O problema de minimização de trocas de ferramentas

Andreza Cristina Beezão Moreira

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Andreza Cristina Beezão Moreira

O problema de minimização de trocas de ferramentas

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutora em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Horacio Hideki Yanasse

Coorientador: Profa. Dra. Franklina Maria Bragion de Toledo

USP – São Carlos
Novembro de 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

M634o Moreira, Andreza Cristina Beezão
O problema de minimização de trocas de ferramentas
/ Andreza Cristina Beezão Moreira; orientador Horacio
Hideki Yanasse; coorientador Franklina Maria Bragion
de Toledo. - São Carlos - SP, 2016.
104 p.

Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional)
- Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2016.

1. Troca de ferramentas. 2. Sistema flexível
de manufatura. 3. Redução de domínio. 4. Adaptive
large neighborhood search. 5. *Job shop scheduling
problem*. I. Yanasse, Horacio Hideki, orient. II.
Toledo, Franklina Maria Bragion de, coorient. III.
Título.

Andreza Cristina Beezão Moreira

The minimization of tool switches problem

Doctoral dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Horacio Hideki Yanasse

Co-advisor: Profa. Dra. Franklina Maria Bragion de Toledo

USP – São Carlos
November 2016

*Ao Mayron, meu carinho e
minha gratidão.*

Agradecimentos

Li, algum dia, em algum lugar que não me lembro, que as palavras *agradecida* e *grata* carregam graças, são leves e singelas. Utilizarei, portanto, estas palavras cheias de significados para agradecer aos muitos que contribuíram, direta ou indiretamente, durante esta minha jornada.

A Deus, por me ensinar a perceber Sua presença e seguir em frente, sempre.

Aos meus pais, Pedro e Zilda, pelo amor e apoio constantes, mesmo a milhares de quilômetros de distância. Esta realização também pertence a vocês. Aos meus irmãos, Adriana e Adilson, e aos meus cunhados, Rogério e Natasha, pelo carinho e pela esperança de que tudo terminaria bem. À minha família mineira, pelo amor, pela dedicação e por todas as orações. Às minhas sobrinhas, Gabriela, Mariana, Júlia e Lara, por encherem de luz os meus dias.

Ao Professor Horacio Yanasse, pela orientação e pelas oportunidades profissionais decorrentes deste projeto de doutorado.

À Professora Franklina Toledo, pela orientação e pela amizade. Talvez sem que ela percebesse, seus dizeres “Que bom, tudo está caminhando bem!” eram um grande incentivo para mim.

Aos Professores Gilbert Laporte e Jean-François Cordeau, pela recepção, pela orientação e pelo cuidado, tão perceptível, durante o período em que estive trabalhando na Université de Montréal, Canadá.

Ao Professor Alysson Costa, por ter me incentivado à aventura do doutorado-sanduíche. Ele me dizia muito sobre a cidade, o CIRRELT e as pessoas. Mesmo assim, me surpreendi, positivamente.

Aos demais membros da comissão julgadora, Profa. Dra. Débora Ronconi, Prof. Dr. Sílvio Araújo, Prof. Dr. Marco Antônio de Carvalho e Prof. Dr. Reinaldo Morabito, pelas valiosas sugestões.

Contei também com a generosidade de pessoas queridas. Graças a Deus são muitas, e faço questão de lembrá-las aqui. Aos amigos de São Carlos: Adriana, Alcebíades, Alex, Ana Carolina, Apoenã, Bruna, Camila Peverari, Camila Ruiz, Carlos, Daiane, Felipe Louza, Flávio, Ginnara, Gisele, Natália, Nelson, Patrícia, Thaís, Vanessa, Vinícius, Dona Wilma e Sr. Luiz. Ao pessoal do Laboratório de Otimização (LOt), em especial, à Aline e ao Everton. Por fim, às amizades feitas em Montréal e que resistem à distância: David, Gisa, Larissa, M. Bruno e Serge. A todos vocês, agradeço pelos momentos de descontração, de união e de apoio.

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo fomento. À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa concedida nos três primeiros meses dessa pesquisa.

À USP e, mais particularmente, ao ICMC, por todo suporte fornecido durante os últimos onze anos. Tive a oportunidade de uma formação privilegiada.

As minhas últimas palavras são para o Mayron. Pretendo, um dia, retribuir a paciência, a generosidade e as palavras de incentivo que me dedicou. No início, você me ajudou com coisas

práticas, como relembrar os detalhes de uma linguagem de programação. No fim, me ensinou a garra e a determinação realmente fundamentais para a conclusão de um doutorado. A você, a minha mais sincera e eterna gratidão.

*“Caminante, no hay camino,
se hace camino al andar.”
(Antônio Machado)*

RESUMO

MOREIRA, A. C. B. **O problema de minimização de trocas de ferramentas.** 2016. 104 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2016.

Especialmente nas últimas quatro décadas, muitos estudos se voltaram às variáveis determinantes para a implementação efetiva de sistemas flexíveis de manufatura, tais como seu *design*, sequenciamento e controle. Neste ínterim, o manejo apropriado do conjunto de ferramentas necessárias para a fabricação de um respectivo lote de produtos foi destacado como fator crucial no desempenho do sistema de produção como um todo. Neste trabalho, abordamos a otimização do número de inserções e remoções de ferramentas no magazine de uma ou mais máquinas numericamente controladas, admitindo-se que uma parcela significativa do tempo de produção é dispensada com estas trocas de ferramentas.

De forma mais precisa, a minimização do número de trocas de ferramentas consiste em determinar a ordem de processamento de um conjunto de tarefas, bem como o carregamento ótimo do(s) compartimento(s) de ferramentas da(s) máquina(s), a fim de que o número de trocas seja minimizado. Como demonstrado na literatura, mesmo o caso restrito à existência de apenas uma máquina de manufatura (MTSP, do inglês *Minimization of Tool Switches Problem*) é um problema NP-difícil, o que pode justificar o fato observado de que a maioria dos métodos de solução existentes o abordam de maneira heurística. Consequentemente, concluímos que a extensão ao contexto de múltiplas máquinas é também um problema NP-difícil, intrinsecamente complicado de se resolver.

Nosso objetivo consiste em estudar formas eficientes de otimizar o número de trocas de ferramentas em ambientes equipados com máquinas flexíveis de manufatura. Para tanto, abordamos o problema básico, MTSP, e duas de suas variantes, em níveis crescentes de abrangência, que consideram o sequenciamento de tarefas em um conjunto de: (i) máquinas paralelas e idênticas (IPMTC, do inglês *Identical Parallel Machines problem with Tooling Constraints*); e (ii) máquinas paralelas e idênticas inseridas em um ambiente do tipo *job shop* (JSSPTC, do inglês *Job Shop Scheduling Problem with Tooling Constraints*).

Classificamos as principais contribuições desta tese com respeito a três aspectos. Primeiramente, empurramos as fronteiras da literatura do MTSP propondo formulações matemáticas para os problemas IPMTC e JSSPTC. Desenvolvemos, também, algoritmos baseados em diferentes técnicas de resolução, como redução de domínio, *Path relinking*, *Adaptive large neighborhood search* e a elaboração de regras de despacho. Por último, com o intuito de bem avaliar a eficiência e o alcance de nossos métodos, propomos três novos conjuntos de instâncias teste. Acreditamos, assim, que este trabalho contribui positivamente com pesquisas futuras em um cenário abrangente

dentro da minimização das trocas de ferramentas em um sistema flexível de manufatura.

Palavras-chave: Troca de ferramentas, Sistema flexível de manufatura, Redução de domínio, Adaptive large neighborhood search, *Job shop scheduling problem*.

ABSTRACT

MOREIRA, A. C. B. **The minimization of tool switches problem**. 2016. 104 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2016.

Several studies, especially in the last four decades, have focused on decisive elements for the effective implementation of flexible manufacturing systems, such as their design, scheduling and control. In the meantime, the appropriate management of the set of tools needed to manufacture a certain lot of products has been highlighted as a crucial factor in the performance of the production system as a whole. This work deals with the optimization of the number of insertions and removals from the magazine of one or more numerical controlled machines, assuming that a significant part of the production time is wasted with such tool switches.

More precisely, the minimization of tool switches problem (MTSP) consists on determining the processing order of a set of jobs, as well as the optimal loading of the magazine(s) of the machine(s), so that the total number of switches is minimized. As formally demonstrated in the literature, the MTSP is a NP-hard problem even when considering the existence of only one manufacturing machine, which could justify the fact that most of the solution methods tackles it heuristically. We thus conclude that its extension to the case of multiples machines is also NP-hard and, therefore, a problem intrinsically difficult to solve.

Our goal consists in studying efficient ways to optimize the number of tool switches in environments equipped with flexible manufacturing machines. For that, we address the basic problem, MTSP, and two MTSP variants, in increasing levels of reach, that consider the job sequencing in a set of: (i) identical parallel machines (*Identical Parallel Machines problem with Tooling Constraints, IPMTC*); and (ii) identical parallel machines inserted in a job shop environment (*Job Shop Scheduling Problem with Tooling Constraints, JSSPTC*).

The main contributions of this thesis are classified according three aspects. First, we pushed the frontier of the MTSP literature by proposing mathematical formulations for IPMTC and JSSPTC. We also developed algorithms based on different solution techniques, such as domain reduction, Path Relinking, Adaptive Large Neighborhood Search and dispatching rules. Finally, to fully evaluate the effectiveness and limits of our methods, three new sets of benchmark instances were generated. We believe that this work contributes positively to the future of research in a broad scenario inside the minimization of tool switches in flexible manufacturing systems.

Keywords: Tool switches, Flexible manufacturing systems, Domain reduction, Adaptive large neighborhood search, Job shop scheduling problem..

Lista de Figuras

1.1	Sistema flexível de manufatura (Fonte: Zeballos (2010)).	1
2.1	Exemplos de ferramentas e magazine utilizados em uma máquina de manufatura (Fonte: da Cunha Jr. (2012)).	7
2.2	Uma solução para a instância-teste.	8
2.3	Solução ótima para a instância-teste.	8
2.4	Contagem de trocas pelo modelo TD - Solução ótima para a instância-teste.	9
2.5	Contagem de trocas modificada - Solução ótima para a instância-teste.	13
2.6	Contagem do <i>makespan</i> - Uma solução para a instância-teste.	17
2.7	Contagem do <i>makespan</i> - Solução ótima para a instância-teste.	17
2.8	Solução $\mathcal{O}_1$ para a instância-teste: ótima para o JSSP e factível para o JSSPTC. . .	18
2.9	Solução $\mathcal{O}_2$ para a instância-teste: factível para o JSSP e ótima para o JSSPTC. . .	19
4.1	Transformação GTSP inclusivo - GTSP exclusivo.	34
4.2	Transformação GTSP exclusivo - TSP.	34
4.3	Uma solução para a instância-teste.	35
4.4	Solução ótima para a instância-teste.	35
4.5	Distância $h = 1$	38
4.6	Distância h incrementada	38
4.7	Transformação ATSP - TSP simétrico.	39
4.8	Completamento de nós - Parte 1	43
4.9	Completamento de nós - Parte 2	43
4.10	Completamento de nós - Parte 3	43
4.11	Exemplo para aplicação do algoritmo <i>path relinking</i>	46
4.12	Funcionamento do algoritmo <i>path relinking</i>	47
5.1	Processo de remoção e inserção de tarefas.	62
5.2	Remoção de tarefas consideradas críticas.	64
5.3	Ilustração do funcionamento do operador “Remoção - Grafo de vizinhança”.	65
5.4	Ilustração do funcionamento do operador “Remoção - Grafo de requisições”.	65
5.5	Comportamento dos métodos ALNS-B e ALNS-R para um problema com 4 máquinas e 25 tarefas (instância 1262 do conjunto IPMTC-I).	70
5.6	Comportamento dos métodos ALNS-B e ALNS-R para um problema com 4 máquinas e 25 tarefas (instância 1301 do conjunto IPMTC-I).	70
6.1	Grafo do JSSP para instância com 3 máquinas e 3 tarefas.	81

6.2	Solução factível, conforme o caminho crítico do grafo JSSP.	82
6.3	Grafo do JSSPTC para instância com 3 máquinas e 3 tarefas.	82
6.4	Comparação do desempenho das regras de despacho - Heurística Construtiva Determinística.	89
6.5	Comparação do desempenho das regras de despacho - Heurística Construtiva Estocástica.	89

Lista de Tabelas

2.1	Notações	6
2.2	Instância-teste do MTSP.	7
2.3	Exemplo-teste - Laporte <i>et al.</i> (2004).	11
2.4	Exemplo de bloco-0.	14
2.5	Instância-teste - IPMTC.	16
2.6	Instância-teste - JSSPTC.	18
2.7	Plano de processamento - JSSPTC.	18
4.1	Instância-teste do MTSP - Estratégia reversa.	42
4.2	Características dos grupos de instâncias.	48
4.3	Resultados relativos à RD1.	48
4.4	Resultados da geração de nós - RD2	48
4.5	Resultados relativos ao Grupo A - RD2.	49
4.6	Resultados relativos ao Grupo B - RD2.	49
4.7	Resultados relativos ao Grupo C - RD2.	49
4.8	Resultados relativos ao Grupo D - RD2.	50
4.9	Resultados relativos ao Grupo E - RD2.	50
4.10	Resultados da geração de nós - RD3.	51
4.11	Resultados relativos ao Grupo A - RD3.	51
4.12	Resultados relativos ao Grupo B - RD3.	52
4.13	Resultados relativos ao Grupo C - RD3.	52
4.14	Resultados relativos ao Grupo D - RD3.	52
4.15	Resultados relativos ao Grupo E - RD3.	52
4.16	Resultados relativos ao Grupo A - RD4.	53
4.17	Resultados relativos ao Grupo B - RD4.	53
4.18	Resultados relativos ao Grupo C - RD4.	54
4.19	Resultados relativos ao Grupo D - RD4.	54
4.20	Resultados relativos ao Grupo E - RD4.	54
5.1	Combinação de parâmetros analisados na fase de calibração da ALNS.	67
5.2	Valores finais dos parâmetros da ALNS.	67
5.3	Análise comparativa do desempenho das formulações matemáticas	68
5.4	<i>Gap</i> relativo à melhor solução encontrada considerando os métodos CLIP1, CLIP2, ALNS-B e ALNS-R - Cenário 1.	68

5.5	<i>Gap</i> relativo à melhor solução encontrada considerando os métodos CLIP1, CLIP2, ALNS-B e ALNS-R - Cenário 4.	69
5.6	Comparação do desempenho de todos os métodos e cenários analisados.	69
5.7	Estatísticas dos operadores de remoção.	71
5.8	Estatísticas dos operadores de inserção.	71
5.9	Desempenho da ALNS-B - Cenário 4.	72
5.10	<i>Gap</i> relativo à melhor solução encontrada considerando os métodos CLIP1, CLIP2 e ALNS-R - Cenário 4.	72
6.1	Análise do desempenho da formulação matemática.	85
6.2	Análise comparativa do desempenho das versões determinística e estocástica da heurística construtiva CH.	87
6.3	Análise comparativa do desempenho das versões determinística e estocástica da busca local BL.	88
6.4	Análise comparativa do desempenho das versões finais, determinística e estocástica, da heurística construtiva CH combinada ao <i>Path relinking</i> PR.	88

Sumário

Lista de Figuras	xvii
Lista de Tabelas	xix
1 Introdução	1
1.1 Organização da tese	3
2 Descrição formal dos problemas estudados	5
2.1 Notações	5
2.2 MTSP	5
2.2.1 Formulações matemáticas	8
2.2.1.1 Formulação de Tang e Denardo (1988a)	8
2.2.1.2 Formulação de Laporte <i>et al.</i> (2004)	10
2.2.1.3 Aplicação das melhorias de Laporte <i>et al.</i> (2004) ao modelo de Tang e Denardo (1988)	12
2.2.1.4 Formulação de Catanzaro <i>et al.</i> (2015)	14
2.3 IPMTC	15
2.4 JSSPTC	17
3 Revisão bibliográfica	21
3.1 Trocas de ferramentas em uma única máquina - MTSP	22
3.1.1 Variantes do MTSP	25
3.1.2 Problemas correlatos e aplicações	27
3.2 O contexto de máquinas paralelas	28
3.2.1 PM	28
3.2.2 Trocas de ferramentas em máquinas paralelas - IPMTC	29
3.3 O contexto do ambiente <i>job shop</i>	30
3.3.1 JSSP	30
3.3.2 Trocas de ferramentas no ambiente <i>job shop</i> - JSSPTC	32
4 MTSP	33
4.1 O MTSP visto como um GTSP	33
4.1.1 Redução de domínio	36
4.1.1.1 Estrutura geral	36
4.1.1.2 Estratégia “gulosa”	39
4.1.1.3 Estratégia reversa	39

4.1.1.4	Estratégia de aproximação dos nós isolados	43
4.1.1.5	<i>Path relinking</i>	45
4.2	Experimentos computacionais	46
4.2.1	Instâncias teste	47
4.2.2	Experimento 1: estratégia “gulosa”	47
4.2.3	Experimento 2: estratégia reversa	48
4.2.4	Experimento 3: estratégia de aproximação dos nós isolados	51
4.2.5	Experimento 4: <i>Path relinking</i>	53
4.3	Conclusões	55
5	IPMTC	57
5.1	Formulações matemáticas	57
5.2	Meta-heurística ALNS	60
5.2.1	Solução inicial	60
5.2.2	Operadores de remoção e inserção	62
5.2.2.1	Remoção randômica	62
5.2.2.2	Remoção “Shaw”	63
5.2.2.3	Remoção de tarefa crítica	63
5.2.2.4	Remoção de maior desvio	63
5.2.2.5	Remoção randômica de máquina	63
5.2.2.6	Remoção pseudo-randômica de máquina	63
5.2.2.7	Remoção de máquina crítica	64
5.2.2.8	Remoção - Grafo de vizinhança	64
5.2.2.9	Remoção - Grafo de requisições	64
5.2.2.10	Inserção gulosa _p	64
5.2.2.11	Inserção gulosa _Δ	65
5.2.2.12	Inserção gulosa _p - <i>Look-ahead</i>	65
5.2.2.13	Inserção gulosa _Δ - <i>Look-ahead</i>	65
5.2.2.14	Inserção <i>2-Regret</i>	66
5.3	Experimentos computacionais	66
5.3.1	Instâncias teste	66
5.3.2	Etapa de calibração da ALNS	66
5.3.3	Resultados	67
5.3.3.1	Experimento 1: formulações matemáticas	67
5.3.3.2	Experimento 2: ALNS no conjunto de instâncias IPMTC-I	68
5.3.3.3	Experimento 3: ALNS no conjunto de instâncias IPMTC-II	71
5.4	Conclusões	73
6	JSSPTC	75
6.1	Formulação matemática	75
6.1.1	Minimização do número de trocas de ferramentas	75
6.2	Heurística construtiva	77
6.2.1	Solução inicial	77
6.2.1.1	Regras de despacho	79

6.2.1.2	Algoritmo do caminho crítico	81
6.3	Heurística de pós-otimização	83
6.3.1	Busca local (BL)	83
6.3.2	<i>Path relinking</i>	84
6.4	Experimentos computacionais	84
6.4.1	Instâncias teste	84
6.4.2	Resultados	85
6.4.2.1	Experimento 1: formulação matemática	85
6.4.2.2	Experimento 2: heurística construtiva	86
6.4.2.3	Experimento 3: heurística construtiva pós-otimizada	87
6.5	Conclusões	89
7	Conclusões	91
7.1	Perspectivas de trabalho	92
	Referências Bibliográficas	93

Capítulo 1

Introdução

O constante avanço tecnológico em todos os setores que envolvem o ambiente de produção, bem como a competitividade característica de um mundo globalizado, exigem das empresas de manufatura um grau cada vez mais elevado de flexibilidade e adaptatividade diante das incertezas do mercado.

Dizia Henry Ford em meados do início do século XX: “O cliente pode ter um automóvel pintado em qualquer cor, contanto que seja preta” (Ford, 1922). Pois, o aperfeiçoamento das linhas de montagem culminou no modelo de produção denominado *fordista* - inicialmente implementado nas indústrias automobilísticas da companhia Ford - cujo objetivo principal era a produção em larga escala de produtos altamente padronizados. Durante cerca de seis décadas, o *fordismo* se consolidou também nos setores industriais têxtil, siderúrgico e energético.

A partir de 1970, porém, a crise econômica advinda da Segunda Guerra Mundial (1939-1945) exigia um modelo de produção ágil e diversificado, requisitos estes que marcaram o declínio *fordista*, devido à rigidez do seu modelo de gestão industrial. Nesse ínterim, surgiram nas indústrias japonesas os primeiros sinais do *toyotismo*, um novo modelo de produção cujo propósito era prover um tipo de automação flexível e baseada na fabricação de lotes menores de uma variedade maior de produtos.

Constituíram-se, a partir de então, os sistemas flexíveis de manufatura (SFM), descritos por Zeballos (2010) como uma rede de máquinas flexíveis, interligadas por um sistema automático de manuseio de materiais e capazes de executar um grande número de tarefas, desde que as ferramentas necessárias para tal sejam administradas e alocadas corretamente. Na Figura 1.1, vemos uma representação esquemática de um SFM composto por três máquinas, magazines com capacidade para cinco ferramentas, um sistema de manuseio de materiais e *buffers* em cada uma das máquinas, bem como para armazenar peças pré-processadas e finalizadas.

Figura 1.1: Sistema flexível de manufatura (Fonte: Zeballos (2010)).

Segundo Amaya *et al.* (2008), “um sistema flexível de manufatura incorpora versatilidade e eficiência à produção em massa”. Por outro lado, os custos envolvidos com sua implementação

são altos, de modo que o ideal é que “um sistema produtivo seja tão flexível quanto necessário e tão rígido quanto possível” (Morales, 2009). Para isso, contudo, é essencial o manejo correto do conjunto de ferramentas requeridas pelas tarefas a serem sequenciadas, um “desafio” diretamente relacionado com a performance do sistema como um todo. Na década de 80, Bard (1988) já constata que alguns tipos de máquinas numericamente controladas exigem ajustes durante as trocas de ferramentas e que esta parcela de tempo de processamento da máquina influencia de forma significativa o tempo total de processamento das tarefas. De fato, segundo Cumings (1986), Tomek (1986) e Ayres (1988), o deslocamento gerado pela movimentação de ferramentas entre o depósito e o magazine da máquina representa entre 25% e 30% dos custos fixos e variáveis em um SFM. Sabendo, portanto, que o processo de trocas de ferramentas consome mais tempo que qualquer outra atividade de *setup* (Hop e Nagarur, 2004a) e que estas trocas são inevitáveis na prática dos SFM, a falta de solidez na gestão do conjunto de ferramentas “pode afetar diversas áreas centrais no processo de produção, tais como *design* de produtos, carregamento de máquinas, capacidade de sequenciamento, decisões de roteamento em tempo real” e custos de produção. Com base nisto, um sistema flexível de manufatura deve considerar a integração entre o manejo das ferramentas e características intrínsecas do sistema em questão, como seu formato, controle e planejamento (Gray *et al.*, 1993).

