento o número total de transportadores e, em seguida, o procedimento espalha estes nas máquinas até que não sobre nenhum transportador. Se a mesma posição for escolhida mais de uma vez para ser a posição inicial do transportador, então novo local é gerado evitando assim que mais de um transportador esteja na mesma posição. Ou seja, se a incógnita “armazena_valor[i]” for diferente de zero, significa que o transportador (bem como um funcionário tipo 3) já está alocado na máquina da posição i.

Com isso, todos os transportadores intermediários têm as suas respectivas posições iniciais definidas.

Figura 3.18 - Procedimento para a geração das posições iniciais dos funcionários tipo 3 (Fonte: autor)

Considerou-se nesse trabalho de pesquisa que existe apenas uma empilhadeira realizando esta tarefa de movimentação de peças semi-acabadas entre as máquinas de diferentes departamentos e esta movimenta apenas uma caixa por vez;

- Funcionários tipo 4: Estes funcionários são responsáveis pela coleta de peças acabadas, nas quais estão armazenadas em caixas. Estes são só solicitados quando existem peças acabadas formadas nas máquinas.

Adotou-se a hipótese de que enquanto a capacidade da empilhadeira ainda não tiver sido atingida, a posterior não deve sair do L2 para realizar a atividade de coleta. Enfim, após a coleta de peças acabadas, estes funcionários então retornam ao L2.

- 2) Arranjo físico distribuído: neste trabalho de pesquisa, os recursos de apoio à fabricação de peças no arranjo físico distribuído são similares ao do arranjo físico funcional, exceto a maneira de como as peças são movimentadas de uma máquina a outra.

A hipótese adotada neste trabalho de pesquisa é de que a transferência de peças é feita de forma unitária através de mecanismos que permitam o envio por gravidade, tais como hastes interligados entre pares de máquinas. Dessa forma, não existem empilhadeiras e também não existem caixas para serem usadas na movimentação de peças semi-acabadas entre as máquinas.

- Funcionários tipo 2: estes operadores ainda continuam realizando tarefas como: retirada de matéria-prima das caixas; checagem do ferramental a ser usado; fixação da matéria-prima/peça semi-acabada nas máquinas.

Em decorrência da transferência de peças semi-acabadas ser de forma unitária, muda-se a maneira de como os funcionários tipo 2 atuam. A diferença é que logo que terminar a operação na máquina, a peça semi-acabada já é enviada até a máquina seguinte sem a necessidade de esperas.

Apesar da maneira de como é feita a checagem de ferramental (juntamente com a possível troca deste) ser a mesma do que acontece no arranjo físico funcional, o que diferencia os dois tipos de arranjos físicos é o momento que ocorre a checagem. No arranjo físico funcional, a checagem só acontece após a chegada do lote de peças (matéria-prima ou peça semi-acabada). Já no arranjo físico distribuído, a checagem pode ser feita simultaneamente.

Para melhor compreender o momento de checagem do ferramental no arranjo físico distribuído, considere inicialmente que o lote de peças (matéria-prima) chega no chão de

fábrica. Este lote de peças deve se submeter às operações $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, ou seja, primeira operação na máquina tipo 1, a segunda na máquina tipo 2, a terceira na máquina tipo 3 e por fim, a última operação na máquina tipo 4. As máquinas que compõem a célula virtual para a fabricação desse lote podem então ser separadas (reservadas), compostas por uma máquina tipo 1, uma máquina tipo 2, uma máquina tipo 3 e uma máquina tipo 4.

A abordagem considerada neste trabalho de pesquisa é por trocas simultâneas. A ideia é que, como as máquinas já estão reservadas para realizar operações do mesmo lote (portanto, são responsáveis pela produção de apenas um único tipo de peça), então todas as máquinas já podem realizar as trocas de ferramentas ao mesmo tempo. Este tipo de abordagem permite reduções consideráveis de tempos de trocas de ferramentas. Assim, quando as peças vão de fato sendo processadas, não é necessário esperar pelo ajuste de ferramental da máquina posterior como acontecia no arranjo físico funcional.

Para melhor visualizar, a Figura 3.19 apresenta esta abordagem por trocas simultâneas. No momento que o lote chega na célula virtual, todos os operadores das máquinas que compõem a célula virtual já são informados sobre os tipos de ferramentais a serem usados. Então todos já fazem trocas necessárias ou possíveis ajustes. Note, então, que na célula virtual, não é simplesmente reservar máquinas, mas existe também uma filosofia de trabalho em equipe.

Após a descrição dos recursos adotados neste estudo para a fábrica fictícia, é possível identificar, basicamente, três tipos de filas de caixas:

- Fila aguardando processamento: este tipo de fila se forma a partir do momento em que houver existência de caixas contendo peças (matéria-prima ou peças semi-acabadas) e as máquinas requisitadas se encontram indisponíveis;
- Fila aguardando movimentação até outra máquina: este tipo de fila se forma a partir do momento em que houver existência de caixas contendo peças semi-acabadas e encontra a empilhadeira indisponível;
- Fila aguardando movimentação até o L2: este tipo de fila se forma a partir do momento em que houver existência de caixas contendo peças acabadas e as empilhadeiras se encontram indisponíveis. Estas caixas então aguardam a chegada do funcionário tipo 4, com a sua respectiva empilhadeira, para serem movimentadas ao L2.

Figura 3.19 - Abordagem por trocas simultâneas (Fonte: autor)

3.10 CRITÉRIOS DE SELEÇÃO DE CAIXAS

Após a definição dos principais recursos adotados na fabricação das peças da fábrica fictícia, bem como a descrição da forma de interação entre esses recursos, pode-se definir os critérios de seleção de peças e de máquinas mais adequados para este trabalho de pesquisa. Este tópico tem, portanto, a finalidade de discutir os critérios abordados pela literatura e selecionar os critérios mais realistas para este estudo.

Analisando criteriosamente os critérios de seleção de peças da literatura, verifica-se que para trabalhar com eles, basicamente, são necessárias algumas informações, tais como: tipos de peça em movimentação; o destino; o tipo de processamento que está sendo realizado e em qual máquina; o tempo que resta para concluir o processamento nas máquinas; a etapa de processamento em que cada peça se encontra etc. Baseadas nessas variáveis é que decisões são tomadas. Estas variáveis são, na realidade, atributos das peças que sofrem variações ao longo do tempo. Na prática, como estes

atributos mudam ao longo do tempo, também demandam maiores esforços para mantê-los atualizados. Naturalmente, para este estudo, os critérios de seleção de peças referem-se às caixas.

Os critérios usados com maior grau de realismo em relação ao que acontece na prática são FCFS (o que chega primeiro é atendido primeiro, conforme a ordem de chegada) e LCFS (o último que chega tem a prioridade de ser atendido). Para trabalhar com estes dois critérios, basta que o próprio operador separe a caixa em fila independente do estado em que o lote de peças se encontra.

Todavia, este trabalho também não irá considerar o critério LCFS. Um dos problemas eminentes ao considerar o critério LCFS é que o lote das peças da caixa que chegou posteriormente pode requisitar da máquina elevado tempo de operação (ainda na primeira operação). Isso pode atrasar os demais lotes que já estão nas últimas operações.

Devido aos problemas citados acima, adota-se apenas o critério FCFS, ou seja, o lote das peças de cada caixa é processado conforme a ordem de chegada na máquina. Naturalmente, o critério FCFS também não evita que caixas de peças sejam acumuladas em fila, mas pelo menos é mais prático porque é mais fácil ao funcionário de identificar a que chegou primeiro na máquina.

3.11 CRITÉRIOS DE SELEÇÃO DE MÁQUINAS

Este tópico tem por finalidade discutir os critérios de seleção de máquinas da literatura. Após essa discussão, define-se então os critérios mais adequados para este estudo.

Pode-se usar o critério de seleção de máquinas baseado nas trocas de ferramentais. Inicialmente deve-se identificar todas as réplicas de um determinado tipo de máquina, e escolher a que não necessita fazer trocas porque a ferramenta que uma determinada peça requisita já está acoplada a ela. Porém, de acordo com Ham et al. (1985), o tempo de troca de ferramentas não é um componente crítico do tempo total de produção da peça em um arranjo físico funcional, mas sim apenas o tempo de deslocamento e espera em fila.

Com relação ao recurso máquinas, as réplicas do mesmo tipo são idênticas na sua forma de processar, ou seja, nenhuma se diferencia na sua velocidade de processamento (portanto, desconsiderada como critério), porque o que poderia ser adotado como critério seria escolher a réplica da máquina com maior velocidade de processamento. Não será considerada também quebra de máquinas.

Outra forma seria escolher máquinas analisando as peças sendo nelas processadas com menor tempo de processamento restante. Caso a máquina esteja ocupada, porém sem fila, o funcionário que está usando uma empilhadeira precisaria ser informado, pelo sistema PCP (Planejamento e Controle da Produção), sobre o tempo de processamento restante da peça que já está na máquina. Esta é uma abordagem de difícil implementação, pois o sistema teria que obter constantemente informações

atualizadas de cada etapa de produção, por peça, em todo instante de tempo. Portanto, não serão levadas em consideração neste trabalho de pesquisa.

Os tópicos seguintes analisam o percurso realizado pela peça e a partir disso, finalmente, identificar os critérios mais adequados para o problema estudado. Os critérios a serem adotados, basicamente, estão fundamentados nas estratégias de entrega e estratégias de coleta de caixas de peças.

A seguir, as nomenclaturas dadas às empilhadeiras que realizam a movimentação de matéria-prima do L1 até as máquinas, movimentação entre as máquinas e movimentação das máquinas até o L2 serão transportadores iniciais, transportadores intermediários e transportadores finais, respectivamente.

3.11.1 Local 1 (L1) até as máquinas

Este tópico define os critérios de seleção de máquinas mais adequados para este estudo. Como existem três nomenclaturas dadas aos transportadores, inicia-se, então, com a definição dos critérios de seleção de máquinas para os transportadores iniciais.

Sabe-se que os transportadores iniciais chegam até às máquinas para o descarregamento de caixas. Estes só podem encontrar as réplicas da máquina requisitada pela peça em dois estados, ocupada ou livre. Por isso, separa-se em dois grupos.

a) No estado livre, existem duas possibilidades. A primeira é que a caixa (onde as matérias-primas estão armazenadas) é imediatamente conduzida até a réplica da máquina e assim o processamento das peças dessa caixa ocorre. A segunda é que, caso houver mais de duas réplicas da mesma máquina requisitada, é natural e mais realista escolher a mais próxima.

b) No estado ocupado, o transportador aguarda no L1 até que alguma réplica da máquina requisitada seja liberada e então acontece o critério de primeira máquina disponível. Por estar aguardando no L1, a formação da fila de caixas acontece não antes da máquina, mas sim no próprio L1. Ainda no estado ocupado, uma outra situação é quando o transportador mesmo assim conduz a caixa de matéria-prima e a leva até uma das réplicas da máquina requisitada (usando o critério de menor fila ou menor distância). Nestes dois últimos casos, a formação de fila de caixas ocorre antes das máquinas e não mais no L1.

Porém, tais critérios não são aplicáveis a um arranjo físico distribuído. Isso porque, após a divisão dos pedidos por carga do transportador no L1, a caixa (onde está armazenado o lote) só entra efetivamente no chão de fábrica quando de fato a célula virtual (conjunto de máquinas) que vai processar o lote de uma determinada peça estiver necessariamente livre.

Naturalmente, também existem critérios a serem usados para solicitar o transportador intermediário, pois as caixas (as que armazenam peças semi-acabadas) necessitam ser movimentadas até as máquinas seguintes. Neste caso, a melhor opção é que o transportador intermediário se direcione

até a maior fila de caixas de peças semi-acabadas. E se houver empate com relação ao tamanho das filas de caixas em várias máquinas, o transportador então se direciona à máquina mais próxima em relação a ele.

3.11.2 Entre as máquinas

Após a realização do carregamento de uma caixa, o transportador intermediário encontra a réplica da máquina requisitada em dois estados: livre ou ocupada, e a análise é similar ao do 3.11.1.

Já no arranjo físico distribuído, antes mesmo de iniciar o processamento, deve-se identificar as máquinas para formar células virtuais, e todas tem que estar necessariamente livres. Se, por exemplo, apenas uma máquina desta célula virtual estiver ainda ocupada, a hipótese adotada neste estudo é que esta célula virtual não deve ser formada e, portanto, o lote não deve entrar no chão de fábrica.

O ponto chave é que, como podem existir várias réplicas do mesmo tipo de máquina, existem também várias possibilidades de formar células virtuais. Neste trabalho de pesquisa, escolhe-se o conjunto de máquinas que farão parte do processamento da peça e que tenha soma das distâncias entre elas a menor possível.

Considere o exemplo do arranjo físico distribuído ilustrado na Figura 3.20. A parte inteira do número representa o tipo de máquina, e a parte decimal representa a réplica. Considere também, como exemplo, uma determinada peça cujo processamento seja $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. Pode-se notar, de acordo com a Figura 3.21, que existem várias rotas possíveis, o que significa dizer que existem várias maneiras de se formar células virtuais. Existe, porém, sempre uma rota que é a melhor dentre as rotas, exatamente aquela que tenha a menor soma total das distâncias entre as máquinas.

2.1	4.1	4.2
2.2	1.1	4.3
3.1	3.2	3.3

Figura 3.20 - Arranjo físico (Fonte: Gorgulho Júnior & Gonçalves Filho, 2007)

Figura 3.21 - Árvore de rotas (Fonte: Gorgulho Júnior & Gonçalves Filho, 2007)

3.11.3 Chamar o transportador final do Local 2 (L2)

Assim que termina a fabricação da peça (última operação), acontece o transporte até o L2. Para que a movimentação de caixa de peças aconteça, é necessário solicitar o transportador do L2 para que este se desloque até a máquina que o requisitou.

Adota-se neste trabalho de pesquisa um número ilimitado de transportadores e, portanto, não é necessário avaliar se o estado deste é livre ou ocupado, pois é garantia de que existam sempre transportadores livres. Mas existem regras a seguir para o chamamento do transportador final, como será comentado em 3.11.4.

De fato, é desejável que a peça, uma vez encerrada, já saia do chão de fábrica. Mas, considerando que o cliente espera receber o pedido todo, então não faz sentido criar um sistema produtivo só para permitir a saída de um tipo de peça por vez. Por isso, neste estudo, adotou-se a hipótese de que tanto no arranjo físico funcional quanto no arranjo físico distribuído, a coleta de caixas de peças acabadas é feita por empilhadeiras.

3.11.4 Máquinas até o Local 2 (L2)

E por fim, este tópico define os critérios para a coleta de caixas (as que armazenam peças acabadas) das máquinas e transportá-las ao L2. Os critérios só devem ser definidos quando mais de uma máquina possuir peças acabadas, porque se apenas uma máquina tiver peças acabadas, o transportador só tem que ir até ela, não tendo disputa das peças acabadas para chamar o transportador final. Se várias máquinas possuírem peças acabadas, então os transportadores irão seguir os seguintes critérios:

a) Escolher a máquina mais próxima ao local onde o transportador se encontra. Se várias máquinas estão a uma mesma distância em relação a ele, deve-se escolher a de maior fila;

b) Escolher a máquina que tenha maior fila formada de peças acabadas. Se várias máquinas tiverem tamanho das filas iguais, então deve-se escolher a mais próxima.

Em ambos os critérios, foi adotada a hipótese de que o transportador deve remover as caixas até que a sua capacidade de carga seja atingida, o que significa que poderão existir caixas restantes nas máquinas.

Enquanto a carga deste primeiro transportador usado não tenha atingido o seu limite de carregamento, o segundo transportador (apesar de estar disponível) não deve partir do L2. Quando esta condição estiver satisfeita (limite de carregamento atingido), antes mesmo do transportador carregado retornar ao L2, neste mesmo instante outro transportador já parte do L2.

Sabe-se que os critérios para os arranjos físicos funcionais e arranjos físicos distribuídos são distintos. Mas independente disso, em resumo, os critérios apresentados nos tópicos anteriores estão apresentados nas Figuras 3.22, 3.23, 3.24, 3.25 e 3.26, sendo A como caixa de peças em fila aguardando processamento, B como caixa de peças em fila aguardando movimentação até outra máquina, C como caixa de peças em fila aguardando movimentação até o L2, D como sendo distância do transportador até uma determinada máquina e R como sendo réplica da máquina requisitada.

- Critério da primeira máquina disponível: este critério, denominado a partir deste momento de MD, é usado quando o transportador está carregado e este encontra todas as máquinas ocupadas. Este aguarda no mesmo local onde fez o carregamento até que a primeira máquina ficar livre, independente da distância de percurso.

A Figura 3.22 apresenta um exemplo a ser adotado neste estudo. De acordo com o exemplo, na parte esquerda, o transportador está carregado com caixa(s) e este se encontra no L1 ou na máquina. Neste momento, as réplicas da máquina requisitada pela peça encontram-se todas ocupadas. Este, então, aguarda até que uma das réplicas seja liberada, conforme mostrada na parte direita da Figura 3.22. De acordo com o exemplo, a réplica da máquina liberada é R3 e, então, é a escolhida para o descarregamento da caixa.

Figura 3.23 - Exemplo da dinâmica de funcionamento usando critério baseado em menor distância (Fonte: autor)

- Critério da máquina de menor fila: este critério, denominado a partir deste momento de MF, é usado quando o transportador está carregado e encontra todas as réplicas da máquina requisitada ocupadas. Mesmo assim, o transportador efetua o encaminhamento da caixa de peças a uma das réplicas da máquina que tenha menor fila.

De acordo com o exemplo ilustrado na Figura 3.24, na parte esquerda, o transportador está carregado com caixa(s) e este se encontra também no L1 ou na máquina. Neste momento, todas as réplicas da máquina requisitada encontram-se ocupadas. Mesmo assim, a movimentação da caixa é realizada até a réplica da máquina requisitada baseada na que tiver menor fila de caixas. Então, de acordo com o exemplo ilustrado na parte direita da Figura 3.24, a réplica da máquina escolhida é R2 porque possui o menor A em relação à réplica 1 e em relação à réplica 3;

Figura 3.25 - Exemplo da dinâmica de funcionamento usando critério baseado em maior fila e menor distância (Fonte: autor)

- Critérios da máquina de menor distância e critério da máquina de maior fila: este conjunto de critérios, denominado a partir deste momento de MDMF, é usado quando o transportador final deve se deslocar até a máquina para a coleta de caixas (onde as peças acabadas estão armazenadas).

Como pode existir formação de filas de caixas em várias máquinas, então de acordo com o critério de menor distância, deve-se escolher a máquina cuja posição desta é a menor possível em relação ao transportador. Caso houver várias máquinas de mesma proximidade cuja fila seja de mesmo tamanho, deve-se escolher a máquina de maior fila. Este critério está representado na Figura 3.26.

Conforme o exemplo ilustrado na parte esquerda da Figura 3.26, existe fila de caixas na máquina 8 e na máquina 4, ou seja, C8 e C4 são diferentes de zero. Então o transportador deve realizar a coleta da caixa localizada na máquina 8, porque $C8 > C4$.

Figura 3.26 - Exemplo da dinâmica de funcionamento usando critério baseado em menor distância e maior fila (Fonte: autor)

Levando-se em consideração as possíveis combinações dos critérios quando as máquinas estão ocupadas, é possível construir tabelas para melhor organização dos critérios a fim de serem implementados e executados no modelo de simulação. Como os critérios são diferentes entre o funcional e distribuído, a Tabela 3.2 mostra primeiramente as possíveis combinações para o arranjo físico funcional e na Tabela 3.3 mostra as combinações para os arranjos físicos distribuídos.

Tabela 3.2 - Possíveis combinações dos critérios adotados neste trabalho (quando as máquinas estão ocupadas) para o funcional (Fonte: autor)

Transportador Inicial			Transportador Intermediário			Transportador Final	
Tipos de critério			Tipos de critério			Tipos de critério	
MD	-	-	MD	-	-	MFMD	-
MD	-	-	MD	-	-	-	MDMF
MD	-	-	-	MP	-	MFMD	-
MD	-	-	-	MP	-	-	MDMF
MD	-	-	-	-	MF	MFMD	-
MD	-	-	-	-	MF	-	MDMF
-	MP	-	MD	-	-	MFMD	-
-	MP	-	MD	-	-	-	MDMF
-	MP	-	-	MP	-	MFMD	-
-	MP	-	-	MP	-	-	MDMF
-	MP	-	-	-	MF	MFMD	-
-	MP	-	-	-	MF	-	MDMF
-	-	MF	MD	-	-	MFMD	-
-	-	MF	MD	-	-	-	MDMF
-	-	MF	-	MP	-	MFMD	-
-	-	MF	-	MP	-	-	MDMF
-	-	MF	-	-	MF	MFMD	-
-	-	MF	-	-	MF	-	MDMF

Tabela 3.3 - Possíveis combinações dos critérios adotados neste trabalho para o distribuído (Fonte: autor)

Transportador Final	
Tipos de critério	
MFMD	-
-	MDMF

4 DADOS DE ENTRADA AO MODELO DE SIMULAÇÃO

4.1 CARACTERIZAÇÃO DA FÁBRICA FICTÍCIA NO INSTANTE 0

Como a lógica do modelo de simulação já está definida, agora a etapa seguinte é definir o perfil da fábrica fictícia no instante 0 (momento em que se inicia o estudo comparativo). Uma vez que o perfil estiver caracterizado, pode-se submeter, então, esta fábrica às condições de um ambiente de instabilidade em um instante t , variando os fatores de experimentação no modelo de simulação.

Existem dois assuntos a serem abordados neste tópico no que diz respeito à definição do perfil da fábrica fictícia no instante 0, que é definir a quantidade total de máquinas e a forma de como realizar alocações destas, sendo este último o que distingue as características dos dois tipos de arranjos físicos que se deseja comparar (arranjo físico funcional e arranjo físico distribuído).

Pode-se também, com o auxílio dos procedimentos, gerar a demanda e separar a matéria-prima das peças por capacidade da empilhadeira neste instante 0, caso o objetivo seja avaliar o desempenho da fábrica fictícia no instante 0. Entretanto, como o objetivo é simular o desempenho da fábrica fictícia obtida no instante 0 sob as condições de instabilidade no instante t , o procedimento referente à separação da matéria-prima por capacidade da empilhadeira só deve ser usado no instante t .

Para este estudo, considera-se três níveis relacionados à quantidade total de máquinas, dados por 24 máquinas, 56 máquinas e 150 máquinas, os quais serão submetidos a um ambiente de instabilidade t .

Todas essas quantidades de máquinas (24 máquinas, 56 máquinas e 150 máquinas), na realidade, deveriam ser inicialmente geradas a partir das informações da demanda do instante 0. Em função da dificuldade de representação dos arranjos físicos superiores a uma quantidade de 50 máquinas, fica estabelecido que todo o detalhamento dos procedimentos propostos para a geração da quantidade total de máquinas da fábrica fictícia será ilustrado apenas para o primeiro nível, ou seja, 24 máquinas.

Para obter as 24 máquinas, é necessário definir hipóteses referentes à informação de demanda em um determinado instante de tempo 0:

- Demanda de até 100 tipos de peças;
- As operações das peças requisitadas exigem até 15 tipos de máquinas;
- O número de operações de cada peça não ultrapassa 8 operações;
- São necessárias até 9 tipos de ferramentas para cada tipo de máquina;
- 5 clientes enviaram os pedidos, e em cada pedido, existem cerca de 40 tipos de peças;
- Lotes de cada tipo de peça, por cliente, variando de 10 unidades até 30 unidades;
- Os tempos de cada operação variando de 30s a 150s.

Com o auxílio dos procedimentos propostos, pode-se definir a característica da fábrica fictícia no instante 0, ou seja, definir o primeiro nível. A aplicação dos procedimentos pode se vista nas etapas seguintes.

- Dimensionamento das réplicas de cada tipo de máquina: Ao inserir as informações da demanda (que chegam ao L1 na forma de pedidos) no código de programação do procedimento proposto, é possível, então, definir as réplicas de cada tipo de máquina. O código de programação deste procedimento tem a finalidade de vincular os tipos de operações, os lotes, os tempos de cada operação e de armazenar todas essas informações nas variáveis internas do código de programação.

Ao fim da execução deste código, este então, possui as informações da carga horária demandada das peças por cada tipo de máquina e então calcula as réplicas de cada uma delas.

A tela de execução é similar a da Figura 4.1. De acordo com esta figura, a quantidade total de máquinas está apresentada na última linha, que é 24 máquinas. Na antepenúltima linha estão 1 1 2 2 2 2 2 2 2 1 1 1 1 2 2, o que significa que a máquina tipo 1 possui uma réplica, a máquina tipo 2 possui uma réplica, a máquina tipo 3 possui duas réplicas etc. A soma de todas essas réplicas resulta justamente em 24 máquinas.

```

C:\DOCUMENTOS\shih\MEUSDO\1\TP\BIN\TURBO.EXE

A carga horaria da maquina 13 foi de:
18.414 horas
O numero de maquinas 13 necessarias e:
1

A carga horaria da maquina 14 foi de:
34.297 horas
O numero de maquinas 14 necessarias e:
2

A carga horaria da maquina 15 foi de:
29.608 horas
O numero de maquinas 15 necessarias e:
2

1 1 2 2 2 2 2 2 2 1 1 1 1 2 2
O total de maquinas necessarias e de:
24
  
```

Figura 4.1 - Tela de execução do código de programação para o dimensionamento das réplicas de cada tipo de máquina (Fonte: autor)

- Obtenção dos possíveis formatos formados pelos locais no chão de fábrica e distâncias De-Para: a partir da informação da quantidade total de máquinas, é possível definir o número de locais onde as máquinas deverão ser alocadas, pois o número de locais é justamente igual à quantidade total de máquinas. Além disso, é possível obter os possíveis formatos que os locais podem assumir.

A Figura 4.2 mostra a execução do código do procedimento proposto. A segunda linha, referente ao número de tipos de máquinas, é digitada manualmente, informando ao código de programação que ao todo são 15 tipos de máquinas, para que a leitura de 1 1 2 2 2 2 2 2 1 1 1 2 2 possa ser feita corretamente. Em seguida, o número 24 é obtido com a soma das réplicas de cada tipo de máquina, representando a quantidade total de máquinas necessárias para atender à demanda hipotética bem como o número dos respectivos locais necessários para a alocação dessas máquinas.

```

C:\DOCUMENTOS\shih\MEUSDO-1\TP\BIN\TURBO.EXE
Entre com o numero de Tipos de maquinas da fabrica <max. 100>
15
0 total de maquinas necessarias e: 24
0 numero de locais necessarios e: 24

2 1
2 2
2 3
3 1

```

Figura 4.2 - Tela de execução do código de programação para a obtenção dos possíveis formatos dos locais (Fonte: autor)

Pode-se notar, na parte inferior da Figura 4.2, que existem duas colunas de números. A primeira coluna (localizada na parte inferior da figura à esquerda) são os divisores primos do número 24. A segunda coluna apresenta o contador dos divisores primos localizados à sua esquerda.

Para simplificar, o procedimento inicia-se a partir do número dois até o número 24, sendo que na realidade deveria começar a partir do número 1. Neste intervalo, todos os números em que a divisão resulta em resto igual a zero é armazenado, o que significa necessariamente que existe um formato para o chão de fábrica, conforme mostrado no procedimento proposto. Como são três números “2”, então a coluna da direita soma-se até 3. Como é apenas um número “3” então a coluna da direita é apenas 1. Enfim, os possíveis formatos formados pelos locais são: $(2) \times (2^2 \times 3)$, $(2^2) \times (2 \times 3)$, $(2^2 \times 2) \times (3)$, ou seja, 2×12 , 4×6 , 8×3 . O código de programação faz automaticamente a inversão, obtendo também 12×2 , 6×4 , 3×8 e este também gera as configurações 1×24 e 24×1 .

Este mesmo código, então, já gera as distâncias da matriz De-Para de um local a outro. Como são oito possíveis formatos dos locais, são geradas oito matrizes de distâncias De-Para. A Figura 4.3 mostra a continuação da execução do código de programação ilustrada na Figura 4.2 e a matriz de distância ilustrada é justamente a de configuração 24x1.

```

C:\DOCUMENTS\1\shih\MEUSDO-1\TP\BIN\TURBO.EXE
01234567891011121314151617181920212223
1012345678910111213141516171819202122
210123456789101112131415161718192021
32101234567891011121314151617181920
4321012345678910111213141516171819
543210123456789101112131415161718
65432101234567891011121314151617
7654321012345678910111213141516
876543210123456789101112131415
98765432101234567891011121314
10987654321012345678910111213
11109876543210123456789101112
12111098765432101234567891011
13121110987654321012345678910
14131211109876543210123456789
151413121110987654321012345678
1615141312111098765432101234567
17161514131211109876543210123456
181716151413121110987654321012345
1918171615141312111098765432101234
20191817161514131211109876543210123
212019181716151413121110987654321012
2221201918171615141312111098765432101
23222120191817161514131211109876543210
  
```

Figura 4.3 - Tela de execução do código de programação para a obtenção das distâncias da matriz de distância De-Para de uma das configurações (Fonte: autor)

As linhas informam a posição de origem De, enquanto as colunas informam as posições de destino Para. A interseção entre a linha escolhida e a coluna escolhida é a distância entre essas duas posições. Por exemplo, com a interseção da linha 3 e a coluna 7, tem-se como resultado o número 4. O significado disso é que a distância do local associado à posição 3 (De) e do local associado à posição 7 (Para) é 4m.

A Figura 4.3 está sendo mostrada apenas para fins de visualização. Da maneira como está apresentada, não é possível saber se a distância possui um ou mais dígitos. Na realidade, com a execução do código de programação, este também imprime estas distâncias em um arquivo externo (em formato .txt) já com a separação correta.

Infelizmente, a tela de programação não permite ilustrar todas as oito matrizes ao mesmo tempo, mas estas estão localizadas na parte superior à da configuração 24x1. Para este trabalho de pesquisa, considera-se apenas a configuração 4x6 por ser o formato mais abordado na literatura.

- Distâncias do L1 até os locais e dos locais até o L2: como o formato dos locais onde as máquinas serão alocadas no chão da fábrica fictícia já está definido, pode-se obter as distâncias de cada local, onde as máquinas serão alocadas, aos dois locais externos (L1 e L2).

O primeiro passo é gerar as distâncias entre a entrada e os locais do chão de fábrica, e em seguida, gerar as distâncias entre os locais do chão de fábrica e a saída. Depois disso, soma-se um metro à todas as distâncias dessas duas matrizes para obter as distâncias do L1 aos locais e dos locais ao L2.

A Figura 4.4 mostra as distâncias após a execução do código de programação do procedimento proposto, referentes ao L1 até os locais (matriz localizada na parte superior) e referentes aos locais até o L2 (matriz localizada na parte inferior). Então, de acordo com a primeira matriz, por exemplo, o valor de distância 7.5 da posição 1 significa que a posição da máquina a ser alocada nesta posição até o L1 vale 7,5m, e assim por diante. Enquanto na segunda matriz, de maneira similar, o valor de distância 4.5 da posição 1 significa que a posição da máquina a ser alocada nesta posição até o L2 vale 4,5m e assim por diante.

```

C:\DOCUMENTOS\shih\MEUSDO\1\TP\BIN\TURBO.EXE
Entre com a configuracao em X
4
Entre com a configuracao em Y
6
A distancia a ser obtida eh do layout funcional? <1>Sim <2>Nao
2
7.5 6.5 5.5 5.5 6.5 7.5
6.5 5.5 4.5 4.5 5.5 6.5
5.5 4.5 3.5 3.5 4.5 5.5
4.5 3.5 2.5 2.5 3.5 4.5

4.5 3.5 2.5 2.5 3.5 4.5
5.5 4.5 3.5 3.5 4.5 5.5
6.5 5.5 4.5 4.5 5.5 6.5
7.5 6.5 5.5 5.5 6.5 7.5

```

Figura 4.4 - Tela de execução do código de programação para a obtenção das distâncias da entrada aos locais e dos locais à saída (Fonte: autor)

- Geração do arranjo físico funcional: como as réplicas de cada tipo de máquina já são conhecidas, pode-se então gerar os arranjos físicos focados neste estudo. A Figura 4.5 mostra a tela durante a execução do código de programação do procedimento proposto que gera as amostras do arranjo físico funcional.

Pode-se notar que as duas amostras ilustradas na Figura 4.5 apresentam dois blocos de informações em comum. O primeiro bloco refere-se à representação do arranjo físico funcional visto superiormente, onde os números representam os tipos de máquinas. Implicitamente, este

código já apresenta a maneira correta de alocação, usando a curva de preenchimento espacial. Já o segundo bloco é a forma de como o modelo de simulação faz a leitura.

```

C:\DOCUME~1\shih\MEUSDO~1\TP\BIN\TURBO.EXE
10 9 9 15 15 12
1 13 4 4 5 5 7 6 6 3 3 11 7 2 8 8 14 14 10 9 9 15 15 12

Aguarde um momento....

12 11 9 9 4 4
2 5 5 10 15 15
8 8 14 14 6 6
7 7 13 3 3 1

12 11 9 9 4 4 2 5 5 10 15 15 8 8 14 14 6 6 7 7 13 3 3 1

Aguarde um momento....

12 8 8 6 6 13
4 11 1 2 3 3
4 9 9 5 5 10
15 15 7 7 14 14

12 8 8 6 6 13 4 11 1 2 3 3 4 9 9 5 5 10 15 15 7 7 14 14

```

Figura 4.5 - Tela de execução do código de programação para a obtenção do arranjo físico funcional (Fonte: autor)

A Figura 4.6 mostra um exemplo em que 20 amostras do arranjo físico funcional são obtidas logo após o término da execução do código de programação.

```

C:\DOCUME~1\shih\MEUSDO~1\TP\BIN\TURBO.EXE
4
Entre com a configuracao em Y
6
Entre com os tipos de maquinas <max. 130>
15
1 13 4 4 5 5 7 6 6 3 3 11 7 2 8 8 14 14 10 9 9 15 15 12
12 11 9 9 4 4 2 5 5 10 15 15 8 8 14 14 6 6 7 7 13 3 3 1
12 8 8 6 6 13 4 11 1 2 3 3 4 9 9 5 5 10 15 15 7 7 14 14
6 6 2 15 15 5 9 9 14 14 1 5 11 7 7 8 8 13 10 4 4 3 3 12
14 14 9 9 5 5 3 3 4 4 7 7 11 15 15 10 2 12 13 8 8 6 6 1
9 9 4 4 6 6 7 12 11 10 14 14 7 13 3 3 5 5 2 8 8 1 15 15
11 5 5 12 10 9 4 4 1 14 14 9 3 3 7 7 8 8 15 15 6 6 2 13
3 3 1 2 13 15 8 8 12 4 4 15 9 9 14 14 10 11 7 7 5 5 6 6
12 11 6 6 10 8 13 1 2 15 15 8 4 4 3 3 7 7 5 5 9 9 14 14
1 10 5 5 7 7 6 6 13 2 4 4 8 8 14 14 11 12 15 15 3 3 9 9
1 5 5 14 14 8 9 9 7 7 11 8 3 3 15 15 10 13 2 4 4 6 6 12
10 1 9 9 12 3 8 8 13 14 14 3 6 6 11 5 5 2 4 4 15 15 7 7
6 6 2 8 8 9 7 3 3 4 4 9 7 15 15 12 14 14 5 5 10 13 1 11
13 3 3 6 6 7 10 1 15 15 2 7 5 5 11 8 8 12 4 4 14 14 9 9
10 1 8 8 13 14 11 15 15 4 4 14 5 5 6 6 7 7 2 3 3 12 9 9
6 6 14 14 4 4 1 12 9 9 15 15 10 8 8 5 5 3 11 13 7 7 2 3
8 8 7 7 11 1 9 10 13 4 4 2 9 6 6 14 14 12 15 15 5 5 3 3
4 4 13 7 7 11 9 2 12 1 15 15 9 10 14 14 5 5 8 8 3 3 6 6
14 14 7 7 3 3 6 6 5 5 10 2 9 9 13 12 11 4 15 15 8 8 1 4
8 8 15 15 5 5 12 2 1 11 7 7 3 3 14 14 4 4 6 6 13 10 9 9

```

Figura 4.6 - Término da execução do código de programação do procedimento proposto para a geração do arranjo físico funcional (Fonte: autor)

Considere a última amostra gerada pelo código de programação do procedimento referente ao arranjo físico funcional, conforme ilustrada na Figura 4.6. A primeira réplica da máquina tipo 8 está alocada na posição 1, a segunda réplica da máquina tipo 8 está alocada na posição 2, e assim por diante.

- Geração das posições iniciais dos transportadores intermediários: após a geração do arranjo físico funcional, pode-se usar o código de programação referente ao procedimento responsável pelo espalhamento dos transportadores intermediários nos locais do chão de fábrica para determinar, assim, as posições iniciais destes.

O procedimento proposto solicita ao usuário que insira uma determinada quantidade de transportadores intermediários a serem espalhados. Esta quantidade não deve ultrapassar o número 24 porque existem, ao todo, 24 possíveis locais que os transportadores intermediários podem ocupar (sendo um transportador intermediário para cada local).

Como o número de transportadores intermediários adotado neste estudo é igual a um, então ao executar o código de programação referente ao procedimento proposto, resulta conforme o que está ilustrado na Figura 4.7. O usuário insere o número 1 por ser apenas um transportador intermediário, e também insere o formato dos locais no chão de máquina (onde as máquinas serão alocadas). O código então inicialmente preenche todos os locais como sendo zero, artifício usado para indicar que não existe transportador intermediário alocado. Então este seleciona aleatoriamente um número de 1 a 24 e, no caso, selecionou a posição 13 (localizada na linha 3 e coluna 1). Como este código verificou que este tem um número zero no local, então a alocação do primeiro transportador intermediário é feita. Ao realizar a alocação, marca a posição 13 por 1 por ser o primeiro transportador intermediário alocado.

```

C:\Dev-Pas\Bin\ESTINI.exe
Entre com o numero de transportadores intermediarios
1
Entre com a configuracao em X
4
Entre com a configuracao em Y
6

000000
000000
100000
000000

```

Figura 4.7 - Tela de execução do código de programação responsável pela alocação dos transportadores intermediários nos locais (Fonte: autor)

Caso houvesse um segundo transportador intermediário a ser alocado, então o código escolhe um número aleatório de 1 a 24 e ao escolher, verifica se a posição é igual a zero. Se coincidentemente a posição 13 for novamente a escolhida, então nova posição deverá escolhida,

já que a posição 13 já será diferente de zero. Enfim, ao escolher a posição para o segundo transportador, a posição escolhida será preenchida por 2, por ser o segundo transportador intermediário alocado no local, e assim por diante.

- Geração do arranjo físico aleatoriamente distribuído: as amostras do arranjo físico aleatoriamente distribuído podem ser obtidas ao realizar a execução do código de programação do procedimento proposto. De acordo com o exemplo ilustrado na Figura 4.8, também foram geradas 20 amostras. De forma similar, os arranjos físicos aleatoriamente distribuídos estão no formato horizontal, que é justamente a maneira pela qual o modelo de simulação realiza a leitura dos tipos de máquinas. Pode-se notar que ao todo são 20 “linhas” representando, portanto, 20 amostras.

```

20
Digite 1 para obter uma vez da amostra, digite 2 para duas vezes da amostra,etc.
1
1 13 4 5 11 5 3 6 7 2 8 14 6 12 3 7 4 8 14 15 9 10 9 15
10 15 4 11 5 2 12 8 14 8 9 15 6 1 3 3 14 9 13 5 6 7 4 7
9 9 14 11 2 8 7 1 14 13 5 12 8 10 3 4 15 6 15 5 7 6 3 4
13 9 2 10 14 9 5 7 4 3 11 15 12 3 4 14 7 15 1 6 6 8 5 8
7 1 3 11 5 6 9 4 3 5 2 12 10 9 14 4 14 7 8 13 8 6 15 15
2 4 12 15 8 1 9 13 14 8 10 15 11 14 6 5 4 6 7 3 7 3 9 5
1 10 5 7 4 2 13 6 5 8 14 11 7 4 6 14 12 9 3 15 8 9 3 15
8 3 5 10 13 8 3 7 5 12 6 4 1 14 11 2 9 14 6 9 7 15 4 15
8 9 6 6 12 14 11 9 8 14 4 1 15 15 4 13 2 7 10 5 3 7 3 5
13 2 11 7 8 8 15 5 15 5 12 7 6 6 9 3 1 3 9 14 10 14 4 4
15 11 5 14 4 8 6 13 10 1 7 4 8 9 6 9 12 3 14 7 15 2 5 3
3 15 2 6 3 7 8 6 5 5 13 4 9 1 4 12 7 11 8 10 9 14 14 15
4 13 7 11 15 7 1 12 2 9 10 14 5 14 5 4 6 3 6 3 8 9 8 15
8 15 5 7 11 8 1 2 12 3 14 4 9 3 10 15 4 13 5 14 6 7 6 9
6 11 9 6 5 2 5 12 9 1 8 4 8 14 7 10 4 7 15 13 14 3 3 15
1 2 15 5 15 4 5 12 6 13 6 4 14 9 7 11 14 9 8 3 10 7 8 3
7 3 3 15 1 2 14 13 14 4 10 11 12 8 5 7 6 15 9 6 5 8 4 9
1 5 15 5 10 15 2 9 12 6 13 6 9 7 8 14 3 8 3 11 7 14 4 4
14 7 6 11 2 1 8 14 3 15 4 10 9 6 7 12 9 8 4 5 15 5 3 13
6 1 15 5 15 7 7 10 2 9 13 6 4 11 8 8 14 12 3 5 3 9 14 4

```

Figura 4.8 - Término da execução do código de programação do procedimento proposto para a geração do arranjo físico aleatoriamente distribuído (Fonte: autor)

- Geração do arranjo físico parcialmente distribuído: As amostras dos arranjos físicos parcialmente distribuídos podem ser obtidas usando-se também o código de programação do procedimento proposto. Para a obtenção das amostras, foi escolhido um grau de desagregação, ou seja, dois grupos do mesmo tipo de máquina espalhadas no chão de fábrica.

De acordo com as informações sobre as réplicas de cada tipo de máquina, obtidas a partir da realização do dimensionamento de máquinas (1 1 2 2 2 2 2 2 1 1 1 1 2 2), sabe-se então que a máquina tipo 1 possui uma réplica, a máquina tipo 2 possui uma réplica, a máquina tipo 3 possui duas réplicas e assim por diante. Como o grau de desagregação escolhido foi um (dois grupos do mesmo tipo de máquina), é de se esperar que as máquinas com duas réplicas tenham dois grupos do mesmo tipo de máquina, sendo que em cada grupo existe apenas uma réplica, como é o caso da máquina tipo 3, máquina tipo 4 etc. Isso pode ser constatado na Figura 4.9 para

cada arranjo físico parcialmente distribuído das 20 amostras geradas, as quais foram geradas para fins ilustrativos.

```

C:\DOCUME~1\shih\MEUSDO~1\TP\BIN\TURBO.EXE
- 20>
20
A desagregacao escolhida é 2 ou 3?
2
1 13 4 5 11 3 12 14 8 2 7 6 15 9 10 9 4 15 7 3 6 14 8 5
12 8 6 13 3 2 10 5 9 4 11 1 14 7 15 6 15 5 4 3 8 7 9 14
14 9 5 7 4 3 1 12 2 10 15 11 6 8 13 9 4 6 8 15 5 3 7 14
11 5 12 10 9 14 13 8 7 3 4 1 2 6 15 3 15 4 7 5 6 14 9 8
12 11 6 10 8 15 7 3 4 13 1 2 14 9 5 5 7 4 15 3 9 14 8 6
1 5 14 8 11 7 12 13 10 15 3 9 6 4 2 9 3 14 4 15 7 5 6 8
6 2 8 9 4 3 1 11 14 12 15 7 13 10 5 3 6 7 4 14 9 8 5 15
10 1 8 13 14 4 9 7 6 5 11 15 12 3 2 6 14 4 7 3 5 8 9 15
8 7 11 1 2 4 12 14 6 9 10 13 3 5 15 4 7 15 8 3 6 5 14 9
14 7 3 2 10 5 4 11 12 13 9 6 1 8 15 8 15 5 6 9 4 14 3 7
14 8 7 12 2 11 1 6 15 4 13 5 10 9 3 14 3 15 8 7 9 6 4 5
5 4 7 11 10 8 1 3 14 15 13 12 2 6 9 6 5 14 7 9 3 4 15 8
8 12 1 2 13 9 7 11 10 15 3 14 4 6 5 7 8 14 9 6 4 3 5 15
7 2 13 6 4 11 3 12 14 10 15 8 5 9 1 6 7 8 3 9 15 4 14 5
15 2 9 12 5 6 8 14 4 10 7 13 3 11 1 14 6 7 5 9 4 3 15 8
11 2 10 12 5 3 4 7 8 6 14 9 1 13 15 8 15 7 5 14 9 6 3 4
7 9 11 6 12 1 5 2 13 15 3 10 8 4 14 3 4 6 7 5 8 14 9 15
11 4 13 2 7 1 14 8 6 9 15 5 3 10 12 8 3 9 15 14 7 5 4 6
4 11 13 5 10 9 14 15 6 3 7 2 12 8 1 6 4 3 9 8 15 14 7 5
9 2 6 13 11 15 10 4 8 3 12 5 1 7 14 8 7 6 14 5 4 15 9 3
  
```

Figura 4.9 - Término da execução do código de programação do procedimento proposto para a geração do arranjo físico parcialmente distribuído (Fonte: autor)

- Geração do arranjo físico maximamente distribuído: Para a geração do arranjo físico maximamente distribuído, foi desenvolvido um algoritmo genético. O conceito por trás deste algoritmo bem como os principais operadores usados na literatura podem ser vistos no Apêndice 21. A tela de execução do código de programação referente ao procedimento proposto para a obtenção do arranjo físico maximamente distribuído pode ser vista na Figura 4.10.

```

C:\DOCUME~1\shih\MEUSDO~1\TP\BIN\TURBO.EXE
13814611101256712437414151585993      4.0333333333E+01
15459121310151137692651437148148      3.9433333333E+01
13314899411115614287371015124565      3.9900000000E+01
13361071491134551421546911288715      3.9666666667E+01
13310514428415656319779814151112      4.0466666667E+01
13814642971183431469101151557512      4.1233333333E+01
13314391011795811254461415681572      4.1633333333E+01
13895471098657326143151215141114      4.1633333333E+01
13312149316159106154857414857211      4.1233333333E+01
13885614241012147734113151595961      3.9200000000E+01
13361052512114141597848156497311      4.0333333333E+01
13361052512114141597869117158344      3.9866666667E+01
13814642101563148715712913559411      4.2566666667E+01

Maior Funcao Objetivo: 4.2566666667E+01  Media das Funcoes Objetivo: 40.577
Menor Funcao Objetivo: 3.9200000000E+01  Soma das funcoes Objetivo: 5.27500000
00E+02
N. cruzamentos :23290 N. mutacoes :32794 N. geracoes: 2227
  
```

Figura 4.10 - Tela de execução do código de programação para a obtenção do arranjo físico maximamente distribuído (Fonte: autor)

Este código de programação inicia-se com a geração de uma população semente, que neste estudo foi adotado como sendo tamanho 13, o que significa 13 amostras sementes de arranjos físicos. O arranjo físico escolhido para ser a semente é o arranjo físico aleatoriamente distribuído. Esta população agora será denominada de arranjos físicos pais.

Cada amostra do arranjo físico pai tem o seu grau de distribuição calculado, que é a função-objetivo. O grau de distribuição calcula o grau de dispersão de diferentes tipos de máquinas, oriundo da função obtida de Minmax. Em cada população de arranjos físicos pais existe um arranjo físico com maior valor do grau de distribuição (solução piorada) bem como um arranjo físico com menor grau de distribuição (solução melhorada).

Para iniciar o “cruzamento” das amostras, primeiramente deve-se selecionar duas amostras dos arranjos físicos pais através da aplicação da roleta-russa. Em seguida, realiza-se o “cruzamento” entre eles para a obtenção de duas amostras de arranjo físico, denominados de arranjos físicos filhos. Logo após o “cruzamento”, aplica-se a mutação nestes dois arranjos físicos filhos. Deve-se agora calcular os seus graus de distribuição.

Este procedimento de seleção por roleta russa para a escolha dos arranjos físicos pais para formar arranjos físicos filhos deve continuar até completar população de 13 amostras de arranjos físicos filhos.

A Figura 4.10 ilustra a tela de execução do código de programação em um determinado instante de tempo. No lado esquerdo da Figura 4.10 está representada uma população de tamanho 13, ou seja, 13 amostras de arranjos físicos filhos, sendo que cada linha representa uma amostra de arranjo físico filho. As amostras foram obtidas através dos cruzamentos dos arranjos físicos pais e foram aplicadas também mutações logo após os cruzamentos para a obtenção dos arranjos físicos filhos.

Pode-se notar que o formato de cada amostra de arranjo físico filho está representado na horizontal, maneira de como o modelo de simulação realiza a leitura da amostra de arranjo físico.

No lado direito, está apresentada a função objetivo de cada amostra de arranjo físico filho. Por exemplo, na primeira linha está ilustrado 4,033333333E+01 o que significa que o grau de distribuição referente à primeira amostra do arranjo físico filho é 40,33m.

O arranjo físico filho com o menor grau de distribuição, bem como o seu valor da função objetivo são armazenados na memória do computador para que possa fazer comparação com a população seguinte de arranjos físicos filhos. Após n gerações (ou seja, n populações) de arranjos físicos filhos, é possível obter então o arranjo físico com o menor grau de distribuição de todas as populações obtidas, que é também denominada de arranjo físico maximamente distribuído.

O número n de gerações escolhido neste estudo foi de 3000. Após completar as 3000 gerações, a tela de execução é similar ao que está apresentada na Figura 4.11. A amostra de

arranjo físico filho obtida após a execução das 3000 gerações com o menor grau de distribuição está ilustrada nesta Figura 4.11, que então agora é denominada de arranjo físico maximamente distribuído, cujo valor do cálculo do grau de distribuição é 33,9m. O formato do arranjo físico maximamente distribuído está ilustrado apenas para fins de visualização, não sendo importante para o modelo de simulação.

```

C:\DOCUMENTOS\shih\MEUSDO-1\TP\BIN\TURBO.EXE
O melhor layout distribuido eh:
39104561312111158714829315546714
-----Formato do layout-----
3910456
1312111158
7148293
15546714

O menor valor da FO do maximamente distribuido eh : 3.3900000000E+01
Tempo execucao : 0h 1min10s

```

Figura 4.11 - Término da execução do código de programação para a obtenção do arranjo físico maximamente distribuído (Fonte: autor)

Deve-se repetir os mesmos procedimentos para a obtenção dos arranjos físicos com 56 máquinas e 150 máquinas, a partir das hipóteses adotadas referentes à demanda no instante 0. Após isso, é que os três níveis do fator de experimentação quantidade total de máquinas estão definidos para serem submetidos ao modelo de simulação sob as condições de instabilidade no instante t .

4.2 INSTABILIDADE DE MERCADO NO INSTANTE t

Este tópico tem por finalidade definir os dados de entrada ao modelo de simulação para fins de comparação de desempenho entre o arranjo físico funcional e o arranjo físico distribuído no instante t . Neste trabalho de pesquisa, estes dados podem ser divididos em dois grupos: parâmetros variados e parâmetros fixos.

Como o objetivo é avaliar o desempenho de dois tipos de arranjos físicos em um ambiente de instabilidade, então os fatores de experimentação previamente selecionados fazem parte do grupo de parâmetros variados. Estes parâmetros, então, serão usados para fins de análise de sensibilidade.

- Parâmetros variados: os cinco fatores de experimentação adotados neste estudo são: número de operações, quantidade de cada tipo de peça demandada por cliente, tempo de cada operação, quantidade total de máquinas alocadas no chão de fábrica e tipo de arranjo físico.

Para definir os níveis de cada fator de experimentação, exceto o fator quantidade total de máquinas, deve-se primeiramente definir o instante em análise. Adotou-se o referido instante em

análise como sendo t . Adota-se também que em uma lista do pedido, enviado por um cliente, requisite necessariamente um número fixo de 40 tipos de peças.

- Primeiro fator: o número de operações refere-se ao número de diferentes ferramentais usados até que se resulte em peça acabada. Para este trabalho de pesquisa, adota-se dois níveis para este fator, sendo que no primeiro nível as peças podem receber até quatro operações (SURESH & SLOMP, 2005) e no segundo nível, as peças podem receber até oito operações (PORTO, 1990);
- Segundo fator: a quantidade de cada tipo de peça demandada por cliente refere-se à quantidade de cada tipo de peça da lista do pedido que o cliente enviou.

Por exemplo, se o local 1 (ou L1) recebesse uma lista no instante t , significa que um cliente enviou um pedido. Sabendo que em cada pedido estão definidos 40 tipos de peças, e considerando que a quantidade de cada tipo de peça seja adotada como sendo 20 unidades, tem-se uma quantidade total de peças de 800 unidades a produzir no instante t , ou seja, $1 \cdot 40 \cdot 20 = 800$ peças.

Para este estudo, adota-se dois níveis para o segundo fator, sendo o primeiro nível como sendo variando de 10 peças a 20 peças e o segundo fator como sendo variando de 10 peças a 50 peças. De acordo com a dinâmica da empresa fictícia adotada neste estudo, este segundo fator é justamente o tamanho do lote de fabricação;

- Terceiro fator: refere-se ao tempo em que a peça recebe uma única operação de uma ferramenta instalada na máquina. Para este fator, adota-se três níveis, sendo que no primeiro, o tempo de cada operação pode variar de 30s a 32s; no segundo, o tempo de cada operação pode variar de 30s a 60s; e, no terceiro nível, o tempo de cada operação pode variar de 30s a 180s;
- Quarto fator: refere-se à própria quantidade total de máquinas alocadas no chão de fábrica. Conforme já explicitado, este estudo leva em consideração três níveis, sendo o primeiro nível como sendo 24 máquinas, o segundo nível como sendo 56 máquinas e o terceiro nível como sendo 150 máquinas;
- Quinto fator: refere-se aos dois tipos de arranjos físicos abordados neste trabalho de pesquisa, que é o arranjo físico funcional e o arranjo físico distribuído, sendo que neste último, existe o arranjo físico aleatoriamente distribuído, o arranjo físico parcialmente distribuído e o arranjo físico maximamente distribuído. Dessa forma, existem ao todo quatro níveis para este fator.

Portanto, para este trabalho de pesquisa, tem-se cinco fatores, sendo que no primeiro fator tem-se dois níveis, no segundo tem-se dois níveis, no terceiro tem-se três níveis, no quarto tem-se três níveis e no quinto, tem-se quatro níveis. Em consequência disso, o número de experimentos a realizar

neste estudo pode ser obtido, que é $2*2*3*3*4=144$ experimentos. Foram obtidas 20 amostras do arranjo físico funcional, 20 amostras do arranjo físico aleatoriamente distribuído, 20 amostras do arranjo físico parcialmente distribuído e uma amostra do arranjo físico maximamente distribuído para serem submetidas à simulação. Com isso, tem-se ao todo $(2*2*3*3*20)+(2*2*3*3*20)+(2*2*3*3*20)+(2*2*3*3*1)=2196$ experimentos a serem realizados. Cada experimento a ser realizado no modelo de simulação consome cerca de 20 minutos. Isso quer dizer que a obtenção de todos os resultados leva em torno de 732 horas.

- Parâmetros fixos: todos os outros dados usados na execução do modelo de simulação, porém não usados para análise de sensibilidade, serão considerados fixos.
 - 100 tipos de peças;
 - Três turnos de 8 horas cada;
 - 15 tipos de máquinas, de acordo com Suresh & Slomp (2005);
 - 9 tipos de ferramentas;
 - Tempo de troca de ferramental igual a 1200s;
 - Empilhadeira de 600kg;
 - A velocidade da empilhadeira no estado descarregado varia de 1,2 a 1,7m/s (COLMANETTI, 2001); entretanto, este pode ser reduzido em função do carregamento. Neste estudo, considera-se um fator de redução de velocidade, dado pela expressão (23).

$$Fator_V = 0,75 + 0,25 \left(\frac{Carg\ máxima\ admissível - Carg\ a}{Carg\ a\ máxima\ admissível} \right) \quad (23)$$

Em que:

Carga máxima admissível é a carga máxima que a empilhadeira pode movimentar, em kg;

Carga é o carregamento em um determinado instante de tempo, em kg.

Assim que ocorre uma atualização da carga (através do carregamento ou descarregamento), novo valor do fator_V é obtido. Para obter a velocidade em cada instante de tempo, basta multiplicar este fator pela velocidade no estado descarregado;

- Os tempos de carregamento e descarregamento de caixas nas empilhadeiras foram considerados como sendo 5s;
- Os pesos das peças variam de muito leves (próximos a zero) até 10kg;
- A potência média de cada máquina é 15kw, de acordo com Máquinas & Metais (2005);

- O custo de energia por hora é R\$0,354838/kWh;
- Para a contabilização dos custos dos funcionários, é necessário separá-los em dois grupos. O primeiro grupo são os que recebem salário mensal, e segundo grupo são os que recebem por hora de trabalho.
 - Por mês: no caso da empresa estudada, como já se sabe a quantidade precisa dos funcionários tipo 2 (pois são iguais à quantidade total de máquinas), então o custo é obtido somando-se o salário mensal de cada um. Adotou-se também que existe apenas um único funcionário tipo 3. Enfim, considerando que o salário mensal seja de R\$1200,00, então o custo de mão de obra é conhecido;
 - Por hora: quanto aos demais funcionários, não é possível conhecer um número preciso destes. Entretanto, pode-se fazer uma estimativa. Por exemplo, se existe movimentação do transportador inicial, quer dizer que algum funcionário está em ação. De uma forma geral, esta ação pode ser contabilizada (em tempo) e, então, é possível obter o custo destes. Para este estudo, adota-se R\$5,00/h.
- O custo médio da matéria-prima foi considerado como sendo, em média, R\$2,00/peça;
- O custo de manutenção de estoque no instante t foi considerado como sendo, em média, R\$0,5/peça;
- Existe também um custo associado à compra de ferramenta. Adota-se neste estudo que a troca da ferramenta desgastada por uma ferramenta nova acontece a cada 120 minutos de uso. Este período de 120 minutos também é chamado de vida útil da ferramenta. O custo para adquirir (comprar) um ferramental novo é, em média, R\$15,00;
- Adota-se um custo médio para a aquisição de uma nova máquina de R\$35000,00, com base nas informações contidas em Máquinas & Metais (2005). Este custo não é contabilizado diretamente no custo de produção, mas é usado para calcular a depreciação. A vida útil adotada é de 5 anos. Considera-se que após os 5 anos, a carga horária usada é de 15000h;

Não será considerado o custo associado à manutenção das máquinas;
- Desconsiderou-se o custo associado ao uso de caixas, pois tratam-se de custos fixos, ou seja, já foram adquiridos e independente de quanto produzir de peças, o custo será o mesmo. Desconsidera-se também a manutenção das caixas;
- As empilhadeiras são alugadas, com um custo de aluguel de R\$3000/mês. Quanto à manutenção, o locador é responsável pela manutenção, sem ter um custo extra para a empresa;

- O custo de combustível é obtido da seguinte forma: Consumo * R\$/litro, onde consumo = 3km/litro e custo do combustível = R\$1,398/litro;
- Quanto aos custos relacionados aos lubrificantes, filtros e graxas, adotou-se que a soma desses custos seja igual a 15% do custo do combustível.

4.3 DESCRIÇÃO DA DINÂMICA DA SIMULAÇÃO DO MODELO

Este tópico tem por finalidade relacionar a dinâmica do funcionamento da empresa fictícia estudada com as ações que devem ser tomadas pelo modelo de simulação para a leitura e para a atualização das informações contidas nos arquivos externos, de modo que após o término da simulação, possa assim obter os resultados do tempo e custo de consolidação do pedido. A descrição mais detalhada da modelagem pode ser vista no Apêndice 1.

Basicamente, ao longo da simulação, o modelo de simulação precisa efetuar a leitura de cinco arquivos externos. Dependendo das condições de instabilidade do instante t no qual os arranjos físicos serão submetidos, estes arquivos também se alteram. Por isso, as informações contidas nos arquivos externos, conforme apresentados nos apêndices 3, 4 e 15, são de apenas uma situação em que peças requisitadas pela demanda apresentam os seguintes perfis: 1-4 operações, variação no tempo de cada operação variando de 30s a 32s e variação do tamanho do lote de cada tipo de peça variando de 10-20.

O primeiro arquivo, denominado de ORDERS.txt, contém as informações relacionadas aos tipos de peças a serem fabricadas bem como as suas respectivas quantidades. As informações contidas neste arquivo podem ser vistas no Apêndice 4.

O segundo arquivo externo é denominado de IN_CPN.sml (arquivo onde estão contidas informações dos arranjos físicos gerados, das réplicas de cada tipo de máquina, da matriz de distâncias De-Para etc.). De forma similar ao ORDERS.txt, este arquivo também varia conforme o instante t analisado. Enfim, um exemplo deste arquivo com o perfil de peças pode ser visto no Apêndice 15.

O terceiro arquivo, denominado de CrFunc.txt, armazena as informações relacionadas ao número de amostras do arranjo físico funcional e às 18 possíveis combinações dos critérios que foram adotadas neste trabalho de pesquisa. As informações contidas neste arquivo podem ser vistas no Apêndice 6.

O quarto arquivo, denominado de CrMax.txt, armazena as informações sobre o número de amostras dos arranjos físicos distribuídos a serem submetidas ao modelo de simulação, que também podem ser vistas no Apêndice 6.

E o quinto arquivo, denominado de TranspInterInitFunc.txt armazena as informações das posições iniciais do transportador intermediário para o arranjo físico funcional.

Por tratar-se de dois tipos de arranjos físicos abordados neste trabalho de pesquisa, a descrição da dinâmica da fábrica fictícia estudada com as ações a serem tomadas pelo modelo de simulação,

então, será dividida em duas partes, iniciando pelo arranjo físico funcional e em seguida, o arranjo físico distribuído. A descrição será feita apenas para o primeiro lote da peça, desde o momento que este é movimentado do L1 até ser movimentado ao L2, porque as ações a serem tomadas pelo modelo de simulação se repetem para os demais lotes. Naturalmente, à medida que a descrição para este primeiro lote vai-se avançando, os demais lotes ocorrem concomitantemente.

1) Arranjo físico funcional:

Local 1 até as máquinas: a primeira informação acessada pelo modelo de simulação é o arquivo IN_CPN.sml. Para que as informações deste arquivo possam ser acessadas, o modelo de simulação ativa a função denominada de `fun DefTransIniQtd()`. As informações contidas nesta função definem o número de partidas dos transportadores iniciais do L1 para a entrega da matéria-prima. Caso não houvesse essa função, o modelo de simulação então “entenderia” que não existem mais peças a serem produzidas e a simulação seria encerrada.

Pode-se notar, nesta função, que existe um número variando de 1 até 6, o que significa o número de partidas dos transportadores iniciais do L1. Caso fossem utilizados transportadores iniciais 15 vezes (independente de ser o mesmo transportador inicial usado), então a função encerraria em `1` (15,"Yes");` e não por `1` (15,"Yes")++` ;.

```
fun DefTransIniQtd() =  
1` (1,"Yes")++  
1` (2,"Yes")++  
1` (3,"Yes")++  
1` (4,"Yes")++  
1` (5,"Yes")++  
1` (6,"Yes");
```

Feito isso, o modelo de simulação também deve efetuar a leitura do local onde está sendo efetuada a partida dos transportadores iniciais, porque é com esta informação que o modelo de simulação pode acessar outra função, onde estão contidas as informações das distâncias de cada local do chão de fábrica em relação ao L1 para fins de contabilização do tempo de deslocamento.

A informação sobre o local de partida dos transportadores iniciais pode ser acessada pelo modelo de simulação ativando-se a função denominada `fun DefTranspLocationIni()`, onde `Local(1)` representa L1.

```
fun DefTranspLocationIni() =  
1` ((Transp,1),Local(1))++  
1` ((Transp,2),Local(1))++  
1` ((Transp,3),Local(1))++
```

```
1`((Transp,4),Local(1))++  
1`((Transp,5),Local(1))++  
1`((Transp,6),Local(1));
```

As informações contidas nesta função, então, informam ao modelo de simulação de que o primeiro transportador inicial usado parte do L1, o segundo transportador usado também parte do L1, e assim por diante.

Como o primeiro transportador inicial deve iniciar a entrega da matéria-prima, o modelo de simulação, então, deve acessar as informações contidas no arquivo ORDERS.txt sobre o tipo de peça. É com esta informação referente ao tipo de peça que pode-se conhecer o tipo de máquina e, conseqüentemente, o local que este deve alcançar. Ao acessar o arquivo ORDERS.txt, tem-se a seguinte informação:

```
1'(1,8,1,1,10,7470,53540)++
```

O significado de cada informação contida no arquivo ORDERS.txt pode ser visto no Apêndice 4. Mas a informação mais importante do arquivo ORDERS.txt para o momento é o tipo de peça, que neste caso é a peça tipo 1.

Ao realizar a leitura do tipo de peça, o modelo de simulação ativa a função denominada fun DefMachPiece (orinfo,pc) para realizar a leitura do arquivo IN_CPN.sml. Ao ativar esta função, este realiza uma busca na lista de informações contidas nesta função até que a peça lida coincida com o que foi lido no ORDERS.txt. Por se tratar da peça tipo 1, o modelo de simulação imediatamente encontra pc=1 (equivale a peça tipo 1) e, então, este faz a leitura da máquina que realizará a primeira operação, que neste caso é a máquina tipo 15.

```
fun DefMachPiece(orinfo,pc) =  
if pc = 1 then (orinfo,Mach(15))  
else if pc = 2 then (orinfo,Mach(10))  
else if pc = 3 then (orinfo,Mach(3))
```

Ao efetuar a leitura sobre o tipo de máquina que executará a primeira operação da peça tipo 1, que é a máquina tipo 15, pode-se então identificar as réplicas (se houver) das máquinas tipo 15.

Para isso, o modelo de simulação ativa a função denominada de DefLayoutIn(laytIn), a qual está localizada neste mesmo arquivo IN_CPN.sml. As informações contidas nesta função são as 20 amostras dos arranjos físicos funcionais, conforme mostrada na Figura 4.12.

Pode-se notar também na Figura 4.12 que esta função apresenta uma estrutura do tipo fun-let-in-if-end.

```

fun DefLayoutIn(laytn) =
let
  val LMList1 = [1,13,4,4,5,5,7,6,6,3,3,11,7,2,8,8,14,14,10,9,9,15,15,12];
  val LMList2 = [12,11,9,9,4,4,2,5,5,10,15,15,8,8,14,14,6,6,7,7,13,3,3,1];
  val LMList3 = [12,8,8,6,6,13,4,11,1,2,3,3,4,9,9,5,5,10,15,15,7,7,14,14];
  val LMList4 = [6,6,2,15,15,5,9,9,14,14,1,5,11,7,7,8,8,13,10,4,4,3,3,12];
  val LMList5 = [14,14,9,9,5,5,3,3,4,4,7,7,11,15,15,10,2,12,13,8,8,6,6,1];
  val LMList6 = [9,9,4,4,6,6,7,12,11,10,14,14,7,13,3,3,5,5,2,8,8,1,15,15];
  val LMList7 = [11,5,5,12,10,9,4,4,1,14,14,9,3,3,7,7,8,8,15,15,6,6,2,13];
  val LMList8 = [3,3,1,2,13,15,8,8,12,4,4,15,9,9,14,14,10,11,7,7,5,5,6,6];
  val LMList9 = [12,11,6,6,10,8,13,1,2,15,15,8,4,4,3,3,7,5,5,9,9,14,14];
  val LMList10 = [1,10,5,5,7,7,6,6,13,2,4,4,8,8,14,14,11,12,15,15,3,3,9,9];
  val LMList11 = [1,5,5,14,14,8,9,9,7,7,11,8,3,3,15,15,10,13,2,4,4,6,6,12];
  val LMList12 = [10,1,9,9,12,3,8,8,13,14,14,3,6,6,11,5,5,2,4,4,15,15,7,7];
  val LMList13 = [6,6,2,8,8,9,7,3,3,4,4,9,7,15,15,12,14,14,5,5,10,13,1,11];
  val LMList14 = [13,3,3,6,6,7,10,1,15,15,2,7,5,5,11,8,8,12,4,4,14,14,9,9];
  val LMList15 = [10,1,8,8,13,14,11,15,15,4,4,14,5,5,6,6,7,7,2,3,3,12,9,9];
  val LMList16 = [6,6,14,14,4,4,1,12,9,9,15,15,10,8,8,5,5,3,11,13,7,7,2,3];
  val LMList17 = [8,8,7,7,11,1,9,10,13,4,4,2,9,6,6,14,14,12,15,15,5,5,3,3];
  val LMList18 = [4,4,13,7,7,11,9,2,12,1,15,15,9,10,14,14,5,5,8,8,3,3,6,6];
  val LMList19 = [14,14,7,7,3,3,6,6,5,5,10,2,9,9,13,12,11,4,15,15,8,8,1,4];
  val LMList20 = [8,8,15,15,5,5,12,2,1,11,7,7,3,3,14,14,4,4,6,6,13,10,9,9];

in

  if laytn = 1 then LMList1 else
  if laytn = 2 then LMList2 else
  if laytn = 3 then LMList3 else
  if laytn = 4 then LMList4 else
  if laytn = 5 then LMList5 else
  if laytn = 6 then LMList6 else
  if laytn = 7 then LMList7 else
  if laytn = 8 then LMList8 else
  if laytn = 9 then LMList9 else
  if laytn = 10 then LMList10 else
  if laytn = 11 then LMList11 else
  if laytn = 12 then LMList12 else
  if laytn = 13 then LMList13 else
  if laytn = 14 then LMList14 else
  if laytn = 15 then LMList15 else
  if laytn = 16 then LMList16 else
  if laytn = 17 then LMList17 else
  if laytn = 18 then LMList18 else
  if laytn = 19 then LMList19 else
  LMList20

end;

```

Figura 4.12 - Informações gravadas no arquivo externo referentes ao número de amostras de arranjo físico funcional (Fonte: autor)

Esta estrutura informa o tipo de informação que o modelo de simulação deve acessar. A leitura das informações dessa estrutura é feita da seguinte forma: quando o lugar de

entrada satisfaz a condição do arco de entrada denominada *laytIn*, então o disparo ocorre e o arco de saída ativa a função denominada *DefLayoutIn(laytIn)*.