Visando, portanto, contribuir com o funcionamento eficaz dos SFM, abordamos a temática da otimização do número de trocas de ferramentas. Para tanto, estudamos primeiramente a minimização das trocas de ferramentas em uma única máquina (MTSP), também conhecido na literatura como SSP (do inglês, *Job sequencing and tool switching problem*), ToSP (do inglês, *Tool switching problem*) e TSFM (do inglês, *Tool switching on a flexible machine*), seguindo para o contexto de múltiplas máquinas paralelas e idênticas (IPMTC) e finalizando com a contextualização das trocas de ferramentas no ambiente de sequenciamento *job shop* (JSSPTC).

Mais precisamente, consideramos um conjunto de tarefas a serem sequenciadas em uma ou mais máquina(s) flexível(is) e numericamente controlada(s) de manufatura. Cada tarefa requer que um ou mais conjunto(s) de ferramentas esteja(m) alocado(s) no(s) magazine(s) da(s) máquina(s) no momento do seu processamento. Este(s) magazine(s), por sua vez, comporta(m) um número limitado de ferramentas e nenhuma tarefa requer um número de ferramentas maior que tal capacidade. Como o total de ferramentas necessárias para a execução de todas as tarefas é, na prática, maior que a capacidade da máquina, trocas devem ser efetuadas ao longo do processo de produção. Objetivamos, portanto, determinar uma sequência de processamento das tarefas na(s) máquina(s) e uma sequência associada referente ao carregamento do(s) compartimento(s) de ferramentas, de modo que o número total de trocas de ferramentas entre as tarefas seja minimizado. Os primeiros registros industriais sobre aplicações do MTSP ocorreram em metalúrgicas e indústrias eletrônicas, enquanto problemas relacionados ao IPMTC podem ser encontrados em indústrias têxteis e de papel.

Oerlemans (1992) e Crama *et al.* (1994) demonstraram formalmente que mesmo o MTSP é um problema NP-difícil, o que nos permite concluir, também, a dificuldade de resolução exata das variantes IPMTC e JSSPTC. Fixando uma sequência de processamento das tarefas, contudo, o MTSP pode ser resolvido em tempo polinomial por meio da política KTNS (do inglês, *Keep Tool Needed Soonest*), devida a Bard (1988) e Tang e Denardo (1988a). Afirmando a dificuldade na resolução da minimização do número de trocas de ferramentas em sistemas flexíveis de manufatura, a literatura expõe muitos trabalhos que a abordam por meio de métodos heurísticos, contrapondo a escassez de formulações matemáticas e algoritmos que garantam a otimalidade de suas soluções.

O objeto de nosso estudo está inserido no seguinte contexto. O(s) magazine(s) da(s) máquina(s) comporta(m) qualquer combinação de ferramentas, desde que sua(s) capacidade(s) seja(m) respeitada(s). Suporemos, também, que todas as ferramentas possuem o mesmo peso e ocupam individualmente uma única e irrelevante posição em uma máquina. O tempo necessário para remover e inserir cada ferramenta é constante, e igual para todas as ferramentas do conjunto, além de significativo quando comparado ao tempo total de processamento de todas as tarefas. Ainda, consideraremos que a(s) máquina(s) é(são) capaz(es) de realizar apenas uma troca por vez. Com efeito,

estas últimas suposições justificam a investigação do número de trocas de ferramentas, ao invés do número de instantes de troca.

Neste trabalho, propomos métodos computacionais mais eficientes na otimização do número de trocas de ferramentas em sistemas flexíveis de manufatura, abordando o MTSP e duas de suas variantes, as quais consideram diferentes disposições do conjunto de máquinas e características distintas quanto à existência de regras de precedência com relação às tarefas a serem sequenciadas. Nosso objetivo é fortalecer a ideia de que o manejo eficaz das ferramentas é parte crucial no sucesso da implementação dos SFM.

1.1 Organização da tese

Nesta seção, apresentamos a organização textual desta tese, enumerando os principais aspectos abordados em cada capítulo e as contribuições mais significativas de cada um deles.

No Capítulo 2, descrevemos formalmente o MTSP, o IPMTC e o JSSPTC, além de apresentar, comparar e discutir as três principais formulações inteiras conhecidas na literatura do MTSP. O Capítulo 3 traz uma revisão bibliográfica dos problemas estudados, abrangendo suas principais extensões e aplicações, assim como problemas correlatos. Os capítulos restantes contêm a metodologia proposta e todos os experimentos computacionais realizados.

O Capítulo 4 apresenta os algoritmos desenvolvidos para a resolução do MTSP. São detalhados os passos da redução de domínio baseada em transformações do MTSP em extensões do problema do caixeiro viajante, além de estratégias aplicadas com o intuito de suavizar aspectos críticos das soluções encontradas pela heurística Lin-Kernighan.

O Capítulo 5 é altamente baseado no artigo “*Scheduling identical parallel machines with tooling constraints*”, de autoria de Andreza Cristina Beezão, Jean-François Cordeau, Gilbert Laporte e Horacio Hideki Yanasse, aceito para publicação no *European Journal of Operational Research* e disponível *online* desde 11 de agosto de 2016. Como contribuições, citamos a elaboração de duas formulações matemáticas para o IPMTC e uma meta-heurística ALNS (do inglês, *Adaptive large neighborhood search*) para sua resolução, a qual supera os algoritmos encontrados na literatura. Além disso, geramos dois novos conjuntos de instâncias teste contendo até 200 tarefas, 10 máquinas e 40 ferramentas.

O Capítulo 6 apresenta uma formulação matemática para o JSSPTC e um novo conjunto de instâncias teste. Como estratégia de solução, propomos uma heurística construtiva cuja solução inicial é determinada pela aplicação de regras de despacho, com escolhas determinísticas ou estocásticas, e uma busca local, desenvolvida para aperfeiçoar os resultados obtidos.

Por fim, as conclusões dos estudos realizados e perspectivas para pesquisas futuras são sumariadas no Capítulo 7.

Capítulo 2

Descrição formal dos problemas estudados

Neste capítulo, apresentamos uma descrição formal do MTSP e de suas variantes IPMTC e JSSPTC. Com esta finalidade, na Seção 2.1 introduzimos algumas notações relacionadas aos problemas estudados. A Seção 2.2 detalha o contexto do MTSP, além de comparar e discutir as três principais modelagens matemáticas já propostas em sua literatura. Adiante, nas Seções 2.3 e 2.4, focamos na definição e contextualização do IPMTC e do JSSPTC. Os mesmos são descritos analiticamente por meio de formulações matemáticas nos Capítulos 5 e 6, respectivamente.

2.1 Notações

A fim de unificar as notações utilizadas no decorrer deste trabalho, apresentamos, na Tabela 2.1, um conjunto de “notações-padrão”.

2.2 MTSP

Analiticamente, o MTSP tem como objetivo o sequenciamento de um conjunto de tarefas $J = \{1, \dots, n\}$ em uma única máquina flexível de manufatura, cujo magazine (ou compartimento de ferramentas) comporta, no máximo, C ferramentas. Para que uma tarefa j seja processada, as ferramentas pertencentes ao conjunto T_j devem estar alocadas na máquina. Dessa forma, como o número l de ferramentas necessárias para a realização de todas as tarefas é geralmente maior que C , trocas devem ser efetuadas ao longo do processo de produção. Destacamos que o problema torna-se trivial quando $l \leq C$.

Para retratar de forma mais específica as circunstâncias em que estudamos o MTSP, consideramos as seguintes suposições, conforme explicitaram Bard (1988), Gómez e Lorena (1998) e Zhou *et al.* (2005):

- as tarefas são conhecidas *a priori*;
- as ferramentas requeridas para cada tarefa são previamente estabelecidas e fixadas;
- nenhuma tarefa requer mais que C ferramentas (*condição de factibilidade*);
- o sistema produtivo não admite *loops*;
- a máquina realiza apenas uma troca de ferramenta por vez;
- o magazine da máquina comporta qualquer combinação do conjunto T de ferramentas, desde que sua capacidade C seja respeitada;

Tabela 2.1: *Notações*

$\bar{t}$	tempo de troca
C	capacidade da máquina
Δ	<i>makespan</i>
Δ_m	tempo total de processamento na máquina m
n	número de tarefas
p	número de máquinas
l	número de ferramentas
i, j	índices para tarefas
k	índice para posições no sequenciamento de tarefas
m	índice para máquinas
t	índice para ferramentas
u	índice para operações
J	conjunto de tarefas
J'	conjunto de tarefas, incluindo uma tarefa fictícia 0
M	conjunto de máquinas
T	conjunto de ferramentas
K	conjunto de posições no sequenciamento das tarefas
T_j	conjunto de ferramentas requeridas pela tarefa j
T_{jm}	conjunto de ferramentas requeridas pela tarefa j na máquina m
J_t	conjunto de tarefas que requerem a ferramenta t
J_C	conjunto de tarefas que requerem exatamente C ferramentas
σ^j	conjunto de operações da tarefa j
σ_u^j	máquina na qual a u -ésima operação da tarefa j deve ser processada
p_j	tempo de processamento da tarefa j
p_{jm}	tempo de processamento da tarefa j na máquina m
$\tau(u_1, u_2)$	número de trocas de ferramentas entre o processamento das operações u_1 e u_2
A^c	conjunto complementar de um conjunto genérico, A
V^t	matriz tranposta de uma matriz genérica, V

- todas as ferramentas possuem o mesmo peso e ocupam individualmente uma única e irrelevante posição na máquina;
- o tempo $\bar{t}$ necessário para inserir e remover cada ferramenta é constante, e o mesmo para todas as ferramentas do conjunto T ;
- as ferramentas não se quebram nem se desgastam.

A escolha pela minimização do número de trocas de ferramentas é adequada neste contexto em que o tempo de troca de ferramentas é proporcional ao número de trocas a serem realizadas. Entretanto, se a máquina for capaz de realizar simultaneamente todas as trocas necessárias entre pares de tarefas sequenciadas consecutivamente, o critério de minimização do número de instantes de troca é mais apropriado, como abordado nos trabalhos de Tang e Denardo (1988b), Widmer (1991), Blazewicz e Finke (1994), Denizel (2003) e Konak *et al.* (2008).

Tang e Denardo (1988a) demonstraram que, quando $C \geq 2$, o MTSP pertence à classe dos problemas NP-difíceis. A prova baseou-se na redução do MTSP a um problema do caixeiro viajante (TSP, do inglês *Traveling Salesman Problem*). Uma análise mais formal e completa acerca da complexidade do MTSP foi realizada por Oerlemans (1992) e Crama *et al.* (1994).

Observemos que o MTSP pode ser naturalmente decomposto em dois problemas “independentes”:

- (i) *sequenciamento*: determinar uma sequência ótima para o processamento das tarefas;

Figura 2.1: Exemplos de ferramentas e magazine utilizados em uma máquina de manufatura (Fonte: da Cunha Jr. (2012)).

(ii) *alocação*: para uma dada sequência de processamento das tarefas, determinar os instantes em que as ferramentas devem ser alocadas no magazine da máquina, de modo a minimizar o número de trocas de ferramentas.

Ao tomarmos uma solução arbitrária para o problema (i), o problema (ii) pode ser resolvido em tempo polinomial - mais precisamente, em $\mathcal{O}(nl)$ operações - por meio da política KTNS (do inglês, *Keep tool needed soonest*) proposta por Tang e Denardo (1988a). Esta política estabelece que uma ferramenta seja inserida somente quando requerida para o processamento da próxima tarefa. Além do mais, quando trocas de ferramentas forem necessárias, “as ferramentas requeridas por uma próxima tarefa devem ser as primeiras a serem mantidas na máquina” (Chaves et al., 2012).

Ao supormos que cada tarefa sequenciada não exige mais do que C ferramentas, garantimos a factibilidade do MTSP. Buscamos, portanto, o menor número possível de trocas de ferramentas diante de todos os sequenciamentos viáveis para o conjunto de tarefas. Para ilustrarmos o problema, consideremos a instância-teste apresentada na Tabela 2.2, em que $n = 5$, $l = 8$ e $C = 5$.

Tabela 2.2: Instância-teste do MTSP.

Tarefa	Ferramentas requeridas
1	A, B, C
2	A, C, D, E, F
3	A, E
4	B, C, G
5	D, E, F, G, H

Adotaremos o seguinte padrão para as figuras deste capítulo: o retângulo externo na cor azul se refere ao magazine da máquina, os espaços coloridos na cor salmão indicam que houveram trocas de ferramentas e os nomes de ferramentas em negrito retratam as ferramentas requeridas para o processamento da tarefa em questão.

Na Figura 2.2, a sequência $2 \rightarrow 3 \rightarrow 1 \rightarrow 5 \rightarrow 4$ resulta em 6 trocas de ferramentas. Por outro lado, a solução ótima representada pela ordem $5 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 4$ realiza apenas 4 trocas, como retratado na Figura 2.3.

Façamos, agora, uma breve análise dos *limitantes inferiores* propostos na literatura do MTSP. Trivialmente, um primeiro limitante é dado por $(l - C)$, uma vez que todas as l ferramentas devem ser alocadas entre os C slots da máquina ao menos uma vez. Laporte et al. (2004) propõem

Figura 2.2: Uma solução para a instância-teste.

Figura 2.3: Solução ótima para a instância-teste.

mais duas formas para calcular tal limitante, ambas utilizadas no desenvolvimento de dois algoritmos exatos. Mais tarde, Yanasse e Rodrigues (2007) sugerem um novo limitante através da resolução de exemplares do MTSP em que somente um subconjunto de tarefas é processado. A última proposta observada é devida a Yanasse (2009), que propõe um limitante inferior baseado na capacidade mínima que o magazine da máquina deva possuir para que exista uma sequência de processamento de tarefas e uma sequência “associada” para o problema de alocação, tais que, sempre que uma ferramenta for retirada da máquina, esta não precise ser novamente inserida. Analiticamente, denotando como C^* tal capacidade mínima do magazine, o limitante proposto é dado por $(l - C) + (C^* - C)$.

2.2.1 Formulações matemáticas

Apresentamos, a seguir, três modelagens matemáticas do MTSP propostas, respectivamente, por Tang e Denardo (1988a), Laporte *et al.* (2004) e Catanzaro *et al.* (2015).

2.2.1.1 Formulação de Tang e Denardo (1988a)

Os autores modelaram o MTSP com base em variáveis binárias que descrevem a posição de uma tarefa na ordem de processamento e a presença de uma ferramenta no magazine da máquina de manufatura. Os números de variáveis e restrições são $(n^2 + 2nl)$ e $(3n + 2nl - l)$, respectivamente, ambos de ordem polinomial.

Consideremos, então, as variáveis de decisão:

- w_{jk} variável binária igual a 1 se a tarefa $j \in J$ é processada na k -ésima posição do sequenciamento;
- y_{kt} variável binária igual a 1 se a ferramenta $t \in T$ está alocada na máquina durante o processamento da k -ésima tarefa;
- z_{kt} variável binária igual a 1 se a ferramenta $t \in T$ participa do processamento da k -ésima tarefa, mas não do processamento da $(k - 1)$ -ésima tarefa.

O modelo (TD) de Tang e Denardo (1988a) é dado por:

$$(TD) \quad \text{Min} \quad \sum_{k \in K \setminus \{1\}} \sum_{t \in T} z_{kt} \quad (2.1)$$

sujeito a

$$\sum_{j \in J} w_{jk} = 1, \quad k \in K \quad (2.2)$$

$$\sum_{k \in K} w_{jk} = 1, \quad j \in J \quad (2.3)$$

$$\sum_{j \in J_t} w_{jk} \leq y_{kt}, \quad k \in K, t \in T \quad (2.4)$$

$$\sum_{t \in T} y_{kt} \leq C, \quad k \in K \quad (2.5)$$

$$y_{kt} - y_{(k-1)t} \leq z_{kt}, \quad k \in K \setminus \{1\}, t \in T \quad (2.6)$$

$$w_{jk} \in \{0, 1\}, \quad j \in J, k \in K \quad (2.7)$$

$$y_{kt} \in \{0, 1\}, \quad k \in K, t \in T \quad (2.8)$$

$$z_{kt} \in \{0, 1\}, \quad k \in K, t \in T. \quad (2.9)$$

A função objetivo minimiza o número de ferramentas inseridas no início do processamento de cada tarefa, minimizando, assim, o número de trocas de ferramentas. As restrições (2.2) e (2.3) dizem respeito à designação das tarefas, impondo que todas as tarefas devem ser processadas, uma por vez. As restrições (2.4) garantem que todas as ferramentas requeridas por uma tarefa estejam na máquina no início do seu processamento, enquanto as restrições (2.5) impõem que a capacidade C do compartimento de ferramentas deve ser respeitada. As trocas realizadas entre tarefas processadas de forma consecutiva são identificadas pelas restrições (2.6). Por fim, as restrições (2.7), (2.8) e (2.9) se referem aos domínios das variáveis.

Na Figura 2.4, ilustramos como o modelo (TD) contabiliza o número de trocas por meio da variável binária z_{kt} .

Figura 2.4: Contagem de trocas pelo modelo TD - Solução ótima para a instância-teste.

Utilizando um exemplo relativamente pequeno ($n = 10$, $l = 9$ e $C = 4$), Tang e Denardo (1988a) avaliaram o esforço computacional necessário para a solução do MTSP por meio da modelagem apresentada. Foram utilizados dois resolvedores da época, chamados MPSX/MIP e PIPX, os quais não apresentaram resultados satisfatórios utilizando um tempo médio de 12 minutos. Laporte *et al.* (2004) justificam este desempenho com o argumento de que a relaxação linear deste modelo resulta um limitante inferior igual a zero, conforme a Proposição 2.2.1.

Proposição 2.2.1. Se $w_{jk} \doteq \frac{1}{n}$, para todo $j \in J$ e todo $k \in K$, então o modelo (TD) relaxado admite uma solução que não requer trocas de ferramentas.

Prova: Ao supormos que $w_{jk} = \frac{1}{n}$ para todo $j \in J$ e todo $k \in K$, satisfazemos as restrições (2.2) e (2.3) de designação, e a restrição (2.4) nos leva a

$$\sum_{j \in J_t} \frac{1}{n} \leq y_{kt}, \quad \forall k \in K, \forall t \in T \Rightarrow \frac{|J_t|}{n} \leq y_{kt}, \quad \forall k \in K, \forall t \in T.$$

Definamos, então, $y_{kt} \doteq \frac{|J_t|}{n}$ para todo $k \in K$ e todo $t \in T$. Isso é possível pois o restante das restrições são respeitadas. Quanto à restrição (2.5), basta observarmos que, no máximo, C ferramentas podem ser requeridas por todas as n tarefas, portanto $\sum_{t \in T} |J_t| \leq n.C$ para todo $k \in K$.

Sendo assim, a restrição de troca (2.6) unida ao objetivo de minimização nos leva a $z_{kt} = 0$ para todo $k \in K$ e todo $t \in T$. $\square$

Devido à baixa qualidade de sua relaxação, a resolução do modelo proposto por Tang e Denardo (1988a) exige que grande parte do processo enumerativo seja realizado. Com base na análise de seus testes, os autores consideraram o subproblema de *alocação de ferramentas* na máquina e o resolveram pela proposição da política KTNS, já comentada na Seção 2.2.

2.2.1.2 Formulação de Laporte *et al.* (2004)

Laporte *et al.* (2004) modelaram o MTSP como um problema do caixeiro viajante, em que após inserir uma tarefa fictícia que represente o começo e o fim das operações, a sequência de processamento das tarefas corresponde a um *tour*. Esta formulação utiliza variáveis com semânticas diferentes das utilizadas na Seção 2.2.1.1, pois tratam da consecutividade de pares de tarefas, ao invés da posição das mesmas na ordem de processamento.

Originalmente, o número de variáveis é $\mathcal{O}(n^2 + nl)$, enquanto que o número de restrições é de ordem exponencial, devido ao uso das restrições de eliminação de subciclos propostas por Dantzig *et al.* (1954).

As variáveis de decisão são:

- x_{ij} variável binária igual a 1 se a tarefa $i \in J'$ é seguida da tarefa $j \in J'$;
- y_{it} variável binária igual a 1 se a ferramenta $t \in T$ está alocada na máquina durante o processamento da tarefa $i \in J$;
- z_{it} variável binária igual a 1 se a ferramenta $t \in T$ é inserida na máquina no momento do processamento da tarefa $i \in J$.

O modelo (LSS) de Laporte *et al.* (2004) é dado por:

$$(LSS) \quad \text{Min} \sum_{i \in J} \sum_{t \in T_i} z_{it} \quad (2.10)$$

sujeito a

$$\sum_{j \in J' \setminus \{i\}} x_{ij} = 1, \quad i \in J' \quad (2.11)$$

$$\sum_{i \in J' \setminus \{j\}} x_{ij} = 1, \quad j \in J' \quad (2.12)$$

$$y_{it} = 1, \quad i \in J, t \in T_i \quad (2.13)$$

$$\sum_{t \in T} y_{it} \leq C, \quad i \in J \quad (2.14)$$

$$x_{ij} + y_{jt} - y_{it} \leq z_{jt} + 1, \quad i \in J', j \in J, t \in T \quad (2.15)$$

$$\sum_{i, j \in S} x_{ij} \leq |S| - 1, \quad S \subset J', 2 \leq |S| \leq n - 1 \quad (2.16)$$

$$z_{it} = 0, \quad i \in J, t \in T \setminus T_i \quad (2.17)$$

$$x_{ij} \in \{0, 1\}, \quad i \in J', j \in J' \quad (2.18)$$

$$y_{it} \in \{0, 1\}, \quad i \in J, t \in T \quad (2.19)$$

$$z_{it} \in \{0, 1\}, \quad i \in J, t \in T. \quad (2.20)$$

A função objetivo (2.10) minimiza o número de trocas de ferramentas realizadas. As restrições (2.11) e (2.12) designam as tarefas às posições da sequência e , pelas restrições (2.16), não há formação de subciclos durante o processamento das tarefas. As restrições (2.13) garantem a alocação das ferramentas sempre que requeridas por uma tarefa e , por (2.14), a capacidade limitada do compartimento de ferramentas é respeitada. As restrições (2.15) se referem à contagem das trocas de ferramentas, pois se uma ferramenta t não está na máquina para o processamento da tarefa i e nem foi inserida para o processamento da tarefa j , então esta não pode seguir aquela e requerer a ferramenta t ao mesmo tempo. As restrições (2.17) tratam da eliminação de simetrias, no sentido que proíbem que uma ferramenta seja inserida na máquina sem ser necessária para a próxima tarefa a ser sequenciada. Isso pode prevenir um esforço computacional considerável, já que podem ser descartadas soluções equivalentes a outras previamente avaliadas. Por último, as restrições (2.18), (2.19) e (2.20) se referem aos domínios das variáveis.

Tabela 2.3: *Exemplo-teste - Laporte et al. (2004).*

Tarefa	Ferramentas requeridas
1	A, D, H, I
2	A, C, E
3	B, F, G, H
4	A, E, G, I
5	C, E, H
6	A, B, D

Os autores testaram a eficácia do modelo (LSS) apenas através de um exemplo pequeno, em que $n = 6$, $l = 9$ e $C = 4$, apresentado na Tabela 2.3. Por isso, provavelmente, optaram pelas restrições de eliminação de subciclos em número exponencial. Entretanto, em vista dos testes computacionais por nós realizados, optamos por um conjunto de restrições mais recente, de ordem polinomial e proposto por Desrochers e Laporte (1991) como uma melhoria das restrições elaboradas por Miller *et al.* (1960). Mais detalhadamente, adotamos as seguintes desigualdades:

$$u_i - u_j + (n - 1)x_{ij} + (n - 3)x_{ji} \leq (n - 2), \quad i, j \in \{2, \dots, n\} \quad (2.21)$$

$$1 + (n - 3)x_{i1} + \sum_{j=2}^n x_{ji} \leq u_i, \quad i \in \{2, \dots, n\} \quad (2.22)$$

$$u_i \leq (n - 1) - (n - 3)x_{1i} - \sum_{j=2}^n x_{ij}, \quad i \in \{2, \dots, n\} \quad (2.23)$$

sendo u_i a ordem em que o vértice i é visitado no *tour*.

O valor da solução ótima do modelo (LSS) relaxado aplicado ao exemplo de Laporte *et al.* (2004) é 2,14, enquanto que a relaxação linear do modelo (TD) é zero e o valor ótimo inteiro é 7.

Apesar dos avanços constatados, Laporte *et al.* (2004) analisaram ainda outras características do MTSP que, inseridas na função objetivo e no conjunto de restrições, podem melhorar de forma significativa o valor de sua relaxação linear, aumentando a eficácia de métodos de solução. Primeiramente, consideremos J_C o conjunto de todas as tarefas que requerem exatamente C ferramentas. Pois, se existe uma tarefa $i \in J_C$, então o número de trocas a serem realizadas entre i e sua sucessora - digamos j - é dado por $|T_j \setminus T_i|$, exatamente, ou seja, o número de ferramentas que são

necessárias para o processamento da tarefa j , mas não são requeridas pela tarefa i . Desta forma, segue que $z_{jt} \geq x_{ij}$ para todo $j \neq i$ e todo $t \in T_j \setminus T_i$, o que nos permite contar precisamente uma parcela do número de trocas de ferramentas a serem efetuadas durante a realização de todas as tarefas.

Consideremos, assim, a função objetivo

$$\text{Min} \sum_{i \in J} \sum_{t \in T_i} z_{it} + \sum_{i \in J_C} \sum_{j \neq i} |T_j \setminus T_i| x_{ij}, \quad (2.24)$$

que nos possibilita eliminar as equações (2.15) para os índices $\{(i, j, t); i \in J_C, j \neq i \text{ e } t \in T_j \setminus T_i\}$. Notemos, portanto, que estas eliminações combinadas com o critério de minimização da variável z_{it} garantem que $z_{it} = 0$ para todo $i \in J_C$ e todo $t \in T$, motivo pelo qual podemos manter o primeiro somatório da função objetivo (2.24) para todo $i \in J$.

Ainda mais, definindo $l_{ij} \doteq \max\{0, |T_i \cup T_j| - C\}$, os autores propuseram outras três restrições capazes de melhorar consideravelmente a proximidade entre a relaxação linear e a envoltória convexa deste modelo:

$$\sum_{t \in T_j} z_{jt} \geq \sum_{\substack{i \in J \setminus \{j\} \\ |T_i| \neq C}} l_{ij} x_{ij}, \quad j \in J \quad (2.25)$$

$$\sum_{i \in J \setminus \{j\}} x_{ij} + z_{jt} \leq 1, \quad t \in T, j \in J \quad (2.26)$$

$$\sum_{t \in T_i \setminus T_j} y_{jt} \geq (C - |T_j|) x_{ij}, \quad i \in J_C, j \in J : |T_j| < C. \quad (2.27)$$

As restrições (2.25) definem um limitante inferior para o número de trocas de ferramentas, já que consideram os *slots* não utilizados pela tarefa i onde podem ser inseridas ferramentas no início do processamento da tarefa j . Pelas restrições (2.26), se uma tarefa j requer uma ferramenta t , então esta deve ser inserida no momento do processamento de j ou j deve seguir imediatamente uma tarefa i que utilize tal ferramenta. Por fim, as restrições (2.27) previnem trocas desnecessárias ao permitirem que, durante o processamento de uma tarefa j , o magazine da máquina seja preenchido pelo conjunto T_j e por $(C - |T_j|)$ ferramentas “herdadas” do conjunto T_i .

Os valores ótimos da relaxação linear obtidos pela adição da função objetivo (2.24), sem e com o conjunto de restrições (2.25), (2.26) e (2.27), foram 2,91 e 6,0, respectivamente. Para finalizar, observemos que quando $|T_i| = C$ para todo $i \in J$, o modelo (LSS) se reduz ao TSP, uma vez que o termo l_{ij} passa a representar exatamente o número de trocas de ferramentas entre tarefas $i, j \in J$ (Hertz *et al.*, 1998). Neste caso, podemos ignorar as restrições de troca (2.15) e considerar a função objetivo

$$\text{Min} \sum_{i \in J_C} \sum_{j \in J \setminus \{i\}} l_{ij} x_{ij}. \quad (2.28)$$

2.2.1.3 Aplicação das melhorias de Laporte *et al.* (2004) ao modelo de Tang e Denardo (1988)

Influenciados pelas melhorias obtidas por Laporte *et al.* (2004) referentes à relaxação linear do modelo (LSS), nos pareceu natural investigar a adaptação de tais propostas à semântica do modelo de Tang e Denardo (1988a). Para isso, definimos uma nova variável de decisão:

p_{ijk} variável binária igual a 1 se a tarefa $j \in J$ é processada na posição $k \in K$ imediatamente após uma tarefa $i \in J_C$.

Isso nos permite escrever a função objetivo

$$\text{Min} \sum_{k \in K \setminus \{1\}} \sum_{t \in T} z_{kt} + \sum_{i \in J_C} \sum_{j \neq i} \sum_{k \in K \setminus \{1\}} |T_j \setminus T_i| \max\{0; w_{i(k-1)} + w_{jk} - 1\} \quad (2.29)$$

na forma linear

$$\text{Min} \sum_{k \in K \setminus \{1\}} \sum_{t \in T} z_{kt} + \sum_{i \in J_C} \sum_{j \neq i} \sum_{k \in K \setminus \{1\}} |T_j \setminus T_i| p_{ijk}. \quad (2.30)$$

Definimos, também, a função $FR(t, j) : T \times J \rightarrow \{0, 1\}$ que retorna 1 se, e somente se, a ferramenta $t \in T$ é requerida pela tarefa $j \in J$. Sendo assim, as ideias abordadas na seção anterior podem ser aplicadas ao modelo (TD) através da seguinte formulação:

$$\text{Min} \sum_{k \in K \setminus \{1\}} \sum_{t \in T} z_{kt} + \sum_{i \in J_C} \sum_{j \neq i} \sum_{k \in K \setminus \{1\}} |T_j \setminus T_i| p_{ijk}. \quad (2.31)$$

sujeito a

$$(2.2) - (2.5), (2.7) - (2.9)$$

$$y_{kt} - y_{(k-1)t} \leq z_{kt} + \sum_{i \in J_C} \sum_{j \neq i} |T_j \setminus T_i| p_{ijk}, \quad k \in K \setminus \{1\}, t \in T \quad (2.32)$$

$$w_{i(k-1)} + w_{jk} - 1 \leq p_{ijk}, \quad i \in J_C, j \neq i, k \in K \setminus \{1\} \quad (2.33)$$

$$w_{i(k-1)} + w_{jk} \geq 2p_{ijk}, \quad i \in J_C, j \neq i, k \in K \setminus \{1\} \quad (2.34)$$

$$z_{kt} \leq 1 - w_{jk} + FR(t, j), \quad j \in J, k \in K \setminus \{1\}, t \in T \quad (2.35)$$

$$\sum_{t \in T_j} z_{kt} \geq l_{ij} (w_{i(k-1)} + w_{jk} - 1), \quad i \in J : |T_i| < C, j \neq i, k \in K \setminus \{1\} \quad (2.36)$$

$$y_{(k-1)t} + z_{kt} \leq 3 - w_{i(k-1)} - w_{jk}, \quad i \in J, j \in J_t, k \in K \setminus \{1\}, t \in T \quad (2.37)$$

$$\sum_{t \in T_i \setminus T_j} y_{kt} \geq (C - |T_j|)(w_{i(k-1)} + w_{jk} - 1), \quad i \in J_C, j : |T_j| < C, k \in K \setminus \{1\} \quad (2.38)$$

$$p_{ijk} \in \{0, 1\}, \quad i \in J_C, j \in J \setminus \{i\}, k \in K. \quad (2.39)$$

Figura 2.5: Contagem de trocas modificada - Solução ótima para a instância-teste.

Nas restrições (2.32) descrevemos a nova maneira de contabilizar as trocas de ferramentas, como exemplificado na Figura 2.5. Destacamos que se uma instância não possui tarefas que utilizem exatamente C ferramentas, a restrição é idêntica à (2.6) apresentada na Seção 2.2.1.1. As restrições (2.33), (2.34) e (2.39) definem analiticamente a variável p_{ijk} . Abordamos a eliminação de simetria nas restrições (2.35), que impõem que uma ferramenta t não deve ser inserida na máquina no início do processamento de uma tarefa j que não a requeira. Finalmente, as restrições (2.36), (2.37) e (2.38) traduzem para o modelo (TD) o significado das restrições de melhoria (2.25), (2.26) e (2.27) relativas ao modelo (LSS).

Entretanto, verificamos que o conjunto de soluções $w_{jk} = \frac{1}{n}$ para todo $j \in J$ e todo $k \in K$ e $y_{kt} = \frac{|J_t|}{n}$ para todo $k \in K$ e todo $t \in T$ (como na Proposição 2.2.1) satisfaz todas as novas restrições adicionadas ao modelo (TD), donde concluímos que $z_{kt} = 0$ e $p_{ijk} = 0$ para todo $i \in J_C$, todo $j \in J$, todo $k \in K$ e todo $t \in T$. Em síntese, as desigualdades válidas adicionadas ao modelo (TD) não são eficazes, pois, ao constataremos que o valor da relaxação linear se mantém nulo, estas modificações apenas expandem as dimensões do problema.

2.2.1.4 Formulação de Catanzaro *et al.* (2015)

Segundo Catanzaro *et al.* (2015), a inspiração para a proposição de modelagens mais recentes para o MTSP surgiu ao perceberem que, apesar de melhores que os limitantes inferiores do modelo (TD), os limitantes do modelo (LSS) ainda eram considerados insatisfatórios, pois resolviam apenas problemas com até 10 tarefas e gasto computacional médio de 3600 segundos. São propostas, então, duas formulações para o MTSP, com relaxações lineares de melhor qualidade. A proposta inicial é fortemente inspirada no trabalho de Laporte *et al.* (2004) e, por meio de manipulação de variáveis, da inserção de desigualdades válidas e de conceitos matriciais como matrizes unimodulares e blocos-0 (conhecido, em inglês, como *block-0*), obtém a formulação (CGL) baseada em fluxo em arcos.

Por definição, um bloco-0 com relação a uma ferramenta t é uma sequência de tarefas com início e término em i e j , respectivamente, cujas tarefas intermediárias não necessitam de t para seus processamentos. Por exemplo, a Tabela 2.4 mostra a matriz de requerimentos correspondente ao exemplo descrito na Tabela 2.3. Vemos, na linha 4, um bloco-0 com respeito à ferramenta D com início na tarefa 1 e término na tarefa 6.

Tabela 2.4: Exemplo de bloco-0.

Ferramentas	Tarefas					
	1	2	3	4	5	6
A	1	1	0	1	0	1
B	0	0	1	0	0	1
C	0	1	0	0	1	0
D	1	0	0	0	0	1
E	0	1	0	1	1	0
F	0	0	1	0	0	0
G	0	0	1	1	0	0
H	1	0	1	0	1	0
I	1	0	0	1	0	0

Portanto, considerando G como o grafo completo tendo como vértices as tarefas do conjunto J' , naturalmente podemos enxergar um bloco-0 com relação à t como um caminho no subgrafo direcionado $G_{ij}^t = (V_{ij}^t, A_{ij}^t)$ de G , sendo o conjunto de vértices dado por $V_{ij}^t = (J_t)^c \cup \{i, j\}$, para todos i, j tais que $i \neq j$ e $t \in (T_i \cap T_j)$, e o conjunto de arestas, A_{ij}^t , definido pelo subconjunto de arcos que têm como destino elementos de V_{ij}^t , com exceção do(s): (i) arco (i, j) , (ii) arcos entrando por i , e (iii) arcos divergindo por j . Sendo assim, como alternativa de solução para o MTSP, a formulação (CGL) visa a determinação de uma ordem de sequenciamento de tarefas que minimize o número de blocos-0 com relação às ferramentas do conjunto T .

Para sua escrita, consideremos as seguintes variáveis de decisão:

- x_{ij} variável binária igual a 1 se a tarefa $i \in J'$ é seguida da tarefa $j \in J'$;
- $f_{ij,kl}^t$ variável igual a 1 se o vértice k é imediatamente seguido pelo vértice l em um caminho de G_{ij}^t .

O modelo (CGL) de [Catanzaro et al. \(2015\)](#) é dado por:

$$(CGL) \quad \text{Min} \sum_{j \in J} |T_j| x_{ij} + \sum_{\substack{i, j \in J: i \neq j \\ t \in T_j \setminus T_i}} (x_{ij} - \sum_{\substack{k \in J: k \neq j \\ t \in T_k}} f_{kj, ij}^t) \quad (2.40)$$

sujeito a

$$\sum_{j \in J': j \neq i} x_{ij} = 1, \quad i \in J' \quad (2.41)$$

$$\sum_{i \in J': i \neq j} x_{ij} = 1, \quad j \in J' \quad (2.42)$$

$$\sum_{i, j \in S: i \neq j} x_{ij} \leq |S| - 1, \quad S \subset J', 2 \leq |S| \leq n - 1 \quad (2.43)$$

$$\sum_{l \in V_{ij}^t: l \neq k} f_{ij, kl}^t - \sum_{l \in V_{ij}^t: l \neq k} f_{ij, lk}^t \geq 0, \quad i, j \in J: i \neq j, t \in (T_i \cap T_j), k \in (J_t)^c \quad (2.44)$$

$$\sum_{i \in J_t} f_{il, kl}^t \leq x_{kl}, \quad t \in T, k \in (J_t)^c, l \in J_t \quad (2.45)$$

$$\sum_{j \in J_t} f_{kj, kl}^t \leq x_{kl}, \quad t \in T, k \in J_t, l \in (J_t)^c \quad (2.46)$$

$$\sum_{i, j \in J_t: i \neq j} f_{ij, kl}^t \leq x_{kl}, \quad t \in T, k, l \in (J_t)^c: k \neq l \quad (2.47)$$

$$\sum_{\substack{t \in T: \\ t \in T_k, t \notin T_l}} \sum_{j \in J_t: j \neq k} f_{kj, kl}^t + \sum_{\substack{t \in T: \\ t \notin T_k, t \notin T_l}} \sum_{i, j \in J_t: i \neq j} f_{ij, kl}^t \leq \\ \leq (C - |T_l|) x_{kl}, \quad k, l \in J: k \neq l \quad (2.48)$$

$$x_{ij} \in \{0, 1\}, \quad i, j \in J': i \neq j \quad (2.49)$$

$$f_{ij, kl}^t \geq 0, \quad t \in T, i, j \in J: i \neq j, \quad (2.50)$$

$$k, l \in V_{ij}^t: k \neq l, \{k, l\} \neq \{i, j\}. \quad (2.51)$$

As restrições (2.41)-(2.43) correspondem às restrições (2.11), (2.12) e (2.16) do modelo (LSS). As restrições (2.44) definem as variáveis $f_{ij, kl}^t$, enquanto o conjunto de restrições (2.45)-(2.47) impõem a permanência da ferramenta t no magazine da máquina quando as tarefas k e l são processadas de maneira consecutiva e pertencem a um bloco-0 de tarefas. As restrições (2.48) se referem ao respeito à limitação do magazine e, por fim, (2.49) e (2.50) definem os domínios das variáveis de decisão.

[Catanzaro et al. \(2015\)](#) mostraram que a formulação (CGL) produz, em geral, os melhores valores de limitantes inferiores para o MTSP. Em termos de tempo computacional e quantidade de soluções ótimas alcançadas, o modelo (LSS) não foi capaz, por exemplo, de resolver qualquer uma das dez instâncias do subconjunto $|J| = 10$, $|T| = 10$ e $C = 6$, enquanto que todos estes exemplares foram solucionados pela formulação (CGL). É importante ressaltarmos que, apesar dos avanços observados e indicados ao longo desta seção, todas as três modelagens matemáticas propostas são ainda incapazes de resolver problemas com dimensões maiores que as citadas logo acima, demonstrando a importância do estudo de novos métodos, heurísticos ou exatos, para a resolução do MTSP.

2.3 IPMTC

A gestão de ferramentas em um SFM composto por várias máquinas paralelas é um problema real. Ela pode ser vista como uma extensão natural e ainda pouco estudada do MTSP apresentado

na seção anterior. O IPMTC trata do problema de sequenciamento de um conjunto de tarefas $J = \{1, \dots, N\}$ em um SFM com p máquinas idênticas e paralelas, com o objetivo de minimizar o tempo de finalização de todo o processamento. Não há compartilhamento de ferramentas entre as máquinas, cada máquina é capaz de processar todas as tarefas, entretanto uma tarefa deve ser processada em apenas uma das máquinas. Não há relações de precedência entre as tarefas, as quais devem ser processadas sem interrupção. O termo “idênticas” significa que os tempos de processamento $\{p_j; j \in J\}$, a capacidade C dos magazines e o conjunto $T = \{1, \dots, l\}$ de ferramentas disponíveis são os mesmos em todas as máquinas. O tempo $\bar{t}$ necessário para a realização de uma troca de ferramenta é constante e igual para todas as ferramentas e em todas as máquinas, sendo o tempo total de troca em uma máquina obtido pela multiplicação de $\bar{t}$ pelo número de trocas de ferramentas decorrente da respectiva ordem de processamento escolhida. Sendo assim, o *makespan* Δ é o resultado da soma dos tempos de processamento com o tempo de troca na última máquina a ser finalizada, digamos $m^* \in M$.

Surge, assim, um problema com três variáveis de decisão, as quais tratam da designação de tarefas a máquinas, de seus respectivos sequenciamentos e da alocação de ferramentas aos magazines das máquinas. O primeiro componente é conhecido na literatura por *Parallel machine scheduling problem (PM)*, *Multiprocessor scheduling problem* ou *Identical parallel machines scheduling problem*, enquanto o segundo e o terceiro, juntos, representam o próprio MTSP. Conforme esta descrição e a classificação de Allahverdi (2015) feita para problemas de sequenciamento com tempos e custos de *setup*, vemos que o MTSP pertence à classe $[P, ST_{sd}, C_{max}]$, em que P representa a disposição paralela do conjunto de máquinas, ST_{sd} os tempos de *setup* dependente da sequência de processamento, e C_{max} o *makespan*.

Os problemas PM e MTSP são ambos NP-difíceis, como demonstrado por Garey e Johnson (1979) e Oerlemans (1992) e Crama *et al.* (1994). Ao restringirmos o IPMTC a uma configuração que não exija trocas de ferramentas, obtemos o PM. Ainda, o MTSP pode ser visto como o IPMTC em um ambiente com apenas uma máquina de manufatura. Portanto, sendo PM e MTSP casos especiais do IPMTC, concluímos que também o IPMTC é um problema NP-difícil.

Na Tabela 2.5, apresentamos um exemplo pequeno, em que $p = 2$, $n = 8$, $l = 15$ e $C = 5$, a fim de ilustrar as dificuldades inerentes ao IPMTC. Neste caso, temos a variabilidade dentre as diferentes máquinas, tempos de processamento e a exigência de que cada tarefa seja executada exatamente uma vez, em exatamente uma máquina.

Tabela 2.5: *Instância-teste - IPMTC.*

Tempo de troca (s) = 15		
Tarefa	Tempo de processamento (s)	Ferramentas requeridas
1	56	B, H, I, K, M
2	10	H, J
3	42	C, G
4	22	L, M, O
5	44	A, I, J, L, N
6	20	C, F, I, K, N
7	21	C, D, E, J, N
8	40	D, E, G, I, O

Seguindo as legendas de cores definidas na Seção 2.2, vemos na Figura 2.6 uma sequência de processamento factível para as oito tarefas nas duas máquinas disponíveis, em que a máquina 1 é o elemento crítico para a obtenção do *makespan* igual a 296. Na Figura 2.7, entretanto, apresentamos um sequenciamento ótimo de tarefas, cujo *makespan* gerado é igual a 213 segundos.

Por ser um problema altamente complexo, o IPMTC exige a elaboração de métodos refinados para a sua resolução. De nosso conhecimento, não há na literatura modelos que tratem deste

Figura 2.6: Contagem do makespan - Uma solução para a instância-teste.

Figura 2.7: Contagem do makespan - Solução ótima para a instância-teste.

contexto em que as trocas de ferramentas são consideradas variáveis de decisão, juntamente com a ordem de processamento das tarefas. Isso é proposto, pela primeira vez, no Capítulo 5 desta tese.

2.4 JSSPTC

O problema de escalonamento de tarefas *job shop* com trocas de ferramentas (JSSPTC, do inglês *Job shop scheduling problem with tooling constraints*) foi proposto por Widmer (1991) e aparece como um complemento natural aos problemas MTSP e IPMTC descritos nas seções anteriores. Em suma, neste problema objetivamos determinar o sequenciamento de um conjunto de tarefas $J = \{1, \dots, n\}$ em um SFM com p máquinas inseridas em um ambiente do tipo *job shop*, de modo que o *makespan* seja minimizado. Conforme o estudo de Allahverdi (2015), o JSSPTC se enquadra na classe $[J, ST_{sd}, C_{max}]$, em que J representa o ambiente *job shop*, ST_{sd} os tempos de *setup* dependentes da sequência de processamento, e C_{max} o *makespan* como medida a ser otimizada.

Analiticamente, cada tarefa $j \in J$ é composta por um conjunto de operações $\sigma^j = (\sigma_1^j \sigma_2^j \dots \sigma_p^j)$, previamente definido, em que $\sigma_u^j = m$, $u \in \{1, \dots, p\}$, indica que a u -ésima operação da tarefa j deve ser executada na máquina m . Cada operação, por sua vez, requer um conjunto de ferramentas relativo à máquina na qual será processada, e este nunca deverá exceder a capacidade C dos magazines. Quanto às condições do ambiente *job shop*, cada máquina pode executar uma operação por vez e operações de uma mesma tarefa não podem ser processadas, simultaneamente, em máquinas distintas. Os tempos de execução das operações, $\{p_{jm}; j \in J, m \in M\}$, variam quanto às máquinas e o tempo para uma troca, $\bar{t}$, é constante para todas as ferramentas e em todas as máquinas. Portanto, sendo o tempo total de troca em cada máquina obtido pelo produto entre $\bar{t}$ e a respectiva quantidade de trocas de ferramentas, o *makespan* Δ é o resultado da soma do tempo gasto com

trocas com a soma dos tempos de processamento das operações executadas na última máquina a ser finalizada, digamos $m^* \in M$.