Como o número de amostras é 20, então o número que o arco de entrada *laytIn* pode assumir é de 1 a 20. A escolha da amostra é feita em sequência pelo modelo de simulação, basta que informe no arquivo externo (conforme ilustrado no Apêndice 6) o número de amostras 20, que o modelo de simulação realiza a leitura de 1 a 20 automaticamente e exatamente nesta sequência. Este arquivo é *CrFunc.txt*. Em função deste número, o arquivo informa ao modelo de simulação a amostra do arranjo físico a ser lida. Por exemplo, se a variável *laytIn* for 3, então *LMList3* e, dessa forma, a leitura é feita em *val LMList3*.

Considere que o modelo de simulação esteja fazendo a simulação para a primeira amostra (*LMList1*), então, estas duas réplicas da máquina tipo 15 estão localizadas nas posições 22 e 23.

Por ser o primeiro lote, então o modelo de simulação identifica a réplica 15 mais próxima do L1. Então este ativa a função denominada *fun DefMachDist(p)* localizado no arquivo *IN_CPN.sml*. Esta função apresenta as distâncias (em metros) de todos os locais em relação ao L1. Como as réplicas 15 estão localizadas nas posições 22 e 23, então a de menor distância em relação ao L1 é a réplica localizada na posição 22. Caso houvesse empate, o modelo de simulação escolhe a primeira máquina mais próxima à esquerda, porque este faz a leitura da lista de distância sempre da esquerda para a direita.

```
fun DefMachDist(p)=  
let
```

```
val MachListDist = [8,7,6,6,7,8,7,6,5,5,6,7,6,5,4,4,5,6,5,4,3,3,4,5];
```

Por ter tido a réplica escolhida da máquina 15 da posição 22 escolhida, esta lista deve ser atualizada para que outros transportadores iniciais, ou outros transportadores intermediários (conforme serão discutidos mais adiante) não escolham esta máquina, senão ocorre o travamento do modelo de simulação.

A maneira adotada neste trabalho de pesquisa de não disponibilizar a réplica da máquina é zerar a máquina localizada nesta posição, ficando:

```
fun DefLayoutIn(laytIn) =  
let
```

```
val LMList1 = [1,13,4,4,5,5,7,6,6,3,3,11,7,2,8,8,14,14,10,9,9,0,15,12];
```

Como já está escolhido o destino do transportador inicial, o modelo de simulação deve efetuar a leitura de outras informações referentes à peça tipo 1 do arquivo ORDERS.txt. A informação importante para o momento é efetuar a leitura de 53540, que significa 535,40kg. Ou seja, o transportador inicial, antes de efetuar o descarregamento no local de destino, está com um carregamento total de 535,40kg.

1'(1,8,1,1,10,7470,53540)++

Em seguida, o modelo de simulação ativa uma função do arquivo IN_CPN.sml, denominada de fun DefWeigthFactor(). Esta função calcula o fator de redução da velocidade da empilhadeira. Se o fator é 0,7, então a velocidade da empilhadeira é reduzida em 30% da velocidade no estado descarregado.

```
fun DefWeigthFactor() =  
let  
  val w = !wet;  
  val wr = real(w) / 1000.0  
  val TotalTranspW = real(!TranspW) / 1000.0  
in  
  0.75 + 0.25 * ((TotalTranspW - wr) / TotalTranspW)  
end;
```

O valor de 53540 do arquivo ORDERS.txt é armazenado pela variável w da função DefWeigthFactor(). Sabendo-se que a capacidade máxima de carregamento do transportador inicial adotada neste estudo é de 600kg, então a variável TotalTranspW equivale a 600000, e assim este o fator de redução do transportador inicial pode ser obtido.

Com a informação da velocidade do transportador inicial bem como a distância percorrida por este, o tempo de deslocamento pode ser calculado.

Logo que ocorre o descarregamento, o modelo de simulação ativa a função denominada de fun DefLayoutMachQueue() para a atualização das filas. Esta função é ativada pelo modelo de simulação para acessar as informações contidas nesta função, as quais estão armazenadas no arquivo externo denominado IN_CPN.sml. Note que inicialmente todas as posições contidas na função DefLayoutQueue() estão zeradas, isso porque o arquivo IN_CPN.sml foi gerado para iniciar a simulação e, portanto, ainda não existem filas formadas nas máquinas.

fun DefLayoutMachQueue() = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];

Na medida em que a simulação é executada, esta lista será constantemente atualizada.

Assim que ocorre o descarregamento da caixa, a posição 22 da função DefLayoutMachQueue() deveria incrementar em uma unidade o tamanho da fila, indicando que existe uma caixa na fila deixada pelo transportador inicial. Mas como a máquina 15 da posição 22 estava livre, então este incremento não ocorre. O modelo de simulação, então, armazena as informações sobre o tipo da peça que se encontra na caixa deixada pelo transportador inicial, o tamanho do lote, o peso do lote e o tipo da peça. Essas informações são oriundas do ORDERS.txt.

1'(1,8,1,1,10,7470,53540)++

Dando continuidade à entrega da matéria-prima, o modelo de simulação acessa novamente o arquivo ORDERS.txt e realiza a leitura comparativa entre os tipos de peças que o transportador inicial está movimentando com o contador do número de descarregamentos que devem ser realizados. Se ambos são diferentes, significa que o primeiro transportador inicial ainda deve continuar a descarregar as caixas nas máquinas.

De acordo com ORDERS.txt, 8 é diferente de 1:

1'(1,8,1,1,10,7470,53540)++

Esta comparação sobre a continuidade ou não da entrega da matéria-prima é realizada por uma função denominada de fun CheckOrderStatus(pc,oq,or2,or). Esta função está localizada no arquivo IN_CPN.sml, e as informações contidas nesta função são:

```
fun CheckOrderStatus(pc,oq,or2,or) =  
  if (pc = oq) andalso (or2 = or) then  
    1'(or, "OrderCompleted")  
    else 1'(or, "OrderNotCompleted");
```

Portanto, o descarregamento das caixas deve ser continuado a partir do local onde o transportador se encontra. O modelo de simulação acessa o arquivo ORDERS.txt e faz a leitura referente à próxima peça, que é a peça tipo 87:

1'(1,8,2,87,10,7610,46070)++

Como ocorreu o descarregamento da caixa de peso 74,7kg, onde estavam contidas as matérias-primas para a fabricação das peças tipo 1, a carga do transportador que era de 535,40kg passa a ser de 460,70, ou seja, $535,4 - 74,7 = 460,7$ kg ou 46070.

Ao efetuar a leitura do tipo de peça 87, o modelo de simulação faz a leitura do arquivo IN_CPN.sml usando a função `fun DefMachPiece(orinfo,pc)` para identificar qual o tipo de máquina responsável pela primeira operação da peça tipo 87, conforme ilustrado a seguir:

```
else if pc = 86 then (orinfo,Mach(13))  
else if pc = 87 then (orinfo,Mach(4))  
else if pc = 88 then (orinfo,Mach(6))
```

De acordo com as informações contidas no arquivo IN_CPN.sml, o tipo de máquina responsável pela primeira operação da peça tipo 87 é a máquina tipo 4.

Com esta informação sobre o tipo de máquina, o modelo de simulação então ativa a função `DefLayoutIn(laytIn)` para realizar a busca das réplicas da máquina tipo 4 da primeira amostra do arranjo físico funcional.

```
fun DefLayoutIn(laytIn) =  
let  
val LMList1 = [1,13,4,4,5,5,7,6,3,3,11,7,2,8,8,14,14,10,9,9,0,15,12];
```

Note que existem também duas réplicas da máquina tipo 4, localizadas na posição 3 e na posição 4. Deve-se agora identificar qual das réplicas que está mais próxima da posição onde se encontra o transportador inicial (que agora está na posição 22).

Considerando que as réplicas da máquina tipo 4 estejam todas livres, o modelo de simulação ativa agora uma nova função, denominada de `fun DefMachDistIn(loc,p)`, para a escolha da réplica da máquina tipo 4 mais próxima. As informações contidas nesta função se encontram no arquivo IN_CPN.sml e podem ser vistas na Figura 4.13, que é justamente as distâncias da Matriz De-Para.

A estrutura desta função está apresentada com a seguinte forma: `fun-let-val-in-if-end`. São blocos de informações, sendo a parte superior, denominada de `val`, e a parte inferior denominada de `if`. Se a incógnita `loc` é Local (4), por exemplo, então o arquivo informa ao modelo de simulação de que este deve percorrer a lista denominada `MachListDist` e atingir em `p-1`, ou seja, 4-1. Nesse sentido, o modelo de simulação é informado de que a posição do local (4) é referente à posição 3. Sendo assim, este então faz a leitura do “let” e realiza a busca até alcançar a linha referente à posição 3, que é `MachListDist3`.


```

fun DefMachDistIn(loc, p) =
let
  val MachListDist1=[0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7,3,4,5,6,7,8];
  val MachListDist2=[1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6,4,3,4,5,6,7];
  val MachListDist3=[2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5,5,4,3,4,5,6];
  val MachListDist4=[3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4,6,5,4,3,4,5];
  val MachListDist5=[4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3,7,6,5,4,3,4];
  val MachListDist6=[5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2,8,7,6,5,4,3];
  val MachListDist7=[1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7];
  val MachListDist8=[2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6];
  val MachListDist9=[3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5];
  val MachListDist10=[4,3,2,1,2,3,3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4];
  val MachListDist11=[5,4,3,2,1,2,4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3];
  val MachListDist12=[6,5,4,3,2,1,5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2];
  val MachListDist13=[2,3,4,5,6,7,1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6];
  val MachListDist14=[3,2,3,4,5,6,2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5];
  val MachListDist15=[4,3,2,3,4,5,3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4];
  val MachListDist16=[5,4,3,2,3,4,4,3,2,1,2,3,3,2,1,0,1,2,4,3,2,1,2,3];
  val MachListDist17=[6,5,4,3,2,3,5,4,3,2,1,2,4,3,2,1,0,1,5,4,3,2,1,2];
  val MachListDist18=[7,6,5,4,3,2,6,5,4,3,2,1,5,4,3,2,1,0,6,5,4,3,2,1];
  val MachListDist19=[3,4,5,6,7,8,2,3,4,5,6,7,1,2,3,4,5,6,0,1,2,3,4,5];
  val MachListDist20=[4,3,4,5,6,7,3,2,3,4,5,6,2,1,2,3,4,5,1,0,1,2,3,4];
  val MachListDist21=[5,4,3,4,5,6,4,3,2,3,4,5,3,2,1,2,3,4,2,1,0,1,2,3];
  val MachListDist22=[6,5,4,3,4,5,5,4,3,2,3,4,4,3,2,1,2,3,3,2,1,0,1,2];
  val MachListDist23=[7,6,5,4,3,4,6,5,4,3,2,3,5,4,3,2,1,2,4,3,2,1,0,1];
  val MachListDist24=[8,7,6,5,4,3,7,6,5,4,3,2,6,5,4,3,2,1,5,4,3,2,1,0];

in
  if loc = Local(2) then List.nth(MachListDist1,p-1) else
  if loc = Local(3) then List.nth(MachListDist2,p-1) else
  if loc = Local(4) then List.nth(MachListDist3,p-1) else
  if loc = Local(5) then List.nth(MachListDist4,p-1) else
  if loc = Local(6) then List.nth(MachListDist5,p-1) else
  if loc = Local(7) then List.nth(MachListDist6,p-1) else
  if loc = Local(8) then List.nth(MachListDist7,p-1) else
  if loc = Local(9) then List.nth(MachListDist8,p-1) else
  if loc = Local(10) then List.nth(MachListDist9,p-1) else
  if loc = Local(11) then List.nth(MachListDist10,p-1) else
  if loc = Local(12) then List.nth(MachListDist11,p-1) else
  if loc = Local(13) then List.nth(MachListDist12,p-1) else
  if loc = Local(14) then List.nth(MachListDist13,p-1) else
  if loc = Local(15) then List.nth(MachListDist14,p-1) else
  if loc = Local(16) then List.nth(MachListDist15,p-1) else
  if loc = Local(17) then List.nth(MachListDist16,p-1) else
  if loc = Local(18) then List.nth(MachListDist17,p-1) else
  if loc = Local(19) then List.nth(MachListDist18,p-1) else
  if loc = Local(20) then List.nth(MachListDist19,p-1) else
  if loc = Local(21) then List.nth(MachListDist20,p-1) else
  if loc = Local(22) then List.nth(MachListDist21,p-1) else
  if loc = Local(23) then List.nth(MachListDist22,p-1) else
  if loc = Local(24) then List.nth(MachListDist23,p-1) else
  List.nth(MachListDist24,p-1)

end;

```

Figura 4.13 - Informações gravadas no arquivo externo referentes à matriz de distância De-Para, cujo o formato dos locais é 4x6 (Fonte: autor)

A busca das distâncias na vertical é feita de forma similar. Se a incógnita loc é conhecida, então a incógnita p-1 também é conhecida. O modelo de simulação faz a criação de um contador para realizar a leitura após p-2 casas e, assim, a distância é obtida.

Este artifício, ou seja, p-1, foi adotado para que o modelo de simulação faça a distinção com os dois locais externos adotados neste trabalho de pesquisa, denominados de L1 e L2. Então, como o total de máquinas é 24, o L1 será denominado local (1), a posição 1 será denominada de local (2), ..., a posição 24 será denominada de local (25) e, por fim, o L2 será denominada de local (26).

O local (1) é inserido diretamente no modelo de simulação, já o local (26) é informado ao modelo de simulação através da ativação de uma função denominada de `fun DefTranspFinalLocation()`, a qual realiza a leitura no arquivo externo `IN_CPN.sml`.

fun DefTranspFinalLocation() = Local(26);

Em resumo, como o transportador inicial se encontra na posição 22 (então `Val MachListDist22`), e uma réplica da máquina tipo 4 se encontra na posição 3 e outra na posição 4, então o modelo de simulação efetua a leitura da distância da réplica localizada na posição 3 (que em relação à posição 22 é 4m) e efetua a leitura da réplica localizada na posição 4 (que em relação à posição 22 é 3m). Então, a de menor distância é a réplica da máquina tipo 4 localizada na posição 4.

A comparação das distâncias é feita através de uma outra função denominada de `fun CalcMinTranspDistIn(TranspDistListIn)`, localizada no arquivo `IN_CPN.sml`. As informações contidas nesta função podem ser vistas a seguir:

```
fun CalcMinTranspDistIn(TranspDistListIn) =  
let  
  fun aux(f,L1) =  
    if List.null(L1) then  
      f  
    else  
      if f > hd(L1) then  
        aux(hd(L1),tl(L1))  
      else  
        aux(f,tl(L1))  
in  
  aux(hd(TranspDistListIn),tl(TranspDistListIn))  
end;
```

Existe um problema eminente ao se trabalhar com arranjos físicos de grande porte. Quanto maior o tamanho do arranjo físico, maior a matriz de distância `De_Para`. Constata-se que, quando houver a necessidade de se gerar a matriz de distância `De_Para` para um arranjo físico com aproximadamente 180 máquinas, o arquivo em txt da matriz de Distância `De-Para` simplesmente é gerado de forma incompleta porque esta ultrapassa o limite de colunas permitido pelo arquivo txt. Isso quer dizer que torna-se mais complicado efetuar a simulação para arranjo físico com quantidades superiores a 180 máquinas.

Após a identificação da máquina tipo 4 de menor distância, o modelo de simulação ativa a função `fun DefLayoutIn(layoutIn)` e atualiza o número 4 da posição 4 para o número

0, indicando que a máquina tipo 4 da posição 4 não está disponível, conforme mostrado a seguir:

```
fun DefLayoutIn(layoutIn) =  
let  
    val LMList1 = [1,13,4,0,5,5,7,6,6,3,3,11,7,2,8,8,14,14,10,9,9,0,15,12];
```

Com isso, o modelo de simulação faz todo o procedimento descrito anteriormente para a contabilização do tempo ao se deslocar até a máquina tipo 4.

Considerando que o modelo de simulação esteja em pleno regime, pode acontecer de o transportador inicial encontrar as réplicas das máquinas requisitadas ocupadas (por exemplo, a máquina tipo 4):

```
fun DefLayoutIn(layoutIn) =  
let  
    val LMList1 = [1,13,0,0,5,5,7,6,6,3,3,11,7,2,8,8,14,14,10,9,9,0,0,12];
```

Ou todas as máquinas indisponíveis, como mostrada pela função a seguir:

```
fun DefLayoutIn(layoutIn) =  
let  
    val LMList1 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
```

Ao verificar que não existe disponibilidade da réplica da máquina, o modelo de simulação então pode ativar outras funções, dependendo do critério adotado. Se for a primeira máquina disponível, este transportador inicial simplesmente aguarda no local onde foi feito último descarregamento sem ativar nenhuma outra função.

Considerando que a máquina tipo 4 é o que está sendo requisitada pela peça:

```
fun DefLayoutIn(layoutIn) =  
let  
    val LMList1 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
```

Após um certo tempo houve a liberação de uma réplica da máquina tipo 4, então tem-se:

```
fun DefLayoutIn(layoutIn) =  
let  
    val LMList1 = [0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
```

Imediatamente o modelo de simulação ativa esta função `fun DefLayoutIn(laytIn)` fazendo com que esta máquina fique indisponível para outros transportadores, senão também ocorre o travamento do modelo de simulação.

Se o modelo de simulação estiver executando o critério de menor fila, este então ativa a função, denominada de `fun DefLayoutMachQueue()`.

Como o arquivo foi gerado para fins de simulação, então ainda não existem filas de caixas formadas nas máquinas. Por isso, de acordo com as informações contidas no arquivo `IN_CPN.sml` referentes à esta função, as filas de caixas que se formam antes das máquinas de cada posição são nulas, como pode ser visto a seguir:

`fun DefLayoutMachQueue() = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];`

Mas durante a simulação, pode-se ter:

`fun DefLayoutMachQueue() = [1,0,2,5,0,0,0,8,0,0,0,0,0,1,0,0,0,0,9,0,0];`

Se a máquina requisitada ainda for a máquina tipo 4 e se ela tiver várias réplicas, então o modelo de simulação ativa a função denominada de `fun CalcMinMachQIn(OrdMachInQ)` localizada no arquivo `IN_CPN.sml` para fins de comparação do tamanho da fila de caixas formadas nas réplicas da máquina tipo 4. As informações contidas nesta função no arquivo `IN_CPN.sml` podem ser vistas a seguir:

```
fun CalcMinMachQIn(OrdMachInQ) =
let
  fun aux(f,L1) =
    if List.null(L1) then
      f
    else
      if f > hd(L1) then
        aux(hd(L1),tl(L1))
      else
        aux(f,tl(L1))
  in
    aux(hd(OrdMachInQ),tl(OrdMachInQ))
end;
```

Ao realizar a comparação, o modelo de simulação escolhe a réplica da máquina tipo 4 localizada na posição 3, por apresentar a menor fila de caixas. Então o modelo de simulação ativa novamente a função `fun DefLayoutMachQueue()` e acrescenta-se uma unidade à fila (o que significa que a caixa transportada pelo transportador inicial é descarregada na fila), resultando em:

`fun DefLayoutMachQueue() = [1,0,3,5,0,0,0,8,0,0,0,0,0,1,0,0,0,0,9,0,0];`

Se o critério for a de menor distância em relação à última posição onde o transportador inicial fez o descarregamento, então o modelo de simulação ativa a função `fun DefMachDistIn(loc, p)` e faz a comparação entre as réplicas da máquina requisitada pela peça. A comparação também é feita através da função `fun CalcMinTranspDistIn(TranspDistListIn)`.

Como este critério de menor distância foi ativado em função das réplicas estarem todas ocupadas, então deve-se ativar novamente a função `fun DefLayoutMachQueue()` para a atualização do tamanho da fila formada na máquina requisitada pela peça.

Todo procedimento para a contabilização do tempo de deslocamento é feito similarmente ao que foi descrito anteriormente.

Os procedimentos descritos se repetem até que o modelo de simulação, ao efetuar a leitura do arquivo `ORDERS.txt`, verificar que o contador do número de tipos de peças é igual ao número de descarregamentos realizados. Quando esta condição é satisfeita, então este retorna ao L1, encerrando assim a atividade de entrega de matéria-prima para o primeiro transportador inicial.

O tempo de deslocamento de retorno do transportador inicial ao L1 não é contabilizado pelo modelo de simulação porque este já está descarregado.

Encerra-se assim a descrição da dinâmica bem como as ações tomadas pelo modelo de simulação para a entrega da matéria-prima das caixas nas máquinas.

- Nas máquinas: O modelo de simulação executa constantemente a função denominada de `fun DefLayoutIn(layoutIn)` para verificar se alguma posição é igual a zero. Se for igual a zero, significa que a máquina está ocupada, e se estiver ocupada, significa que alguma peça deve ser submetida à fabricação.

```
fun DefLayoutIn(layoutIn) =  
let  
val LMList1 = [1,13,0,0,5,5,7,6,6,3,3,11,7,2,8,8,14,14,10,9,9,0,0,12];
```

Para esta lista apresentada em `Val LMList1`, que é a primeira amostra do arranjo físico funcional, as máquinas localizadas na posição 3, posição 4, posição 22 e posição 23 estão ocupadas. Então para estas posições, o modelo de simulação faz a identificação do tipo de peça e o tamanho do lote da caixa que chegou primeiro (de acordo com FIFO), os quais já foram previamente armazenados pelo modelo de simulação logo que ocorreu o descarregamento da caixa pelo transportador inicial.

Após a identificação do tipo de peça, o modelo de simulação ativa a nova função, denominada de `fun DefMachToolListProd(ProdT,OPTlistOPTon)`. Esta função, uma vez

ativada, acessa as informações contidas no arquivo IN_CPN. Por exemplo, se a peça é do tipo 1 (e, portanto, deve submeter-se à fabricação na máquina tipo 15), o ferramental a ser usado é a ferramenta tipo 6, tal como mostrado abaixo:

```
fun DefMachToolListProd(ProdT,OPTlistOPTon) =  
let  
val MachToolListProd1 = [6];
```

Esta representação informa que a máquina tipo 15 deve usar o ferramental tipo 6 para a realizar a transformação da peça tipo 1. Agora, o operador desta máquina tipo 15 deve efetuar a checagem do ferramental para verificar se existe necessidade de realizar trocas de ferramentais. Esta checagem é realizada pela função denominada de fun DefMachToolListInit(). As informações contidas nesta função no arquivo IN_CPN.sml representam os ferramentais inicialmente instalados em cada local (ou máquina). Então, de acordo com as informações contidas nesta função, se a posição da máquina tipo 15 fosse a posição 22 (e portanto local(23)), por exemplo, então:

```
1` (Local(23),[1,6])++
```

Isso significa que, inicialmente, estão instaladas duas ferramentas na máquina tipo 15 da posição 22, que são a ferramenta tipo 1 e a ferramenta tipo 6. Neste caso, não é necessário fazer a troca da ferramenta. Note que estas ferramentas estão armazenadas em um vetor.

Neste mesmo instante, a função fun DefLayoutMachQueue() é ativada para a atualização do tamanho da fila que se forma antes da máquina, decrementando uma unidade, indicando assim que uma caixa é retirada da fila para fins de iniciar o processo de transformação. Note que o modelo de simulação decrementou uma caixa da posição 22.

```
fun DefLayoutMachQueue() = [1,0,3,5,0,0,0,0,8,0,0,0,0,0,1,0,0,0,0,0,8,0,0];
```

Se a máquina 15 escolhida estivesse na posição 23, então

```
1` (Local(24),[5,1])++
```

Neste caso, é necessário que se faça a troca da ferramenta. Se a troca da ferramenta tiver que ser realizada, então o modelo de simulação ativa a função fun SetupTime() do arquivo IN_CPN.sml:

```
fun SetupTime() = 1200;
```

A informação contida nesta função significa que o tempo de troca é de 1200s. Então inicialmente, tem-se:

fun DefMachToolListInit() =

1` (Local(24),[5,1])++

Agora passa a ser:

fun DefMachToolListInit() =

1` (Local(24),[6,1])++

O modelo de simulação sempre efetua a troca a partir do lado esquerdo, porque é assim que este faz a leitura e faz a atualização do vetor.

Enfim, após a identificação da necessidade de troca da ferramenta (e uma posterior troca de ferramenta), ativa-se a função denominada de *fun TimeProdOPT(ProdT,OPTlistOPTon)* para a leitura do tempo de apenas uma única operação. De acordo com as informações contidas nesta função do arquivo *IN_CPN.sml*, o tempo de cada operação varia de acordo com a distribuição uniforme de 30 a 30s.

fun TimeProdOPT(ProdT,OPTlistOPTon) = uniform(30.0,32.0);

Então ao multiplicar o tempo de cada operação com o tamanho do lote tem-se o tempo total de fabricação do lote da peça tipo 1.

- Chamar o transportador intermediário: após as peças terem sido submetidas ao processo de transformação, elas são novamente colocadas nas caixas. Por tratar-se do arranjo físico funcional, toda vez que o lote é processado, incrementa-se uma caixa na fila que se forma após a máquina. O modelo de simulação deve então criar uma lista para a atualização dessas filas. Se alguma posição desta lista for diferente de zero, significa que existe caixa de peças semi-acabadas que necessita ser movimentada de uma máquina à outra.

Por exemplo, considere que a lista do modelo de simulação apresente a seguinte informação:

[0,0,0,0,...]

Isso significa que não existem peças semi-acabadas que necessitam ser movimentadas de uma máquina à outra. Enfim, o modelo de simulação deve estar continuamente realizando a checagem até que alguma fila de caixas seja formada para que, assim, as caixas necessitem ser movimentadas pelo transportador intermediário de

um local ao outro. Tal lista, conforme ocorre a execução da simulação, pode-se ter uma situação tal como ilustrada a seguir:

[0,0,1,0,3,...]

Isso significa que existe uma caixa de fila formada após a máquina localizada na posição 3 e três caixas de fila formadas após a máquina localizadas na posição 5.

Como existem caixas formadas após as máquinas, conforme registradas pelo vetor, então o modelo de simulação efetua a leitura de um outro arquivo externo denominado de *TranspInterInitFunc.txt*. Neste arquivo estão contidas as informações sobre a posição em que se encontra o transportador intermediário. De acordo com as informações deste arquivo, o transportador intermediário está na posição 9, por isso Local (10):

1`(*Transp,Local(10)*)

Em seguida, o modelo de simulação ativa a função *fun DefMachDistIn(loc, p)* e realiza a busca por *MachListDist9*. Como existe fila formada nas máquinas localizadas na posição 3 e posição 5, o transportador intermediário escolhe a de maior fila, então a posição 5 é a escolhida. A função que executa a comparação entre os tamanhos das filas é *fun CalcMinMachQIn(OrdMachInQ)*, também localizada no arquivo *IN_CPN.sml*. As informações contidas nesta função podem ser vistas a seguir:

```
fun CalcMaxMachQIn(OrdMachInQ) =
let
  fun aux(f,L1) =
    if List.null(L1) then
      f
    else
      if f < hd(L1) then
        aux(hd(L1),tl(L1))
      else
        aux(f,tl(L1))
    in
      aux(hd(OrdMachInQ),tl(OrdMachInQ))
end;
```

Com a posição identificada referente ao de maior fila, ativa-se agora a função *fun DefMachDistIn(loc, p)*. O modelo de simulação realiza a leitura na quinta casa e assim obtém a distância de 3m para fins de contabilização do tempo de deslocamento. Como o transportador intermediário está ainda descarregado, não é necessário considerar fator de redução de velocidade.


```
fun DefMachDistIn(loc, p) =  
let
```

```
val MachListDist1=[0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7,3,4,5,6,7,8];  
val MachListDist2=[1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6,4,3,4,5,6,7]  
val MachListDist3=[2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5,5,4,3,4,5,6]  
val MachListDist4=[3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4,6,5,4,3,4,5]  
val MachListDist5=[4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3,7,6,5,4,3,4]  
val MachListDist6=[5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2,8,7,6,5,4,3]  
val MachListDist7=[1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7]  
val MachListDist8=[2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6]  
val MachListDist9=[3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5]
```

Ao chegar neste local, o transportador intermediário faz o carregamento de apenas uma caixa e, então, o modelo de simulação faz a atualização no tamanho da fila formada na posição de 3 para 2.

Atualiza-se agora a posição do transportador intermediário, de 1` (Transp,Local(10)) para 1` (Transp,Local(6)).

Caso houvesse empate entre os tamanhos das filas, então o modelo de simulação ativa primeiramente a função fun DefMachDistIn(loc, p) e em seguida ativa a função fun CalcMinTranspDistInter(TranspDistListInter) para fazer comparações das distâncias e assim obter a posição mais próxima de onde está localizado o transportador intermediário.

```
fun CalcMinTranspDistInter(TranspDistListInter) =  
let  
  fun aux(f,L2) =  
    if List.null(L2) then  
      f  
    else  
      if f > hd(L2) then  
        aux(hd(L2),tl(L2))  
      else  
        aux(f,tl(L2))  
  in  
    aux(hd(TranspDistListInter),tl(TranspDistListInter))  
end;
```

- Movimentação entre as máquinas: Considere que a peça carregada na posição 3 seja a peça tipo 38 e que acabou de ser submetida à primeira operação na máquina tipo 4. O modelo de simulação, após o carregamento, ativa a função denominada de fun DefMaqtipoProd (ProdT,OPTlistOPTon), localizada no arquivo externo denominado de IN_CPN.sml. De acordo com as informações contidas nesta função, a próxima operação deve ser executada na máquina tipo 10.

```
fun DefMaqTipoProd(ProdT,OPTlistOPTon) =  
let  
val MachListProd38 = [Mach(4),Mach(10),Mach(1),Mach(12)]
```

O procedimento adotado agora se repete, conforme descrito anteriormente, identificando as réplicas das máquinas tipo 10 da amostra do arranjo físico funcional que estejam livres e ao escolher uma réplica, deve-se zerar a posição da réplica escolhida. Deve-se também armazenar a posição da réplica escolhida, com a finalidade de obter a distância movimentada pela caixa (onde estão as peças semi-acabadas).

As informações sobre as posições das réplicas das máquinas agora são obtidas ativando-se a função `fun DefLayoutInter(laytInter)` para efetuar a leitura externa no arquivo `IN_CPN.sml`. As informações contidas na função `fun DefLayoutInter(laytInter)` em nada difere das que estão contidas na função `fun DefLayoutIn(laytIn)`, apenas criou-se duas funções por questões de modelagem. Assim, se o modelo estiver considerando o transportador intermediário, a função a ser ativada é a `fun DefLayoutInter(laytInter)`, caso contrário, se estiver considerando o transportador inicial, então `fun DefLayoutIn(laytIn)`.

A fila da posição escolhida deve ser incrementada em uma unidade por causa da entrega da caixa na fila pelo transportador intermediário.

Caso as réplicas estivessem todas ocupadas, então o modelo de simulação inicia a análise baseada na primeira máquina disponível ou na menor fila formada na máquina ou na de menor distância. Enfim, como o arranjo físico funcional possui apenas uma réplica da máquina tipo 10, que se encontra na posição 19, então a posição do transportador intermediário deve ser atualizado para `1` (Transp,Local(20))`.

- Máquinas até o L2: O modelo de simulação cria um vetor apenas para armazenar informações relacionadas ao tamanho das filas das caixas (onde estão armazenadas as peças acabadas). Existem algumas possibilidades ao longo da simulação. Se não existem filas de caixas, então este vetor simplesmente tem todas as suas posições zeradas, ou seja, as máquinas localizadas em cada posição no chão de fábrica não possuem nenhuma fila formada. Neste caso, o transportador final não deve partir do L2. Se pelo menos uma das posições for diferente de zero, então tem-se algumas possibilidades:

Se existe fila formada em apenas uma máquina, o transportador final só deve se encaminhar até ela e retornar ao L2. Por exemplo, considere que existe caixa (ou caixas) formada(s) na máquina localizada na posição 13. O modelo de simulação ativa a função `fun DefMachDistOut(p)` para obter a distância a ser percorrida pelo transportador final, saindo do L2 até alcançar a máquina localizada na posição 13.

De acordo com as informações desta função no arquivo IN_CPN.sml, a distância é de apenas 7m.

```
fun DefMachDistOut(p)=  
let
```

```
val MachListDist = [5,4,3,3,4,5,6,5,4,4,5,6,7,6,5,5,6,7,8,7,6,6,7,8];
```

Esta distância de 7m é usada para a contabilização do tempo de deslocamento do transportador final para a realização da coleta de caixa na máquina localizada na posição 13.

Se existirem filas de caixas formadas em mais de uma máquina, então deve-se definir os critérios de seleção ao modelo de simulação. Considere que em um determinado momento durante a simulação do modelo, a lista apresenta as seguintes informações:

```
[0,4,2,0,4,0,...]
```

Esta lista informa que a primeira posição não apresenta fila de caixas, a segunda posição apresenta 4 caixas em fila, a terceira posição apresenta 2 caixas em fila etc.

O modelo de simulação, então, faz a checagem e verifica que existem várias posições com filas. Se for adotado o critério de maior fila, o modelo de simulação ativa a função denominada de `fun CalcMaxMachQIn(OrdMachInQ)`, localizada no arquivo externo denominado IN_CPN.sml. As informações contidas nesta função têm a finalidade de efetuar a comparação dos tamanhos das filas das caixas formadas, onde estão contidas as peças acabadas, como se pode ver a seguir:

```
fun CalcMaxMachQIn(OrdMachInQ) =  
let  
  fun aux(f,L1) =  
    if List.null(L1) then  
      f  
    else  
      if f < hd(L1) then  
        aux(hd(L1),tl(L1))  
      else  
        aux(f,tl(L1))  
in  
  aux(hd(OrdMachInQ),tl(OrdMachInQ))  
end;
```

Ao efetuar a comparação, verifica-se que existem duas posições com as maiores filas, localizadas na posição 2 e na posição 5. Por causa do empate, o modelo de

simulação ativa a função denominada de `fun CalcMinTranspDistExit(TranspDistListExit)` para fins de comparação de distância e assim obter a posição de menor distância em relação ao L2

```
fun CalcMinTranspDistExit(TranspDistListExit) =
let
  fun aux(f,L4) =
    if List.null(L4) then
      f
    else
      if f > hd(L4) then
        aux(hd(L4),tl(L4))
      else
        aux(f,tl(L4))
  in
    aux(hd(TranspDistListExit),tl(TranspDistListExit))
end;
```

Se ainda persistir o empate, a escolha é feita da esquerda para a direita de `MachListDist = [5,4,3,3,4,5,6,5,4,4,5,6,7,6,5,5,6,7,8,7,6,6,7,8]` da função `fun DefMachDistOut(p)`, que é o procedimento padrão do modelo de simulação para a leitura de vetores.

De acordo com as informações desta função, houve empate novamente, então a escolhida é fila formada na posição 2. Dependendo da capacidade remanescente do transportador final, pode acontecer de ainda restarem caixas não coletadas, conforme mostradas a seguir:

[0,3,2,0,4,0,...] ou [0,2,2,0,4,0,...] ou [0,1,2,0,4,0,...] ou [0,0,2,0,4,0,...]

Para a primeira situação, o transportador efetuou a coleta de apenas uma caixa; na segunda situação o transportador final coletou duas caixas; na terceira situação, o transportador final coletou três caixas e por fim, o transportador final coletou todas as caixas da posição 2.

Caso o transportador já tivesse feito carregamento de caixas, e ainda existe capacidade remanescente, então em vez de partir do L2, este agora parte do local onde fez o último carregamento.

Considere outro exemplo: se a última coleta foi feita na posição 9 e ainda existe capacidade remanescente do transportador final então o modelo de simulação ativa a função `fun CalcMaxMachQIn(OrdMachInQ)` e verifica que existem duas caixas com as maiores filas, localizadas na posição 2 e na posição 5. Por causa do empate, deve-se

escolher a fila que se encontra na posição mais próxima da posição 9. Então o modelo de simulação ativa a função `fun DefMachDistIn(loc, p)`.

```
fun DefMachDistIn(loc, p) =  
let  
  
val MachListDist1=[0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7,3,4,5,6,7,8];  
val MachListDist2=[1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6,4,3,4,5,6,7]  
val MachListDist3=[2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5,5,4,3,4,5,6]  
val MachListDist4=[3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4,6,5,4,3,4,5]  
val MachListDist5=[4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3,7,6,5,4,3,4]  
val MachListDist6=[5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2,8,7,6,5,4,3]  
val MachListDist7=[1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7]  
val MachListDist8=[2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6]  
val MachListDist9=[3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5]
```

O modelo de simulação ativa a função denominada `fun CalcMinTranspDistInter(TranspDistListInter)` para efeito de comparação das distâncias para obter a que tiver a menor em relação à posição 9. Esta função comparativa está ilustrado a seguir:

```
fun CalcMinTranspDistInter(TranspDistListInter) =  
let  
    fun aux(f,L2) =  
        if List.null(L2) then  
            f  
        else  
            if f > hd(L2) then  
                aux(hd(L2),tl(L2))  
            else  
                aux(f,tl(L2))  
    in  
        aux(hd(TranspDistListInter),tl(TranspDistListInter))  
end;
```

E a posição escolhida para realizar a coleta é a posição 2, por apresentar a menor distância em comparação ao da posição 5, conforme mostrada na última linha na casa 2 e na casa 5.

O critério apresentado para a coleta de caixas é justamente o de maior fila e menor distância (por causa do empate).

Existe também a possibilidade de efetuar a coleta das caixas iniciando pela que estiver localizada mais próxima. Ainda considerando o exemplo ilustrado:

[0,4,2,0,4,0,...]

Estas são as informações coletadas pelo modelo de simulação em um determinado instante da simulação, as quais representam as filas das caixas formadas em cada posição aguardando a chegada do transportador final. Se o transportador final não efetuou nenhuma coleta anterior, então este está partindo do L2. Se está partindo do L2, o modelo de simulação ativa primeiramente a função `fun DefMachDistOut(p)`. De acordo com as informações contidas nesta função no arquivo `IN_CPN.sml`, a menor distância em relação ao L2 é a posição 3.

```
fun DefMachDistOut(p)=  
let
```

```
val MachListDist = [5,4,3,3,4,5,6,5,4,4,5,6,7,6,5,5,6,7,8,7,6,6,7,8];
```

A comparação destas distâncias para obter a posição cuja distância seja menor em relação ao L2 é obtida através da função `fun CalcMinTranspDistExit(TranspDistListExit)`, que de acordo com o arquivo `IN_CPN.sml` apresenta as seguintes informações:

```
fun CalcMinTranspDistExit(TranspDistListExit) =  
let  
  fun aux(f,L4) =  
    if List.null(L4) then  
      f  
    else  
      if f > hd(L4) then  
        aux(hd(L4),tl(L4))  
      else  
        aux(f,tl(L4))  
  in  
    aux(hd(TranspDistListExit),tl(TranspDistListExit))  
end;
```

Caso houvesse empate em termos de distância, a escolha seria baseada em maior fila, usando-se agora a função `fun CalcMaxMachQIn(OrdMachInQ)` para a escolha da maior, conforme mostrado a seguir:

```
fun CalcMaxMachQIn(OrdMachInQ) =  
let  
  fun aux(f,L1) =  
    if List.null(L1) then  
      f  
    else  
      if f < hd(L1) then  
        aux(hd(L1),tl(L1))  
      else
```

```

aux(f,tl(L1))
in
aux(hd(OrdMachInQ),tl(OrdMachInQ))
end;

```

Se o transportador final já realizou coleta de caixas anteriormente, então ativa-se a função `fun DefMachDistIn(loc, p)` e todo o procedimento volta a se repetir.

Com isso, encerra-se a descrição sobre as ações tomadas pelo modelo de simulação para o arranjo físico funcional.

2) Arranjo físico distribuído:

Inicia-se agora a descrição das ações tomadas pelo modelo de simulação para o arranjo físico aleatoriamente distribuído, arranjo físico parcialmente distribuído e arranjo físico maximamente distribuído. Felizmente, pode-se antecipar a descrição referente à coleta de caixas (onde estão armazenadas as peças acabadas), pois este é similar ao do arranjo físico funcional.

Local 1 até as máquinas: De forma similar ao do arranjo físico funcional, o modelo de simulação ativa a função do arquivo `IN_CPN.sml` denominada de `fun DefTransplniQtd()`.

```

fun DefTransplniQtd() =
1`1,"Yes")++
1`2,"Yes")++
1`3,"Yes")++
1`4,"Yes")++
1`5,"Yes")++
1`6,"Yes");

```

Esta função informa ao modelo de simulação o número de partidas do transportador inicial do L1. Neste caso, como a informação desta função informa de 1 até 6, então ao todo são 6 partidas do L1 para a entrega da matéria-prima.

Da mesma forma, o modelo de simulação deve ativar a função denominada de `fun DefTranspLocationIni()`, informando ao modelo de simulação de que o deslocamento a ser realizado parte do L1. Então:

```

fun DefTranspLocationIni() =
1`((Transp,1),Local(1))++
1`((Transp,2),Local(1))++
1`((Transp,3),Local(1))++
1`((Transp,4),Local(1))++
1`((Transp,5),Local(1))++
1`((Transp,6),Local(1));

```

Em seguida, o modelo de simulação efetua a leitura do tipo de peça do arquivo ORDERS.txt. Esta leitura é importante porque com esta informação, o modelo de simulação pode, então, ser informado sobre os tipos de operações que a peça deve se submeter para que, com isso, este faça a separação das máquinas para a formação da célula virtual.

Então, de acordo com as informações contidas no arquivo ORDERS.txt, conforme mostradas a seguir, o processo de entrega de matéria-prima deve ser iniciado pela peça tipo 1.

```
1`1,8,1,1,10,7470,53540)++  
1`1,8,2,87,10,7610,46070)++  
1`1,8,3,28,20,6300,38460)++  
1`1,8,4,32,10,3190,32160)++  
1`1,8,5,38,10,7240,28970)++
```

Em seguida, o modelo de simulação deve efetuar a leitura sobre todas as operações da peça tipo 1. Para isso, o modelo de simulação ativa a função denominada de fun DefOperacao (ProdT, OPTlistOPTon) e ao ativá-la, este então realiza a leitura das informações contidas nesta função do arquivo IN_CPN.sml.

```
fun DefOperacao(ProdT,OPTlistOPTon) =  
let  
  val OPTListProd1 = [OP(15)];  
  val OPTListProd2 = [OP(10)]
```

Infelizmente, a peça tipo 1 possui apenas uma operação, não sendo possível ilustrar o conjunto de máquinas formando a célula virtual. Por isso, considere o exemplo do terceiro transportador inicial usado para a entrega da matéria-prima, sendo que o conteúdo contido na primeira caixa (matérias-primas) é para a produção da peça tipo 78. Conforme as informações apresentadas pelo arquivo ORDERS.txt:

```
1`3,7,1,78,10,5360,59450)++  
1`3,7,2,50,20,13480,54090)++  
1`3,7,3,83,10,3540,40610)++
```

De acordo com as informações contidas no arquivo IN_CPN.sml, a peça tipo 78 possui a primeira operação na máquina tipo 3, a segunda operação na máquina tipo 5, a terceira operação ainda na máquina tipo 5 e por fim, a quarta operação ocorre na máquina tipo 14.

val OPTListProd78 = [OP(3),OP(5),OP(5),OP(14)]

Em seguida, o modelo de simulação ativa a função denominada de fun DefLayoutIn(laytIn). Considerando que o arranjo físico distribuído desta função seja o arranjo físico maximamente distribuído, então as informações contidas nesta função no arquivo IN_CPN.sml são:

fun DefLayoutIn(laytIn) = [3,9,10,4,5,6,13,12,11,1,15,8,7,14,8,2,9,3,15,5,4,6,7,14];

O modelo de simulação efetua a leitura de apenas uma amostra devido à informação contida no arquivo externo denominado de CrMax.txt. Caso contrário, se fosse o arranjo físico aleatoriamente distribuído ou o arranjo físico parcialmente distribuído, pode-se ter várias amostras. Neste caso, a leitura feita pela função DefLayoutIn e pela função DefLayoutInter dependem da variável laytIn e da variável laytInter, respectivamente.

Enfim, ao efetuar a leitura da amostra, este modelo de simulação faz a criação de um vetor interno para o armazenamento das máquinas deste arranjo físico maximamente distribuído. Assim, tem-se um vetor para a identificação das posições das máquinas requisitadas pelas peças para a formação da célula virtual (tal como mostrada pelo fun DefLayoutIn(laytIn)) e o outro, para a verificação sobre a disponibilidade das máquinas (tal como mostrada pelo fun DefLayoutIn1(laytIn)).

De acordo com as informações desta função fun DefLayoutIn(laytIn), pode-se notar que existem algumas máquinas com várias réplicas, tais como a máquina tipo 14, a máquina tipo 3 etc.

O modelo de simulação ativa agora a função denominada de fun DefMachDistIn(loc, p), que é a matriz de distâncias De-Para.

***fun DefMachDistIn(loc, p) =
let***

***val MachListDist1=[0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7,3,4,5,6,7,8];
val MachListDist2=[1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6,4,3,4,5,6,7]
val MachListDist3=[2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5,5,4,3,4,5,6]
val MachListDist4=[3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4,6,5,4,3,4,5]
val MachListDist5=[4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3,7,6,5,4,3,4]
val MachListDist6=[5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2,8,7,6,5,4,3]
val MachListDist7=[1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7]
val MachListDist8=[2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6]
val MachListDist9=[3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5]
val MachListDist10=[4,3,2,1,2,3,3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4]
val MachListDist11=[5,4,3,2,1,2,4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3]
val MachListDist12=[6,5,4,3,2,1,5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2]
val MachListDist13=[2,3,4,5,6,7,1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6]***

```

val MachListDist14=[3,2,3,4,5,6,2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5]
val MachListDist15=[4,3,2,3,4,5,3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4]
val MachListDist16=[5,4,3,2,3,4,4,3,2,1,2,3,3,2,1,0,1,2,4,3,2,1,2,3]
val MachListDist17=[6,5,4,3,2,3,5,4,3,2,1,2,4,3,2,1,0,1,5,4,3,2,1,2]
val MachListDist18=[7,6,5,4,3,2,6,5,4,3,2,1,5,4,3,2,1,0,6,5,4,3,2,1]
val MachListDist19=[3,4,5,6,7,8,2,3,4,5,6,7,1,2,3,4,5,6,0,1,2,3,4,5]
val MachListDist20=[4,3,4,5,6,7,3,2,3,4,5,6,2,1,2,3,4,5,1,0,1,2,3,4]
val MachListDist21=[5,4,3,4,5,6,4,3,2,3,4,5,3,2,1,2,3,4,2,1,0,1,2,3]
val MachListDist22=[6,5,4,3,4,5,5,4,3,2,3,4,4,3,2,1,2,3,3,2,1,0,1,2]
val MachListDist23=[7,6,5,4,3,4,6,5,4,3,2,3,5,4,3,2,1,2,4,3,2,1,0,1]
val MachListDist24=[8,7,6,5,4,3,7,6,5,4,3,2,6,5,4,3,2,1,5,4,3,2,1,0]

```

A comparação das distâncias entre os tipos de máquinas que fazem parte das operações da peça tipo 78 é realizada por uma nova função, denominada de `fun CalcMinListDistIn(CellDist)`. Esta função tem a finalidade de obter a árvore de rotas de menor distância, compostas por máquinas que irão executar a peça tipo 78. As informações contidas nesta função podem ser vistas abaixo:

```

fun CalcMinListDistIn(CellDist) =
let
  fun aux(f,L1) =
    if List.null(L1) then
      f
    else
      if f > hd(L1) then
        aux(hd(L1),tl(L1))
      else
        aux(f,tl(L1))
  in
    aux(hd(CellDist),tl(CellDist))
end;

```

Inicialmente esta função identifica as posições das réplicas das máquinas responsáveis pelas operações da peça tipo 78, conforme mostradas na Tabela 4.1.

Tabela 4.1 - Posições das réplicas das máquinas para a fabricação da peça tipo 78 (Fonte: autor)

Tipos de máquinas	3	5	14
Posições	1	5	14
	18	20	24

De acordo com a Tabela 4.2, as réplicas da máquina tipo 3 estão localizadas na posição 1 e na posição 18. As réplicas da máquina tipo 5 estão localizadas na posição 5

e na posição 20 e por fim, as réplicas da máquina tipo 14 estão localizadas na posição 14 e na posição 24. Sendo assim, existem ao todo oito possíveis rotas.

A Tabela 4.2 apresenta as oito possíveis rotas e as suas respectivas distâncias (em que Máq.3 representa a máquina tipo 3, Máq.5 representa a máquina tipo 5 e Máq.14 representa a máquina tipo 14). Note que a melhor rota é aquela que possui 5m.

Tabela 4.2 - Distâncias das oito possíveis rotas (Fonte: autor)

Posições			Distância (em metros)		Total
Máq. 3	Máq. 5	Máq. 14	Máq.3–Máq. 5	Máq.5–Máq.14	
1	5	14	4m	5m	9m
1	5	24	4m	4m	8m
1	20	14	4m	1m	5m
1	20	24	4m	4m	8m
18	5	14	3m	5m	8m
18	5	24	3m	4m	7m
18	20	14	5m	1m	6m
18	20	24	5m	4m	9m

Em seguida, o modelo de simulação faz a checagem no vetor da função `fun DefLayoutIn1(laytIn)` para identificar as disponibilidades da réplica da máquina tipo 3 localizada na posição 1, da réplica da máquina tipo 5 localizada na posição 20 e da réplica da máquina tipo 14 localizada na posição 14. Ou seja, este deve fazer a identificação na cópia da amostra do arranjo físico maximamente distribuído para verificar se as máquinas são diferentes de zero.

fun DefLayoutIn1(laytIn) = [3,9,10,4,5,6,13,12,11,1,15,8,7,14,8,2,9,3,15,5,4,6,7,14];

Note que todas as máquinas responsáveis pelas operações da peça tipo 78 estão livres. Deve-se agora reservar todas essas máquinas. Similarmente ao que ocorreu no arranjo físico funcional, o artifício usado para a separação dessas máquinas é zerar o tipo de máquina da amostra do arranjo físico, conforme mostrado a seguir:

fun DefLayoutIn1(laytIn)=[0 ,9,10,4,5,6,13,12,11,1,15,8,7,0,8,2,9,3,15,0,4,6,7,14];

Depois de realizar a reserva deste conjunto de máquinas, as quais são responsáveis pelas operações da peça tipo 78, o modelo de simulação ativa a função denominada de `fun DefMachToolListInit()` ao arquivo `IN_CPN.sml` com a finalidade de identificar quais são as ferramentas já instaladas em cada tipo de máquina. Como as posições escolhidas foram a posição 1, a posição 20 e a posição 14, então de acordo com as informações contidas nesta função no arquivo `IN_CPN.sml`, tem-se:

```
fun DefMachToolListInit() =  
1`Local(2),[9,4]++)  
1`Local(15),[5,7]++)  
1`Local(21),[1,8]++)
```

Em seguida, o modelo de simulação ativa a função `fun DefMachToolListProd(ProdT,OPTlistOPTon)` com a finalidade de identificar as ferramentas requisitadas pela peça tipo 78.

```
fun DefMachToolListProd(ProdT,OPTlistOPTon) =  
let  
val MachToolListProd78 = [9,9,3,4]
```

De acordo com a informação contida nesta função, a primeira operação (máquina tipo 3) utiliza-se da ferramenta tipo 9, a segunda operação (máquina tipo 5) utiliza-se da ferramenta tipo 9, a terceira operação (máquina tipo 5) utiliza-se da ferramenta tipo 3 e por fim, a quarta operação (máquina tipo 14) utiliza-se da ferramenta tipo 4.

O modelo de simulação faz a comparação entre as informações fornecidas pela função `fun DefMachToolListProd(ProdT,OPTlistOPTon)` e pela função `fun DefMachToolListInit()`. De acordo com a comparação, salvo a máquina tipo 3, todas as demais máquinas devem efetuar a troca de ferramenta. Esta troca de ferramenta é feita simultaneamente em todas as máquinas, e em decorrência dessa troca, deve-se levar em consideração o tempo de troca. Este tempo de troca é obtido através de uma função denominada de `fun SetupTime()`, similar ao do arranjo físico funcional, cujo o tempo é 1200s. Com a realização da troca, tem-se:

```
fun DefMachToolListInit() =  
1`Local(2),[9,4]++)  
1`Local(15),[9,3]++)  
1`Local(21),[4,8]++)
```

Com isso, a célula virtual está devidamente ajustada para o processamento do lote da peça tipo 78.

O modelo de simulação agora deve realizar a leitura da distância do L1 até a réplica da máquina tipo 3 localizada na posição 1. A leitura é feita ao ativar a função denominada de fun DefMachDist(p). De acordo com as informações contidas nesta função no arquivo externo IN_CPN.sml, a distância da posição 1 até o L1 é 8m, como vê-se a seguir:

```
fun DefMachDist(p)=  
let  
val MachListDist = [8,7,6,6,7,8,7,6,5,5,6,7,6,5,4,4,5,6,5,4,3,3,4,5];  
in  
  List.nth(MachListDist,p-1)  
end;
```

Esta distância é importante para a contabilização do tempo de deslocamento do transportador inicial para a entrega da matéria para a fabricação da peça tipo 78. O procedimento para a realização do cálculo do tempo é similar ao que foi apresentado para o arranjo físico funcional.

Como uma única empilhadeira é capaz de movimentar várias caixas (e, portanto, são vários tipos de peças), pode ser necessário continuar efetuando a entrega das caixas. O procedimento para a identificação das operações da peça a ser fabricada bem como o procedimento para a identificação da melhor rota também se repetem, fechando assim o ciclo. A única diferença é que em vez do modelo de simulação ativar a função fun DefMachDist(p), deve-se ativar a função fun DefMachDistIn(loc, p) para obter as distâncias.

5 ANÁLISE DOS RESULTADOS DO MODELO DE SIMULAÇÃO

Este tópico apresenta a síntese dos resultados do arranjo físico funcional, do arranjo físico aleatoriamente distribuído, do arranjo físico parcialmente distribuído e do arranjo físico maximamente distribuído obtidos do modelo de simulação. Os resultados do modelo de simulação podem ser vistos do do Apêndice 7 ao Apêndice 14.

As sínteses desses resultados obtidos do modelo de simulação serão apresentadas separadamente para cada tamanho de arranjo físico (quantidade total de máquinas), que são 24 máquinas, 56 máquinas e 150 máquinas, cujos formatos são 4x6, 7x8 e 10x15 respectivamente:

- 24 máquinas: os quatro tipos de arranjos físicos são submetidos a um mesmo ambiente de instabilidade, representados por cada combinação dos níveis dos fatores de experimentação.

De acordo com a segunda linha da Tabela 5.1, as amostras dos arranjos físicos foram submetidas ao modelo de simulação para produzir um conjunto de peças de um pedido (40 tipos de peças). Estas peças requisitam de uma a quatro operações, tamanho do lote de cada tipo de peça variando de 10 unidades a 20 unidades e tempo de cada operação variando de 30s a 32s.

Tabela 5.1 - Tempo de consolidação do pedido para 24 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-20	30-32	14326,04	7703,00	7733,00	7415,00
1-4	10-20	30-60	17176,33	11295,00	11122,00	10241,00
1-4	10-20	30-180	31325,11	25465,00	27665,00	25612,00

Após a execução do modelo de simulação, o tempo de consolidação do pedido obtido no arranjo físico funcional foi de 14326,04s para produzir as peças. Já no arranjo físico aleatoriamente distribuído foi de 7703s, no arranjo físico parcialmente distribuído foi de 7733s e no arranjo físico maximamente distribuído foi de 7415s. Para esta quantidade de máquinas (total de 24 máquinas), o desempenho do arranjo físico maximamente distribuído foi melhor.

Comparando-se o desempenho do arranjo físico maximamente distribuído com os demais arranjos físicos distribuídos, pode-se dizer que os desempenhos são muito similares. O bom

desempenho dos arranjos físicos distribuídos já era esperado devido à baixa formação de filas entre as máquinas, em virtude dos tempos de cada operação serem similares.

Comparando-se os desempenhos dos arranjos físicos distribuídos com o arranjo físico funcional, o tempo de consolidação do pedido do arranjo físico funcional é praticamente o dobro. Mas isso não implica necessariamente que o arranjo físico funcional deva ser descartado. É necessário também analisar o custo. Ao analisar a Tabela 5.2, verifica-se que o custo de consolidação do pedido no arranjo físico funcional também é maior (cerca de R\$400,00 de diferença), o que indica, então, que não é recomendável a implementação dos arranjos físicos funcionais num ambiente com este perfil.

Tabela 5.2 - Custo de consolidação do pedido para 24 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-20	30-32	4818,65	4469,31	4469,24	4460,87
1-4	10-20	30-60	4983,00	4660,82	4654,02	4623,89
1-4	10-20	30-180	5764,33	5445,94	5502,51	5440,29

Ao submeter-se os arranjos físicos para produzir as peças com aumento da variação no tempo de cada operação (variando de 30s a 60s), é de se esperar que exista maior formação de filas entre as máquinas, o que teoricamente prejudicaria o desempenho do arranjo físico distribuído. De fato, o aumento do tempo pode ser visto comparando-se a terceira linha com a segunda linha da Tabela 5.1. Entretanto, verifica-se que mesmo com este aumento dos tempos, os arranjos físicos distribuídos ainda continuam sendo melhores que o arranjo físico funcional, e mais uma vez, o arranjo físico maximamente distribuído se destaca.

Já com relação ao custo de consolidação do pedido, o custo nos arranjos físicos distribuídos, apesar de ser menor que o do arranjo físico funcional, pode-se dizer que estão muito próximos. Comparando-se os custos da segunda linha da Tabela 5.2 com os custos da terceira linha da Tabela 5.2, era de se esperar também que houvesse aumento dos custos, devido ao aumento da formação de filas entre as máquinas. Mas note que o aumento dos custos é praticamente insignificante e é proporcional a todos os tipos de arranjos físicos.

Novamente, ao submeter o modelo de simulação para produzir peças com variação ainda maior dos tempos de cada operação, conforme ilustrada na quarta linha da Tabela 5.1, o

desempenho do arranjo físico maximamente distribuído é melhor que o do arranjo físico aleatoriamente distribuído, porém os desempenhos estão muito próximos (diferença de cerca de 200s no tempo e de cerca de R\$6,00 no custo). Mais uma vez, os desempenhos dos dois tipos de arranjos físicos (arranjo físico funcional e arranjo físico distribuído) são prejudicados por causa da formação das filas em decorrência do aumento da variação dos tempos de cada operação. Isso pode ser constatado comparando-se a quarta linha da Tabela 5.1 com as demais linhas anteriores. Existe também aumento da formação de filas por causa da alta variação dos tempos de cada operação e, portanto, é de se esperar que também exista aumento dos custos. Este aumento dos custos pode ser notado comparando-se a quarta linha da Tabela 5.2 com as demais anteriores.

Até o presente resultado, de uma forma geral, era de se esperar mesmo que os tempos de consolidação do pedido dos arranjos físicos funcionais (ver linhas 2, 3 e 4 da Tabela 5.1) fossem maiores em função, principalmente, do tempo de troca de ferramental. Ou seja, sabe-se que como a troca de ferramenta num arranjo físico funcional ocorre logo que o lote chega na máquina responsável pela operação posterior, então todo lote aguarda até que a troca tenha sido feita e assim sucessivamente para todas as operações. Por outro lado, isso já não ocorre em um arranjo físico distribuído porque a troca é feita simultaneamente. Isso explica a diferença significativa entre os tempos de consolidação do pedido dos dois tipos de arranjos físicos.

Esta diferença seria menos significativa se houvesse aumento do tempo para executar a operação do lote (através do aumento do tamanho do lote ou do aumento do tempo de cada operação), conforme mostradas nas linhas 2 e 3 da Tabela 5.3. Daí, o tempo crítico passa a não ser o de troca de ferramenta e sim o de fabricação do lote. Como o impacto causado pelo tempo de troca de ferramenta é minimizado, é de se esperar, então, que o desempenho de um arranjo físico funcional se aproxime de um arranjo físico distribuído.

Tabela 5.3 - Tempo de consolidação do pedido para 24 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-50	30-32	20787,78	14887,00	13946,00	14972,00
1-4	10-50	30-60	26907,83	21737,00	20138,00	21921,00
1-4	10-50	30-180	51404,00	54298,00	44982,00	50826,00

De uma forma geral, também era de se esperar que os custos dos arranjos físicos funcionais (ver linhas 2, 3 e 4 da Tabela 5.2) fossem maiores que os dos arranjos físicos distribuídos porque existem custos associados ao uso dos recursos tais como transportador intermediário, caixas para o armazenamento de peças semi-acabadas e funcionário tipo 3.

Mesmo assim, considerando um arranjo físico de pequeno porte (até 24 máquinas, por exemplo), com o número de operações muito pequeno (até quatro operações) e pouca variação no tamanho do lote, de uma forma geral, é altamente recomendável a implementação dos arranjos físicos distribuídos.

Também foram submetidas as amostras dos arranjos físicos funcionais e dos arranjos físicos distribuídos aumentando-se a variação do tamanho do lote de cada tipo de peça, ou seja, os lotes variam de 10 a 50 unidades. A síntese dos resultados do modelo de simulação pode ser vista na Tabela 5.3 e na Tabela 5.4.

Tabela 5.4 - Custo de consolidação do pedido para 24 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-50	30-32	6413,83	5999,52	5970,14	6003,99
1-4	10-50	30-60	6762,34	6370,46	6319,51	6379,84
1-4	10-50	30-180	8208,71	8062,32	7742,85	7931,05

De acordo com a segunda linha da Tabela 5.3, verifica-se que o desempenho do arranjo físico funcional foi de 20787,78s, enquanto no arranjo físico aleatoriamente distribuído foi de 14887s, no arranjo físico parcialmente distribuído foi de 13946s e no arranjo físico maximamente distribuído foi de 14972s. Nota-se que mesmo com o aumento da variação do tamanho do lote de cada tipo de peça, o desempenho dos arranjos físicos distribuídos ainda continua sendo melhor que o arranjo físico funcional.

Em particular, destaca-se o desempenho, em tempo, do arranjo físico parcialmente distribuído, pois o seu desempenho é de cerca de 1000s menor que os demais arranjos físicos distribuídos.

Ao analisar a segunda linha da Tabela 5.4, verifica-se que os custos de consolidação do pedido dos arranjos físicos distribuídos ainda continuam sendo menores que o arranjo físico

funcional. O que significa que é recomendável a implementação do arranjo físico distribuído parcialmente distribuído.

Ao submeter os arranjos físicos para a produção de peças com poucas operações, grande variação do tamanho do lote de cada tipo de peça (variando de 10 unidades a 50 unidades) e aumento da variação nos tempos de cada operação (variando de 30s a 180s), o arranjo físico parcialmente distribuído continua se destacando, ou seja, a diferença do tempo de consolidação do pedido deste arranjo físico em relação aos outros dois arranjos físicos distribuídos pode chegar a 1800s.

O desempenho do arranjo físico funcional continua sendo pior que os arranjos físicos distribuídos, pois a diferença é de aproximadamente 5200s.

Nota-se que quanto maior a variação no tamanho dos lotes de cada tipo de peça, maior é a formação de filas (basta comparar, por exemplo, a terceira linha da Tabela 5.4 com a terceira linha da Tabela 5.2), o que reflete em aumento de custos de consolidação do pedido. Mas mesmo com este aumento, ainda é recomendado o arranjo físico distribuído, em especial o arranjo físico parcialmente distribuído.

De forma similar, no geral, uma possível justificativa para o tempo de consolidação do pedido no arranjo físico funcional estar cada vez mais próximo do arranjo físico distribuído se deve ao fato de que, com o aumento do tempo de fabricação do lote, o tempo de troca de ferramenta passa a não ser o crítico. Esta aproximação dos desempenhos em tempo podem ser vistos nos resultados apresentados na Tabela 5.3.

Agora, submetendo os arranjos físicos para produzir peças com alta variabilidade do tamanho dos lotes de cada tipo de peça (variando de 10 unidades a 50 unidades) e alta variação nos tempos de cada operação (variando de 30s a 180s), era de se esperar maior formação de fila entre as máquinas, e isso pode ser constatado comparando a terceira com a quarta linha da Tabela 5.4, ou seja, existe aumento do custo proveniente da manutenção de estoque de filas.

Note que os desempenhos em tempos dos arranjos físicos distribuídos podem até ser piores que o arranjo físico funcional, como é o caso do arranjo físico aleatoriamente distribuído. Para este perfil de peças da demanda, o arranjo físico parcialmente continua sendo melhor.

As amostras dos arranjos físicos agora são submetidas ao modelo de simulação para produzir as peças com alta variação no número de operações (variando de uma operação até oito operações) e pouca variação no tamanho dos lotes de cada tipo de peça (variando de 10 unidades a 20 unidades).

Com o aumento do número de operações, é de se esperar que a formação de células virtuais fique prejudicada, porque quanto maior o número de operações, menor é a probabilidade de ter as máquinas livres para a formação da célula virtual. Mas de acordo com os resultados da Tabela 5.5, a formação de células virtuais não prejudica o tempo de consolidação dos pedidos,

porque todos os resultados dos arranjos físicos distribuídos foram menores que o arranjo físico funcional. Para os perfis de peças apresentados na Tabela 5.5, pode-se dizer que independente da variação nos tempos de cada operação (de 30s até 180s), o desempenho do aleatoriamente é menor, desde que a variação do tamanho do lote de cada tipo de peça seja pequena.

Tabela 5.5 - Tempo de consolidação do pedido para 24 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-20	30-32	30391,50	15682,00	16887,00	16722,00
1-8	10-20	30-60	34808,56	23705,00	23911,00	23034,00
1-8	10-20	30-180	61069,61	51742,00	59329,00	55739,00

Com relação ao aumento da variação do número de operações, é de se esperar que o tempo de consolidação do pedido, de uma forma geral, aumente. Este fato pode ser visto ao compararmos os resultados apresentados na Tabela 5.5 com os que estão apresentados na Tabela 5.1.

De acordo com a Tabela 5.6, quanto maior a variação nos tempos de operação, maior é a formação de filas entre as máquinas, e consequentemente os custos são cada vez maiores. Isso pode ser constatado partindo da análise da segunda linha para terceira linha e da terceira para a quarta linha. Mesmo assim, com a formação de filas, os desempenhos dos arranjos físicos distribuídos continuam satisfazendo os dois requisitos de desempenho, ou seja, em termos de tempo e custo. Em particular, destaca-se o desempenho do arranjo físico aleatoriamente distribuído.

Ao submeter as amostras dos arranjos físicos para a produção de peças com alta variação do número de operações (variando de uma operação a oito operações), alta variação dos lotes de cada tipo de peça (variando de 10 unidades a 50 unidades), tem-se os resultados, conforme mostrados na Tabela 5.7 e Tabela 5.8.

Tabela 5.6 - Custo de consolidação do pedido para 24 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-20	30-32	5520,75	4857,74	4897,77	4892,35
1-8	10-20	30-60	5797,06	5249,72	5259,99	5231,18
1-8	10-20	30-180	7264,17	6731,88	6980,79	6862,12

Tabela 5.7 - Tempo de consolidação do pedido para 24 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-50	30-32	42923,89	27479,00	30160,00	27038,00
1-8	10-50	30-60	54344,22	40293,00	45315,00	41010,00
1-8	10-50	30-180	100852,30	97182,00	103587,00	91973,00

Tabela 5.8 - Custo de consolidação do pedido para 24 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-50	30-32	7464,24	6670,81	6759,68	6656,50
1-8	10-50	30-60	8113,89	7352,53	7512,76	7379,09
1-8	10-50	30-180	10842,94	10332,98	10562,81	10166,59

Para peças com pouca variação nos tempos de cada operação, o desempenho do arranjo físico maximamente distribuído é o menor (27038s), porém aumentando-se a variação dos

tempos de cada operação (variando de 30s até 60s), o arranjo físico aleatoriamente distribuído apresenta o menor resultado. Aumentando-se ainda mais a variação no tempo de cada operação (variando de 30s a 180s), a situação se inverte novamente, ou seja, o desempenho do arranjo físico maximamente distribuído volta a ser novamente o menor.