Tabela 2.6: *Instância-teste - JSSPTC.*

Tarefas	Tempo de troca (s) = 2					
	Tempos de processamento (s)			Ferramentas requeridas		
	Máquina 1	Máquina 2	Máquina 3	Máquina 1	Máquina 2	Máquina 3
1	1	1	1	A	A	A
2	1	2	1	A	A	A
3	2	1	2	A	B	A

Tabela 2.7: *Plano de processamento - JSSPTC.*

Tarefas	Operações		
1	1	2	3
2	2	1	3
3	3	2	1

Consideremos uma instância-teste bastante simples para o JSSPTC, com $p = 3$, $n = 3$, $l = 2$ e $C = 1$, descrita na Tabela 2.6. O plano de processamento de suas operações é dado pela Tabela 2.7 e duas soluções distintas $\mathcal{O}_1$ e $\mathcal{O}_2$ são apresentadas nas Figuras 2.8 e 2.9. Para facilitar a compreensão, representamos as tarefas 1, 2 e 3 por j_1 , j_2 e j_3 , seus processamentos por blocos nas cores violeta, vermelho e verde, respectivamente, e o tempo gasto com trocas pela linha tracejada. A representação de uma solução do JSSPTC é expressa através de um arranjo de tamanho $n \times m$. Tomando como exemplo o sequenciamento à esquerda na Figura 2.8, vemos que as sequências de tarefas nas máquinas 1, 2 e 3 são (j_1, j_2, j_3) , (j_2, j_3, j_1) e (j_3, j_2, j_1) , respectivamente, nos fornecendo a solução (1 2 3 2 3 3 1 2 1): processamos a primeira operação da tarefa 1, então a primeira operação da tarefa 2, depois a primeira operação da tarefa 3, daí a segunda operação da tarefa 2 e assim por diante.

Figura 2.8: *Solução $\mathcal{O}_1$ para a instância-teste: ótima para o JSSP e factível para o JSSPTC.*

As variáveis de decisão associadas ao JSSPTC se assemelham às variáveis definidas para o

IPMTC, pois dizem respeito à atribuição de tarefas às máquinas, aos seus sequenciamentos e à alocação de ferramentas nos magazines das máquinas. A diferença, entretanto, reside no fato de que a ordem de sequenciamento das tarefas deve ser determinada pela integração dos problemas *Job shop scheduling problem (JSSP)*, bastante conhecido na literatura, com o próprio MTSP. Uma estratégia de desagregação do JSSPTC entre o JSSP e o MTSP acarretaria em um sequenciamento de tarefas baseado ou apenas nos tempos de processamento ou apenas no número de trocas de ferramentas, respectivamente, podendo gerar soluções totalmente ineficazes para o problema original. De fato, vemos que $\mathcal{O}_1$ induz um sequenciamento ótimo para o JSSP (à esquerda), mas não para o JSSPTC (à direita). Em contrapartida, $\mathcal{O}_2$ produz um sequenciamento ótimo para o JSSPTC (à direita), mas apenas factível para o JSSP (à esquerda).

Figura 2.9: Solução $\mathcal{O}_2$ para a instância-teste: factível para o JSSP e ótima para o JSSPTC.

O JSSPTC é um problema pouco estudado na literatura. Não existem modelos matemáticos propostos para sua resolução, nem mesmo heurísticas construtivas. No Capítulo 6, apresentamos nossas contribuições para este contexto em que trocas de ferramentas são inseridas no ambiente *job shop*.

Capítulo 3

Revisão bibliográfica

Quando o setor industrial cogitava a implementação de sistemas flexíveis de manufatura, como uma tentativa de reverter os efeitos da crise econômica instalada desde o fim da Segunda Guerra Mundial, surgiram as primeiras pesquisas relacionadas à gestão do conjunto de ferramentas utilizadas pelas máquinas flexíveis de manufatura. Pois, houve antes a constatação de que tal manejo devia ser planejado. De fato, segundo o relato de [Ingersoll \(1984\)](#), o controle das ferramentas estava entre as principais variáveis que determinavam ou não a implementação de um SFM. Entretanto, a efetividade desta instalação dependia ainda dos benefícios gerados às empresas, como o aumento dos lucros e da produtividade, a diminuição do tempo ocioso das máquinas, entre outros. Sendo assim, a partir dos anos 70, muitos autores estudaram características gerais dos sistemas flexíveis de manufatura, tais como *design* (tipos e quantidade de máquinas, capacidade do sistema de manuseio de materiais e do magazine de ferramentas, *buffers*, etc), planejamento (designação de tarefas às máquinas e de ferramentas aos magazines, por exemplo), sequenciamento (ordens corretas e precisas para o processamento das tarefas e para a alocação das ferramentas) e controle (monitoramento de falhas e cumprimento de compromissos com clientes, etc), como pode ser visto em [Coffman e Denning \(1973\)](#), [Stecke \(1983\)](#), [Stecke \(1985\)](#), [Gray et al. \(1988\)](#), [Borenstein e Becker \(1994\)](#), [Crama e Oerlemans \(1996\)](#) e [Crama \(1997\)](#).

Outros textos foram mais específicos e levantaram questões diretamente relacionadas à inserção e à remoção de ferramentas nas máquinas, ou seja, propuseram a otimização das trocas de ferramentas requisitadas ao longo da produção. [Melnyk et al. \(1989\)](#) usam um modelo de simulação para avaliar o impacto da gestão de ferramentas - com diferentes níveis de disponibilidade - no desenvolvimento do sistema produtivo. [Veeramani et al. \(1992\)](#) apresentam um *survey* que relata os esforços dedicados pela academia e pelas indústrias às várias facetas do problema de controle de ferramentas. Similarmente, [Gray et al. \(1993\)](#) avaliam os dilemas característicos deste contexto e apresentam uma estrutura hierárquica de planejamento de recursos (máquinas, ferramentas e produtos), visando uma coordenação apropriada entre todas as variáveis de decisão.

Uma vez estabelecido o problema da otimização de trocas de ferramentas, muitos pesquisadores passaram a investigar maneiras concretas para solucioná-lo, através de modelagens matemáticas e proposição de métodos, exatos ou heurísticos. Apresentamos, neste capítulo, um levantamento bibliográfico de problemas referentes à minimização do número de trocas de ferramentas durante o sequenciamento de tarefas em um SFM. A organização desta parte do trabalho possui a seguinte estrutura: primeiramente, na Seção 3.1, são considerados o MTSP, suas extensões mais discutidas na literatura e alguns problemas correlatos, isto é, que se assemelham ao MTSP em contexto e/ou estrutura. A Seção 3.2 apresenta artigos relacionados ao contexto do IPMTC, o que inclui também os principais textos da literatura do problema PM. Por fim, a bibliografia do JSSPTC e de problemas a ele adjacentes é detalhada na Seção 3.3.

3.1 Trocas de ferramentas em uma única máquina - MTSP

Os autores precursores na abordagem direta da minimização do número de trocas de ferramentas em uma única máquina foram Tang e Denardo (1988a) e Bard (1988). Os primeiros modelam o MTSP como um problema inteiro linear e, inicialmente, tentam resolvê-lo por meio de resolvidores comerciais da época, denominados MPSX/MIP (um código de programação inteira mista da IBM Computadores) e PIPX (Crowder *et al.*, 1983). Diante da pouca eficiência dos resultados, propõem uma heurística que resolve os problemas de *sequenciamento de tarefas* e *alocação de ferramentas* de forma independente: um procedimento “guloso” (GP, do inglês *Greedy procedure*) determina uma sequência de processamento das tarefas, a política KTNS (do inglês, *Keep tool needed soonest*) determina a ordem exata em que as ferramentas devem ser alocadas na máquina e, por último, as sequências obtidas são perturbadas de modo a determinar a que gera o menor número de trocas de ferramentas. Utilizando conjuntos de instâncias randômicas, os autores realizam testes para comprovar a eficiência do procedimento GP. Além disso, mostram que a política KTNS resolve o problema de alocação de ferramentas de maneira ótima e em tempo polinomial. Por outro lado, Bard (1988) propõe uma formulação inteira com função objetivo não linear e a resolve por meio da relaxação das variáveis duais associadas às restrições de requerimento de ferramentas e capacidade do magazine da máquina. A partir da sequência de processamento obtida heurísticamente, a política KTNS é usada para determinar uma solução local para o MTSP. Os resultados demonstram que a otimalidade é alcançada em grande parte dos casos.

Crama *et al.* (1994) apresentam uma nova formulação para o MTSP, também inteira e com função objetivo quadrática. Esse é um trabalho importante e muito referenciado, uma vez que contém a prova formal de que, quando $C \geq 2$, o MTSP pertence à classe dos problemas NP-difíceis. Os autores propõem várias heurísticas construtivas e de refinamento baseadas em heurísticas clássicas para o problema do caixeiro viajante (TSP), como *Shortest edge*, *Nearest neighbor*, *Farthest insertion*, *NN Block Minimization*, *FI Block minimization*, *Multiple-start greedy* e *Load-and-optimize*. Os testes computacionais comprovam, por exemplo, que as heurísticas baseadas no TSP apresentam melhores resultados para grupos de instâncias densas e, no caso de instâncias grandes, a *Farthest insertion* é uma boa opção. Nesta mesma linha, Hertz *et al.* (1998) apresentam e analisam heurísticas baseadas no TSP, porém, como principal contribuição, os autores sugerem duas novas formas de medir a “distância” entre as tarefas, consideradas menos “miópes”, no sentido que analisam não somente o par de tarefas i e j sequenciadas consecutivamente, mas também a situação do magazine antes da execução da primeira tarefa. Tais melhorias proporcionam um desempenho superior às heurísticas de Crama *et al.* (1994). Também Privault e Finke (2000) utilizam parte deste grupo de heurísticas, adaptando o problema da localização de k -servidores móveis (do inglês, *k-server problems*) ao MTSP.

No mesmo ano, Djellab *et al.* (2000) formulam o MTSP utilizando uma representação por hipergrafos chamada MGEP-Cap (do inglês, *Minimizing the number of total weighted gap in edge-projection*). Então, desenvolvem uma heurística iterativa denominada *Iterated Best Insertion* (IBI) para determinar uma solução ótima local de tal representação. Logo, estabelecida uma permutação inicial dos vértices do hipergrafo, os autores aplicam a política KTNS com o objetivo de otimizar a alocação das ferramentas.

Outro trabalho, muito referenciado na literatura do MTSP, é devido à Laporte *et al.* (2004). Através da inserção de uma tarefa fictícia que representa o início e o término das operações, o MTSP é modelado como um problema do caixeiro viajante. Além disso, neste texto são propostos e comparados dois algoritmos exatos para a minimização do número de trocas de ferramentas, uma versão do *Branch-and-Bound* ($B\&B$) e um *Branch-and-Cut* ($B\&C$) testado através do modelo proposto e que, no nó raiz, utiliza a heurística GENIUS (Gendreau *et al.*, 1992) para obter uma solução factível.

Yanasse e Lamosa (2005) formulam o MTSP como um Problema do caixeiro viajante generalizado (GTSP, do inglês *Generalized traveling salesman problem*), ideia proposta inicialmente no trabalho Yanasse e Pinto (2002). Em linhas gerais, a abordagem consiste em determinar um circuito de custo mínimo em um grafo MTSP que passe por pelo menos um nó de cada *cluster*

contendo subconjuntos de nós deste grafo. Behzad e Modarres (2002) demonstram que o GTSP pode ser transformado em um TSP, um problema estudado há tempos por muitos pesquisadores e cujos avanços podem ser favoráveis ao desenvolvimento de métodos de solução para o MTSP.

Uma abordagem inovadora para o MTSP foi proposta por Zeballos (2010). O autor apresenta uma formulação baseada em programação por restrições (*Constraint programming*). O texto considera, de forma integrada, várias questões industriais, entre estas a minimização do número de trocas de ferramentas. Ainda, a estratégia de busca chamada GLMT (do inglês, *Guided load of machines and tools*) é utilizada e permite que soluções factíveis para instâncias grandes sejam alcançadas em baixo tempo computacional. Os testes apresentados relatam a eficiência do método com relação a problemas de pequeno e médio portes, pois garantem, se não ótimas, soluções de boa qualidade.

Devido à dificuldade em se resolver o MTSP de forma exata, a grande maioria dos métodos desenvolvidos para sua solução são heurísticos ou meta-heurísticos, como podemos constatar na sequência.

Algoritmos híbridos e heurísticas para o MTSP incluem Busca Tabu, Busca por agrupamento (CS, do inglês *Clustering search*), estratégias construtivas, entre outros. Gómez e Lorena (1998) apresentam um modelo para o sequenciamento de tarefas e carregamento de ferramentas usando uma função objetivo penalizada de acordo com interesses relacionados a prazos de entrega de produtos, turnos de produção e restrições quanto à capacidade do magazine da máquina. Para a obtenção de uma solução inicial, os autores propõem um algoritmo modificado de identificação de grupos (AMIG), uma adaptação do método desenvolvido em Kusiak e Chow (1986) que leva em consideração a limitação da máquina. Em seguida, aplicam a meta-heurística Busca Tabu para o aperfeiçoamento das sequências determinadas no passo anterior e, ainda, para sincronizar os conflitos entre os diferentes objetivos abordados. Desde que os pesos da função objetivo não relacionados às trocas de ferramentas sejam anulados, esta heurística constitui mais uma alternativa para a solução do MTSP. Al-Fawzan e Al-Sultan (2002) também fazem uso de Busca Tabu para solucionar o problema de sequenciamento. A alocação de ferramentas, contudo, é resolvida pela política KTNS. De maneira semelhante, Zhou *et al.* (2005) otimizam as trocas de ferramentas por meio da política KTNS, enquanto que o sequenciamento das tarefas é tratado com a meta-heurística *Beam Search*. Neste último artigo, foi realizado um pré-processamento sobre os conjuntos de tarefas que requerem, entre si, uma grande parcela de ferramentas em comum.

Outro trabalho que se baseia na estratégia de agrupamento de tarefas é Salonen *et al.* (2006). Dois grupos são unidos até que o resultado ainda gere uma solução factível para o MTSP. Para compor este método, as heurísticas *Grouping with minimum setup algorithm* (GMSA3) e GENIUS foram empregadas em conjunto com a política KTNS. Os diferentes pontos iniciais decorrentes das várias opções de agrupamentos evitam que o algoritmo se fixe em mínimos locais.

Senne e Yanasse (2009) apresentam três variações da meta-heurística *Beam Search*, onde adotam a estratégia de busca em profundidade para a ramificação da árvore de enumeração e utilizam ordenamentos parciais das tarefas. Chaves *et al.* (2012) desenvolvem uma heurística composta por duas fases, uma construtiva baseada em um grafo MTSP em que os vértices correspondem a ferramentas e uma fase de refinamento em que foi utilizada uma heurística de busca local iterativa. Os resultados computacionais mostram que o procedimento apresenta bons resultados e, mais ainda, produz bons limitantes superiores. Os autores realizam testes unindo a heurística a um método exato enumerativo (Yanasse *et al.*, 2009) e constataam melhorias na redução de seu número de nós e do tempo computacional. Mais recentemente, Chaves *et al.* (2016) propõem um método híbrido combinando CS e o algoritmo *Biased Random Key Genetic Algorithm* (BRKGA) para a identificação de regiões promissoras do espaço de busca a serem exploradas através de buscas locais. Comprovam, por meio de testes computacionais e estatísticos, a melhoria na qualidade de suas soluções quando comparadas às obtidas por Chaves *et al.* (2012).

São restritas as propostas de métodos que resolvam a minimização do número de trocas de ferramentas de forma exata e, quanto aos algoritmos já propostos, esses têm aplicabilidade prática apenas na solução de exemplares relativamente pequenos, da ordem de 25 tarefas. Existem pesquisas di-

reacionadas a questões que possam melhorar o desempenho destes métodos, como a determinação de bons limitantes inferiores e a eliminação de simetria. Laporte *et al.* (2004), Yanasse e Rodrigues (2007), Yanasse (2009) e Catanzaro *et al.* (2015) propõem limitantes inferiores para o problema, enquanto que Ghiani *et al.* (2007), Yanasse (2008) e Ghiani *et al.* (2010) abordam questões relacionadas a propriedades simétricas do MOSP e do MTSP. A seguir, revisamos todos os métodos exatos presentes na literatura do MTSP.

Yanasse (1997) aborda o problema da minimização do número de pilhas abertas (MOSP) durante operações de corte de painéis em uma indústria madeireira e pesquisa relações entre este problema e o MTSP. Mais detalhadamente, considerando tarefas como padrões de corte e ferramentas como os tipos de painéis a serem cortados, o autor modela o MOSP através da formulação do MTSP proposta em Tang e Denardo (1988a). Para a solução, é desenvolvido um algoritmo do tipo *branch-and-bound* em que os nós a serem ramificados são determinados por um procedimento “guloso”. Sendo l o número de ferramentas necessárias para o processamento de todas as tarefas e C a capacidade do magazine da máquina, outra contribuição deste trabalho é a demonstração de que a solução ótima do MTSP é $(l - C)$ se, e somente se, a solução ótima do MOSP exigir que, no máximo, C pilhas sejam abertas durante a produção. Neste caso, a analogia com o MTSP significa que uma ferramenta retirada do magazine não mais retorna à máquina. Assim, sob algumas condições relacionadas à otimalidade do MOSP, tal *branch-and-bound* é também uma estratégia capaz de resolver o MTSP de forma exata.

Mais tarde, Yanasse *et al.* (2009) propõem um esquema enumerativo que utiliza regras simples para determinar uma sequência ótima para o processamento de um subconjunto de tarefas do MTSP. Em seguida, as demais tarefas são inseridas neste conjunto ordenado de modo que o número de trocas de ferramentas não cresça ou cresça o mínimo possível. Experimentos computacionais revelam uma certa “complementaridade” deste algoritmo com o *branch-and-bound* desenvolvido por Laporte *et al.* (2004), já que ambos resolvem instâncias compostas por até 25 tarefas, mas existem problemas que apenas um dos métodos é capaz de resolver na otimalidade. Por exemplo, a enumeração de Yanasse *et al.* (2009) resolve cerca de 8% de instâncias a mais que Laporte *et al.* (2004), entretanto o último algoritmo apresenta um desempenho muito melhor nos casos críticos em que a folga entre l e C é relativamente grande.

Ghiani *et al.* (2010) consideram o MTSP como uma abordagem não linear do problema de se determinar um ciclo Hamiltoniano de custo mínimo. Como método de solução, desenvolvem um algoritmo do tipo *branch-and-cut*. Os resultados computacionais mostram que este procedimento resolve instâncias com até 45 tarefas e 30 ferramentas, marca que, ao nosso conhecimento, nunca foi alcançada por métodos exatos aplicados ao MTSP. Os autores também comparam a eficiência da sua proposta com o *branch-and-bound* de Laporte *et al.* (2004), concluindo que há instâncias em que este último método apresenta resultados melhores, mas, de forma geral, a performance de seus *branch-and-cut* é superior.

Entre as últimas propostas de métodos que resolvam o MTSP na otimalidade, citamos Yanasse (2011), Yanasse (2012) e Catanzaro *et al.* (2015). No primeiro texto, a ideia central se baseia no fato de que quando um grafo MOSP correspondente ao MTSP possui algum vértice de corte (isto é, a remoção do vértice e suas arestas provocam uma redução na conexidade do grafo), ao removê-lo, juntamente com suas arestas, uma solução ótima para o problema original pode ser construída a partir de soluções ótimas dos MTSPs correspondentes às componentes conexas do grafo resultante. Por outro lado, no segundo trabalho, o autor usa uma adaptação do algoritmo para árvores desenvolvido para o MOSP (Yanasse, 2011) para apresentar um método polinomial para um caso especial do MTSP em que seu grafo MOSP correspondente é uma árvore. Como citado anteriormente, Yanasse (1997) mostra uma relação de “dependência” entre as soluções do MTSP e do MOSP. Portanto, uma vantagem deste trabalho é a possibilidade da solução ótima para a minimização do número de trocas de ferramentas ser determinada independentemente do valor ótimo do MOSP. Por último e mais recentemente, Catanzaro *et al.* (2015) propõem uma nova formulação para o MTSP baseada em fluxo de arcos, fazendo uso de conceitos matriciais como matrizes unimodulares e blocos-0. Os experimentos computacionais realizados comprovam a superioridade desta

formulação na obtenção de melhores valores de limitantes inferiores para o MTSP, assim como na solução de exemplares maiores, em comparação com a modelagem desenvolvida por Laporte *et al.* (2004).

Além dos métodos de resolução descritos anteriormente, o MTSP também foi estudado do ponto de vista empírico. Este trabalho é devido à Shirazi e Frizelle (2001), que avaliam como o problema é administrado em sete indústrias de manufatura de pequeno e médio portes do Reino Unido. Os principais objetivos do texto são investigar se as empresas conhecem as técnicas estudadas para a solução do manejo de ferramentas e comparar os dados fornecidos por essas instituições com a performance obtida pela aplicação de seis heurísticas desenvolvidas para o MTSP. Através de entrevistas e documentos, os autores constatarem que, apesar de concordarem que a alocação das ferramentas deve ser otimizada, nenhuma das indústrias investe em técnicas para a solução do problema, e um trabalhador “mais experiente” na operação das máquinas é o responsável por tais decisões. Quanto à eficácia dos métodos adotados, os testes mostram que as heurísticas podem reduzir o número de trocas apresentado pelas empresas de 1,7% a 75,9%, em média 21,1%. Uma última conclusão é relacionada às instâncias randômicas geralmente utilizadas na literatura do MTSP: as heurísticas se comportam de forma semelhante quando testadas com os dados “artificiais” e com os empíricos fornecidos pelas empresas, contudo, os parâmetros de geração randômica de tais dados “artificiais” diferem bastante do contexto observado nas instituições entrevistadas.

Nas próximas duas seções deste capítulo, referenciamos métodos de resolução aplicados a variantes do MTSP e também problemas correlatos. Sendo assim, sob algumas suposições e adaptações, tais algoritmos podem ser aplicados para a resolução do próprio MTSP e vice-versa.

3.1.1 Variantes do MTSP

Abordaremos, nesta seção, as principais variantes do problema da minimização do número de trocas de ferramentas. Além disso, analisaremos aplicações relativas a estas novas versões do problema, bem como a forma que geralmente são solucionadas.

Alguns sistemas de manufatura são compostos por máquinas flexíveis capazes de trocar, simultaneamente, um grande número de ferramentas. Neste contexto, se o tempo gasto nos instantes de troca for independente do número de trocas realizadas, torna-se mais apropriada a minimização do número destes instantes. Tang e Denardo (1988b) foram os precursores desta abordagem e, para a resolução deste problema, propõem um algoritmo do tipo *branch-and-bound*. Widmer (1991) e Konak *et al.* (2008) desenvolvem métodos baseados na meta-heurística Busca Tabu. Nos trabalhos de Blazewicz e Finke (1994) e Gómez e Lorena (1998), são relatados problemas gerais do ambiente produtivo em que o número de instantes de trocas é um dos critérios considerados nas funções objetivo propostas. Seguindo esta mesma temática, Marvizadeh e Choobineh (2013) minimizam o número de operações de *setup* no contexto de máquinas punçoneiras, pela proposição de três tipos de heurísticas: algoritmo genético, SMA (do inglês, *Simple set merging algorithm*) e uma versão randômica desta, denominada RSMA (do inglês, *Randomized simple set merging algorithm*).

Um problema similar é o *Job-grouping*, também citado como *Batching problem*. Mais precisamente, dada uma ordem de produção em que são conhecidos os tempos de processamento e os conjuntos de ferramentas requeridos para cada tarefa, o objetivo é determinar subconjuntos (*batches*) de tarefas que possam ser processadas sequencialmente sem que sejam necessárias trocas de ferramentas, isto é, de modo que a limitação do magazine seja respeitada. Vários trabalhos consideram este problema sob diferentes objetivos, como a minimização do número de subconjuntos de tarefas, a maximização dos lucros ou a minimização do número de instantes de troca (Denizel, 2003), discutido anteriormente. Stecké (1983), uma das primeiras autoras a atentar para a otimização do manejo de ferramentas em um ambiente de manufatura, já propunha esta variante do MTSP. Como método de solução, citamos Srivastava e Chen (1996), que desenvolvem uma adaptação da meta-heurística *Simulated annealing* (SA). Outras considerações podem ser encontradas nos artigos Hirabayashi *et al.* (1984) e Hwang e Shogan (1989).

Trabalhos empíricos realizados em algumas indústrias relatam a não-uniformidade das ferramen-

tas utilizadas no processo de produção. Atentos a este fato, alguns autores estudam a minimização do número de trocas de ferramentas de diferentes tamanhos, tipos de demanda e/ou vida útil. Gronalt *et al.* (1997), Matzliach (1998), Matzliach e Tzur (2000) e Crama *et al.* (2007) tratam de ferramentas com tamanhos distintos e que, por isso, geram uma nova variável de decisão relativa à posição em que devem ser alocadas na máquina e ao número de *slots* a serem ocupados por tais ferramentas. Um dos métodos de solução propostos é devido a Tzur e Altman (2004), se chama *Aladdin* e apresenta desempenho superior, quando comparado à heurística KSTNS (do inglês, *Keep smaller tools needed soonest*), uma adaptação desenvolvida pelos autores com base na política KTNS de Tang e Denardo (1988a).

Ferramentas de diferentes tamanhos podem possuir pesos diferenciados e isso pode prejudicar o equilíbrio físico do magazine da máquina, uma questão estudada por Privault e Finke (1995) e eficientemente resolvida através de algumas heurísticas, como *Farthest insertion* e *Best insertion*. A maioria dos trabalhos que têm como foco o MTSP não considera o tempo de vida das ferramentas utilizadas. Entretanto, Sarin e Chen (1987), Denizel (2003) e Lamond e Sodhi (2003) investigam a influência da estocasticidade do tempo de vida das ferramentas no controle das trocas realizadas pelas máquinas de um sistema flexível de manufatura. Outra suposição relata o caso em que as demandas por produtos são dinâmicas e, conseqüentemente, o número de ferramentas a serem utilizadas não é conhecido *a priori*. Balakrishnan e Chakravarty (2001) exemplificam esta situação: se uma das máquinas de um ambiente produtivo precisa ser desligada, as tarefas a ela designadas devem ser subitamente alocadas a outras máquinas, o que exige uma reorganização da linha de produção. Outros exemplos são os problemas relacionados com o gerenciamento das memórias *fast* e *slow* em um computador (*Paging problem*), devido ao acesso *online* às páginas, como citado por Matzliach (1998). Há também a possibilidade de que existam relações de precedências entre as tarefas a serem sequenciadas, como abordado por Avci e Akturk (1996).

O tempo gasto com os percursos do sistema transportador entre a máquina e o armazém de ferramentas têm impacto considerável no tempo total de processamento das tarefas, de forma que alguns trabalhos focam na minimização dos movimentos destes dispositivos, como pode ser visto em Gaalman e Nawijn (1996) e Matzliach e Tzur (2000). Song e Hwang (2002) mostram que, sob esta função objetivo, a política KTNS não garante a otimalidade do problema. Diante desta constatação, apresentam uma nova modelagem para o problema que é resolvida por meio de um procedimento heurístico que utiliza o conceito de *Early insertion*, descrito em Rupe e Kuo (1997). Koulamas (1992) analisa proposições sobre o *design* dos magazines das máquinas e resolve este problema através de um algoritmo “guloso”.

Quando o tempo necessário para as trocas varia conforme a ferramenta, um critério de otimização coerente é a minimização da soma dos tempos gastos nos instantes de troca. Esta variante do MTSP é citada por um número pequeno de pesquisadores, como Crama *et al.* (1994), Privault e Finke (1995) e Matzliach (1998). No primeiro destes artigos, foram aplicados dois procedimentos para o sequenciamento das tarefas, chamados *Super task* e *Partitioning method*, e a política KTNS para o arranjo das ferramentas.

Em alguns sistemas de manufatura, a produção é baseada na distribuição de tarefas a um grupo de várias máquinas flexíveis e idênticas. Cada máquina é capaz de processar todas as tarefas necessárias para a fabricação dos produtos, mas cada tarefa é designada a apenas uma máquina, carregada com seu respectivo conjunto de ferramentas. Logo, a solução do problema depende também da designação de tarefas às máquinas. Neste temática, há artigos que consideram o manejo de ferramentas, mas minimizam o tempo final de produção (*makespan*), como Hertz e Widmer (1996) e Fathi e Barnette (2002). Por outro lado, alguns textos mantêm a otimização do manejo de ferramentas, como Sarin e Chen (1987), que propõem refinamentos baseados em relaxação Lagrangeana, Bard (1988) e Zeballos (2010). Gaalman e Nawijn (1996) são ainda mais abrangentes pois, ao permitirem que as máquinas compartilhem ferramentas, apresentam três regras para o aperfeiçoamento do sistema transportador responsável pelo manejo das ferramentas, de modo que o tempo ocioso (*idle time*) das máquinas seja o mínimo possível. Este contexto de múltiplas máquinas será revisado mais detalhadamente na Seção 3.2.

3.1.2 Problemas correlatos e aplicações

Nesta seção, listaremos alguns problemas que se assemelham em semântica e/ou estrutura ao problema de se minimizar o número de trocas de ferramentas em uma máquina, bem como contextos diferenciados em que o MTSP pode ser aplicado.

De modo geral, o problema de corte em uma máquina de manufatura é associado ao problema de sequenciamento de padrões, que consiste em determinar uma sequência em que os padrões devem ser cortados, conforme um objetivo específico. Cada item diferente cortado abre uma pilha em torno da máquina, a qual permanece aberta até que o último exemplar deste tamanho seja cortado, comprometendo questões como o manejo da produção, custos de estoque e restrições de espaço de armazenagem (Pinto, 2004). Neste contexto, Yanasse (1997), Faggioli e Bentivoglio (1998), Linhares e Yanasse (2002), Becceneri *et al.* (2004) e de la Banda e Stuckey (2007) consideram a minimização do número de pilhas abertas (MOSP, do inglês *Minimization of open stack problem*) durante o processamento dos padrões de corte. Ao admitirmos tais padrões como tarefas e os produtos como ferramentas, o MOSP torna-se um caso especial do MTSP em que, uma vez utilizada, uma ferramenta não retorna ao magazine da máquina. Yanasse (1997) ressalta, entretanto, que estes problemas não são equivalentes.

Para a produção de produtos perecíveis ou considerados frágeis, como o vidro, a minimização do tempo em que as pilhas permanecem abertas (MORP, do inglês *Minimization of order spread problem*) é um fator crucial, como discutido por Madsen (1988) e Pinto e Yanasse (2001). Quando se trata de ferramentas, portanto, esta abordagem minimiza o tempo em que a mesma permanece na máquina, antes de ser definitivamente removida. Ainda neste contexto, alguns trabalhos abordam a minimização do número de vezes que o processamento de um determinado produto é reiniciado (MDP, do inglês *Minimization of discontinuities problem*), um objetivo diretamente relacionado com o MOSP, o MORP e o MTSP (Pinto, 2004).

Em indústrias eletrônicas, a grande maioria dos produtos fabricados contém muitos tipos de circuitos impressos (PCB, do inglês *Printed Circuit Board*). A montagem destes circuitos é feita por máquinas de manufatura especializadas e consiste na inserção de miniaturas de componentes eletrônicos. Portanto, a fabricação de um produto neste tipo de linha de produção exige uma ordem para a fabricação dos circuitos impressos e, de maneira associada, a alocação dos respectivos componentes eletrônicos em um compartimento com capacidade limitada. Devido aos altos custos envolvidos no ambiente produtivo, muitos trabalhos consideram a otimização do uso das máquinas através de objetivos como a minimização do tempo gasto entre as trocas de componentes eletrônicos ou, diretamente, a minimização do número destas trocas, como pode ser visto em Sabah *et al.* (1985), Fathi e Taheri (1989), Sule (1992), Barnea e Sipper (1993) e Leon e Peters (1998). Sendo assim, ao admitirmos os circuitos impressos como tarefas a serem sequenciadas e os componentes eletrônicos como ferramentas, esta última abordagem reflete, exatamente, o contexto do MTSP (Ghrayeb *et al.*, 2003).

Belady (1966), Coffman e Denning (1973) e Mcgeoch e Sleator (1991) foram os pioneiros em problemas relacionados com o gerenciamento das memórias *fast* e *slow* em um computador, conhecidos como *Paging problems*. Mais detalhadamente, este problema ocorre quando, durante a execução de algumas tarefas, páginas precisam ser transferidas de uma memória secundária para a memória principal. Representando as páginas de acesso como ferramentas e a memória principal como o magazine da máquina, a semelhança com o MTSP decorre da necessidade de se minimizar o número de vezes em que uma página é movimentada entre as diferentes memórias, como sugerido e estudado em Blazewicz e Finke (1994), Crama (1997) e Crama *et al.* (2007). Privault e Finke (2000) resolvem esta questão por meio de uma versão clássica do problema da localização de k servidores móveis (do inglês, *K-server problems*), conforme uma sequência *online* de pedidos.

Um outro contexto industrial onde podemos encontrar o MTSP foi descrito por Burger e Jacobs (2015). Em indústrias de embalagens de produtos alimentícios, geralmente os embrulhos de plástico ou papel alumínio utilizados em barras de doces, chocolate ou salgadinhos sortidos requerem a sobreposição de camadas de impressão, em cores variadas. Uma vez que as máquinas de impressão

utilizadas nestes ambientes comportam um número limitado de cores, haverá uma operação de *setup* sempre que o processamento consecutivo de duas tarefas de impressão exigirem um número de cores maior que a capacidade da máquina. Além disso, o respectivo tempo de *setup* dependerá do número de cores envolvidas e, portanto, da sequência de tarefas. Neste trabalho, os autores consideram uma única máquina de impressão e fazem uma análise numérica acerca dos modelos de Tang e Denardo (1988a) e Laporte *et al.* (2004).

Quando o sequenciamento de tarefas e o manejo de ferramentas estudado no MTSP é feito sob a ótica da minimização do número de instantes de troca, uma consequência é o agrupamento das tarefas em subconjuntos que possam ser sequenciados ininterruptamente e sem a necessidade de trocas de ferramentas. Como constatado por Tang e Denardo (1988b), isso constitui uma generalização natural do *Bin packing problem*, em que uma lista de objetos deve ser alocada em um número mínimo de caixas com capacidades limitadas e fixas. Esta correlação foi citada também por Rajagopalan (1986).

3.2 O contexto de máquinas paralelas

O problema de otimizar as inserções e remoções de ferramentas em máquinas flexíveis de manufatura tem sido abordado, predominantemente, por métodos heurísticos. Como visto anteriormente, ambos MTSP e PM pertencem à classe dos problemas NP-difíceis (Crama *et al.*, 1994; Garey e Johnson, 1979) e podem ser considerados como casos especiais do IPMTC. Concluímos, assim, que o IPMTC também é NP-difícil e, portanto, não são conhecidos algoritmos exatos que o resolvam em tempo polinomial. Para lidar com isso, muitos pesquisadores optam por abordar o IPMTC de forma decomposta, onde um MTSP é resolvido em cada máquina com base em um escalonamento de tarefas previamente determinado pela aplicação de um algoritmo para o PM (Agnetis *et al.*, 1997; Fathi e Barnette, 2002; Fowler *et al.*, 2003; Hoh e Kim, 1997; Kurz e Askin, 2001; Turkcan *et al.*, 2007).

Uma vez que a revisão bibliográfica da componente MTSP do IPMTC já foi apresentada na Seção 3.1, consideraremos, aqui, dois aspectos principais: em primeiro lugar, o problema do sequenciamento de tarefas em um conjunto de máquinas paralelas com tempos de *setup* dependentes da sequência (PM) e, em seguida, o próprio IPMTC.

3.2.1 PM

Uma característica comum a várias heurísticas no campo de sequenciamento e escalonamento de tarefas é a noção de lista de processamento (*List scheduling*, em inglês), que primeiro define uma regra para a atribuição de prioridades às tarefas ainda não processadas e, em fases posteriores, designa cada primeira tarefa da lista ordenada para ser processada na máquina menos carregada. Várias regras de despacho foram propostas para a determinação de tais prioridades. Quatro décadas atrás, Panwalkar e Islander (1977) apontou a relação entre a forma como as listas são construídas e a precisão dos algoritmos gerados. As heurísticas *Knuth-Kleitman* e *Longest Processing Time (LPT)* são bastante aplicadas para a solução do PM convencional (Hwang *et al.*, 2005; Lee, 1991; Lee *et al.*, 2000; Parker, 1995). Os dois métodos consistem na ordenação de uma parcela ou de todo o conjunto de tarefas em ordem não-crescente de seus respectivos tempos de processamento para, em seguida, aplicar o algoritmo LPT mencionado acima. Listas geradas de maneira aleatória são adotadas como sub-rotinas de procedimentos mais complexos. Fathi e Barnette (2002), por exemplo, minimizam o *makespan* por meio de uma heurística gulosa *multi-start* (Crama *et al.*, 1994) aplicada a cinquenta listas aleatórias de tarefas. O método *Multifit* (Coffman *et al.*, 1978), por sua vez, é baseado na técnica *First Fit Decreasing* advinda do problema *Bin Packing*, e é comparável ao LPT em termos de eficiência computacional.

Grande parte das pesquisas que objetivam a minimização do *makespan* no contexto de máquinas paralelas se concentra na análise do pior caso dos algoritmos, confiando na melhoria ou na combinação de métodos já estabelecidos como uma estratégia para a determinação de melhores limi-

tantes. Lee e Massey (1988) combinam o *Multifit* com o LPT, enquanto Koulamas e Kypariris (2009) propõem uma versão modificada da heurística LPT para o problema da minimização do *makespan* considerando duas máquinas uniformes.

Allahverdi *et al.* (2008) e Allahverdi (2015) analisam, juntos, mais de 800 trabalhos publicados entre meados da década de 1990 e o final de 2014, classificando problemas de sequenciamento de tarefas inseridos em diferentes ambientes de produção. Em particular, o contexto de máquinas paralelas e idênticas com tempos de *setup* dependentes da sequência ($[P/st_{sd}]$) tem sido bastante estudado, sendo a maioria dos métodos de solução baseada em Algoritmo Genético, *Simulated Annealing* e Busca Tabu. Mendes *et al.* (2002) resolvem o $[P/st_{sd}/C_{max}]$ pela proposição de uma Busca Tabu e de um algoritmo memético baseado em um método populacional com procedimentos de busca local, enquanto Tahar *et al.* (2006) abordam o mesmo problema através da combinação de um modelo linear com considerações de divisão de tarefas. Behnamian *et al.* (2009), por sua vez, propõem um algoritmo híbrido utilizando Colônias de formigas (ACO, do inglês *Ant Colony Optimization*), SA e a meta-heurística *Variable Neighborhood Search* (VNS), e Hu e Yao (2011) apresentam uma modelagem inteira-mista e um algoritmo genético, os quais foram avaliados de acordo com um novo limitante inferior. Lee *et al.* (2010) tratam da minimização do tempo máximo de atraso (*maximum lateness*) durante o processamento de tarefas em máquinas paralelas dinâmicas em que os tempos de *setup* são dependentes da sequência, representado pela classe $[P/st_{sd}, r_j/L_{max}]$. Os autores desenvolvem um modelo matemático, donde extraem um *lower bound* para o problema e, como heurística, propõem uma versão “restrita” de SA que determina o melhor sequenciamento em uma dada vizinhança através da desconsideração de movimentos ineficazes de tarefas.

3.2.2 Trocas de ferramentas em máquinas paralelas - IPMTC

Apesar de se tratar de um importante problema de programação da produção, o IPMTC continua pouco representado na literatura. Foi investigado, inicialmente, por Berrada e Stecké (1986), que propuseram uma formulação matemática inteira e não-linear para determinar a designação de operações às máquinas e o respectivo carregamento de ferramentas nos magazines, de forma a minimizar o *makespan*. Para lidar com a intratabilidade do modelo, em si, os autores desenvolvem um B&B que resolve a versão relaxada de cada subproblema, verifica a condição de factibilidade e, por fim, rearranja a designação de forma a não mais violar as restrições antes relaxadas.

Mohamed *et al.* (1999) e Persi *et al.* (1999) estudam o sequenciamento em máquinas com magazines limitados e, para lidarem com tal situação, consideram a estratégia de agrupamento de tarefas. O primeiro trabalho analisa o desempenho de modelos já existentes na literatura e, como contribuição, propõem um modelo englobando todas as facetas do problema (agrupamento e roteamento de tarefas e carregamento das máquinas), visando a minimização do *makespan*. Persi *et al.* (1999), por sua vez, enfrentam o problema de maneira hierárquica. Propõem uma formulação inteira-mista (MIP, do inglês *Mixed Integer Programming*) para a solução do agrupamento de tarefas e, posteriormente, tratam do sequenciamento através de um modelo baseado em *Job shop scheduling problem* (JSSP), o qual é resolvido por meio da aplicação de regras de despacho.

Desde então, ao nosso conhecimento, Fathi e Barnette (2002) foram os únicos pesquisadores que estudaram o IPMTC conforme descrito na Seção 2.3. Em suma, são desenvolvidos três métodos heurísticos para a atribuição de tarefas às máquinas. O primeiro trata-se de um algoritmo de melhoria local onde são testadas várias vizinhanças e estratégias de busca. O segundo procedimento baseia-se na rotina LPT e o último é uma adaptação de uma heurística construtiva implementada para o *k*-TSP. Como um segundo passo, os autores aplicam a política KTNS para a otimização do manejo das ferramentas, de forma a respeitar a capacidade dos magazines das máquinas. O procedimento inspirado no LPT se mostra como pior opção em termos de *makespan*, uma vez que não considera restrições relacionadas com os requerimentos e trocas de ferramentas.

Do ponto de vista industrial, Hop e Nagarur (2004b) desenvolvem um algoritmo genético para o sequenciamento de placas de circuitos impressos (PCB) em máquinas paralelas, porém não-idênticas. O problema é modelado matematicamente e considera, de forma simultânea, o agrupa-

mento das placas, o balanceamento em cada máquina, o sequenciamento das placas e componentes de trocas de ferramentas. Como método de solução, foi desenvolvido um algoritmo genético multi-objetivo. Dada uma ordem de sequenciamento das placas, a KTNS (Tang e Denardo, 1988a) é utilizada para a otimização das trocas de ferramentas. Testes computacionais atestam a eficiência do método em reduzir os tempos de *setup*. Como propostas de pesquisas futuras, os autores sugerem a consideração de custos de *setup* e ferramentas não uniformes.

Outra condição comum no contexto de sequenciamento de tarefas em múltiplas máquinas é o compartilhamento de ferramentas, como pode ser visto em Özpeynirci (2015). São considerados o sequenciamento de tarefas e o carregamento de ferramentas em máquinas paralelas, mas não idênticas, de um SFM. Há uma quantidade limitada de cópias de cada ferramenta, de forma que as máquinas compartilham ferramentas entre si e, por simplicidade, o tempo de troca de ferramentas entre o maquinário é considerado insignificante. Para solucionar esta variante do IPMTC, os autores propõem um modelo matemático e uma heurística.

Estudos mais amplos tratam o IPMTC sob diferentes circunstâncias. Atan e Pandit (1996) e Kumar e Sridharan (2009) consideram os SFMs sob a ótica do compartilhamento de ferramentas, visando reduzir o estoque e melhorar a utilização do conjunto de ferramentas. Máquinas e ferramentas distintas entre si foram estudadas por Keung *et al.* (2001) e Buyurgana *et al.* (2004). Por último, Chan e Swarnkar (2006), Das *et al.* (2009), Zeballos (2010) e Novas e Henning (2014) abordam diferentes níveis e facetas da gestão de um SFM, onde as funções objetivo representam custos de maquinário e de *setup*, roteamento de tarefas e movimentos do sistema transportador dos materiais necessários durante a produção.

Do ponto de vista prático, problemas relacionados com o IPMTC podem ser encontrados em indústrias têxteis e de papel (Gendreau *et al.*, 2001), e também em indústrias de embalagens de produtos alimentícios, como discutido no final da Seção 3.1.2.

3.3 O contexto do ambiente *job shop*

A inserção de restrições de trocas de ferramentas em ambientes de produção do tipo *job shop* foi muito pouco estudada na literatura do escalonamento de tarefas. O mesmo não ocorre com o JSSP, em si, altamente complexo e vastamente estudado, principalmente considerando a minimização do *makespan* como medida de desempenho (Scrich, 1997).

Para melhor compreendermos o estado da arte desta temática, apresentaremos na Seção 3.3.1 uma revisão da literatura do JSSP, algumas de suas principais variantes e a importância da aplicação de regras de despacho. Em seguida, na Seção 3.3.2, veremos as técnicas de solução desenvolvidas para o próprio JSSPTC, suas eficácias e as lacunas ainda existentes na literatura.

3.3.1 JSSP

O JSSP é, reconhecidamente, um dos mais complexos problemas da otimização combinatória. De fato, Lawler *et al.* (1993) o enquadram na classe de problemas NP-difícil. Justamente por isso, a maior parte dos estudos já realizados abordam o JSSP de maneira heurística e priorizam a otimização de um único critério por vez, geralmente a minimização do *makespan* ou a minimização do tempo total de atraso após o sequenciamento de todas as tarefas.

No que diz respeito ao *makespan*, foram propostos, principalmente, métodos baseados em Busca Tabu e Algoritmo Genético. Éric Taillard foi o pioneiro na aplicação da Busca Tabu para a resolução do JSSP (Taillard, 1989, 1994). O autor faz uso da representação do problema através de um grafo disjuntivo, devida a Balas (1969), e da definição da região de vizinhança proposta por Laarhoven *et al.* (1992) para elaborar um método superior ao *Shifting Bottleneck Procedure* de Adams *et al.* (1988) e à adaptação da técnica *Simulated Annealing* do próprio Laarhoven *et al.* (1992). Outros dois pontos importantes do trabalho de Taillard (1994) são a implementação paralela do JSSP e a eficiência de seu método no caso de instâncias “retangulares”, onde $p \ll n$. Desde então, muitos pesquisadores propuseram melhorias ao trabalho de Taillard (1994), dentre eles

Brandimarte (1993), Dell’Amico e Trubian (1993), Novicki e Smutnicki (1996) e Zhang *et al.* (2007).