Com relação à Tabela 5.8, era de se esperar que quanto maior a variação dos tempos de cada operação, maior seria a formação de fila, o que reflete em aumento dos custos. Mas mesmo assim, os custos (independente da variação dos tempos de cada operação) nos arranjos físicos distribuídos são menores que o arranjo físico funcional.

Encerra-se, então, a análise para os arranjos físicos com 24 máquinas. De uma forma geral, pode-se afirmar que é vantajoso implementar os arranjos físicos distribuídos, principalmente quando as peças apresentarem alta variação do número de operações, alta variação dos tamanhos dos lotes de cada tipo de peça e alta variação dos tempos de cada operação.

Em particular, o arranjo físico maximamente é melhor em cinco experimentos, o arranjo físico aleatoriamente é melhor em quatro experimentos e o arranjo físico parcialmente distribuído é melhor em três experimentos. Isso confirma os estudos de Gorgulho Júnior & Gonçalves Filho (2007). De acordo com os autores, a ordem do desempenho (do pior para o melhor) é: arranjo físico funcional, arranjo físico parcialmente distribuído, arranjo físico aleatoriamente distribuído e arranjo físico maximamente distribuído.

Entretanto, de acordo com os estudos de Benjaafar & Sheikhzadeh (2000) e Lahmar & Benjaafar (2001, 2002 e 2005), o arranjo físico maximamente distribuído não se justifica pois a maior parte dos benefícios seriam obtidos quando os departamentos são divididos em duas partes.

Deve-se então realizar mais experimentos para outros tamanhos de arranjos físicos para comprovar a hipótese desses autores. O próximo experimento é para arranjos físicos com um total de 56 máquinas.

- 56 máquinas: a Tabela 5.9 apresenta os perfis das peças a serem produzidas nos arranjos físicos com 56 máquinas. Ao submeter as amostras dos arranjos físicos ao modelo de simulação para a produção de peças com poucas operações (variando de uma a quatro operações), pouca variação do tamanho do lote de cada tipo de peça (variando de 10 unidades a 20 unidades) e pouca variação nos tempos de cada operação (variando de 30s a 32s), o arranjo físico parcialmente distribuído apresenta o melhor desempenho, porém este desempenho é de cerca de 200s menor em relação aos demais arranjos físicos distribuídos, conforme mostrada na segunda linha da Tabela 5.9. Em termos de custo, este arranjo físico também apresenta o menor custo, conforme apresentada na Tabela 5.10.

Tabela 5.9 - Tempo de consolidação do pedido para 56 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-20	30-32	9314,44	6320,00	6053,00	6298,00
1-4	10-20	30-60	11171,17	9664,00	9681,00	9337,00
1-4	10-20	30-180	22286,61	20426,00	19988,00	22059,00

Tabela 5.10 - Custo de consolidação do pedido para 56 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-20	30-32	8906,12	8582,28	8569,27	8586,01
1-4	10-20	30-60	9112,28	8903,97	8905,77	8879,21
1-4	10-20	30-180	10270,68	10033,34	10014,68	10169,56

Aumentando-se a variabilidade nos tempos de cada operação (variando de 30s a 60s), o desempenho do arranjo físico maximamente distribuído passa a ser o melhor (cerca de 300s menor em relação ao segundo melhor, que é o arranjo físico aleatoriamente distribuído).

Com relação ao custo, também era de se esperar de que houvesse aumento dos custos em todos os arranjos físicos. O custo do arranjo físico maximamente distribuído é o menor, apesar de estar muito próximo dos demais arranjos físicos distribuídos.

Aumentando-se novamente a variação nos tempos de cada operação (variando de 30s a 180s), o arranjo físico parcialmente distribuído volta a ser o melhor, por ter o menor tempo de consolidação do pedido, satisfazendo tanto o tempo quanto o custo.

Ao submeter as amostras dos arranjos físicos contendo 56 máquinas com aumento da variação dos tamanhos dos lotes de cada tipo de peça e fixando o número de operações (variando de uma operação a quatro operações), tem-se os resultados conforme apresentados na Tabela 5.11 e na Tabela 5.12.

Tabela 5.11 - Tempo de consolidação do pedido para 56 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-50	30-32	13408,89	11538,00	14057,00	11581,00
1-4	10-50	30-60	16729,83	15242,00	16780,00	19612,00
1-4	10-50	30-180	34370,28	36071,00	37953,00	37505,00

Tabela 5.12 - Custo de consolidação do pedido para 56 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-50	30-32	10578,32	10296,41	10487,03	10320,34
1-4	10-50	30-60	9702,13	10708,43	10823,85	11037,03
1-4	10-50	30-180	11433,17	12937,36	13083,94	13043,86

Os resultados mostram que quando a variação do tempo de cada operação aumenta, o arranjo físico funcional tende a ser o melhor. Inclusive o arranjo físico também tende a ter o menor custo.

Em função dos resultados apresentados, pode-se dizer que em um ambiente de alta variação no tamanho do lote e alta variação nos tempos de cada operação, os arranjos físicos funcionais são os melhores para serem implementados para a produção de peças com poucas operações.

Aumentando-se a variação do número de operações de cada tipo de peça, podendo variar agora de uma operação a oito operações, tem-se os seguintes resultados, conforme mostrados na Tabela 5.13 e na Tabela 5.14.

Para peças que apresentarem pequenas variações nos tempos de cada operação e que apresentarem pouca variação do tamanho do lote de cada tipo de peça (lotes variando de 10 unidades a 20 unidades), o desempenho do arranjo físico parcialmente distribuído é melhor (tempo e custo).

Tabela 5.13 - Tempo de consolidação do pedido para 56 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-20	30-32	14922,61	13419,00	10622,00	14284,00
1-8	10-20	30-60	18385,61	16790,00	14866,00	19501,00
1-8	10-20	30-180	34574,17	40315,00	36674,00	44811,00

Tabela 5.14 - Custo de consolidação do pedido para 56 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-20	30-32	9486,56	9247,02	9031,52	9314,17
1-8	10-20	30-60	9873,99	9636,08	9488,54	9846,94
1-8	10-20	30-180	11683,46	12005,17	11735,44	12362,16

Quando o tempo de cada operação da peça apresentar variação de 30s a 60s, tal como mostrada na segunda linha da Tabela 5.13, o desempenho do arranjo físico parcialmente distribuído é cerca de 2500s menor em comparação aos demais arranjos físicos distribuídos. O custo do arranjo físico parcialmente distribuído também é menor de todos, apesar de este custo estar bem próximo dos demais arranjos físicos distribuídos (cerca de R\$200,00 de diferença).

Porém, quando a peça apresentar ainda maior variação no tempo de cada operação (tal como variando de 30s a 180s), então o arranjo físico funcional passa a ser o melhor (2100s menor em relação ao segundo melhor desempenho, o arranjo físico parcialmente distribuído), inclusive o custo do arranjo físico funcional também é o menor, apesar da diferença ser pequena em relação ao arranjo físico parcialmente distribuído (cerca de R\$150,00 menor).

Agora, considere que os arranjos físicos devam produzir peças com alta variação (variando de uma a oito operações) e alta variação no tamanho dos lotes de cada tipo de peça. Ao executar o modelo de simulação, tem-se os seguintes resultados, conforme mostrados na Tabela 5.15 e na Tabela 5.16.

Tabela 5.15 - Tempo de consolidação do pedido para 56 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-50	30-32	20960,06	24065,00	18163,00	25957,00
1-8	10-50	30-60	26233,78	35599,00	26191,00	36490,00
1-8	10-50	30-180	54529,00	73747,00	67603,00	82242,00

Tabela 5.16 - Custo de consolidação do pedido para 56 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-50	30-32	11443,18	11496,15	11040,64	11644,91
1-8	10-50	30-60	12097,60	12649,92	11918,82	12715,02
1-8	10-50	30-180	15417,77	16712,86	16229,65	17369,89

Para peças que apresentarem pouca variação nos tempos de cada operação, o desempenho do arranjo físico parcialmente distribuído é o menor (cerca de 2800s em relação ao segundo melhor desempenho, que é do arranjo físico funcional). O custo também é o menor (cerca de R\$400,00 a menos que o custo do arranjo físico funcional).

Aumentando-se a variação dos tempos de cada operação (ou seja, os tempos de cada operação podem variar de 30s a 60s), o desempenho do arranjo físico parcialmente distribuído continua sendo o melhor, porém muito próximo ao desempenho do arranjo físico funcional (diferença de cerca de 40s). O custo no arranjo físico parcialmente distribuído continua sendo menor que no arranjo físico funcional (diferença de R\$170,00).

O desempenho do arranjo físico funcional se torna melhor que os demais arranjos físicos distribuídos quando as peças apresentarem alta variação dos tempos de cada operação (de 30s a 180s). A menor diferença está entre o arranjo físico funcional e o arranjo físico parcialmente distribuído (com uma diferença entre os tempos de cerca de 13000s e com uma diferença entre os custos de cerca de R\$800,00).

Encerra-se então a análise dos resultados para os arranjos físicos com 56 máquinas. De acordo com os resultados, o arranjo físico funcional apresenta o melhor desempenho em três experimentos, dois experimentos para o arranjo físico aleatoriamente distribuído, seis experimentos para o arranjo físico parcialmente distribuído e apenas um experimento para o arranjo físico maximamente distribuído. Para esta quantidade de máquinas, comprova-se a hipótese de Benjaafar & Sheikhzadeh (2000) e Lahmar & Benjaafar (2001, 2002 e 2005).

Caso houvesse construído o modelo de simulação para analisar o desempenho dos arranjos físicos com apenas 24 máquinas, a conclusão obtida seria que o arranjo físico maximamente distribuído seria melhor, mas isso não seria válido para um arranjo físico com 56 máquinas, daí a importância de se criar um modelo sem a necessidade de mudança em estrutura lógica.

Para o arranjo físico com 56 máquinas, vale destacar o desempenho do arranjo físico funcional, pois este se sobressaiu em um ambiente de alta variação nos tempos de cada operação, alta variação do número de operações e alta variação no tamanho dos lotes de cada tipo de peça.

- 150 máquinas: Agora, os arranjos físicos a serem submetidos na produção de peças possuem uma quantidade total de 150 máquinas. A Tabela 5.17 apresenta os resultados da simulação. De acordo com esta tabela, peças com poucas operações (variando de uma operação a quatro operações), com pouca variação nos tamanhos dos lotes de cada tipo de peça (variando de 10 unidades a 20 unidades) e com pouca variação nos tempos de cada operação (variando de 30s a 32s), o desempenho do arranjo físico parcialmente distribuído é o melhor (cerca de 260s menor que o segundo melhor, que é do arranjo físico maximamente distribuído). Em termos de custo, o custo do arranjo físico parcialmente distribuído também é o menor, porém a diferença com o arranjo físico maximamente distribuído é de cerca de R\$55,00, conforme mostrada na Tabela 5.18.

Tabela 5.17 - Tempo de consolidação do pedido para 150 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-20	30-32	8875,33	5694,00	5239,00	5498,00
1-4	10-20	30-60	10638,22	7558,00	6998,00	6192,00
1-4	10-20	30-180	22157,67	14504,00	14998,00	13782,00

Tabela 5.18 - Custo de consolidação do pedido para 150 máquinas: peças com baixa variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-20	30-32	21377,80	20695,92	20589,64	20646,36
1-4	10-20	30-60	21806,43	21116,31	20999,65	20846,57
1-4	10-20	30-180	24512,5	22844,05	22945,44	22691,91

Aumentando-se a variação do tempo de cada operação (variando de 30s a 60s), o desempenho do arranjo físico maximamente distribuído passa a ser o melhor (uma diferença de cerca de 800s em relação ao parcialmente distribuído), conforme a Tabela 5.17. Com relação ao custo, o custo no arranjo físico maximamente distribuído é cerca de R\$150,00 menor que no arranjo físico parcialmente distribuído, ver Tabela 5.18.

Aumentando-se ainda mais a variação nos tempos de cada operação (de 30s a 180s), o desempenho do arranjo físico maximamente distribuído continua sendo o melhor em relação aos demais arranjos físicos, tanto em termos de tempo quanto em termos de custo, conforme mostradas na Tabela 5.17 e Tabela 5.18.

Aumentando-se a variação do tamanho do lote de cada tipo de peça (variando de 10 unidades a 50 unidades), o desempenho do arranjo físico parcialmente distribuído é melhor em cerca de 140s em relação ao segundo melhor desempenho, que é do arranjo físico maximamente distribuído, ver Tabela 5.19. Em termos de custo, o arranjo físico parcialmente distribuído é menor em cerca de R\$30,00 do arranjo físico maximamente distribuído, ver Tabela 5.20.

Ainda de acordo com a Tabela 5.19 e com a Tabela 5.20, ao submeter as amostras dos arranjos físicos para produzir peças que tenham uma variação nos tempos de cada operação de 30s a 60s, o desempenho do arranjo físico aleatoriamente distribuído passa a ser o melhor cerca de 200s menor que os demais arranjos físicos distribuídos). Com relação ao custo do arranjo físico aleatoriamente distribuído, este apresenta um custo menor em cerca de R\$10,00 em relação aos demais arranjos físicos distribuídos.

Tabela 5.19 - Tempo de consolidação do pedido para 150 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-50	30-32	12146,11	11828,00	10827,00	10960,00
1-4	10-50	30-60	15528,33	12186,00	12406,00	12437,00
1-4	10-50	30-180	31817,67	24605,00	27141,00	24829,00

Tabela 5.20 - Custo de consolidação do pedido para 150 máquinas: peças com baixa variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-4	10-50	30-32	23378,91	23359,45	23115,83	23146,45
1-4	10-50	30-60	24207,68	23534,02	23543,76	23546,04
1-4	10-50	30-180	28184,54	26617,19	27149,50	26666,48

Aumentando-se ainda mais a variação nos tempos de cada operação (variando de 30s a 180s), o arranjo físico aleatoriamente distribuído também continua sendo menor, cerca de 200s menor que o arranjo físico maximamente distribuído.

Aumentando-se a variação do número de operações de cada peça, porém com pouca variação do tamanho do lote, o modelo de simulação apresenta os resultados, conforme mostrados na Tabela 5.21 e na Tabela 5.22.

Conforme a Tabela 5.21, quando as peças apresentarem pouca variação nos tempos de cada operação, o desempenho do arranjo físico maximamente distribuído é o melhor (cerca de 200s a menos que o desempenho do segundo melhor, que é do arranjo físico aleatoriamente distribuído). O custo do arranjo físico maximamente distribuído também é menor, em cerca de R\$40,00 a menos que o custo no arranjo físico aleatoriamente distribuído.

Quando as peças apresentarem maior variação nos tempos de cada operação (variando de 30s a 60s), o arranjo físico parcialmente distribuído passa a ser o melhor (cerca de 2800s em relação ao segundo melhor, que é do arranjo físico aleatoriamente distribuído). O custo do arranjo

físico parcialmente distribuído também é menor, em cerca de R\$600,00 em relação ao arranjo físico aleatoriamente distribuído.

Tabela 5.21 - Tempo de consolidação do pedido para 150 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-20	30-32	15226,94	9209,00	9508,00	9036,00
1-8	10-20	30-60	18508,17	14096,00	11256,00	14665,00
1-8	10-20	30-180	34280,06	25375,00	33376,00	22364,00

Tabela 5.22 - Custo de consolidação do pedido para 150 máquinas: peças com alta variação do número de operações e baixa variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-20	30-32	22848,29	21500,07	21554,83	21461,49
1-8	10-20	30-60	23651,03	22630,04	22042,41	22750,14
1-8	10-20	30-180	27490,96	25534,08	27178,1	24887,16

Aumentando-se a variação nos tempos de cada operação (variando de 30s a 180s), o desempenho do arranjo físico maximamente distribuído passa a ser o melhor (cerca de 3000s menor que o segundo melhor, que é do arranjo físico aleatoriamente distribuído). O custo do arranjo físico maximamente distribuído também é menor (cerca de R\$650,00 a menos que o segundo melhor, que é do arranjo físico aleatoriamente distribuído).

Agora, a última parte da análise no que diz respeito aos arranjos físicos contendo 150 máquinas, é simular as peças com alta variação dos tamanhos dos lotes de cada tipo de peça. Neste caso, os tamanhos dos lotes variam de 15 unidades a 50 unidades. Ao executar o modelo de simulação, tem-se os resultados, conforme apresentados na Tabela 5.23 e na Tabela 5.24.

Tabela 5.23 - Tempo de consolidação do pedido para 150 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-50	30-32	19906,00	14316,00	16707,00	13124,00
1-8	10-50	30-60	24284,89	20947,00	19254,00	19585,00
1-8	10-50	30-180	48986,50	48561,00	55931,00	38823,00

Tabela 5.24 - Custo de consolidação do pedido para 150 máquinas: peças com alta variação do número de operações e alta variação do tamanho do lote (Fonte: autor)

Número de operações	Quantidade de cada tipo de peça demandada pelo cliente	Tempo de cada operação	Arranjo físico funcional	Arranjo físico aleatoriamente distribuído	Arranjo físico parcialmente distribuído	Arranjo físico maximamente distribuído
1-8	10-50	30-32	25281,31	24009,62	24500,89	23801,88
1-8	10-50	30-60	26440,58	25628,26	25264,49	25352,04
1-8	10-50	30-180	32688,76	32459,85	34003,78	30444,96

Nota-se na Tabela 5.23 que quando as peças apresentarem pouca variação nos tempos de cada operação (variando de 30s a 32s), o arranjo físico maximamente distribuído é o melhor (cerca de 1200s menor que o segundo melhor desempenho, que é do arranjo físico aleatoriamente distribuído). O custo do arranjo físico maximamente distribuído também é menor (cerca de R\$200,00 a menos que o do arranjo físico aleatoriamente distribuído).

Aumentando-se a variação dos tempos de cada operação (variando de 30s a 60s), o arranjo físico parcialmente distribuído passa a ser o melhor (cerca de 330s a menos que o segundo melhor desempenho, que é do arranjo físico maximamente distribuído). O custo do arranjo físico parcialmente distribuído é R\$90,00 a menos que o do arranjo físico maximamente distribuído).

Em uma situação em que as peças possuem alta variação nos tempos de cada operação (variando de 30s a 180s), o arranjo físico maximamente distribuído volta a ser o melhor (cerca de 9700s a menos que o segundo melhor, que é o desempenho do arranjo físico aleatoriamente distribuído). Quanto ao custo, o do arranjo físico maximamente distribuído é R\$2000,00 a menos que o do arranjo físico aleatoriamente distribuído.

Verificando-se todos os resultados dos experimentos realizados para um arranjo físico contendo 150 máquinas, era de se esperar que o modelo de simulação deparasse com a maior probabilidade de encontrar máquinas disponíveis para a formação de células virtuais e, portanto, o lote de matéria-prima a ser processada esperaria menos tempo para iniciar as operações. Entretanto, não era de se esperar que o desempenho do arranjo físico funcional se aproximasse dos arranjos físicos distribuídos. Uma possível justificativa é que também aumentam-se as possibilidades do modelo de simulação encontrar réplicas das máquinas requisitadas pelas peças estarem livres. Com isso, os critérios de seleção de máquinas não são adotados pelo modelo de simulação, já que este escolhe sempre a réplica que estiver livre e a que estiver mais próxima do transportador (inicial ou intermediário). Por isso, o lote de matéria-prima/peça semi-acabada praticamente recebe a operação desejada sem esperas.

Com esses resultados, o desempenho do arranjo físico aleatoriamente distribuído é melhor dois experimentos, o arranjo físico parcialmente distribuído é melhor em quatro experimentos e o arranjo físico maximamente distribuído é melhor em seis experimentos. Vale lembrar que em um ambiente com alta variação do número de operações das peças, alta variação do tamanho dos lotes de cada tipo de peça e alta variação dos tempos de cada operação, destaca-se o arranjo físico maximamente distribuído. Novamente, isso confirma novamente as hipóteses de Gorgulho Júnior & Gonçalves Filho (2007).

6 CONCLUSÕES E TRABALHOS FUTUROS

Após a análise dos resultados, pode-se dizer que de uma forma geral, os arranjos físicos distribuídos são vantajosos para serem implementados no chão de fábrica, pois satisfazem ambos os parâmetros de desempenho adotados neste estudo (tempo e custo de consolidação do pedido). Entretanto, vale lembrar que o arranjo físico funcional teve o melhor desempenho com 56 máquinas, em um ambiente em que as peças apresentavam alta variação no número de operações, alta variação nos lotes de cada tipo de peça e alta variação nos tempos de cada operação.

Realizando-se a análise por tamanho de arranjo físico (quantidade total de máquinas), verifica-se que após a execução do modelo de simulação para 24 máquinas, o arranjo físico maximamente distribuído é o que apresenta o melhor desempenho. Já ao realizar a execução do modelo de simulação para 56 máquinas, o arranjo físico parcialmente distribuído é o que apresenta o melhor desempenho, e por fim, ao realizar a execução do modelo de simulação para 150 máquinas, o arranjo físico maximamente distribuído volta a ter o melhor desempenho.

Ao que tudo indica, aparentemente os resultados comprovam mais as hipóteses de Gorgulho Júnior & Gonçalves Filho (2007) que as de Benjaafar & Sheikhzadeh (2000) e Lahmar & Benjaafar (2001, 2002 e 2005).

Analisando agora em termos de ordem de desempenho para todos os experimentos realizados, Gorgulho Júnior & Gonçalves Filho (2007) afirmam que o desempenho dos arranjos físicos está na seguinte ordem (do pior para o melhor): arranjo físico funcional, arranjo físico parcialmente distribuído, arranjo físico aleatoriamente distribuído e arranjo físico maximamente distribuído. Já Benjaafar & Sheikhzadeh (2000) e Lahmar & Benjaafar (2001, 2002 e 2005), infelizmente, não apresentam a ordem de desempenho dos arranjos físicos.

Para este estudo, ao realizar a contagem de todos os 36 pares de experimentos, o arranjo físico funcional foi melhor em três pares de experimentos, o arranjo físico aleatoriamente distribuído foi melhor em oito pares de experimentos, o arranjo físico parcialmente distribuído foi melhor em treze pares de experimentos e o arranjo físico maximamente distribuído foi melhor em doze pares de experimentos.

Então, a ordem de desempenho neste estudo é (do pior para o melhor): arranjo físico funcional, arranjo físico aleatoriamente distribuído, arranjo físico maximamente distribuído e arranjo físico parcialmente distribuído. Portanto, a maior parte dos benefícios realmente é conseguida implementando o arranjo físico parcialmente distribuído dividindo cada departamento em duas partes, conforme afirmam Benjaafar & Sheikhzadeh (2000) e Lahmar & Benjaafar (2001, 2002 e 2005).

Era de se esperar que quanto maior o número de operações das peças, maior seria a dificuldade para a separação do conjunto de máquinas para a formação das células virtuais porque menor seria a probabilidade de encontrar máquinas livres. Isso quer dizer que os lotes de peças demandariam mais

tempo para iniciar as operações nos arranjos físicos distribuídos, resultando em tempos maiores de consolidação do pedido que no arranjo físico funcional. Mas nota-se que isso não ocorre. Mesmo com grande número de operações, os arranjos físicos distribuídos se mostraram até melhores que o arranjo físico funcional, tal como ocorrido com 24 máquinas.

Os artigos consultados neste estudo referentes aos arranjos físicos distribuídos não comentam como as peças foram geradas, como os arranjos físicos são gerados, como as máquinas são espalhadas, como se faz a checagem das situações das máquinas, como as peças são movimentadas no chão de fábrica etc. para fins de estudo em modelagem e simulação. Com este trabalho, dados podem ser gerados e implementados em qualquer código de programação.

Além disso, tais artigos consultados consideram que já existe prontamente a informação sobre o número de réplicas de cada tipo de máquina bem como a informação do formato dos locais no chão de fábrica. Mas quando se for fazer um real estudo, nem sempre estas informações estão disponíveis e, portanto, necessitam ser avaliadas e geradas.

Ainda com relação aos artigos consultados, estes apresentam estudos de desempenho usando-se de modelos matemáticos implementados às linguagens de programação. Entretanto, não levam em consideração as concomitâncias na utilização dos recursos produtivos, o que pode tornar o estudo discrepante.

A modelagem realizada na *software* de simulação CPN tools permite que se faça ativação de diversas partes do modelo simultaneamente. Essa opção é muito útil nos casos em que, quando uma entidade (por exemplo uma peça) necessitar chegar até a máquina que executa a operação seguinte, o modelo “enxergue” antes a máquina escolhida (conforme os critérios estabelecidos) e faça a escolha para assim liberar a entidade a ela.

Apesar das técnicas de modelagem e simulação serem muito usadas em diversas áreas, diversos trabalhos apresentam a modelagem com capacidade de ser usada na simulação de apenas um único tamanho do sistema modelado. Isso ocorre devido à dificuldade de se criar um modelo sem a necessidade de mudança em sua estrutura lógica. Em consequência disso, muitas conclusões precipitadas podem estar sendo obtidas.

Este trabalho de pesquisa mostrou que é possível construir um modelo de simulação capaz de simular vários tamanhos de chão de fábrica (quantidade total de máquinas) sem a necessidade de modificação em sua estrutura lógica. Isso só foi possível em virtude de trabalhar juntamente com os códigos de programação.

Com relação à forma de atuação dos funcionários no arranjo físico distribuído, nota-se que como as peças já têm o percurso certo definido (por causa da célula virtual), todos os operadores das máquinas desta célula virtual já podem efetuar as trocas simultaneamente. Diferente do que ocorria no arranjo físico funcional, em que a troca só era feita quando a peça alcançava o departamento. Portanto, a característica do arranjo físico distribuído não é simplesmente definida pelo espalhamento das

máquinas e nem pela formação de células virtuais, mas também toda uma filosofia de trabalho em equipe envolvida. Caso contrário, um arranjo físico distribuído, mesmo com a formação de células virtuais, pode incorporar as características de um arranjo físico funcional, tais como longas esperas em fila, como longos tempos de trocas de ferramentais etc.

A forma abordada neste trabalho no que diz respeito aos parâmetros de entrada atende às expectativas de um mercado instável. As peças demandadas chegam ao longo do tempo, e o modelo identifica instantaneamente a célula virtual desejada. Os lotes aqui considerados, também variam e são considerados pesos das peças.

Como proposta para trabalhos futuros:

- Fazer a execução do modelo de simulação para várias formatações dos locais formados no chão de fábrica. Pode-se, também, realizar a execução do modelo de simulação considerando o problema de formação do vão vazio no chão de fábrica, usando-se o procedimento apresentado nesta tese;
- Analisar o desempenho do arranjo físico distribuído considerando-se “quebras” de lotes grandes em lotes menores (diferente de tamanho unitário);
- Fazer a modelagem, usando outro *software* de simulação, para a formação correta das células virtuais;
- Propôr um sistema de programação da produção em tempo-real para a identificação efetiva das máquinas para formação das células virtuais no chão de fábrica; por enquanto, esta identificação em “tempo-real” é feita pelo modelo de simulação, conforme as peças da demanda chegam à fábrica, mas agora é necessário este tipo de sistema para ser implementado numa fábrica real;
- Considerar a hipótese de trabalhar com operadores multifuncionais;
- Implementar delineamento de experimentos para identificar quais fatores são os mais críticos. Como não era o enfoque deste estudo, isso pode ser feito posteriormente complementando a esta tese;

REFERÊNCIA BIBLIOGRÁFICA

ASKIN, R. G. et al. An empirical evaluation of holonic and fractal layouts. **International Journal of Production Research**, v. 37, n. 5, p. 961-978, 1999.

BANKS, J. **Handbook of simulation: principles, methodology, advances, applications, and practice**. New York: John Wiley & Sons, 1998.

BANKS, J. et al. **Discrete-event system simulation**. 2nd ed. New Jersey: Prentice-Hall, 1984.

BATEMAN, R. E. et al. **A simulação: aprimorando os sistemas**. São Paulo: Belge, 1999.

BENJAAFAR, S. Models for performance evaluation of flexibility in manufacturing systems. **International Journal of Production Research**, v. 32 No.6, pp.1383-1402, 1994.

BENJAAFAR, S. et al. Next generation factory layouts: Research challenges and recent progress. **Interfaces**, v. 32, n. 6, p. 58-76, 2002.

BENJAAFAR, S.; SHEIKHZADEH, M. Design of flexible layouts for manufacturing systems. In: INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, Minneapolis, Minnesota, 1996. **Proceedings of the 1996 IEEE**, Minneapolis, Minnesota, 1996.

BENJAAFAR, S.; SHEIKHZADEH, M. Design of flexible plant layouts. **IIE Transactions**, v. 32, p. 309-322, 2000.

BLACK, J. T. **O projeto da fábrica com futuro**. Porto Alegre: Artes Médicas Editora Bookman, 1998.

BLAGA, F. et al. Coloured Petri nets modeling using CPNTools. In: FASCICLE OF MANAGEMENT AND TECHNOLOGICAL ENGINEERING, 2008. **Annals of the Oradea University**, 2008, v. VII, n. XVII.

CARSON, J. S. Introduction to modeling and simulation. In: WINTERSIM, 2003. **Proceedings of the 2003 Winter Simulation Conference**, S. Chick, P. J. Sánchez, D. Ferrin, D. J. Morrice, eds, 2003

CARVALHO, H. J. R. **Utilização de redes de Petri na modelagem e simulação de uma arquitetura robótica de controle para robôs móveis baseada em comportamentos**. 2006. Dissertação (Mestrado em Engenharia Mecânica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2006.

CERVO, A. L.; BERVIAN, P. A. **Metodologia científica**. 4ed. São Paulo: McGraw-Hill do Brasil, 1996.

CHIZZOTTI, A. **Pesquisa em ciências humanas e sociais**. Série 1. Escola, v. 16. São Paulo: Cortez Editora (Biblioteca da educação), 1991.

COLMANETTI, M. S. **Modelagem de sistemas manufatura orientada pelo custeio das atividades e processos**. 2001. Dissertação (Mestrado em Engenharia Mecânica) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2001.

COSTA, A. J. **Otimização do layout de produção de um processo de pintura de ônibus**. 2004. Dissertação (Mestrado) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

CONWAY, D. G.; VENKATARAMANAM, M. A. Genetic search and the dynamic facility layout problem. **Computational Operations Research**, v. 21, p. 955-960, 1994.

CPN Tools. Disponível em <<http://wiki.daimi.au.dk/cpntools/cpntools.wiki>>. Acesso em: 03 Out. 2009.

DROLET, J. R. **Scheduling Virtual Cellular Manufacturing Systems**. 1989. Thesis (Ph.D.) - School of Industrial Engineering, Purdue University, West Lafayette, Indiana, 1989.

DESROCHERS, A.; AL-JAAR, R. **Applications of Petri Nets in Manufacturing Systems: modeling, control, and performance analysis**. New York: IEEE Press, 1995.

EHRlich, P. J. **Engenharia Econômica: avaliação e seleção de projetos de investimento**. 5ª ed. São Paulo: Atlas, 1989.

EREL, E. et al. New heuristic for the dynamic layout problem. **Journal of the Operational Research Society**, v. 54, p. 1275-1282, 2003.

FRANCIS, R. L. F.; WHITE, J. A. **Facility layout and location: An Analytical approach**. New Jersey: Prentice-Hall, 1974.

Goldberg, D. E. **Genetic algorithm in search, optimization, and machine learning**. Canada: Addison-Wesley, 1989.

GORDON, G. **System simulation**. 2nd ed. New Jersey: Prentice-Hall, 1969.

GORGULHO JÚNIOR, J. H. C. **Análise do desempenho dos arranjos físicos distribuídos em ambiente de programação de tarefas com flexibilidade de seqüência de fabricação**. 2006. Tese (Doutorado em Engenharia Mecânica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2006.

GORGULHO JÚNIOR, J. H. C.; GONÇALVES FILHO, E. V. Análise do desempenho dos arranjos físicos distribuídos operando sob roteamento de peças com flexibilidade de sequenciamento. **Revista Gestão Industrial**, v. 3, n. 1, p. 1-12, 2007.

GUPTA, J. N. D. Part dispatching and machine loading in flexible manufacturing systems using central queues. **International Journal of Production Research**, v. 37, n. 6, p. 1427-1435, 1999.

HAM, I. et al. **Group Technology: Applications to production management**. The Pennsylvania State University: International Series in Management Science/Operations Research, 1985.

HEGENBERG, L. **Etapas da investigação científica: Leis, teorias, método**. 2^a. ed. São Paulo: EDUSP, 1976.

HERAGU, S. Modeling the machine layout problem. **Computers & Industrial Engineering**, v. 19, n. 1-4, p. 294-298, 1990.

IRANI, S. A. **Handbook of cellular manufacturing systems**. New York: Editora John Wiley & Sons, 1999.

ISLIER, A. A. A genetic algorithm approach for multiple criteria facility layout design. **International Journal of Production Research**, v. 36, pp. 1549-1569, 1998.

JENSEN, K. **Colored Petri nets: basic concepts, analysis methods and practical use**. Berlin: Springer-Verlag, 1991.

KUSIAK, A.; HERAGU, S. S. The facility layout problem. **European Journal of Operational Research**, v. 29, p. 229-251, 1987.

LAHMAR, M.; BENJAAFAR, S. Design of dynamic distributed layouts. **INFORMS Fall Meeting**, Session TC17 (Advances in Factory Layouts), Nov., Miami, FL, 2001.

LAHMAR, M.; BENJAAFAR, S. Design of dynamic distributed layouts. In: ANNUAL INDUSTRIAL ENGINEERING RESEARCH CONFERENCE, Orlando, Florida, 2002. **Proceedings of the 11th Annual Industrial Engineering Research Conference (IERC)**, March 19-21, Orlando, Florida, 2002

LAHMAR, M.; BENJAAFAR, S. Design of distributed layouts. **IEE Transactions**, v. 37, p. 303-318, 2005.

LAW, A. M.; KELTON, W. D. **Simulation modeling and analysis**. 2. ed. New York: Mc Graw-Hill, 1991.

LOBÃO, E. C.; PORTO, A. J. V. Proposta de sistematização de estudos de simulação. (CD ROM). In: ENCONTRO NACIONAL DE ENGENHARIS DE PRODUÇÃO, Gramado, RS, 1997. **Anais do Enegep 1997**, Gramado, RS, 1997

MAGGIO, E. G. R. **Uma heurística para a programação da produção de sistemas flexíveis de manufatura usando modelagem em redes de Petri**. 2005. Dissertação (Mestrado em Ciências da Computação) - Universidade Federal de São Carlos, São Carlos, 2005.

MÁQUINAS E METAIS. Ano XLI, n. 469, Ed. Aranda, 2005.

MARIA, A. Introduction to modeling and simulation. In: THE 1997 WINTER SIMULATION CONFERENCE, Piscataway, New Jersey. **Proceedings of the Wintersim**. Piscataway, New Jersey, 1997.

MONTGOMERY, D. C. **Design and analysis of experiment**. 4. ed. New York: Editora John Wiley & Sons, 1997.

MONTREUIL, B.; LAFORGE, A. Dynamic layout design given a scenario tree of probable futures. **European Journal of Operational Research**, v. 63, p. 271-286, 1992.

MONTREUIL, B. et al. (1993). **Layout for chaos – Holographic layout of manufacturing systems operating in highly volatile environments**. Québec: Université Laval, 1993. Document de Travail 93-53.

MONTREUIL, B. et al. **Holographic layout of manufacturing systems**. Québec: Laval University, 1991. Technical report No. 91-76.

MONTREUIL, B. et al. Fractal layout organization for job shop environments. **International Journal of Production Research**, v. 37, n. 3, p. 501-521, 1999.

MORÁBITO, R. **Análise de curvas de *trade-off* baseada em teoria de redes de filas para o projeto e planejamento de sistemas discretos de manufatura**. 1998. Tese (livre-docência em engenharia de produção) - Universidade de São Paulo, São Carlos, 1998.

PITOMBEIRA NETO, A. R. Procedimento para a geração dos arranjos físicos distribuídos baseado em otimização da simulação e algoritmos genéticos. In: XXVII ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO. **Anais do Enegep 2007**, Foz do Iguaçu, PR, 2007

PITOMBEIRA NETO, A. R. et al. Design of distributed layouts with operational efficiency considerations. In: 19th INTERNATIONAL CONFERENCE ON PRODUCTION RESEARCH, ValParaiso, Chile, 2007. **Proceedings the 19th ICPR**, ValParaiso, Chile, 2007.

OLIVÉRIO, J. L. **Projeto de fábrica. Produtos, processos e instalações industriais**. São Paulo: IBLC – Instituto Brasileiro do Livro Científico Ltda., 1985.

PALEKAR, U. S. et al. Modeling uncertainties in plant layout problems. **European Journal of Operational Research**, v. 63, p. 347-359, 1992.

PORTO, A. J. V. **Desenvolvimento de um método de integração do planejamento do processo de fabricação e do planejamento e controle da produção, baseado na flexibilidade do processo de fabricação**. 1990. Tese (Doutorado em Engenharia Mecânica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 1990.

PRATA, B. A. **Avaliação de desempenho operacional de terminais portuários de carga unitizada: Uma aplicação das redes de Petri coloridas**. 2006. Monografia (Graduação) - Centro de Tecnologia, Universidade Federal do Ceará, 2006.

REAES, P. A. **Comparação de desempenho entre as configurações celular, funcional e celular virtual através de simulação e projeto de experimentos**. 2008. Tese (Doutorado) - Universidade Federal de Santa Catarina, 2008.

RHEAULT, M. et al. Physically reconfigurable virtual cells: a dynamic model for a highly dynamic environment. **Computers Industrial Engineering**, v. 29, n. 1-4, p. 221-225, 1995.

ROSENBLATT, M. J. The dynamics of plant layout. **Management Science**, v. 32, p. 76-86, 1986.

RUIZ, J. A. **Metodologia científica: guia para eficiência nos estudos**. 6^a. ed. São Paulo: Editora Atlas, 2006.

SAGAN, H. **Space-filling curves**. New York: Springer-Verlag, 1991.

SAWIK, T. Dispatching scheduling of machines and vehicles in a flexible manufacturing system. In: PROCEEDINGS OF EMERGING TECHNOLOGIES AND FACTORY AUTOMATION, v. 2, p. 3-13, 1995

SHIH, Y. C. **Simulação do processo de retirada de itens em CDP: um estudo de caso em empresa do ramo automotivo**. 2005. Dissertação (Mestrado em Engenharia Mecânica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2005.

SLACK, N. et al. **Administração da produção**. Edição Compacta. São Paulo: Editora Atlas, 1999.

SOCIETY OF MANUFACTURING ENGINEERS **Die design handbook**. 2 ed. New York: McGraw-Hill Book Company, 1965.

SUBRAMANIAM, V. et al. Machine selection rules in a dynamic job shop. **International Journal of Advanced Manufacturing Technology**, v. 16, p. 902-908, 2000.

SULE, D. R. **Manufacturing facilities: location planning and design**. Boston: PWS, 1994.

SURESH, N. C.; SLOMP, J. Performance comparison of virtual cellular manufacturing with functional and cellular layouts in DRC settings. **International Journal of Production Research**, v. 43, n. 5, p. 945-979, 2005.

TATE, D. M.; SMITH, A. E. Unequal-area facility layout by genetic search. **IIE Transactions**, v. 27, p. 465-472, 1995.

TOMPKINS, J. A. et al. **Facilities planning**. 2.ed. New York: John Wiley & Sons, 1996.

ULGEN, O.; GUNAL, A. **Simulation of manufacturing and material handling systems: Handbook of simulation**. Jerry Banks ed. New York: John Wiley & Sons, 1998.

WEBSTER, D. B.; TYBERGYHEIN, M. B. Measuring flexibility of jobshop layouts. **International Journal of Production Research**, v. 18, n. 1, p. 21-29, 1980.

WINSTON, W. L. **Operations research: applications and algorithms**. Belmont: International Thomson Publishing, 1994.

YAMADA, M. C. **Especificação de experimentos, modelos e interfaces padrões para o apoio ao planejamento da produção no setor sucroalcooleiro, via simulação**. 2004. Tese (Doutorado em Engenharia Mecânica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2004.

ZAGONEL, E. **Implantação do fluxo unitário de peças numa célula de usinagem: estudo de caso por meio da simulação.** 2006. Dissertação (Mestrado em Engenharia de Produção) - Setor de Tecnologia, Universidade Federal do Paraná, Curitiba, 2006.

BIBLIOGRAFIA CONSULTADA

BENJAAFAR, S. Design of flexible layouts for manufacturing systems. In: ENGINEERING MANAGEMENT CONFERENCE. **Proceedings of the 1995 IEEE**, p. 421-427, 1995.

BENJAAFAR, S. Design of manufacturing plant layouts with queueing effects. In: INTERNATIONAL CONFERENCE ON ROBOTICS & AUTOMATION, 1998, Leuven, Belgium. **Proceedings of the 1998 IEEE**, Leuven, Belgium, 1998

BENNETT, B. S. **Simulation fundamentals**. London: Prentice-Hall, 1995.

BERTHOMIEU, B.; DIAZ, M. Modeling and verification of time dependent systems using time petri nets. **IEEE Transactions on Software Engineering**, v. 17, n.3, 1991.

CARDOSO, J.; VALETTE, R. **Redes de Petri**. Florianópolis: UFSC, 1997.

GONÇALVES FILHO, E. V. **Sistemas de manufatura**: projeto do arranjo-físico. Apostila para alunos de pós graduação. São Carlos: Escola de Engenharia de São Carlos, Universidade de São Paulo, 2005

GROOVER, M. P.; ZIMMERS, E. W. JR. **CAD/CAM: computer-aided design and manufacturing**. New Jersey: Prentice-Hall, 1984.

HERAGU, S. **Facilities Design**. PWS publishing company, 1997.

HERAGU, S. S.; KOCHHAR, J. S. **Material handling issues in adaptative manufacturing systems**. The materials handling engineering division 75th anniversary commemorative volume, New York: ASME, 1994.

MURATA, T. Petri Net: properties, analysis. **Proceedings of the IEEE**, v. 77, n. 4, p. 541-579, April.1989

PORTO, A. J. V. **Simulação de sistemas de discretos**. Disciplina do Programa de Pós-Graduação do Departamento de Engenharia Mecânica. Notas de aula. São Carlos: Escola de Engenharia de São Carlos, Universidade de São Paulo, 2000.

PRITSKER, A. A. B. **Introduction to simulation and SLAM II**. 4th ed. New York: John Wiley & Sons, 1995.

SHIH, Y. C. et al. Desbalanceamento do número de itens das listas de coleta para reduzir o tempo do processo de retirada de itens. **Produto & Produção**, v. 7, n. 2, p. 27-38, 2004.

SINGH, N.; RAJAMANI, D. **Cellular manufacturing systems: design, planning and control**. London: Chapman & Hall, 1996.

YAMADA, M. C. et al. Aplicação dos conceitos de modelagem e de redes de Petri na análise do processo produtivo da indústria sucroalcooleira. **Pesquisa Agropecuária Brasileira**, v. 37, n. 6, p. 809-820, 2002.

APÊNDICE 1 DESCRIÇÃO DO MODELO DE SIMULAÇÃO DA FÁBRICA FICTÍCIA

O modelo de simulação proposto é composto por uma página principal a qual estão associadas subpáginas iniciais, das quais, podem derivar, sucessivamente, outras subpáginas. No contexto dessa hierarquia, as subpáginas correspondem a submodelos em redes de Petri de alto nível, mais particularmente, redes de Petri coloridas, nas quais são traduzidas as necessidades lógicas presentes nos sistemas de produção em estudo.

A Figura A1.1 apresenta a interface do editor e simulador de redes de Petri coloridas, CPN tools versão 2.2.0, e a primeira página do modelo a ser apresentado.

Figura A1.1 - Interface do CPN tools e a estrutura hierárquica de páginas (Fonte: autor)

Na Figura A1.1, tem-se o modelo de simulação de sistemas de manufatura do arranjo físico funcional e do arranjo físico distribuído em redes de Petri coloridas com a sua marcação inicial, ou seja, no seu estado inicial. Nas seções seguintes desse tópico, a sua hierarquia de páginas é descrita com maiores detalhes, a começar pela sua página principal. Os nomes definidos para os elementos de modelagem, em especial, os lugares e as transições, têm validade apenas para fins de descrição. Daí

que, em alguns momentos, podem ser repetidos em submodelos diversos sem prejudicar a figuração intentada.

A1.1 ESTRUTURA HIERÁRQUICA DO MODELO DE SIMULAÇÃO: A DEFINIÇÃO DA PÁGINA PRINCIPAL

O nível principal da página hierárquica do modelo proposto é apresentado na Figura A1.2.

Figura A1.2 - Página hierárquica principal e suas ramificações (Fonte: autor)

Na Figura A1.2, o nó raiz *ManufacturingLayoutSimulationModel* define a página principal da qual derivam as subpáginas *Transportation*, *ModelLogicControl* e *ResourcesAllocation&OrdersInitialization&Processing*, articulando-as entre si.

A subpágina *ResourcesAllocation&OrdersInitialization&Processing* deve ser entendida como um submodelo que define a parte da modelagem concernente à lógica de entrada de peças, da alocação dos diversos processos de transformação e dos recursos produtivos para aquelas sucessivamente.

Já, a subpágina *Transportation* deve ser entendida como um submodelo que define a parte da modelagem concernente à lógica geral de movimentação das peças ao longo do chão de fábrica, ou seja, a lógica de movimentação dos transportadores iniciais, dos transportadores intermediários e dos transportadores finais.

Por fim, a subpágina *ModelLogicControl* deve ser entendida como um submodelo que define a parte da modelagem concernente à lógica geral das tomadas de decisão que articulam as diversas partes do modelo representado entre si, inclusive do que diz respeito da estrutura de controle dos diversos eventos possíveis, garantindo assim a sua unidade lógico-formal como um todo.

No item A1.2., têm-se as subpáginas que compõem o submodelo *ResourcesAllocation&OrdersInitialization&Processing*. Já no item A1.3, as subpáginas que compõem o submodelo *Transportation* são apresentadas. Por fim, no item A1.4, são ilustradas as subpáginas que compõem o submodelo *ModelLogicControl*.

A1.2 MODELAGEM DA ENTRADA, DA ALOCAÇÃO E DA DETERMINAÇÃO DOS PROCESSOS DE TRANSFORMAÇÃO DOS PEDIDOS DE PEÇAS

A estrutura hierárquica particular da subpágina *ResourcesAllocation&OrdersInitialization&Processing* é apresentada na Figura A1.3.

Figura A1.3 - Subpágina *ResourcesAllocation&OrdersInitialization&Processing* e suas ramificações (Fonte: autor)

Na Figura A1.3, o nó raiz *ResourcesAllocation&OrdersInitialization&Processing* define a subpágina da qual derivam as subpáginas *OrdersInput*, *ProcessDefinition*, *MachQueueSubQueue*, *MachOutQueueSemiProdInAdd* e *ToolMachSetup*, articulando-as entre si e com as outras partes do modelo.

A subpágina *OrdersInput* corresponde ao submodelo que define a parte da modelagem concernente à lógica de entrada de pedidos de peças. É por intermédio de tal submodelo que, efetivamente, a simulação do sistema representado inicia-se. Com efeito, a entrada de pedidos se dá de tal maneira que a estrutura do modelo baseado em redes de Petri coloridas é reaproveitada, uma vez que essa flexibilidade está presente já como uma das características definidoras desse tipo de técnica de modelagem. Para facilitar o entendimento, será descrito o percurso de um lote de peças ao longo do modelo. No entanto, vale frisar que, em regime, simultaneamente, lotes de peças diversas percorrem as diversas partes do modelo. A Figura A1.4 apresenta o encadeamento lógico-causal correspondente.

Na Figura A1.4, o lugar de entrada de marcas *Orders*, localizado na parte inferior desta figura, fornece as marcas correspondentes aos diversos lotes de cada tipo de peça dos pedidos. A presença de pelo menos uma dessas marcas é a condição de verdade mais fundamental para habilitar a transição *OrdersInput* ao disparo. A retirada de marcas do lugar *Orders* e a alocação posterior das mesmas no

lugar *OrderInfo* é realizada, de maneira iterativa, pela estrutura de controle formada pelos lugares *PieceCounter*, *OrderQtd* e *OrderCounter* juntamente com a transição *OrdersInput*. Cada disparo da transição *OrdersInput* corresponde à entrada de um lote no modelo, mas a afiguração da entrada “real” de peças no sistema de produção é feita pela transição *OrderQueueInput*. Essa, na medida em que vão se dando os disparos de *OrdersInput*, com a alocação consequente no lugar *OrderInfo*, fica habilitada ao disparo.

Figura A1.4 - Submodelo da subpágina *OrdersInput* (Fonte: autor)

Com o disparo da transição *OrderQueueInput*, a função *FillOutOrderBuf* é chamada. Tal função é responsável pelo preenchimento de listas de pedidos (*orderbuf*) cujos elementos são os lotes de cada tipo de peça correspondentes (*orinfo*), que são consequentemente alocadas como marcas de listas no lugar *OrderQueue*. O código da função concernente em CPNML é apresentado a seguir.

```
fun FillOutOrderBuf(orinfo,orderbuf) =  
  let  
    fun AuxFillOrder(aux1,aux2) =  
      if aux2 = nil then aux1::aux2  
      else
```



```

    AuxFillOrder(aux1,tl(aux2));
in
    if orderbuf = nil then orinfo::orderbuf
    else
        orderbuf^^AuxFillOrder(orinfo,tl(orderbuf))
    end;

```

Tão logo, no lugar *OrderQueue*, as marcas de listas para cada pedido em particular são completadas com o número total de lotes correspondentes, a função *CheckOrderStatus* é chamada com o disparo da transição *OrdersInput*. Para cada pedido, tal função faz a checagem da quantidade de lotes de peças que entraram com as que deveriam entrar ($pc = oq$), resultando numa marca com a seguinte estrutura, $1^*(or, "OrderCompleted")$, em que a variável de arco *or* corresponde ao número do pedido, sendo alocada no lugar *StatusOrder*. Por conseguinte, a transição *OrdersIn* fica habilitada ao disparo e, com isso, inicia-se o processo de entrada de peças por intermédio dos transportadores iniciais, então, devidamente carregados com os pedidos correspondentes.

Já a subpágina *ProcessesDefinition* corresponde ao submodelo que define a parte da modelagem concernente à lógica de determinação e de articulação das operações iniciais e subsequentes de transformação por que passam os diversos tipos de peças. A Figura A1.5 apresenta a estrutura em redes de Petri coloridas do submodelo supracitado.

Na Figura A1.5, seja para o caso do arranjo físico funcional ou para o caso do arranjo físico distribuído, todas as operações iniciais de transformação são definidas pela função *DefProclniProd*. Para tal, uma vez determinada a réplica de máquina correspondente à primeira operação, por um critério de decisão pré-determinado, o lugar de entrada de marcas *Order_Seq* recebe as marcas às quais estão associadas informações acerca do lote de cada tipo de peça (*orinfo*) e do local de trabalho em que se encontra a réplica da máquina escolhida (*loc*). Consequentemente, a transição *T1* fica habilitada ao disparo.

O disparo da transição *T1* estabelece a condição de verdade para habilitar o disparo da transição *DefProclni*, qual seja, ao menos uma marca alocada no lugar *P1*, contendo as informações acerca do lote (*orinfo*), do tipo de peça (*ProdT*), da quantidade a ser produzida (*QtdProdT*) e do local de trabalho (*loc*). Consequentemente, com o disparo de *DefProclni*, uma marca é alocada no lugar *ProcQueue*, contendo informações acerca do lote (*orinfo*), do tipo de peça (*ProdT*), da sequencia da operação a ser realizada (*OPTlistOPTon*) – no caso em particular, “*OPTlistOPTon = 1*” –, da quantidade a ser produzida (*QtdProdT*) e do local de trabalho (*loc*).

Já no caso da determinação das operações subsequentes de transformação das peças, há aspectos comuns e específicos entre o que concerne ao arranjo físico funcional ou o que concerne ao arranjo físico distribuído.

Figura A1.5 - Submodelo da subpágina *ProcessesDefinition* (Fonte: autor)

Descrevendo inicialmente o que há de comum entre eles, uma vez determinada a réplica de máquina correspondente à operação subsequente, por um critério de decisão pré-determinado, o lugar de entrada de marcas *ProdSemi* fornece as marcas às quais estão associadas informações acerca do lote (*orinfo*), do tipo de peça (*ProdTSemi*), da sequência da operação anterior realizada (*OPTlistSemi*), da quantidade a ser produzida (*QtdProdT*) e do local de trabalho da réplica de máquina em que se deu o último processamento (*locIn*). Consequentemente, a transição *DefProcSemi* fica habilitada ao disparo.

O disparo da transição *DefProcSemi* estabelece a condição de verdade mais fundamental para habilitar o disparo tanto da transição *T4* quanto da transição *T5*, quais sejam, ao menos uma marca alocada no lugar *P7*, contendo informações acerca do lote (*orinfo*), do tipo de peça (*ProdT*), da sequência da operação a ser realizada (*OPTlistOPTon*) – no caso em particular, “*OPTlistOPTon* = *OPTlistSemi* + 1” –, da quantidade a ser produzida (*QtdProdT*) e do local de trabalho da réplica de máquina em que se deu o último processamento (*locIn*).

A partir daqui, mostram-se as diferenças entre a lógica de determinação das operações das peças para o caso do arranjo físico funcional e do arranjo físico distribuído.

Passando primeiramente para a descrição particular do que concerne a tal lógica do arranjo físico funcional, para que a transição $T4$ fique habilitada, efetivamente, o lugar de entrada de marcas $P4$

deve fornecer a informação acerca do transportador intermediário escolhido anteriormente para a movimentação do lote de peças.

Assim, com o disparo de *T4*, o lugar de saída de marcas *P5* recebe uma marca contendo informações sobre o lote (*orinfo*), o tipo de peça (*ProdT*), a sequência da operação a ser realizada (*OPTlistOPTon*) e, paralelamente, o lugar *P3* recebe uma marca a qual estão associadas as informações acerca do lote (*orinfo*), do tipo de peça (*ProdT*), da sequência da operação a ser realizada (*OPTlistOPTon*), da quantidade a ser produzida (*QtdProdT*) e do local de trabalho da réplica de máquina em que se deu o último processamento (*locIn*).

A marca do lugar *P5* serve para a determinação da réplica de máquina subsequente, por algum critério pré-determinado, num submodelo específico, de tal maneira que, após ocorrer, o lugar de entrada de marcas *Transp_ProdT* recebe uma marca contendo, especialmente, a informação do local de trabalho em que se encontra a réplica da máquina escolhida onde se dará o próximo processo (*locOut*). Consequentemente, a transição *T2* fica habilitada ao disparo.

Com o disparo de *T2*, uma marca é alocada no lugar *ProcQueue*, contendo informações acerca do lote (*orinfo*), do tipo de peça (*ProdT*), da sequência da operação a ser realizada (*OPTlistOPTon*), da quantidade a ser produzida (*QtdProdT*) e do local de trabalho correspondente (*locOut*).

Passando agora para a descrição particular do que concerne à lógica do arranjo físico distribuído, com o disparo da transição *T5*, uma marca é alocada no lugar *P6*, contendo informações acerca do lote (*orinfo*), do tipo de peça (*ProdT*) e da sequência da operação a ser realizada (*OPTlistOPTon*) e, paralelamente, uma marca é alocada no lugar *P3*, contendo informações acerca do lote (*orinfo*), do tipo de peça (*ProdT*), da sequência da operação a ser realizada (*OPTlistOPTon*), da quantidade a ser produzida (*QtdProdT*) e do local de trabalho da réplica de máquina em que se deu o último processamento (*locIn*). Consequentemente, a transição *T6* fica habilitada ao disparo.

Na Figura A1.5, o lugar de entrada e de saída de marcas *P2* pode ser visto como um repositório de marcas com as informações quantitativas e qualitativas de todas as células virtuais que poderão realizar-se para cada lote de peças. Por isso mesmo, com o disparo da transição *T6*, a função *List.nth* retorna, a partir das variáveis de arco *OPTlistOPTon ListLocal*, a informação do local de trabalho da réplica da máquina onde se dará a próxima operação. Assim, o lugar *Transp_ProdTDist* recebe marcas contendo informações sobre o lote (*orinfo*), a sequência da operação a ser realizada (*OPTlistOPTon*) e o local de trabalho correspondente (*locOut*). Consequentemente, a transição *T3* fica habilitada ao disparo.

Com o disparo de *T3*, uma marca é alocada no lugar *ProcQueue*, contendo informações acerca do lote (*orinfo*), do tipo de peça (*ProdT*), da sequência da operação a ser realizada (*OPTlistOPTon*), da quantidade a ser produzida (*QtdProdT*) e do local de trabalho (*locOut*).

Por sua vez, a subpágina *ToolMachSetup* corresponde ao submodelo que define a parte da modelagem concernente à lógica de verificação e de realização, caso necessária, da troca de

ferramentas em geral nas máquinas – válida tanto para o caso de um arranjo físico funcional quanto para o caso de um arranjo físico distribuído.

Antes que as peças venham a ser processadas nas máquinas determinadas, pode ser que o estado do bloco de ferramentas tenha que ser alterado, demandando a incorporação naquele da ferramenta requisitada para a realização da operação correspondente. A subpágina *ToolMachSetup* e suas ramificações estão ilustradas na Figura A1.6.

Figura A1.6 - Subpágina *ToolMachSetup* e suas ramificações (Fonte: autor)

Na Figura A1.6, a subpágina *CheckToolMach* deriva da subpágina *ToolMachSetup*, que, por sua vez, deriva da subpágina *ChangeMachToolList*. O submodelo da subpágina *ToolMachSetup* inicia a etapa de checagem da necessidade da troca de ferramenta, recebendo diretamente da subpágina *ProcessDefinition* as marcas correspondentes aos processos iniciais e subsequentes, bem como articula a comunicação entre ela mesma e a subpágina seguinte, *MachQueueSubQueue*. A Figura A1.7 ilustra tal encadeamento lógico-causal.

Na Figura A1.7, o lugar de entrada de marcas *ProcQueue*, localizado na parte esquerda desta figura, recebe marcas contendo informações acerca do lote (*orinfo*), do tipo de peça (*ProdT*), da sequência da operação a ser realizada (*OPTlistOPTon*), da quantidade a ser produzida (*QtdProdT*) e do local de trabalho correspondente (*loc*).

A presença de pelo menos uma marca no lugar *ProcQueue* já é condição suficiente para habilitar as transições *T1* e *T2* ao disparo. A transição *T1* corresponde ao início do processo de troca de ferramenta para o caso do arranjo físico funcional, enquanto a transição *T2* para o caso do arranjo físico distribuído. O disparo de qualquer uma dessas transições chama a função *DefMachToolListProd*. A partir das variáveis de arco *ProdT* e *OPTlistOPTon*, tal função retorna o tipo de ferramenta (valor inteiro) necessária para a execução do processo de transformação da peça.

Figura A1.7 - Submodelo da subpágina *ToolMachSetup* (Fonte: autor)

Por conseguinte, uma marca é alocada no lugar *ToolProdIn*, contendo informações sobre o lote (*orinfo*), o local de trabalho (*loc*) e a ferramenta correspondente (*ToolMach*). Consequentemente, o submodelo da subpágina *CheckToolMach* recebe tais informações para verificar se a ferramenta requisitada encontra-se no bloco de ferramentas da réplica da máquina no local de trabalho específico (*loc*). A Figura A1.8 ilustra a estrutura lógica do submodelo concernente.

Na Figura A1.8, caso seja constatada a presença da ferramenta (*ToolMach*) no bloco de ferramentas representado pela variável de arco do tipo lista *TList*, ou seja, se a função *mem* da guarda da transição *T2*, ao varrer *TList*, retornar o valor booleano *true*, uma marca é alocada no lugar *P4*, contendo informações acerca do lote (*orinfo*) e do local de trabalho (*loc*). Caso contrário, o submodelo da subpágina *ChangeMachToolList* inicia a troca efetiva de ferramenta. A Figura A1.9 ilustra a estrutura lógica do processo concernente.

Figura A1.8 - Submodelo da subpágina *CheckToolMach* (Fonte: autor)

Figura A1.9 - Submodelo da subpágina *CheckMachToolList* (Fonte: autor)

Na Figura A1.9, os lugares de entrada de marcas *P1* e *P3*, armazenam, respectivamente, marcas contendo informações acerca do bloco de ferramentas corrente da réplica da máquina (*TList*) e do tipo de ferramenta requisitada pela peça (*ToolMach*). Com o disparo da transição *T1*, considerando que a função *mem* da guarda da transição *T1*, ao varrer *TList*, retornou o valor booleano *false*, as condições de verdade para habilitar a transição *T2* ao disparo são satisfeitas, de sorte que, no lugar *P6*, a marca alocada não contém o primeiro elemento da lista *TList*, uma vez que o mesmo foi eliminado pela função *tl*.

O disparo da transição *T2* faz com que a nova lista *TList* seja atualizada com a ferramenta necessária *ToolMach*, sendo esta colocada ao final daquela pela estrutura funcional “*TList*^[*ToolMach*]”. Paralelamente, a função *SetupTime* é chamada para temporizar a marca alocada, consecutivamente, no lugar *P7*. Ao final do tempo de troca de ferramenta, uma marca é alocada no lugar *ToolMachDef*, com o novo estado do bloco de ferramentas (*TList*) na réplica de máquina no local de trabalho correspondente (*loc*).

Consecutivamente, a subpágina *MachQueueSubQueue* corresponde ao submodelo que define a parte da modelagem concernente a uma das estruturas lógicas de atualização quantitativa das diversas filas de entrada e saída de peças das máquinas nos locais de trabalho. Assim, dada a lista de valores inteiros *MinQ*, cada elemento dessa lista representa uma fila de entrada de peças àquelas réplicas de máquinas. Nesse caso em particular, dada uma máquina qualquer, se esta se encontra disponível para ser carregada com uma peça específica, a sua fila de entrada sofre um decremento correspondente à uma caixa – no arranjo físico funcional – e decremento unitário – no caso do arranjo físico distribuído. Tão logo tal atualização é concluída, dá-se a determinação do tempo de operação a ser realizada na máquina. Para tal, a função *TimeProdOPT* é definida, como ilustrada na Figura A1.10.

Na Figura A1.10, as variáveis de arco *ProdT* e *OPTlistOPTon*, correspondendo, respectivamente, ao tipo de peça e ao processo de transformação a ser realizado nessa, possibilitam a determinação do tempo de processamento concernente – a partir de uma distribuição estatística pré-definida, no caso desse estudo, uma distribuição uniforme – sendo este valor resultante multiplicado pela quantidade de peças no lote a ser processado, representada pelo variável de arco *QtdProdT*. Com o disparo da transição *TimeProdOPTDef*, uma marca temporizada é alocada no lugar *Posto*, contendo as informações referentes ao pedido de peça (*orinfo*), à quantidade de peças (*QtdProdT*) e ao local de trabalho em que se encontra a réplica da máquina escolhida (*loc*), bem como acerca da operação a ser realizada, sendo esta última determinada pela função *OperacaoOn*.

Figura A1.10 - Submodelo da subpágina *MachQueueSubQueue* e a função *TimeProdOPT*
(Fonte: autor)

Por fim, a subpágina *MachOutQueueSemiProdInAdd* corresponde ao submodelo que define a parte da modelagem também concernente a uma das estruturas lógicas de atualização quantitativa das diversas filas de entrada e saída de peças das máquinas nos locais de trabalho. Assim, dada a lista de valores inteiros *MSubQ*, cada elemento dessa lista representa uma fila de saída de peças àquelas réplicas de máquinas. Nesse caso em particular, dada uma máquina qualquer, se esta conclui a operação que lhe compete, a sua fila de saída sofre um incremento correspondente à uma caixa – no caso do arranjo físico distribuído, tal incremento é unitário. A subpágina *MachOutQueueSemiProdInAdd* e suas ramificações estão ilustradas na Figura A1.11.

Figura A1.11 - Subpágina *MachOutQueueSemiProdInAdd* e suas ramificações (Fonte: autor)

Na Figura A1.11, a subpágina *UpdateLayoutDist* deriva da subpágina *UpdateLayoutPosAdd*, que, por sua vez, deriva da subpágina *MachOutQueueSemiProdInAdd*. O submodelo da subpágina *UpdateLayoutPosAdd* define a parte da modelagem concernente a uma das estruturas lógicas de atualização qualitativa das condições de estado das máquinas nos locais de trabalho para o caso do arranjo físico funcional. Já o submodelo da subpágina *UpdateLayoutDist* define a parte da modelagem concernente a uma das estruturas lógicas de atualização qualitativa das condições de estado das máquinas nos locais de trabalho para o caso do arranjo físico distribuído.

Descrevendo primeiramente o submodelo da subpágina *UpdateLayoutPosAdd*, dada a lista de valores inteiros *LMListIn*, cada elemento dessa lista representa o estado corrente de disponibilidade de cada uma das réplicas de máquinas do arranjo físico. Nesse caso em particular, dada uma máquina qualquer, se esta conclui a operação que lhe compete, a lista *LMListIn* é atualizada justamente na posição que afigura a posição daquela réplica na distribuição real das máquinas no chão de fábrica, pela substituição do valor corrente “0”, que se refere a indisponibilidade de máquinas, por um valor inteiro que a represente e que a distinga das demais. Por exemplo, a presença do número “6” na posição 10 da lista *LMListIn* significa que a réplica da máquina do tipo “6” no local de trabalho 10 encontra-se disponível; ao contrário, a presença do número “0” na posição 10 da lista *LMListIn* significa que a réplica da máquina do tipo “6” no local de trabalho 10 encontra-se indisponível. Tal encadeamento lógico-causal é apresentado na Figura A1.12.

Na Figura A1.12, o lugar de entrada e de saída de marcas *Layout_Main_Info* concentra a informação global dos estados sucessivos de disponibilidade da totalidade das réplicas de máquinas, a *LMListIn* supracitada. Por sua vez, o lugar de entrada de marcas *MaqPos_Upd* fornece a informação do tipo de máquina (*Mc*) e da posição da réplica da mesma na lista *LMListIn* a ser atualizada (*nPos*). Tais são as condições de verdade para habilitar o disparo da transição *T1* e iniciar o processo de atualização de *LMListIn*.

Com o disparo da transição *T1*, as funções *List.take* e *List.drop* cindem a lista *LMListIn* associada à variável de arco de mesmo nome, em duas listas de inteiros distintas, de tal maneira que, respectivamente, uma marca de lista é alocada no lugar *P1*, com os elementos do local de trabalho 1 até *nPos*, enquanto, paralelamente, uma marca de lista é alocada no lugar *P4*, com os elementos do local de trabalho *nPos* até *m*, onde *m* representa o número total de locais de trabalho do arranjo físico. Ademais, uma marca é alocada no lugar *P3*, com os parâmetros *Mc* e *nPos* associados. Consequentemente, a transição *T2* fica habilitada.

Com o disparo da transição *T2*, a função *List.take* é novamente chamada. Porém, desta vez, ela toma como argumentos a marca presente no lugar *P1*, por intermédio da variável de arco *LMListInTakeAdd*, e a marca do lugar *P3*, por intermédio das variáveis de arco *Mc* e *nPos*. Disso, resulta uma marca de lista com os elementos do local de trabalho 1 até “*nPos* – 1”, que é alocada no seu lugar de saída *P2*. Consequentemente, a transição *T3* fica habilitada ao disparo.

Figura A1.12 - Submodelo da subpágina *UpdateLayoutPosAdd* (Fonte: autor)

Com o disparo da transição *T3*, a função *ins* é chamada, tomando como argumentos a marca presente no lugar *P2*, por intermédio da variável de arco *LMListInTakeAdd*, e a marca do lugar *P3*, por intermédio das variáveis de arco *Mc* e *nPos*. Disso, resulta uma marca de lista em que lhe é inserida um novo elemento final, cujo valor *Mc* corresponde ao tipo de máquina a ser disponibilizada, sendo, consecutivamente, alocada no lugar *P5*. Consequentemente, a transição *T4* fica habilitada ao disparo.

Por fim, com o disparo da transição *T4*, tem-se a recomposição da lista global dos estados de disponibilidade das réplicas de máquinas (*LMListIn*).

Passando agora para a descrição do submodelo da subpágina *UpdateLayoutDist*, o que a distingue da descrição anterior é o fato de que a atualização da lista *LMListIn* deverá ocorrer, iterativamente, tantas vezes quanto for o número de máquinas que compõe a célula virtual determinada para o tipo de peça. A Figura A1.13 apresenta o encadeamento lógico-causal concernente.

Na Figura A1.13, o lugar de entrada e de saída de marcas *P2* armazena as informações sobre as células virtuais possíveis para cada lote de peças. Já o lugar *P1* diz respeito da quantidade de peças do lote que ainda faltam percorrer a célula virtual correspondente (*qd2*). Com efeito, quando qualquer uma das marcas estocadas em *P1*, tem seu valor de *qd2* igualado a zero, isto significa que as sequencias de operações de todas as peças do lote foram completadas. Consequentemente, a transição *T1* fica habilitada ao disparo.