Os algoritmos evolucionários aparecem frequentemente hibridizados. Gonçalves *et al.* (2005) utilizam um algoritmo genético na determinação da prioridade de escolha das operações a serem sequenciadas e, após a obtenção de um sequenciamento factível, o melhoram por meio da aplicação de uma heurística de busca local. Meeran e Morshed (2012) combinam a diversificação advinda do algoritmo genético com a consequente intensificação da Busca Tabu. Como resultado, provam que o método proposto obtém soluções ótimas ou quase-ótimas tanto para instâncias artificialmente geradas quanto para problemas reais enfrentados por indústrias dos ramos automobilísticos e siderúrgico. Qing-dao-er-ji e Wang (2012) propõem um algoritmo genético híbrido e comprovam sua convergência à solução ótima global com probabilidade 1. Além disso, desenvolvem uma nova maneira para o cálculo do caminho crítico (ou caminho mais longo) no grafo JSSP, o que antes era feito conforme Balas (1969). Estes resultados são superados por Cheng *et al.* (2013), os quais demonstram a superioridade de seu algoritmo evolucionário (HEA, do inglês *Hybrid Evolucionary Algorithm*) em 90% das instâncias grandes utilizadas na literatura.

Outros três exemplos de métodos para o JSSP são o *Guided Local Search (GLS)*, o GRASP (do inglês, *Greedy Randomized Adaptive Search Procedure*) e o *Path relinking*. Balas e Vazacopoulos (1998) utilizam o conceito de árvore de vizinhança para implementar a heurística GLS combinada com *Shifting Bottleneck Procedure*, responsável por encontrar soluções ótimas para as instâncias com 10 máquinas e 10 tarefas de Fisher e Thompson (1963), até então conhecidamente não solucionadas. Binato *et al.* (2002) incorporam estratégias de intensificação à meta-heurística GRASP tradicional, enquanto Aiex *et al.* (2003) aperfeiçoam este algoritmo movendo sua fase de intensificação para o fim de cada iteração do GRASP e realizando-a por meio do *Path relinking*. Peng *et al.* (2015), por sua vez, combinam o *Path relinking* com técnicas de Busca Tabu, e obtêm resultados competitivos em relação aos existentes na literatura do JSSP.

Uma estratégia comum é a utilização de regras de despacho durante a determinação de uma solução factível inicial para o JSSP. Em síntese, sempre que uma máquina se torna disponível, a regra de despacho em questão determina um valor de prioridade para cada operação passível de ser processada nesta máquina e, então, a operação com maior prioridade é escolhida. Uma compilação destas regras pode ser vista em Montazeri e Wassenhove (1990), Holthaus e Rajendran (1997), Jayamohan e Rajendran (2000) e Zahmani *et al.* (2015).

Do ponto de vista exato, existem poucos algoritmos ótimos propostos para o JSSP, sendo a maioria versões do B&B. Ainda assim, são métodos capazes de resolver apenas instâncias de pequeno porte em um tempo computacional razoável (Brucker *et al.*, 1994; Carlier e Pinson, 1989). Applegate e Cook (1991) implementam um método heurístico, um plano de corte para obtenção de um novo *lower bound* e um algoritmo B&B. O método proposto por Ten Eikelder *et al.* (1999), denominado *Bowtie*, é um dos mais eficientes para o JSSP, mas ainda requer um tempo computacional excessivo. Para o leitor interessado, o trabalho de Wang e Zou (2003) contém mais considerações, exemplos de métodos de solução e análises acerca do JSSP.

Existem problemas que tratam o JSSP sob diferentes condições ou funções objetivo. Dentre estas variantes, começamos pela minimização do tempo total de atraso, um problema comprovadamente NP-difícil (Du e Leung, 1990) e bastante conhecido na literatura (Armentano e Scrich, 2000; He *et al.*, 1993; Scrich, 1997; Scrich *et al.*, 2004). Kuhpfahl e Bierwirth (2016) seguem a mesma temática ao tratarem da minimização do tempo total de atraso ponderado. Um outro problema bastante interessante é o *Job shop* flexível, em que o número de operações por tarefa pode ser distinto e, para cada operação, existe um subconjunto de máquinas onde esta pode ser executada (Birgin *et al.*, 2015; Fattahi *et al.*, 2007; Scrich, 1997). Caumond *et al.* (2008) e González *et al.* (2015) abordam o JSSP com tempos de *setup* dependente da sequência de processamento e, além disso, restrições do tipo *time lag*, onde uma operação pode requerer o uso ininterrupto e exclusivo de uma máquina durante um período específico de tempo. Por fim, através da combinação de várias técnicas de *Constraint Programming*, Grimes e Hebrard (2015) introduzem uma nova metodologia para a resolução de problemas de sequenciamento disjuntivos. Os benefícios da nova abordagem

foram testados em variantes do JSSP como, por exemplo, restrições *time lag* e função objetivo multi-critério.

3.3.2 Trocas de ferramentas no ambiente *job shop* - JSSPTC

O JSSP, em si, representa uma gama de problemas reais de um ambiente de produção. Entretanto, em muitas situações, é necessária a consideração de características adicionais e/ou restrições mais complexas. Em indústrias automobilísticas, químicas, farmacêuticas ou de impressão de rótulos, operações de *setup* como limpeza ou trocas de ferramentas precisam ser realizadas entre o processamento de duas tarefas na mesma máquina (González *et al.*, 2015). Como este *setup* depende do par de tarefas realizadas em sequência, teremos, claramente, uma influência na determinação da ordem de processamento do conjunto total de tarefas, o que nos leva ao estudo do JSSPTC.

Widmer (1991) foi o precursor no estudo do *job shop* sob restrições de trocas de ferramentas. O autor define as premissas do JSSPTC e, como método de solução, propõe uma adaptação da meta-heurística Busca Tabu, denominada JEST (do inglês, *Job shop Extended Solved with Tabu search*) e baseada em vizinhanças definidas por movimentos de trocas entre operações sequenciadas em uma mesma máquina. Por meio de experimentos computacionais realizados em 100 instâncias, foram resolvidos exemplares pequenos, da ordem de 5 máquinas e 5 tarefas.

Mais tarde, Hertz e Widmer (1996) desenvolveram o algoritmo TOMATO (do inglês, *Tool Management with Tabu Optimization*), em que consideram duas abordagens distintas: (i) a minimização do número de instantes de troca e (ii) a minimização do número de trocas de ferramentas. Partindo de uma solução inicial advinda do trabalho de Taillard (1994) e, posteriormente, aplicando a vizinhança definida por Widmer (1991), o método TOMATO toma o melhor vizinho de cada iteração e aplica uma busca local específica para a minimização do número de instantes de troca. No caso das trocas de ferramentas, a política KTNS determina a alocação ótima das ferramentas nos magazines das máquinas. Como critério de desempate, são evitadas soluções que possuam o maior número de máquinas cujos tempos totais de processamento sejam próximos do valor do *makespan* obtido por meio do caminho crítico. Finalmente, uma outra contribuição importante de Hertz e Widmer (1996) se refere à adaptação do grafo JSSP (Balas, 1969) ao contexto das trocas de ferramentas. Em síntese, dado um par de operações op_1 e op_2 que exijam trocas de ferramentas entre si, os autores passam a considerar a existência de um nó fictício entre os vértices op_1 e op_2 , com peso proporcional ao número de instantes de troca, no caso (i), e ao número de trocas de ferramentas na abordagem (ii). De nosso conhecimento, este trabalho constitui o estado da arte para o JSSPTC nas mesmas condições descritas na Seção 2.4.

Gómez *et al.* (2012) propõem um modelo considerando o JSSP com respeito à otimização simultânea do *makespan*, do tempo total de atraso e de condições relacionadas a trocas de ferramentas. Esta abordagem consiste na adaptação da meta-heurística Busca Tabu implementada por Novicki e Smutnicki (1996) e, quanto aos experimentos computacionais, foram realizados em instâncias clássicas do JSSP propostas por Fisher e Thompson (1963) e validados de acordo com os trabalhos de Tang e Denardo (1988a) e Hertz e Widmer (1996). As soluções obtidas são de boa qualidade quanto à minimização do *makespan*, mas isso não se repete com relação aos segundo e terceiro objetivos.

Capítulo 4

MTSP

Neste capítulo, tratamos da minimização do número de trocas de ferramentas durante o processamento de um conjunto de tarefas em uma única máquina flexível, uma questão reconhecidamente crucial para a implementação eficaz de um sistema flexível de manufatura. Pois, a parcela de tempo de processamento da máquina devida à gestão das ferramentas influencia de forma significativa o tempo total de processamento das tarefas.

Na Seção 4.1, descrevemos os passos da redução de domínio implementada para o MTSP e baseada em transformações deste problema em extensões do problema do caixeiro viajante. Isso é feito com o intuito de contornar o esforço computacional relacionado ao grande número de combinações geradas pelas possíveis configurações do magazine da máquina. Desenvolvemos quatro algoritmos como estratégia para suavizar elementos críticos identificados nas soluções encontradas pela heurística Lin-Kernighan, como pode ser visto nas Seções 4.1.1.2–4.1.1.5. Encerramos este capítulo apresentando os experimentos computacionais realizados (Seção 4.2) e conclusões acerca dos resultados obtidos (Seção 4.3).

4.1 O MTSP visto como um GTSP

Henry-Labordere (1969) foi um dos precursores no estudo do GTSP. Mais detalhadamente, o GTSP é representado por um grafo G em que o conjunto de nós é dado pela união de um número finito de *clusters* de nós, disjuntos ou não. O problema consiste em determinar um circuito de custo mínimo que passe ao menos uma vez em cada *cluster* S_i . Alguns textos classificam o GTSP como inclusivo ou exclusivo, conforme os *clusters* tenham ou não intersecção (Laporte *et al.*, 1996).

Lien *et al.* (1993) mostram que qualquer grafo GTSP inclusivo $G = (V, A)$ pode ser reescrito como um grafo GTSP exclusivo $G' = (V', A')$, sem que o valor da solução ótima “original” seja alterado. Analiticamente, para cada nó $x \in V$ associamos o número $q_x \doteq |P_x|$, sendo $P_x \doteq \{i \in \{1, \dots, |V|\}; x \in S_i\}$ o conjunto dos índices de todos os *clusters* que contêm x . Então, criamos q_x réplicas do nó x no conjunto V' , denotadas por $u(x, i)$ para cada $i \in P_x$. Arestas relacionadas a réplicas de um mesmo nó x têm custo zero, ou seja, $c'(u(x, i), u(x, j)) = 0$, para todos $i, j \in P_x$. Ainda, dados dois nós distintos $x, y \in V$, definimos os custos em V' exatamente como em V , isto é, $c'(u(x, i), u(y, j)) = c(x, y)$. Os passos desta transformação são explicitados na Figura 4.1.

Mais tarde, Behzad e Modarres (2002) comprovam que todo grafo GTSP exclusivo $G' = (V', A')$ pode ser transformado em um TSP, não necessariamente simétrico, digamos $G'' = (V'', A'')$, composto pelo mesmo conjunto de nós e cujo valor da solução ótima equivalente novamente se mantém. Para isso, primeiramente consideramos que todos os nós de um *cluster* S_i' pertençam a um mesmo ciclo C_i' cujos custos das arestas são todos nulos. Então, definindo $v_r'^i$ como o r -ésimo nó do *cluster* S_i' e $v_{r(s)}'^i$ como seu sucessor imediato no ciclo C_i' , completamos a matriz de custos tomando $c''(v_r'^i, v_t'^j) = c'(v_{r(s)}'^i, v_t'^j)$, em que S_i' e S_j' são *clusters* disjuntos. Utilizando o exemplo da figura anterior, retratamos este procedimento na Figura 4.2.

Pelo fato do TSP ser um problema muito estudado na literatura, muitos algoritmos já foram propostos para a sua resolução. Baseados nisto, Yanasse e Pinto (2002) e Yanasse e Lamosa (2005)

Figura 4.1: Transformação GTSP inclusivo - GTSP exclusivo.

Figura 4.2: Transformação GTSP exclusivo - TSP.

modelam o MTSP através de uma representação por grafos resultando em um GTSP. Em suma, no contexto de um SFM e no manejo do conjunto de ferramentas, consideramos um grafo $G = (V, A)$, simétrico e não-direcionado, em que o conjunto de vértices (ou nós) V representa as combinações de C ferramentas distintas escolhidas de um total de l disponíveis. É importante notarmos que,

Figura 4.3: Uma solução para a instância-teste.

Figura 4.4: Solução ótima para a instância-teste.

dessa maneira, cada nó representa uma configuração do magazine da máquina. Um *cluster* S_i é composto pelos nós que contêm o conjunto de todas as ferramentas requeridas para a realização da tarefa i , em que $i \in \{1, \dots, n\}$, enquanto cada aresta do conjunto A representa o número exato de trocas de ferramentas necessárias para o processamento consecutivo de um dado par de nós. Sendo assim, pela inserção dos nós fictícios O e D , origem e destino final, respectivamente, o MTSP corresponde ao problema de se encontrar um caminho mínimo entre O e D que visite pelo menos um nó de cada *cluster*, sem repetição.

Considerando a instância-teste apresentada na Tabela 2.2, utilizamos as Figuras 4.3 e 4.4 para ilustrar como a otimização das rotas do GTSP pode minimizar o número de trocas de ferramentas. De fato, enquanto o percurso $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 5$ tem custo 7, ou seja, requer 7 trocas, a rota ótima $5 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 4$ tem custo 4, isto é, exige que apenas 4 ferramentas sejam trocadas.

4.1.1 Redução de domínio

A ideia da implementação de algoritmos para a redução do número de nós do GTSP surge do fato de que a geração do conjunto de nós do grafo G é computacionalmente custosa. Além disso, a partir de características do MTSP, sabemos que alguns nós nunca aparecerão em uma solução de boa qualidade e, portanto, não precisam ser gerados. Podemos citar, por exemplo, um nó incapaz de processar qualquer que seja a tarefa do conjunto $\{1, \dots, n\}$. Baseados neste tipo de observação, desenvolvemos diferentes versões de uma redução de domínio de modo a tornar o GTSP correspondente computacionalmente viável, mas sem afetar ou afetar pouco a qualidade das soluções obtidas.

Na Seção 4.1.1.1, apresentamos os elementos da estrutura geral da redução de domínio proposta como solução para o MTSP. As Seções 4.1.1.2 a 4.1.1.5 contêm as quatro estratégias adotadas como critério para a geração do conjunto de nós do grafo G e a ordem em que foram inseridas no texto advém do sucesso dos respectivos testes computacionais, quando comparados à heurística também construtiva de Chaves *et al.* (2012). As duas primeiras seções apresentam as tentativas iniciais, fundamentadas em escolhas “gulosas”. Na Seção 4.1.1.4, analisamos de forma mais específica as características dos nós que, nas duas versões anteriores, se mantinham isolados dos *clusters* contendo nós populares. Implementamos também um algoritmo do tipo *Path relinking*, que, através de modificações gradativas, busca atingir uma certa solução partindo de uma outra conhecida, com o intuito de encontrar soluções de melhor qualidade que as duas sabidas *a priori*.

4.1.1.1 Estrutura geral

O Algoritmo 1 apresenta os principais passos da heurística de redução de domínio. Nos passos 1 e 2, a função *CrucialJobs* define as tarefas chamadas cruciais do conjunto S , enquanto nos passos 3 e 4 determinamos o conjunto Z dos nós cruciais do grafo G através da função *FillNodes*. Exatamente estas duas funções são responsáveis pela distinção entre as quatro estratégias de geração de nós inseridas na redução de domínio e, por isso, as variações de seus algoritmos são detalhadas apropriadamente nas seções a seguir. Dando continuidade, do passo 5 ao fim do algoritmo, executamos o que chamamos de *Busca-h*, em que utilizamos a métrica $h \doteq \max\{0, |T_i \cup T_j| - C\}$ para medir a distância entre pares de nós (i, j) . Isso se justifica pelo fato de h representar um limite inferior ao número de trocas de ferramentas entre qualquer par de nós (i, j) . Completamos a busca quando passamos por ao menos um nó de cada *cluster* do grafo (Passos 9–11 e 25–27). Caso a busca não se complete, incrementamos a distância h e a busca é reiniciada partindo da distância $(h + 1)$ (Figuras 4.5 e 4.6). Através da função *Update*, obtemos o conjunto de nós Q resultante da heurística de redução de domínio.

Algoritmo 1 Redução de domínio - Estrutura geral

entradas $J = \{1, \dots, n\}$ (Conjunto de tarefas);
 $T = \{1, \dots, l\}$ (Conjunto de ferramentas);
 C (Capacidade do magazine);
 $T_i, i \in J$ (Conjuntos de requerimentos);
 1: Seja S um conjunto de tarefas; $S \leftarrow \emptyset$;
 2: $S \leftarrow \text{CrucialJobs}(J, C, \{T_i; i \in J\})$;
 3: Seja Z um conjunto de nós; $Z \leftarrow \emptyset$;
 4: $Z \leftarrow \text{FillNodes}(S)$;
 5: Seja Q o conjunto de todos os nós gerados; $Q \leftarrow \emptyset$;
 6: $Q \leftarrow Z$;
 7: Seja P o conjunto dos *clusters* que possuem ao menos um nó; $P \leftarrow \emptyset$;
 8: $P \leftarrow \text{Update}(Q)$;
 9: **se** $|P| = n$ **então**
 10: **retorne** Q ;
 11: **fim**
 12: Seja U um conjunto de tarefas; $U \leftarrow \emptyset$;
 13: $U \leftarrow J \setminus P$;
 14: **para todo** $h \leftarrow 1$ **a** C **faça**
 15: **para todo** $i \in Z$ **faça**
 16: Seja E_i^h um conjunto de tarefas; $E_i^h \leftarrow \emptyset$;
 17: **para todo** $j \in U$ **faça**
 18: **se** $\text{Distance}(i, j) = h$ **então**
 19: $E_i^h \leftarrow E_i^h \cup \{j\}$;
 20: **fim**
 21: **fim**
 22: **fim**
 23: $Q \leftarrow Q \cup \text{GenerateNodes}(E_i^h)$;
 24: $P \leftarrow \text{Update}(Q)$;
 25: **se** $|P| = n$ **então**
 26: **retorne** Q ;
 27: **fim**
 28: **fim**
saídas Conjunto de nós gerados Q .

Figura 4.5: Distância $h = 1$ Figura 4.6: Distância h incrementada

É importante observarmos que o método proposto é finito e, portanto, bem definido. De fato, consideremos o limite dinâmico (Lim_D) relativo à quantidade de nós gerados por *cluster*, dado por

$$Lim_D = \frac{Lim_T - Nos_G}{Clust_{NG}}, \quad (4.1)$$

em que Lim_T representa o número máximo de nós a serem gerados, Nos_G o número de nós até então gerados e $Clust_{NG}$ a quantidade de *clusters* vazios ou que contenham apenas nós herdados de tarefas cruciais através da função *FillNodes*. Como a geração de nós é controlada pela distância h e $1 \leq h \leq C$, garantimos a parada do método e o preenchimento de todos os *clusters*.

Finalmente, em posse do conjunto de nós Q , seguimos as modificações propostas por Lien *et al.* (1993) e Behzad e Modarres (2002) para transformarmos o GTSP em um TSP assimétrico (ATSP, do inglês *Asymmetric Traveling Salesman Problem*). Como primeira tentativa, aplicamos o método *Algorithm 750: CDT* implementado por Carpaneto *et al.* (1995), o que resultou em um gasto computacional excessivo durante a resolução do ATSP. Decidimos, assim, pela transformação do ATSP em um TSP simétrico, devida a Jonker e Volgenant (1983), e passamos a utilizar a heurística Lin-Kernighan (Lin e Kernighan, 1973) para os experimentos realizados com a redução de domínio.

Em poucas palavras e seguindo a notação utilizada na Seção 4.1, tomamos um ATSP representado por um grafo $G'' = (V'', A'')$ e matriz de custos $C'' = (c''_{ij})$, $i, j \in J$, e o transformamos em um TSP simétrico $G''' = (V''', A''')$ com o dobro do número de nós. Para tanto, sendo M uma constante suficientemente grande e $U = (u_{ij})$ uma matriz tal que $u_{ij} = \infty$ para todo $i, j \in J$, definimos a matriz de custos $C''' = (c'''_{ij})$ como

$$C''' = \begin{bmatrix} U & C''^t \\ C'' & U \end{bmatrix},$$

com conjunto de nós dado por $\{1, 2, \dots, n, n+1, \dots, 2n\}$. Como consequência da simetria da matriz C''' , teremos sempre pares de soluções equivalentes para o grafo G''' , cuja forma seguirá o padrão

$$i_1 \rightarrow (i_1 + n) \rightarrow i_2 \rightarrow (i_2 + n) \rightarrow \dots \rightarrow i_n \rightarrow (i_n + n) \rightarrow i_1, \quad (4.2)$$

em que $i_k \in G''$ para todo $k \in J$. Há uma correspondência biunívoca entre as soluções dos grafos G'' e G''' , de modo que ao removermos de (4.2) os nós com índices maiores que n e somarmos a quantia nM ao valor encontrado como função objetivo, obtemos a solução do grafo G'' inicial. Partindo da solução do ATSP apresentada na Figura 4.2, ilustramos a ideia de Jonker e Volgenant (1983) na Figura 4.7.

Figura 4.7: Transformação ATSP - TSP simétrico.

4.1.1.2 Estratégia “gulosa”

Nesta seção, apresentamos a primeira versão da heurística de redução de domínio. Como requisito, dizemos que uma ferramenta é *popular* quando é requerida por uma grande quantidade de tarefas.

Através do Algoritmo 2, definimos como *tarefa crucial* uma tarefa que exige C ferramentas durante seu processamento (Passo 4). Na falta deste tipo de tarefa, criamos um nó crucial artificial I (Passos 5–16). O raciocínio “guloso” relacionado ao nó I é que ele seja de qualidade, mas também capaz de processar o maior número possível de tarefas. Por isso, iniciamos seu preenchimento a partir da tarefa $\bar{j}$ com maior requerimento (Passo 6) e, em seguida, utilizamos a função *Backtracking* para adicionar ao I as ferramentas requeridas pelas tarefas restantes, ordenadas pela função *SortIncreasing* em ordem crescente do número de ferramentas que requerem (Passos 9–12). Como critério de desempate na ordenação de tarefas i e j tais que $|T_i| = |T_j|$, optamos pela que utiliza um maior número de ferramentas populares. Se a igualdade persistir, seguimos para o desempate lexicográfico. Nos passos 13 a 15, quando $|I| = C$, tomamos como cruciais as tarefas utilizadas em sua construção.

Para a determinação dos *nós cruciais*, propomos o Algoritmo 3. Basicamente, na existência de tarefas que utilizem exatamente C ferramentas, há uma equivalência entre as tarefas e os nós cruciais (Passos 2–5). Caso contrário, cria-se um único nó crucial artificial, dado pelo conjunto I , conforme os passos 6 a 8.

4.1.1.3 Estratégia reversa

As mudanças inseridas nesta seção são resultado da observação dos experimentos computacionais realizados com os algoritmos propostos anteriormente (Seção 4.2.2). Passamos a evitar grandes distâncias durante a *Busca-h*, o que foi feito pela definição de um novo conjunto de nós cruciais, dado por tarefas que, juntas, utilizam todas as l ferramentas disponíveis. Além disso, quando necessária a ordenação de um subconjunto de tarefas ou a decisão sobre um empate, revertermos nosso

Algoritmo 2 CrucialJobs - Estratégia “gulosa”

entradas $J = \{1, \dots, n\}$ (Conjunto de tarefas);
 J_c (Conjunto de tarefas que requerem C ferramentas);
 C (Capacidade do magazine);
 $T_i, i \in J$ (Conjuntos de requerimentos);

- 1: Seja S um conjunto de tarefas; $S \leftarrow \emptyset$;
- 2: Seja S_{aux} uma lista ordenada de tarefas; $S_{aux} \leftarrow \emptyset$;
- 3: Seja I um conjunto de ferramentas; $I \leftarrow \emptyset$;
- 4: $S \leftarrow J_c$;
- 5: **se** $S = \emptyset$ **então**
- 6: Seja $\bar{j} \in J$ tal que $|T_{\bar{j}}| = \max\{|T_i|; i \in J\}$;
- 7: $S \leftarrow S \cup \{\bar{j}\}$;
- 8: $I \leftarrow T_{\bar{j}}$;
- 9: $S_{aux} \leftarrow \text{SortIncreasing}(J \setminus \bar{j}, \{T_i; i \in J \setminus \bar{j}\})$;
- 10: **para todo** $j \in S_{aux}$ **faça**
- 11: $S \leftarrow S \cup \{j\}$;
- 12: $I \leftarrow \text{Backtracking}(S, S_{aux})$;
- 13: **se** $|I| = C$ **então**
- 14: **retorne** S ; **retorne** I ;
- 15: **fim**
- 16: **fim**
- 17: **senão**
- 18: **retorne** S ; **retorne** I ;
- 19: **fim**

saídas Conjunto de tarefas cruciais S e, caso necessário, o conjunto de ferramentas I .

Algoritmo 3 FillNodes - Estratégia “gulosa”

entradas S (Conjunto de tarefas cruciais);
 J_c (Conjunto de tarefas que requerem C ferramentas);
 $T_i, i \in J$ (Conjuntos de requerimentos);
 I (Conjunto de ferramentas);

- 1: Seja Z um conjunto de nós; $Z \leftarrow \emptyset$;
- 2: **se** $J_c \neq \emptyset$ **então**
- 3: **para todo** $i \in J_c$ **faça**
- 4: $Z \leftarrow Z \cup \{T_i\}$;
- 5: **fim**
- 6: **senão**
- 7: $Z \leftarrow I$;
- 8: **fim**

saídas Conjunto de nós cruciais Z .

critério, condicionando o resultado à maior quantidade de ferramentas requeridas.

No Algoritmo 4, primeiramente consideramos como cruciais as tarefas que requerem o uso de C ferramentas (Passo 2), como anteriormente. Caso estas não existam, tomamos a tarefa com o maior conjunto de requerimento (Passos 3–6). Se o critério de parada (Passos 7 e 8) não é atingido, selecionamos a tarefa $j \in V$ com mais ferramentas distintas da união $\cup_{i \in S} T_i$ (Passos 10–17), o que é feito pela função *Best1*, que retorna TRUE quando $|T_j \setminus (\cup_{i \in S} T_i)| > d^*$. Finalmente, os conjuntos S e V são atualizados e o critério de parada novamente analisado (Passos 18–22).

Algoritmo 4 CrucialJobs - Estratégia reversa

entradas $J = \{1, \dots, n\}$ (Conjunto de tarefas);
 J_c (Conjunto de tarefas que requerem C ferramentas);
 C (Capacidade do magazine);
 $T_i, i \in J$ (Conjuntos de requerimentos);

- 1: Seja S um conjunto de tarefas; $S \leftarrow \emptyset$;
- 2: $S \leftarrow J_c$;
- 3: **se** $S = \emptyset$ **então**
- 4: Seja $\bar{j} \in J$ tal que $|T_{\bar{j}}| = \max\{|T_i|; i \in J\}$;
- 5: $S \leftarrow S \cup \{\bar{j}\}$;
- 6: **fim**
- 7: **se** $(\bigcup_{i \in S} |T_i| = l)$ **então**
- 8: **retorne** S ;
- 9: **senão**
- 10: Seja V o conjunto $\{i \in J; i \notin S\}$;
- 11: $d^* \leftarrow -\infty$;
- 12: **para todo** $j \in V$ **faça**
- 13: **se** *Best1*(T_j, S, d^*, j^*) = TRUE **então**
- 14: $d^* \leftarrow |T_j \setminus \bigcup_{i \in S} T_i|$;
- 15: $j^* \leftarrow j$;
- 16: **fim**
- 17: $S \leftarrow S \cup \{j^*\}$;
- 18: $V \leftarrow V \setminus \{j^*\}$;
- 19: **se** $(\bigcup_{i \in S} |T_i| = l)$ **então**
- 20: **retorne** S ;
- 21: **fim**
- 22: **fim**
- 23: **fim**

saídas Conjunto de tarefas cruciais S .

Explicamos, a seguir, o novo conceito de nós cruciais. Devido à forma como definimos as tarefas cruciais, os nós a elas associados não necessariamente são compostos por C ferramentas, comprometendo a qualidade da *Busca-h*, uma vez que nós pouco preenchidos prejudicariam a boa estimativa da quantidade de trocas de ferramentas decorrente da métrica h . Através do Algoritmo 5 e ilustrações, descrevemos os critérios utilizados para o completamento de nós cruciais relacionados a estas tarefas cruciais que não requeiram C ferramentas.

Inicialmente, associamos um nó crucial a cada tarefa crucial que utilize todo o magazine da máquina (Passos 2–5). Na falta deste tipo de tarefa, a função *FirstNodes* toma como cruciais ao menos um par de tarefas de uma lista ordenada pelo maior número de ferramentas requeridas e utiliza a métrica h para fazer combinações entre seus conjuntos de requerimentos (Passos 6–8). Escolhemos o par de tarefas $(i', j') \in S \times S$ mais distante entre si, completamos o nó crucial j' através de i' e vice-versa.

Detalhamos estes passos por meio da instância descrita na Tabela 4.1, em que $n = 5$, $l = 20$ e $C = 10$ e não há tarefas utilizando exatamente 10 ferramentas. Como resultado da função *FirstNodes* obtemos $S = \{1, 2, 3\}$, sendo $dist(1, 2) = 5$, $dist(1, 3) = 5$ e $dist(2, 3) = 6$, logo $i' = 2$

and $j' = 3$. Para completar o nó 3 através do nó 2, adicionamos ao seu magazine as ferramentas da intersecção $T_3 \cap T_2$ e inserimos o conjunto $T_3 \setminus T_2$ nos *slots* relativos às ferramentas menos populares do conjunto $T_2 \setminus T_3$ (Figura 4.8). Para preencher o único *slot* restante, buscamos dentre as tarefas remanescentes do conjunto $X = \{1, 4, 5\}$ a que possui o maior número de intersecções com $T_2 \cup T_3$, ou seja, a tarefa 5. A seguir, escolhemos em T_5 a ferramenta com menor popularidade (neste caso, isso não foi necessário, pois $|T_5 \setminus T_2 \cup T_3| = 1$) e a inserimos para finalizar o nó $j' = 3$. (Figura 4.9). A criação do nó $i' = 2$ através do nó $j' = 3$ segue o mesmo processo.

Algoritmo 5 FillNodes - Estratégia reversa

entradas S (Conjunto de tarefas cruciais);
 J_c (Conjunto de tarefas que requerem C ferramentas);
 C (Capacidade do magazine);
 $T_i, i \in J$ (Conjuntos de requerimentos);

- 1: Seja Z um conjunto de nós; $Z \leftarrow \emptyset$;
- 2: **se** $J_c \neq \emptyset$ **então**
- 3: **para todo** $i \in J_c$ **faça**
- 4: $Z \leftarrow Z \cup \{T_i\}$;
- 5: **fim**
- 6: **senão**
- 7: $Z \leftarrow FirstNodes(S)$;
- 8: **fim**
- 9: Seja X um conjunto de tarefas; $X \leftarrow \emptyset$;
- 10: $X \leftarrow RemJobs(Z)$;
- 11: $d^* \leftarrow -\infty$;
- 12: **para todo** $L \in Z$ **faça**
- 13: **para todo** $j \in X$ **faça**
- 14: **se** $Best2(L, j, d^*, j^*) = \text{TRUE}$ **então**
- 15: $d^* \leftarrow \max\{0, |L \cup T_j| - C\}$;
- 16: $j^* \leftarrow j$;
- 17: $L^* \leftarrow L$;
- 18: **fim**
- 19: **fim**
- 20: **fim**
- 21: Seja L' um nó;
- 22: $L' \leftarrow FillNodes(L^*, j^*)$;
- 23: $X \leftarrow X \setminus \{j^*\}$;
- 24: $Z \leftarrow Z \cup \{L'\}$;
- 25: **se** $X = \emptyset$ **então**
- 26: **retorne** Z ;
- 27: **senão**
- 28: GOTO 12;
- 29: **fim**

saídas Conjunto de nós cruciais Z .

Tabela 4.1: Instância-teste do MTSP - Estratégia reversa.

Tarefa	Ferramentas requeridas
1	A, C, F, J, M, N, Q, R, T
2	A, B, E, L, N, O, P, Q, S
3	C, D, G, H, I, K, L, M, Q
4	B, C, J, R, S
5	F, G, K, L, O, P

Figura 4.8: Completamento de nós - Parte 1

Figura 4.9: Completamento de nós - Parte 2

Dando continuidade ao Algoritmo 5, usamos a função *RemJobs* para determinar as tarefas remanescentes do conjunto X (Passo 10). Então, preenchemos seus nós por meio dos nós já completos do conjunto Z (Passos 11–20). Para tanto, utilizamos a função *Best2* para selecionar a tarefa $j \in X$ com maior distância de um nó $L \in Z$, e, aplicando a função *FillNodes*, preenchemos o nó relacionado à tarefa j^* através do nó L^* . O processo de completamento é muito similar à segunda parte da função *FirstNodes*, descrita no parágrafo anterior, com exceção da possibilidade da herança de ferramentas advindas do nó L^* (Figura 4.10). Finalmente, os conjuntos X e Z são atualizados e o laço se repete até que todos os nós cruciais sejam preenchidos com $C = 10$ ferramentas.

Figura 4.10: Completamento de nós - Parte 3

4.1.1.4 Estratégia de aproximação dos nós isolados

Após uma análise aprofundada de todos os algoritmos anteriormente propostos, focamos nossa pesquisa em movimentos capazes de aproximar nós que se mantiveram isolados dos *clusters* contendo nós cruciais e, assim, diminuir o alto valor de algumas arestas da solução encontrada pela

heurística Lin-Kernighan.

A primeira mudança significativa se refere à determinação do conjunto de tarefas cruciais (Algoritmo 6). Das Seções 4.1.1.2 e 4.1.1.3, mantivemos a identificação de tarefas que utilizem C ferramentas como cruciais (Passo 3). Então, com o intuito de obter uma maior variabilidade no conjunto de tarefas consideradas cruciais, ao invés de definir uma única sequência que, unida, faz uso de todas as l ferramentas, implementamos a formulação M_{Pool} para a determinação de um *pool* de até cinco sequências deste tipo, porém distintas entre si (passos 2–10).

Para a redação do modelo M_{Pool} , consideremos P o *pool* de sequências de tarefas cruciais e $R = (r_{tj})$, $t \in T$ e $j \in J$, a matriz contendo informações sobre os conjuntos de requerimentos das tarefas, de modo que $r_{tj} = 1$ quando $t \in T_j$ e $r_{tj} = 0$, caso contrário. Definamos, assim, a variável de decisão:

z_j variável binária igual a 1 se a tarefa $j \in J$ pertence à sequência de tarefas cruciais atual.

O modelo M_{Pool} é dado por:

$$(M_{Pool}) \quad \text{Min} \sum_{j \in J} z_j \quad (4.3)$$

sujeito a

$$\sum_{j \in J} r_{tj} z_j \geq 1, \quad t \in T \quad (4.4)$$

$$\sum_{j \in \hat{J}} z_j \geq 1, \quad \hat{J} = J \setminus \Lambda, \Lambda \subseteq P \quad (4.5)$$

$$z_j \in \{0, 1\}, \quad j \in J. \quad (4.6)$$

A função objetivo (4.3) minimiza a cardinalidade de uma sequência de tarefas cruciais, respeitando a condição de que, juntas, utilizem todas as ferramentas do problema (restrições (4.4)) e que a sequência em questão seja única, quando comparada às já existentes em um subconjunto Λ do conjunto P (restrições (4.5)). Por fim, as restrições (4.6) se referem ao domínio da variável z .

Algoritmo 6 CrucialJobs - Estratégia de aproximação dos nós isolados

entradas $J = \{1, \dots, n\}$ (Conjunto de tarefas);
 J_c (Conjunto de tarefas que requerem C ferramentas);
 C (Capacidade do magazine);
 T_i , $i \in J$ (Conjuntos de requerimentos);
 P (*Pool* de sequências de tarefas cruciais);

- 1: Seja S um conjunto de tarefas; $S \leftarrow \emptyset$;
- 2: Seja p_{aux} uma sequência de tarefas;
- 3: $S \leftarrow J_c$;
- 4: **enquanto** $|P| < 5$ **faça**
- 5: $p_{aux} \leftarrow M_{Pool}(P)$;
- 6: $P \leftarrow p_{aux}$;
- 7: **para todo** $j \in p_{aux}$ **faça**
- 8: $S \leftarrow S \cup \{j\}$
- 9: **fim**
- 10: **fim**

saídas Conjunto de tarefas cruciais S .

Uma vez determinado o conjunto de tarefas cruciais S , a definição dos nós cruciais a elas associados segue os passos do Algoritmo 5, descrito na Seção 4.1.1.3. Em seguida, utilizamos a função *Busca-h* para a criação dos nós remanescentes, num limite total de 5000 nós distribuídos entre todos os *clusters*.

Após essa primeira fase padrão, propomos uma complementação à heurística de redução de domínio apresentada no Algoritmo 1. Os principais elementos desta nova abordagem são descritos entre os passos 21 e 27 do Algoritmo 7. Em suma, a função *MoreIntersection* determina o nó mais popular, x , que possui mais intersecções com os *clusters* $S_1, \dots, S_n$ (Passos 21 e 22) e, então, aplicamos a função *IsolatedClusters* para identificar os *clusters* de G sem qualquer intersecção com *clusters* que contenham x (Passos 23 e 24). Neste estágio, temos o conjunto dos nós isolados do grafo. A heurística de Renaud *et al.* (1996), desenvolvida para o GTSP, é então utilizada para a determinação de um ciclo mínimo, $\mathcal{C}$, que passe por exatamente um nó de cada *cluster* do conjunto W (Passos 25 e 26). Como forma de unir e aproximar as partes isoladas do grafo atual, utilizamos a função *GenerateNodes* para gerar um limite de 5000 nós divididos entre os *clusters* isolados. De fato, considerando um *cluster* $S_j \in W$, partimos do conjunto de ferramentas requeridas pela tarefa j e criamos novos nós utilizando também as ferramentas requeridas pelo nó crucial x , finalizando a redução de domínio (Passo 27).

Algoritmo 7 Redução de domínio - Nova abordagem baseada na estratégia de aproximação dos nós isolados

entradas $J = \{1, \dots, n\}$ (Conjunto de tarefas);
 $T = \{1, \dots, l\}$ (Conjunto de ferramentas);
 C (Capacidade do magazine);
 $T_i, i \in J$ (Conjuntos de requerimentos);

- 1: Seja S um conjunto de tarefas; $S \leftarrow \emptyset$;
- 2: $S \leftarrow \text{CrucialJobs}(J, C, \{T_i; i \in J\})$;
- 3: Seja Z um conjunto de nós; $Z \leftarrow \emptyset$;
- 4: $Z \leftarrow \text{FillNodes}(S)$;
- 5: Seja Q o conjunto de todos os nós gerados; $Q \leftarrow \emptyset$;
- 6: $Q \leftarrow Q \cup Z$;
- 7: Seja P o conjunto dos *clusters* que possuem ao menos um nó; $P \leftarrow \emptyset$;
- 8: $P \leftarrow \text{Update}(Q)$;
- 9: **se** $|P| = n$ **então**
- 10: **retorne** Q ;
- 11: **fim**
- 12: Seja U um conjunto de tarefas; $U \leftarrow \emptyset$;
- 13: $U \leftarrow J \setminus P$;
- 14: **para todo** $h \leftarrow 1$ **a** C **faça**
- 15: $Q \leftarrow Q \cup \text{Busca-}h(h)$;
- 16: $P \leftarrow \text{Update}(Q)$;
- 17: **se** $|P| = n$ **então**
- 18: **retorne** S ;
- 19: **fim**
- 20: **fim**
- 21: Seja x um nó de Q ;
- 22: $x \leftarrow \text{MoreIntersection}(Q, P)$;
- 23: Seja W um conjunto de *clusters*; $W \leftarrow \emptyset$;
- 24: $W \leftarrow \text{IsolatedClusters}(x, Q, P)$;
- 25: Seja $\mathcal{C}$ um ciclo entre os elementos de W ;
- 26: $\mathcal{C} \leftarrow \text{Heuristic}_{GTSP}(x, W, P, Q)$;
- 27: $Q \leftarrow Q \cup \text{GenerateNodes}(W)$;

saídas Conjunto de nós Q .

4.1.1.5 Path relinking

Nesta seção, apresentamos a última variação da heurística de redução de domínio proposta para o grafo MTSP/GTSP. Partindo do conjunto de nós Q , obtido através do Algoritmo 7 da seção anterior, propomos uma estratégia baseada no método *path relinking*, com o objetivo de intensificar a busca por melhores soluções no espaço envolvendo os *clusters* isolados.

Basicamente, o *path relinking* realiza movimentos exploratórios na trajetória que interconecta duas soluções de elite, chamadas *origem* e *destino*, respectivamente (Glover, 1999). Neste trabalho, aplicamos esta ideia tomando como origens os nós $\{\omega_1, \dots, \omega_\gamma\}$ por onde o caminho mínimo $\mathcal{C}$, resultante da heurística de Renaud *et al.* (1996), visita os *clusters* isolados do conjunto W . Como destino, consideramos o nó x , o mais popular entre os *clusters* $S_1, \dots, S_n$.

Cada nó $\omega_k, k = 1, \dots, \gamma$, é transformado em x através de movimentos de troca das ferramentas de sua composição. Inicialmente, são identificadas as ferramentas comuns entre cada ω_k e a solução destino x . Estas ferramentas não são movidas. As ferramentas complementares são trocadas, gerando uma gama de soluções vizinhas, a cada etapa do *path relinking*. Para que a trajetória prossiga, escolhemos a solução cujas ferramentas sejam utilizadas no maior número de *clusters* do conjunto W . Este critério foi aplicado com o intuito de selecionar nós com ferramentas populares e que sejam capazes de processar múltiplas tarefas até então isoladas. A cada passo do método, obtemos soluções mais diferentes dos ω_k 's e mais similares ao nó x . O algoritmo termina quando a solução destino é alcançada, para cada origem $\omega_k, k = 1, \dots, \gamma$.

Para melhor ilustrar o funcionamento do método proposto, apresentamos as Figuras 4.11 e 4.12, em que consideramos uma instância com $n = 12, l = 10$ e $C = 4$. Temos, neste caso, cinco *clusters* isolados, S_2, S_5, S_6, S_8 e S_{11} , cujas respectivas tarefas possuem os conjuntos de requerimentos $T_2 = \{2, 3, 8\}, T_5 = \{1, 3, 8\}, T_6 = \{7, 9\}, T_8 = \{3, 5, 7\}$, e $T_{11} = \{4, 5, 6, 10\}$. O caminho mínimo $\mathcal{C}$ passa por três nós, ω_1, ω_2 e ω_3 , exatamente as três soluções origem do *path relinking*. Na Figura 4.12, seguimos a mesma legenda utilizada no Capítulo 2, em que o retângulo azul claro representa o magazine da máquina e cada retângulo menor na cor salmão sinaliza a ferramenta que foi trocada, em comparação com a etapa anterior do método. Na coluna *Popularidade* apresentamos, para cada ferramenta do magazine, o número de vezes em que são requeridas pelas tarefas do conjunto $\{2, 5, 6, 8, 11\}$; os valores ordenados se encontram na coluna *Popularidade ordenada*. As soluções escolhidas para dar continuidade à trajetória do *path relinking* são destacadas por uma envoltória triangular em azul e o método continua até que as ferramentas de ω_1 e ω_2 também se igualem às ferramentas do nó x .

Figura 4.11: Exemplo para aplicação do algoritmo *path relinking*

4.2 Experimentos computacionais

Nesta seção, apresentamos os experimentos computacionais relacionados à heurística de redução de domínio proposta para a resolução do MTSP por meio do GTSP. Retratamos as características das instâncias na Seção 4.2.1. Nas quatro seções seguintes, detalhamos e discutimos os resultados relativos às quatro estratégias de geração de nós descritas anteriormente.

Figura 4.12: Funcionamento do algoritmo path relinking

4.2.1 Instâncias teste

Utilizamos o conjunto de instâncias geradas por Yanasse *et al.* (2009) e disponibilizadas no sítio: www.feg.unesp.br/~elfsenne/instances. O procedimento de geração destas instâncias foi inicialmente proposto por Crama *et al.* (1994) que, por sua vez, basearam-se em Tang e Denardo (1988a). Na Tabela 4.2, representamos o número de tarefas, ferramentas e a capacidade do magazine por $|J|$, $|T|$ e $|C|$, respectivamente. O número de ferramentas requeridas para cada tarefa é gerado aleatoriamente, conforme quantidades mínimas (Min) e máximas (Max) definidas *a priori*. Em seguida, um novo processo aleatório estabelece exatamente o conjunto de ferramentas necessárias para a execução de cada tarefa. Todas as ferramentas presentes em uma instância são utilizadas pelas respectivas tarefas. São gerados, no total, 1350 exemplares distribuídos entre os grupos A, B, C, D e E.

4.2.2 Experimento 1: estratégia “gulosa”

Nos primeiros passos da heurística de redução de domínio, utilizamos a estratégia “gulosa” para ordenar o conjunto de nós do grafo G priorizando as ferramentas mais requisitadas e, ainda, definimos um nó composto por C ferramentas que seja capaz de processar o maior número possível de tarefas.

Chamamos de RD1 a versão da redução de domínio descrita na Seção 4.1.1.2. Os algoritmos

Tabela 4.2: *Características dos grupos de instâncias.*

Grupo	J		T		C		Número de instâncias
	Min	Max	Min	Max	Min	Max	
A	8	8	15	25	5	20	340
B	9	9	15	25	5	20	330
C	15	15	15	25	5	20	340
D	20	25	15	25	5	20	260
E	10	15	10	20	4	12	80

foram implementados em C++ e os experimentos realizados em um computador 3.4 GHz Intel Core i7 com 16 Gb de memória. Comparada aos resultados obtidos por [Chaves et al. \(2012\)](#), os quais referenciamos por CSY, a DR1 teve pior desempenho em 50,69% das instâncias do Grupo A, 62,12% do Grupo B, 75,29% do Grupo C, 81,92% do Grupo D e 70% do Grupo E. Nos demais exemplares, os dois métodos ficaram empatados quanto ao valor da função objetivo e, em poucos casos, nosso algoritmo se sobressaiu.