Figura A1.13 - Submodelo da subpágina *UpdateLayoutDist* (Fonte: autor)

Com o disparo da transição *T1*, inicia-se o ciclo de atualização da lista *LMListIn* da mesma maneira que se dá no submodelo da subpágina *UpdateLayoutPosAdd*. Para tal, a composição de funções “*List.nth(LMListInCell,DefTranspLocation(hd(ListLocal))-1),DefTranspLocation(hd(ListLocal))*” retorna, iterativamente, a partir da variável de arco *ListLocal*, varrendo sucessivamente a variável de arco *LMListInCell*, marcas, que deverão ser alocadas no lugar *P4*, contendo informações sobre os tipos de máquinas a serem atualizados (*Mc*) e as posições das réplicas das mesmas na lista *LMListIn* a ser atualizada (*nPos*).

A1.3 MODELAGEM DO TRANSPORTE

A estrutura hierárquica particular da subpágina *Transportation* é apresentada na Figura A1.14.

Na Figura A1.14, o nó raiz *Transportation* define a subpágina da qual derivam as subpáginas *TranspInterFromTo*, *TranspInterMotionControl*, *TranspExitMotionControl* e *TranspInitialMotionControl*, articulando-as entre si e com as outras partes do modelo.

A subpágina *TranspInitialMotionControl* corresponde ao submodelo que define a parte da modelagem concernente à lógica de escolha da máquina responsável pela primeira operação e da

movimentação dos lotes de matéria-prima do L1 para aquela. Isso se dá de acordo com um critério estabelecido antecipadamente, de sorte que os diversos transportadores iniciais se movimentarão carregados para a réplica da máquina responsável pela primeira operação. O A1.3.1 apresenta a subpágina *TranspInitialMotionControl* e suas ramificações.

Figura A1.14 - Subpágina *Transportation* e suas ramificações (Fonte: autor)

Já as subpáginas *TranspInterFromTo* e *TranspInterMotionControl* correspondem, em conjunto, aos submodelos que definem a parte da modelagem concernente à lógica de escolha da estação de trabalho subsequente e da movimentação dos lotes de peças do local de trabalho onde ocorreu a última operação para o local de trabalho onde se dará o próximo processo de transformação. Mais particularmente, o submodelo da subpágina *TranspInterFromTo* é responsável por verificar e atualizar o estado de disponibilidade do transportador intermediário e de retornar, em caso favorável, o local em que o mesmo se encontra correntemente para o submodelo da subpágina *TranspInterMotionControl*, onde se dá, de fato, a determinação do local de trabalho subsequente. O A1.3.2 apresenta a subpágina *TranspInterMotionControl* e suas ramificações.

Por fim, a subpágina *TranspExitMotionControl* corresponde ao submodelo que define a parte da modelagem concernente à lógica de movimentação do transporte final, de acordo com um critério estabelecido antecipadamente, das peças ou dos lotes de peças acabadas das estações de trabalho para um local de expedição comum (*Local(k)*, onde $k = m + 1$, e m corresponde à quantidade total de máquinas). O A1.3.3 apresenta a subpágina *TranspExitMotionControl* e suas ramificações.

A1.3.1 Modelagem do transporte inicial

Considerando que as caixas (onde estão contidas as matérias-primas das peças a serem fabricadas) foram carregadas nos transportadores iniciais correspondentes, o local de trabalho inicial da primeira operação para cada lote de peças é determinado, ou seja, escolhe-se uma réplica de máquina

dentre um conjunto de máquinas de mesmo tipo - no arranjo físico funcional - ou verifica-se a disponibilidade das máquinas componentes da célula virtual - no arranjo físico distribuído. Em ambos os casos, o local de trabalho se iniciará a primeira operação para cada lote de peças é determinado a partir do submodelo da subpágina *TranspInitialMotionControl*. A subpágina *TranspInitialMotionControl* e suas ramificações estão ilustradas na Figura A1.15.

Figura A1.15 - Subpágina *TranspInitialMotionControl* e suas ramificações (Fonte: autor)

Na Figura A1.15, o submodelo da subpágina derivada *DefByLayType* faz a distinção entre as etapas seguintes para o caso do arranjo físico funcional e para o caso do arranjo físico distribuído. Com efeito, se for o caso de um arranjo físico funcional, o submodelo da subpágina *TranspInitLayout* é iniciado. Caso contrário, ou seja, trate-se de um arranjo físico distribuído, o submodelo *VirtCellDist* é iniciado antes do submodelo da subpágina *TranspInitLayout*.

Descrevendo primeiramente para o caso do arranjo físico distribuído, a subpágina *VirtCellDist* e suas ramificações estão ilustradas na Figura A1.16.

Figura A1.16 - Subpágina *VirtCellDist* e suas ramificações (Fonte: autor)

Na Figura A1.16, as informações sobre as diversas células virtuais e as informações sobre a disponibilidade corrente das máquinas são tomadas pelo submodelo da subpágina *VirtCellDist*. A Figura A1.17 apresenta o encadeamento lógico-causal concernente.

Figura A1.17 - Submodelo da subpágina *VirtCellDist* (Fonte: autor)

Na Figura A1.17, enquanto houver marcas nos lugares *P1* e *P2*, isto significa que ainda há lotes de peças a serem distribuídas nas células virtuais correspondentes e, por conseguinte, as transições *T1* e *T2* estarão habilitadas ao disparo.

O disparo da transição *T1* chama a composição de função “*List.nth(LMListInCell, DefTranspLocation(hd(ListLocal))-1)*”, que após verificar a lista de inteiros *LMListInCell*, retorna uma marca de valor inteiro no lugar *P3*, que corresponde a um dos tipos de máquinas que compõe a célula virtual no local de trabalho correspondente (*ListLocal*) ao tipo de peça sob exame. Já o disparo da transição *T2* chama a composição de função “*List.nth(LMListIn, DefTranspLocation(hd(ListLocal))-1)*”, retornando uma marca de valor inteiro no lugar *P5*, que corresponde ao estado de disponibilidade corrente da máquina requisitada. Caso as variáveis de arco *LTot* (= *LTot2*) e *LVCell* (= *LVCell2*) assumam valores diversos, ou seja, “*LTot = LVCell*” é falso, tal como se faz valer na expressão da guarda da transição *T3*, então o submodelo da subpágina *CheckListLocal* é iniciado. A Figura A1.18 apresenta o encadeamento lógico-causal concernente.

Figura A1.18 - Submodelo da subpágina *CheckListLocal* (Fonte: autor)

Na Figura A1.18, uma vez que “ $LTot2 \neq LVCell2$ ” é verdade, o disparo da transição $T1$ habilita o disparo da transição $T2$. O disparo consequente dessa última faz com que o conjunto de lotes de peças a serem fabricadas, correspondente a *orqueue2*, seja atualizada com o lote anteriormente analisado pela alocação de uma marca de lista no lugar $P10$.

Ao contrário, caso as variáveis de arco $LTot$ e $LVCell$ assumam valores iguais, ou seja, “ $LTot = LVCell$ ” é verdade, então o ciclo de checagem da disponibilidade de máquinas continua, até que todas sejam analisadas. Caso a expressão “ $LTot = LVCell$ ” retorne um valor booleano *true* para todas as máquinas checadas, isto significa que a célula virtual poderá ser realizada e o submodelo da subpágina *VirtualCellLayPosSub* é iniciado. A Figura A1.19 e a Figura A1.20 apresentam a estrutura lógica do submodelo correspondente.

Na Figura A1.19, o disparo da transição $T1$ inicia a alteração do estado de disponibilidade das réplicas de máquinas que compõem a célula virtual do lote de peças que será movimentado pelo transportador inicial. A estrutura apresentada corresponde a um processo iterativo que somente finda quando todas as máquinas passam a ficar indisponíveis para outros lotes de peças que, simultânea e sucessivamente, venham a requisitá-las. A Figura A1.20, por sua vez, apresenta a estrutura em que se efetiva tal iteração.

Figura A1.19 - Submodelo da subpágina *VirtualCellLaytPosSub* (Fonte: autor)

Figura A1.20 - Submodelo da subpágina *VirtualCellLaytPosSub* (continuação) (Fonte: autor)

Na Figura A1.20, cada disparo da transição *T2* corresponde à alocação de uma marca de valor inteiro com a informação acerca da posição na lista *LMListIn* em que se deverá mudar o valor corrente

para “0” pela sequencia de disparos das transições *T3*, *T5* e *T6*. Tal nulidade simboliza a indisponibilidade da réplica da máquina correspondente. Tão logo tal ciclo de atualizações finda, o submodelo da subpágina *ToolMachSetupVirt* é iniciado. A Figura A1.21 apresenta a subpágina *ToolMachSetupVirt* e suas ramificações.

Figura A1.21 - Subpágina *ToolMachSetupVirt* e suas ramificações (Fonte: autor)

Na Figura A1.21, tem-se a etapa de troca de ferramentas específica para o caso do arranjo físico distribuído. Antes que as peças sejam efetivamente alocadas na célula virtual que lhes corresponde, realiza-se algo similar ao transcorrido para a troca de ferramentas em geral – anteriormente descrita pelo conjunto das Figuras A1.6 a A1.9 –, mas que difere daquela no fato de afigurar a troca de ferramentas do conjunto de máquinas da célula virtual paralelamente (trocas simultâneas).

Passando então à descrição do submodelo da subpágina *TranspInitLayout*, tem-se a figuração da movimentação do transportador inicial para as réplicas de máquinas responsáveis pelas primeiras operações, tanto para o caso do arranjo físico funcional quanto para o caso do arranjo físico distribuído. A diferença entre os dois reside no fato de que, para o último, já se sabe, com antecipação e necessariamente, a sequencia dos locais para os quais o transportador inicial deverá dirigir-se, uma vez que se parte das células virtuais devidamente determinadas, enquanto, para o primeiro, a escolha da réplica é feita ao longo do tempo de maneira contingente. A Figura A1.22 mostra a subpágina *TranspInitLayout* e suas ramificações.

Na Figura A1.22, as derivações da subpágina *TranspInitLayout* decorrem, em primeiro lugar, do tipo de arranjo físico em estudo. Daí que a subpágina *TranspFromToInitVirtDist* tem a ver com a modelagem da movimentação do transportador inicial para as diversas células virtuais, que, inclusive, podem realizar-se simultaneamente. A Figura A1.23 apresenta a subpágina *TranspFromToInitVirtDist* e suas ramificações.

Figura A1.22 - Subpágina *TranspInitLayout* e suas ramificações (Fonte: autor)

Figura A1.23 - Subpágina *TranspFromToInitVirtDist* e suas ramificações (Fonte: autor)

Na Figura A1.23, da subpágina *TranspFromToInitVirtDist* deriva subpágina *MachQueueInUpdVirtDist* e, dessa última, a subpágina *MachInQueueAddDispVirtDist*. Ambas, em conjunto, são responsáveis pela atualização da lista de filas de entrada das réplicas na posição correspondente ao local de trabalho do próximo processo de transformação a ser realizado na peça correspondente, sendo um incremento unitário, tão logo o submodelo da subpágina

Na Figura A1.24, o lugar de entrada de marcas $P1$ fornece as informações sobre os lotes de peças a serem movimentados. A partir da informação da célula virtual para o tipo de peça específico, armazenada no lugar $P2$, com o disparo da transição $T1$, a função *DefTranspLocation* é chamada, retornando um valor inteiro correspondente ao local da máquina de processo de transformação inicial (*MachPos*). Consequentemente, a transição $T3$ fica habilitada ao disparo.

Figura A1.24 - Submodelo da subpágina *TranspFromToInitVirtDist* (Fonte: autor)

O disparo da transição *T3* chama a função responsável pelo cálculo da distância a ser percorrida pelo transportador inicial do local único de entrada de pedidos (*Local(1)*) para o local de realização da primeira operação nas peças do lote. O código da função concernente é apresentado a seguir.

```

fun DefTranspDistIn(MachPos,loc) =
if loc = Local(1) then DefMachDist(MachPos)
    else
        DefMachDistIn(loc,MachPos);

```

A função *DefTranspDistIn* é, na verdade, uma composição de funções, servindo de interface para as outras. Assim, caso a movimentação se dê do *Local(1)* para qualquer outro local no arranjo físico, a função *DefTranspDistIn* chama a função *DefMachDist*. Caso a movimentação se dê entre os locais de trabalho dentro do arranjo físico, a função *DefMachDistIn* é chamada. Como aqui se trata da movimentação do transportador inicial partindo do *Local(1)*, então o primeiro caso é satisfeito. Um código possível para a função *DefMachDistIn* é apresentado a seguir.

```
fun DefMachDist(p)=
let
    val MachListDist = [8,7,6,6,7,8,7,6,5,5,6,7,6,5,4,4,5,6,5,4,3,3,4,5];
in
    List.nth(MachListDist,p-1)
end;
```

A partir do valor da variável de arco *MachPos*, em que *p* assume seu valor, define-se a posição da lista de valores inteiros *MachListDist* da função *DefMachDistIn* a ser acessada pela função *List.nth*, retornando o elemento correspondente. O valor resultante determina a distância que deverá ser percorrida pelo transportador inicial que, por sua vez, com o disparo consequente da transição *T4*, permite o cálculo do seu tempo de deslocamento.

Ainda no submodelo da subpágina *TranspFromToInitVirtDist*, é necessário realizar a unitarização da distribuição de peças nas células virtuais. A parte do submodelo referido responsável por tal procedimento é apresentado na Figura A1.25.

Na Figura A1.25, o lugar *P9* recebe a marca temporizada correspondente ao deslocamento do transportador inicial do seu local corrente para o local da réplica da máquina responsável pela primeira operação da célula virtual da peça. Depois de transcorrido o tempo de simulação concernente, por conseguinte, a transição *T5* fica habilitada ao disparo.

Com o disparo da transição *T5*, simultaneamente, uma marca de valor inteiro é alocada no lugar *P10*, correspondendo à quantidade de peças no lote “e” e uma marca é alocada no lugar *P11*, contendo as informações do *ORDERS.txt* acerca do lote de peças (*a,b,c,d,e,f,g*) e o local da réplica da máquina responsável pela primeira operação na célula virtual. Consequentemente, a transição *T6* fica habilitada ao disparo.

Com o disparo da transição *T6*, um número de marcas definido pelo parâmetro “e” é alocado no lugar *Order_Seq*, em que cada marca contém as informações de uma peça ((*a,b,c,d,1,f,g*)) e o local da réplica da máquina responsável pela primeira operação na célula virtual.

Figura A1.26 - Subpágina *TranspFromToInitDisp* e suas ramificações (Fonte: autor)

Na Figura A1.26, da subpágina *TranspFromToInitDisp* deriva subpágina *MachQueueInUpdInitDisp* e, dessa última, a subpágina *MachInQueueAddInitDisp* e a subpágina *UpdateLayoutPosSubInitDisp*. O submodelo da subpágina *MachInQueueAddInitDisp* é responsável pela atualização da lista de filas de entrada das réplicas na posição correspondente ao local de trabalho do próximo processo de transformação a ser realizado na peça correspondente, sendo o incremento igual à quantidade de peças do lote. Já o submodelo da subpágina *UpdateLayoutPosSubInitDisp* é responsável pela atualização do estado de disponibilidade da réplica de máquina para a qual o lote de peças será movimentado; isto é, o valor corrente é alterado para “0”. Tudo isso se dá tão logo o submodelo do submodelo da subpágina *TranspFromToInitDisp* conclua os cálculos que lhe definem. A Figura A1.27 apresenta a estrutura desse último submodelo referido.

Na Figura A1.27, o lugar de entrada de marcas *P1* armazena as informações sobre os locais de todas as réplicas de máquinas iniciais disponíveis em dado momento (*MachPos*), bem como do lote de peças correspondente (*orinfo*), oriundas do submodelo da subpágina *DefinitionMachPosInitDisp*. A partir da informação do local em que se encontra, correntemente, o transportador inicial específico, armazenada no lugar *P2*, com o disparo da transição *T1*, a função *DefTranspDistIn* é chamada – tantas vezes quanto existirem réplicas de mesmo tipo disponíveis – retornando um valor inteiro correspondente à distância entre o local corrente do transportador e o local da réplica (*TrDist*).

Figura A1.27 - Submodelo da subpágina *TranspFromToInitDisp* (Fonte: autor)

Como já referido, a função *DefTranspDistIn* é, na verdade, uma composição de funções, servindo de interface para as outras. Se a movimentação se der do *Local(1)* para qualquer outro local no arranjo físico, a função *DefTranspDistIn* chama a função *DefMachDist*. Se a movimentação se der entre os locais de trabalho, internamente ao arranjo físico, a função *DefMachDistIn* é chamada. Como aqui se trata da movimentação do transportador inicial partindo não apenas do *Local(1)*, então ambos os casos podem ser satisfeitos. Um código possível para a função *DefMachDistIn* é apresentado a seguir.

```
fun DefMachDistIn(loc, p) =  
let
```

```
val MachListDist1=[0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7,3,4,5,6,7,8];
val MachListDist2=[1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6,4,3,4,5,6,7];
val MachListDist3=[2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5,5,4,3,4,5,6];
val MachListDist4=[3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4,6,5,4,3,4,5];
val MachListDist5=[4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3,7,6,5,4,3,4];
val MachListDist6=[5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2,8,7,6,5,4,3];
val MachListDist7=[1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6,2,3,4,5,6,7];
val MachListDist8=[2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5,3,2,3,4,5,6];
val MachListDist9=[3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4,4,3,2,3,4,5];
val MachListDist10=[4,3,2,1,2,3,3,2,1,0,1,2,4,3,2,1,2,3,5,4,3,2,3,4];
val MachListDist11=[5,4,3,2,1,2,4,3,2,1,0,1,5,4,3,2,1,2,6,5,4,3,2,3];
val MachListDist12=[6,5,4,3,2,1,5,4,3,2,1,0,6,5,4,3,2,1,7,6,5,4,3,2];
val MachListDist13=[2,3,4,5,6,7,1,2,3,4,5,6,0,1,2,3,4,5,1,2,3,4,5,6]
```

```

val MachListDist14=[3,2,3,4,5,6,2,1,2,3,4,5,1,0,1,2,3,4,2,1,2,3,4,5]
val MachListDist15=[4,3,2,3,4,5,3,2,1,2,3,4,2,1,0,1,2,3,3,2,1,2,3,4]
val MachListDist16=[5,4,3,2,3,4,4,3,2,1,2,3,3,2,1,0,1,2,4,3,2,1,2,3]
val MachListDist17=[6,5,4,3,2,3,5,4,3,2,1,2,4,3,2,1,0,1,5,4,3,2,1,2]
val MachListDist18=[7,6,5,4,3,2,6,5,4,3,2,1,5,4,3,2,1,0,6,5,4,3,2,1]
val MachListDist19=[3,4,5,6,7,8,2,3,4,5,6,7,1,2,3,4,5,6,0,1,2,3,4,5]
val MachListDist20=[4,3,4,5,6,7,3,2,3,4,5,6,2,1,2,3,4,5,1,0,1,2,3,4]
val MachListDist21=[5,4,3,4,5,6,4,3,2,3,4,5,3,2,1,2,3,4,2,1,0,1,2,3]
val MachListDist22=[6,5,4,3,4,5,5,4,3,2,3,4,4,3,2,1,2,3,3,2,1,0,1,2]
val MachListDist23=[7,6,5,4,3,4,6,5,4,3,2,3,5,4,3,2,1,2,4,3,2,1,0,1]
val MachListDist24=[8,7,6,5,4,3,7,6,5,4,3,2,6,5,4,3,2,1,5,4,3,2,1,0]

```

in

```

if loc = Local(2) then List.nth(MachListDist1,p-1) else
if loc = Local(3) then List.nth(MachListDist2,p-1) else
if loc = Local(4) then List.nth(MachListDist3,p-1) else
if loc = Local(5) then List.nth(MachListDist4,p-1) else
if loc = Local(6) then List.nth(MachListDist5,p-1) else
if loc = Local(7) then List.nth(MachListDist6,p-1) else
if loc = Local(8) then List.nth(MachListDist7,p-1) else
if loc = Local(9) then List.nth(MachListDist8,p-1) else
if loc = Local(10) then List.nth(MachListDist9,p-1) else
if loc = Local(11) then List.nth(MachListDist10,p-1) else
if loc = Local(12) then List.nth(MachListDist11,p-1) else
if loc = Local(13) then List.nth(MachListDist12,p-1) else
if loc = Local(14) then List.nth(MachListDist13,p-1) else
if loc = Local(15) then List.nth(MachListDist14,p-1) else
if loc = Local(16) then List.nth(MachListDist15,p-1) else
if loc = Local(17) then List.nth(MachListDist16,p-1) else
if loc = Local(18) then List.nth(MachListDist17,p-1) else
if loc = Local(19) then List.nth(MachListDist18,p-1) else
if loc = Local(20) then List.nth(MachListDist19,p-1) else
if loc = Local(21) then List.nth(MachListDist20,p-1) else
if loc = Local(22) then List.nth(MachListDist21,p-1) else
if loc = Local(23) then List.nth(MachListDist22,p-1) else
if loc = Local(24) then List.nth(MachListDist23,p-1) else
List.nth(MachListDist24,p-1)

```

end;

Em primeiro lugar, a partir do local corrente em que se encontra o transportador inicial (*loc*), toma-se uma das listas de valores inteiros *MachListDistI* (em que *I* varia de 1 até *m*, o número total de máquinas alocadas no arranjo físico) para, logo em seguida, a partir de *MachPos*, em que *p* assume seu valor, definir-se a posição da lista de valores inteiros *MachListDistI* a ser acessada pela função *List.nth*, retornando o elemento correspondente. O valor resultante determina a distância que poderá ser percorrida pelo transportador inicial. Consequentemente, a transição *T2* fica habilitada ao disparo.

O disparo recorrente da transição *T2* faz com que os valores das distâncias sejam armazenados numa marca de lista de valores inteiros no lugar *P5*. Tão logo as diversas distâncias são calculadas para todas as réplicas disponíveis em dado momento, a transição *T3* fica habilitada.

O disparo da transição *T3* chama a função *CalcMinTranspDistIn*.

```
CalcMinTranspDistIn(TranspDistListIn) =
let
  fun aux(f,L1) =
    if List.null(L1) then
      f
    else
      if f > hd(L1) then
        aux(hd(L1),tl(L1))
      else
        aux(f,tl(L1))
in
  aux(hd(TranspDistListIn),tl(TranspDistListIn))
end;
```

A partir da lista de valores inteiros *TranspDistListIn*, a função *CalcMinTranspDistIn* retorna o menor valor. O valor resultante determina a distância que deverá ser percorrida pelo transportador inicial que, por sua vez, com o disparo consequente da transição *T4*, permite o cálculo do seu tempo de deslocamento.

Por sua vez, a partir das informações fornecidas pelo submodelo da subpágina *BlockingInitControl*, o submodelo da subpágina *DefinitionMachPosDistQueueCrInit*, busca na lista de distribuição das máquinas, independentemente do fato de estarem ou não disponíveis, os locais em que as réplicas se encontram no arranjo físico. As marcas definidas como saída do submodelo da subpágina *DefinitionMachPosDistQueueCrInit* são as entradas para os procedimentos concernentes às subpáginas *TranspFromToDistCrInit* e *TranspFromToQueueCrInit*.

A Figura A1.28 ilustra a subpágina *TranspFromToDistCrInit* e suas ramificações. Já Figura A1.29 ilustra a subpágina *TranspFromToQueueCrInit* e suas ramificações.

Na Figura A1.28, a subpágina *TranspFromToDistCrInit* tem a ver com o critério da menor distância entre o local de origem – onde se deu o processo de transformação anterior da peça – e o local de destino de deslocamento do transportador – onde se dará o processo posterior de transformação da peça. Da subpágina *TranspFromToDistCrInit*, a subpágina *MachQueueInUpdDistCrInit* deriva e, dessa última, a subpágina *MachInQueueAddDistCrInit*. Ambas, em conjunto, são responsáveis pela atualização da lista de filas de entrada das réplicas na posição correspondente ao local de trabalho do próximo processo de transformação a ser realizado na peça correspondente, sendo o incremento igual à quantidade peças do lote, tão logo o submodelo do

submodelo da subpágina *TranspFromToDistCrInit* conclua os cálculos que lhe definem. Esses seguem o mesmo encadeamento lógico-causal daquele descrito para o submodelo da subpágina *TranspFromToInitDisp*, exceto que o estado de disponibilidade da réplica não é levado em consideração.

Figura A1.28 - Subpágina *TranspFromToDISTCrInit* e suas ramificações (Fonte: autor)

Por fim, a Figura A1.29 mostra a subpágina *TranspFromToQueueCrInit* e suas ramificações.

Figura A1.29 - Subpágina *TranspFromToQueueCrInit* e suas ramificações (Fonte: autor)

Na Figura A1.29, a subpágina *TranspFromToQueueCrInit* tem a ver com o critério de menor fila de caixas formadas antes das réplicas das máquinas (filas de entrada). Da subpágina *TranspFromToQueueCrInit*, derivam a subpágina *CalcDistMaqQueueCrInit* e a subpágina *MachQueueInUpdQueueCrInit* e, dessa última, a subpágina *MachInQueueAddQueueCrInit*. Ambas, em conjunto, são responsáveis pela atualização da lista de filas de entrada das réplicas na posição correspondente ao local de trabalho do próximo processo de transformação a ser realizado na peça correspondente, sendo o incremento igual à uma caixa, tão logo o submodelo do submodelo da subpágina *TranspFromToQueueCrInit* conclua os cálculos que lhe definem. A Figura A1.30 apresenta a estrutura do desse último submodelo referido.

Figura A1.30 - Submodelo da subpágina *TranspFromToQueueCrInit* (Fonte: autor)

Na Figura A1.30, o lugar de entrada de marcas *P1* armazena as informações sobre os locais de todas as réplicas de máquinas responsáveis pelas operações iniciais disponíveis em dado momento (*MachPos*), bem como sobre o lote de peças correspondente (*orinfo*), oriundas do submodelo da subpágina *DefinitionMachPosDistQueueCrInit*. A partir da informação do local em que se encontra, correntemente, o transportador inicial específico, armazenada no lugar *P2*, com o disparo da transição *T1*, as funções *DefTranspDistIn* e *DefMachQueue* são chamadas – tantas vezes quanto existirem réplicas de mesmo tipo – retornando, respectivamente, um valor inteiro correspondente à distância entre o local corrente do transportador e o local da réplica (*TrDist*) e um valor inteiro correspondente ao tamanho da fila de entrada nesse último. A função *DefTranspDistIn* já foi devidamente descrita acima. Já a função *DefMachQueue* tem a seguinte estrutura:

fun DefMachQueue(MInQ,MachPos) = List.nth(MInQ,MachPos-1);

Na função *DefMachQueue*, a variável de arco *MInQ*, a lista global das filas de entradas a todas as réplicas de máquinas, é acessada na posição que armazena o tamanho corrente da fila de entrada no local da réplica específica. Com o disparo consequente da transição *T2*, os valores dos tamanhos das filas das réplicas escolhidas são armazenados numa marca de lista de valores inteiros no lugar *P4*.

Tão logo esse processo iterativo termine, ou seja, com a obtenção de todos os tamanhos de filas, a transição *T3* fica habilitada ao disparo.

O disparo da transição *T3* chama a função *CalcMinMachQIn*. O código dessa função é apresentado a seguir.

```

fun CalcMinMachQIn(OrdMachInQ) =
  let
    fun aux(f,L1) =
      if List.null(L1) then
        f
      else
        if f > hd(L1) then
          aux(hd(L1),tl(L1))
        else
          aux(f,tl(L1))
    in
      aux(hd(OrdMachInQ),tl(OrdMachInQ))
    end;

```

A função *CalcMinMachQIn* retorna o valor inteiro correspondente ao menor tamanho de fila. A marca resultante inicia a determinação da distância a ser percorrida pelo transportador inicial no submodelo da subpágina *CalcDistMaqQueueCrInit*. Em caso de duas ou mais réplicas apresentarem o mesmo valor de menor tamanho de fila, a escolha é feita aleatoriamente.

A1.3.2 Modelagem do transporte intermediário

A movimentação de lotes de peças semi-acabadas entre os locais de trabalho é realizada por um transportador intermediário. Na medida em que os lotes de peças semi-acabadas vão sendo concluídos, a movimentação desses entre as máquinas subsequentes passa a ser necessária. A Figura A1.30 apresenta a subpágina *TranspInterMotionControl* e suas ramificações.

Na Figura A1.31, uma vez que são definidas células virtuais para o caso do arranjo físico distribuído, não há necessidade de utilização de transportadores para a movimentação intracelular. Assim, todas as ramificações têm a ver com o arranjo físico funcional. Aqui, também, o critério de decisão de escolha da réplica também permite distinguir as subpáginas em dois blocos com funções distintas, mas que estão diretamente associadas à subpágina *BlockingInitControl*.

Um dos blocos de subpáginas, formado pelas subpáginas *DefinitionMachPosInterDisp* e *TranspFromToInterDisp*, é determinado pelo critério da primeira réplica de máquina disponível. O outro bloco, as subpáginas *DefinitionMachPosDistQueueCrInter*, *TranspFromToDistCrInter* e *TranspFromToQueueCrInter*, é determinado pelos critérios da menor distância entre o local de origem –

onde se deu o processo de transformação anterior da peça – e o local de destino de deslocamento do transportador – onde se dará o processo posterior de transformação da peça – e da fila de menor tamanho das diversas réplicas disponíveis correntemente.

Figura A1.31 - Subpágina *TranspInterMotionControl* e suas ramificações (Fonte: autor)

O submodelo da subpágina *BlockingInterControl* é responsável pelo controle da entrada de informações sobre as peças e pela determinação do tipo de máquina requisitada para cada uma dessas. A partir de tal informação, consecutivamente, o submodelo da subpágina *DefinitionMachPosInterDisp* busca na lista do estado de disponibilidade corrente das máquinas, os locais em que as réplicas disponíveis se encontram no arranjo físico.

A Figura A1.32 ilustra a subpágina *TranspFromToInterDisp* e suas ramificações.

Figura A1.32 - Subpágina *TranspInterFromToInterDisp* e suas ramificações (Fonte: autor)

Na Figura A1.32, da subpágina *TranspFromToInterDisp*, a subpágina *MachQueueInUpdInterDisp* deriva e, dessa última, derivam a subpágina *MachInQueueAddInterDisp*, a subpágina *UpdateLayoutPosSubInterDisp* e a subpágina *MachOutQueueSemiProdSubInterDisp*. O submodelo da subpágina *MachInQueueAddInterDisp* é responsável pela atualização da lista de filas de entrada das réplicas na posição correspondente ao local de trabalho do próximo processo de transformação a ser realizado na peça correspondente, sendo o incremento igual à uma caixa. Já o submodelo da subpágina *UpdateLayoutPosSubInterDisp* é responsável pela atualização do estado de disponibilidade da réplica de máquina para a qual o lote de peças será movimentado; o valor corrente é alterado para “0”.

Já o submodelo da subpágina *MachOutQueueSemiProdSubInterDisp*, ao contrário, é responsável pela atualização da lista de filas de entrada das réplicas na posição correspondente ao local de trabalho do último processo de transformação concluído na peça correspondente, sendo o decremento igual à uma caixa.

Tudo isso se dá tão logo o submodelo do submodelo da subpágina *TranspFromToInterDisp* conclua os cálculos que lhe definem. Esses seguem o mesmo encadeamento lógico-causal daquele descrito para o submodelo da subpágina *TranspFromToInitDisp*, exceto que a determinação da movimentação do transportador leva em consideração apenas as relações quantitativas entre os locais internos ao arranjo físico.

Por sua vez, a partir das informações fornecidas pelo submodelo da subpágina *BlockingInterControl*, o submodelo da subpágina *DefinitionMachPosDistQueueCrInter* busca, na lista de distribuição das máquinas, independentemente do fato de estarem ou não disponíveis, os locais em que as réplicas se encontram no arranjo físico. As marcas definidas como saída do submodelo da subpágina *DefinitionMachPosDistQueueCrInter* são as entradas para os procedimentos concernentes às subpáginas *TranspFromToDistCrInter* e *TranspFromToQueueCrInter*.

A Figura A1.33 ilustra a subpágina *TranspFromToDistCrInter* suas ramificações. Já Figura A1.33 ilustra a subpágina *TranspFromToQueueCrInter* e suas ramificações.

Figura A1.33 - Subpágina *TranspFromToInterDistCr* e suas ramificações (Fonte: autor)

Na Figura A1.33, a subpágina *TranspFromToDistCrInter* tem a ver com o critério da menor distância entre o local de origem – onde se deu o processo de transformação anterior da peça – e o local de destino de deslocamento do transportador – onde se dará o processo posterior de transformação da peça. Da subpágina *TranspFromToDistCrInter*, deriva subpágina *MachQueueUpdDistCrInter* e, dessa última, a subpágina *MachInQueueAddDistCrInter* e a subpágina *MachOutQueueSemiProdSubDistCrInter*. As duas primeiras, em conjunto, são responsáveis pela atualização da lista de filas de entrada das réplicas na posição correspondente ao local de trabalho do próximo processo de transformação a ser realizado na peça correspondente, sendo o incremento igual à uma caixa.

Já o submodelo da subpágina *MachOutQueueSemiProdSubDistCrInter*, ao contrário, é responsável pela atualização da lista de filas de entrada das réplicas na posição correspondente ao local de trabalho do último processo de transformação concluído na peça correspondente, sendo o decremento igual à uma caixa.

Tudo isso se dá tão logo o submodelo do submodelo da subpágina *TranspFromToDistCrInter* conclua os cálculos que lhe definem. Esses seguem o mesmo encadeamento lógico-causal daquele descrito para o submodelo da subpágina *TranspFromToDistCrInit*, exceto que a determinação da movimentação do transportador leva em consideração apenas as relações quantitativas entre os locais internos ao arranjo físico.

Por fim, a Figura A1.34 mostra a subpágina *TranspFromToQueueCrInter* e suas ramificações.

Figura A1.34 - Subpágina *TranspFromToQueueCrInter* e suas ramificações (Fonte: autor)

Na Figura A1.34, a subpágina *TranspFromToQueueCrInit* tem a ver com o critério de menor fila de caixas formadas nas diversas réplicas correntemente. Da subpágina *TranspFromToQueueCrInit*, deriva subpágina *CalcDistMaqQueueCrInit* e a subpágina *MachQueueInUpdQueueCrInit* e, dessa última, a subpágina *MachInQueueAddQueueCrInit* e a subpágina *MachOutQueueSemiProdSubDistCrInter*. As duas primeiras, em conjunto, são responsáveis pela

atualização da lista de filas de entrada das réplicas na posição correspondente ao local de trabalho do próximo processo de transformação a ser realizado na peça correspondente, sendo o incremento igual à uma caixa.

Já o submodelo da subpágina *MachOutQueueSemiProdSubDistCrInter*, ao contrário, é responsável pela atualização da lista de filas de entrada das réplicas na posição correspondente ao local de trabalho do último processo de transformação concluído na peça correspondente, sendo o decremento igual à uma caixa.

Tudo isso se dá tão logo o submodelo do submodelo da subpágina *TranspFromToQueueCrInter* conclua os cálculos que lhe definem. Esses seguem o mesmo encadeamento lógico-causal daquele descrito para o submodelo da subpágina *TranspFromToQueueCrInter*, exceto que a determinação da movimentação do transportador leva em consideração apenas as relações quantitativas entre os locais internos ao arranjo físico.

A1.3.3 Modelagem do transporte final

À medida que as peças, para o caso do arranjo físico distribuído, ou os lotes de peças, para o caso do arranjo físico funcional, vão sendo concluídos, faz-se necessária a movimentação dos mesmos das réplicas das máquinas finais para o L2. Tal função é exercida pelo transportador final. A Figura A1.35 ilustra a subpágina *TranspExitMotionControl* e suas ramificações.

Figura A1.35 - Subpágina *TranspExitMotionControl* e suas ramificações (Fonte: autor)

Na Figura A1.35, o submodelo da subpágina *MachOutQueueExitProdInAddQueue* é responsável pela atualização da lista das filas formadas após as réplicas de máquinas (filas de saída) na posição correspondente ao local de trabalho, sendo o incremento unitário, para o caso do arranjo físico distribuído ou igual à uma caixa, para o caso do arranjo físico funcional. Ao final dessa etapa, uma das suas subpáginas, *TranspFromToExitDistQueueCr* ou *TranspFromToExitQueueDistCr*, é realizada. A escolha depende do critério de decisão particular pré-determinado para a movimentação do

transportador final à medida que os lotes de peças acabadas vão sendo formados. A Figura A1.36 apresenta a subpágina *TranspFromToExitDistQueueCr* e suas ramificações. Já a Figura A1.37 mostra a subpágina *TranspFromToExitQueueDistCr* e suas ramificações.

Figura A1.36 - Subpágina *TranspFromToExitDistQueueCr* e suas ramificações (Fonte: autor)

Na Figura A1.36, o submodelo da subpágina *TranspFromToExitDistQueueCr* é iniciado caso o critério de decisão seja, primeiro, o da determinação da réplica de máquina com a menor distância para o local corrente do transportador final, caso existam peças acabadas (armazenadas em caixas) formadas em máquinas. Se houver empate, o submodelo da subpágina *CalcDistMqExitDistQueueCr* usa, como critério de desempate, a escolha da réplica que apresente o maior tamanho da fila de saída. Já o submodelo da subpágina *MachOutExitProdInSubDistQueueCr*, ao contrário, é responsável pela atualização da lista de filas de saída das réplicas na posição correspondente, sendo o decremento igual a uma caixa (tanto para o arranjo físico funcional quanto para o arranjo físico distribuído).

Ao contrário, na Figura A1.37, o submodelo da subpágina *TranspFromToExitQueueDistCr* é iniciado caso o critério de decisão seja, primeiro, o da determinação da réplica de máquina que apresente a fila de saída com o maior tamanho, enquanto o critério de desempate utilizado, se necessário, no submodelo da subpágina *CalcDistMqExitQueueDistCr*, é o de menor distância para o local corrente do transportador final. Já o submodelo da subpágina *MachOutExitProdInSubQueueDistCr*, ao contrário, é responsável pela atualização da lista de filas de saída das réplicas na posição correspondente, sendo o decremento igual à uma caixa (tanto para o arranjo físico funcional quanto para o arranjo físico distribuído).

Figura A1.37 - Subpágina *TranspFromToExitQueueDistCr* e suas ramificações (Fonte: autor)

Por fim, a marca resultante com o valor inteiro da distância a ser percorrida pelo transportador final será utilizada, consecutivamente, para o cálculo do seu tempo de deslocamento no submodelo da subpágina *TranspToExp*.

A1.4 MODELAGEM DA TOMADA DE DECISÃO GERAL

Numa perspectiva de totalidade, a tomada de decisão geral é mais do que uma mera parte da modelagem em si. Na verdade, ela é uma seção do modelo total que lhe garante a subsistência lógica necessária, ou seja, uma rede de Petri viva, sem travamentos lógicos. Isso se dá pelo simples fato de que todos os submodelos anteriormente descritos, em maior ou em menor grau, dependem da estrutura lógica de controle presente na subpágina supracitada, uma vez que os critérios de decisão formalmente incorporados buscam afigurar o modo de alocação e de distribuição das peças ou lotes de peças ao longo do arranjo físico como um todo.

Daí que a centralidade do modelo ora proposto só é possível na medida em que deve haver uma articulação permanente entre as partes que o compõem, no que diz respeito a todos os fatos ou eventos que, em última instância, definem-no, sucessiva e simultaneamente, alterando o estado dos recursos produtivos articulados. Para tal, a subpágina *ModelLogicControl* e suas ramificações é apresentada na Figura A1.38.

Figura A1.38 - Subpágina *ModelLogicControl* e suas ramificações (Fonte: autor)

Na Figura A1.38, o nó raiz *ModelLogicControl* define a subpágina da qual derivam as subpáginas *CriteriaControl*, *CheckMachDispOkInit*, *CheckMachDispOkInt*, *DistQueueCriterion* e *DefinitionVirtualCellsLayout*, articulando-as entre si e, principalmente, com as outras partes do modelo. O submodelo da subpágina *ModelLogicControl* é responsável pela inicialização geral do modelo. Nele, os arquivos externos (.txt), contendo informações sobre os pedidos (*ORDERS.txt*, conforme mostrado

no Apêndice 4), sobre os critérios de decisão a serem implementados – CrFunc.txt, no caso de arranjo físico funcional e CrMax.txt, no caso de arranjo físico distribuído – e sobre os diversos blocos iniciais de ferramentas das máquinas são lidos e incorporados ao modelo. Tanto as informações contidas no arquivo CrFunc.txt quanto no arquivo CrMax.txt podem ser vistas no Apêndice 6.

Consecutivamente, o submodelo da subpágina derivada CriteriaControl estabelece e controla a sequência de critérios de decisão para cada série de experimentos.

Já os submodelos das subpáginas CheckMachDispOkInit e CheckMachDispOkInt são os responsáveis diretos pela articulação lógica entre os fatos que concernem as etapas de movimentação das peças ou dos lotes de peças pelos transportadores iniciais e pelos transportadores intermediários, no que se refere ao critério de escolha da primeira réplica de máquina disponível.

Tendo em vista o caráter contingente desse processo decisório, os transportadores podem dirigir-se a qualquer local de trabalho. Daí ser necessário o uso de listas globais contendo as informações atualizadas acerca da totalidade das filas de entrada das réplicas de máquinas e do estado de disponibilidade das mesmas.

Assim, o submodelo da subpágina CheckMachDispOkInit, antes mesmo de verificar a disponibilidade de um tipo de máquina requisitada por uma peça específica na lista global de disponibilidade, num dado instante, trava o acesso do submodelo da subpágina CheckMachDispOkInt a mesma lista para o mesmo fim, e vice-versa. O destravamento se dá apenas após a atualização da lista global de disponibilidade.

No entanto, caso um dos ou mesmo ambos os submodelos não identifiquem nenhuma máquina disponível, as informações dos lotes ficam armazenadas em lugares específicos, internos aos mesmos, para que, à medida que as máquinas tornem-se disponíveis novamente, o ciclo de verificação e atualização supracitado é recommçado, findando somente quando todas as peças tenham sido devidamente alocadas e distribuídas. Uma lógica similar perpassa o submodelo da subpágina DistQueueCriterion. A Figura A1.39 apresenta a subpágina DistQueueCriterion e suas ramificações.

Figura A1.39 - Subpágina DistQueueCriterion e suas ramificações (Fonte: autor)

Na Figura A1.39, da subpágina *DistQueueCriterion*, a subpágina *UpdateMachSub* deriva, da qual derivam a subpágina *InitUpdMachSubMachLayout* e a subpágina *InitUpdMachSubMachLayout*. O submodelo da subpágina *DistQueueCriterion* é responsável pelo armazenamento das informações de todas as peças correntemente alocadas nas diversas filas de entrada das réplicas de máquinas, dos locais dessas últimas correspondentes e do tipo de máquina. À medida que os estados de disponibilidade das réplicas sofrem alterações, ou seja, ficam disponíveis, o submodelo da subpágina *UpdateMachSub* é iniciado.

Uma de suas ramificações, o submodelo da subpágina *InitUpdMachSubMachLayout*, fica recorrentemente verificando a possibilidade de movimentação de cada uma das peças armazenadas no submodelo *UpdateMachSub*, tanto no âmbito do transporte final quanto no âmbito do transporte intermediário. Caso seja constatada uma movimentação real, o submodelo da subpágina *UpdMachSubMachLayout* atualiza a lista de disponibilidade das máquinas na posição concernente à réplica requisitada pela peça a ser movimentada.

Por fim, na subpágina *DefinitionVirtualCellsLayout* e em suas ramificações, dá-se a construção das células virtuais para o caso do arranjo físico distribuído. Somente quando a célula virtual de cada tipo de peça é determinada antecipadamente, a simulação pode ser iniciada para o caso particular do arranjo físico distribuído. Na Figura A1.40, a subpágina *DefinitionVirtualCellsLayout* e suas ramificações são apresentadas.

Figura A1.40 - Subpágina *DefinitionVirtualCellsLayout* e suas ramificações (Fonte: autor)

Na Figura A1.40, da subpágina *DefinitionVirtualCellsLayout*, derivam a subpágina *DefInitMachOrdersVC*, a subpágina *DefinitionMachPosVC*, a subpágina *TranspFromToDistVC* e a subpágina *DefineVirtCellPieceVC*.

O submodelo da subpágina *DefInitMachOrdersVC* recebe do submodelo da subpágina *ModelLogicControl* as informações dos lotes de peças. Para cada lote, *DefInitMachOrdersVC* retorna uma lista da sequência das máquinas que devem compor a célula virtual que lhe corresponde (*ListMach*), bem como, a partir de uma cópia da lista global de distribuição das réplicas de máquinas no arranjo

físico (LMListInCell), a quantidade de réplicas da máquina correspondente ao processo inicial. A Figura A1.41 apresenta a estrutura do submodelo referido.

Figura A1.41 - Submodelo da subpágina DefInitMachOrdersVC (Fonte: autor)

Na Figura A1.41, o lugar de entrada de marcas OrdersDistInIt, localizado na parte superior à direita, fornece marcas contendo as informações acerca do lote de peças (orinfoDist). Uma vez habilitada, o disparo consequente da transição T1 faz com que, no lugar P2, seja alocada uma marca contendo as informações acerca do lote de peças (orinfoDist) e o tipo de peça que compõe o lote (pc). Consequentemente, a transição T2 fica habilitada ao disparo.

O disparo da transição T2 chama a função DefMachPieceDist, conforme mostrada no Apêndice 15.

A partir das variáveis de arco `orinfoDist` (= `orinfo`) e `pc`, a função `DefMachPieceDist` retorna, particularmente, a lista de máquinas que compõem a sequência de operações correspondente para cada peça. Por conseguinte, no lugar `P3`, uma marca contendo as informações acerca do lote de peças (`orinfoDist`) e a lista de máquinas correspondente (`ListMaq`) é alocada. Consequentemente, a transição `T3` fica habilitada ao disparo.

O disparo da transição T3 faz com que seja alocada, simultaneamente, uma marca no lugar P4 contendo a informação do tipo da primeira máquina de ListMaq (MaqIn), uma marca no lugar P5 contendo ListMag, uma marca no lugar P7 contendo as informações acerca do lote de peças (orinfoDist)

e da lista de máquinas (ListMach) e uma marca no lugar P8 contendo as informações acerca do lote de peças (orinfoDist). Consequentemente, a transição T4 fica habilitada ao disparo.

O disparo da transição T4 aloca, no lugar P6, uma marca de valor inteiro simbolizando a informação do tipo da primeira máquina (McIn). Tal marca no lugar P6 é a informação necessária para que seja checada a presença de outras réplicas da máquina responsável pela primeira operação e, por conseguinte, pela recorrência ao ciclo de checagem para a peça corrente, determinar a quantidade de células virtuais possíveis – a partir de LMListInCell, caso sejam verificadas as condições de disparo da transição T5 para a proposição de sua guarda “mem LMListInCell McIn = true” –, bem como o início do ciclo de checagem para um novo tipo de peça – caso sejam verificadas as condições de disparo da transição T6 para a proposição de sua guarda “mem LMListInCell McIn = false”.

Consecutivamente, o submodelo da subpágina DefinitionMachPosVC, a partir de LMListInCell e a partir de ListMach, identifica, primeiramente, o local de uma das réplicas da máquina responsável pela primeira operação. A Figura A1.42 apresenta a estrutura do submodelo referido.

Figura A1.42 - Submodelo da subpágina DefinitionMachPosVC (Fonte: autor)

Na Figura A1.42, o lugar de entrada de marcas P1 fornece as informações acerca do lote de peças (orinfoInit) e do tipo de máquina responsável pela primeira operação (McIn). Uma vez habilitada, o disparo consequente da transição T1 aloca uma marca no lugar MachU_In2. Consequentemente, a transição T2 fica habilitada ao disparo.

O disparo da transição T2 faz com que, simultaneamente, uma marca seja alocada no lugar P3, contendo a informação da posição da primeira réplica da máquina responsável pela primeira operação, uma marca no lugar P4, contendo a informação da lista global de distribuição das réplicas de máquinas no arranjo físico (LMListIn), uma marca no lugar P5, contendo as informações acerca do lote de peças (orinfoDist), uma marca no lugar MachU_In2, contendo as informações acerca do lote de peças (orinfoLit) e do tipo da primeira máquina (McIn) e uma marca no lugar P6, contendo a informação da posição da primeira réplica da máquina responsável pela primeira operação. Especialmente nos casos das alocações de marca nos lugares P3 e P6, a função DefMachPosIn é chamada. Um código possível para a referida função é apresentado a seguir.

```
fun DefMachPosIn(Maqln,LMListIn) =
let
val L = length(LMListIn);
in
if Maqln = Mach(1) then CalcMachPosIn(1,LMListIn,L) else
if Maqln = Mach(2) then CalcMachPosIn(2,LMListIn,L) else
if Maqln = Mach(3) then CalcMachPosIn(3,LMListIn,L) else
if Maqln = Mach(4) then CalcMachPosIn(4,LMListIn,L) else
if Maqln = Mach(5) then CalcMachPosIn(5,LMListIn,L) else
if Maqln = Mach(6) then CalcMachPosIn(6,LMListIn,L) else
if Maqln = Mach(7) then CalcMachPosIn(7,LMListIn,L) else
if Maqln = Mach(8) then CalcMachPosIn(8,LMListIn,L) else
if Maqln = Mach(9) then CalcMachPosIn(9,LMListIn,L) else
if Maqln = Mach(10) then CalcMachPosIn(10,LMListIn,L) else
if Maqln = Mach(11) then CalcMachPosIn(11,LMListIn,L) else
if Maqln = Mach(12) then CalcMachPosIn(12,LMListIn,L) else
if Maqln = Mach(13) then CalcMachPosIn(13,LMListIn,L) else
if Maqln = Mach(14) then CalcMachPosIn(14,LMListIn,L) else
CalcMachPosIn(15,LMListIn,L)
end;
```

A função DefMachPosIn, a partir dos valores das variáveis de arco Maqln e LMListIn, calcula a quantidade de réplicas distribuídas no arranjo físico (L) e chama a função CalcMachPosIn. Um código possível para a referida função é apresentado a seguir.

```
fun CalcMachPosIn(pin,LMListIn,L) =
if hd(LMListIn) = pin then L - length(tl(LMListIn))
else
CalcMachPosIn(pin,tl(LMListIn),L);
```

A função CalcMachPosIn identifica e retorna a posição da primeira réplica da máquina responsável pela primeira operação em LMListIn (LthListIn). Consequentemente, as transições T3 e T4 ficam habilitadas ao disparo.

O disparo da transição T3 faz com que uma marca contendo informações acerca do lote de peças (orinfoDist) e da posição da primeira réplica de máquina responsável pela primeira operação (i11) seja alocada no lugar P7.

O disparo da transição T4 faz com que a função List.drop seja chamada. A partir da variável de arco LthListIn, a referida função retorna LMListIn atualizada, com a exclusão dos LthListIn primeiro elementos da mesma. Consequentemente, a transição T5 fica habilitada ao disparo.

O disparo da transição T5 faz com que uma marca com a lista atualizada LMListIn seja alocada no lugar P2, enquanto houver réplicas do tipo de máquina requisitadas.

Uma vez identificando a posição da primeira réplica da máquina responsável pela primeira operação, o submodelo da subpágina TranspFromToDistVC calcula a distância do local dessa réplica para com o Local(1); local este em que se encontra inicialmente o transportador inicial. A Figura A1.43 apresenta a estrutura do submodelo referido.

Figura A1.43 - Submodelo da subpágina TranspFromToDistVC (Fonte: autor)

Na Figura A1.43, o lugar de entrada de marcas P7 fornece as informações acerca do lote de peças (orinfoPosIn) e da posição da primeira réplica de máquina responsável pela primeira operação (MachPos). Consequentemente, a transição T1 fica habilitada ao disparo.

O disparo da transição T1 chama a função DefTranspDistIn. Tal função retorna a distância da primeira réplica para com o local em que se encontra inicialmente o transportador inicial (Local(1)). Com o disparo da transição T2, tal informação é armazenada no lugar P3. A partir da marca alocada nesse

lugar, a função CalcMinTranspDistIn retorna o valor da menor distância (TrDist). Nesse caso, uma vez que se trata do ciclo inicial, apenas um valor é armazenado no lugar P3, de tal maneira que a réplica de menor distância é a primeira réplica.

Consecutivamente, tendo atualizado ListMq com a exclusão do tipo de máquina responsável pela primeira operação, restam os tipos de máquinas subsequentes que podem compor a célula virtual. Assim, novamente em DefinitionMachPosVC, identificam-se os locais de todas as réplicas subsequentes à réplica de máquina responsável pela primeira operação, para que, em TranspFromToDistVC seja calculada, dentre essas, a réplica mais próxima.

O ciclo DefinitionMachPosVC-TranspFromToDistVC continua até que todas as máquinas de ListMq sejam devidamente analisadas e, ao final, conforme-se uma célula virtual candidata. A subpágina TranspFromToDistVC e sua ramificação, a subpágina UpdateVirtLayoutPosSubDistVC, são apresentadas na Figura A1.44.

Figura A1.44 - Subpágina TranspFromToDistVC e sua ramificação (Fonte: autor)

Na Figura A1.44, ao final de cada ciclo de construção de uma célula virtual candidata, o submodelo da subpágina UpdateVirtLayoutPosSubDistVC atualiza a cópia da lista global de distribuição das réplicas de máquinas no arranjo físico, acessando-a nas posições concernentes aos locais das réplicas de máquinas que a compõem; o valor corrente é alterado para “0”. Tão logo isso ocorra, o ciclo DefinitionMachPosVC-TranspFromToDistVC recomeça, em que a marcação inicial é retomada – a informação do local do transportador inicial volta a ser Local(1) –, identificando outras células virtuais candidatas, de acordo com a quantidade restante de réplicas da máquina responsável pela primeira operação e das réplicas subsequentes mais próximas entre si da peça sob análise.

A partir desse conjunto de células virtuais candidatas, deve-se ativar uma função que calcula aquela que apresentar a menor distância a ser percorrida pela peça. Esta função está localizada no submodelo da subpágina DefineVirtCellPieceVC. A Figura A1.45 apresenta a subpágina DefineVirtCellPieceVC e sua ramificação, a subpágina VirtualCellLaytPosAddVC.

Figura A1.45 - Subpágina DefineVirtCellPiece e sua ramificação (Fonte: autor)

A Figura A1.46 apresenta a estrutura do submodelo da subpágina DefineVirtCellPieceVC.

Figura A1.46 - Submodelo da subpágina DefineVirtCellPiece (Fonte: autor)

Na Figura A1.46, como resultado de cada ciclo de construção de célula virtual candidata, o lugar P1 armazena as informações correspondentes. Uma vez habilitada, o disparo da transição T1 armazena no lugar P3 uma marca de lista de inteiros, a distância total interna de cada célula virtual candidata para cada tipo de peça do pedido. Uma vez sendo todas devidamente identificadas e calculadas, com o disparo da transição T2, a função CalcMinListDistIn é chamada. Tal função calcula o menor valor de distância interna (TrDistFin), possibilitando identificar, assim, a posição das réplicas de máquinas no arranjo físico que compõem a célula virtual mais adequada para cada tipo de peça do pedido.

Por fim, o submodelo da subpágina VirtualCellLayPosAddVC consolida e armazena todas as informações das células virtuais para cada tipo de peça a ser manufaturada. Cada célula virtual

escolhida, agora, é determinada por uma marca contendo informações acerca do lote de peças, uma lista das distâncias subsequentes entre os locais de trabalho (TrDistList), uma lista dos locais subsequentes (ListLocal) e uma lista global com a distribuição da célula virtual no arranjo físico como um todo (LMListInCell).

APÊNDICE 2 VERIFICAÇÃO E VALIDAÇÃO DO MODELO DE SIMULAÇÃO

Este apêndice tem por finalidade descrever os procedimentos adotados para a verificação e validação do modelo de simulação construído antes de iniciar o processo de coleta de dados para fins de análise comparativa de desempenho entre o arranjo físico funcional e o arranjo físico distribuído.

Law & Kelton (1991) definem o processo de verificação como sendo o ato de avaliar se o modelo conceitual de simulação foi corretamente traduzido num programa de computador.

Para este trabalho de pesquisa, o procedimento adotado para a verificação foi realizar o acompanhamento no modelo de simulação, passo a passo, para verificar se este se comporta conforme a lógica definida pelo sistema teórico. Essa verificação foi possível reduzindo-se a velocidade de execução do modelo de simulação. Durante a modelagem, acompanhou-se os comportamentos das entidades bem como dos eventos, observando se estão atuando conforme planejados, desde a chegada das informações da demanda no L1 até a chegada das peças acabadas no L2. Além disso, o próprio simulador CPNTools dispõe-se de uma ferramenta que comunica, apesar de não informar exatamente qual página, ao modelador sobre uma possível inconsistência na estrutura lógica. Esta comunicação é feita pelo simulador através do travamento da simulação do modelo. Essa foi, então, a ferramenta usada durante o processo de verificação.

Zagonel (2006) também adota este método de redução de velocidade de execução do modelo de simulação para fins de verificação. Este método de verificação também é denominado de verificação por rastreamento (MARIA, 1997).

Com relação à validação, dependendo do sistema modelado, pode não ser possível de ser realizada. Por exemplo, se o modelo construído é referente a um sistema real, então os dados obtidos pelo modelo de simulação podem ser comparados com os dados do sistema real. Apesar de ter vantagem na validação, a simulação deste é mais restrita, porque o modelo permite apenas avaliar o desempenho, único e exclusivamente, do sistema real.

Por outro lado, quando trata-se de um modelo de um sistema teórico, ganha-se vantagem na flexibilidade na simulação, porém o processo de validação do modelo se torna um problema.

Problemas de validação ocorrem com muita frequência quando o modelo construído refere-se a um sistema teórico, tais como encontradas nos trabalhos de Zagonel (2006) e Reaes (2008).

Zagonel (2006) apresenta um sistema teórico de manufatura celular contendo apenas seis máquinas. Para a validação do modelo de simulação, este autor realiza a comparação dos dados de saída obtidos da simulação do modelo com os dados teóricos. No caso do sistema abordado pelo autor, é possível realizar esta comparação porque todas as máquinas estão alocadas sempre nos mesmos locais, ou seja, elas não mudam de local após a alocação. Entretanto, para este trabalho de pesquisa, o chão de fábrica pode conter 24 máquinas, 56 máquinas ou até 150 máquinas, sendo que todas as

máquinas podem ser alocadas em qualquer local. Por isso, o modelo construído nessa tese é muito mais complexo e não pode ser utilizado este método de validação, que é por dados teóricos.

Reaes (2008) também afirma que não foi possível realizar a validação do modelo construído porque o sistema modelado não refere-se a uma empresa real. O autor, mesmo assim, procurou validar o modelo ajustando os seus dados de entrada ao de Suresh & Slomp (2005). Mas como estes autores não expuseram dados por completo, Reaes (2008) reconhece que não foi possível realizar a validação do seu modelo de simulação por não conseguir reproduzir o sistema de Suresh & Slomp (2005).

Existe um método de validação, também denominado de validação subjetiva, que pode ser usado para a validação de modelos de sistemas teóricos. A validação subjetiva consiste em submeter a análise dos dados gerados pelo modelo de simulação a uma equipe de especialistas. Estes profissionais, com base em suas experiências, farão julgamento quanto a confiabilidade dos dados gerados (LAW & KELTON, 1991; WINSTOM, 1994 e BATEMAN et al., 1999).

APÊNDICE 3 GERAÇÃO DOS PERFIS DAS PEÇAS

Este apêndice apresenta a execução do código de programação referente ao procedimento proposto para a geração dos perfis de peças adotados neste estudo.

Para este trabalho de pesquisa, considera-se que a fábrica é capaz de produzir até 100 tipos de peças. Deve-se então gerar informações dessas 100 peças para fins de estudo comparativo.

Entretanto, os perfis das peças podem variar. Por exemplo, se o objetivo da simulação é gerar peças, cujo número de operações varie de uma a quatro operações, então tem-se uma lista com informações das 100 peças, com o número de operações variando de um a quatro. Se o objetivo é fazer a simulação para uma a oito operações, então nova lista de peças com este perfil é gerada. Enfim, não é possível definir uma única lista de 100 peças neste estudo.

Para fins de ilustração, um exemplo é apresentado mostrando-se a execução do código de programação referente à geração dos perfis das peças e também apresentar como estas informações são impressas nos arquivos externos, dados em txt.

Por exemplo, se o objetivo é realizar a simulação dos arranjos físicos para peças com até oito operações, então deve-se executar o código de programação referente ao procedimento de geração das peças e ao executá-lo, a tela apresentada é similar ao que está apresentada na Figura A3.1.

```
C:\DOCUMENTOS\shih\MEUSDO\1\TP\BIN\TURBO.EXE
Entre com o numero de tipos de pecas da fabrica <max. 100>
100
Entre com o numero de tipos de maquinas da fabrica <max. 130>
15
Entre com o maior numero de processos que uma peca pode ter <max. 100>
8
Entre com o numero total de ferramentas que as maquinas podem trabalhar <max. 9>
9
```

Figura A3.1 - Tela de execução do código de programação para a geração dos perfis das peças
(Fonte: autor)

Após a execução do código de programação, este então deve gerar dois arquivos externos, um referente às operações das peças e o outro, referente aos pesos das peças. Na Tabela A3.1 estão as informações geradas pelo código de programação ao primeiro arquivo externo. Note que as informações geradas são: código da peça, número de operações e o ferramental usado. Por exemplo: a peça de código 10 apresenta 4 operações, sendo a primeira operação a ser realizada na máquina tipo 15 usando o ferramental tipo 2, a segunda operação a ser realizada na máquina tipo 1 usando o ferramental tipo 3, a terceira operação a ser realizada na máquina tipo 14 usando o ferramental tipo 3 e por fim, a quarta operação a ser realizada na máquina tipo 13 usando o ferramental tipo 7.

Tabela A3.1 - Informações dos perfis das peças impressas pelo código de programação (primeiro arquivo) (Fonte: autor)

n. peça, n. op, seq e ferram	
1 1 55	51 5 65 12 155 79 136
2 1 42	52 2 73 136
3 7 119 107 91 122 48 114 152	53 6 56 143 101 88 142 157
4 2 12 18	54 5 113 143 84 136 14
5 3 64 25 85	55 8 45 83 96 66 127 39 77 95
6 6 118 93 66 146 47 98	56 6 87 145 108 71 151 119
7 3 64 108 72	57 8 46 158 11 68 49 99 124 11
8 2 118 129	58 2 108 85
9 3 37 53 115	59 6 129 52 39 24 34 81
10 4 152 13 143 137	60 3 63 51 58
11 1 153	61 1 44
12 4 68 13 19 71	62 7 91 44 27 71 115 77 11
13 1 37	63 4 45 136 48 85
14 7 54 93 33 27 58 106 141	64 8 17 114 46 61 136 137 117 134
15 1 48	65 5 138 141 45 128 74
16 3 111 32 74	66 5 39 82 146 91 71
17 8 31 87 93 131 81 66 48 84	67 8 112 31 66 76 57 71 119 48
18 3 19 31 11	68 6 77 154 17 26 109 101
19 7 139 22 129 81 98 145 109	69 3 56 143 41
20 3 127 149 118	70 1 61
21 6 14 51 123 139 19 145	71 6 58 31 144 33 89 72
22 7 114 77 114 105 151 153 104	72 8 26 54 98 132 127 112 63 144
23 6 86 136 155 89 42 77	73 7 43 18 131 35 56 96 78
24 3 121 108 72	74 6 37 42 149 23 147 24
25 2 101 158	75 2 74 128
26 3 144 12 98	76 2 79 91
27 4 15 148 137 152	77 8 146 69 146 104 21 155 157 77
28 2 36 132	78 7 81 27 38 45 36 109 153
29 7 18 95 72 128 26 122 84	79 5 13 39 158 35 56
30 3 87 123 16	80 7 83 58 27 141 69 84 121
31 4 51 72 26 134	81 6 118 52 148 23 154 138
32 2 55 147	82 3 151 13 113
33 7 71 133 14 148 73 71 104	83 1 71
34 3 107 135 96	84 6 111 109 117 94 116 63
35 7 53 77 102 151 59 85 147	85 3 61 58 95
36 8 76 39 82 56 135 35 36 108	86 8 17 83 56 45 144 103 97 45
37 4 132 37 54 41	87 2 103 95
38 8 147 26 16 97 39 136 88 122	88 3 72 88 113
39 7 69 88 113 91 64 76 147	89 5 79 128 71 145 85
40 1 129	90 8 98 21 102 17 148 12 131 151
41 2 61 73	91 2 29 59
42 2 143 61	92 2 11 15
43 5 85 93 34 155 94	93 3 57 68 129
44 1 68	94 2 68 49
45 5 147 95 143 19 89	95 5 69 54 155 54 53
46 1 48	96 4 145 131 45 122
47 7 24 19 67 138 66 58 34	97 3 127 121 116
48 6 71 151 31 16 58 27	98 2 53 43
49 7 75 81 153 158 128 75 57	99 5 77 74 46 41 128
50 6 64 79 66 115 33 75	100 6 129 147 34 37 151 29

O segundo arquivo a ser gerado pelo código de programação refere-se aos pesos das peças. Ao dar continuidade à execução do código de programação, as informações dos pesos das peças são impressas ao arquivo externo, conforme mostradas na Tabela A3.2.

As informações impressas são: código da peça e o peso unitário. Por exemplo, a peça de código 53 apresenta um peso unitário de 8,39kg.

Tabela A3.2 - Informações dos perfis das peças impressas pelo código de programação
(segundo arquivo) (Fonte: autor)

n. peça, peso unitario	
1 7.470	50 6.740
2 0.810	51 0.810
3 0.770	52 9.220
4 0.600	53 8.390
5 1.070	54 6.820
6 7.960	55 9.280
7 8.060	56 2.780
8 3.820	57 7.010
9 6.130	58 7.350
10 4.370	59 5.000
11 7.410	60 6.640
12 4.940	61 1.590
13 3.420	62 2.310
14 8.190	63 9.470
15 8.180	64 3.650
16 1.420	65 1.270
17 0.940	66 9.870
18 0.680	67 3.070
19 0.610	68 0.550
20 1.940	69 1.020
21 6.730	70 8.670
22 2.280	71 5.550
23 5.710	72 6.980
24 2.550	73 1.010
25 3.660	74 6.690
26 8.380	75 0.170
27 3.150	76 3.100
28 3.150	77 9.050
29 6.570	78 5.360
30 6.930	79 3.280
31 9.030	80 4.430
32 3.190	81 9.010
33 4.790	82 3.480
34 2.970	83 3.540
35 7.670	84 4.880
36 7.410	85 0.720
37 5.520	86 8.510
38 7.240	87 7.610
39 4.120	88 1.810
40 5.430	89 9.230
41 5.960	90 4.940
42 8.810	91 1.280
43 6.690	92 8.930
44 1.130	93 4.040
45 4.720	94 2.080
46 4.040	95 8.080
47 1.150	96 5.880
48 0.540	97 1.100
49 5.490	98 6.240
	99 9.280
	100 5.410

APÊNDICE 4 SEPARAÇÃO DA MATÉRIA-PRIMA POR CARGA DA EMPILHADEIRA

Este apêndice tem a finalidade de ilustrar a separação da matéria-prima por capacidade da empilhadeira para dar início à fabricação das peças do(s) pedido(s) enviado(s) pelo(s) cliente(s) ao L1.

Por exemplo, considere que no instante t cinco clientes enviaram os pedidos (e, portanto, são cinco listas de 40 tipos de peças).

Existem dois problemas com relação aos momentos em que os pedidos são enviados ao L1:

- Nem sempre os clientes enviam os pedidos ao mesmo tempo, então não é possível fazer o agrupamento dos pedidos (batching orders) para otimizar o processo de fabricação. A abordagem dada neste trabalho de pesquisa é que a separação das matérias-primas das peças a serem fabricadas é feita conforme a ordem de chegada dos pedidos. Dessa forma, toda estocasticidade é mantida. Se porventura vários clientes enviassem os pedidos ao mesmo tempo, então no L1 seria feita a separação da matéria-prima das peças a serem fabricadas um pedido de cada vez;
- Em um único pedido, pode conter duas quantidades distintas do mesmo tipo de peça. Isso acontece quando o cliente decide acrescentar alguma peça posteriormente. A abordagem dada neste trabalho de pesquisa é que a leitura da peça a ser fabricada é feita conforme a ordem das peças que o cliente enviou sem realizar agrupamentos das quantidades de mesmos tipos de peças.

Enfim, dado o esclarecimento sobre as abordagens dadas neste estudo, considere agora que os perfis das peças solicitadas pelos cinco clientes no instante t apresentem até 8 operações e tamanho dos lotes de cada tipo de peça variando de 10 a 20 unidades.

Como os clientes enviam pedidos e neles estão contidas informações das peças e as suas respectivas quantidades a serem fabricadas, o procedimento proposto então faz a identificação das operações e dos ferramentais a serem usados a partir do código da peça.

Ao executar o código de programação referente ao procedimento proposto para a separação das matérias-primas das peças por capacidade da empilhadeira, este então faz a impressão das informações no arquivo externo, conforme mostrada na Figura A4.1. Neste arquivo estão contidas as informações, tais como os tipos das peças, a quantidade solicitada pelo cliente para cada tipo de peça e as respectivas operações e os ferramentais de cada tipo de peça. Conforme a Figura A4.1, de 1 até 0 informa que a carga do primeiro transportador inicial está completada, ou seja, este carregou a matéria-prima da peça do tipo 1 até a matéria-prima da peça tipo 6 e, agora, está pronto para iniciar a movimentação no chão de fábrica para a entrega da matéria-prima. Enquanto isso, o segundo

transportador inicial é usado para carregar a matéria-prima da peça tipo 92 até a matéria-prima da peça tipo 78, e assim por diante. Se o tipo da peça for 101, então significa que não existem mais peças a serem fabricadas (artifício usado para que o código do procedimento encerre a execução).

Por motivos de restrições no uso de variáveis do código de programação, está impresso apenas 60kg e as quantidades de cada tipo de peça variando de 1 até 2. Mas, na realidade, significa 600kg e as quantidades são 10 e 20 peças.

O modelo de simulação, então, realiza a “movimentação” do transportador inicial do L1 até a máquina tipo 5 (referente à primeira operação da peça tipo 1) e faz o descarregamento da caixa no local onde se encontra esta máquina tipo 5. Nesta caixa estão contidas as matérias-primas da peça tipo 1. Em seguida, este é movimentado pelo modelo de simulação até a máquina tipo 10 para o descarregamento da caixa no local onde se encontra esta máquina tipo 10. Nesta caixa estão contidas as matérias-primas da peça tipo 87, e assim sucessivamente até que todas as caixas são descarregadas do transportador inicial e então este retorna ao L1.

Note que neste arquivo também estão impressas informações sobre os ferramentais a serem usados pelas máquinas na transformação das peças. Esta informação é particularmente importante ao operador das máquinas, que neste trabalho de pesquisa é denominado de funcionário tipo 2. Como o operador já está posicionado na máquina, este então precisa desta informação para verificar se existe necessidade de efetuar a troca de ferramenta na máquina.

peca escolhida,Qtde, por carga admissivel de 60kg e	55	1	45 83 96 66 127 39 77 95
operacoes e ferramentais	75	1	74 128
1 1 55	24	2	121 108 72
87 1 103 95	72	1	26 54 98 132 127 112 63 144
28 2 36 132	0		
32 1 55 147	99	1	77 74 46 41 128
38 1 147 26 16 97 39 136 88 122	4	1	12 18
9 1 37 53 115	1	1	55
8 2 118 129	40	1	129
6 1 118 93 66 146 47 98	9	1	37 53 115
0	51	1	65 12 155 79 136
92 1 11 15	68	2	77 154 17 26 109 101
78 1 81 27 38 45 36 109 153	57	1	46 158 11 68 49 99 124 11
70 2 61	38	2	147 26 16 97 39 136 88 122
72 1 26 54 98 132 127 112 63 144	88	2	72 88 113
17 1 31 87 93 131 81 66 48 84	0		
47 1 24 19 67 138 66 58 34	26	2	144 12 98
83 1 71	56	2	87 145 108 71 151 119
49 1 75 81 153 158 128 75 57	0		
88 1 72 88 113	37	1	132 37 54 41
78 1 81 27 38 45 36 109 153	63	2	45 136 48 85
0	46	1	48
78 1 81 27 38 45 36 109 153	70	2	61
50 2 64 79 66 115 33 75	76	2	79 91
83 1 71	19	2	139 22 129 81 98 145 109
15 1 48	27	1	15 148 137 152
51 1 65 12 155 79 136	0		
60 1 63 51 58	71	1	58 31 144 33 89 72
78 4 81 27 38 45 36 109 153	97	1	127 121 116
0	2	1	42
56 1 87 145 108 71 151 119	90	1	98 21 102 17 148 12 131 151
69 2 56 143 41	87	2	103 95
96 2 145 131 45 122	97	1	127 121 116
100 1 129 147 34 37 151 29	40	2	129
68 1 77 154 17 26 109 101	3	1	119 107 91 122 48 114 152
9 2 37 53 115	6	2	118 93 66 146 47 98
50 2 64 79 66 115 33 75	0		
51 2 65 12 155 79 136	43	1	85 93 34 155 94
96 1 145 131 45 122	16	2	111 32 74
0	32	1	55 147

96	1	145 131 45 122
34	1	107 135 96
71	2	58 31 144 33 89 72
78	2	81 27 38 45 36 109 153
20	1	127 149 118
92	2	11 15
59	1	129 52 39 24 34 81
0		
59	1	129 52 39 24 34 81
0		
69	1	56 143 41
10	2	152 13 143 137
29	2	18 95 72 128 26 122 84
17	1	31 87 93 131 81 66 48 84
55	2	45 83 96 66 127 39 77 95
18	1	19 31 11
27	1	15 148 137 152
58	1	108 85
0		
29	1	18 95 72 128 26 122 84
61	2	44
75	1	74 128
8	1	118 129
11	2	153
81	1	118 52 148 23 154 138
62	1	91 44 27 71 115 77 11
75	1	74 128
35	2	53 77 102 151 59 85 147
82	1	151 13 113
0		
10	1	152 13 143 137
7	1	64 108 72
68	1	77 154 17 26 109 101
58	1	108 85
37	2	132 37 54 41
32	1	55 147
66	2	39 82 146 91 71
91	1	29 59
48	1	71 151 31 16 58 27
0		
77	2	146 69 146 104 21 155 157 77
56	2	87 145 108 71 151 119
42	2	143 61
60	2	63 51 58
67	1	112 31 66 76 57 71 119 48
48	1	71 151 31 16 58 27
0		
12	1	68 13 19 71
55	2	45 83 96 66 127 39 77 95
9	2	37 53 115
84	2	111 109 117 94 116 63
93	1	57 68 129
71	1	58 31 144 33 89 72
0		
71	1	58 31 144 33 89 72
50	2	64 79 66 115 33 75
0		
16	1	111 32 74
95	1	69 54 155 54 53
13	2	37
31	1	51 72 26 134
11	2	153
56	2	87 145 108 71 151 119
11	1	153
0		
11	1	153
2	1	42
91	2	29 59
33	1	71 133 14 148 73 71 104
91	1	29 59
36	1	76 39 82 56 135 35 36 108
8	2	118 129
77	1	146 69 146 104 21 155 157 77
93	1	57 68 129
70	2	61
63	2	45 136 48 85
55	1	45 83 96 66 127 39 77 95
16	1	111 32 74
8	2	118 129
34	2	107 135 96
66	2	39 82 146 91 71
0		
90	1	98 21 102 17 148 12 131 151
26	2	144 12 98
69	1	56 143 41
16	1	111 32 74
42	1	143 61
15	1	48
53	2	56 143 101 88 142 157
0		
58	1	108 85
87	1	103 95
49	1	75 81 153 158 128 75 57
37	2	132 37 54 41
21	2	14 51 123 139 19 145
53	1	56 143 101 88 142 157
4	2	12 18
17	1	31 87 93 131 81 66 48 84
0		
6	1	118 93 66 146 47 98
0		
81	2	118 52 148 23 154 138
12	1	68 13 19 71
77	2	146 69 146 104 21 155 157 77
50	1	64 79 66 115 33 75
60	1	63 51 58
0		
60	1	63 51 58
93	2	57 68 129
67	2	112 31 66 76 57 71 119 48
77	2	146 69 146 104 21 155 157 77
90	2	98 21 102 17 148 12 131 151
74	1	37 42 149 23 147 24
0		
74	1	37 42 149 23 147 24
2	1	42
32	1	55 147
79	1	13 39 158 35 56
86	2	17 83 56 45 144 103 97 45
5	2	64 25 85
93	1	57 68 129
69	1	56 143 41
43	2	85 93 34 155 94
0		
70	1	61
63	2	45 136 48 85
97	1	127 121 116
100	1	129 147 34 37 151 29
62	1	91 44 27 71 115 77 11
52	2	73 136
82	1	151 13 113
0		
82	1	151 13 113
96	1	145 131 45 122
49	2	75 81 153 158 128 75 57
24	1	121 108 72
44	2	68
		76 1 79 91
62	2	91 44 27 71 115 77 11
47	1	24 19 67 138 66 58 34
64	1	17 114 46 61 136 137 117 134
97	2	127 121 116
88	1	72 88 113
1	1	55
58	1	108 85
0		
58	1	108 85
6	1	118 93 66 146 47 98
94	2	68 49
86	2	17 83 56 45 144 103 97 45
101		

Figura A4.1 - Separação da matéria-prima por carga da empilhadeira (Fonte:autor)

As informações contidas no arquivo externo, conforme ilustradas na Figura A4.1, têm for finalidade apenas facilitar na visualização. Na realidade, este código de programação gera estas

informações em dois arquivos externos, denominados de ORDERS.txt e IN_CPN.sml, para que o modelo de simulação faça a leitura. O arquivo gerado, denominado ORDERS.txt, contém apenas as informações sobre o código de peça e a separação da matéria-prima por capacidade da empilhadeira, conforme apresentadas a seguir na Figura A4.2. Este, porém, não gera a informação quanto ao tipo de máquina onde a peça será transformada. A informação quanto ao tipo de máquina onde a peça será transformada é informada ao modelo de simulação através do arquivo denominado por IN_CPN.sml.

De uma forma geral, os dados dessa Figura A4.2 podem ser representados por:
a`(b,c,d,e,f,g,h)++.

Em que:

- a: representa a unidade da cor;
- b: representa o número de transportadores iniciais usados saindo do L1;
- c: tipos de peças movimentadas pelo transportador inicial; É este que define o número de descarregamentos das caixas que deve ser efetuado pelo transportador inicial nos locais onde as máquinas estão alocadas; Este número também pode ser entendido como sendo o número total de lotes a serem fabricados;
- d: contador do número de descarregamentos a serem realizados pelo transportador inicial;
- e: tipo de peça;
- f: quantidade de peças que devem ser produzidas. Esta quantidade é o que o cliente informou no pedido. Este é, agora, denominado de tamanho do lote;
- g: peso total do lote. Este peso total é obtido multiplicando a quantidade total de peças do lote e o peso unitário;
- h: Peso das matérias-primas contidas em todas as caixas que o transportador inicial está carregando no momento.

Para uma melhor compreensão, considere o exemplo da segunda linha, a qual informa que: 1`(1,8,2,87,10,7610,46070)++. Essa linha informa, então, que o primeiro transportador inicial “1” deve se deslocar até a máquina para a realização do segundo descarregamento “2” de um total de oito descarregamentos que devem ser realizados, representado por “8”, cuja caixa de matéria-prima a ser descarregada é para a fabricação da peça tipo 87, daí “87”. Esta caixa possui um lote de 10 unidades de matéria-prima, conforme informado em “10”, cujo o peso de todas as 10 unidades é de 76,10kg, daí “7610” e antes do descarregamento desta caixa, a carga deste primeiro transportador inicial é de 460,70kg, conforme informado em “46070”. Com o descarregamento desta caixa na máquina, o peso total do primeiro transportador inicial passa a ser $46070 - 7610 = 38460$ ou 384,6kg, conforme mostrada no final da terceira linha.

1^(1,8,1,1,10,7470,53540)++	1^(15,5,4,33,10,4790,10500)++
1^(1,8,2,87,10,7610,46070)++	1^(15,5,5,23,10,5710,5710)++
1^(1,8,3,28,20,6300,38460)++	1^(16,6,1,70,20,17340,57810)++
1^(1,8,4,32,10,3190,32160)++	1^(16,6,2,63,20,18940,40470)++
1^(1,8,5,38,10,7240,28970)++	1^(16,6,3,55,10,9280,21530)++
1^(1,8,6,9,10,6130,21730)++	1^(16,6,4,75,10,170,12250)++
1^(1,8,7,8,20,7640,15600)++	1^(16,6,5,24,20,5100,12080)++
1^(1,8,8,6,10,7960,7960)++	1^(16,6,6,72,10,6980,6980)++
1^(2,10,1,92,10,8930,56899)++	1^(17,10,1,99,10,9280,55930)++
1^(2,10,2,78,10,5360,47970)++	1^(17,10,2,4,10,600,46650)++
1^(2,10,3,70,20,17340,42610)++	1^(17,10,3,1,10,7470,46050)++
1^(2,10,4,72,10,6980,25270)++	1^(17,10,4,40,10,5430,38580)++
1^(2,10,5,17,10,940,18289)++	1^(17,10,5,9,10,6130,33150)++
1^(2,10,6,47,10,1150,17349)++	1^(17,10,6,51,10,810,27020)++
1^(2,10,7,83,10,3540,16199)++	1^(17,10,7,68,20,1100,26210)++
1^(2,10,8,49,10,5490,12659)++	1^(17,10,8,57,10,7010,25110)++
1^(2,10,9,88,10,1810,7169)++	1^(17,10,9,38,20,14480,18100)++
1^(2,10,10,78,10,5360,5359)++	1^(17,10,10,88,20,3620,3620)++
1^(3,7,1,78,10,5360,59450)++	1^(18,2,1,26,20,16760,22320)++
1^(3,7,2,50,20,13480,54090)++	1^(18,2,2,56,20,5560,5560)++
1^(3,7,3,83,10,3540,40610)++	1^(19,7,1,37,10,5520,56410)++
1^(3,7,4,15,10,8179,37070)++	1^(19,7,2,63,20,18940,50890)++
1^(3,7,5,51,10,810,28890)++	1^(19,7,3,46,10,4040,31950)++
1^(3,7,6,60,10,6640,28080)++	1^(19,7,4,70,20,17340,27910)++
1^(3,7,7,78,40,21440,21440)++	1^(19,7,5,76,20,6200,10570)++
1^(4,9,1,56,10,2780,55779)++	1^(19,7,6,19,20,1220,4370)++
1^(4,9,2,69,20,2040,53000)++	1^(19,7,7,27,10,3150,3150)++
1^(4,9,3,96,20,11760,50960)++	1^(20,9,1,71,10,5550,56270)++
1^(4,9,4,100,10,5410,39200)++	1^(20,9,2,97,10,1100,50720)++
1^(4,9,5,68,10,550,33790)++	1^(20,9,3,2,10,810,49620)++
1^(4,9,6,9,20,12260,33240)++	1^(20,9,4,90,10,4940,48810)++
1^(4,9,7,50,20,13480,20980)++	1^(20,9,5,87,20,15220,43870)++
1^(4,9,8,51,20,1620,7500)++	1^(20,9,6,97,10,1100,28650)++
1^(4,9,9,96,10,5880,5880)++	1^(20,9,7,40,20,10860,27550)++
1^(5,7,1,96,10,5880,55470)++	1^(20,9,8,3,10,770,16690)++
1^(5,7,2,34,10,2970,49589)++	1^(20,9,9,6,20,15920,15920)++
1^(5,7,3,71,20,11100,46620)++	1^(21,8,1,43,10,6690,56740)++
1^(5,7,4,78,20,10720,35520)++	1^(21,8,2,16,20,2840,50050)++
1^(5,7,5,20,10,1940,24800)++	1^(21,8,3,32,10,3190,47210)++
1^(5,7,6,92,20,17860,22860)++	1^(21,8,4,55,10,9280,44020)++
1^(5,7,7,59,10,5000,5000)++	1^(21,8,5,16,10,1420,34740)++
1^(6,1,1,59,10,5000,5000)++	1^(21,8,6,8,20,7640,33320)++
1^(7,8,1,69,10,1020,53580)++	1^(21,8,7,34,20,5940,25680)++
1^(7,8,2,10,20,8740,52560)++	1^(21,8,8,66,20,19740,19740)++
1^(7,8,3,29,20,13140,43820)++	1^(22,7,1,90,10,4940,57910)++
1^(7,8,4,17,10,940,30680)++	1^(22,7,2,26,20,16760,52970)++
1^(7,8,5,55,20,18560,29740)++	1^(22,7,3,69,10,1020,36210)++
1^(7,8,6,18,10,680,11180)++	1^(22,7,4,16,10,1420,35190)++
1^(7,8,7,27,10,3150,10500)++	1^(22,7,5,42,10,8810,33770)++
1^(7,8,8,58,10,7350,7350)++	1^(22,7,6,15,10,8179,24960)++
1^(8,10,1,29,10,6570,58870)++	1^(22,7,7,53,20,16780,16780)++
1^(8,10,2,61,20,3180,52300)++	1^(23,8,1,58,10,7350,55480)++
1^(8,10,3,75,10,170,49120)++	1^(23,8,2,87,10,7610,48130)++
1^(8,10,4,8,10,3820,48950)++	1^(23,8,3,49,10,5490,40520)++
1^(8,10,5,11,20,14820,45130)++	1^(23,8,4,37,20,11040,35030)++
1^(8,10,6,81,10,9010,30310)++	1^(23,8,5,21,20,13460,23990)++
1^(8,10,7,62,10,2310,21300)++	1^(23,8,6,53,10,8390,10530)++
1^(8,10,8,75,10,170,18990)++	1^(23,8,7,4,20,1200,2140)++
1^(8,10,9,35,20,15340,18820)++	1^(23,8,8,17,10,940,940)++
1^(8,10,10,82,10,3480,3480)++	1^(24,1,1,6,10,7960,7960)++
1^(9,9,1,10,10,4370,56120)++	1^(25,5,1,81,20,18020,54440)++
1^(9,9,2,7,10,8060,51750)++	1^(25,5,2,12,10,4940,36420)++
1^(9,9,3,68,10,550,43690)++	1^(25,5,3,77,20,18100,31480)++
1^(9,9,4,58,10,7350,43140)++	1^(25,5,4,50,10,6740,13380)++
1^(9,9,5,37,20,11040,35790)++	1^(25,5,5,60,10,6640,6640)++
1^(9,9,6,32,10,3190,24750)++	1^(26,6,1,60,10,6640,55530)++
1^(9,9,7,66,20,19740,21560)++	1^(26,6,2,93,20,8080,48890)++
1^(9,9,8,91,10,1280,1820)++	1^(26,6,3,67,20,6140,40810)++
1^(9,9,9,48,10,540,540)++	1^(26,6,4,77,20,18100,34670)++
1^(10,6,1,77,20,18100,58170)++	1^(26,6,5,90,20,9880,16569)++
1^(10,6,2,56,20,5560,40070)++	1^(26,6,6,74,10,6690,6689)++
1^(10,6,3,42,20,17620,34510)++	1^(27,9,1,74,10,6690,51570)++
1^(10,6,4,60,20,13280,16890)++	1^(27,9,2,2,10,810,44880)++
1^(10,6,5,67,10,3070,3610)++	1^(27,9,3,32,10,3190,44070)++
1^(10,6,6,48,10,540,540)++	1^(27,9,4,79,10,3280,40880)++

1^(11,6,1,12,10,4940,55110)++	1^(27,9,5,86,20,17020,37600)++
1^(11,6,2,55,20,18560,50170)++	1^(27,9,6,5,20,2140,20580)++
1^(11,6,3,9,20,12260,31610)++	1^(27,9,7,93,10,4040,18440)++
1^(11,6,4,84,20,9760,19350)++	1^(27,9,8,69,10,1020,14400)++
1^(11,6,5,93,10,4040,9590)++	1^(27,9,9,43,20,13380,13380)++
1^(11,6,6,71,10,5550,5550)++	1^(28,7,1,70,10,8670,58350)++
1^(12,2,1,71,10,5550,19030)++	1^(28,7,2,63,20,18940,49680)++
1^(12,2,2,50,20,13480,13480)++	1^(28,7,3,97,10,1100,30740)++
1^(13,7,1,16,10,1420,53160)++	1^(28,7,4,100,10,5410,29640)++
1^(13,7,2,95,10,8080,51740)++	1^(28,7,5,62,10,2310,24230)++
1^(13,7,3,13,20,6840,43660)++	1^(28,7,6,52,20,18440,21920)++
1^(13,7,4,31,10,9030,36820)++	1^(28,7,7,82,10,3480,3480)++
1^(13,7,5,11,20,14820,27790)++	1^(29,13,1,82,10,3480,56500)++
1^(13,7,6,56,20,5560,12970)++	1^(29,13,2,96,10,5880,53020)++
1^(13,7,7,11,10,7410,7410)++	1^(29,13,3,49,20,10980,47140)++
1^(14,11,1,11,10,7410,57790)++	1^(29,13,4,24,10,2550,36160)++
1^(14,11,2,2,10,810,50380)++	1^(29,13,5,44,20,2260,33610)++
1^(14,11,3,91,20,2560,49570)++	1^(29,13,6,76,10,3100,31350)++
1^(14,11,4,33,10,4790,47010)++	1^(29,13,7,62,20,4620,28250)++
1^(14,11,5,91,10,1280,42220)++	1^(29,13,8,47,10,1150,23630)++
1^(14,11,6,36,10,7410,40940)++	1^(29,13,9,64,10,3650,22480)++
1^(14,11,7,8,20,7640,33530)++	1^(29,13,10,97,20,2200,18830)++
1^(14,11,8,77,10,9050,25890)++	1^(29,13,11,88,10,1810,16630)++
1^(14,11,9,93,10,4040,16840)++	1^(29,13,12,1,10,7470,14820)++
1^(14,11,10,13,20,6840,12800)++	1^(29,13,13,58,10,7350,7350)++
1^(14,11,11,41,10,5960,5960)++	1^(30,4,1,58,10,7350,36490)++
1^(15,5,1,81,20,18020,51420)++	1^(30,4,2,6,10,7960,29139)++
1^(15,5,2,11,20,14820,33400)++	1^(30,4,3,94,20,4160,21180)++
1^(15,5,3,93,20,8080,18580)++	1^(30,4,4,86,20,17020,17020)++

Figura A4.2 - Informações impressas pelo código de programação para a separação da matéria-prima por carga da empilhadeira (Fonte: autor)

APÊNDICE 5 APLICAÇÃO DO CÓDIGO DE PROGRAMAÇÃO PARA A OBTENÇÃO DAS POSIÇÕES DO VÃO VAZIO

Este apêndice apresenta alguns exemplos de como o procedimento proposto gera todas as posições do vão vazio nos locais do chão de fábrica.

Considere que um conjunto de 20 amostras de cada tipo de arranjo físico tenham sido geradas. Os tipos de arranjos físicos são o arranjo físico funcional, o arranjo físico aleatoriamente distribuído e o arranjo físico parcialmente distribuído. Considere também que apenas uma amostra do arranjo físico maximamente distribuído tenha sido gerada.

Após submeter essas amostras de cada tipo de arranjo físico, tem-se então os resultados do arranjo físico funcional, do arranjo físico aleatoriamente distribuído, do arranjo físico parcialmente distribuído e do arranjo físico maximamente distribuído. Ao realizar a comparação dos resultados, verifica-se que o desempenho do arranjo físico aleatoriamente distribuído é o melhor.

O arranjo físico aleatoriamente distribuído com o melhor desempenho tem a seguinte característica: 0 2 2 2 5 6 6 6 5 4 4 4 3 1 1 4, onde “0” representa o vão vazio. De acordo com a representação, significa que o vão vazio está na posição 1, a máquina tipo 2 está na posição 2, e assim por diante. Este arranjo físico deve estar impresso em um arquivo externo, em txt.

O código de programação do procedimento proposto inicia-se com a entrada do número de tipos de máquinas, etapa importante para efetuar a leitura das réplicas de cada tipo de máquina. Deve-se inserir também o formato desejado dos locais a serem formados no chão de fábrica (em x e em y), conforme mostrado na Figura A5.1.

Com a leitura das réplicas de cada tipo de máquina, o código de programação faz a soma destas para então definir o tamanho do vetor que armazena o melhor arranjo físico obtido. Em seguida, o código de programação faz a leitura do arranjo físico aleatoriamente distribuído localizado no arquivo externo, cuja representação é: 0 2 2 2 5 6 6 6 5 4 4 4 3 1 1 4.

Dando continuidade à execução do código de programação referente ao procedimento proposto, este, então, gera todas as posições possíveis, conforme mostrada na Figura A5.1.

```

C:\DOCUME~1\shih\MEUSDO~1\TP\BIN\TURBO.EXE
6
Entre com a configuracao em x
4
Entre com a configuracao em y
4
Entre com o tipo de layout analisado
Funcional<1>; Aleatorio<2>; Parcial <3>; maximamente<4>
2
0222566654443114
2022566654443114
2202566654443114
2220566654443114
2225066654443114
2225606654443114
2225660654443114
2225666054443114
2225666504443114
2225666540443114
2225666544043114
2225666544403114
2225666544430114
2225666544431014
2225666544431104
2225666544431140

```

Figura A5.1 - Tela de execução do código de programação para a geração das possíveis posições do vão vazio (Fonte: autor)

De acordo com a Figura A5.1, cada linha representa um novo arranjo físico aleatoriamente distribuído, iniciando por “0 2 2 2 5 6 6 6 5 4 4 4 3 1 1 4”, depois o próximo arranjo físico aleatoriamente distribuído é “ 2 0 2 2 5 6 6 6 5 4 4 4 3 1 1 4, e assim por diante. Ao todo, são 16 novos arranjos físicos aleatoriamente distribuídos.

Por tratar-se do arranjo físico aleatoriamente distribuído, a posição do vão vazio pode estar em qualquer posição do vetor, pois não existe problema de separação de máquinas do mesmo tipo.

Ainda de acordo com a Figura A5.1, cada novo arranjo físico aleatoriamente distribuído pode ser representado conforme a Figura A5.2, para melhor visualização. Note que cada representação da vista superior do arranjo físico aleatoriamente distribuído corresponde a cada linha do arranjo físico ilustrado na Figura A5.1.

Agora, os arranjos físicos aleatoriamente distribuídos obtidos devem ser submetidos ao modelo de simulação. A idéia, portanto, de propôr este procedimento é a de melhorar a solução inicialmente obtida pelo modelo de simulação, que era de “0 2 2 2 5 6 6 6 5 4 4 4 3 1 1 4”.

Figura A5.2 - Representação das vistas superiores dos arranjos físicos aleatoriamente distribuídos com as posições dos vãos vazios (Fonte: autor)

2	2	2	5
6	6	6	5
4	4	4	3
0	1	1	4

2	2	2	5
6	6	6	5
4	4	4	3
1	0	1	4

2	2	2	5
6	6	6	5
4	4	4	3
1	1	0	4

2	2	2	5
6	6	6	5
4	4	4	3
1	1	4	0

Figura A5.2 - Representação das vistas superiores dos arranjos físicos aleatoriamente distribuídos com as posições dos vãos vazios – continuação (Fonte: autor)

O código de programação também deve gerar, em arquivo externo (em txt), as informações dos arranjos físicos aleatoriamente distribuídos para que o modelo de simulação faça a leitura. A Figura A5.3 mostra o arquivo externo, onde nele estão impressas as informações correspondentes às 16 possibilidades de posição do vão vazio e, portanto, são 16 novos arranjos físicos aleatoriamente distribuídos. Note então que LMList varia de 1 até 16, tanto para fun DefLayoutInt(laytIn) quanto para fun DefLayoutInter(laytInter).

```

fun DefLayoutIn(laytIn) =
let
  val LMList1 = [0,2,2,2,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList2 = [2,0,2,2,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList3 = [2,2,0,2,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList4 = [2,2,2,0,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList5 = [2,2,2,5,0,6,6,6,5,4,4,4,3,1,1,4];
  val LMList6 = [2,2,2,5,6,0,6,6,5,4,4,4,3,1,1,4];
  val LMList7 = [2,2,2,5,6,6,0,6,5,4,4,4,3,1,1,4];
  val LMList8 = [2,2,2,5,6,6,6,0,5,4,4,4,3,1,1,4];
  val LMList9 = [2,2,2,5,6,6,6,5,0,4,4,4,3,1,1,4];
  val LMList10 = [2,2,2,5,6,6,6,5,4,0,4,4,3,1,1,4];
  val LMList11 = [2,2,2,5,6,6,6,5,4,4,0,4,3,1,1,4];
  val LMList12 = [2,2,2,5,6,6,6,5,4,4,4,0,3,1,1,4];
  val LMList13 = [2,2,2,5,6,6,6,5,4,4,4,3,0,1,1,4];
  val LMList14 = [2,2,2,5,6,6,6,5,4,4,4,3,1,0,1,4];
  val LMList15 = [2,2,2,5,6,6,6,5,4,4,4,3,1,1,0,4];
  val LMList16 = [2,2,2,5,6,6,6,5,4,4,4,3,1,1,4,0];
in
  if laytIn = 1 then LMList1 else
  if laytIn = 2 then LMList2 else
  if laytIn = 3 then LMList3 else
  if laytIn = 4 then LMList4 else
  if laytIn = 5 then LMList5 else
  if laytIn = 6 then LMList6 else

```

```

if laytIn = 7 then LMList7 else
if laytIn = 8 then LMList8 else
if laytIn = 9 then LMList9 else
if laytIn = 10 then LMList10 else
if laytIn = 11 then LMList11 else
if laytIn = 12 then LMList12 else
if laytIn = 13 then LMList13 else
if laytIn = 14 then LMList14 else
if laytIn = 15 then LMList15 else
LMList16

end;

fun DefLayoutInter(laytInter) =
let
  val LMList1 = [0,2,2,2,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList2 = [2,0,2,2,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList3 = [2,2,0,2,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList4 = [2,2,2,0,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList5 = [2,2,2,5,0,6,6,6,5,4,4,4,3,1,1,4];
  val LMList6 = [2,2,2,5,6,0,6,6,5,4,4,4,3,1,1,4];
  val LMList7 = [2,2,2,5,6,6,0,6,5,4,4,4,3,1,1,4];
  val LMList8 = [2,2,2,5,6,6,6,0,5,4,4,4,3,1,1,4];
  val LMList9 = [2,2,2,5,6,6,6,5,0,4,4,4,3,1,1,4];
  val LMList10 = [2,2,2,5,6,6,6,5,4,0,4,4,3,1,1,4];
  val LMList11 = [2,2,2,5,6,6,6,5,4,4,0,4,3,1,1,4];
  val LMList12 = [2,2,2,5,6,6,6,5,4,4,4,0,3,1,1,4];
  val LMList13 = [2,2,2,5,6,6,6,5,4,4,4,3,0,1,1,4];
  val LMList14 = [2,2,2,5,6,6,6,5,4,4,4,3,1,0,1,4];
  val LMList15 = [2,2,2,5,6,6,6,5,4,4,4,3,1,1,0,4];
  val LMList16 = [2,2,2,5,6,6,6,5,4,4,4,3,1,1,4,0];

in

if laytInter = 1 then LMList1 else
if laytInter = 2 then LMList2 else
if laytInter = 3 then LMList3 else
if laytInter = 4 then LMList4 else
if laytInter = 5 then LMList5 else
if laytInter = 6 then LMList6 else
if laytInter = 7 then LMList7 else
if laytInter = 8 then LMList8 else
if laytInter = 9 then LMList9 else
if laytInter = 10 then LMList10 else
if laytInter = 11 then LMList11 else
if laytInter = 12 then LMList12 else
if laytInter = 13 then LMList13 else
if laytInter = 14 then LMList14 else
if laytInter = 15 then LMList15 else
LMList16

end;

```

Figura A5.3 - Informações impressas ao arquivo externo das diversas posições do vão vazio para o modelo de simulação (Fonte: autor)

Em virtude do melhor arranjo físico aleatoriamente distribuído (inicialmente obtido após a execução do modelo de simulação) possuir várias réplicas do mesmo tipo de máquina próximas uma das outras, ou seja, 0 2 2 2 5 6 6 6 5 4 4 4 3 1 1 4, este também pode ser usado para ilustrar como o procedimento gera as possíveis posições do vão vazio para o arranjo físico funcional.

A representação da vista superior de “0 2 2 2 5 6 6 6 5 4 4 4 3 1 1 4” é similar ao que está ilustrada na Figura A5.4.

0	2	2	2
5	6	6	6
5	4	4	4
3	1	1	4

Figura A5.4 - Representação da vista superior do melhor arranjo físico obtido pelo modelo de simulação (Fonte: autor)

Para o arranjo físico funcional, como as réplicas dos mesmos tipos de máquinas devem estar alocadas próximas uma das outras para a formação do departamento, a posição a ser alocada do vão vazio não pode ocorrer de modo que o vão vazio separe as réplicas do mesmo tipo de máquina. Deve-se, então, criar uma maneira para realizar a leitura. A maneira proposta é justamente fazer a leitura seguindo a curva de preenchimento espacial, resultando em “0 2 2 2 6 6 6 5 5 4 4 4 1 1 3”, ou seja, se for linha par, então a leitura das máquinas é invertida da direita para a esquerda. Com isso, pode-se notar que as réplicas dos mesmos tipos de máquinas não estão separadas entre si.

Note que as informações de x e de y inicialmente inseridas no código de programação só serão usadas para identificar se a linha a ser lida é par ou é ímpar, importantes apenas para o arranjo físico funcional e para o arranjo físico parcialmente distribuído.

Ao informar ao procedimento proposto que as possíveis posições a serem geradas do vão vazio são do arranjo físico funcional, o procedimento proposto, então, faz primeiramente a leitura externa do melhor arranjo físico e, em seguida, faz a inversão da leitura caso a linha for par.

Com a execução do código de programação referente ao procedimento proposto, pode-se obter vários arranjos físicos funcionais, sendo que cada arranjo físico funcional possui uma posição diferente do vão vazio, conforme mostrada na Figura A5.5. Para o exemplo ilustrado, existem sete novos arranjos físicos funcionais, sendo que cada linha representa um arranjo físico funcional.

```

C:\DOCUME~1\shih\MEUSDO~1\TP\BIN\TURBO.EXE
Entre com o numero de tipos de maquinas da fabrica <max. 130>
6
Entre com a configuracao em x
4
Entre com a configuracao em y
4
Entre com o tipo de layout analisado
Funcional<1>; Aleatorio<2>; Parcial<3>; maximamente<4>
1
0222666554444113
2220666554444113
2226660554444113
2226665504444113
2226665544440113
2226665544441103
2226665544441130

```

Figura A5.5 - Tela de execução do código de programação para a geração das possíveis posições do vão vazio no arranjo físico funcional (Fonte: autor)

Em seguida, deve-se fazer novamente a alocação dessas máquinas aos locais de forma invertida. Por exemplo, considere o exemplo do arranjo físico da terceira linha da Figura A5.5, cujo arranjo físico funcional é: 2226660554444113. O procedimento faz então a devida alocação das máquinas nos locais de forma invertida, conforme mostrada na Figura A5.6.

Este procedimento de alocação segue o mesmo padrão para outros seis arranjos físicos funcionais restantes. Após a alocação, todos os arranjos físicos funcionais são armazenados na memória do computador.

2	2	2	6
5	0	6	6
5	4	4	4
3	1	1	4

Figura A5.6 - Alocação invertida do arranjo físico funcional (Fonte: autor)

Em seguida, o código de programação deve fazer a impressão dos sete novos arranjos físicos funcionais para serem lidos pelo modelo de simulação, em arquivo externo de formato txt, conforme mostrada na Figura A5.7. Agora, a leitura de cada arranjo físico funcional, que estão armazenados na memória do computador, é feita normalmente (esquerda para a direita e de cima para baixo). As informações impressas ao arquivo externo, para serem lidas pelo modelo de simulação, podem ser vistas na Figura A5.7. Agora o modelo de simulação pode usar diretamente as distâncias da matriz De-Para porque as máquinas estão alocadas nos locais de forma correta.

```

fun DefLayoutIn(layoutIn) =
let
  val LMList1 = [0,2,2,2,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList2 = [2,2,2,0,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList3 = [2,2,2,6,5,0,6,6,5,4,4,4,3,1,1,4];
  val LMList4 = [2,2,2,6,5,5,6,6,0,4,4,4,3,1,1,4];
  val LMList5 = [2,2,2,6,5,5,6,6,4,4,4,4,3,1,1,0];
  val LMList6 = [2,2,2,6,5,5,6,6,4,4,4,4,3,0,1,1];
  val LMList7 = [2,2,2,6,5,5,6,6,4,4,4,4,0,3,1,1];

in

  if layoutIn = 1 then LMList1 else
  if layoutIn = 2 then LMList2 else
  if layoutIn = 3 then LMList3 else
  if layoutIn = 4 then LMList4 else
  if layoutIn = 5 then LMList5 else
  if layoutIn = 6 then LMList6 else
  LMList7

end;

fun DefLayoutInter(layoutInter) =
let
  val LMList1 = [0,2,2,2,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList2 = [2,2,2,0,5,6,6,6,5,4,4,4,3,1,1,4];
  val LMList3 = [2,2,2,6,5,0,6,6,5,4,4,4,3,1,1,4];
  val LMList4 = [2,2,2,6,5,5,6,6,0,4,4,4,3,1,1,4];
  val LMList5 = [2,2,2,6,5,5,6,6,4,4,4,4,3,1,1,0];
  val LMList6 = [2,2,2,6,5,5,6,6,4,4,4,4,3,0,1,1];
  val LMList7 = [2,2,2,6,5,5,6,6,4,4,4,4,0,3,1,1];

in

  if layoutInter = 1 then LMList1 else
  if layoutInter = 2 then LMList2 else
  if layoutInter = 3 then LMList3 else
  if layoutInter = 4 then LMList4 else
  if layoutInter = 5 then LMList5 else
  if layoutInter = 6 then LMList6 else
  LMList7

end;

```

Figura A5.7 - Informações sobre novos arranjos físicos funcionais impressas ao arquivo externo para serem lidas pelo modelo de simulação (Fonte: autor)

APÊNDICE 6 OUTRAS INFORMAÇÕES RELEVANTES

Este apêndice tem a finalidade de apresentar as informações contidas nos dois arquivos externos, denominados de CrFunc.txt e CrMax.txt, a serem lidas pelo modelo de simulação. Caso houver necessidade, algumas dessas informações devem ser modificadas para dar início à simulação do modelo.

A Figura A6.1 ilustra as informações contidas no arquivo externo, denominado CrFunc.txt, referentes às 18 combinações possíveis para a simulação do arranjo físico funcional. Quando houver a necessidade para a realização de algum teste mais específico no modelo de simulação, estas, então, podem ser modificadas.

```
1`1,5,"~Dist","NoDISTQueueIn","NoDISTQueueInt","","QueueDIST")++
1`2,5,"~Dist","NoDISTQueueIn","NoDISTQueueInt","","DISTQueue")++
1`3,5,"~Dist","NoDISTQueueIn","DISTQueueInt","","DISTInt","QueueDIST")++
1`4,5,"~Dist","NoDISTQueueIn","DISTQueueInt","","DISTInt","DISTQueue")++
1`5,5,"~Dist","NoDISTQueueIn","DISTQueueInt","","QueueInt","QueueDIST")++
1`6,5,"~Dist","NoDISTQueueIn","DISTQueueInt","","QueueInt","DISTQueue")++
1`7,5,"~Dist","DISTQueueIn","NoDISTQueueInt","DISTIn","","QueueDIST")++
1`8,5,"~Dist","DISTQueueIn","NoDISTQueueInt","DISTIn","","DISTQueue")++
1`9,5,"~Dist","DISTQueueIn","DISTQueueInt","DISTIn","DISTInt","QueueDIST")++
1`10,5,"~Dist","DISTQueueIn","DISTQueueInt","DISTIn","DISTInt","DISTQueue")++
1`11,5,"~Dist","DISTQueueIn","DISTQueueInt","DISTIn","QueueInt","QueueDIST")++
1`12,5,"~Dist","DISTQueueIn","DISTQueueInt","DISTIn","QueueInt","DISTQueue")++
1`13,5,"~Dist","DISTQueueIn","NoDISTQueueInt","QueueIn","","QueueDIST")++
1`14,5,"~Dist","DISTQueueIn","NoDISTQueueInt","QueueIn","","DISTQueue")++
1`15,5,"~Dist","DISTQueueIn","DISTQueueInt","QueueIn","DISTInt","QueueDIST")++
1`16,5,"~Dist","DISTQueueIn","DISTQueueInt","QueueIn","DISTInt","DISTQueue")++
1`17,5,"~Dist","DISTQueueIn","DISTQueueInt","QueueIn","QueueInt","QueueDIST")++
1`18,5,"~Dist","DISTQueueIn","DISTQueueInt","QueueIn","QueueInt","DISTQueue")++
```

Figura A6.1 - Informações contidas no arquivo CrFunc.txt (Fonte: autor)

De uma forma genérica, essas informações podem ser escritas da seguinte forma: a`(b,c,"d","e","f","g","h","i"). Sendo que "e" e "g" estão interligadas, o mesmo acontecendo entre "f" e "h". Em que:

- a: representa a unidade da cor;
- b: representa uma das 18 combinações dos critérios que se deseja analisar;
- c: representa o número de amostras do arranjo físico funcional. De acordo com o exemplo da Figura A6.1, são cinco amostras. Se o número de amostras de interesse do arranjo físico funcional para teste em modelo de simulação fosse dez, significa que cada amostra das 10 geradas deve ser submetida a todas as 18 combinações;
- d: define se o arranjo físico é do tipo arranjo físico distribuído ou arranjo físico funcional. Por exemplo, se for "DIST", significa que o arranjo físico a ser submetido ao modelo de simulação é do tipo arranjo físico distribuído. Já se for "~DIST" significa que é "não DIST", ou seja, não

distribuído, então o arranjo físico a ser submetido ao modelo de simulação é do tipo arranjo físico funcional;

- e: representa o critério adotado pelo transportador inicial. Se for “NoDISTQueueIn”, significa que o critério é a primeira máquina disponível. Então, este não ativa o “g”. Se for “DISTQueueIn”, significa que o critério pode ser a máquina de menor distância ou pode ser a máquina que tiver a menor fila formada, então ativa o “g”;
- f: representa o critério adotado pelo transportador intermediário. Se for “NODISTQueueIn”, significa que o critério adotado é a primeira máquina disponível, então este não ativa o “h”. Se for “DISTQueueIn”, significa que o critério pode ser a máquina de menor distância ou pode ser a máquina que tiver a menor fila formada, então ativa o “h”.
- g: este é ativado para definir um critério para o transportador inicial, caso o critério seja diferente da primeira máquina disponível. Se for “DISTIn”, significa que o critério é a máquina de menor distância. Se for “QueueIn”, significa que o critério é a máquina que tiver a menor fila formada. Quando está “”, significa que o “g” está desativado;
- h: este é ativado para definir um critério para o transportador intermediário, caso o critério seja diferente da primeira máquina disponível. Se for “DISTInt”, significa que o critério é a máquina de menor distância. Se for “QueueInt”, significa que o critério é a máquina que tiver a menor fila formada. Quando está “”, significa que o “h” está desativado;
- i: representa o critério adotado pelo transportador final. Se for “QueueDIST”, significa que o critério adotado é a máquina que tiver a maior fila formada e, se houver empate, adota-se o critério da máquina de menor distância. Se for “DISTQueue”, significa que o critério adotado é a máquina que tiver a menor distância e, se houver empate, adota-se o critério de maior fila formada no local onde a máquina está alocada.

Da mesma forma, é necessário informar também ao modelo de simulação quais são os critérios para a coleta de peças acabadas nos arranjos físicos distribuídos, bem como os tamanhos das amostras a serem submetidas ao modelo de simulação.

A Figura A6.2 ilustra as informações gravas em arquivo externo, denominada de CrMax.txt. De uma forma genérica, essas informações contidas no arquivo CrMax.txt podem ser representadas da seguinte forma:

a`(b,c,"d", "", "", "", "", "e")

Em que:

- a: representa a unidade da cor;
- b: representa as possíveis combinações dos critérios no arranjo físico distribuído para a coleta

de peças acabadas. Na realidade, podem ser atribuídos até dois tipos de critérios para a coleta de peças acabadas nos arranjos físicos distribuídos e, então, o “b” deveria variar de 1 até 2. Mas neste estudo, adotou-se um número fixo 1 para o “b”, e alterar “e” diretamente;

- c: representa o número de amostras do arranjo físico distribuído;
- d: define apenas que o arranjo físico é do tipo arranjo físico distribuído, ou seja, DISTribuído;
- e: representa o tipo de critério a ser adotado para a coleta das caixas, onde estão contidas as peças acabadas. Se for DISTQueue, significa que o critério adotado é a máquina de menor distância e, se houver empate, escolher a máquina que tiver a menor fila formada. Se for QueueDIST, significa que o critério é a máquina que tiver a menor fila formada e, caso houver empate, escolher a máquina de menor distância.

```
1`(1,1,"Dist","", "", "", "", "DISTQueue")
```

Figura A6.2 - Informações contidas no arquivo CrMax.txt (Fonte: autor)

APÊNDICE 7 TEMPO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO FUNCIONAL (20 AMOSTRAS)

Este apêndice irá apresentar os resultados do modelo de simulação sujeitando as 20 amostras do arranjo físico funcional para produzir peças que apresentam os seguintes perfis: uma a quatro operações, variação do lote de 10 unidades a 20 unidades, tempos de cada operação variando de 30s a 32s.

Devido à maneira pela qual o modelo de simulação imprime os resultados, é necessário primeiramente fazer uma interpretação destes.

A Figura A7.1 é dividida em duas colunas (a da direita é a continuação da esquerda). Para facilitar na leitura, nomeiam-se os dados dessa figura em a, b e c. A letra “a” representa o tipo de critério usado para a simulação do modelo, então, este varia de 1 até 18 já que existem ao todo 18 critérios possíveis para o arranjo físico funcional. A letra “b” representa a amostra de um a vinte, já que são vinte amostras do arranjo físico funcional. A letra “c” representa o tempo de consolidação do pedido, porém o tempo não pode ser usado diretamente para fins de comparação porque este é acumulativo. Por exemplo, considere a=7 e b=3. Isso significa que o modelo de simulação executou o sétimo critério (dos 18 possíveis) para a amostra 3 (das vinte amostras geradas). O tempo de consolidação dessa amostra é 1692516-1677416=15100s.

Pode-se, então, calcular o tempo médio de consolidação do pedido dos arranjos físicos funcionais, no qual resultou em 14326,04s da Tabela 5.1 no Tópico 5. Portanto, todos os resultados apresentados na figura desse apêndice são usados para preencher apenas uma lacuna da Tabela 5.1.

Os resultados da Tabela 5.1 são, na verdade, apenas uma síntese dos resultados do modelo de simulação para permitir melhor visualização e melhor comparação com outros tipos de arranjos físicos distribuídos.

a	b	c	10	1	2492508
1	1	13546	10	2	2502825
1	2	27136	10	3	2512514
1	3	42609	10	4	2525159
1	4	59202	10	5	2534838
1	5	74785	10	6	2545959
1	6	87455	10	7	2557190
1	7	100734	10	8	2566849
1	8	116682	10	9	2578952
1	9	130146	10	10	2589877
1	10	145656	10	11	2600135
1	11	160513	10	12	2611424
1	12	173862	10	13	2624696
1	13	188619	10	14	2636248
1	14	204790	10	15	2649827
1	15	220258	10	16	2660652
1	16	236485	10	17	2673323
1	17	253716	10	18	2685450
1	18	270013	10	19	2695174
1	19	284511	10	20	2706597
1	20	300445	11	1	2717407
2	1	316470	11	2	2727990

2	2	333401	11	3	2739134
2	3	349236	11	4	2749452
2	4	363551	11	5	2759974
2	5	379177	11	6	2772452
2	6	392574	11	7	2783294
2	7	409659	11	8	2793558
2	8	424442	11	9	2804415
2	9	438408	11	10	2816281
2	10	453608	11	11	2827507
2	11	470790	11	12	2840526
2	12	485818	11	13	2854055
2	13	501566	11	14	2864213
2	14	516793	11	15	2873589
2	15	531091	11	16	2884169
2	16	544275	11	17	2895113
2	17	558316	11	18	2905738
2	18	573327	11	19	2916337
2	19	585949	11	20	2929541
2	20	603480	12	1	2942438
3	1	617361	12	2	2954232
3	2	629753	12	3	2966356
3	3	641848	12	4	2976506
3	4	658223	12	5	2987677
3	5	670090	12	6	2999666
3	6	683172	12	7	3009584
3	7	697146	12	8	3020791
3	8	710784	12	9	3031992
3	9	722158	12	10	3041988
3	10	736685	12	11	3052549
3	11	748554	12	12	3063821
3	12	762976	12	13	3075042
3	13	777833	12	14	3084986
3	14	790492	12	15	3095021
3	15	805053	12	16	3106244
3	16	816225	12	17	3116814
3	17	829840	12	18	3127926
3	18	844001	12	19	3138113
3	19	856026	12	20	3148172
3	20	869094	13	1	3161840
4	1	883386	13	2	3178294
4	2	896151	13	3	3193341
4	3	909999	13	4	3208953
4	4	922745	13	5	3223683
4	5	934556	13	6	3238651
4	6	949148	13	7	3253756
4	7	962254	13	8	3268732
4	8	975597	13	9	3285000
4	9	992200	13	10	3302155
4	10	1006728	13	11	3315765
4	11	1020094	13	12	3328648
4	12	1033879	13	13	3346297
4	13	1048203	13	14	3361058
4	14	1062833	13	15	3377804
4	15	1078568	13	16	3394733
4	16	1089443	13	17	3409259
4	17	1103069	13	18	3426037
4	18	1115472	13	19	3440069
4	19	1128250	13	20	3455634
4	20	1139342	14	1	3470951
5	1	1151857	14	2	3487500
5	2	1165484	14	3	3502058
5	3	1177229	14	4	3516234
5	4	1188105	14	5	3529970
5	5	1199315	14	6	3544443
5	6	1211439	14	7	3558380
5	7	1225091	14	8	3573513
5	8	1237529	14	9	3588527
5	9	1252069	14	10	3605818
5	10	1265431	14	11	3622463
5	11	1279675	14	12	3639895
5	12	1294417	14	13	3659766
5	13	1307690	14	14	3676935
5	14	1321304	14	15	3691104
5	15	1333807	14	16	3708315
5	16	1346470	14	17	3720407

5	17	1360151	14	18	3738131
5	18	1372598	14	19	3752040
5	19	1384951	14	20	3769194
5	20	1397428	15	1	3779212
6	1	1409856	15	2	3789722
6	2	1422816	15	3	3799680
6	3	1436473	15	4	3810233
6	4	1448478	15	5	3822555
6	5	1459865	15	6	3833427
6	6	1471379	15	7	3847007
6	7	1485077	15	8	3856569
6	8	1497850	15	9	3869913
6	9	1512352	15	10	3880225
6	10	1523927	15	11	3890531
6	11	1536096	15	12	3901753
6	12	1549739	15	13	3913856
6	13	1562169	15	14	3924734
6	14	1575263	15	15	3936098
6	15	1586707	15	16	3950321
6	16	1598220	15	17	3961506
6	17	1611478	15	18	3974268
6	18	1622777	15	19	3986119
6	19	1636060	15	20	3998180
6	20	1647298	16	1	4010186
7	1	1662235	16	2	4022128
7	2	1677416	16	3	4032402
7	3	1692516	16	4	4042435
7	4	1707140	16	5	4052967
7	5	1720138	16	6	4064990
7	6	1736216	16	7	4075318
7	7	1750196	16	8	4084430
7	8	1765925	16	9	4095883
7	9	1780433	16	10	4111327
7	10	1795064	16	11	4121605
7	11	1809201	16	12	4132726
7	12	1825494	16	13	4146034
7	13	1840130	16	14	4156778
7	14	1857917	16	15	4167665
7	15	1872640	16	16	4178831
7	16	1887017	16	17	4190564
7	17	1901759	16	18	4202993
7	18	1916525	16	19	4213603
7	19	1929106	16	20	4225739
7	20	1944413	17	1	4236882
8	1	1957714	17	2	4247875
8	2	1972586	17	3	4258146
8	3	1988017	17	4	4269360
8	4	2002132	17	5	4279440
8	5	2016995	17	6	4291623
8	6	2035843	17	7	4303764
8	7	2052220	17	8	4314593
8	8	2065733	17	9	4324280
8	9	2082545	17	10	4334672
8	10	2098078	17	11	4344661
8	11	2116996	17	12	4355292
8	12	2131869	17	13	4365562
8	13	2145691	17	14	4376443
8	14	2161103	17	15	4387629
8	15	2174341	17	16	4399421
8	16	2189482	17	17	4409663
8	17	2204135	17	18	4422068
8	18	2221854	17	19	4432371
8	19	2236015	17	20	4441955
8	20	2251468	18	1	4454404
9	1	2263293	18	2	4464639
9	2	2274484	18	3	4475687
9	3	2285058	18	4	4485215
9	4	2296828	18	5	4495582
9	5	2307451	18	6	4507345
9	6	2319607	18	7	4518263
9	7	2331405	18	8	4529531
9	8	2342297	18	9	4541617
9	9	2356135	18	10	4552793
9	10	2367554	18	11	4563978
9	11	2377515	18	12	4574594

9	12	2389443	18	13	4584858
9	13	2401816	18	14	4597841
9	14	2414509	18	15	4608429
9	15	2423772	18	16	4620547
9	16	2435156	18	17	4631408
9	17	2447256	18	18	4643469
9	18	2458746	18	19	4653627
9	19	2469652	18	20	4663625
9	20	2481118			

Figura A7.1 - Tempo de consolidação do pedido - arranjo físico funcional (Fonte: autor)

APÊNDICE 8 CUSTO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO FUNCIONAL (20 AMOSTRAS)

Este apêndice apresenta outros resultados do modelo de simulação usados para a contabilização do custo de consolidação do pedido, tais como tamanho de filas de caixas, distância percorrida pelas peças etc. Os resultados apresentados referem-se às 20 amostras dos arranjos físicos funcionais submetidas ao modelo de simulação para a produção de peças com os seguintes perfis: uma a quatro operações, variação do lote de 10 unidades a 20 unidades, tempos de cada operação variando de 30s a 32s.

Para distinguir as informações da Figura A8.1, também serão usadas letras, conforme mostradas a seguir:

- a: representa os critérios do arranjo físico funcional, que neste caso varia de um até 18 por ter 18 combinações;
- b: representa o número da amostra, variando de um a 20. Todas as vinte amostras são submetidas ao mesmo critério (dos 18 possíveis) antes de iniciar a simulação do próximo critério;
- c: apresenta o resultado do cálculo da soma, em metros, da distância total percorrida por todas as caixas, onde as peças estão contidas;
- d: apresenta o resultado do cálculo da soma do custo de energia consumida por todas as máquinas para a consolidação do pedido, dado em R\$;
- e: apresenta o resultado do cálculo da soma do custo de mão de obra de todos os funcionários tipo 2, em R\$;
- f: apresenta o resultado do cálculo do custo do funcionário tipo 3, ou seja, o que é encarregado de controlar o transportador intermediário para a movimentação das caixas entre as máquinas. São nestas caixas que estão contidas as peças semi-acabadas. A unidade deste custo também está em R\$;
- g: apresenta o resultado do cálculo da soma do custo do aluguel das empilhadeiras, em R\$;
- h: apresenta o resultado do cálculo da soma do custo de todos os funcionários tipo 1, também em R\$;
- i: apresenta o resultado do cálculo da soma do custo de todos os funcionários tipo 4, em R\$;
- j: apresenta o resultado do cálculo da soma de todos os custos das matérias-primas usadas na fabricação das peças do pedido, em R\$;
- k: apresenta o resultado do cálculo da soma dos custos de todas as novas ferramentas que foram adquiridas para a fabricação das peças do pedido;
- l: apresenta o resultado do cálculo da soma da depreciação de todas as máquinas durante o período em que foram usadas para a fabricação das peças do pedido, em R\$;

- m: apresenta o resultado do cálculo do custo de manutenção de estoque (podendo ser matéria-prima, peças semi-acabadas ou peças acabadas) dos materiais que se encontram no chão de fábrica. A unidade do custo é em R\$;
- n: apresenta o resultado do cálculo da soma de todos os componentes anteriores mais a soma dos combustíveis mais a soma dos lubrificantes resultando, assim, no parâmetro usado neste estudo para fins de comparação de desempenho, que é o custo de consolidação do pedido. Este custo está em R\$;
- o: apresenta o resultado do cálculo da divisão entre o custo total de consolidação do pedido e a quantidade total de peças produzidas, em R\$/peça, que é também chamado de custo unitário médio da peça.

Da mesma forma, todos os resultados dessa Figura A8.1 são usados para preencher apenas uma lacuna da Tabela 5.2 do Tópico 5. Ou seja, o resultado médio de todos custos de consolidação do pedido resulta em R\$4818,65.

a	b h n	c	d i o	e j	k	f l	g m
1	1	320	65.161563225	451.533333333		18.8138888889	15.8036666667
	0.623611111111		0.588888888889	1140.0	91.81875	2899.7260274	105.420918367
	4789.66213588		8.40291602786				
1	2	392	65.214788925	453.0	18.875	15.855	0.681944444444
	1140.0	91.89375	2899.7260274	102.53968254		4788.59209944	8.40103877095
1	3	360	65.0240635	515.766666667		21.4902777778	18.0518333333
	0.645833333333		0.551388888889	1140.0	91.625	2899.7260274	114.48156682
	4867.55558172		8.539571196				
1	4	366	65.1452998167	553.1	23.0458333333	19.3585	0.630555555556
	0.579166666667		1140.0	91.7958333333	2899.7260274	85.1298701299	4878.70722563
	8.55913548357						
1	5	410	65.3108908833	519.433333333		21.6430555556	18.1801666667
	0.677777777778		0.620833333333	1140.0	92.0291666667	2899.7260274	103.536866359
	4861.37783697		8.52873304732				
1	6	322	65.3330682583	422.333333333		17.5972222222	14.7816666667
	0.644444444444		0.576388888889	1140.0	92.0604166667	2899.7260274	106.4
	4759.62512768		8.35021952224				
1	7	372	65.3197618333	442.633333333		18.4430555556	15.4921666667
	0.684722222222		0.5625	1140.0	92.0416666667	2899.7260274	98.2671957672
	4773.36978424		8.37433295481				
1	8	358	65.1305149	531.6	22.15	18.606	0.672222222222
	1140.0	91.775	2899.7260274	108.995535714		4879.38604132	8.56032638829
1	9	410	65.3197618333	448.8	18.7	15.708	0.713888888889
	1140.0	92.0416666667	2899.7260274	86.2037037037		4768.08554527	8.36506236012
1	10	358	65.3434177	517.0	21.5416666667	18.095	0.673611111111
	0.555555555556		1140.0	92.075	2899.7260274	106.382488479	4861.58461911
	8.529095823						
1	11	396	65.1319933917	495.233333333		20.6347222222	17.3331666667
	0.683333333333		0.565277777778	1140.0	91.7770833333	2899.7260274	114.678571429
	4845.97572528		8.50171179874				
1	12	324	65.4439551333	444.966666667		18.5402777778	15.5738333333
	0.598611111111		1140.0	92.2166666667	2899.7260274	99.1758241758	4777.11549386
	8.3809043752						
1	13	412	65.13938585	491.9	20.4958333333	17.2165	0.656944444444
	0.540277777778		1140.0	91.7875	2899.7260274	107.892857143	4835.57611675
	8.48346687148						
1	14	344	65.1305149	539.033333333		22.4597222222	18.8661666667
	0.666666666667		0.6	1140.0	91.775	2899.7260274	103.058035714
	8.56403476579						4881.4998165
1	15	388	64.9737947833	515.6	21.4833333333	18.046	0.625
	1140.0	91.5541666667	2899.7260274	113.168202765		4865.89973192	0.515277777778
1	16	348	65.2428802667	540.9	22.5375	18.9315	0.675
	91.9333333333		2899.7260274	99.453125	4880.16780364	8.56169790113	0.581944444444
1	17	314	65.0314559583	574.366666667		23.9319444444	20.1028333333
	0.698611111111		0.583333333333	1140.0	91.6354166667	2899.7260274	110.12244898
	4926.36701049		8.64274914121				
1	18	344	65.2827995417	543.233333333		22.6347222222	19.0131666667

	0.691666666667	0.490277777778	1140.0	91.9895833333	2899.7260274	116.590909091
	4899.83683563	8.59620497479				
1	19 338	64.9560528833	483.266666667	20.1361111111	16.9143333333	
	0.695833333333	0.547222222222	1140.0	91.5291666667	2899.7260274	95.0
	4812.95254781	8.44377639967				
1	20 436	64.9471819333	531.133333333	22.1305555556	18.5896666667	
	0.681944444444	0.588888888889	1140.0	91.5166666667	2899.7260274	100.034562212
	4869.5824795	8.54312715701				
2	1 324	65.4069928417	534.166666667	22.2569444444	18.6958333333	
	0.611111111111	0.520833333333	1140.0	92.1645833333	2899.7260274	94.3638392857
	4868.08646335	8.54050256728				
2	2 392	65.2133104333	564.366666667	23.5152777778	19.7528333333	
	0.659722222222	0.525 1140.0	91.8916666667	2899.7260274	93.2037815126	
	4899.06435881	8.5948497523				
2	3 364	64.930918525	527.833333333	21.9930555556	18.4741666667	
	0.640277777778	0.630555555556	1140.0	91.49375	2899.7260274	100.9375 4866.85465241
	8.53834149546					
2	4 362	64.8569939417	477.166666667	19.8819444444	16.7008333333	
	0.641666666667	0.623611111111	1140.0	91.3895833333	2899.7260274	100.849753695
	4812.03107639	8.44215978314				
2	5 400	65.1379073583	520.866666667	21.7027777778	18.2303333333	
	0.683333333333	0.556944444444	1140.0	91.7854166667	2899.7260274	88.2142857143
	4847.11805269	8.50371588192				
2	6 338	65.1940900417	446.566666667	18.6069444444	15.6298333333	
	0.615277777778	0.520833333333	1140.0	91.8645833333	2899.7260274	97.5132275132
	4776.41861804	8.37968178604				
2	7 372	65.3448961917	569.5	23.7291666667	19.9325 0.672222222222	
	0.563888888889	0.520833333333	1140.0	92.0770833333	2899.7260274	108.970588235 4920.71572774
	8.63283461006					
2	8 378	64.9338755083	492.766666667	20.5319444444	17.2468333333	
	0.681944444444	0.506944444444	1140.0	91.4979166667	2899.7260274	113.485221675
	4841.57994478	8.49399990312				
2	9 398	65.072853725	465.533333333	19.3972222222	16.2936666667	
	0.711111111111	0.627777777778	1140.0	91.69375	2899.7260274	96.9387755102
	4796.20780594	8.41439965955				
2	10 356	65.2354878083	506.666666667	21.1111111111	17.7333333333	
	0.679166666667	0.591666666667	1140.0	91.9229166667	2899.7260274	101.566820276
	4845.42397699	8.50074381929				
2	11 380	65.2621006583	572.733333333	23.8638888889	20.0456666667 0.725	
	0.623611111111	0.919604166667	2899.7260274	96.7962184874	4911.93990521	
	8.61743843019					
2	12 316	65.1837406	500.933333333	20.8722222222	17.5326666667	
	0.668055555556	0.548611111111	1140.0	91.85	2899.7260274	113.547619048
	4851.03162033	8.51058179006				
2	13 422	65.1142514917	524.933333333	21.8722222222	18.3726666667	
	0.677777777778	0.551388888889	1140.0	91.7520833333	2899.7260274	90.546875
	4853.77277591	8.51539083493				
2	14 354	65.1985255167	507.566666667	21.1486111111	17.7648333333	
	0.683333333333	0.588888888889	1140.0	91.8708333333	2899.7260274	101.107142857
	4845.84457104	8.50148170357				
2	15 404	64.9634453417	476.6	19.8583333333	16.681 0.597222222222	
	0.515277777778	0.539583333333	2899.7260274	80.4926108374	4791.19000384	
	8.40559649797					
2	16 376	65.1807836167	439.466666667	18.3111111111	15.3813333333	
	0.672222222222	0.544444444444	1140.0	91.8458333333	2899.7260274	107.266483516
	4778.59640404	8.38350246323				
2	17 310	65.2029609917	468.033333333	19.5013888889	16.3811666667	
	0.698611111111	0.55 1140.0	91.8770833333	2899.7260274	99.1198979592	
	4801.25659868	8.42325719067				
2	18 334	65.1837406	500.366666667	20.8486111111	17.5128333333	
	0.705555555556	0.526388888889	1140.0	91.85	2899.7260274	80.0714285714
	4816.97024272	8.45082498724				
2	19 352	65.2192244	420.733333333	17.5305555556	14.7256666667 0.725	
	0.569444444444	1140.0 91.9	2899.7260274	96.0857142857	4747.40360288	
	8.32877825067					
2	20 436	65.3759445167	584.366666667	24.3486111111	20.4528333333	
	0.669444444444	0.581944444444	1140.0	92.1208333333	2899.7260274	114.0
	4941.87595765	8.66995782043				
3	1 318	65.0654612667	462.7	19.2791666667	16.1945 0.611111111111	
	0.570833333333	1140.0 91.6833333333	2899.7260274	53.3163265306	4749.31717584	
	8.33213539621					
3	2 374	65.0950311	413.066666667	17.2111111111	14.4573333333	
	0.702777777778	0.579166666667	1140.0	91.725	2899.7260274	71.6571428571
	4714.42068351	8.27091347984				
3	3 342	65.1467783083	403.166666667	16.7986111111	14.1108333333	
	0.615277777778	0.554166666667	1140.0	91.7979166667	2899.7260274	75.2083333333
	4707.30788906	8.25843489309				
3	4 348	65.276885575	545.833333333	22.7430555556	19.1041666667	
	0.633333333333	0.534722222222	1140.0	91.98125	2899.7260274	57.987012987
	4844.00628027	8.49825663205				
3	5 396	65.161563225	395.566666667	16.4819444444	13.8448333333	
	0.676388888889	0.570833333333	1140.0	91.81875	2899.7260274	73.7946428571
	4697.85386655	8.24184888868				
3	6 310	65.1659987	436.066666667	18.1694444444	15.2623333333	
	0.626388888889	0.526388888889	1140.0	91.825	2899.7260274	72.8159340659

3	4740.35031139 7 350 0.60138888889 8.35345172851	8.31640405506 65.1482568 1140.0 91.8	465.8 2899.7260274	19.4083333333 61.7984693878	16.303 4761.46748525	0.694444444444
3	8 366 0.561111111111 8.32140953237	65.1911330583 1140.0 91.8604166667	454.6 2899.7260274	18.9416666667 2899.7260274	15.911 55.5423280423	0.673611111111 4743.20343345
3	9 404 0.697222222222 4686.08309128	65.0299774667 0.626388888889 8.22119840575	379.1333333333 1140.0	15.7972222222 91.6333333333	13.2696666667 2899.7260274	79.9534161491
3	10 338 0.695833333333 4772.9222602	65.0033646167 0.544444444444 8.37354782492	484.2333333333 1140.0	20.1763888889 91.5958333333	16.9481666667 2899.7260274	53.8177339901
3	11 376 0.708333333333 4697.80614507	65.108337525 0.558333333333 8.24176516679	395.6333333333 1140.0	16.4847222222 91.74375	13.8471666667 2899.7260274	73.7946428571
3	12 306 0.705555555556 4775.28294938	65.0270204833 0.581944444444 8.37768938488	480.7333333333 1140.0	20.0305555556 91.6291666667	16.8256666667 2899.7260274	59.8596938776
3	13 386 0.661111111111 4791.09004103	65.188176075 0.536111111111 8.40542112462	495.2333333333 1140.0	20.6347222222 91.85625	17.3331666667 2899.7260274	59.7142857143
3	14 348 0.655555555556 4719.77123554	65.276885575 0.584722222222 8.28030041322	421.9666666667 1140.0	17.5819444444 91.98125	14.7688333333 2899.7260274	67.0428571429
3	15 376 0.645833333333 4785.94150877	65.2221813833 0.520833333333 8.39638861187	485.3666666667 1140.0	20.2236111111 91.9041666667	16.9878333333 2899.7260274	65.1428571429
3	16 350 0.602777777778 8.18705124734	65.0033646167 1140.0	372.4 91.5958333333	15.5166666667 2899.7260274	13.034 67.8571428571	0.695833333333 4666.61921098
3	17 300 0.720833333333 4756.54675384	65.409949825 0.619444444444 8.34481886638	453.8333333333 1140.0	18.9097222222 92.16875	15.8841666667 2899.7260274	69.1137566138
3	18 332 0.705555555556 4771.65181925	64.9900581917 0.501388888889 8.37131898114	472.0333333333 1140.0	19.6680555556 91.5770833333	16.5211666667 2899.7260274	65.7512315271
3	19 336 0.701388888889 4695.68626531	65.2133104333 0.531944444444 8.23804607949	400.8333333333 1140.0	16.7013888889 91.8916666667	14.0291666667 2899.7260274	65.8779761905
3	20 422 91.78125 2899.7260274	65.134950375 58.4615384615	435.6 4725.63147159	18.15 15.246	0.65 8.2905815291	0.655555555556 1140.0
4	1 308 1140.0	65.1231224417 91.7645833333	476.4 2899.7260274	19.85 55.7397959184	16.674 4766.57730851	0.615277777778 0.519444444444 8.36241633072
4	2 390 0.593055555556 8.27113200714	64.948660425 1140.0	425.5 91.51875	17.7291666667 2899.7260274	14.8925 58.7225274725	0.705555555556 4714.54524407
4	3 348 0.522222222222 8.34276832528	64.966402325 1140.0	461.6 91.54375	19.2333333333 2899.7260274	16.156 60.8201058201	0.623611111111 4755.37794541
4	4 346 0.636111111111 4708.90030153	65.330111275 0.545833333333 8.26122859917	424.8666666667 1140.0	17.7027777778 92.05625	14.8703333333 2899.7260274	52.9807692308
4	5 390 1140.0	65.2221813833 91.9041666667	393.7 2899.7260274	16.4041666667 75.2083333333	13.7795 4697.39226534	0.663888888889 0.575 8.24103906199
4	6 306 1140.0	65.037369925 91.64375	486.4 2899.7260274	20.2666666667 60.1354679803	17.024 4781.57643404	0.629166666667 0.55 8.38873058603
4	7 346 0.673611111111 4733.52877567	65.2073964667 0.506944444444 8.30443644854	436.8666666667 1140.0	18.2027777778 91.8833333333	15.2903333333 2899.7260274	15.2903333333 64.9862637363
4	8 360 0.669444444444 4742.48147376	65.2177459083 0.568055555556 8.32014293642	444.7666666667 1140.0	18.5319444444 91.8979166667	15.5668333333 2899.7260274	65.3439153439
4	9 400 0.715277777778 4860.34540653	65.3596811083 0.568055555556 8.52692176584	553.4333333333 1140.0	23.0597222222 92.0979166667	19.3701666667 2899.7260274	65.8008658009
4	10 352 0.679166666667 4784.13095282	65.0787676917 0.547222222222 8.39321219794	484.2666666667 1140.0	20.1777777778 91.7020833333	16.9493333333 2899.7260274	64.815270936
4	11 374 0.694444444444 4729.21330952	65.0935526083 0.563888888889 8.29686545529	445.5333333333 1140.0	18.5638888889 91.7229166667	15.5936666667 2899.7260274	51.5211640212
4	12 304 0.559722222222 8.35181267355	65.1837406 1140.0	459.5 91.85	19.1458333333 2899.7260274	16.0825 67.6058201058	0.716666666667 4760.53322393
4	13 394 0.668055555556 4781.19933379	65.3611596 0.501388888889 8.38806900664	477.4666666667 1140.0	19.8944444444 92.1	16.7113333333 2899.7260274	68.5591133005
4	14 338 0.672222222222 4786.07158181	65.1305149 0.588888888889 8.39661681019	487.6666666667 1140.0	20.3194444444 91.775	17.0683333333 2899.7260274	62.9433497537
4	15 376 1140.0	64.8170746667 91.3333333333	524.5 2899.7260274	21.8541666667 60.4352678571	18.3575 4822.38181277	0.631944444444 0.525 8.46031896976
4	16 338 1140.0	65.0772892 91.7	362.5 2899.7260274	15.1041666667 83.0442176871	12.6875 4671.31894626	0.673611111111 0.625 8.19529639695

4	17	302	65.1733911583	454.2	18.925	15.897	0.711111111111	0.577777777778
	1140.0	91.8354166667	2899.7260274	66.4030612245	4753.61062714	8.3396677669		
4	18	330	65.0920741167	413.433333333	17.2263888889	14.4701666667		
	0.661111111111	0.547222222222	1140.0	91.7208333333	2899.7260274	75.7738095238		
	4718.82781359	8.27864528701						
4	19	330	65.4158637917	425.933333333	17.7472222222	14.9076666667		
	0.693055555556	0.604166666667	1140.0	92.1770833333	2899.7260274	60.8104395604		
	4718.19170553	8.27752930794						
4	20	424	65.1600847333	369.733333333	15.4055555556	12.9406666667		
	0.673611111111	0.558333333333	1140.0	91.8166666667	2899.7260274	77.7272727273		
	4673.96877312	8.19994521601						
5	1	324	65.13938585	417.166666667	17.3819444444	14.6008333333	0.6125	
	0.519444444444	1140.0	91.7875	2899.7260274	72.7428571429	4719.85079088		
	8.280439984							
5	2	386	65.05067635	454.233333333	18.9263888889	15.8981666667		
	0.681944444444	0.576388888889	1140.0	91.6625	2899.7260274	70.6216931217		
	4757.58397649	8.34663855525						
5	3	358	65.161563225	391.5	16.3125	13.7025	0.616666666667	0.529166666667
	1140.0	91.81875	2899.7260274	71.5327380952	4691.09176425	8.22998555132		
5	4	370	65.2443587583	362.533333333	15.1055555556	12.6886666667		
	0.645833333333	0.601388888889	1140.0	91.9354166667	2899.7260274	72.7922077922		
	4661.47107139	8.1780194235						
5	5	414	65.1068590333	373.666666667	15.5694444444	13.0783333333		
	0.666666666667	0.563888888889	1140.0	91.7416666667	2899.7260274	75.5681818182		
	4675.90959752	8.20335017108						
5	6	324	65.1423428333	404.133333333	16.8388888889	14.1446666667	0.6125	
	0.551388888889	1140.0	91.7916666667	2899.7260274	77.1875	4710.30194628		
	8.26368762504							
5	7	368	65.2251383667	455.066666667	18.9611111111	15.9273333333		
	0.658333333333	0.594444444444	1140.0	91.9083333333	2899.7260274	55.2551020408		
	4743.51970123	8.32196438812						
5	8	368	64.957531375	414.6	17.275	14.511	0.691666666667	0.536111111111
	1140.0	91.53125	2899.7260274	59.1714285714	4703.19722632	8.25122320407		
5	9	400	65.010757075	484.666666667	20.1944444444	16.9633333333		
	0.694444444444	0.623611111111	1140.0	91.60625	2899.7260274	58.0295566502		
	4777.72945112	8.3819814932						
5	10	356	65.2251383667	445.4	18.5583333333	15.589	0.683333333333	
	0.579166666667	1140.0	91.9083333333	2899.7260274	60.8104395604	4738.67055239		
	8.31345710946							
5	11	392	65.1778266333	474.8	19.7833333333	16.618	0.723611111111	
	0.597222222222	1140.0	91.8416666667	2899.7260274	63.1773399015	4772.65510007		
	8.37307912292							
5	12	312	65.0669397583	491.4	20.475	17.199	0.695833333333	0.531944444444
	1140.0	91.6854166667	2899.7260274	55.2216748768	4782.16903728	8.38977024084		
5	13	428	65.081724675	442.433333333	18.4347222222	15.4851666667	0.675	
	0.576388888889	1140.0	91.70625	2899.7260274	62.8306878307	4737.17866621		
	8.31083976529							
5	14	342	65.1423428333	453.8	18.9083333333	15.883	0.681944444444	
	0.579166666667	1140.0	91.7916666667	2899.7260274	58.0555555556	4744.7513147		
	8.3241251135							
5	15	404	65.2842780333	416.766666667	17.3652777778	14.5868333333		
	0.641666666667	0.547222222222	1140.0	91.9916666667	2899.7260274	67.5857142857		
	4714.71185665	8.27142430991						
5	16	364	65.1275579167	422.1	17.5875	14.7735	0.684722222222	0.577777777778
	1140.0	91.7708333333	2899.7260274	70.8428571429	4723.38584339	8.28664183051		
5	17	316	65.0965095917	456.033333333	19.0013888889	15.9611666667		
	0.704166666667	0.594444444444	1140.0	91.7270833333	2899.7260274	61.3137755102		
	4750.32724023	8.333907439						
5	18	340	65.0388484167	414.9	17.2875	14.5215	0.683333333333	0.529166666667
	1140.0	91.6458333333	2899.7260274	70.3	4714.81441515	8.2716042371		
5	19	350	65.1837406	411.766666667	17.1569444444	14.4118333333		
	0.680555555556	0.569444444444	1140.0	91.85	2899.7260274	57.1130952381		
	4698.64587268	8.24323837312						
5	20	434	64.9235260667	415.9	17.3291666667	14.5565	0.683333333333	
	0.638888888889	1140.0	91.4833333333	2899.7260274	61.0714285714	4706.54478486		
	8.25709611379							
6	1	342	65.268014625	414.266666667	17.2611111111	14.4993333333		
	0.619444444444	0.519444444444	1140.0	91.96875	2899.7260274	71.6571428571		
	4715.96921268	8.27363019768						
6	2	380	65.1201654583	432.0	18.0	15.12	0.701388888889	0.601388888889
	1140.0	91.7604166667	2899.7260274	63.4203296703	4726.65335897	8.29237431398		
6	3	358	65.2118319417	455.233333333	18.9680555556	15.9331666667		
	0.629166666667	0.591666666667	1140.0	91.8895833333	2899.7260274	60.3174603175		
	4748.69214408	8.33103884926						
6	4	356	65.3641165833	400.166666667	16.6736111111	14.0058333333		
	0.613888888889	0.575	1140.0	92.1041666667	2899.7260274	55.1339285714		
	4684.55401962	8.21851582389						
6	5	404	65.29906295	379.566666667	15.8152777778	13.2848333333	0.675	
	0.626388888889	1140.0	92.0125	2899.7260274	74.6428571429	4681.86511776		
	8.2137984522							
6	6	326	65.15712775	383.8	15.9916666667	13.433	0.620833333333	
	0.577777777778	1140.0	91.8125	2899.7260274	79.6583850932	4690.95202142		
	8.22974038845							
6	7	364	64.9678808167	456.6	19.025	15.981	0.698611111111	0.543055555556
	1140.0	91.5458333333	2899.7260274	57.6785714286	4746.96104724	8.32800183727		

6	8	384	65.0225850083	425.766666667	17.7402777778	14.9018333333	
	0.677777777778		0.536111111111	1140.0	91.6229166667	2899.7260274	68.1285714286
	4724.32855277		8.28829570661				
6	9	414	65.126079425	483.4	20.1416666667	16.919	0.686111111111
	1140.0	91.76875	2899.7260274	66.6871921182	4785.38918932	8.39541963038	
6	10	352	65.112773 385.833333333	16.0763888889	13.5041666667	0.661111111111	
	0.606944444444		1140.0	91.75	2899.7260274	72.5776397516	4686.03702139
	8.22111758139						
6	11	392	65.2783640667	405.633333333	16.9013888889	14.1971666667	
	0.690277777778		0.566666666667	1140.0	91.9833333333	2899.7260274	58.9
	4694.08663093		8.23523970339				
6	12	296	65.0713752333	454.766666667	18.9486111111	15.9168333333	
	0.669444444444		0.559722222222	1140.0	91.6916666667	2899.7260274	52.0238095238
	4739.532783		8.31496979473				
6	13	398	64.993015175	414.333333333	17.2638888889	14.5016666667	
	0.694444444444		0.5375	1140.0	91.58125	2899.7260274	73.0142857143
	8.27519070144						4716.85869982
6	14	338	65.2783640667	436.466666667	18.1861111111	15.2763333333	
	0.681944444444		0.579166666667	1140.0	91.9833333333	2899.7260274	66.0302197802
	4734.389301		8.3059461421				
6	15	400	65.26357915	381.466666667	15.8944444444	13.3513333333	
	0.623611111111		0.522222222222	1140.0	91.9625	2899.7260274	74.9378881988
	4683.96263252		8.21747830267				
6	16	364	65.2916704917	383.766666667	15.9902777778	13.4318333333	0.675
	0.630555555556		0.00208333333	2899.7260274	74.347826087	4686.05700824	
	8.22115264604						
6	17	322	65.0876386417	441.933333333	18.4138888889	15.4676666667	0.725
	0.633333333333		0.917145833333	2899.7260274	67.0741758242	4740.94820722	
	8.31745299512						
6	18	344	65.2162674167	376.633333333	15.6930555556	13.1821666667	
	0.718055555556		0.516666666667	1140.0	91.8958333333	2899.7260274	61.6614906832
	4665.42724621		8.18496008107				
6	19	346	65.1068590333	442.766666667	18.4486111111	15.4968333333	
	0.686111111111		0.593055555556	1140.0	91.7416666667	2899.7260274	63.082010582
	4737.83326286		8.31198818045				
6	20	442	64.9161336083	374.6	15.6083333333	13.111	0.686111111111
	0.652777777778		0.914729166667	2899.7260274	77.1103896104	4678.1205573	
	8.2072290479						
7	1	330	65.0536333333	497.9	20.7458333333	17.4265	0.606944444444
	0.534722222222		0.916666666667	2899.7260274	134.357142857	4868.19431725	
	8.54069178466						
7	2	380	65.2354878083	506.033333333	21.0847222222	17.7111666667	
	0.681944444444		0.509722222222	1140.0	91.9229166667	2899.7260274	137.75
	4880.85896276		8.56291046098				
7	3	368	65.2547082	503.333333333	20.9722222222	17.6166666667	0.6
	0.566666666667		0.9195	2899.7260274	119.953917051	4860.17075274	
	8.52661535568						
7	4	370	65.1674771917	487.466666667	20.3111111111	17.0613333333	
	0.602777777778		0.568055555556	1140.0	91.8270833333	2899.7260274	142.5
	4865.42881537		8.53584002696				
7	5	410	65.1467783083	433.266666667	18.0527777778	15.1643333333	
	0.644444444444		0.544444444444	1140.0	91.7979166667	2899.7260274	129.18956044
	4793.75266848		8.41009240084				
7	6	314	65.2192244	535.933333333	22.3305555556	18.7576666667	
	0.586111111111		0.565277777778	1140.0	91.9	2899.7260274	136.138392857
	4911.3248617		8.61635940649				
7	7	378	65.1246009333	466.0	19.4166666667	16.31	0.680555555556
	0.570833333333		0.917666666667	2899.7260274	121.415816327	4821.21373708	
	8.45826971417						
7	8	372	65.338982225	524.3	21.8458333333	18.3505	0.645833333333
	0.563888888889		0.9206875	2899.7260274	124.988479263	4888.02764924	
	8.57548710393						
7	9	432	65.3404607167	483.6	20.15	16.926	0.697222222222
	1140.0	92.0708333333	2899.7260274	123.54679803	4842.87773939	8.49627673577	0.588888888889
7	10	352	65.06841825	487.7	20.3208333333	17.0695	0.652777777778
	0.581944444444		0.916875	2899.7260274	132.321428571	4855.31706657	
	8.5181001168						
7	11	384	65.3715090417	471.233333333	19.6347222222	16.4931666667	
	0.669444444444		0.575	1140.0	92.1145833333	2899.7260274	133.533163265
	4839.5567353		8.49045041281				
7	12	298	65.1512137833	543.1	22.6291666667	19.0085	0.659722222222
	0.531944444444		0.918041666667	2899.7260274	141.651785714	4924.42222509	
	8.63933723701						
7	13	406	65.1822621083	487.866666667	20.3277777778	17.0753333333	
	0.683333333333		0.561111111111	1140.0	91.8479166667	2899.7260274	123.273809524
	4846.76181332		8.50309090056				
7	14	350	65.1911330583	592.9	24.7041666667	20.7515	0.647222222222
	0.559722222222		0.918604166667	2899.7260274	133.0	4969.52775323	
	8.71846974251						
7	15	398	64.8407305333	490.766666667	20.4486111111	17.1768333333	
	0.602777777778		0.520833333333	1140.0	91.3666666667	2899.7260274	129.380952381
	4855.0433874		8.5176199779				
7	16	360	65.21035345	479.233333333	19.9680555556	16.7731666667	
	0.641666666667		0.5875	1140.0	91.8875	2899.7260274	127.290640394
	8.49387924116						4841.51116746

7	17	308	65.112773	491.4	20.475	17.199	0.681944444444	0.591666666667	1140.0
	91.75	2899.7260274		145.073891626		4872.17536033		8.54767607076	
7	18	344	65.3448961917	492.2		20.5083333333	17.227	0.675	0.476388888889
	1140.0	92.0770833333	2899.7260274			142.273809524	4870.69288827		8.54507524258
7	19	350	64.975273275		419.366666667	17.4736111111		14.6778333333	
	0.666666666667	0.590277777778		1140.0	91.55625	2899.7260274		143.585714286	
	4792.80588551		8.40843137809						
7	20	430	65.4025573667		510.233333333	21.2597222222		17.8581666667	
	0.668055555556	0.6	1140.0		92.1583333333	2899.7260274		127.119047619	
	4875.25568049		8.55308014122						
8	1	328	65.0580688083		443.366666667	18.4736111111		15.5178333333	
	0.584722222222	0.536111111111		1140.0	91.6729166667	2899.7260274		136.71957672	
	4811.83130924		8.44180931445						
8	2	378	65.0536333333		495.733333333	20.6555555556		17.3506666667	
	0.693055555556	0.523611111111		1140.0	91.6666666667	2899.7260274		126.666666667	
	4858.27178649		8.52328383594						
8	3	360	65.0654612667		514.366666667	21.4319444444		18.0028333333	0.625
	0.525	1140.0	91.6833333333		2899.7260274	137.684331797		4889.30352224	
	8.57772547761								
8	4	376	64.8436875167		470.5	19.6041666667	16.4675	0.623611111111	
	0.552777777778	1140.0	91.3708333333		2899.7260274		131.352040816	4835.24214302	
	8.48288095267								
8	5	382	65.117208475		495.433333333	20.6430555556		17.3401666667	
	0.652777777778	0.561111111111		1140.0	91.75625	2899.7260274		147.647783251	
	4879.08242737		8.55979373222						
8	6	312	65.2547082		628.266666667	26.1777777778		21.9893333333	
	0.591666666667	0.5375	1140.0		91.95	2899.7260274	110.357142857	4985.0180237	
	8.74564565561								
8	7	378	65.3729875333		545.9	22.7458333333	19.1065	0.644444444444	
	0.505555555556	1140.0	92.1166666667		2899.7260274		137.359307359	4923.67989249	
	8.63803489911								
8	8	370	65.2931489833		450.433333333	18.7680555556		15.7651666667	
	0.668055555556	0.5625	1140.0		92.0041666667	2899.7260274		123.399470899	
	4806.81820806		8.4330144001						
8	9	408	64.993015175		560.4	23.35	19.614	0.698611111111	0.652777777778
	1140.0	91.58125	2899.7260274		114.558823529	4915.79315219		8.62419851262	
8	10	364	65.3996003833		517.766666667	21.5736111111		18.1218333333	0.6625
	0.488888888889	1140.0	92.1541666667		2899.7260274		140.529953917	4896.61831596	
	8.59055844906								
8	11	372	65.0270204833		630.6	26.275	22.071	0.694444444444	0.561111111111
	1140.0	91.6291666667	2899.7260274		129.478764479	5006.26188938		8.78291559541	
8	12	310	65.1955685333		495.766666667	20.6569444444		17.3518333333	0.6625
	0.544444444444	1140.0	91.8666666667		2899.7260274		128.928571429	4860.86535191	
	8.52783395073								
8	13	396	65.3153263583		460.733333333	19.1972222222		16.1256666667	
	0.659722222222	0.577777777778		1140.0	92.0354166667	2899.7260274		125.535714286	
	4820.11842333		8.45634811111						
8	14	350	65.2916704917		513.733333333	21.4055555556		17.9806666667	
	0.622222222222	0.554166666667		1140.0	92.0020833333	2899.7260274		126.739631336	
	4878.242922		8.55832091579						
8	15	390	64.930918525		441.266666667	18.3861111111		15.4443333333	
	0.597222222222	0.554166666667		1140.0	91.49375	2899.7260274		127.884615385	
	4800.49281231		8.42191721457						
8	16	358	65.2088749583		504.7	21.0291666667	17.6645	0.65	0.586111111111
	1140.0	91.8854166667	2899.7260274		150.19047619	4891.83242519		8.58216214946	
8	17	306	65.40551435		488.433333333	20.3513888889		17.0951666667	
	0.666666666667	0.575	1140.0		92.1625	2899.7260274	126.666666667	4851.24624937	
	8.51095833223								
8	18	334	65.1482568		590.633333333	24.6097222222		20.6721666667	
	0.655555555556	0.513888888889		1140.0	91.8	2899.7260274		139.397959184	
	4973.33590065		8.72515070289						
8	19	344	65.1408643417		472.033333333	19.6680555556		16.5211666667	
	0.668055555556	0.555555555556		1140.0	91.7895833333	2899.7260274		122.610837438	
	4828.89782878		8.4717505768						
8	20	438	65.5001378167		515.1	21.4625	18.0285	0.655555555556	0.598611111111
	1140.0	92.2958333333	2899.7260274		148.380952381	4901.98284179		8.59996989789	
9	1	326	65.2606221667		394.166666667	16.4236111111		13.7958333333	
	0.597222222222	0.573611111111		1140.0	91.9583333333	2899.7260274		116.7708333333	
	4739.44746408		8.31482011241						
9	2	352	65.161563225		373.033333333	15.5430555556		13.0561666667	
	0.679166666667	0.576388888889		1140.0	91.81875	2899.7260274		123.993506494	
	4723.77659503		8.2873273597						
9	3	350	65.01519255		352.466666667	14.6861111111		12.3363333333	
	0.629166666667	0.581944444444		1140.0	91.6125	2899.7260274		104.253246753	
	4681.49475392		8.21314869109						
9	4	350	64.9767517667		392.333333333	16.3472222222		13.7316666667	
	0.606944444444	0.548611111111		1140.0	91.5583333333	2899.7260274		122.437888199	
	4742.45434347		8.32009533943						
9	5	382	65.3685520583		354.1	14.7541666667	12.3935	0.645833333333	
	0.626388888889	1140.0	92.1104166667		2899.7260274		127.959183673	4707.88878248	
	8.25945400436								
9	6	302	65.3330682583		405.2	16.8833333333	14.182	0.598611111111	
	0.538888888889	1140.0	92.0604166667		2899.7260274		113.185714286	4747.86990174	
	8.32959631884								
9	7	362	65.090595625		393.266666667	16.3861111111		13.7643333333	

	0.676388888889	0.497222222222	1140.0	91.71875	2899.7260274	118.602484472	
	4739.92257552	8.31565364126					
9	8 352	65.0270204833	363.066666667	15.1277777778	12.7073333333		
	0.630555555556	0.566666666667	1140.0	91.6291666667	2899.7260274	120.292207792	
	4708.96205914	8.26133694586					
9	9 400	65.1201654583	461.266666667	19.2194444444	16.1443333333		
	0.693055555556	0.55 1140.0	91.7604166667	2899.7260274	95.9693877551		
	4790.66385728	8.40467343382					
9	10 340	65.3360252417	380.633333333	15.8597222222	13.3221666667	0.65	
	0.545833333333	1140.0 92.0645833333	2899.7260274	124.50310559	4732.82300312		
	8.30319825108						
9	11 374	65.2310523333	332.033333333	13.8347222222	11.6211666667	0.6875	
	0.598611111111	1140.0 91.9166666667	2899.7260274	133.0	4688.84950633		
	8.22605176549						
9	12 294	64.984144225	397.6	16.5666666667	13.916	0.675	0.563888888889
	1140.0 91.56875	2899.7260274	112.406832298	4738.16486408	8.31256993698		
9	13 398	65.0669397583	412.433333333	17.1847222222	14.4351666667		
	0.655555555556	0.498611111111	1140.0	91.6854166667	2899.7260274	107.214285714	
	4749.11334663	8.3317778011					
9	14 346	65.1186869667	423.1	17.6291666667	14.8085	0.643055555556	
	0.591666666667	1140.0 91.7583333333	2899.7260274	113.791208791	4767.35206678		
	8.36377555575						
9	15 394	65.268014625	308.766666667	12.8652777778	10.8068333333		
	0.609722222222	0.527777777778	1140.0	91.96875	2899.7260274	121.388888889	
	4652.13910329	8.16164754963					
9	16 350	65.0211065167	379.466666667	15.8111111111	13.2813333333		
	0.686111111111	0.6 1140.0	91.6208333333	2899.7260274	118.897515528		
	4725.29827	8.28999696491					
9	17 296	65.13938585	403.333333333	16.8055555556	14.1166666667	0.6625	
	0.606944444444	1140.0 91.7875	2899.7260274	107.44047619	4739.77701584		
	8.3153982734						
9	18 338	65.2872350167	383.0	15.9583333333	13.405	0.665277777778	
	0.563888888889	1140.0 91.9958333333	2899.7260274	121.847826087	4732.63055603		
	8.30286062462						
9	19 340	65.179305125	363.533333333	15.1472222222	12.7236666667		
	0.677777777778	0.583333333333	1140.0	91.84375	2899.7260274	124.301948052	
	4713.89856991	8.26999749107					
9	20 426	65.3626380917	382.2	15.925	13.377	0.659722222222	0.651388888889
	1140.0 92.1020833333	2899.7260274	115.652173913	4725.88432725	8.29102513552		
10	1 322	65.12164395	379.666666667	15.8194444444	13.2883333333		
	0.581944444444	0.547222222222	1140.0	91.7625	2899.7260274	117.422360248	
	4724.10870251	8.2879100044					
10	2 370	65.1201654583	343.9	14.3291666667	12.0365	0.675	0.583333333333
	1140.0 91.7604166667	2899.7260274	124.178571429	4692.50746395	8.232469235		
10	3 348	65.063982775	322.966666667	13.4569444444	11.3038333333		
	0.631944444444	0.544444444444	1140.0	91.68125	2899.7260274	116.375	4661.93658671
	8.17883611703						
10	4 354	65.1689556833	421.5	17.5625	14.7525	0.611111111111	0.540277777778
	1140.0 91.8291666667	2899.7260274	108.842857143	4760.72310438	8.35214579716		
10	5 386	65.15712775	322.633333333	13.4430555556	11.2921666667		
	0.666666666667	0.636111111111	1140.0	91.8125	2899.7260274	127.142857143	
	4672.71670302	8.1977486018					
10	6 306	65.4202992667	370.7	15.4458333333	12.9745	0.584722222222	0.5375
	1140.0 92.1833333333	2899.7260274	135.097402597	4732.83360355	8.30321684833		
10	7 362	65.1866975833	374.366666667	15.5986111111	13.1028333333		
	0.663888888889	0.536111111111	1140.0	91.8541666667	2899.7260274	118.897515528	
	4720.12651409	8.28092370892					
10	8 356	64.9649238333	321.966666667	13.4152777778	11.2688333333		
	0.663888888889	0.568055555556	1140.0	91.5416666667	2899.7260274	136.785714286	
	4681.09183481	8.21244181545					
10	9 386	65.0580688083	403.433333333	16.8097222222	14.1201666667		
	0.686111111111	0.576388888889	1140.0	91.6729166667	2899.7260274	109.985119048	
	4742.27471154	8.31978019569					
10	10 338	65.1940900417	364.166666667	15.1736111111	12.7458333333		
	0.651388888889	0.536111111111	1140.0	91.8645833333	2899.7260274	123.376623377	
	4713.61606946	8.26950187625					
10	11 378	65.0891171333	341.933333333	14.2472222222	11.9676666667		
	0.672222222222	0.579166666667	1140.0	91.7166666667	2899.7260274	130.867346939	
	4697.00133945	8.2403532271					
10	12 306	65.1955685333	376.3	15.6791666667	13.1705	0.659722222222	
	0.552777777778	1140.0 91.8666666667	2899.7260274	114.74025974	4718.0546744		
	8.27728890246						
10	13 396	65.2118319417	442.4	18.4333333333	15.484	0.65	0.543055555556
	1140.0 91.8895833333	2899.7260274	99.7751322751	4774.32518024	8.37600908813		
10	14 338	65.1098160167	385.066666667	16.0444444444	13.4773333333		
	0.666666666667	0.606944444444	1140.0	91.7458333333	2899.7260274	118.307453416	
	4730.93231992	8.29988126302					
10	15 386	65.2532297083	452.633333333	18.8597222222	15.8421666667		
	0.618055555556	0.547222222222	1140.0	91.9479166667	2899.7260274	108.068783069	
	4793.70331424	8.41000581446					
10	16 348	65.13938585	360.833333333	15.0347222222	12.6291666667		
	0.641666666667	0.479166666667	1140.0	91.7875	2899.7260274	122.142857143	
	4708.60031915	8.26070231429					
10	17 302	65.232530825	422.366666667	17.5986111111	14.7828333333		
	0.686111111111	0.606944444444	1140.0	91.91875	2899.7260274	110.2	4763.28031669

10	8.35663213454 18 328 0.5125 1140.0 8.3318873611	65.1497352917 91.8020833333	404.233333333 2899.7260274	16.8430555556 115.922619048	14.1481666667 4749.17579583	0.6625
10	19 340 0.67777777778 4671.14585137	64.92648305 0.644444444444 8.1949927217	324.133333333 1140.0 91.4875 2899.7260274	13.5055555556 2899.7260274	11.3446666667 124.517857143	
10	20 426 0.59722222222 8.30065089112	65.1231224417 1140.0 91.7645833333	380.766666667 2899.7260274	15.8652777778 123.322981366	13.3268333333 4731.37100794	0.65
11	1 346 0.58888888889 4708.15786725	65.4513475917 0.523611111111 8.2599260829	360.333333333 1140.0 92.2270833333	15.0138888889 2899.7260274	12.6116666667 121.496598639	
11	2 380 0.669444444444 4692.98063695	65.1423428333 0.601388888889 8.23329936307	352.766666667 1140.0 91.7916666667	14.6986111111 2899.7260274	12.3468333333 115.034013605	
11	3 370 0.644444444444 4708.78677846	65.2029609917 0.555555555556 8.2610294359	371.466666667 1140.0 91.8770833333	15.4777777778 2899.7260274	13.0013333333 110.636645963	
11	4 372 0.60138888889 4692.08750901	65.1157299833 0.55 1140.0 8.23173247194	343.933333333 91.7541666667	14.3305555556 2899.7260274	12.0376666667 123.839285714	
11	5 400 0.65416666667 4697.72628779	65.1807836167 0.6625 1140.0 8.24162506631	350.733333333 91.8458333333	14.6138888889 2899.7260274	12.2756666667 121.819727891	
11	6 324 0.6 1140.0 4708.15786725	65.276885575 91.98125 2899.7260274 65.0876386417	415.933333333 107.757142857 361.4	17.3305555556 4753.94899298 15.0583333333	14.5576666667 8.3402613912 12.649	0.6125
11	7 382 1140.0 91.7145833333	8.2599260829 2899.7260274	119.983766234	4707.02545163	0.663888888889 8.25793938882	0.5375
11	8 380 0.64027777778 4702.67425956	64.9501389167 0.586111111111 8.25030571853	342.133333333 1140.0 91.5208333333	14.2555555556 2899.7260274	11.9746666667 136.683673469	
11	9 420 0.58472222222 8.23295582101	65.1231224417 1140.0 91.7645833333	361.9 2899.7260274	15.0791666667 105.017006803	12.6665 0.698611111111 4692.78481797	
11	10 354 0.64722222222 4743.47708796	65.1201654583 0.577777777778 8.32188962801	395.533333333 1140.0 91.7604166667	16.4805555556 2899.7260274	13.8436666667 119.598214286	
11	11 394 0.52083333333 8.26880602023	65.2088749583 1140.0 91.8854166667	374.2 2899.7260274	15.5916666667 112.111801242	13.097 0.666666666667 4713.21943153	
11	12 308 0.55833333333 8.36547621505	65.223659875 1140.0 91.90625	433.966666667 2899.7260274	18.0819444444 102.82967033	15.1888333333 4768.32144258	0.675
11	13 422 0.66388888889 4782.96833434	65.2694931167 0.55 1140.0 8.39117251638	450.966666667 91.9708333333	18.7902777778 2899.7260274	15.7838333333 99.0211640212	
11	14 354 1140.0 91.7416666667	65.1068590333 2899.7260274	338.6 132.660714286	14.1083333333 4695.23153154	11.851 0.6625 8.23724830094	0.584722222222
11	15 408 0.61666666667 4660.22782822	65.1482568 0.50972222222 8.17583829513	312.533333333 1140.0 91.8	13.0222222222 2899.7260274	10.9386666667 125.714285714	
11	16 374 0.644444444444 4699.0714866	64.9368324917 0.5375 1140.0 8.24398506421	352.666666667 91.5020833333	14.6944444444 2899.7260274	12.3433333333 121.819727891	
11	17 332 1140.0 91.8416666667	65.1778266333 2899.7260274	364.8 112.272727273	15.2 12.768 4703.19472233	0.672222222222 8.2512188111	0.558333333333
11	18 352 0.673611111111 4702.62711469	65.214788925 0.498611111111 8.25022300822	354.166666667 1140.0 91.89375	14.7569444444 2899.7260274	12.3958333333 123.112244898	
11	19 344 0.573611111111 8.23993417502	65.2798425583 1140.0 91.9854166667	353.3 2899.7260274	14.7208333333 117.942176871	12.3655 0.684722222222 4696.76247976	
11	20 452 0.655555555556 8.37692801076	65.1364288667 1140.0 91.7833333333	440.133333333 2899.7260274	18.3388888889 102.791005291	15.4046666667 4774.84896613	0.6375
12	1 340 1140.0 91.75625	65.117208475 2899.7260274	429.9 106.671428571	17.9125 15.0465 4767.442676	0.611111111111 8.3639345193	0.519444444444
12	2 378 0.673611111111 8.28438660593	65.03293445 0.616666666667	393.133333333 1140.0 91.6375	16.3805555556 2899.7260274	13.7596666667 100.9375	4722.10036538
12	3 378 0.63333333333 4747.90644767	65.3611596 0.540277777778 8.32966043451	404.133333333 1140.0 92.1	16.8388888889 2899.7260274	14.1446666667 114.226190476	
12	4 380 0.60833333333 4691.67436251	65.1645202083 0.625 1140.0 8.23100765352	338.333333333 91.8229166667	14.0972222222 2899.7260274	11.8416666667 129.25170068	
12	5 402 0.631944444444 4707.33334451	65.285756525 0.611111111111 8.25847955177	372.366666667 1140.0 91.99375	15.5152777778 2899.7260274	13.0328333333 107.954545455	
12	6 322 0.59027777778 4737.26138238	64.966402325 0.570833333333 8.31098488137	399.633333333 1140.0 91.54375	16.6513888889 2899.7260274	13.9871666667 109.419642857	
12	7 390 1140.0 91.7020833333	65.0787676917 2899.7260274	330.6 125.196428571	13.775 11.571 4679.01108577	0.647222222222 8.20879137855	0.505555555556

12	8	380	65.2399232833	373.566666667	15.5652777778	13.0748333333	
	0.640277777778		0.551388888889	1140.0	91.9291666667	2899.7260274	114.431818182
	4714.92902197		8.27180530171				
12	9	420	65.3419392083	373.366666667	15.5569444444	13.0678333333	
	0.688888888889		0.591666666667	1140.0	92.0729166667	2899.7260274	114.431818182
	4715.06977945		8.27205224466				
12	10	358	65.2295738417	333.2	13.8833333333	11.662	0.645833333333
	0.511111111111		1140.0	91.9145833333	2899.7260274	132.660714286	4689.62502884
	8.22741233129						
12	11	394	65.1379073583	352.033333333	14.6680555556	12.3211666667	
	0.679166666667		0.629166666667	1140.0	91.7854166667	2899.7260274	117.207792208
	4694.39917712		8.23578803003				
12	12	330	65.1467783083	375.733333333	15.6555555556	13.1506666667	
	0.656944444444		0.543055555556	1140.0	91.7979166667	2899.7260274	123.993506494
	4726.58063142		8.29224672179				
12	13	400	65.1098160167	374.033333333	15.5847222222	13.0911666667	
	0.645833333333		0.523611111111	1140.0	91.7458333333	2899.7260274	117.422360248
	4718.09706366		8.27736326958				
12	14	366	65.0255419917	331.466666667	13.8111111111	11.6013333333	
	0.645833333333		0.581944444444	1140.0	91.6270833333	2899.7260274	130.357142857
	4685.03882387		8.21936635766				
12	15	402	65.0625042833	334.5	13.9375	11.7075	0.611111111111
	1140.0	91.6791666667	2899.7260274	108.894557823	4666.85185464	8.1874593941	0.518055555556
12	16	364	65.2354878083	374.1	15.5875	13.0935	0.675
	91.9229166667		2899.7260274	112.88961039	4713.9584432	8.27010253192	0.533333333333
12	17	322	65.1186869667	352.333333333	14.6805555556	12.3316666667	1140.0
	0.693055555556		0.572222222222	1140.0	91.7583333333	2899.7260274	8.27010253192
	4705.66875376		8.25555921712				
12	18	344	65.3626380917	370.4	15.4333333333	12.964	0.645833333333
	0.515277777778		1140.0	92.1020833333	2899.7260274	120.909090909	4718.24263378
	8.27761865575						
12	19	356	65.4661325083	339.566666667	14.1486111111	11.8848333333	
	0.670833333333		0.5625	1140.0	92.2479166667	2899.7260274	116.972789116
	4681.43709053		8.21304752725				
12	20	446	65.12164395	335.3	13.9708333333	11.7355	0.670833333333
	0.606944444444		1140.0	91.7625	2899.7260274	124.517857143	4683.651151
	8.21693184386						
13	1	330	65.2547082	455.6	18.9833333333	15.946	0.623611111111
	0.590277777778		1140.0	91.95	2899.7260274	151.044973545	4839.89577836
	8.4910452252						
13	2	380	65.2428802667	548.466666667	22.8527777778	19.1963333333	
	0.716666666667		0.604166666667	1140.0	91.9333333333	2899.7260274	123.170995671
	4912.11348978		8.61774296453				
13	3	368	64.9811872417	501.566666667	20.8986111111	17.5548333333	
	0.651388888889		0.531944444444	1140.0	91.5645833333	2899.7260274	130.05952381
	4867.73197743		8.53988066215				
13	4	362	65.4424766417	520.4	21.6833333333	18.214	0.65
	1140.0	92.2145833333	2899.7260274	138.341013825	4897.47515255	8.59206167115	0.609722222222
13	5	392	65.0846816583	491.0	20.4583333333	17.185	0.676388888889
	0.620833333333		1140.0	91.7104166667	2899.7260274	141.369047619	4868.0408017
	8.54042245912						
13	6	330	65.010757075	498.933333333	20.7888888889	17.4626666667	
	0.627777777778		0.573611111111	1140.0	91.60625	2899.7260274	137.976190476
	4872.88234973		8.54891640303				
13	7	356	65.2354878083	503.5	20.9791666667	17.6225	0.665277777778
	0.554166666667		1140.0	91.9229166667	2899.7260274	140.464285714	4880.8606091
	8.56291334929						
13	8	374	65.2266168583	499.2	20.8	17.472	0.661111111111
	1140.0	91.9104166667	2899.7260274	149.964285714	4885.64005101	8.57129833511	0.479166666667
13	9	418	65.1586062417	542.266666667	22.5944444444	18.9793333333	
	0.723611111111		0.615277777778	1140.0	91.8145833333	2899.7260274	131.807359307
	4913.90991581		8.62089458915				
13	10	358	65.1319933917	571.833333333	23.8263888889	20.0141666667	
	0.666666666667		0.529166666667	1140.0	91.7770833333	2899.7260274	134.317226891
	4948.01390544		8.68072614989				
13	11	386	65.3271542917	453.666666667	18.9027777778	15.8783333333	0.6875
	0.570833333333		1140.0	92.0520833333	2899.7260274	148.029100529	4835.04733406
	8.48253918257						
13	12	314	64.7919403083	429.433333333	17.8930555556	15.0301666667	
	0.679166666667		0.594444444444	1140.0	91.2979166667	2899.7260274	145.10989011
	4804.72421375		8.42934072588				
13	13	408	65.3936864167	588.3	24.5125	20.5905	0.681944444444
	1140.0	92.1458333333	2899.7260274	137.265306122	4969.40527825	8.71825487412	0.570833333333
13	14	346	65.21035345	492.033333333	20.5013888889	17.2211666667	
	0.669444444444		0.6125	1140.0	91.8875	2899.7260274	138.880952381
	8.53847032976						4866.92808796
13	15	404	65.0137140583	558.2	23.2583333333	19.537	0.638888888889
	0.505555555556		1140.0	91.6104166667	2899.7260274	123.170995671	4921.87743517
	8.63487269328						
13	16	354	65.223659875	564.3	23.5125	19.7505	0.6875
	91.90625	2899.7260274	151.281512605	4957.16099181	8.69677366984	0.583333333333	1140.0
13	17	326	65.1319933917	484.2	20.175	16.947	0.691666666667
	1140.0	91.7770833333	2899.7260274	132.204433498	4851.58207435	8.51154749887	0.554166666667
13	18	346	65.42325625	559.266666667	23.3027777778	19.5743333333	
	0.715277777778		0.515277777778	1140.0	92.1875	2899.7260274	146.292016807

13	4947.18855519 19 350 0.69305555556 8.49151470085	8.679278167 65.1837406 0.55 1140.0	467.733333333 91.85 2899.7260274	19.4888888889 138.380102041	16.3706666667 4840.16337948
13	20 442 0.67777777778 4899.7279866	65.0270204833 0.66388888889 8.59601401157	518.833333333 1140.0 91.6291666667	21.6180555556 2899.7260274	18.1591666667 143.156682028
14	1 310 0.58055555556 8.57828114094	65.1659987 1140.0 91.825	510.566666667 2899.7260274	21.2736111111 141.821428571	17.8698333333 4889.62025034
14	2 378 0.67777777778 4919.67416366	65.5356216167 0.59444444444 8.63100730466	551.633333333 1140.0 92.3458333333	22.9847222222 2899.7260274	19.3071666667 126.666666667
14	3 370 0.53055555556 8.4964153117	65.17486965 1140.0 91.8375	485.266666667 2899.7260274	20.2194444444 122.369047619	16.9843333333 4842.95672767
14	4 360 0.63333333333 4845.38992782	65.1334718833 0.54166666667 8.50068408389	472.533333333 1140.0 91.7791666667	19.6888888889 2899.7260274	16.5386666667 138.62244898
14	5 400 0.66527777778 4834.07901105	65.134950375 0.58472222222 8.48084037027	457.866666667 1140.0 91.78125	19.0777777778 2899.7260274	16.0253333333 143.002645503
14	6 320 0.62361111111 4855.01303865	64.94422495 0.56111111111 8.51756673447	482.433333333 1140.0 91.5125	20.1013888889 2899.7260274	16.8851666667 138.054187192
14	7 370 0.68611111111 4836.56387468	65.3079339 0.54166666667 8.48519978015	464.566666667 1140.0 92.025	19.3569444444 2899.7260274	16.2598333333 137.895408163
14	8 372 0.69444444444 4870.13103016	64.9723162917 0.51388888889 8.54408952659	504.433333333 1140.0 91.5520833333	21.0180555556 2899.7260274	17.6551666667 129.366359447
14	9 410 0.52638888889 8.54923837984	64.9368324917 1140.0 91.5020833333	500.466666667 2899.7260274	20.8527777778 136.619047619	17.5163333333 4873.06587651
14	10 358 0.65833333333 4929.68492612	65.1778266333 0.56805555556 8.64857004582	576.366666667 1140.0 91.8416666667	24.0152777778 2899.7260274	20.1728333333 110.966386555
14	11 384 0.69861111111 4930.50288915	65.1024235583 0.56666666667 8.65000506869	554.833333333 1140.0 91.7354166667	23.1180555556 2899.7260274	19.4191666667 135.097402597
14	12 320 0.70277777778 4946.55335463	65.1334718833 0.50694444444 8.67816378005	581.066666667 1140.0 91.7791666667	24.2111111111 2899.7260274	20.3373333333 122.918367347
14	13 402 0.67638888889 5047.33120284	65.06841825 0.58611111111 8.85496702253	662.366666667 1140.0 91.6875	27.5986111111 2899.7260274	23.1828333333 136.223214286
14	14 348 0.58611111111 8.66679472434	65.0846816583 1140.0 91.7104166667	572.3 23.8458333333 2899.7260274	20.0305 0.66805555556 125.93487395	0.66805555556 4940.07299287
14	15 402 1140.0 91.8125	65.15712775 2899.7260274	472.3 19.6791666667 139.224137931	16.5305 0.6375 4845.85322488	0.570833333333 8.50149688575
14	16 362 0.59444444444 8.68049278152	65.285756525 1140.0 91.99375	573.7 23.9041666667 2899.7260274	20.0795 0.68055555556 131.722689076	0.68055555556 4947.88088546
14	17 324 0.72777777778 4765.82427138	65.0299774667 0.54722222222 8.36109521296	403.066666667 1140.0 91.6333333333	16.7944444444 2899.7260274	14.1073333333 134.017857143
14	18 350 1140.0 91.7625	65.12164395 2899.7260274	590.8 24.6166666667 134.583333333	20.678 0.7125 4968.67712524	0.48888888889 8.71697741269
14	19 358 0.69444444444 4824.5238106	65.1556492583 0.53472222222 8.4640768607	463.633333333 1140.0 91.8104166667	19.3180555556 2899.7260274	16.2271666667 127.232142857
14	20 434 1140.0 91.8854166667	65.2088749583 2899.7260274	571.8 23.825 20.013 138.308823529	0.68194444444 4952.30111204	0.61944444444 8.68824756498
15	1 318 0.61944444444 4685.60475236	65.214788925 0.53472222222 8.22035921467	333.933333333 1140.0 91.89375	13.9138888889 2899.7260274	11.6876666667 127.910714286
15	2 378 0.57916666667 8.23319021435	65.2621006583 1140.0 91.9604166667	350.333333333 2899.7260274	14.5972222222 117.295918367	12.2616666667 4692.91842218
15	3 360 0.58888888889 8.19953731628	65.08616015 1140.0 91.7125	331.933333333 2899.7260274	13.8305555556 118.410714286	11.6176666667 4673.73627028
15	4 360 0.66111111111 4700.12286118	65.0698967417 0.6125 1140.0 8.24582958101	351.766666667 91.6895833333	14.6569444444 2899.7260274	12.3118333333 123.43537415
15	5 382 0.69861111111 4754.66569895	65.1600847333 0.56527777778 8.34151877008	410.733333333 1140.0 91.8166666667	17.1138888889 2899.7260274	14.3756666667 114.271428571
15	6 306 1140.0 91.7125	65.08616015 2899.7260274	362.4 15.1 12.684 128.605442177	0.61111111111 4716.71700401	0.62777777778 8.2749421123
15	7 354 0.68888888889 4784.20130596	64.984144225 0.55277777778 8.39333562449	452.666666667 1140.0 91.56875	18.8611111111 2899.7260274	15.8433333333 99.1198979592
15	8 364 65.1319933917		318.733333333	13.2805555556	11.1556666667

	0.677777777778	0.509722222222	1140.0	91.7770833333	2899.7260274	123.571428571	
15	4664.75865585	8.18378711552					
	9 390	65.1201654583	444.8	18.5333333333	15.568	0.716666666667	0.5625
	1140.0 91.7604166667	2899.7260274		99.9587912088	4776.95490173	8.38062263462	
15	10 346	65.2754070833	343.733333333	14.3222222222	12.0306666667		
	0.686111111111	0.619444444444	1140.0	91.9791666667	2899.7260274	124.404761905	
	4692.96256223	8.23326765303					
15	11 372	65.1659987	343.533333333	14.3138888889	12.0236666667		
	0.691666666667	0.594444444444	1140.0	91.825	2899.7260274	125.37414966	
	4693.44753056	8.23411847466					
15	12 304	65.0181495333	374.066666667	15.5861111111	13.0923333333	0.675	
	0.5375 1140.0	91.6166666667	2899.7260274	118.133116883	4718.61448519		
	8.27827102665						
15	13 396	65.1556492583	403.433333333	16.8097222222	14.1201666667		
	0.654166666667	0.529166666667	1140.0	91.8104166667	2899.7260274	116.770833333	
	4749.22169861	8.3319678923					
15	14 338	65.01519255	362.6	15.1083333333	12.691	0.679166666667	0.6
	1140.0 91.6125	2899.7260274	124.61038961	4712.82374376	8.26811183115		
15	15 394	65.0891171333	378.8	15.7833333333	13.258	0.656944444444	
	0.541666666667	1140.0 91.7166666667	2899.7260274	112.701863354	4718.4847636		
	8.2780434449						
15	16 342	65.4202992667	474.1	19.7541666667	16.5935	0.704166666667	
	0.588888888889	1140.0 92.1833333333	2899.7260274	98.6352040816	4807.8888641		
	8.43489274404						
15	17 304	65.3360252417	372.833333333	15.5347222222	13.0491666667		
	0.723611111111	0.533333333333	1140.0	92.0645833333	2899.7260274	108.866459627	
	4708.83017587	8.2611055717					
15	18 332	64.9693593083	425.4	17.725	14.889	0.708333333333	0.525 1140.0
	91.5479166667	2899.7260274	102.328571429	4757.99712693	8.34736338059		
15	19 338	65.1112945083	395.033333333	16.4597222222	13.8261666667		
	0.691666666667	0.619444444444	1140.0	91.7479166667	2899.7260274	120.729166667	
	4744.12587277	8.32302784697					
15	20 424	65.232530825	402.033333333	16.7513888889	14.0711666667		
	0.681944444444	0.556944444444	1140.0	91.91875	2899.7260274	102.068452381	
	4733.26775998	8.30397852628					
16	1 318	65.0137140583	400.2	16.675	14.007	0.629166666667	0.581944444444
	1140.0 91.6104166667	2899.7260274	117.053571429	4745.66725686	8.32573202958		
16	2 362	64.8289026	398.066666667	16.5861111111	13.9323333333		
	0.688888888889	0.5 1140.0	91.35	2899.7260274	106.026785714	4731.89971151	
	8.30157844125						
16	3 362	65.099466575	342.466666667	14.2694444444	11.9863333333	0.6625	
	0.627777777778	1140.0 91.73125	2899.7260274	119.767857143	4686.53131914		
	8.22198477042						
16	4 354	65.19261155	334.433333333	13.9347222222	11.7051666667		
	0.630555555556	0.536111111111	1140.0	91.8625	2899.7260274	126.553571429	
	4684.76430786	8.21888475064					
16	5 380	65.03293445	351.066666667	14.6277777778	12.2873333333		
	0.701388888889	0.638888888889	1140.0	91.6375	2899.7260274	127.636054422	
	4703.55821382	8.25185651548					
16	6 306	65.0580688083	400.766666667	16.6986111111	14.0268333333	0.625	
	0.533333333333	1140.0 91.6729166667	2899.7260274	108.854166667	4738.12560938		
	8.31250106909						
16	7 350	65.0846816583	344.266666667	14.3444444444	12.0493333333		
	0.683333333333	0.570833333333	1140.0	91.7104166667	2899.7260274	120.850340136	
	4689.47364197	8.2271467403					
16	8 354	65.2118319417	303.733333333	12.6555555556	10.6306666667		
	0.651388888889	0.570833333333	1140.0	91.8895833333	2899.7260274	144.007936508	
	4669.26686556	8.19169625536					
16	9 388	65.2221813833	381.766666667	15.9069444444	13.3618333333		
	0.690277777778	0.590277777778	1140.0	91.9041666667	2899.7260274	114.767080745	
	4724.14338539	8.28797085157					
16	10 344	65.1733911583	514.8	21.45	18.018	0.6875	0.559722222222 1140.0
	91.8354166667	2899.7260274	91.0599078341	4843.49431488	8.49735844716		
16	11 372	65.2798425583	342.6	14.275	11.991	0.697222222222	0.563888888889
	1140.0 91.9854166667	2899.7260274	132.482993197	4699.80074573	8.24526446619		
16	12 304	65.0787676917	370.7	15.4458333333	12.9745	0.708333333333	
	0.563888888889	1140.0 91.7020833333	2899.7260274	124.301948052	4721.36429563		
	8.28309525549						
16	13 396	65.1600847333	443.6	18.4833333333	15.526	0.656944444444	
	0.536111111111	1140.0 91.8166666667	2899.7260274	112.592592593	4788.30997668		
	8.40054381873						
16	14 330	65.2532297083	358.133333333	14.9222222222	12.5346666667		
	0.655555555556	0.538888888889	1140.0	91.9479166667	2899.7260274	115.680272109	
	4699.56895955	8.24485782377					
16	15 382	65.2207028917	362.9	15.1208333333	12.7015	0.638888888889	
	0.545833333333	1140.0 91.9020833333	2899.7260274	119.058441558	4708.01902454		
	8.25968249919						
16	16 352	64.9146551167	372.2	15.5083333333	13.027	0.681944444444	
	0.605555555556	1140.0 91.4708333333	2899.7260274	118.441558442	4716.76454442		
	8.27502551653						
16	17 302	65.2532297083	391.1	16.2958333333	13.6885	0.740277777778	
	0.541666666667	1140.0 91.9479166667	2899.7260274	104.047619048	4723.5029124		
	8.28684721473						
16	18 334	65.161563225	414.3	17.2625	14.5005	0.711111111111	0.5125 1140.0
	91.81875 2899.7260274	102.068452381	4746.24039471	8.32673753459			

16	19	344	65.3153263583	353.666666667	14.7361111111	12.3783333333	
	0.661111111111		0.551388888889	1140.0	92.0354166667	2899.7260274	111.964285714
	4691.21901685		8.23020880149				
16	20	428	65.072853725	404.533333333	16.8555555556	14.1586666667	
	0.693055555556		0.544444444444	1140.0	91.69375	2899.7260274	102.633928571
	4736.14098045		8.30901926395				
17	1	336	65.1423428333	371.433333333	15.4763888889	13.0001666667	
	0.601388888889		0.5375	1140.0	91.7916666667	2899.7260274	126.769480519
	4724.65835759		8.28887431157				
17	2	388	65.1556492583	366.433333333	15.2680555556	12.8251666667	
	0.695833333333		0.627777777778	1140.0	91.8104166667	2899.7260274	107.954545455
	4700.70473464		8.24685041166				
17	3	366	65.3463746833	342.366666667	14.2652777778	11.9828333333	
	0.619444444444		0.5625	1140.0	92.0791666667	2899.7260274	114.710884354
	4681.85531472		8.2137812539				
17	4	378	64.9501389167	373.8	15.575	13.083	0.644444444444
	1140.0	91.5208333333	2899.7260274		118.133116883	4718.2198534	8.27757869017
17	5	400	65.3079339	336.0	14.0	11.76	0.7
	92.025	2899.7260274	131.303571429		4691.65355939	8.23097115683	0.616666666667
							1140.0
17	6	328	65.38777245	406.1	16.9208333333	14.2135	0.627777777778
	0.547222222222	1140.0	92.1375	2899.7260274		111.828571429	4747.66497981
	8.32923680668						
17	7	392	65.0772892	404.7	16.8625	14.1645	0.663888888889
	1140.0	91.7	2899.7260274		105.461309524	4739.09753225	8.31420619694
17	8	386	65.1674771917	360.966666667	15.0402777778	12.6338333333	
	0.681944444444		0.506944444444	1140.0	91.8270833333	2899.7260274	123.993506494
	4710.75061848		8.26447476927				
17	9	430	64.9338755083	322.9	13.4541666667	11.3015	0.719444444444
	0.629166666667	1140.0	91.4979166667	2899.7260274		118.571428571	4663.96396292
	8.18239291741						
17	10	368	65.072853725	346.4	14.4333333333	12.124	0.659722222222
	0.573611111111	1140.0	91.69375	2899.7260274		121.173469388	4692.05397838
	8.23167364627						
17	11	400	65.2561866917	332.966666667	13.8736111111	11.6538333333	
	0.698611111111		0.623611111111	1140.0	91.9520833333	2899.7260274	128.589285714
	4685.55427647		8.22027066047				
17	12	320	65.276885575	354.366666667	14.7652777778	12.4028333333	
	0.706944444444		0.534722222222	1140.0	91.98125	2899.7260274	127.959183673
	4707.89127909		8.25945838437				
17	13	416	65.12164395	342.333333333	14.2638888889	11.9816666667	
	0.684722222222	0.575	1140.0	91.7625	2899.7260274	132.482993197	4699.15471006
	8.24413107027						
17	14	346	65.2384447917	362.7	15.1125	12.6945	0.679166666667
	91.9270833333		2899.7260274		115.034013605	4703.89715719	8.25245115297
17	15	394	65.0166710417	372.866666667	15.5361111111	13.0503333333	
	0.636111111111		0.563888888889	1140.0	91.6145833333	2899.7260274	105.031055901
	4704.25259338		8.25307472524				
17	16	362	64.8747358417	393.066666667	16.3777777778	13.7573333333	
	0.681944444444		0.590277777778	1140.0	91.4145833333	2899.7260274	105.916149068
	4726.59949144		8.29227980954				
17	17	320	65.21035345	341.4	14.225	11.949	0.704166666667
	1140.0	91.8875	2899.7260274		117.942176871	4683.80876794	0.593055555556
17	18	358	65.0832031667	413.5	17.2291666667	14.4725	0.679166666667
	1140.0	91.7083333333	2899.7260274		101.242857143	4744.34560657	0.5125
17	19	358	64.9900581917	343.433333333	14.3097222222	12.0201666667	8.32341334487
	0.694444444444		0.577777777778	1140.0	91.5770833333	2899.7260274	120.446428571
	4687.96689414		8.22450332305				
17	20	444	65.1186869667	319.466666667	13.3111111111	11.1813333333	
	0.690277777778		0.634722222222	1140.0	91.7583333333	2899.7260274	131.071428571
	4673.19652698		8.19859039821				
18	1	328	65.3774230083	414.966666667	17.2902777778	14.5238333333	
	0.622222222222		0.538888888889	1140.0	92.1229166667	2899.7260274	107.485714286
	4752.82974545		8.33829779903				
18	2	362	64.8658648917	341.166666667	14.2152777778	11.9408333333	
	0.701388888889		0.595833333333	1140.0	91.4020833333	2899.7260274	119.234693878
	4684.0426653		8.21761871105				
18	3	358	65.3360252417	368.266666667	15.3444444444	12.8893333333	
	0.638888888889		0.590277777778	1140.0	92.0645833333	2899.7260274	113.506493506
	4708.55459279		8.26062209261				
18	4	368	65.1512137833	317.6	13.2333333333	11.116	0.618055555556
	1140.0	91.8041666667	2899.7260274		131.785714286	4671.83172222	0.6
18	5	414	64.8451660083	345.566666667	14.3986111111	12.0948333333	8.1961960039
	0.683333333333		0.655555555556	1140.0	91.3729166667	2899.7260274	118.911564626
	4688.4765373		8.22539743386				
18	6	328	64.9530959	392.1	16.3375	13.7235	0.629166666667
	1140.0	91.525	2899.7260274		119.782608696	4739.49434053	0.541666666667
18	7	380	65.2606221667	363.933333333	15.1638888889	12.7376666667	8.3149023518
	0.686111111111		0.569444444444	1140.0	91.9583333333	2899.7260274	0.516666666667
	4710.22283558		8.26354883434				119.983766234
18	8	384	64.9560528833	375.6	15.65	13.146	0.686111111111
	1140.0	91.5291666667	2899.7260274		119.192546584	4721.23196802	0.540277777778
18	9	422	65.1733911583	402.866666667	16.7861111111	14.1003333333	8.28286310179
	0.713888888889		0.623611111111	1140.0	91.8354166667	2899.7260274	14.1003333333
	4722.8105247		8.28563249948				90.7589285714
18	10	360	64.984144225	372.533333333	15.5222222222	13.0386666667	

	0.673611111111	0.531944444444	1140.0	91.56875	2899.7260274	112.581168831	
	4711.35279223	8.26553121444					
18	11 378	65.223659875	372.833333333		15.5347222222	13.0491666667	
	0.683333333333	0.566666666667	1140.0	91.90625	2899.7260274	118.75	4718.47572969
	8.27802759596						
18	12 320	65.2251383667	353.866666667		14.7444444444	12.3853333333	
	0.704166666667	0.594444444444	1140.0	91.9083333333	2899.7260274	117.516233766	
	4696.84227642	8.24007416916					
18	13 420	65.1586062417	342.133333333		14.2555555556	11.9746666667	
	0.690277777778	0.519444444444	1140.0	91.8145833333	2899.7260274	123.160714286	
	4689.65828704	8.22747067901					
18	14 352	65.2310523333	432.766666667		18.0319444444	15.1468333333	
	0.673611111111	0.547222222222	1140.0	91.9166666667	2899.7260274	107.788461538	
	4772.01712251	8.37195986406					
18	15 400	65.3360252417	352.933333333		14.7055555556	12.3526666667	
	0.622222222222	0.548611111111	1140.0	92.0645833333	2899.7260274	118.588435374	
	4697.09182024	8.24051196533					
18	16 380	65.2369663	403.933333333		16.8305555556	14.1376666667	
	0.695833333333	0.558333333333	1140.0	91.925	2899.7260274	102.871428571	
	4736.11878649	8.30898032718					
18	17 320	65.330111275	362.033333333		15.0847222222	12.6711666667	0.725
	0.508333333333	1140.0 92.05625	2899.7260274		117.942176871	4706.2486091	
	8.25657650719						
18	18 350	65.2561866917	402.033333333		16.7513888889	14.0711666667	
	0.716666666667	0.6 1140.0	91.9520833333		2899.7260274	109.702380952	
	4740.99679893	8.31753824374					
18	19 362	65.1246009333	338.6	14.1083333333	11.851	0.691666666667	
	0.586111111111	1140.0 91.7666666667	2899.7260274		119.880952381	4682.52935429	
	8.21496377945						
18	20 446	65.1334718833	333.266666667		13.8861111111	11.6643333333	
	0.677777777778	0.619444444444	1140.0	91.7791666667	2899.7260274	122.821428571	
	4679.81343925	8.21019901623					

Figura A8.1 - Custo de consolidação do pedido - arranjo físico funcional (Fonte: autor)

APÊNDICE 9 TEMPO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO ALEATORIAMENTE DISTRIBUÍDO (20 AMOSTRAS)

Este apêndice apresenta os resultados obtidos diretamente do modelo de simulação para as 20 amostras do arranjo físico aleatoriamente distribuído, as quais foram submetidas ao modelo de simulação. Os perfis das peças consideradas na simulação são idênticos aos que foram submetidos ao arranjo físico funcional. Enfim, os perfis das peças eram: uma operação a quatro operações, tamanho do lote de cada tipo de peça variando de 10 unidades a 20 unidades e tempo de cada operação variando de 30s a 32s.

Como são dois critérios, então tem-se dois arquivos externos, onde o modelo de simulação realizou a impressão dos resultados. A Figura A9.1 e a Figura A9.2 apresentam os resultados impressos considerando o critério de menor distância e (caso houver empate) adota-se o de maior fila formada na máquina (daí o nome menor distância maior fila), e o critério da maior fila formada na máquina e (caso houver empate) adota-se o de menor distância (daí o nome maior fila menor distância). Para obter os resultados de cada critério, 20 amostras foram submetidas ao modelo de simulação.

1	1	7559
1	2	15267
1	3	23027
1	4	30701
1	5	38337
1	6	46200
1	7	54108
1	8	61987
1	9	69851
1	10	77606
1	11	85465
1	12	93256
1	13	101120
1	14	108842
1	15	116631
1	16	124245
1	17	131975
1	18	139611
1	19	147426
1	20	155060

Figura A9.1 - Tempo de consolidação do pedido – critério menor distância maior fila (Fonte: autor)

1	1	7045
1	2	13402
1	3	19755
1	4	26335
1	5	34219
1	6	42102
1	7	49975
1	8	57851
1	9	66447
1	10	74304
1	11	82162
1	12	90038
1	13	97937
1	14	105817
1	15	113680

1	16	121559
1	17	129420
1	18	137305
1	19	145187
1	20	153060

Figura A9.2 - Tempo de consolidação do pedido - critério maior fila menor distância (Fonte: autor)

Os resultados apresentados também são acumulativos e, portanto, para obter o tempo de consolidação de cada amostra, deve-se realizar a subtração com o resultado acumulado da amostra anterior. Por exemplo, considere o exemplo da terceira linha da Figura A9.1. De acordo com essa linha, está ilustrado: “1 3 23027”. O número “1” significa que o critério o qual as 20 amostras foram submetidas é o primeiro (para o arranjo físico distribuído, apesar de considerar dois critérios, a execução do modelo de simulação é feita separadamente, por isso que em vez de “2” é “1”). O número 3 representa a terceira amostra das 20 geradas. O número “23027” é o tempo acumulado da terceira amostra, que quando subtraído da amostra anterior, obtém-se assim o tempo de consolidação do pedido (em segundos). Com isso, o tempo de consolidação do pedido para a terceira amostra é : $23027 - 15267 = 7760s$.

Baseado nesta mesma forma de interpretação dos resultados, pode-se obter os tempos de consolidação do pedido de cada amostra para os resultados apresentados na Figura A9.2.

Ao somar-se os tempos de consolidação do pedido das 20 amostras dos dois critérios adotados dividido por 40, resulta em apenas um resultado da lacuna da Tabela 5.1 do Tópico 5, que é 7703s.

APÊNDICE 10 CUSTO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO ALEATORIAMENTE DISTRIBUÍDO (20 AMOSTRAS)

Este apêndice apresenta os resultados que foram impressos pelo modelo de simulação ao arquivo externo para as 20 amostras. A Figura A10.1 apresenta, então, os resultados impressos pelo modelo de simulação considerando o critério de menor distância e caso houver empate, escolher a máquina de maior fila para a coleta de caixas (onde estão contidas as peças acabadas). A interpretação desses resultados é similar ao do arranjo físico funcional.

Já na Figura A10.2, estão apresentados resultados do modelo de simulação para as 20 amostras submetidas ao critério de maior fila menor distância para a coleta das caixas (onde estão contidas as peças acabadas). A interpretação desses resultados também é similar ao do arranjo físico funcional, exceto que agora, não existe custo associado ao funcionário tipo 3.

De forma similar, o resultado médio obtido através da soma do tempo de consolidação do pedido de cada amostra dividido por 40 resulta em único resultado da Tabela 5.2 do Tópico 5, que é R\$4469,31.

1	1	0	66.1240613	260.5	0.443055555556	4.16527777778	1140.0	93.175
	2899.7260274		10.0	4474.13342203	7.84935688076			
1	2	0	66.1077978917	564.3	0.469444444444	3.95833333333	1140.0	
	93.1520833333		2899.7260274	10.0	4777.7136864	8.38195383579		
1	3	0	66.1063194	867.233333333	0.5	4.49166666667	1140.0	93.15
	2899.7260274		8.86842105263	5080.07576785	8.91241362781			
1	4	0	66.137367725	1180.7	0.481944444444	3.95277777778	1140.0	93.19375
	2899.7260274		7.86111111111	5392.05297846	9.45974206747			
1	5	0	66.1166688417	1523.6	0.490277777778	4.47361111111	1140.0	
	93.1645833333		2899.7260274	8.16666666667	5735.73783513	10.0626979564		
1	6	0	66.1403247083	1865.76666667	0.518055555556	4.35	1140.0	
	93.1979166667		2899.7260274	7.225	6076.92399099	10.6612701596		
1	7	0	66.1536311333	2208.4	0.506944444444	4.475	1140.0	93.2166666667
	2899.7260274		6.57142857143	6419.04969821	11.2614906986			
1	8	0	66.119625825	2536.4	0.519444444444	4.63333333333	1140.0	93.16875
	2899.7260274		7.625	6748.192181	11.8389336509			
1	9	0	66.13293225	2879.5	0.486111111111	4.4875	1140.0	93.1875
	2899.7260274		7.45	7090.97007076	12.4402983698			
1	10	0	66.1536311333	3222.63333333	0.511111111111	4.36388888889	1140.0	
	93.2166666667		2899.7260274	8.85714285714	7435.46180139	13.044669827		
1	11	0	66.1625020833	3564.8	0.477777777778	4.52638888889	1140.0	
	93.2291666667		2899.7260274	6.4	7775.32186281	13.6409155488		
1	12	0	66.1432816917	3907.73333333	0.490277777778	4.475	1140.0	
	93.2020833333		2899.7260274	8.69047619048	8120.46047972	14.2464218943		
1	13	0	66.128496775	4250.86666667	0.4875	4.52222222222	1140.0	93.18125
	2899.7260274		7.80952380952	8462.72168687	14.8468801524			
1	14	0	66.1595451	4593.43333333	0.523611111111	4.25555555556	1140.0	
	93.225	2899.7260274	7.75	8805.0730725	15.4474966184			
1	15	0	66.09744845	4936.1	0.505555555556	4.16805555556	1140.0	93.1375
	2899.7260274		7.7619047619	9147.49649172	16.0482394592			
1	16	0	66.1063194	5278.66666667	0.509722222222	4.45	1140.0	93.15
	2899.7260274		7.625	9490.23373569	16.6495328696			
1	17	0	66.11519035	5620.5	0.509722222222	4.29027777778	1140.0	93.1625
	2899.7260274		7.14285714286	9831.44657489	17.2481518858			
1	18	0	66.1004054333	5962.76666667	0.475	4.24861111111	1140.0	
	93.1416666667		2899.7260274	7.625	10174.0833773	17.8492690829		
1	19	0	66.1403247083	6305.33333333	0.498611111111	4.33472222222	1140.0	
	93.1979166667		2899.7260274	6.97619047619	10516.2071259	18.4494861858		
1	20	0	66.1521526417	6647.86666667	0.4875	4.21666666667	1140.0	
	93.2145833333		2899.7260274	7.225	10858.8885967	19.0506817486		

Figura A10.1 - Custo de consolidação do pedido - Critério menor distância maior fila (Fonte: autor)

1	1	0	66.1225828083	314.933333333	0.440277777778	3.86944444444	1140.0
		93.1729166667	2899.7260274	7.52631578947	4525.79089822	7.93998403196	
1	2	0	66.1388462167	606.9	0.45	3.82361111111	93.1958333333
		2899.7260274	9.16666666667	4819.40098473	8.45508944689		
1	3	0	66.1595451	898.766666667	0.472222222222	4.11388888889	1140.0
		93.225	2899.7260274	8.82352941176	5111.28687969	8.96716996436	
1	4	0	66.1565881167	1198.2	0.4625	3.6	93.2208333333
		2899.7260274	9.75	5411.11594885	9.49318587517		
1	5	0	66.1358892333	1541.06666667	0.498611111111	4.0375	1140.0
		93.1916666667	2899.7260274	9.66666666667	5754.32302774	10.0953035574	
1	6	0	66.101883925	1883.93333333	0.504166666667	4.10833333333	1140.0
		93.14375	2899.7260274	7.5	6095.01749466	10.6930131485	
1	7	0	66.1418032	2226.46666667	0.497222222222	3.88611111111	1140.0
		93.2	2899.7260274	7.925	6437.8428306	11.2944611063	
1	8	0	66.1240613	2569.1	0.495833333333	3.88333333333	1140.0
		2899.7260274	7.57142857143	6780.07568394	11.8948696209		93.175
1	9	0	66.137367725	2935.7	0.5125	3.95416666667	93.19375
		2899.7260274	6.18181818182	7145.40562997	12.5357993508		
1	10	0	66.1477171667	3277.7	0.515277777778	4.02083333333	1140.0
		93.2083333333	2899.7260274	7.3	7488.61818901	13.1379266474	
1	11	0	66.0723140917	3619.73333333	0.488888888889	3.91388888889	1140.0
		93.1020833333	2899.7260274	8.21428571429	7831.25082165	13.7390365292	
1	12	0	66.1299752667	3962.33333333	0.488888888889	3.96944444444	1140.0
		93.1833333333	2899.7260274	7.675	8173.50600266	14.3394842152	
1	13	0	66.0590076667	4305.73333333	0.4875	3.9	93.0833333333
		2899.7260274	8.47619047619	8517.46539221	14.9429217407		
1	14	0	66.1137118583	4648.5	0.483333333333	4.0	93.1604166667
		2899.7260274	8.65	8860.63348926	15.5449710338		
1	15	0	66.20389985	4990.66666667	0.472222222222	4.10277777778	1140.0
		93.2875	2899.7260274	7.92857142857	9202.38766534	16.1445397638	
1	16	0	66.128496775	5333.4	0.513888888889	4.24722222222	93.18125
		2899.7260274	7.52380952381	9544.72069481	16.745124026		
1	17	0	66.1048409083	5675.53333333	0.509722222222	4.02361111111	1140.0
		93.1479166667	2899.7260274	7.825	9886.87045164	17.3453867573	
1	18	0	66.1166688417	6018.46666667	0.501388888889	3.95138888889	1140.0
		93.1645833333	2899.7260274	8.5	10230.426724	17.9481170597	
1	19	0	66.119625825	6361.26666667	0.513888888889	4.0125	93.16875
		2899.7260274	7.65	10572.4574588	18.5481709803		
1	20	0	66.1211043167	6703.8	0.4875	4.05833333333	93.1708333333
		2899.7260274	8.59523809524	10915.9590365	19.1508053272		

Figura A10.2 - Custo de consolidação do pedido - Critério maior fila menor distância (Fonte: autor)

APÊNDICE 11 TEMPO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO PARCIALMENTE DISTRIBUÍDO (20 AMOSTRAS)

Este apêndice apresenta os resultados, em tempo acumulado, impressos pelo modelo de simulação ao arquivo externo. A Figura A11.1 apresenta, então, os resultados das 20 amostras submetidas ao critério de menor distância maior fila. Já a Figura A11.2 apresenta os resultados da simulação submetendo-se as 20 amostras dos arranjos físicos parcialmente distribuídos ao critério de maior fila menor distância. Da mesma forma, os resultados apresentados também são acumulativos e com isso, pode-se obter o resultado médio dos tempos de consolidação do pedido, o qual resulta em apenas uma lacuna da Tabela 5.1 do Tópico 5, que é 7733s.

1	1	8018
1	2	15827
1	3	23764
1	4	31887
1	5	39708
1	6	47722
1	7	55769
1	8	63679
1	9	71555
1	10	79440
1	11	87538
1	12	95391
1	13	103228
1	14	110784
1	15	118887
1	16	126747
1	17	134556
1	18	142546
1	19	150366
1	20	158211

Figura A11.1 - Tempo de consolidação do pedido - critério menor distância maior fila (Fonte: autor)

1	1	7453
1	2	15137
1	3	22989
1	4	30400
1	5	37733
1	6	45500
1	7	53058
1	8	60744
1	9	67762
1	10	75871
1	11	82487
1	12	89958
1	13	97491
1	14	104051
1	15	111841
1	16	119645
1	17	127718
1	18	135478
1	19	143130
1	20	151109

Figura A11.2 - Tempo de consolidação do pedido - critério maior fila menor distância (Fonte: autor)

APÊNDICE 12 CUSTO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO PARCIALMENTE DISTRIBUÍDO (20 AMOSTRAS)

Este apêndice apresenta os resultados que foram impressos pelo modelo de simulação ao arquivo externo para as 20 amostras. A Figura A12.1 apresenta, então, os resultados impressos pelo modelo de simulação considerando o critério de menor distância e caso houver empate, escolher a máquina de maior fila para a coleta de caixas (onde estão contidas as peças acabadas). A interpretação desses resultados é similar ao do arranjo físico funcional.

Já na Figura A12.2, estão apresentados resultados do modelo de simulação para as 20 amostras submetidas ao critério de maior fila menor distância para a coleta das caixas (onde estão contidas as peças acabadas).

De forma similar, o resultado do cálculo da soma do tempo de consolidação do pedido de cada amostra dividido por 40 resulta em único resultado da Tabela 5.2 do Tópico 5, que é R\$4469,24.

1	1	0	66.1403247083	330.566666667	0.448611111111	4.59305555556	1140.0
	93.1979166667	2899.7260274	7.05	4541.72260211	7.96793438966		
1	2	0	66.1225828083	658.133333333	0.470833333333	4.36111111111	1140.0
	93.1729166667	2899.7260274	8.225	4870.21180465	8.54423123623		
1	3	0	66.1418032	952.033333333	0.459722222222	4.45833333333	1140.0
	93.2	2899.7260274	9.33333333333	5165.35255282	9.06202202249		
1	4	0	66.163980575	1257.166666667	0.470833333333	4.00555555556	1140.0
	93.23125	2899.7260274	7.55555555556	5468.31986908	9.59354362997		
1	5	0	66.119625825	1584.6	0.480555555556	4.68472222222	1140.0
	2899.7260274	5.575	5794.354681	10.1655345281			93.16875
1	6	0	66.163980575	1876.966666667	0.479166666667	4.72222222222	1140.0
	93.23125	2899.7260274	7.61764705882	6088.90696059	10.6822929133		
1	7	0	66.1758085083	2174.33333333	0.477777777778	4.61944444444	1140.0
	93.2479166667	2899.7260274	9.22222222222	6387.80253035	11.2066711059		
1	8	0	66.1137118583	2470.6	0.4875	4.68194444444	1140.0
	2899.7260274	8.05555555556	6682.82515592	11.7242546595			93.1604166667
1	9	0	66.1447601833	2820.6	0.491666666667	4.49166666667	1140.0
	93.2041666667	2899.7260274	7.33333333333	7031.99162091	12.3368274051		
1	10	0	66.1758085083	3128.266666667	0.494444444444	4.56388888889	1140.0
	93.2479166667	2899.7260274	8.83333333333	7341.30808591	12.87948787		
1	11	0	66.1270182833	3446.466666667	0.493055555556	4.73055555556	1140.0
	93.1791666667	2899.7260274	7.71052631579	7658.43301644	13.4358473973		
1	12	0	66.15067415	3736.8	0.495833333333	4.96666666667	1140.0
	2899.7260274	9.27777777778	7950.62947933	13.9484727707			93.2125
1	13	0	66.1211043167	4039.366666667	0.501388888889	4.89861111111	1140.0
	93.1708333333	2899.7260274	8.38888888889	8252.1735206	14.4774974046		
1	14	0	66.11519035	4350.23333333	0.470833333333	4.83055555556	1140.0
	93.1625	2899.7260274	6.21052631579	8560.74896629	15.0188578356		
1	15	0	66.11519035	4622.966666667	0.473611111111	4.84583333333	1140.0
	93.1625	2899.7260274	9.375	8836.66482886	15.5029207524		
1	16	0	66.146238675	4948.43333333	0.498611111111	4.89166666667	1140.0
	93.20625	2899.7260274	7.07894736842	9159.98107455	16.0701422361		
1	17	0	66.101883925	5275.83333333	0.475	4.57083333333	1140.0
	2899.7260274	9.075	9488.92582799	16.6472382947			93.14375
1	18	0	66.128496775	5598.93333333	0.468055555556	4.66527777778	1140.0
	93.18125	2899.7260274	6.94736842105	9810.04980926	17.2106137005		
1	19	0	66.1565881167	5862.5	0.468055555556	4.725	1140.0
	2899.7260274	8.5625	10075.3590044	17.6760684288			93.2208333333
1	20	0	66.1743300167	6174.43333333	0.490277777778	4.80833333333	1140.0
	93.2458333333	2899.7260274	8.21052631579	10387.0886615	18.222962564		

Figura A12.1 - Custo de consolidação do pedido - critério menor distância maior fila (Fonte: autor)

1	1	0	66.155109625	306.033333333	0.493055555556	4.1625	1140.0	93.21875
	2899.7260274		8.44736842105	4518.23614433	7.92673007778			
1	2	0	66.0885775	619.8	0.456944444444	4.16527777778	1140.0	93.125
	2899.7260274		6.76315789474	4830.12498501	8.47390348248			
1	3	0	66.1491956583	939.133333333	0.4875	3.92777777778	1140.0	
	93.2104166667		2899.7260274	8.34210526316	5150.9763561	9.03680062473		
1	4	0	66.1625020833	1243.8	0.470833333333	3.68055555556	1140.0	
	93.2291666667		2899.7260274	8.5	5455.56908504	9.5711738334		
1	5	0	66.1477171667	1545.83333333	0.469444444444	4.34166666667	1140.0	
	93.2083333333		2899.7260274	9.05555555556	5758.7820779	10.1031264525		
1	6	0	66.0915344833	1862.36666667	0.473611111111	4.36805555556	1140.0	
	93.1291666667		2899.7260274	8.68421052632	6074.83927241	10.6576127586		
1	7	0	66.1403247083	2171.9	0.472222222222	4.58611111111	1140.0	
	93.1979166667		2899.7260274	7.81578947368	6383.83839158	11.1997164765		
1	8	0	66.1225828083	2485.73333333	0.484722222222	4.6	1140.0	
	93.1729166667		2899.7260274	7.76315789474	6697.60274032	11.7501802462		
1	9	0	66.1166688417	2777.26666667	0.491666666667	4.34444444444	1140.0	
	93.1645833333		2899.7260274	8.38235294118	6989.49241029	12.2622673865		
1	10	0	66.13293225	3105.2	0.480555555556	4.51944444444	1140.0	93.1875
	2899.7260274		8.9	7318.14645965	12.838853438			
1	11	0	66.1240613	3383.33333333	0.490277777778	4.43333333333	1140.0	
	93.175	2899.7260274	7.94117647059	7595.22320961	13.3249529993			
1	12	0	66.101883925	3690.0	0.473611111111	4.27916666667	1140.0	93.14375
	2899.7260274		8.83333333333	7902.55777243	13.8641364429			
1	13	0	66.07970655	3998.7	0.491666666667	4.63194444444	1140.0	93.1125
	2899.7260274		9.02777777778	8211.76962284	14.4066133734			
1	14	0	66.13293225	4275.0	0.473611111111	4.49027777778	1140.0	93.1875
	2899.7260274		9.70588235294	8488.71623089	14.8924846156			
1	15	0	66.2024213583	4592.26666667	0.493055555556	4.18194444444	1140.0	
	93.2854166667		2899.7260274	8.89473684211	8805.05026893	15.4474566122		
1	16	0	66.1240613	4910.03333333	0.505555555556	4.43472222222	1140.0	
	93.175	2899.7260274	8.55263157895	9122.55133139	16.00447602			
1	17	0	66.1447601833	5236.73333333	0.477777777778	4.57916666667	1140.0	
	93.2041666667		2899.7260274	7.25	9448.11523203	16.5756407579		
1	18	0	66.1358892333	5553.03333333	0.481944444444	4.66388888889	1140.0	
	93.1916666667		2899.7260274	6.55263157895	9763.78538154	17.1294480378		
1	19	0	66.11519035	5865.7	0.501388888889	4.26388888889	1140.0	93.1625
	2899.7260274		10.2777777778	10079.7467733	17.683766269			
1	20	0	66.1033624167	6189.3	0.4875	4.4	1140.0	93.1458333333
	2899.7260274		8.25	10401.4127231	18.2480924967			

Figura A12.2 - Custo de consolidação do pedido - critério maior fila menor distância (Fonte: autor)

APÊNDICE 13 TEMPO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO MAXIMAMENTE DISTRIBUÍDO

Este apêndice apresenta os resultados impressos pelo modelo de simulação ao arquivo externo. A Figura A13.1 apresenta os resultados, em tempo, dos dois critérios adotados para a coleta de caixas (onde estão contidas peças acabadas). Note que os resultados de ambos os critérios são representados por “1 1 7545” e “1 1 7285” para o critério de menor distância maior fila e para o critério de maior fila menor distância, respectivamente. Em ambos os casos, o número “1” representa 1 critério, em seguida o número “1” representa uma amostra e, por fim, o tempo representa o tempo de consolidação do pedido para o arranjo físico maximamente distribuído.

Por tratar-se de apenas uma amostra submetida a cada um dos critérios, então o tempo obtido pelo modelo de simulação pode ser usado diretamente, ou seja, este tempo não é acumulativo. Assim, a média dos tempos desses dois critérios resulta também em apenas uma lacuna da Tabela 5.1 do Tópico 5, que 7415s.

	Resultado		
Menor distância maior fila	1	1	7545
Maior fila menor distância	1	1	7285

Figura A13.1 - Tempo de consolidação do pedido – critério menor distância maior fila e maior fila menor distância (Fonte: autor)

APÊNDICE 14 CUSTO DE CONSOLIDAÇÃO DO PEDIDO – ARRANJO FÍSICO MAXIMAMENTE DISTRIBUÍDO

Por fim, este apêndice apresenta o custo de consolidação do pedido para o arranjo físico maximamente distribuído submetido aos dois critérios para a coleta de caixas (onde estão contidas as peças acabadas), os quais são: critério menor distância maior fila e critério maior fila menor distância.

Os resultados impressos pelo modelo de simulação podem ser vistos na Figura A14.1 e Figura A14.2. A interpretação destes é similar ao dos demais arranjos físicos distribuídos. Com tais resultados, é possível obter a média dos custos de consolidação do pedido, a qual resulta em R\$4460, 87. Este resultado médio preenche apenas uma lacuna da Tabela 5.2 do Tópico 5.

1	1	0	66.048658225	320.0	0.522222222222	4.14305555556	1140.0	93.06875
			2899.7260274	6.35	4391.8587134	7.94712054982		

Figura A14.1 - Custo de consolidação do pedido – critério menor distância maior fila (Fonte: autor)

1	1	0	66.1211043167	329.8	0.4875	4.02777777778	1140.0	93.1708333333
			2899.7260274	5.825	4529.15824283	7.96343551373		

Figura A14.2 - Custo de consolidação do pedido – critério maior fila menor distância (Fonte: autor)

APÊNDICE 15 INFORMAÇÕES CONTIDAS NO ARQUIVO IN_CPN.sml

Este apêndice apresenta o arquivo externo, denominado de IN_CPN.sml, que também deve ser gerado por um código de programação. É neste arquivo que devem ser copiadas as informações geradas pelo código de programação, tais como as amostras de cada tipo de arranjo físico e as distâncias De-Para. Entretanto, amostras de tipos distintos de arranjos físicos não devem ser inseridos no mesmo arquivo. Isso quer dizer que para cada tipo de arranjo físico, existe um arquivo IN_CPN.sml.

Além dessas informações, existem também os tempos de cada operação, os tipos de máquinas, as réplicas de cada tipo de máquina etc. Ao realizar a simulação, dependendo da informação desejada, o modelo de simulação faz então o acesso a estas informações através de funções.

A Figura A15.1 apresenta as informações comuns aos arranjos físicos abordados neste estudo contidas IN_CPN.sml. O que difere uma da outra são as informações a serem inseridas das amostras dos tipos de arranjos físicos.

```

fun CheckOrderStatus(pc,oq,or2,or) =
  if (pc = oq) andalso (or2 = or) then
    1^(or, "OrderCompleted")
  else 1^(or, "OrderNotCompleted");

fun FillOutOrderBuf(orinfo,orderbuf) =
  let
    fun AuxFillOrder(aux1,aux2) =
      if aux2 = nil then aux1::aux2
      else
        AuxFillOrder(aux1,tl(aux2));
  in
    if orderbuf = nil then orinfo::orderbuf
    else
      orderbuf^^AuxFillOrder(orinfo,tl(orderbuf))
  end;

fun CalcMinTranspDistIn(TranspDistListIn) =
  let
    fun aux(f,L1) =
      if List.null(L1) then
        f
      else
        if f > hd(L1) then
          aux(hd(L1),tl(L1))
        else
          aux(f,tl(L1))
  in
    aux(hd(TranspDistListIn),tl(TranspDistListIn))
  end;

fun CalcMinTranspDistInter(TranspDistListInter) =
  let
    fun aux(f,L2) =
      if List.null(L2) then
        f
      else
        if f > hd(L2) then
          aux(hd(L2),tl(L2))
        else
          aux(f,tl(L2))
  in
    aux(hd(TranspDistListInter),tl(TranspDistListInter))
  end;

```

```

fun CalcMinTranspDistInterT(TranspDistListInterT) =
let
  fun aux(f,L3) =
    if List.null(L3) then
      f
    else
      if f > hd(L3) then
        aux(hd(L3),tl(L3))
      else
        aux(f,tl(L3))
in
  aux(hd(TranspDistListInterT),tl(TranspDistListInterT))
end;

fun CalcMinTranspDistExit(TranspDistListExit) =
let
  fun aux(f,L4) =
    if List.null(L4) then
      f
    else
      if f > hd(L4) then
        aux(hd(L4),tl(L4))
      else
        aux(f,tl(L4))
in
  aux(hd(TranspDistListExit),tl(TranspDistListExit))
end;

fun TransfMachToINTIn(Maq) =
if Maq = Mach(1) then 1
else if Maq = Mach(2) then 2
else if Maq = Mach(3) then 3
else if Maq = Mach(4) then 4
else if Maq = Mach(5) then 5
else if Maq = Mach(6) then 6
else if Maq = Mach(7) then 7
else if Maq = Mach(8) then 8
else if Maq = Mach(9) then 9
else if Maq = Mach(10) then 10
else if Maq = Mach(11) then 11
else if Maq = Mach(12) then 12
else if Maq = Mach(13) then 13
else if Maq = Mach(14) then 14
else 15;

fun TransfMachToINTInter(Maq) =
if Maq = Mach(1) then 1
else if Maq = Mach(2) then 2
else if Maq = Mach(3) then 3
else if Maq = Mach(4) then 4
else if Maq = Mach(5) then 5
else if Maq = Mach(6) then 6
else if Maq = Mach(7) then 7
else if Maq = Mach(8) then 8
else if Maq = Mach(9) then 9
else if Maq = Mach(10) then 10
else if Maq = Mach(11) then 11
else if Maq = Mach(12) then 12
else if Maq = Mach(13) then 13
else if Maq = Mach(14) then 14
else 15;

fun DefMachLocal(MachPos) =
if MachPos = 1 then Local(2) else
if MachPos = 2 then Local(3) else
if MachPos = 3 then Local(4) else
if MachPos = 4 then Local(5) else
if MachPos = 5 then Local(6) else
if MachPos = 6 then Local(7) else
if MachPos = 7 then Local(8) else
if MachPos = 8 then Local(9) else
if MachPos = 9 then Local(10) else
if MachPos = 10 then Local(11) else
if MachPos = 11 then Local(12) else
if MachPos = 12 then Local(13) else

```

```

if MachPos = 13 then Local(14) else
if MachPos = 14 then Local(15) else
if MachPos = 15 then Local(16) else
if MachPos = 16 then Local(17) else
if MachPos = 17 then Local(18) else
if MachPos = 18 then Local(19) else
if MachPos = 19 then Local(20) else
if MachPos = 20 then Local(21) else
if MachPos = 21 then Local(22) else
if MachPos = 22 then Local(23) else
if MachPos = 23 then Local(24) else
Local(25);

fun DefTranspLocation(loc) =
  if loc = Local(2) then 1 else
  if loc = Local(3) then 2 else
  if loc = Local(4) then 3 else
  if loc = Local(5) then 4 else
  if loc = Local(6) then 5 else
  if loc = Local(7) then 6 else
  if loc = Local(8) then 7 else
  if loc = Local(9) then 8 else
  if loc = Local(10) then 9 else
  if loc = Local(11) then 10 else
  if loc = Local(12) then 11 else
  if loc = Local(13) then 12 else
  if loc = Local(14) then 13 else
  if loc = Local(15) then 14 else
  if loc = Local(16) then 15 else
  if loc = Local(17) then 16 else
  if loc = Local(18) then 17 else
  if loc = Local(19) then 18 else
  if loc = Local(20) then 19 else
  if loc = Local(21) then 20 else
  if loc = Local(22) then 21 else
  if loc = Local(23) then 22 else
  if loc = Local(24) then 23 else
  24;

fun DefMachPiece(orinfo,pc) =
if pc = 1 then (orinfo,Mach(15))
else if pc = 2 then (orinfo,Mach(10))
else if pc = 3 then (orinfo,Mach(3))
else if pc = 4 then (orinfo,Mach(3))
else if pc = 5 then (orinfo,Mach(4))
else if pc = 6 then (orinfo,Mach(6))
else if pc = 7 then (orinfo,Mach(13))
else if pc = 8 then (orinfo,Mach(8))
else if pc = 9 then (orinfo,Mach(9))
else if pc = 10 then (orinfo,Mach(14))
else if pc = 11 then (orinfo,Mach(12))
else if pc = 12 then (orinfo,Mach(4))
else if pc = 13 then (orinfo,Mach(14))
else if pc = 14 then (orinfo,Mach(12))
else if pc = 15 then (orinfo,Mach(4))
else if pc = 16 then (orinfo,Mach(5))
else if pc = 17 then (orinfo,Mach(5))
else if pc = 18 then (orinfo,Mach(1))
else if pc = 19 then (orinfo,Mach(6))
else if pc = 20 then (orinfo,Mach(12))
else if pc = 21 then (orinfo,Mach(1))
else if pc = 22 then (orinfo,Mach(3))
else if pc = 23 then (orinfo,Mach(4))
else if pc = 24 then (orinfo,Mach(9))
else if pc = 25 then (orinfo,Mach(6))
else if pc = 26 then (orinfo,Mach(15))
else if pc = 27 then (orinfo,Mach(3))
else if pc = 28 then (orinfo,Mach(4))
else if pc = 29 then (orinfo,Mach(15))
else if pc = 30 then (orinfo,Mach(15))
else if pc = 31 then (orinfo,Mach(15))
else if pc = 32 then (orinfo,Mach(6))
else if pc = 33 then (orinfo,Mach(2))
else if pc = 34 then (orinfo,Mach(7))
else if pc = 35 then (orinfo,Mach(11))

```

```

else if pc = 36 then (orinfo,Mach(1))
else if pc = 37 then (orinfo,Mach(7))
else if pc = 38 then (orinfo,Mach(4))
else if pc = 39 then (orinfo,Mach(3))
else if pc = 40 then (orinfo,Mach(4))
else if pc = 41 then (orinfo,Mach(14))
else if pc = 42 then (orinfo,Mach(7))
else if pc = 43 then (orinfo,Mach(14))
else if pc = 44 then (orinfo,Mach(9))
else if pc = 45 then (orinfo,Mach(4))
else if pc = 46 then (orinfo,Mach(3))
else if pc = 47 then (orinfo,Mach(14))
else if pc = 48 then (orinfo,Mach(10))
else if pc = 49 then (orinfo,Mach(2))
else if pc = 50 then (orinfo,Mach(7))
else if pc = 51 then (orinfo,Mach(4))
else if pc = 52 then (orinfo,Mach(4))
else if pc = 53 then (orinfo,Mach(6))
else if pc = 54 then (orinfo,Mach(7))
else if pc = 55 then (orinfo,Mach(11))
else if pc = 56 then (orinfo,Mach(7))
else if pc = 57 then (orinfo,Mach(15))
else if pc = 58 then (orinfo,Mach(7))
else if pc = 59 then (orinfo,Mach(5))
else if pc = 60 then (orinfo,Mach(11))
else if pc = 61 then (orinfo,Mach(1))
else if pc = 62 then (orinfo,Mach(10))
else if pc = 63 then (orinfo,Mach(14))
else if pc = 64 then (orinfo,Mach(12))
else if pc = 65 then (orinfo,Mach(5))
else if pc = 66 then (orinfo,Mach(8))
else if pc = 67 then (orinfo,Mach(3))
else if pc = 68 then (orinfo,Mach(7))
else if pc = 69 then (orinfo,Mach(7))
else if pc = 70 then (orinfo,Mach(12))
else if pc = 71 then (orinfo,Mach(10))
else if pc = 72 then (orinfo,Mach(15))
else if pc = 73 then (orinfo,Mach(15))
else if pc = 74 then (orinfo,Mach(14))
else if pc = 75 then (orinfo,Mach(1))
else if pc = 76 then (orinfo,Mach(3))
else if pc = 77 then (orinfo,Mach(9))
else if pc = 78 then (orinfo,Mach(3))
else if pc = 79 then (orinfo,Mach(7))
else if pc = 80 then (orinfo,Mach(15))
else if pc = 81 then (orinfo,Mach(7))
else if pc = 82 then (orinfo,Mach(8))
else if pc = 83 then (orinfo,Mach(8))
else if pc = 84 then (orinfo,Mach(5))
else if pc = 85 then (orinfo,Mach(4))
else if pc = 86 then (orinfo,Mach(13))
else if pc = 87 then (orinfo,Mach(4))
else if pc = 88 then (orinfo,Mach(6))
else if pc = 89 then (orinfo,Mach(6))
else if pc = 90 then (orinfo,Mach(9))
else if pc = 91 then (orinfo,Mach(10))
else if pc = 92 then (orinfo,Mach(8))
else if pc = 93 then (orinfo,Mach(11))
else if pc = 94 then (orinfo,Mach(13))
else if pc = 95 then (orinfo,Mach(1))
else if pc = 96 then (orinfo,Mach(9))
else if pc = 97 then (orinfo,Mach(3))
else if pc = 98 then (orinfo,Mach(15))
else if pc = 99 then (orinfo,Mach(15))
else (orinfo,Mach(12));

```

```

fun TransfProdPc(pc) =
case pc of 1 => Prod(1) |
2 => Prod(2) |
3 => Prod(3) |
4 => Prod(4) |
5 => Prod(5) |
6 => Prod(6) |
7 => Prod(7) |
8 => Prod(8) |

```



```
9 => Prod(9) |
10 => Prod(10) |
11 => Prod(11) |
12 => Prod(12) |
13 => Prod(13) |
14 => Prod(14) |
15 => Prod(15) |
16 => Prod(16) |
17 => Prod(17) |
18 => Prod(18) |
19 => Prod(19) |
20 => Prod(20) |
21 => Prod(21) |
22 => Prod(22) |
23 => Prod(23) |
24 => Prod(24) |
25 => Prod(25) |
26 => Prod(26) |
27 => Prod(27) |
28 => Prod(28) |
29 => Prod(29) |
30 => Prod(30) |
31 => Prod(31) |
32 => Prod(32) |
33 => Prod(33) |
34 => Prod(34) |
35 => Prod(35) |
36 => Prod(36) |
37 => Prod(37) |
38 => Prod(38) |
39 => Prod(39) |
40 => Prod(40) |
41 => Prod(41) |
42 => Prod(42) |
43 => Prod(43) |
44 => Prod(44) |
45 => Prod(45) |
46 => Prod(46) |
47 => Prod(47) |
48 => Prod(48) |
49 => Prod(49) |
50 => Prod(50) |
51 => Prod(51) |
52 => Prod(52) |
53 => Prod(53) |
54 => Prod(54) |
55 => Prod(55) |
56 => Prod(56) |
57 => Prod(57) |
58 => Prod(58) |
59 => Prod(59) |
60 => Prod(60) |
61 => Prod(61) |
62 => Prod(62) |
63 => Prod(63) |
64 => Prod(64) |
65 => Prod(65) |
66 => Prod(66) |
67 => Prod(67) |
68 => Prod(68) |
69 => Prod(69) |
70 => Prod(70) |
71 => Prod(71) |
72 => Prod(72) |
73 => Prod(73) |
74 => Prod(74) |
75 => Prod(75) |
76 => Prod(76) |
77 => Prod(77) |
78 => Prod(78) |
79 => Prod(79) |
80 => Prod(80) |
81 => Prod(81) |
82 => Prod(82) |
83 => Prod(83) |
```

```

84 => Prod(84) |
85 => Prod(85) |
86 => Prod(86) |
87 => Prod(87) |
88 => Prod(88) |
89 => Prod(89) |
90 => Prod(90) |
91 => Prod(91) |
92 => Prod(92) |
93 => Prod(93) |
94 => Prod(94) |
95 => Prod(95) |
96 => Prod(96) |
97 => Prod(97) |
98 => Prod(98) |
99 => Prod(99) |
100 => Prod(100);

{ *****Insiro aqui as amostras de layout***** }

fun CalcMachPosIn(pin,LMListIn,L) =
  if hd(LMListIn) = pin then L - length(tl(LMListIn))
  else
    CalcMachPosIn(pin,tl(LMListIn),L);

fun DefMachPosIn(MaqIn,LMListIn) =
  let
    val L = length(LMListIn);
  in
    if MaqIn = Mach(1) then CalcMachPosIn(1,LMListIn,L) else
    if MaqIn = Mach(2) then CalcMachPosIn(2,LMListIn,L) else
    if MaqIn = Mach(3) then CalcMachPosIn(3,LMListIn,L) else
    if MaqIn = Mach(4) then CalcMachPosIn(4,LMListIn,L) else
    if MaqIn = Mach(5) then CalcMachPosIn(5,LMListIn,L) else
    if MaqIn = Mach(6) then CalcMachPosIn(6,LMListIn,L) else
    if MaqIn = Mach(7) then CalcMachPosIn(7,LMListIn,L) else
    if MaqIn = Mach(8) then CalcMachPosIn(8,LMListIn,L) else
    if MaqIn = Mach(9) then CalcMachPosIn(9,LMListIn,L) else
    if MaqIn = Mach(10) then CalcMachPosIn(10,LMListIn,L) else
    if MaqIn = Mach(11) then CalcMachPosIn(11,LMListIn,L) else
    if MaqIn = Mach(12) then CalcMachPosIn(12,LMListIn,L) else
    if MaqIn = Mach(13) then CalcMachPosIn(13,LMListIn,L) else
    if MaqIn = Mach(14) then CalcMachPosIn(14,LMListIn,L) else
    CalcMachPosIn(15,LMListIn,L)
  end;

fun CalcMachPosInter(pint,LMListInter,L) =
  if hd(LMListInter) = pint then L - length(tl(LMListInter))
  else
    CalcMachPosInter(pint,tl(LMListInter),L);

fun DefMachPosInter(MaqInter,LMListInter) =
  let
    val L = length(LMListInter);
  in
    if MaqInter = Mach(1) then CalcMachPosInter(1,LMListInter,L) else
    if MaqInter = Mach(2) then CalcMachPosInter(2,LMListInter,L) else
    if MaqInter = Mach(3) then CalcMachPosInter(3,LMListInter,L) else
    if MaqInter = Mach(4) then CalcMachPosInter(4,LMListInter,L) else
    if MaqInter = Mach(5) then CalcMachPosInter(5,LMListInter,L) else
    if MaqInter = Mach(6) then CalcMachPosInter(6,LMListInter,L) else
    if MaqInter = Mach(7) then CalcMachPosInter(7,LMListInter,L) else
    if MaqInter = Mach(8) then CalcMachPosInter(8,LMListInter,L) else
    if MaqInter = Mach(9) then CalcMachPosInter(9,LMListInter,L) else
    if MaqInter = Mach(10) then CalcMachPosInter(10,LMListInter,L) else
    if MaqInter = Mach(11) then CalcMachPosInter(11,LMListInter,L) else
    if MaqInter = Mach(12) then CalcMachPosInter(12,LMListInter,L) else
    if MaqInter = Mach(13) then CalcMachPosInter(13,LMListInter,L) else
    if MaqInter = Mach(14) then CalcMachPosInter(14,LMListInter,L) else
    CalcMachPosInter(15,LMListInter,L)
  end;

fun DefMachDist(p)=
  let

```

```

val MachListDist = [8,7,6,6,7,8,7,6,5,5,6,7,6,5,4,4,5,6,5,4,3,3,4,5];

in

    List.nth(MachListDist,p-1)
end;

{ ***** Insira aqui as distancias De-Para***** }

fun DefMachDistOut(p)=
let

val MachListDist = [5,4,3,3,4,5,6,5,4,4,5,6,7,6,5,5,6,7,8,7,6,6,7,8];

in

    List.nth(MachListDist,p-1)
end;

fun DefTranspDistIn(MachPos,loc) =
if loc = Local(1) then DefMachDist(MachPos)
else
    DefMachDistIn(loc,MachPos);

fun DefTranspDistInter(MachPos,loc) =
if loc = Local(1) then DefMachDist(MachPos)
else
    DefMachDistIn(loc,MachPos);

fun DefTranspFromToInter(TransPos,loc) =

if loc = Local(1) then DefMachDist(TransPos)
else if loc = Local(26) then DefMachDistOut(TransPos)
else DefMachDistIn(loc,TransPos);

fun DefTranspFromToExit(TransPos,loc) =

if loc = Local(1) then DefMachDist(TransPos)
else if loc = Local(26) then DefMachDistOut(TransPos)
else DefMachDistIn(loc,TransPos);

fun DefWeigthFactor() =
let
    val w = !wet;
    val wr = real(w) / 1000.0
    val TotalTranspW = real(!TranspW) / 1000.0
in
    0.75 + 0.25 * ((TotalTranspW - wr) / TotalTranspW)
end;

fun DefWeigthFactorInter(w) =
let
    val wr = real(w) / 1000.0;
    val TotalTranspW = real(!TranspW) / 1000.0
in
    0.75 + 0.25 * ((TotalTranspW - wr) / TotalTranspW)
end;

fun DefWeigthFactorExit() =
let
    val w = !TranspWetExit;
    val wr = real(w) / 1000.0
    val TotalTranspW = real(!TranspW) / 1000.0
in
    0.75 + 0.25 * ((TotalTranspW - wr) / TotalTranspW)
end;

fun DefWeigthFactorExp() =
let
    val w = !TranspWetExp;
    val wr = real(w) / 1000.0
    val TotalTranspW = real(!TranspW) / 1000.0
in
    0.75 + 0.25 * ((TotalTranspW - wr) / TotalTranspW)
end;

```

```

fun DefInputMaq(Maq) =
case Maq of Mach(1) => 1^(1,Mach(1)) |
Mach(2) => 1^(1,Mach(2)) |
Mach(3) => 1^(2,Mach(3)) |
Mach(4) => 1^(2,Mach(4)) |
Mach(5) => 1^(2,Mach(5)) |
Mach(6) => 1^(2,Mach(6)) |
Mach(7) => 1^(2,Mach(7)) |
Mach(8) => 1^(2,Mach(8)) |
Mach(9) => 1^(2,Mach(9)) |
Mach(10) => 1^(1,Mach(10)) |
Mach(11) => 1^(1,Mach(11)) |
Mach(12) => 1^(1,Mach(12)) |
Mach(13) => 1^(1,Mach(13)) |
Mach(14) => 1^(2,Mach(14)) |
Mach(15) => 1^(2,Mach(15));

fun TimeProdOPT(ProdT,OPTlistOPTon) = uniform(30.0,32.0);

fun SetupTime() = 1200;

fun DefProcIniProd(ProdT) = (ProdT,1);

fun DefProcSemiProd(ProdTSemi,OPTlistSemi) = (ProdTSemi,OPTlistSemi + 1);

fun DefOperacao(ProdT,OPTlistOPTon) =
let
  val OPTListProd1 = [OP(15)];
  val OPTListProd2 = [OP(10)];
  val OPTListProd3 = [OP(3),OP(3),OP(14),OP(7)];
  val OPTListProd4 = [OP(3)];
  val OPTListProd5 = [OP(4),OP(6)];
  val OPTListProd6 = [OP(6),OP(7),OP(7)];
  val OPTListProd7 = [OP(13),OP(4)];
  val OPTListProd8 = [OP(8)];
  val OPTListProd9 = [OP(9),OP(10)];
  val OPTListProd10 = [OP(14),OP(6)];
  val OPTListProd11 = [OP(12)];
  val OPTListProd12 = [OP(4),OP(13)];
  val OPTListProd13 = [OP(14)];
  val OPTListProd14 = [OP(12),OP(5),OP(7),OP(4)];
  val OPTListProd15 = [OP(4)];
  val OPTListProd16 = [OP(5),OP(11)];
  val OPTListProd17 = [OP(5),OP(13),OP(4),OP(7)];
  val OPTListProd18 = [OP(1),OP(10)];
  val OPTListProd19 = [OP(6),OP(5),OP(5),OP(10)];
  val OPTListProd20 = [OP(12),OP(8)];
  val OPTListProd21 = [OP(1),OP(14),OP(1)];
  val OPTListProd22 = [OP(3),OP(6),OP(1),OP(11)];
  val OPTListProd23 = [OP(4),OP(2),OP(2)];
  val OPTListProd24 = [OP(9),OP(12)];
  val OPTListProd25 = [OP(6)];
  val OPTListProd26 = [OP(15),OP(2)];
  val OPTListProd27 = [OP(3),OP(15)];
  val OPTListProd28 = [OP(4)];
  val OPTListProd29 = [OP(15),OP(1),OP(12),OP(7)];
  val OPTListProd30 = [OP(15),OP(10)];
  val OPTListProd31 = [OP(15),OP(12)];
  val OPTListProd32 = [OP(6)];
  val OPTListProd33 = [OP(2),OP(5),OP(14),OP(14)];
  val OPTListProd34 = [OP(7),OP(6)];
  val OPTListProd35 = [OP(11),OP(5),OP(7),OP(3)];
  val OPTListProd36 = [OP(1),OP(6),OP(9),OP(9)];
  val OPTListProd37 = [OP(7),OP(15)];
  val OPTListProd38 = [OP(4),OP(10),OP(1),OP(12)];
  val OPTListProd39 = [OP(3),OP(2),OP(12),OP(7)];
  val OPTListProd40 = [OP(4)];
  val OPTListProd41 = [OP(14)];
  val OPTListProd42 = [OP(7)];
  val OPTListProd43 = [OP(14),OP(11),OP(4)];
  val OPTListProd44 = [OP(9)];
  val OPTListProd45 = [OP(4),OP(13),OP(7)];
  val OPTListProd46 = [OP(3)];
  val OPTListProd47 = [OP(14),OP(8),OP(4),OP(6)];

```

```

val OPTListProd48 = [OP(10),OP(5),OP(9)]
val OPTListProd49 = [OP(2),OP(4),OP(9),OP(5)]
val OPTListProd50 = [OP(7),OP(11),OP(3)]
val OPTListProd51 = [OP(4),OP(5),OP(14)]
val OPTListProd52 = [OP(4)]
val OPTListProd53 = [OP(6),OP(6),OP(11)]
val OPTListProd54 = [OP(7),OP(9),OP(4)]
val OPTListProd55 = [OP(11),OP(4),OP(1),OP(14)]
val OPTListProd56 = [OP(7),OP(9),OP(8)]
val OPTListProd57 = [OP(15),OP(3),OP(9),OP(7)]
val OPTListProd58 = [OP(7)]
val OPTListProd59 = [OP(5),OP(13),OP(3)]
val OPTListProd60 = [OP(11),OP(5)]
val OPTListProd61 = [OP(1)]
val OPTListProd62 = [OP(10),OP(9),OP(9),OP(12)]
val OPTListProd63 = [OP(14),OP(9)]
val OPTListProd64 = [OP(12),OP(4),OP(14),OP(14)]
val OPTListProd65 = [OP(5),OP(3),OP(6)]
val OPTListProd66 = [OP(8),OP(9),OP(15)]
val OPTListProd67 = [OP(3),OP(5),OP(4),OP(3)]
val OPTListProd68 = [OP(7),OP(4),OP(6)]
val OPTListProd69 = [OP(7),OP(6)]
val OPTListProd70 = [OP(12)]
val OPTListProd71 = [OP(10),OP(8),OP(4)]
val OPTListProd72 = [OP(15),OP(15),OP(13),OP(7)]
val OPTListProd73 = [OP(15),OP(10),OP(13),OP(9)]
val OPTListProd74 = [OP(14),OP(6),OP(3)]
val OPTListProd75 = [OP(1)]
val OPTListProd76 = [OP(3)]
val OPTListProd77 = [OP(9),OP(12),OP(10),OP(7)]
val OPTListProd78 = [OP(3),OP(5),OP(5),OP(14)]
val OPTListProd79 = [OP(7),OP(11),OP(6)]
val OPTListProd80 = [OP(15),OP(8),OP(7),OP(5)]
val OPTListProd81 = [OP(7),OP(7),OP(4)]
val OPTListProd82 = [OP(8),OP(14)]
val OPTListProd83 = [OP(8)]
val OPTListProd84 = [OP(5),OP(8),OP(9)]
val OPTListProd85 = [OP(4),OP(2)]
val OPTListProd86 = [OP(13),OP(4),OP(9),OP(5)]
val OPTListProd87 = [OP(4)]
val OPTListProd88 = [OP(6),OP(9)]
val OPTListProd89 = [OP(6),OP(7),OP(4)]
val OPTListProd90 = [OP(9),OP(8),OP(10),OP(14)]
val OPTListProd91 = [OP(10)]
val OPTListProd92 = [OP(8)]
val OPTListProd93 = [OP(11),OP(8)]
val OPTListProd94 = [OP(13)]
val OPTListProd95 = [OP(1),OP(1),OP(14)]
val OPTListProd96 = [OP(9),OP(12)]
val OPTListProd97 = [OP(3),OP(14)]
val OPTListProd98 = [OP(15)]
val OPTListProd99 = [OP(15),OP(6),OP(1)]
val OPTListProd100 = [OP(12),OP(7),OP(15)]

in
case ProdT of Prod(1) => List.nth(OPTListProd1,OPTlistOPTon-1) |
Prod(2) => List.nth(OPTListProd2,OPTlistOPTon-1) |
Prod(3) => List.nth(OPTListProd3,OPTlistOPTon-1) |
Prod(4) => List.nth(OPTListProd4,OPTlistOPTon-1) |
Prod(5) => List.nth(OPTListProd5,OPTlistOPTon-1) |
Prod(6) => List.nth(OPTListProd6,OPTlistOPTon-1) |
Prod(7) => List.nth(OPTListProd7,OPTlistOPTon-1) |
Prod(8) => List.nth(OPTListProd8,OPTlistOPTon-1) |
Prod(9) => List.nth(OPTListProd9,OPTlistOPTon-1) |
Prod(10) => List.nth(OPTListProd10,OPTlistOPTon-1) |
Prod(11) => List.nth(OPTListProd11,OPTlistOPTon-1) |
Prod(12) => List.nth(OPTListProd12,OPTlistOPTon-1) |
Prod(13) => List.nth(OPTListProd13,OPTlistOPTon-1) |
Prod(14) => List.nth(OPTListProd14,OPTlistOPTon-1) |
Prod(15) => List.nth(OPTListProd15,OPTlistOPTon-1) |
Prod(16) => List.nth(OPTListProd16,OPTlistOPTon-1) |
Prod(17) => List.nth(OPTListProd17,OPTlistOPTon-1) |
Prod(18) => List.nth(OPTListProd18,OPTlistOPTon-1) |
Prod(19) => List.nth(OPTListProd19,OPTlistOPTon-1) |
Prod(20) => List.nth(OPTListProd20,OPTlistOPTon-1) |
Prod(21) => List.nth(OPTListProd21,OPTlistOPTon-1) |

```



```

Prod(97) => List.nth(OPTListProd97,OPTlistOPTon-1) |
Prod(98) => List.nth(OPTListProd98,OPTlistOPTon-1) |
Prod(99) => List.nth(OPTListProd99,OPTlistOPTon-1) |
Prod(100) => List.nth(OPTListProd100,OPTlistOPTon-1)
end;

fun DefMaqTipoProd(ProdT,OPTlistOPTon) =
let
  val MachListProd1 = [Mach(15)];
  val MachListProd2 = [Mach(10)];
  val MachListProd3 = [Mach(3),Mach(3),Mach(14),Mach(7)];
  val MachListProd4 = [Mach(3)];
  val MachListProd5 = [Mach(4),Mach(6)];
  val MachListProd6 = [Mach(6),Mach(7),Mach(7)];
  val MachListProd7 = [Mach(13),Mach(4)];
  val MachListProd8 = [Mach(8)];
  val MachListProd9 = [Mach(9),Mach(10)];
  val MachListProd10 = [Mach(14),Mach(6)];
  val MachListProd11 = [Mach(12)];
  val MachListProd12 = [Mach(4),Mach(13)];
  val MachListProd13 = [Mach(14)];
  val MachListProd14 = [Mach(12),Mach(5),Mach(7),Mach(4)];
  val MachListProd15 = [Mach(4)];
  val MachListProd16 = [Mach(5),Mach(11)];
  val MachListProd17 = [Mach(5),Mach(13),Mach(4),Mach(7)];
  val MachListProd18 = [Mach(1),Mach(10)];
  val MachListProd19 = [Mach(6),Mach(5),Mach(5),Mach(10)];
  val MachListProd20 = [Mach(12),Mach(8)];
  val MachListProd21 = [Mach(1),Mach(14),Mach(1)];
  val MachListProd22 = [Mach(3),Mach(6),Mach(1),Mach(11)];
  val MachListProd23 = [Mach(4),Mach(2),Mach(2)];
  val MachListProd24 = [Mach(9),Mach(12)];
  val MachListProd25 = [Mach(6)];
  val MachListProd26 = [Mach(15),Mach(2)];
  val MachListProd27 = [Mach(3),Mach(15)];
  val MachListProd28 = [Mach(4)];
  val MachListProd29 = [Mach(15),Mach(1),Mach(12),Mach(7)];
  val MachListProd30 = [Mach(15),Mach(10)];
  val MachListProd31 = [Mach(15),Mach(12)];
  val MachListProd32 = [Mach(6)];
  val MachListProd33 = [Mach(2),Mach(5),Mach(14),Mach(14)];
  val MachListProd34 = [Mach(7),Mach(6)];
  val MachListProd35 = [Mach(11),Mach(5),Mach(7),Mach(3)];
  val MachListProd36 = [Mach(1),Mach(6),Mach(9),Mach(9)];
  val MachListProd37 = [Mach(7),Mach(15)];
  val MachListProd38 = [Mach(4),Mach(10),Mach(1),Mach(12)];
  val MachListProd39 = [Mach(3),Mach(2),Mach(12),Mach(7)];
  val MachListProd40 = [Mach(4)];
  val MachListProd41 = [Mach(14)];
  val MachListProd42 = [Mach(7)];
  val MachListProd43 = [Mach(14),Mach(11),Mach(4)];
  val MachListProd44 = [Mach(9)];
  val MachListProd45 = [Mach(4),Mach(13),Mach(7)];
  val MachListProd46 = [Mach(3)];
  val MachListProd47 = [Mach(14),Mach(8),Mach(4),Mach(6)];
  val MachListProd48 = [Mach(10),Mach(5),Mach(9)];
  val MachListProd49 = [Mach(2),Mach(4),Mach(9),Mach(5)];
  val MachListProd50 = [Mach(7),Mach(11),Mach(3)];
  val MachListProd51 = [Mach(4),Mach(5),Mach(14)];
  val MachListProd52 = [Mach(4)];
  val MachListProd53 = [Mach(6),Mach(6),Mach(11)];
  val MachListProd54 = [Mach(7),Mach(9),Mach(4)];
  val MachListProd55 = [Mach(11),Mach(4),Mach(1),Mach(14)];
  val MachListProd56 = [Mach(7),Mach(9),Mach(8)];
  val MachListProd57 = [Mach(15),Mach(3),Mach(9),Mach(7)];
  val MachListProd58 = [Mach(7)];
  val MachListProd59 = [Mach(5),Mach(13),Mach(3)];
  val MachListProd60 = [Mach(11),Mach(5)];
  val MachListProd61 = [Mach(1)];
  val MachListProd62 = [Mach(10),Mach(9),Mach(9),Mach(12)];
  val MachListProd63 = [Mach(14),Mach(9)];
  val MachListProd64 = [Mach(12),Mach(4),Mach(14),Mach(14)];
  val MachListProd65 = [Mach(5),Mach(3),Mach(6)];
  val MachListProd66 = [Mach(8),Mach(9),Mach(15)];
  val MachListProd67 = [Mach(3),Mach(5),Mach(4),Mach(3)];

```

```

val MachListProd68 = [Mach(7),Mach(4),Mach(6)]
val MachListProd69 = [Mach(7),Mach(6)]
val MachListProd70 = [Mach(12)]
val MachListProd71 = [Mach(10),Mach(8),Mach(4)]
val MachListProd72 = [Mach(15),Mach(15),Mach(13),Mach(7)]
val MachListProd73 = [Mach(15),Mach(10),Mach(13),Mach(9)]
val MachListProd74 = [Mach(14),Mach(6),Mach(3)]
val MachListProd75 = [Mach(1)]
val MachListProd76 = [Mach(3)]
val MachListProd77 = [Mach(9),Mach(12),Mach(10),Mach(7)]
val MachListProd78 = [Mach(3),Mach(5),Mach(5),Mach(14)]
val MachListProd79 = [Mach(7),Mach(11),Mach(6)]
val MachListProd80 = [Mach(15),Mach(8),Mach(7),Mach(5)]
val MachListProd81 = [Mach(7),Mach(7),Mach(4)]
val MachListProd82 = [Mach(8),Mach(14)]
val MachListProd83 = [Mach(8)]
val MachListProd84 = [Mach(5),Mach(8),Mach(9)]
val MachListProd85 = [Mach(4),Mach(2)]
val MachListProd86 = [Mach(13),Mach(4),Mach(9),Mach(5)]
val MachListProd87 = [Mach(4)]
val MachListProd88 = [Mach(6),Mach(9)]
val MachListProd89 = [Mach(6),Mach(7),Mach(4)]
val MachListProd90 = [Mach(9),Mach(8),Mach(10),Mach(14)]
val MachListProd91 = [Mach(10)]
val MachListProd92 = [Mach(8)]
val MachListProd93 = [Mach(11),Mach(8)]
val MachListProd94 = [Mach(13)]
val MachListProd95 = [Mach(1),Mach(1),Mach(14)]
val MachListProd96 = [Mach(9),Mach(12)]
val MachListProd97 = [Mach(3),Mach(14)]
val MachListProd98 = [Mach(15)]
val MachListProd99 = [Mach(15),Mach(6),Mach(1)]
val MachListProd100 = [Mach(12),Mach(7),Mach(15)]

in
case ProdT of Prod(1) => List.nth(MachListProd1,OPTlistOPTon-1) |
Prod(2) => List.nth(MachListProd2,OPTlistOPTon-1) |
Prod(3) => List.nth(MachListProd3,OPTlistOPTon-1) |
Prod(4) => List.nth(MachListProd4,OPTlistOPTon-1) |
Prod(5) => List.nth(MachListProd5,OPTlistOPTon-1) |
Prod(6) => List.nth(MachListProd6,OPTlistOPTon-1) |
Prod(7) => List.nth(MachListProd7,OPTlistOPTon-1) |
Prod(8) => List.nth(MachListProd8,OPTlistOPTon-1) |
Prod(9) => List.nth(MachListProd9,OPTlistOPTon-1) |
Prod(10) => List.nth(MachListProd10,OPTlistOPTon-1) |
Prod(11) => List.nth(MachListProd11,OPTlistOPTon-1) |
Prod(12) => List.nth(MachListProd12,OPTlistOPTon-1) |
Prod(13) => List.nth(MachListProd13,OPTlistOPTon-1) |
Prod(14) => List.nth(MachListProd14,OPTlistOPTon-1) |
Prod(15) => List.nth(MachListProd15,OPTlistOPTon-1) |
Prod(16) => List.nth(MachListProd16,OPTlistOPTon-1) |
Prod(17) => List.nth(MachListProd17,OPTlistOPTon-1) |
Prod(18) => List.nth(MachListProd18,OPTlistOPTon-1) |
Prod(19) => List.nth(MachListProd19,OPTlistOPTon-1) |
Prod(20) => List.nth(MachListProd20,OPTlistOPTon-1) |
Prod(21) => List.nth(MachListProd21,OPTlistOPTon-1) |
Prod(22) => List.nth(MachListProd22,OPTlistOPTon-1) |
Prod(23) => List.nth(MachListProd23,OPTlistOPTon-1) |
Prod(24) => List.nth(MachListProd24,OPTlistOPTon-1) |
Prod(25) => List.nth(MachListProd25,OPTlistOPTon-1) |
Prod(26) => List.nth(MachListProd26,OPTlistOPTon-1) |
Prod(27) => List.nth(MachListProd27,OPTlistOPTon-1) |
Prod(28) => List.nth(MachListProd28,OPTlistOPTon-1) |
Prod(29) => List.nth(MachListProd29,OPTlistOPTon-1) |
Prod(30) => List.nth(MachListProd30,OPTlistOPTon-1) |
Prod(31) => List.nth(MachListProd31,OPTlistOPTon-1) |
Prod(32) => List.nth(MachListProd32,OPTlistOPTon-1) |
Prod(33) => List.nth(MachListProd33,OPTlistOPTon-1) |
Prod(34) => List.nth(MachListProd34,OPTlistOPTon-1) |
Prod(35) => List.nth(MachListProd35,OPTlistOPTon-1) |
Prod(36) => List.nth(MachListProd36,OPTlistOPTon-1) |
Prod(37) => List.nth(MachListProd37,OPTlistOPTon-1) |
Prod(38) => List.nth(MachListProd38,OPTlistOPTon-1) |
Prod(39) => List.nth(MachListProd39,OPTlistOPTon-1) |
Prod(40) => List.nth(MachListProd40,OPTlistOPTon-1) |
Prod(41) => List.nth(MachListProd41,OPTlistOPTon-1) |

```



```

Prod(42) => List.nth(MachListProd42,OPTlistOPTon-1) |
Prod(43) => List.nth(MachListProd43,OPTlistOPTon-1) |
Prod(44) => List.nth(MachListProd44,OPTlistOPTon-1) |
Prod(45) => List.nth(MachListProd45,OPTlistOPTon-1) |
Prod(46) => List.nth(MachListProd46,OPTlistOPTon-1) |
Prod(47) => List.nth(MachListProd47,OPTlistOPTon-1) |
Prod(48) => List.nth(MachListProd48,OPTlistOPTon-1) |
Prod(49) => List.nth(MachListProd49,OPTlistOPTon-1) |
Prod(50) => List.nth(MachListProd50,OPTlistOPTon-1) |
Prod(51) => List.nth(MachListProd51,OPTlistOPTon-1) |
Prod(52) => List.nth(MachListProd52,OPTlistOPTon-1) |
Prod(53) => List.nth(MachListProd53,OPTlistOPTon-1) |
Prod(54) => List.nth(MachListProd54,OPTlistOPTon-1) |
Prod(55) => List.nth(MachListProd55,OPTlistOPTon-1) |
Prod(56) => List.nth(MachListProd56,OPTlistOPTon-1) |
Prod(57) => List.nth(MachListProd57,OPTlistOPTon-1) |
Prod(58) => List.nth(MachListProd58,OPTlistOPTon-1) |
Prod(59) => List.nth(MachListProd59,OPTlistOPTon-1) |
Prod(60) => List.nth(MachListProd60,OPTlistOPTon-1) |
Prod(61) => List.nth(MachListProd61,OPTlistOPTon-1) |
Prod(62) => List.nth(MachListProd62,OPTlistOPTon-1) |
Prod(63) => List.nth(MachListProd63,OPTlistOPTon-1) |
Prod(64) => List.nth(MachListProd64,OPTlistOPTon-1) |
Prod(65) => List.nth(MachListProd65,OPTlistOPTon-1) |
Prod(66) => List.nth(MachListProd66,OPTlistOPTon-1) |
Prod(67) => List.nth(MachListProd67,OPTlistOPTon-1) |
Prod(68) => List.nth(MachListProd68,OPTlistOPTon-1) |
Prod(69) => List.nth(MachListProd69,OPTlistOPTon-1) |
Prod(70) => List.nth(MachListProd70,OPTlistOPTon-1) |
Prod(71) => List.nth(MachListProd71,OPTlistOPTon-1) |
Prod(72) => List.nth(MachListProd72,OPTlistOPTon-1) |
Prod(73) => List.nth(MachListProd73,OPTlistOPTon-1) |
Prod(74) => List.nth(MachListProd74,OPTlistOPTon-1) |
Prod(75) => List.nth(MachListProd75,OPTlistOPTon-1) |
Prod(76) => List.nth(MachListProd76,OPTlistOPTon-1) |
Prod(77) => List.nth(MachListProd77,OPTlistOPTon-1) |
Prod(78) => List.nth(MachListProd78,OPTlistOPTon-1) |
Prod(79) => List.nth(MachListProd79,OPTlistOPTon-1) |
Prod(80) => List.nth(MachListProd80,OPTlistOPTon-1) |
Prod(81) => List.nth(MachListProd81,OPTlistOPTon-1) |
Prod(82) => List.nth(MachListProd82,OPTlistOPTon-1) |
Prod(83) => List.nth(MachListProd83,OPTlistOPTon-1) |
Prod(84) => List.nth(MachListProd84,OPTlistOPTon-1) |
Prod(85) => List.nth(MachListProd85,OPTlistOPTon-1) |
Prod(86) => List.nth(MachListProd86,OPTlistOPTon-1) |
Prod(87) => List.nth(MachListProd87,OPTlistOPTon-1) |
Prod(88) => List.nth(MachListProd88,OPTlistOPTon-1) |
Prod(89) => List.nth(MachListProd89,OPTlistOPTon-1) |
Prod(90) => List.nth(MachListProd90,OPTlistOPTon-1) |
Prod(91) => List.nth(MachListProd91,OPTlistOPTon-1) |
Prod(92) => List.nth(MachListProd92,OPTlistOPTon-1) |
Prod(93) => List.nth(MachListProd93,OPTlistOPTon-1) |
Prod(94) => List.nth(MachListProd94,OPTlistOPTon-1) |
Prod(95) => List.nth(MachListProd95,OPTlistOPTon-1) |
Prod(96) => List.nth(MachListProd96,OPTlistOPTon-1) |
Prod(97) => List.nth(MachListProd97,OPTlistOPTon-1) |
Prod(98) => List.nth(MachListProd98,OPTlistOPTon-1) |
Prod(99) => List.nth(MachListProd99,OPTlistOPTon-1) |
Prod(100) => List.nth(MachListProd100,OPTlistOPTon-1) |
end;

fun                                     OperacaoOn(ProdT,OPTlistOPTon)                =
(ProdT,OPTlistOPTon,DefMaqTipoProd(ProdT,OPTlistOPTon),DefOperacao(ProdT,OPTlistOPTon));

fun DefMaqInter(ProdT,OPTlistOPTon) =
let
  val MachListProd1 = [Mach(15)];
  val MachListProd2 = [Mach(10)];
  val MachListProd3 = [Mach(3),Mach(3),Mach(3),Mach(14),Mach(7)]
  val MachListProd4 = [Mach(3)]
  val MachListProd5 = [Mach(4),Mach(6)]
  val MachListProd6 = [Mach(6),Mach(7),Mach(7)]
  val MachListProd7 = [Mach(13),Mach(4)]
  val MachListProd8 = [Mach(8)]
  val MachListProd9 = [Mach(9),Mach(10)]

```

```

val MachListProd10 = [Mach(14),Mach(6)]
val MachListProd11 = [Mach(12)]
val MachListProd12 = [Mach(4),Mach(13)]
val MachListProd13 = [Mach(14)]
val MachListProd14 = [Mach(12),Mach(5),Mach(7),Mach(4)]
val MachListProd15 = [Mach(4)]
val MachListProd16 = [Mach(5),Mach(11)]
val MachListProd17 = [Mach(5),Mach(13),Mach(4),Mach(7)]
val MachListProd18 = [Mach(1),Mach(10)]
val MachListProd19 = [Mach(6),Mach(5),Mach(5),Mach(10)]
val MachListProd20 = [Mach(12),Mach(8)]
val MachListProd21 = [Mach(1),Mach(14),Mach(1)]
val MachListProd22 = [Mach(3),Mach(6),Mach(1),Mach(11)]
val MachListProd23 = [Mach(4),Mach(2),Mach(2)]
val MachListProd24 = [Mach(9),Mach(12)]
val MachListProd25 = [Mach(6)]
val MachListProd26 = [Mach(15),Mach(2)]
val MachListProd27 = [Mach(3),Mach(15)]
val MachListProd28 = [Mach(4)]
val MachListProd29 = [Mach(15),Mach(1),Mach(12),Mach(7)]
val MachListProd30 = [Mach(15),Mach(10)]
val MachListProd31 = [Mach(15),Mach(12)]
val MachListProd32 = [Mach(6)]
val MachListProd33 = [Mach(2),Mach(5),Mach(14),Mach(14)]
val MachListProd34 = [Mach(7),Mach(6)]
val MachListProd35 = [Mach(11),Mach(5),Mach(7),Mach(3)]
val MachListProd36 = [Mach(1),Mach(6),Mach(9),Mach(9)]
val MachListProd37 = [Mach(7),Mach(15)]
val MachListProd38 = [Mach(4),Mach(10),Mach(1),Mach(12)]
val MachListProd39 = [Mach(3),Mach(2),Mach(12),Mach(7)]
val MachListProd40 = [Mach(4)]
val MachListProd41 = [Mach(14)]
val MachListProd42 = [Mach(7)]
val MachListProd43 = [Mach(14),Mach(11),Mach(4)]
val MachListProd44 = [Mach(9)]
val MachListProd45 = [Mach(4),Mach(13),Mach(7)]
val MachListProd46 = [Mach(3)]
val MachListProd47 = [Mach(14),Mach(8),Mach(4),Mach(6)]
val MachListProd48 = [Mach(10),Mach(5),Mach(9)]
val MachListProd49 = [Mach(2),Mach(4),Mach(9),Mach(5)]
val MachListProd50 = [Mach(7),Mach(11),Mach(3)]
val MachListProd51 = [Mach(4),Mach(5),Mach(14)]
val MachListProd52 = [Mach(4)]
val MachListProd53 = [Mach(6),Mach(6),Mach(11)]
val MachListProd54 = [Mach(7),Mach(9),Mach(4)]
val MachListProd55 = [Mach(11),Mach(4),Mach(1),Mach(14)]
val MachListProd56 = [Mach(7),Mach(9),Mach(8)]
val MachListProd57 = [Mach(15),Mach(3),Mach(9),Mach(7)]
val MachListProd58 = [Mach(7)]
val MachListProd59 = [Mach(5),Mach(13),Mach(3)]
val MachListProd60 = [Mach(11),Mach(5)]
val MachListProd61 = [Mach(1)]
val MachListProd62 = [Mach(10),Mach(9),Mach(9),Mach(12)]
val MachListProd63 = [Mach(14),Mach(9)]
val MachListProd64 = [Mach(12),Mach(4),Mach(14),Mach(14)]
val MachListProd65 = [Mach(5),Mach(3),Mach(6)]
val MachListProd66 = [Mach(8),Mach(9),Mach(15)]
val MachListProd67 = [Mach(3),Mach(5),Mach(4),Mach(3)]
val MachListProd68 = [Mach(7),Mach(4),Mach(6)]
val MachListProd69 = [Mach(7),Mach(6)]
val MachListProd70 = [Mach(12)]
val MachListProd71 = [Mach(10),Mach(8),Mach(4)]
val MachListProd72 = [Mach(15),Mach(15),Mach(13),Mach(7)]
val MachListProd73 = [Mach(15),Mach(10),Mach(13),Mach(9)]
val MachListProd74 = [Mach(14),Mach(6),Mach(3)]
val MachListProd75 = [Mach(1)]
val MachListProd76 = [Mach(3)]
val MachListProd77 = [Mach(9),Mach(12),Mach(10),Mach(7)]
val MachListProd78 = [Mach(3),Mach(5),Mach(5),Mach(14)]
val MachListProd79 = [Mach(7),Mach(11),Mach(6)]
val MachListProd80 = [Mach(15),Mach(8),Mach(7),Mach(5)]
val MachListProd81 = [Mach(7),Mach(7),Mach(4)]
val MachListProd82 = [Mach(8),Mach(14)]
val MachListProd83 = [Mach(8)]
val MachListProd84 = [Mach(5),Mach(8),Mach(9)]

```

```

val MachListProd85 = [Mach(4),Mach(2)]
val MachListProd86 = [Mach(13),Mach(4),Mach(9),Mach(5)]
val MachListProd87 = [Mach(4)]
val MachListProd88 = [Mach(6),Mach(9)]
val MachListProd89 = [Mach(6),Mach(7),Mach(4)]
val MachListProd90 = [Mach(9),Mach(8),Mach(10),Mach(14)]
val MachListProd91 = [Mach(10)]
val MachListProd92 = [Mach(8)]
val MachListProd93 = [Mach(11),Mach(8)]
val MachListProd94 = [Mach(13)]
val MachListProd95 = [Mach(1),Mach(1),Mach(14)]
val MachListProd96 = [Mach(9),Mach(12)]
val MachListProd97 = [Mach(3),Mach(14)]
val MachListProd98 = [Mach(15)]
val MachListProd99 = [Mach(15),Mach(6),Mach(1)]
val MachListProd100 = [Mach(12),Mach(7),Mach(15)]

in
case ProdT of Prod(1) => List.nth(MachListProd1,OPTlistOPTon-1) |
Prod(2) => List.nth(MachListProd2,OPTlistOPTon-1) |
Prod(3) => List.nth(MachListProd3,OPTlistOPTon-1) |
Prod(4) => List.nth(MachListProd4,OPTlistOPTon-1) |
Prod(5) => List.nth(MachListProd5,OPTlistOPTon-1) |
Prod(6) => List.nth(MachListProd6,OPTlistOPTon-1) |
Prod(7) => List.nth(MachListProd7,OPTlistOPTon-1) |
Prod(8) => List.nth(MachListProd8,OPTlistOPTon-1) |
Prod(9) => List.nth(MachListProd9,OPTlistOPTon-1) |
Prod(10) => List.nth(MachListProd10,OPTlistOPTon-1) |
Prod(11) => List.nth(MachListProd11,OPTlistOPTon-1) |
Prod(12) => List.nth(MachListProd12,OPTlistOPTon-1) |
Prod(13) => List.nth(MachListProd13,OPTlistOPTon-1) |
Prod(14) => List.nth(MachListProd14,OPTlistOPTon-1) |
Prod(15) => List.nth(MachListProd15,OPTlistOPTon-1) |
Prod(16) => List.nth(MachListProd16,OPTlistOPTon-1) |
Prod(17) => List.nth(MachListProd17,OPTlistOPTon-1) |
Prod(18) => List.nth(MachListProd18,OPTlistOPTon-1) |
Prod(19) => List.nth(MachListProd19,OPTlistOPTon-1) |
Prod(20) => List.nth(MachListProd20,OPTlistOPTon-1) |
Prod(21) => List.nth(MachListProd21,OPTlistOPTon-1) |
Prod(22) => List.nth(MachListProd22,OPTlistOPTon-1) |
Prod(23) => List.nth(MachListProd23,OPTlistOPTon-1) |
Prod(24) => List.nth(MachListProd24,OPTlistOPTon-1) |
Prod(25) => List.nth(MachListProd25,OPTlistOPTon-1) |
Prod(26) => List.nth(MachListProd26,OPTlistOPTon-1) |
Prod(27) => List.nth(MachListProd27,OPTlistOPTon-1) |
Prod(28) => List.nth(MachListProd28,OPTlistOPTon-1) |
Prod(29) => List.nth(MachListProd29,OPTlistOPTon-1) |
Prod(30) => List.nth(MachListProd30,OPTlistOPTon-1) |
Prod(31) => List.nth(MachListProd31,OPTlistOPTon-1) |
Prod(32) => List.nth(MachListProd32,OPTlistOPTon-1) |
Prod(33) => List.nth(MachListProd33,OPTlistOPTon-1) |
Prod(34) => List.nth(MachListProd34,OPTlistOPTon-1) |
Prod(35) => List.nth(MachListProd35,OPTlistOPTon-1) |
Prod(36) => List.nth(MachListProd36,OPTlistOPTon-1) |
Prod(37) => List.nth(MachListProd37,OPTlistOPTon-1) |
Prod(38) => List.nth(MachListProd38,OPTlistOPTon-1) |
Prod(39) => List.nth(MachListProd39,OPTlistOPTon-1) |
Prod(40) => List.nth(MachListProd40,OPTlistOPTon-1) |
Prod(41) => List.nth(MachListProd41,OPTlistOPTon-1) |
Prod(42) => List.nth(MachListProd42,OPTlistOPTon-1) |
Prod(43) => List.nth(MachListProd43,OPTlistOPTon-1) |
Prod(44) => List.nth(MachListProd44,OPTlistOPTon-1) |
Prod(45) => List.nth(MachListProd45,OPTlistOPTon-1) |
Prod(46) => List.nth(MachListProd46,OPTlistOPTon-1) |
Prod(47) => List.nth(MachListProd47,OPTlistOPTon-1) |
Prod(48) => List.nth(MachListProd48,OPTlistOPTon-1) |
Prod(49) => List.nth(MachListProd49,OPTlistOPTon-1) |
Prod(50) => List.nth(MachListProd50,OPTlistOPTon-1) |
Prod(51) => List.nth(MachListProd51,OPTlistOPTon-1) |
Prod(52) => List.nth(MachListProd52,OPTlistOPTon-1) |
Prod(53) => List.nth(MachListProd53,OPTlistOPTon-1) |
Prod(54) => List.nth(MachListProd54,OPTlistOPTon-1) |
Prod(55) => List.nth(MachListProd55,OPTlistOPTon-1) |
Prod(56) => List.nth(MachListProd56,OPTlistOPTon-1) |
Prod(57) => List.nth(MachListProd57,OPTlistOPTon-1) |
Prod(58) => List.nth(MachListProd58,OPTlistOPTon-1) |

```

```

Prod(59) => List.nth(MachListProd59,OPTlistOPTon-1) |
Prod(60) => List.nth(MachListProd60,OPTlistOPTon-1) |
Prod(61) => List.nth(MachListProd61,OPTlistOPTon-1) |
Prod(62) => List.nth(MachListProd62,OPTlistOPTon-1) |
Prod(63) => List.nth(MachListProd63,OPTlistOPTon-1) |
Prod(64) => List.nth(MachListProd64,OPTlistOPTon-1) |
Prod(65) => List.nth(MachListProd65,OPTlistOPTon-1) |
Prod(66) => List.nth(MachListProd66,OPTlistOPTon-1) |
Prod(67) => List.nth(MachListProd67,OPTlistOPTon-1) |
Prod(68) => List.nth(MachListProd68,OPTlistOPTon-1) |
Prod(69) => List.nth(MachListProd69,OPTlistOPTon-1) |
Prod(70) => List.nth(MachListProd70,OPTlistOPTon-1) |
Prod(71) => List.nth(MachListProd71,OPTlistOPTon-1) |
Prod(72) => List.nth(MachListProd72,OPTlistOPTon-1) |
Prod(73) => List.nth(MachListProd73,OPTlistOPTon-1) |
Prod(74) => List.nth(MachListProd74,OPTlistOPTon-1) |
Prod(75) => List.nth(MachListProd75,OPTlistOPTon-1) |
Prod(76) => List.nth(MachListProd76,OPTlistOPTon-1) |
Prod(77) => List.nth(MachListProd77,OPTlistOPTon-1) |
Prod(78) => List.nth(MachListProd78,OPTlistOPTon-1) |
Prod(79) => List.nth(MachListProd79,OPTlistOPTon-1) |
Prod(80) => List.nth(MachListProd80,OPTlistOPTon-1) |
Prod(81) => List.nth(MachListProd81,OPTlistOPTon-1) |
Prod(82) => List.nth(MachListProd82,OPTlistOPTon-1) |
Prod(83) => List.nth(MachListProd83,OPTlistOPTon-1) |
Prod(84) => List.nth(MachListProd84,OPTlistOPTon-1) |
Prod(85) => List.nth(MachListProd85,OPTlistOPTon-1) |
Prod(86) => List.nth(MachListProd86,OPTlistOPTon-1) |
Prod(87) => List.nth(MachListProd87,OPTlistOPTon-1) |
Prod(88) => List.nth(MachListProd88,OPTlistOPTon-1) |
Prod(89) => List.nth(MachListProd89,OPTlistOPTon-1) |
Prod(90) => List.nth(MachListProd90,OPTlistOPTon-1) |
Prod(91) => List.nth(MachListProd91,OPTlistOPTon-1) |
Prod(92) => List.nth(MachListProd92,OPTlistOPTon-1) |
Prod(93) => List.nth(MachListProd93,OPTlistOPTon-1) |
Prod(94) => List.nth(MachListProd94,OPTlistOPTon-1) |
Prod(95) => List.nth(MachListProd95,OPTlistOPTon-1) |
Prod(96) => List.nth(MachListProd96,OPTlistOPTon-1) |
Prod(97) => List.nth(MachListProd97,OPTlistOPTon-1) |
Prod(98) => List.nth(MachListProd98,OPTlistOPTon-1) |
Prod(99) => List.nth(MachListProd99,OPTlistOPTon-1) |
Prod(100) => List.nth(MachListProd100,OPTlistOPTon-1) |
end;

fun DefSemiAcab(ProdT,OPTlistOPTon) =
let
  val OPTListProd1 = [OP(15)];
  val OPTListProd2 = [OP(10)]
  val OPTListProd3 = [OP(3),OP(3),OP(14),OP(7)]
  val OPTListProd4 = [OP(3)]
  val OPTListProd5 = [OP(4),OP(6)]
  val OPTListProd6 = [OP(6),OP(7),OP(7)]
  val OPTListProd7 = [OP(13),OP(4)]
  val OPTListProd8 = [OP(8)]
  val OPTListProd9 = [OP(9),OP(10)]
  val OPTListProd10 = [OP(14),OP(6)]
  val OPTListProd11 = [OP(12)]
  val OPTListProd12 = [OP(4),OP(13)]
  val OPTListProd13 = [OP(14)]
  val OPTListProd14 = [OP(12),OP(5),OP(7),OP(4)]
  val OPTListProd15 = [OP(4)]
  val OPTListProd16 = [OP(5),OP(11)]
  val OPTListProd17 = [OP(5),OP(13),OP(4),OP(7)]
  val OPTListProd18 = [OP(1),OP(10)]
  val OPTListProd19 = [OP(6),OP(5),OP(5),OP(10)]
  val OPTListProd20 = [OP(12),OP(8)]
  val OPTListProd21 = [OP(1),OP(14),OP(1)]
  val OPTListProd22 = [OP(3),OP(6),OP(1),OP(11)]
  val OPTListProd23 = [OP(4),OP(2),OP(2)]
  val OPTListProd24 = [OP(9),OP(12)]
  val OPTListProd25 = [OP(6)]
  val OPTListProd26 = [OP(15),OP(2)]
  val OPTListProd27 = [OP(3),OP(15)]
  val OPTListProd28 = [OP(4)]
  val OPTListProd29 = [OP(15),OP(1),OP(12),OP(7)]

```

```

val OPTListProd30 = [OP(15),OP(10)]
val OPTListProd31 = [OP(15),OP(12)]
val OPTListProd32 = [OP(6)]
val OPTListProd33 = [OP(2),OP(5),OP(14),OP(14)]
val OPTListProd34 = [OP(7),OP(6)]
val OPTListProd35 = [OP(11),OP(5),OP(7),OP(3)]
val OPTListProd36 = [OP(1),OP(6),OP(9),OP(9)]
val OPTListProd37 = [OP(7),OP(15)]
val OPTListProd38 = [OP(4),OP(10),OP(1),OP(12)]
val OPTListProd39 = [OP(3),OP(2),OP(12),OP(7)]
val OPTListProd40 = [OP(4)]
val OPTListProd41 = [OP(14)]
val OPTListProd42 = [OP(7)]
val OPTListProd43 = [OP(14),OP(11),OP(4)]
val OPTListProd44 = [OP(9)]
val OPTListProd45 = [OP(4),OP(13),OP(7)]
val OPTListProd46 = [OP(3)]
val OPTListProd47 = [OP(14),OP(8),OP(4),OP(6)]
val OPTListProd48 = [OP(10),OP(5),OP(9)]
val OPTListProd49 = [OP(2),OP(4),OP(9),OP(5)]
val OPTListProd50 = [OP(7),OP(11),OP(3)]
val OPTListProd51 = [OP(4),OP(5),OP(14)]
val OPTListProd52 = [OP(4)]
val OPTListProd53 = [OP(6),OP(6),OP(11)]
val OPTListProd54 = [OP(7),OP(9),OP(4)]
val OPTListProd55 = [OP(11),OP(4),OP(1),OP(14)]
val OPTListProd56 = [OP(7),OP(9),OP(8)]
val OPTListProd57 = [OP(15),OP(3),OP(9),OP(7)]
val OPTListProd58 = [OP(7)]
val OPTListProd59 = [OP(5),OP(13),OP(3)]
val OPTListProd60 = [OP(11),OP(5)]
val OPTListProd61 = [OP(1)]
val OPTListProd62 = [OP(10),OP(9),OP(9),OP(12)]
val OPTListProd63 = [OP(14),OP(9)]
val OPTListProd64 = [OP(12),OP(4),OP(14),OP(14)]
val OPTListProd65 = [OP(5),OP(3),OP(6)]
val OPTListProd66 = [OP(8),OP(9),OP(15)]
val OPTListProd67 = [OP(3),OP(5),OP(4),OP(3)]
val OPTListProd68 = [OP(7),OP(4),OP(6)]
val OPTListProd69 = [OP(7),OP(6)]
val OPTListProd70 = [OP(12)]
val OPTListProd71 = [OP(10),OP(8),OP(4)]
val OPTListProd72 = [OP(15),OP(15),OP(13),OP(7)]
val OPTListProd73 = [OP(15),OP(10),OP(13),OP(9)]
val OPTListProd74 = [OP(14),OP(6),OP(3)]
val OPTListProd75 = [OP(1)]
val OPTListProd76 = [OP(3)]
val OPTListProd77 = [OP(9),OP(12),OP(10),OP(7)]
val OPTListProd78 = [OP(3),OP(5),OP(5),OP(14)]
val OPTListProd79 = [OP(7),OP(11),OP(6)]
val OPTListProd80 = [OP(15),OP(8),OP(7),OP(5)]
val OPTListProd81 = [OP(7),OP(7),OP(4)]
val OPTListProd82 = [OP(8),OP(14)]
val OPTListProd83 = [OP(8)]
val OPTListProd84 = [OP(5),OP(8),OP(9)]
val OPTListProd85 = [OP(4),OP(2)]
val OPTListProd86 = [OP(13),OP(4),OP(9),OP(5)]
val OPTListProd87 = [OP(4)]
val OPTListProd88 = [OP(6),OP(9)]
val OPTListProd89 = [OP(6),OP(7),OP(4)]
val OPTListProd90 = [OP(9),OP(8),OP(10),OP(14)]
val OPTListProd91 = [OP(10)]
val OPTListProd92 = [OP(8)]
val OPTListProd93 = [OP(11),OP(8)]
val OPTListProd94 = [OP(13)]
val OPTListProd95 = [OP(1),OP(1),OP(14)]
val OPTListProd96 = [OP(9),OP(12)]
val OPTListProd97 = [OP(3),OP(14)]
val OPTListProd98 = [OP(15)]
val OPTListProd99 = [OP(15),OP(6),OP(1)]
val OPTListProd100 = [OP(12),OP(7),OP(15)]
in
case ProdT of Prod(1) => if OPTlistOPTon = length(OPTListProd1) then (ProdT,0)
else
(ProdT,OPTlistOPTon) !

```

```

Prod(2) => if OPTlistOPTon = length(OPTListProd2) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(3) => if OPTlistOPTon = length(OPTListProd3) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(4) => if OPTlistOPTon = length(OPTListProd4) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(5) => if OPTlistOPTon = length(OPTListProd5) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(6) => if OPTlistOPTon = length(OPTListProd6) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(7) => if OPTlistOPTon = length(OPTListProd7) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(8) => if OPTlistOPTon = length(OPTListProd8) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(9) => if OPTlistOPTon = length(OPTListProd9) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(10) => if OPTlistOPTon = length(OPTListProd10) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(11) => if OPTlistOPTon = length(OPTListProd11) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(12) => if OPTlistOPTon = length(OPTListProd12) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(13) => if OPTlistOPTon = length(OPTListProd13) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(14) => if OPTlistOPTon = length(OPTListProd14) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(15) => if OPTlistOPTon = length(OPTListProd15) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(16) => if OPTlistOPTon = length(OPTListProd16) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(17) => if OPTlistOPTon = length(OPTListProd17) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(18) => if OPTlistOPTon = length(OPTListProd18) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(19) => if OPTlistOPTon = length(OPTListProd19) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(20) => if OPTlistOPTon = length(OPTListProd20) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(21) => if OPTlistOPTon = length(OPTListProd21) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(22) => if OPTlistOPTon = length(OPTListProd22) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(23) => if OPTlistOPTon = length(OPTListProd23) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(24) => if OPTlistOPTon = length(OPTListProd24) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(25) => if OPTlistOPTon = length(OPTListProd25) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |
Prod(26) => if OPTlistOPTon = length(OPTListProd26) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon) |

```

[illegible]

[illegible]


```

Prod(77) => if OPTlistOPTon = length(OPTListProd77) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(78) => if OPTlistOPTon = length(OPTListProd78) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(79) => if OPTlistOPTon = length(OPTListProd79) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(80) => if OPTlistOPTon = length(OPTListProd80) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(81) => if OPTlistOPTon = length(OPTListProd81) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(82) => if OPTlistOPTon = length(OPTListProd82) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(83) => if OPTlistOPTon = length(OPTListProd83) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(84) => if OPTlistOPTon = length(OPTListProd84) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(85) => if OPTlistOPTon = length(OPTListProd85) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(86) => if OPTlistOPTon = length(OPTListProd86) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(87) => if OPTlistOPTon = length(OPTListProd87) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(88) => if OPTlistOPTon = length(OPTListProd88) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(89) => if OPTlistOPTon = length(OPTListProd89) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(90) => if OPTlistOPTon = length(OPTListProd90) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(91) => if OPTlistOPTon = length(OPTListProd91) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(92) => if OPTlistOPTon = length(OPTListProd92) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(93) => if OPTlistOPTon = length(OPTListProd93) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(94) => if OPTlistOPTon = length(OPTListProd94) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(95) => if OPTlistOPTon = length(OPTListProd95) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(96) => if OPTlistOPTon = length(OPTListProd96) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(97) => if OPTlistOPTon = length(OPTListProd97) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(98) => if OPTlistOPTon = length(OPTListProd98) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(99) => if OPTlistOPTon = length(OPTListProd99) then (ProdT,0)
           else
             (ProdT,OPTlistOPTon) |
Prod(100) => if OPTlistOPTon = length(OPTListProd100) then (ProdT,0)
            else
              (ProdT,OPTlistOPTon)
end;

fun TransfMachINTToInter(McInter) =

```

```

if McInter = 1 then Mach(1)
else if McInter = 2 then Mach(2)
else if McInter = 3 then Mach(3)
else if McInter = 4 then Mach(4)
else if McInter = 5 then Mach(5)
else if McInter = 6 then Mach(6)
else if McInter = 7 then Mach(7)
else if McInter = 8 then Mach(8)
else if McInter = 9 then Mach(9)
else if McInter = 10 then Mach(10)
else if McInter = 11 then Mach(11)
else if McInter = 12 then Mach(12)
else if McInter = 13 then Mach(13)
else if McInter = 14 then Mach(14)
else Mach(15);

fun DefLayoutMachQueue() = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];

fun DefMachQueue(MInQ,MachPos) = List.nth(MInQ,MachPos-1);

fun CalcMinMachQIn(OrdMachInQ) =
let
  fun aux(f,L1) =
    if List.null(L1) then
      f
    else
      if f > hd(L1) then
        aux(hd(L1),tl(L1))
      else
        aux(f,tl(L1))
in
  aux(hd(OrdMachInQ),tl(OrdMachInQ))
end;

fun CalcMaxMachQIn(OrdMachInQ) =
let
  fun aux(f,L1) =
    if List.null(L1) then
      f
    else
      if f < hd(L1) then
        aux(hd(L1),tl(L1))
      else
        aux(f,tl(L1))
in
  aux(hd(OrdMachInQ),tl(OrdMachInQ))
end;

fun InputQ(loc) =
  if loc = Local(2) then 1 else
  if loc = Local(3) then 2 else
  if loc = Local(4) then 3 else
  if loc = Local(5) then 4 else
  if loc = Local(6) then 5 else
  if loc = Local(7) then 6 else
  if loc = Local(8) then 7 else
  if loc = Local(9) then 8 else
  if loc = Local(10) then 9 else
  if loc = Local(11) then 10 else
  if loc = Local(12) then 11 else
  if loc = Local(13) then 12 else
  if loc = Local(14) then 13 else
  if loc = Local(15) then 14 else
  if loc = Local(16) then 15 else
  if loc = Local(17) then 16 else
  if loc = Local(18) then 17 else
  if loc = Local(19) then 18 else
  if loc = Local(20) then 19 else
  if loc = Local(21) then 20 else
  if loc = Local(22) then 21 else
  if loc = Local(23) then 22 else
  if loc = Local(24) then 23 else
  24;

fun TransfINTToMach(McIn) =

```

```

if McIn = 1 then Mach(1)
  else if McIn = 2 then Mach(2)
  else if McIn = 3 then Mach(3)
  else if McIn = 4 then Mach(4)
  else if McIn = 5 then Mach(5)
  else if McIn = 6 then Mach(6)
  else if McIn = 7 then Mach(7)
  else if McIn = 8 then Mach(8)
  else if McIn = 9 then Mach(9)
  else if McIn = 10 then Mach(10)
  else if McIn = 11 then Mach(11)
  else if McIn = 12 then Mach(12)
  else if McIn = 13 then Mach(13)
  else if McIn = 14 then Mach(14)
  else Mach(15);

fun DefWeigthFactorOrder(w) =
let
  val wr = real(w) / 1000.0;
  val TotalManW = real(!ManW) / 1000.0
in
  0.75 + 0.25 * ((TotalManW - wr) / TotalManW)
end;

fun DefMachToolListInit() =
l^(Local(2),[9,4])++
l^(Local(3),[8,3])++
l^(Local(4),[8,3])++
l^(Local(5),[1,7])++
l^(Local(6),[5,6])++
l^(Local(7),[4,1])++
l^(Local(8),[7,3])++
l^(Local(9),[3,2])++
l^(Local(10),[3,2])++
l^(Local(11),[6,4])++
l^(Local(12),[6,7])++
l^(Local(13),[6,8])++
l^(Local(14),[5,4])++
l^(Local(15),[5,7])++
l^(Local(16),[8,7])++
l^(Local(17),[2,3])++
l^(Local(18),[2,9])++
l^(Local(19),[1,8])++
l^(Local(20),[9,5])++
l^(Local(21),[1,8])++
l^(Local(22),[4,2])++
l^(Local(23),[1,6])++
l^(Local(24),[5,1])++
l^(Local(25),[3,8]);

fun DefMachToolListProd(ProdT,OPTlistOPTon) =
let
  val MachToolListProd1 = [6];
  val MachToolListProd2 = [5]
  val MachToolListProd3 = [7,2,7,9]
  val MachToolListProd4 = [1]
  val MachToolListProd5 = [1,4]
  val MachToolListProd6 = [1,5,2]
  val MachToolListProd7 = [6,4]
  val MachToolListProd8 = [8]
  val MachToolListProd9 = [3,6]
  val MachToolListProd10 = [4,6]
  val MachToolListProd11 = [5]
  val MachToolListProd12 = [7,7]
  val MachToolListProd13 = [2]
  val MachToolListProd14 = [3,7,9,1]
  val MachToolListProd15 = [9]
  val MachToolListProd16 = [8,9]
  val MachToolListProd17 = [4,1,1,4]
  val MachToolListProd18 = [2,3]
  val MachToolListProd19 = [5,2,9,3]
  val MachToolListProd20 = [6,9]
  val MachToolListProd21 = [3,7,2]
  val MachToolListProd22 = [4,2,5,6]
  val MachToolListProd23 = [8,5,2]

```

```

val MachToolListProd24 = [2,5]
val MachToolListProd25 = [1]
val MachToolListProd26 = [2,1]
val MachToolListProd27 = [8,2]
val MachToolListProd28 = [7]
val MachToolListProd29 = [5,6,9,6]
val MachToolListProd30 = [7,9]
val MachToolListProd31 = [7,1]
val MachToolListProd32 = [3]
val MachToolListProd33 = [8,8,1,4]
val MachToolListProd34 = [7,4]
val MachToolListProd35 = [7,6,9,9]
val MachToolListProd36 = [6,5,8,2]
val MachToolListProd37 = [5,3]
val MachToolListProd38 = [4,7,6,5]
val MachToolListProd39 = [6,9,8,1]
val MachToolListProd40 = [6]
val MachToolListProd41 = [1]
val MachToolListProd42 = [9]
val MachToolListProd43 = [8,9,2]
val MachToolListProd44 = [8]
val MachToolListProd45 = [1,6,4]
val MachToolListProd46 = [8]
val MachToolListProd47 = [1,8,5,5]
val MachToolListProd48 = [8,1,3]
val MachToolListProd49 = [5,1,8,3]
val MachToolListProd50 = [9,4,8]
val MachToolListProd51 = [1,8,4]
val MachToolListProd52 = [4]
val MachToolListProd53 = [6,4,8]
val MachToolListProd54 = [5,3,4]
val MachToolListProd55 = [6,9,3,5]
val MachToolListProd56 = [8,4,2]
val MachToolListProd57 = [5,3,8,2]
val MachToolListProd58 = [2]
val MachToolListProd59 = [6,8,2]
val MachToolListProd60 = [3,3]
val MachToolListProd61 = [9]
val MachToolListProd62 = [1,1,5,2]
val MachToolListProd63 = [8,5]
val MachToolListProd64 = [7,4,5,1]
val MachToolListProd65 = [6,4,4]
val MachToolListProd66 = [9,7,4]
val MachToolListProd67 = [4,9,4,5]
val MachToolListProd68 = [5,2,9]
val MachToolListProd69 = [6,4]
val MachToolListProd70 = [9]
val MachToolListProd71 = [6,9,1]
val MachToolListProd72 = [5,4,1,1]
val MachToolListProd73 = [4,8,4,3]
val MachToolListProd74 = [2,4,9]
val MachToolListProd75 = [2]
val MachToolListProd76 = [1]
val MachToolListProd77 = [3,2,4,5]
val MachToolListProd78 = [9,9,3,4]
val MachToolListProd79 = [3,4,4]
val MachToolListProd80 = [4,7,2,4]
val MachToolListProd81 = [4,3,9]
val MachToolListProd82 = [5,8]
val MachToolListProd83 = [4]
val MachToolListProd84 = [8,3,8]
val MachToolListProd85 = [6,5]
val MachToolListProd86 = [8,9,7,9]
val MachToolListProd87 = [5]
val MachToolListProd88 = [8,1]
val MachToolListProd89 = [3,5,5]
val MachToolListProd90 = [4,7,2,4]
val MachToolListProd91 = [6]
val MachToolListProd92 = [5]
val MachToolListProd93 = [9,6]
val MachToolListProd94 = [4]
val MachToolListProd95 = [9,1,3]
val MachToolListProd96 = [9,1]
val MachToolListProd97 = [4,2]
val MachToolListProd98 = [6]

```

[illegible]

```

if ProdT = Prod(73) then List.nth(MachToolListProd73,OPTlistOPTon-1) else
if ProdT = Prod(74) then List.nth(MachToolListProd74,OPTlistOPTon-1) else
if ProdT = Prod(75) then List.nth(MachToolListProd75,OPTlistOPTon-1) else
if ProdT = Prod(76) then List.nth(MachToolListProd76,OPTlistOPTon-1) else
if ProdT = Prod(77) then List.nth(MachToolListProd77,OPTlistOPTon-1) else
if ProdT = Prod(78) then List.nth(MachToolListProd78,OPTlistOPTon-1) else
if ProdT = Prod(79) then List.nth(MachToolListProd79,OPTlistOPTon-1) else
if ProdT = Prod(80) then List.nth(MachToolListProd80,OPTlistOPTon-1) else
if ProdT = Prod(81) then List.nth(MachToolListProd81,OPTlistOPTon-1) else
if ProdT = Prod(82) then List.nth(MachToolListProd82,OPTlistOPTon-1) else
if ProdT = Prod(83) then List.nth(MachToolListProd83,OPTlistOPTon-1) else
if ProdT = Prod(84) then List.nth(MachToolListProd84,OPTlistOPTon-1) else
if ProdT = Prod(85) then List.nth(MachToolListProd85,OPTlistOPTon-1) else
if ProdT = Prod(86) then List.nth(MachToolListProd86,OPTlistOPTon-1) else
if ProdT = Prod(87) then List.nth(MachToolListProd87,OPTlistOPTon-1) else
if ProdT = Prod(88) then List.nth(MachToolListProd88,OPTlistOPTon-1) else
if ProdT = Prod(89) then List.nth(MachToolListProd89,OPTlistOPTon-1) else
if ProdT = Prod(90) then List.nth(MachToolListProd90,OPTlistOPTon-1) else
if ProdT = Prod(91) then List.nth(MachToolListProd91,OPTlistOPTon-1) else
if ProdT = Prod(92) then List.nth(MachToolListProd92,OPTlistOPTon-1) else
if ProdT = Prod(93) then List.nth(MachToolListProd93,OPTlistOPTon-1) else
if ProdT = Prod(94) then List.nth(MachToolListProd94,OPTlistOPTon-1) else
if ProdT = Prod(95) then List.nth(MachToolListProd95,OPTlistOPTon-1) else
if ProdT = Prod(96) then List.nth(MachToolListProd96,OPTlistOPTon-1) else
if ProdT = Prod(97) then List.nth(MachToolListProd97,OPTlistOPTon-1) else
if ProdT = Prod(98) then List.nth(MachToolListProd98,OPTlistOPTon-1) else
if ProdT = Prod(99) then List.nth(MachToolListProd99,OPTlistOPTon-1) else
List.nth(MachToolListProd100,OPTlistOPTon-1)
end;

fun DefTranspIniQtd() =
  1^(1,"Yes")++
  1^(2,"Yes")++
  1^(3,"Yes")++
  1^(4,"Yes")++
  1^(5,"Yes")++
  1^(6,"Yes");

fun DefTranspLocationIni() =
  1^((Transp,1),Local(1))++
  1^((Transp,2),Local(1))++
  1^((Transp,3),Local(1))++
  1^((Transp,4),Local(1))++
  1^((Transp,5),Local(1))++
  1^((Transp,6),Local(1));

fun DefInitialTranspFinalLocation() = 1^(Transp,Local(26));

fun DefTranspFinalLocation() = Local(26);

fun DefMachPieceDist(orinfo,pc) =
if pc = 1 then (orinfo,[Mach(15)])
else if pc = 2 then (orinfo,[Mach(10)])
else if pc = 3 then (orinfo,[Mach(3),Mach(3),Mach(14),Mach(7)])
else if pc = 4 then (orinfo,[Mach(3)])
else if pc = 5 then (orinfo,[Mach(4),Mach(6)])
else if pc = 6 then (orinfo,[Mach(6),Mach(7),Mach(7)])
else if pc = 7 then (orinfo,[Mach(13),Mach(4)])
else if pc = 8 then (orinfo,[Mach(8)])
else if pc = 9 then (orinfo,[Mach(9),Mach(10)])
else if pc = 10 then (orinfo,[Mach(14),Mach(6)])
else if pc = 11 then (orinfo,[Mach(12)])
else if pc = 12 then (orinfo,[Mach(4),Mach(13)])
else if pc = 13 then (orinfo,[Mach(14)])
else if pc = 14 then (orinfo,[Mach(12),Mach(5),Mach(7),Mach(4)])
else if pc = 15 then (orinfo,[Mach(4)])
else if pc = 16 then (orinfo,[Mach(5),Mach(11)])
else if pc = 17 then (orinfo,[Mach(5),Mach(13),Mach(4),Mach(7)])
else if pc = 18 then (orinfo,[Mach(1),Mach(10)])
else if pc = 19 then (orinfo,[Mach(6),Mach(5),Mach(5),Mach(10)])
else if pc = 20 then (orinfo,[Mach(12),Mach(8)])
else if pc = 21 then (orinfo,[Mach(1),Mach(14),Mach(1)])
else if pc = 22 then (orinfo,[Mach(3),Mach(6),Mach(1),Mach(11)])
else if pc = 23 then (orinfo,[Mach(4),Mach(2),Mach(2)])
else if pc = 24 then (orinfo,[Mach(9),Mach(12)])

```

```

else if pc = 25 then (orinfo,[Mach(6)])
else if pc = 26 then (orinfo,[Mach(15),Mach(2)])
else if pc = 27 then (orinfo,[Mach(3),Mach(15)])
else if pc = 28 then (orinfo,[Mach(4)])
else if pc = 29 then (orinfo,[Mach(15),Mach(1),Mach(12),Mach(7)])
else if pc = 30 then (orinfo,[Mach(15),Mach(10)])
else if pc = 31 then (orinfo,[Mach(15),Mach(12)])
else if pc = 32 then (orinfo,[Mach(6)])
else if pc = 33 then (orinfo,[Mach(2),Mach(5),Mach(14),Mach(14)])
else if pc = 34 then (orinfo,[Mach(7),Mach(6)])
else if pc = 35 then (orinfo,[Mach(11),Mach(5),Mach(7),Mach(3)])
else if pc = 36 then (orinfo,[Mach(1),Mach(6),Mach(9),Mach(9)])
else if pc = 37 then (orinfo,[Mach(7),Mach(15)])
else if pc = 38 then (orinfo,[Mach(4),Mach(10),Mach(1),Mach(12)])
else if pc = 39 then (orinfo,[Mach(3),Mach(2),Mach(12),Mach(7)])
else if pc = 40 then (orinfo,[Mach(4)])
else if pc = 41 then (orinfo,[Mach(14)])
else if pc = 42 then (orinfo,[Mach(7)])
else if pc = 43 then (orinfo,[Mach(14),Mach(11),Mach(4)])
else if pc = 44 then (orinfo,[Mach(9)])
else if pc = 45 then (orinfo,[Mach(4),Mach(13),Mach(7)])
else if pc = 46 then (orinfo,[Mach(3)])
else if pc = 47 then (orinfo,[Mach(14),Mach(8),Mach(4),Mach(6)])
else if pc = 48 then (orinfo,[Mach(10),Mach(5),Mach(9)])
else if pc = 49 then (orinfo,[Mach(2),Mach(4),Mach(9),Mach(5)])
else if pc = 50 then (orinfo,[Mach(7),Mach(11),Mach(3)])
else if pc = 51 then (orinfo,[Mach(4),Mach(5),Mach(14)])
else if pc = 52 then (orinfo,[Mach(4)])
else if pc = 53 then (orinfo,[Mach(6),Mach(6),Mach(11)])
else if pc = 54 then (orinfo,[Mach(7),Mach(9),Mach(4)])
else if pc = 55 then (orinfo,[Mach(11),Mach(4),Mach(1),Mach(14)])
else if pc = 56 then (orinfo,[Mach(7),Mach(9),Mach(8)])
else if pc = 57 then (orinfo,[Mach(15),Mach(3),Mach(9),Mach(7)])
else if pc = 58 then (orinfo,[Mach(7)])
else if pc = 59 then (orinfo,[Mach(5),Mach(13),Mach(3)])
else if pc = 60 then (orinfo,[Mach(11),Mach(5)])
else if pc = 61 then (orinfo,[Mach(1)])
else if pc = 62 then (orinfo,[Mach(10),Mach(9),Mach(9),Mach(12)])
else if pc = 63 then (orinfo,[Mach(14),Mach(9)])
else if pc = 64 then (orinfo,[Mach(12),Mach(4),Mach(14),Mach(14)])
else if pc = 65 then (orinfo,[Mach(5),Mach(3),Mach(6)])
else if pc = 66 then (orinfo,[Mach(8),Mach(9),Mach(15)])
else if pc = 67 then (orinfo,[Mach(3),Mach(5),Mach(4),Mach(3)])
else if pc = 68 then (orinfo,[Mach(7),Mach(4),Mach(6)])
else if pc = 69 then (orinfo,[Mach(7),Mach(6)])
else if pc = 70 then (orinfo,[Mach(12)])
else if pc = 71 then (orinfo,[Mach(10),Mach(8),Mach(4)])
else if pc = 72 then (orinfo,[Mach(15),Mach(15),Mach(13),Mach(7)])
else if pc = 73 then (orinfo,[Mach(15),Mach(10),Mach(13),Mach(9)])
else if pc = 74 then (orinfo,[Mach(14),Mach(6),Mach(3)])
else if pc = 75 then (orinfo,[Mach(1)])
else if pc = 76 then (orinfo,[Mach(3)])
else if pc = 77 then (orinfo,[Mach(9),Mach(12),Mach(10),Mach(7)])
else if pc = 78 then (orinfo,[Mach(3),Mach(5),Mach(5),Mach(14)])
else if pc = 79 then (orinfo,[Mach(7),Mach(11),Mach(6)])
else if pc = 80 then (orinfo,[Mach(15),Mach(8),Mach(7),Mach(5)])
else if pc = 81 then (orinfo,[Mach(7),Mach(7),Mach(4)])
else if pc = 82 then (orinfo,[Mach(8),Mach(14)])
else if pc = 83 then (orinfo,[Mach(8)])
else if pc = 84 then (orinfo,[Mach(5),Mach(8),Mach(9)])
else if pc = 85 then (orinfo,[Mach(4),Mach(2)])
else if pc = 86 then (orinfo,[Mach(13),Mach(4),Mach(9),Mach(5)])
else if pc = 87 then (orinfo,[Mach(4)])
else if pc = 88 then (orinfo,[Mach(6),Mach(9)])
else if pc = 89 then (orinfo,[Mach(6),Mach(7),Mach(4)])
else if pc = 90 then (orinfo,[Mach(9),Mach(8),Mach(10),Mach(14)])
else if pc = 91 then (orinfo,[Mach(10)])
else if pc = 92 then (orinfo,[Mach(8)])
else if pc = 93 then (orinfo,[Mach(11),Mach(8)])
else if pc = 94 then (orinfo,[Mach(13)])
else if pc = 95 then (orinfo,[Mach(1),Mach(1),Mach(14)])
else if pc = 96 then (orinfo,[Mach(9),Mach(12)])
else if pc = 97 then (orinfo,[Mach(3),Mach(14)])
else if pc = 98 then (orinfo,[Mach(15)])
else if pc = 99 then (orinfo,[Mach(15),Mach(6),Mach(1)])

```

```

else (orinfo,[Mach(12),Mach(7),Mach(15)]);

fun InitLocalProcList() =
  l`("Yes",Local(2))++
  l`("Yes",Local(3))++
  l`("Yes",Local(4))++
  l`("Yes",Local(5))++
  l`("Yes",Local(6))++
  l`("Yes",Local(7))++
  l`("Yes",Local(8))++
  l`("Yes",Local(9))++
  l`("Yes",Local(10))++
  l`("Yes",Local(11))++
  l`("Yes",Local(12))++
  l`("Yes",Local(13))++
  l`("Yes",Local(14))++
  l`("Yes",Local(15))++
  l`("Yes",Local(16))++
  l`("Yes",Local(17))++
  l`("Yes",Local(18))++
  l`("Yes",Local(19))++
  l`("Yes",Local(20))++
  l`("Yes",Local(21))++
  l`("Yes",Local(22))++
  l`("Yes",Local(23))++
  l`("Yes",Local(24))++
  l`("Yes",Local(25));

fun CalcMinListDistIn(CellDist) =
let
  fun aux(f,L1) =
    if List.null(L1) then
      f
    else
      if f > hd(L1) then
        aux(hd(L1),tl(L1))
      else
        aux(f,tl(L1))
  in
    aux(hd(CellDist),tl(CellDist))
  end;

fun DefLocalDist(ListLocal,OPTlistOPTon) = List.nth(ListLocal,OPTlistOPTon-1);

```

Figura A15.1 - Informações contidas no arquivo externo IN_CPN.sml (Fonte: autor)

APÊNDICE 16 IMPRESSÃO NO ARQUIVO EXTERNO DAS 20 AMOSTRAS DO ARRANJO FÍSICO FUNCIONAL (24 MÁQUINAS)

Este apêndice apresenta as 20 amostras impressas ao arquivo externo após a execução do código de programação referente ao procedimento que gera amostras dos arranjos físicos funcionais, conforme a Figura A16.1. Após a impressão dessas amostras, essas então devem ser copiadas ao arquivo IN_CPN.sml, exatamente da forma como está apresentada nessa figura.

```

fun DefLayoutIn(layoutIn) =
let
  val LMList1 = [1,13,4,4,5,5,7,6,6,3,3,11,7,2,8,8,14,14,10,9,9,15,15,12];
  val LMList2 = [12,11,9,9,4,4,2,5,5,10,15,15,8,8,14,14,6,6,7,7,13,3,3,1];
  val LMList3 = [12,8,8,6,6,13,4,11,1,2,3,3,4,9,9,5,5,10,15,15,7,7,14,14];
  val LMList4 = [6,6,2,15,15,5,9,9,14,14,1,5,11,7,7,8,8,13,10,4,4,3,3,12];
  val LMList5 = [14,14,9,9,5,5,3,3,4,4,7,7,11,15,15,10,2,12,13,8,8,6,6,1];
  val LMList6 = [9,9,4,4,6,6,7,12,11,10,14,14,7,13,3,3,5,5,2,8,8,1,15,15];
  val LMList7 = [11,5,5,12,10,9,4,4,1,14,14,9,3,3,7,7,8,8,15,15,6,6,2,13];
  val LMList8 = [3,3,1,2,13,15,8,8,12,4,4,15,9,9,14,14,10,11,7,7,5,5,6,6];
  val LMList9 = [12,11,6,6,10,8,13,1,2,15,15,8,4,4,3,3,7,7,5,5,9,9,14,14];
  val LMList10 = [1,10,5,5,7,7,6,6,13,2,4,4,8,8,14,14,11,12,15,15,3,3,9,9];
  val LMList11 = [1,5,5,14,14,8,9,9,7,7,11,8,3,3,15,15,10,13,2,4,4,6,6,12];
  val LMList12 = [10,1,9,9,12,3,8,8,13,14,14,3,6,6,11,5,5,2,4,4,15,15,7,7];
  val LMList13 = [6,6,2,8,8,9,7,3,3,4,4,9,7,15,15,12,14,14,5,5,10,13,1,11];
  val LMList14 = [13,3,3,6,6,7,10,1,15,15,2,7,5,5,11,8,8,12,4,4,14,14,9,9];
  val LMList15 = [10,1,8,8,13,14,11,15,15,4,4,14,5,5,6,6,7,7,2,3,3,12,9,9];
  val LMList16 = [6,6,14,14,4,4,1,12,9,9,15,15,10,8,8,5,5,3,11,13,7,7,2,3];
  val LMList17 = [8,8,7,7,11,1,9,10,13,4,4,2,9,6,6,14,14,12,15,15,5,5,3,3];
  val LMList18 = [4,4,13,7,7,11,9,2,12,1,15,15,9,10,14,14,5,5,8,8,3,3,6,6];
  val LMList19 = [14,14,7,7,3,3,6,6,5,5,10,2,9,9,13,12,11,4,15,15,8,8,1,4];
  val LMList20 = [8,8,15,15,5,5,12,2,1,11,7,7,3,3,14,14,4,4,6,6,13,10,9,9];

in
  if layoutIn = 1 then LMList1 else
  if layoutIn = 2 then LMList2 else
  if layoutIn = 3 then LMList3 else
  if layoutIn = 4 then LMList4 else
  if layoutIn = 5 then LMList5 else
  if layoutIn = 6 then LMList6 else
  if layoutIn = 7 then LMList7 else
  if layoutIn = 8 then LMList8 else
  if layoutIn = 9 then LMList9 else
  if layoutIn = 10 then LMList10 else
  if layoutIn = 11 then LMList11 else
  if layoutIn = 12 then LMList12 else
  if layoutIn = 13 then LMList13 else
  if layoutIn = 14 then LMList14 else
  if layoutIn = 15 then LMList15 else
  if layoutIn = 16 then LMList16 else
  if layoutIn = 17 then LMList17 else
  if layoutIn = 18 then LMList18 else

```

```

if laytIn = 19 then LMList19 else
LMList20

end;

fun DefLayoutInter(laytInter) =
let
    val LMList1 = [1,13,4,4,5,5,7,6,6,3,3,11,7,2,8,8,14,14,10,9,9,15,15,12];
    val LMList2 = [12,11,9,9,4,4,2,5,5,10,15,15,8,8,14,14,6,6,7,7,13,3,3,1];
    val LMList3 = [12,8,8,6,6,13,4,11,1,2,3,3,4,9,9,5,5,10,15,15,7,7,14,14];
    val LMList4 = [6,6,2,15,15,5,9,9,14,14,1,5,11,7,7,8,8,13,10,4,4,3,3,12];
    val LMList5 = [14,14,9,9,5,5,3,3,4,4,7,7,11,15,15,10,2,12,13,8,8,6,6,1];
    val LMList6 = [9,9,4,4,6,6,7,12,11,10,14,14,7,13,3,3,5,5,2,8,8,1,15,15];
    val LMList7 = [11,5,5,12,10,9,4,4,1,14,14,9,3,3,7,7,8,8,15,15,6,6,2,13];
    val LMList8 = [3,3,1,2,13,15,8,8,12,4,4,15,9,9,14,14,10,11,7,7,5,5,6,6];
    val LMList9 = [12,11,6,6,10,8,13,1,2,15,15,8,4,4,3,3,7,7,5,5,9,9,14,14];
    val LMList10 = [1,10,5,5,7,7,6,6,13,2,4,4,8,8,14,14,11,12,15,15,3,3,9,9];
    val LMList11 = [1,5,5,14,14,8,9,9,7,7,11,8,3,3,15,15,10,13,2,4,4,6,6,12];
    val LMList12 = [10,1,9,9,12,3,8,8,13,14,14,3,6,6,11,5,5,2,4,4,15,15,7,7];
    val LMList13 = [6,6,2,8,8,9,7,3,3,4,4,9,7,15,15,12,14,14,5,5,10,13,1,11];
    val LMList14 = [13,3,3,6,6,7,10,1,15,15,2,7,5,5,11,8,8,12,4,4,14,14,9,9];
    val LMList15 = [10,1,8,8,13,14,11,15,15,4,4,14,5,5,6,6,7,7,2,3,3,12,9,9];
    val LMList16 = [6,6,14,14,4,4,1,12,9,9,15,15,10,8,8,5,5,3,11,13,7,7,2,3];
    val LMList17 = [8,8,7,7,11,1,9,10,13,4,4,2,9,6,6,14,14,12,15,15,5,5,3,3];
    val LMList18 = [4,4,13,7,7,11,9,2,12,1,15,15,9,10,14,14,5,5,8,8,3,3,6,6];
    val LMList19 = [14,14,7,7,3,3,6,6,5,5,10,2,9,9,13,12,11,4,15,15,8,8,1,4];
    val LMList20 = [8,8,15,15,5,5,12,2,1,11,7,7,3,3,14,14,4,4,6,6,13,10,9,9];

in

if laytInter = 1 then LMList1 else
if laytInter = 2 then LMList2 else
if laytInter = 3 then LMList3 else
if laytInter = 4 then LMList4 else
if laytInter = 5 then LMList5 else
if laytInter = 6 then LMList6 else
if laytInter = 7 then LMList7 else
if laytInter = 8 then LMList8 else
if laytInter = 9 then LMList9 else
if laytInter = 10 then LMList10 else
if laytInter = 11 then LMList11 else
if laytInter = 12 then LMList12 else
if laytInter = 13 then LMList13 else
if laytInter = 14 then LMList14 else
if laytInter = 15 then LMList15 else
if laytInter = 16 then LMList16 else
if laytInter = 17 then LMList17 else
if laytInter = 18 then LMList18 else
if laytInter = 19 then LMList19 else
LMList20

end;

```

Figura A16.1 - Informações gravadas no arquivo externo referentes ao número de amostras do arranjo físico funcional (Fonte: autor)

APÊNDICE 17 IMPRESSÃO NO ARQUIVO EXTERNO DAS 20 AMOSTRAS DO ARRANJO FÍSICO ALEATORIAMENTE DISTRIBUÍDO (24 MÁQUINAS)

Da mesma forma que o apêndice anterior, este também apresenta as informações contidas no arquivo externo após a execução do código de programação que gera amostras de arranjo físico, porém agora as amostras geradas são do arranjo físico aleatoriamente distribuído. Essas informações sobre as vinte amostras podem ser vistas na Figura A17.1. Após a geração dessas informações no arquivo externo, estas também devem ser copiadas ao arquivo externo IN_CPN.sml.

```

fun DefLayoutIn(layoutIn) =
let
    val LMList1 = [1,13,4,5,11,5,3,6,7,2,8,14,6,12,3,7,4,8,14,15,9,10,9,15];
    val LMList2 = [10,15,4,11,5,2,12,8,14,8,9,15,6,1,3,3,14,9,13,5,6,7,4,7];
    val LMList3 = [9,9,14,11,2,8,7,1,14,13,5,12,8,10,3,4,15,6,15,5,7,6,3,4];
    val LMList4 = [13,9,2,10,14,9,5,7,4,3,11,15,12,3,4,14,7,15,1,6,6,8,5,8];
    val LMList5 = [7,1,3,11,5,6,9,4,3,5,2,12,10,9,14,4,14,7,8,13,8,6,15,15];
    val LMList6 = [2,4,12,15,8,1,9,13,14,8,10,15,11,14,6,5,4,6,7,3,7,3,9,5];
    val LMList7 = [1,10,5,7,4,2,13,6,5,8,14,11,7,4,6,14,12,9,3,15,8,9,3,15];
    val LMList8 = [8,3,5,10,13,8,3,7,5,12,6,4,1,14,11,2,9,14,6,9,7,15,4,15];
    val LMList9 = [8,9,6,6,12,14,11,9,8,14,4,1,15,15,4,13,2,7,10,5,3,7,3,5];
    val LMList10 = [13,2,11,7,8,8,15,5,15,5,12,7,6,6,9,3,1,3,9,14,10,14,4,4];
    val LMList11 = [15,11,5,14,4,8,6,13,10,1,7,4,8,9,6,9,12,3,14,7,15,2,5,3];
    val LMList12 = [3,15,2,6,3,7,8,6,5,5,13,4,9,1,4,12,7,11,8,10,9,14,14,15];
    val LMList13 = [4,13,7,11,15,7,1,12,2,9,10,14,5,14,5,4,6,3,6,3,8,9,8,15];
    val LMList14 = [8,15,5,7,11,8,1,2,12,3,14,4,9,3,10,15,4,13,5,14,6,7,6,9];
    val LMList15 = [6,11,9,6,5,2,5,12,9,1,8,4,8,14,7,10,4,7,15,13,14,3,3,15];
    val LMList16 = [1,2,15,5,15,4,5,12,6,13,6,4,14,9,7,11,14,9,8,3,10,7,8,3];
    val LMList17 = [7,3,3,15,1,2,14,13,14,4,10,11,12,8,5,7,6,15,9,6,5,8,4,9];
    val LMList18 = [1,5,15,5,10,15,2,9,12,6,13,6,9,7,8,14,3,8,3,11,7,14,4,4];
    val LMList19 = [14,7,6,11,2,1,8,14,3,15,4,10,9,6,7,12,9,8,4,5,15,5,3,13];
    val LMList20 = [6,1,15,5,15,7,7,10,2,9,13,6,4,11,8,8,14,12,3,5,3,9,14,4];

in
    if layoutIn = 1 then LMList1 else
    if layoutIn = 2 then LMList2 else
    if layoutIn = 3 then LMList3 else
    if layoutIn = 4 then LMList4 else
    if layoutIn = 5 then LMList5 else
    if layoutIn = 6 then LMList6 else
    if layoutIn = 7 then LMList7 else
    if layoutIn = 8 then LMList8 else
    if layoutIn = 9 then LMList9 else
    if layoutIn = 10 then LMList10 else
    if layoutIn = 11 then LMList11 else
    if layoutIn = 12 then LMList12 else
    if layoutIn = 13 then LMList13 else
    if layoutIn = 14 then LMList14 else
    if layoutIn = 15 then LMList15 else
    if layoutIn = 16 then LMList16 else
    if layoutIn = 17 then LMList17 else
    if layoutIn = 18 then LMList18 else

```

```

if laytIn = 19 then LMList19 else
LMList20

end;

fun DefLayoutInter(laytInter) =
let
    val LMList1 = [1,13,4,5,11,5,3,6,7,2,8,14,6,12,3,7,4,8,14,15,9,10,9,15];
    val LMList2 = [10,15,4,11,5,2,12,8,14,8,9,15,6,1,3,3,14,9,13,5,6,7,4,7]
    val LMList3 = [9,9,14,11,2,8,7,1,14,13,5,12,8,10,3,4,15,6,15,5,7,6,3,4]
    val LMList4 = [13,9,2,10,14,9,5,7,4,3,11,15,12,3,4,14,7,15,1,6,6,8,5,8]
    val LMList5 = [7,1,3,11,5,6,9,4,3,5,2,12,10,9,14,4,14,7,8,13,8,6,15,15]
    val LMList6 = [2,4,12,15,8,1,9,13,14,8,10,15,11,14,6,5,4,6,7,3,7,3,9,5]
    val LMList7 = [1,10,5,7,4,2,13,6,5,8,14,11,7,4,6,14,12,9,3,15,8,9,3,15]
    val LMList8 = [8,3,5,10,13,8,3,7,5,12,6,4,1,14,11,2,9,14,6,9,7,15,4,15]
    val LMList9 = [8,9,6,6,12,14,11,9,8,14,4,1,15,15,4,13,2,7,10,5,3,7,3,5]
    val LMList10 = [13,2,11,7,8,8,15,5,15,5,12,7,6,6,9,3,1,3,9,14,10,14,4,4]
    val LMList11 = [15,11,5,14,4,8,6,13,10,1,7,4,8,9,6,9,12,3,14,7,15,2,5,3]
    val LMList12 = [3,15,2,6,3,7,8,6,5,5,13,4,9,1,4,12,7,11,8,10,9,14,14,15]
    val LMList13 = [4,13,7,11,15,7,1,12,2,9,10,14,5,14,5,4,6,3,6,3,8,9,8,15]
    val LMList14 = [8,15,5,7,11,8,1,2,12,3,14,4,9,3,10,15,4,13,5,14,6,7,6,9]
    val LMList15 = [6,11,9,6,5,2,5,12,9,1,8,4,8,14,7,10,4,7,15,13,14,3,3,15]
    val LMList16 = [1,2,15,5,15,4,5,12,6,13,6,4,14,9,7,11,14,9,8,3,10,7,8,3]
    val LMList17 = [7,3,3,15,1,2,14,13,14,4,10,11,12,8,5,7,6,15,9,6,5,8,4,9]
    val LMList18 = [1,5,15,5,10,15,2,9,12,6,13,6,9,7,8,14,3,8,3,11,7,14,4,4]
    val LMList19 = [14,7,6,11,2,1,8,14,3,15,4,10,9,6,7,12,9,8,4,5,15,5,3,13]
    val LMList20 = [6,1,15,5,15,7,7,10,2,9,13,6,4,11,8,8,14,12,3,5,3,9,14,4]

in

if laytInter = 1 then LMList1 else
if laytInter = 2 then LMList2 else
if laytInter = 3 then LMList3 else
if laytInter = 4 then LMList4 else
if laytInter = 5 then LMList5 else
if laytInter = 6 then LMList6 else
if laytInter = 7 then LMList7 else
if laytInter = 8 then LMList8 else
if laytInter = 9 then LMList9 else
if laytInter = 10 then LMList10 else
if laytInter = 11 then LMList11 else
if laytInter = 12 then LMList12 else
if laytInter = 13 then LMList13 else
if laytInter = 14 then LMList14 else
if laytInter = 15 then LMList15 else
if laytInter = 16 then LMList16 else
if laytInter = 17 then LMList17 else
if laytInter = 18 then LMList18 else
if laytInter = 19 then LMList19 else
LMList20

end;

```

Figura A17.1 - Informações gravadas no arquivo externo referentes ao número de amostras do arranjo físico aleatoriamente distribuído (Fonte: autor)

APÊNDICE 18 IMPRESSÃO NO ARQUIVO EXTERNO DAS 20 AMOSTRAS DO ARRANJO FÍSICO PARCIALMENTE DISTRIBUÍDO (24 MÁQUINAS)

Este apêndice apresenta também as amostras de arranjos físicos geradas através do uso do código de programação referente ao procedimento proposto. Agora, este apresenta as 20 amostras geradas pelo código de programação para obter os arranjos físicos parcialmente distribuídos. Para a obtenção das amostras, foi escolhido um grau de desagregação, ou seja, dois grupos do mesmo tipo de máquina espalhadas no chão de fábrica.

De acordo com o dimensionamento das réplicas de cada tipo de máquina, no qual resultou em 24 máquinas (cujas réplicas de cada tipo de máquina são 1 1 2 2 2 2 2 2 1 1 1 1 2 2), significa que a máquina tipo 1 possui uma réplica, a máquina tipo 2 possui uma réplica, a máquina tipo 3 possui duas réplicas e assim por diante. Como o grau de desagregação escolhido foi um (dois grupos do mesmo tipo de máquina), é de se esperar que existam dois grupos do mesmo tipo de máquina, sendo que em cada grupo existe apenas uma réplica, como é o caso da máquina tipo 3, máquina tipo 4 etc. Isso pode ser constatado na Figura A18.1 para cada arranjo físico parcialmente distribuído das 20 amostras geradas.

Esta Figura A18.1 ilustra as representações de vinte amostras gravadas em um arquivo externo obtidas com a execução do código do procedimento referente à geração do arranjo físico parcialmente distribuído. Estas informações contidas neste arquivo externo devem, agora, ser inseridas em um arquivo IN_CPN.sml para serem usadas pelo modelo de simulação.

```

fun DefLayoutIn(layoutIn) =
let
  val LMList1 = [1,13,4,5,11,3,12,14,8,2,7,6,15,9,10,9,4,15,7,3,6,14,8,5];
  val LMList2 = [12,8,6,13,3,2,10,5,9,4,11,1,14,7,15,6,15,5,4,3,8,7,9,14];
  val LMList3 = [14,9,5,7,4,3,1,12,2,10,15,11,6,8,13,9,4,6,8,15,5,3,7,14];
  val LMList4 = [11,5,12,10,9,14,13,8,7,3,4,1,2,6,15,3,15,4,7,5,6,14,9,8];
  val LMList5 = [12,11,6,10,8,15,7,3,4,13,1,2,14,9,5,5,7,4,15,3,9,14,8,6];
  val LMList6 = [1,5,14,8,11,7,12,13,10,15,3,9,6,4,2,9,3,14,4,15,7,5,6,8];
  val LMList7 = [6,2,8,9,4,3,1,11,14,12,15,7,13,10,5,3,6,7,4,14,9,8,5,15];
  val LMList8 = [10,1,8,13,14,4,9,7,6,5,11,15,12,3,2,6,14,4,7,3,5,8,9,15];
  val LMList9 = [8,7,11,1,2,4,12,14,6,9,10,13,3,5,15,4,7,15,8,3,6,5,14,9];
  val LMList10 = [14,7,3,2,10,5,4,11,12,13,9,6,1,8,15,8,15,5,6,9,4,14,3,7];
  val LMList11 = [14,8,7,12,2,11,1,6,15,4,13,5,10,9,3,14,3,15,8,7,9,6,4,5];
  val LMList12 = [5,4,7,11,10,8,1,3,14,15,13,12,2,6,9,6,5,14,7,9,3,4,15,8];
  val LMList13 = [8,12,1,2,13,9,7,11,10,15,3,14,4,6,5,7,8,14,9,6,4,3,5,15];
  val LMList14 = [7,2,13,6,4,11,3,12,14,10,15,8,5,9,1,6,7,8,3,9,15,4,14,5];
  val LMList15 = [15,2,9,12,5,6,8,14,4,10,7,13,3,11,1,14,6,7,5,9,4,3,15,8];
  val LMList16 = [11,2,10,12,5,3,4,7,8,6,14,9,1,13,15,8,15,7,5,14,9,6,3,4];
  val LMList17 = [7,9,11,6,12,1,5,2,13,15,3,10,8,4,14,3,4,6,7,5,8,14,9,15];
  val LMList18 = [11,4,13,2,7,1,14,8,6,9,15,5,3,10,12,8,3,9,15,14,7,5,4,6];
  val LMList19 = [4,11,13,5,10,9,14,15,6,3,7,2,12,8,1,6,4,3,9,8,15,14,7,5];
  val LMList20 = [9,2,6,13,11,15,10,4,8,3,12,5,1,7,14,8,7,6,14,5,4,15,9,3];

in

if layoutIn = 1 then LMList1 else

```

```

if laytIn = 2 then LMList2 else
if laytIn = 3 then LMList3 else
if laytIn = 4 then LMList4 else
if laytIn = 5 then LMList5 else
if laytIn = 6 then LMList6 else
if laytIn = 7 then LMList7 else
if laytIn = 8 then LMList8 else
if laytIn = 9 then LMList9 else
if laytIn = 10 then LMList10 else
if laytIn = 11 then LMList11 else
if laytIn = 12 then LMList12 else
if laytIn = 13 then LMList13 else
if laytIn = 14 then LMList14 else
if laytIn = 15 then LMList15 else
if laytIn = 16 then LMList16 else
if laytIn = 17 then LMList17 else
if laytIn = 18 then LMList18 else
if laytIn = 19 then LMList19 else
LMList20

end;

fun DefLayoutInter(laytInter) =
let
  val LMList1 = [1,13,4,5,11,3,12,14,8,2,7,6,15,9,10,9,4,15,7,3,6,14,8,5];
  val LMList2 = [12,8,6,13,3,2,10,5,9,4,11,1,14,7,15,6,15,5,4,3,8,7,9,14]
  val LMList3 = [14,9,5,7,4,3,1,12,2,10,15,11,6,8,13,9,4,6,8,15,5,3,7,14]
  val LMList4 = [11,5,12,10,9,14,13,8,7,3,4,1,2,6,15,3,15,4,7,5,6,14,9,8]
  val LMList5 = [12,11,6,10,8,15,7,3,4,13,1,2,14,9,5,5,7,4,15,3,9,14,8,6]
  val LMList6 = [1,5,14,8,11,7,12,13,10,15,3,9,6,4,2,9,3,14,4,15,7,5,6,8]
  val LMList7 = [6,2,8,9,4,3,1,11,14,12,15,7,13,10,5,3,6,7,4,14,9,8,5,15]
  val LMList8 = [10,1,8,13,14,4,9,7,6,5,11,15,12,3,2,6,14,4,7,3,5,8,9,15]
  val LMList9 = [8,7,11,1,2,4,12,14,6,9,10,13,3,5,15,4,7,15,8,3,6,5,14,9]
  val LMList10 = [14,7,3,2,10,5,4,11,12,13,9,6,1,8,15,8,15,5,6,9,4,14,3,7]
  val LMList11 = [14,8,7,12,2,11,1,6,15,4,13,5,10,9,3,14,3,15,8,7,9,6,4,5]
  val LMList12 = [5,4,7,11,10,8,1,3,14,15,13,12,2,6,9,6,5,14,7,9,3,4,15,8]
  val LMList13 = [8,12,1,2,13,9,7,11,10,15,3,14,4,6,5,7,8,14,9,6,4,3,5,15]
  val LMList14 = [7,2,13,6,4,11,3,12,14,10,15,8,5,9,1,6,7,8,3,9,15,4,14,5]
  val LMList15 = [15,2,9,12,5,6,8,14,4,10,7,13,3,11,1,14,6,7,5,9,4,3,15,8]
  val LMList16 = [11,2,10,12,5,3,4,7,8,6,14,9,1,13,15,8,15,7,5,14,9,6,3,4]
  val LMList17 = [7,9,11,6,12,1,5,2,13,15,3,10,8,4,14,3,4,6,7,5,8,14,9,15]
  val LMList18 = [11,4,13,2,7,1,14,8,6,9,15,5,3,10,12,8,3,9,15,14,7,5,4,6]
  val LMList19 = [4,11,13,5,10,9,14,15,6,3,7,2,12,8,1,6,4,3,9,8,15,14,7,5]
  val LMList20 = [9,2,6,13,11,15,10,4,8,3,12,5,1,7,14,8,7,6,14,5,4,15,9,3]

in

if laytInter = 1 then LMList1 else
if laytInter = 2 then LMList2 else
if laytInter = 3 then LMList3 else
if laytInter = 4 then LMList4 else
if laytInter = 5 then LMList5 else
if laytInter = 6 then LMList6 else
if laytInter = 7 then LMList7 else
if laytInter = 8 then LMList8 else
if laytInter = 9 then LMList9 else
if laytInter = 10 then LMList10 else

```

```
if laytInter = 11 then LMList11 else  
if laytInter = 12 then LMList12 else  
if laytInter = 13 then LMList13 else  
if laytInter = 14 then LMList14 else  
if laytInter = 15 then LMList15 else  
if laytInter = 16 then LMList16 else  
if laytInter = 17 then LMList17 else  
if laytInter = 18 then LMList18 else  
if laytInter = 19 then LMList19 else  
LMList20  
  
end;
```

Figura A18.1 - Informações gravadas no arquivo externo referentes ao número de amostras do arranjo físico parcialmente distribuído (Fonte: autor)

APÊNDICE 19 IMPRESSÃO NO ARQUIVO EXTERNO DA AMOSTRA DO ARRANJO FÍSICO FUNCIONAL (24 MÁQUINAS)

A execução do código de programação referente ao procedimento que gera arranjo físico maximamente distribuído também permite gravar a amostra obtida em um arquivo externo, em txt, conforme mostrada na Figura A19.1. As informações gravadas neste arquivo externo, agora, devem ser inseridas no arquivo IN_CPN.sml para que o modelo de simulação faça a leitura.

```
fun DefLayoutIn(laytIn) = [3,9,10,4,5,6,13,12,11,1,15,8,7,14,8,2,9,3,15,5,4,6,7,14];  
fun DefLayoutInter(laytInter) = [3,9,10,4,5,6,13,12,11,1,15,8,7,14,8,2,9,3,15,5,4,6,7,14];
```

Figura A19.1 - Impressão do arranjo físico maximamente distribuído em arquivo externo (Fonte: autor)

No caso do arranjo físico maximamente distribuído, é necessário gerar apenas uma amostra deste. Isso porque a ideia do arranjo físico maximamente distribuído é escolher uma amostra de arranjo físico que possuir o menor grau de distribuição dentre todas as que foram geradas, por isso é que existe apenas uma amostra. Em virtude disso, não é necessário gerar uma estrutura do tipo fun-let-in-if-end ao arquivo externo.

APÊNDICE 20 ESPALHAMENTO DO TRANSPORTADOR INTERMEDIÁRIO

Este apêndice tem por finalidade apresentar as informações contidas no arquivo `TranspInterInitFunc.txt`, as quais também foram geradas pelo código de programação do procedimento proposto.

Em virtude deste estudo adotar apenas um único transportador intermediário, então após a execução do código de programação, apenas a informação “1`(Transp,Local(14))” é impressa, conforme mostrada na Figura A20.1. Esta informação indica que o transportador intermediário está alocado na posição 13, daí Local(14).

Caso houvessem dois transportadores intermediários alocados, sendo que este segundo transportador intermediário estivesse alocado na posição 4, por exemplo, então no arquivo externo, tem-se na primeira linha “1`(Transp,Local(14));” e na segunda linha “1`(Transp,Local(5))”.

Figura A20.1 - Arquivo externo informando a posição alocada do transportador intermediário (Fonte: autor)

APÊNDICE 21 ALGORITMO GENÉTICO

É algoritmo de busca baseado em mecanismos de seleção natural. Indivíduos mais resistentes terão maiores chances de herdarem seus genes a fim de gerar indivíduos filhos. Esta ferramenta permite explorar, eficientemente, um vasto campo de informações e identificar quais são os genes que potencialmente são capazes em fornecer melhorias na solução.

Goldberg (1989) comenta que existem alguns operadores genéticos que produzem, na maioria dos problemas do cotidiano, bons resultados. Dentre eles, são:

- Reprodução;
- *Crossover*;
- Mutação.

Antes de implementar os operadores, é necessário conhecer os indivíduos da população inicial. Estes indivíduos são os possíveis pais para geração de uma nova população de filhos, mas ainda não se sabe qual deles realizará a reprodução, porque depende da resistência de cada um deles. Conhecendo-se os genes, é possível obter a resistência de cada um. O nome “resistência” é chamado, em inglês, de *fitness*. Quanto melhor a resistência, maior a chance de herdar os genes resistentes a próxima geração. Melhor não quer dizer necessariamente que seja o de maior valor. Pode ser também o de menor valor, dependendo do problema. Por exemplo, em muitos problemas de engenharia deseja-se obter, em sua função objetivo, resultado maximizado, tais como maximização de lucro, etc, ou também de minimização, como de tempo, de locomoção e de redução de custos. Tal função objetivo é, na realidade, o próprio *fitness*.

Considere o exemplo da Tabela A21.1. A população é constituída de quatro indivíduos (possíveis pais), denominados de 1 a 4, e na segunda coluna as suas respectivas seqüências de genes (*string*), representadas por códigos binários 0 e 1.

Tabela A21.1 - Exemplo de indivíduos de uma população com suas respectivas seqüências genéticas(string) e resistências (fitness) (Fonte: Goldberg, 1989)

No.	<i>String</i>	Fitness	% of Total
1	01101	169	14,4
2	11000	576	49,2
3	01000	64	5,5
4	10011	361	30,9
Total		1170	100,0

Pela tabela, note os valores em % de cada indivíduo. Quanto maior a porcentagem, maior a chance de ser selecionado para geração de novos indivíduos.

O mecanismo para a seleção de indivíduos da população apresentado na Tabela A21.1 é similar ao de roleta russa. A roleta russa é na realidade um círculo, cujos cada setor circular (fatia do círculo) representa a porcentagem de cada indivíduo, ver Figura A21.1. Ao rodar a roleta, indivíduos de maior porcentagem têm maiores chances de serem selecionados pelo ponto de referência, e assim eles participam da reprodução para gerarem novos indivíduos.

Figura A21.1 - Ilustração do método roleta russa de seleção (Fonte: Adaptado de Goldberg, 1989)

Considere que os indivíduos selecionados sejam 11000 e 10011. Deve-se, agora, realizar o procedimento de cruzamento (crossover) para a obtenção de indivíduos filhos. Antes de iniciar o processo de cruzamento, deve-se ressaltar que esta seleção dos indivíduos pais para reprodução não significa necessariamente que ambos irão ter “atração entre si” garantindo que a reprodução ocorra. A seleção dos indivíduos significa apenas em escolher os indivíduos mais resistentes. Existe, portanto, uma probabilidade a ser considerada para a ocorrência do cruzamento (ou seja, os indivíduos devem se sentir atraídos entre si). Comumente, esta probabilidade de ocorrência de cruzamento é definida por uma taxa, cujo valor gira em torno de 0.8 ou 80%. Isso quer dizer que se fosse 100%, significaria que sempre ocorrerão cruzamentos entre o casal de pais. Enfim, considerando que a reprodução ocorra, então os genes são devem ser transmitidos aos filhos. A quantidade dos genes do casal a serem transmitidos aos filhos são definidos durante o cruzamento.

A Figura A21.2 mostra um exemplo em que dois indivíduos filhos foram gerados. O primeiro filho recebeu três genes do pai e dois genes da mãe, enquanto o segundo filho, recebeu dois genes do pai e três genes da mãe.

Figura A21.2 – Exemplo de cruzamento entre os indivíduos pais para transmissão de genes aos indivíduos filhos (Fonte: autor)

Com a obtenção dos indivíduos filhos, a próxima etapa considera a possibilidade destes de sofrerem mutações. O procedimento referente ao operador mutação consiste em alterar os genes dos indivíduos filhos. A alteração não pode ser realizada para todos os genes, pois isso resultaria em um indivíduo totalmente diferente (um novo ser). Por isso, deve-se considerar pequenas probabilidades de ocorrência, para cada gene, sobre a possibilidade de ocorrer ou não a mutação. Geralmente, esta probabilidade, denominada de taxa de mutação, gira em torno de 0.02 ou 2%. De acordo com o exemplo da Figura A21.2, a alteração dos genes consiste em modificar 0 por 1 ou 1 por 0. Em alguns casos, o procedimento de mutação pode também significar alterar genes de posição.

Conforme mostradas na Figura A21.3 (que é continuação da Figura A21.2), note que ocorreram mutações no quinto gene do filho 1 e mutação no terceiro gene do filho 2.

Figura A21.3 – Exemplo de mutação nos indivíduos filhos (Fonte: autor)

Com este roteiro sobre os operadores genéticos, é possível realizar a implementação usando-se códigos de programação.