Tabela 4.3: *Resultados relativos à RD1.*

Grupo	RD1		CSY	
	t(s)	Gap*(%)	t(s)	Gap*(%)
A	3,29	8,80	0,02	0,00
B	1,77	10,61	0,05	0,00
C	1,15	16,44	14,69	0,00
D	1,54	28,83	1362,17	0,00
E	0,41	16,90	14,98	0,00

A Tabela 4.3 sintetiza o comportamento dos métodos analisados. As colunas $t(s)$ representam a média do tempo computacional utilizado, em segundos, enquanto $Gap^*(\%)$ mostra o desvio médio ao considerarmos as melhores soluções encontradas por RD1 ou CSY. Apesar do baixo tempo computacional, os valores dos *gaps* de otimalidade são excessivos, até mesmo para as instâncias de pequeno porte dos Grupos A e B.

A ideia central da redução de domínio RD1 nos permite reduzir o número total de nós gerados, uma vez que temos muitos nós do grafo contendo ferramentas populares e em comum. Por outro lado, este mesmo processo aumenta o número de remoções e inserções de ferramentas nos magazines das máquinas, elevando o número de trocas. Para lidar com este cenário, revertemos o critério de ordenação, como detalhado na Seção 4.1.1.3. Os avanços decorrentes desta decisão podem ser vistos na seção a seguir.

4.2.3 Experimento 2: estratégia reversa

Utilizamos a sigla RD2 para nos referirmos aos algoritmos relativos à Seção 4.1.1.3, os quais foram implementados em C++, testados em um computador 2.67 GHz Intel Xeon X550 com 12 Gb de memória e comparados à heurística de [Chaves et al. \(2012\)](#).

Tabela 4.4: *Resultados da geração de nós - RD2*

Grupo	Nós	Limite $\uparrow$ (%)
A	676,31	9,41
B	651,88	5,76
C	958,96	5,88
D	1532,68	41,54
E	530,10	0,00
Média	869,99	12,59

A Tabela 4.4 sumariza o processo de geração dos nós do grafo MTSP/GTSP. A coluna *Nós* mostra o número médio de nós gerados para os grupos A, B, C, D e E, enquanto *Limite $\uparrow$ (%)*

sumariza a percentagem de instâncias que ultrapassam o limite máximo de 2000 nós gerados.

Tabela 4.5: *Resultados relativos ao Grupo A - RD2.*

$ J $	$ T $	$ C $	RD2			CSY		
			NT*	t(s)	Gap*(%)	NT*	t(s)	Gap*(%)
8	15	5	12,10	0,12	1,00	12,00	0,00	0,00
8	15	10	7,03	0,37	3,00	6,83	0,00	0,00
8	20	5	16,90	0,89	1,00	16,80	0,06	0,00
8	20	10	13,50	0,59	3,00	13,07	0,02	0,00
8	20	15	7,38	0,58	4,00	7,08	0,01	0,00
8	25	5	20,10	4,25	0,00	20,10	0,07	0,00
8	25	10	19,13	1,95	5,00	18,20	0,04	0,00
8	25	15	13,90	1,41	8,00	12,95	0,01	0,00
8	25	20	7,03	1,08	6,00	6,61	0,01	0,00
Média			13,01	1,25	3,00	12,63	0,02	0,00

Os demais resultados computacionais são apresentados nas Tabelas 4.5 a 4.9. Em cada uma delas, as colunas representam o número de tarefas ($|J|$), a quantidade de ferramentas disponíveis ($|T|$), a capacidade do magazine ($|C|$), o melhor valor de função objetivo encontrado (NT^*), o tempo computacional médio em segundos ($t(s)$) e o desvio médio, gap , considerando o melhor valor de função objetivo encontrado pelos dois algoritmos avaliados ($Gap^*(\%)$).

Tabela 4.6: *Resultados relativos ao Grupo B - RD2.*

$ J $	$ T $	$ C $	RD2			CSY		
			NT*	t(s)	Gap*(%)	NT*	t(s)	Gap*(%)
9	15	5	12,40	0,19	2,00	12,20	0,02	0,00
9	15	10	7,60	0,36	3,00	7,37	0,00	0,00
9	20	5	17,40	0,24	0,00	17,40	0,11	0,00
9	20	10	15,00	0,51	6,00	14,17	0,02	0,00
9	20	15	8,08	0,51	6,00	7,60	0,01	0,00
9	25	5	20,40	2073	0,00	20,40	0,16	0,00
9	25	10	19,90	1,56	6,00	18,77	0,05	0,00
9	25	15	15,88	1,55	8,00	14,74	0,04	0,00
9	25	20	7,70	0,93	7,00	7,19	0,02	0,00
Média			13,82	0,95	4,00	13,31	0,05	0,00

Tabela 4.7: *Resultados relativos ao Grupo C - RD2.*

$ J $	$ T $	$ C $	RD2			CSY		
			NT*	t(s)	Gap*(%)	NT*	t(s)	Gap*(%)
15	15	5	17,00	0,15	3,00	16,60	1,67	0,00
15	15	10	10,43	1,69	7,00	9,80	2,19	0,00
15	20	5	21,70	0,28	5,00	20,60	13,26	0,00
15	20	10	20,73	1,02	14,00	18,33	1,04	0,00
15	20	15	11,62	1,27	12,00	10,52	40,67	0,00
15	25	5	28,10	0,27	2,00	27,50	49,17	0,00
15	25	10	28,27	1,39	13,00	25,07	9,22	0,00
15	25	15	22,30	2,57	19,00	19,07	9,79	0,00
15	25	20	11,08	2,14	16,00	9,66	5,16	0,00
Média			19,03	1,09	10,00	17,46	14,69	0,00

Em instâncias de pequeno e médio portes, ou seja, dos Grupos A, B e E, os métodos se comportam de maneira bastante semelhante. Com relação aos Grupos C e D, responsáveis pelas instâncias de grande porte e onde há mais influência do alto nível de combinatoriedade devido a maiores diferenças entre o número de ferramentas e a capacidade do magazine, o algoritmo CSY obtém soluções que acarretam menos trocas de ferramentas. Com efeito, os valores médios dos $gaps$ entre RD2 e CSY variam de 10% a 22,8%.

Tabela 4.8: *Resultados relativos ao Grupo D - RD2.*

$ J $	$ T $	$ C $	RD2			CSY		
			NT*	t(s)	Gap*(%)	NT*	t(s)	Gap*(%)
20	15	5	21,50	0,14	3,00	20,90	153,13	0,00
20	15	10	9,50	1,62	16,00	8,20	767,57	0,00
20	20	5	26,20	0,27	8,00	24,20	244,99	0,00
20	20	10	12,80	3,21	21,00	10,60	1099,98	0,00
20	20	15	8,03	3,08	19,00	6,67	648,28	0,00
20	25	5	32,20	0,33	7,00	30,10	1913,17	0,00
20	25	10	19,70	3,15	28,00	15,40	1172,39	0,00
20	25	15	25,88	4,05	26,00	21,33	1493,62	0,00
20	25	20	7,65	4,74	21,00	6,15	352,11	0,00
20	15	10	7,50	3,61	27,00	5,90	2880,01	0,00
20	20	10	15,70	3,65	36,00	11,60	3240,01	0,00
20	20	15	10,70	3,62	41,00	7,60	2913,22	0,00
20	25	10	22,60	4,30	36,00	16,60	2520,03	0,00
20	25	15	12,90	5,25	29,00	10,00	0,09	0,00
20	25	10	6,90	5,15	25,00	5,50	1033,95	0,00
Média			15,98	3,08	23,00	13,38	1362,17	0,00

Tabela 4.9: *Resultados relativos ao Grupo E - RD2.*

$ J $	$ T $	$ C $	RD2			CSY		
			NT*	t(s)	Gap*(%)	NT*	t(s)	Gap*(%)
10	10	4	9,70	0,02	2,00	9,50	0,00	0,00
10	10	5	6,40	0,07	3,00	6,20	0,01	0,00
10	10	6	4,30	0,18	0,00	4,30	0,01	0,00
10	10	7	3,00	0,24	0,00	3,00	0,00	0,00
10	20	6	23,20	0,30	9,00	21,40	1,36	0,00
10	20	8	16,40	1,01	15,00	14,20	2,37	0,00
10	20	10	11,30	1,32	10,00	10,30	1,34	0,00
10	20	12	9,00	1,87	10,00	8,20	114,76	0,00
Média			10,41	0,63	6,00	9,64	14,98	0,00

Uma vez que o Grupo D é também o mais afetado pela imposição do limite no número de nós gerados, assumimos a possibilidade de que a redução de domínio tenha sido truncada prematuramente. Aumentamos, portanto, o número máximo de nós de 2000 para 10000, sempre respeitando a divisão dinâmica de nós por *cluster* estabelecida pelo limite (4.1). Novamente, não verificamos melhoras significativas nos resultados referentes aos grupos de instâncias grandes.

4.2.4 Experimento 3: estratégia de aproximação dos nós isolados

Os algoritmos propostos na Seção 4.1.1.4 foram implementados em C++ e os experimentos também conduzidos em um computador 2.67 GHz Intel Xeon X550 com 12 Gb de memória.

A Tabela 4.10 diz respeito ao processo de geração de nós do grafo. A coluna *Nós* mostra a média de nós gerados por grupo, enquanto *Limite* $\uparrow$ (%) se refere à parcela de instâncias que ultrapassam o limite fixado de 5000 nós. O tempo requerido durante a fase de geração aparece na coluna $t(s)$. Reiteramos que o tempo necessário para a geração dos nós é próximo de zero, de modo que os valores em $t(s)$ retratam o tempo decorrente da aplicação da heurística de Renaud *et al.* (1996) proposta para a resolução do GTSP.

Esta versão da heurística de redução de domínio é denotada por RD3 e comparada aos resultados apresentados por Chaves *et al.* (2012), conforme sumarizado nas Tabelas 4.11 a 4.15. As três primeiras colunas representam o número de tarefas ($|J|$), o número de ferramentas requeridas ($|T|$) e a capacidade do magazine ($|C|$). A média do número de trocas, dentre as melhores soluções obtidas por RD3 em suas cinco repetições, é mostrada na coluna NT^* , enquanto NT representa a média do número de trocas considerando todas as soluções. A coluna $t(s)$ se refere ao tempo computacional médio, em segundos, e Gap^* (%) trata do *gap* de otimalidade médio relativo à melhor solução obtida por RD3 ou CSY. As colunas NT_{UB} , NT_{YRS} e $t_{YRS}(s)$ sumarizam o número de trocas geradas pela heurística de Chaves *et al.* (2012), o número de trocas resultante da inserção do valor NT_{UB} como *upper bound* para o algoritmo enumerativo de Yanasse *et al.* (2009) e o tempo computacional requerido por este último método, respectivamente.

Tabela 4.10: Resultados da geração de nós - RD3.

Grupo	Nós	Limite $\uparrow$ (%)	$t(s)$
A	1412,35	13,53	38,38
B	1189,76	13,64	21,55
C	1609,19	12,35	9,31
D	3102,80	25,00	18,07
E	980,23	2,50	3,31
Média	1658,87	13,40	18,12

Tabela 4.11: Resultados relativos ao Grupo A - RD3.

$ J $	$ T $	$ C $	RD3				CSY			
			NT^*	NT	$t(s)$	$Gap^*(\%)$	NT_{UB}	NT_{YRS}	$t_{YRS}(s)$	$Gap^*(\%)$
8	15	5	12,00	12,02	0,21	0,00	12,00	12,00	0,00	0,00
8	15	10	6,93	6,95	0,92	1,46	6,83	6,83	0,00	0,00
8	20	5	16,90	16,94	1,00	0,60	16,80	16,80	0,06	0,00
8	20	10	13,30	13,41	2,54	1,76	13,07	13,07	0,02	0,00
8	20	15	7,33	7,39	4,16	3,53	7,08	7,08	0,01	0,00
8	25	5	20,10	20,10	9,38	0,00	20,10	20,10	0,07	0,00
8	25	10	18,80	18,89	13,06	3,30	18,20	18,20	0,04	0,00
8	25	15	13,48	13,64	28,17	4,09	12,95	12,95	0,01	0,00
8	25	20	6,89	7,00	12,64	4,24	6,61	6,61	0,01	0,00
Média			12,86	12,93	8,01	2,11	12,63	12,63	0,02	0,00

Como na seção anterior, ao considerarmos instâncias dos Grupos A , B e E , vemos que o número de trocas obtido pela redução de domínio é similar ao resultado apresentado por Chaves *et al.* (2012). O mesmo não ocorre quando tratamos das instâncias grandes do Grupo D . De fato, a

Tabela 4.12: Resultados relativos ao Grupo B - RD3.

J	T	C	RD3				CSY			
			NT*	NT	t(s)	Gap*(%)	NT _{UB}	NT _{YRS}	t _{YRS} (s)	Gap*(%)
9	15	5	12,20	12,22	0,32	0,00	12,20	12,20	0,02	0,00
9	15	10	7,50	7,58	0,95	1,76	7,37	7,37	0,00	0,00
9	20	5	17,40	17,44	0,39	0,00	17,40	17,40	0,11	0,00
9	20	10	14,57	14,77	1,22	2,82	14,17	14,17	0,02	0,00
9	20	15	7,98	8,11	1,19	5,00	7,60	7,60	0,01	0,00
9	25	5	20,40	20,42	3,08	0,00	20,40	20,40	0,16	0,00
9	25	10	19,50	19,61	9,63	3,89	18,77	18,77	0,05	0,00
9	25	15	15,52	15,70	18,20	5,29	14,74	14,74	0,04	0,00
9	25	20	7,54	7,67	3,62	4,87	7,19	7,19	0,02	0,00
Média			13,62	13,72	4,29	2,63	13,32	13,32	0,05	0,00

Tabela 4.13: Resultados relativos ao Grupo C - RD3.

J	T	C	RD3				CSY			
			NT*	NT	t(s)	Gap*(%)	NT _{UB}	NT _{YRS}	t _{YRS} (s)	Gap*(%)
15	15	5	16,80	17,32	0,28	1,20	16,60	16,60	1,67	0,00
15	15	10	10,53	10,93	1,63	7,45	9,80	9,80	2,19	0,00
15	20	5	21,30	21,84	0,59	3,40	20,60	20,60	13,26	0,00
15	20	10	19,93	20,59	3,49	8,32	18,40	18,33	1,04	0,00
15	20	15	11,43	11,99	3,07	8,65	10,52	10,52	40,67	0,00
15	25	5	27,80	28,22	0,49	1,09	27,50	27,50	49,17	0,00
15	25	10	26,80	27,63	3,72	6,90	25,07	25,07	9,22	0,00
15	25	15	21,33	22,09	8,03	11,85	19,07	19,07	9,79	0,00
15	25	20	10,88	11,37	6,08	12,63	9,66	9,66	5,16	0,00
Média			18,53	19,11	3,04	6,83	17,47	17,46	14,69	0,00

Tabela 4.14: Resultados relativos ao Grupo D - RD3.

J	T	C	RD3				CSY			
			NT*	NT	t(s)	Gap*(%)	NT _{UB}	NT _{YRS}	t _{YRS} (s)	Gap*(%)
20	15	5	21,50	22,20	0,25	1,42	21,20	20,90	153,13	0,00
20	15	10	9,45	10,19	4,29	15,24	8,20	8,20	767,57	0,00
20	20	5	25,70	26,40	0,48	5,33	24,40	24,20	244,99	0,00
20	20	10	13,00	14,12	13,78	22,64	10,60	10,60	1099,98	0,00
20	20	15	8,10	8,64	12,92	21,44	6,67	6,67	648,28	0,00
20	25	5	31,60	32,26	0,71	3,95	30,40	30,10	1913,17	0,00
20	25	10	19,40	20,38	15,23	25,97	15,40	15,40	1172,39	0,00
20	25	15	24,68	25,83	15,67	15,17	21,43	21,33	1493,62	0,00
20	25	20	7,55	8,25	21,37	22,17	6,18	6,15	352,11	0,00
25	15	10	7,30	8,28	9,70	23,73	5,90	5,90	2880,01	0,00
25	20	10	15,40	16,76	19,76	32,76	11,60	11,60	3240,01	0,00
25	20	15	10,30	11,30	9,43	35,53	7,60	7,60	2913,22	0,00
25	25	10	21,90	23,22	21,45	31,93	16,60	16,60	2520,03	0,00
25	25	15	13,20	14,32	52,44	32,00	10,00	10,00	0,09	0,00
25	25	20	7,10	7,87	30,99	29,09	5,50	5,50	1033,95	0,00
Média			15,75	16,67	15,23	21,22	13,44	13,38	1362,17	0,00

Tabela 4.15: Resultados relativos ao Grupo E - RD3.

J	T	C	RD3				CSY			
			NT*	NT	t(s)	Gap*(%)	NT _{UB}	NT _{YRS}	t _{YRS} (s)	Gap*(%)
10	10	4	9,70	9,76	0,04	2,11	9,50	9,50	0,00	0,00
10	10	5	6,30	6,50	0,13	1,61	6,20	6,20	0,01	0,00
10	10	6	4,40	4,58	0,24	2,33	4,30	4,30	0,01	0,00
10	10	7	3,00	3,04	0,28	0,00	3,00	3,00	0,00	0,00
15	20	6	22,60	22,86	0,60	5,61	21,40	21,40	1,36	0,00
15	20	8	15,80	16,48	3,65	10,49	14,30	14,20	2,37	0,00
15	20	10	11,30	11,82	5,57	9,71	10,30	10,30	1,34	0,00
15	20	12	9,10	9,70	9,33	10,98	8,20	8,20	114,76	0,00
Média			10,28	10,59	2,48	5,35	9,65	9,64	14,98	0,00

diferença média entre as colunas NT^* e NT_{UB} da Tabela 4.14 ainda é igual a 2 unidades de troca, aproximadamente. Após investigarmos o conjunto de dados gerados pelos testes computacionais, percebemos que isso se deve a um conjunto de 16 instâncias do Grupo D, em que há uma distância considerável e persistente entre *clusters* contendo o nó popular x e *clusters* ditos isolados. Ressaltamos, entretanto, a eficiência de nosso método com relação ao tempo computacional, gastando em média 15,23 segundos no grupo D.

4.2.5 Experimento 4: *Path relinking*

Nesta seção, avaliamos a última estratégia aplicada à geração dos nós do grafo MTSP/GTSP, denominada RD4. Utilizamos a linguagem de programação C++ e testamos os algoritmos em uma máquina com as mesmas configurações descritas na seção anterior.

Os resultados obtidos são sumarizados nas Tabelas 4.16 a 4.20. A heurística de Chaves *et al.* (2012) é representada pelas colunas CSY, enquanto as siglas NT^* , $t(s)$ e $Gap^*(\%)$ se referem à média do número de trocas ao considerarmos as melhores soluções, ao tempo computacional médio, em segundos, e ao desvio médio relativo ao melhor valor de função objetivo encontrado, respectivamente. Os experimentos computacionais mostram que a decisão de povoamento dos *clusters* através de um processo de busca baseado no método *Path relinking* é assertiva quanto à diminuição dos *gaps* de otimalidade. De fato, registramos os menores *gaps* já obtidos pela heurística de redução de domínio, para os cinco grupos de instâncias. Além disso, o tempo computacional médio exigido se mantém baixo, variando entre o mínimo de 2,17 segundos no Grupo E e o máximo de 20,95 segundos no Grupo D.

Tabela 4.16: Resultados relativos ao Grupo A - RD4.

J	T	C	RD4			CSY		
			NT*	t(s)	Gap*(%)	NT*	t(s)	Gap*(%)
8	15	5	12,00	0,86	0,00	12,00	0,00	0,00
8	15	10	6,83	13,56	0,00	6,83	0,00	0,00
8	20	5	16,80	2,47	0,00	16,80	0,06	0,00
8	20	10	13,23	25,46	1,27	13,07	0,02	0,00
8	20	15	7,25	15,90	2,35	7,08	0,01	0,00
8	25	5	20,10	9,31	0,00	20,10	0,07	0,00
8	25	10	18,57	49,47	2,01	18,20	0,04	0,00
8	25	15	13,75	28,07	6,18	12,95	0,01	0,00
8	25	20	6,96	23,44	5,30	6,61	0,01	0,00
Média			12,83	18,73	1,90	12,63	0,02	0,00

Tabela 4.17: Resultados relativos ao Grupo B - RD4.

J	T	C	RD4			CSY		
			NT*	t(s)	Gap*(%)	NT*	t(s)	Gap*(%)
9	15	5	12,20	0,81	0,00	12,20	0,02	0,00
9	15	10	7,43	6,00	0,90	7,37	0,00	0,00
9	20	5	17,40	5,97	0,00	17,40	0,11	0,00
9	20	10	14,67	20,45	3,53	14,17	0,02	0,00
9	20	15	7,83	8,26	3,07	7,60	0,01	0,00
9	25	5	20,40	5,06	0,00	20,40	0,16	0,00
9	25	10	19,30	45,22	2,84	18,77	0,05	0,00
9	25	15	15,74	17,24	6,78	14,74	0,04	0,00
9	25	20	7,47	12,79	3,89	7,19	0,02	0,00
Média			13,60	13,53	2,34	13,31	0,05	0,00

Tabela 4.18: Resultados relativos ao Grupo C - RD4.

J	T	C	RD4			CSY		
			NT*	t(s)	Gap*(%)	NT*	t(s)	Gap*(%)
15	15	5	17,00	1,19	2,41	16,60	1,67	0,00
15	15	10	10,10	5,26	3,06	9,80	2,19	0,00
15	20	5	21,00	2,11	1,94	20,60	13,26	0,00
15	20	10	19,97	6,63	8,91	18,33	1,04	0,00
15	20	15	11,23	7,16	6,81	10,52	40,67	0,00
15	25	5	28,00	1,43	1,82	27,50	49,17	0,00
15	25	10	27,13	6,90	8,24	25,07	9,22	0,00
15	25	15	21,50	9,73	12,76	19,07	9,79	0,00
15	25	20	10,55	7,35	9,221	9,66	5,16	0,00
Média			18,50	5,31	6,13	17,46	14,69	0,00

Tabela 4.19: Resultados relativos ao Grupo D - RD4.

J	T	C	RD4			CSY		
			NT*	t(s)	Gap*(%)	NT*	t(s)	Gap*(%)
20	15	5	21,60	0,71	3,35	20,90	153,13	0,00
20	15	10	9,00	8,70	9,76	8,20	767,57	0,00
20	20	5	25,20	1,09	4,13	24,20	244,99	0,00
20	20	10	12,90	18,50	21,70	10,60	1099,98	0,00
20	20	15	7,60	15,28	14,00	6,67	648,28	0,00
20	25	5	31,20	1,00	3,65	30,10	1913,17	0,00
20	25	10	19,20	17,68	24,68	15,40	1172,39	0,00
20	25	15	24,55	16,33	15,12	21,33	1493,62	0,00
20	25	20	7,20	29,05	17,07	6,15	352,11	0,00
20	15	10	6,90	31,39	16,95	5,90	2880,01	0,00
20	20	10	15,10	33,68	30,17	11,60	3240,01	0,00
20	20	15	9,80	34,81	28,95	7,60	2913,22	0,00
20	25	10	21,40	33,91	28,92	16,60	2520,03	0,00
20	25	15	13,00	35,02	30,00	10,00	0,09	0,00
20	25	10	6,73	37,27	22,42	5,50	1033,95	0,00
Média			15,43	20,95	18,06	13,38	1362,17	0,00

Tabela 4.20: Resultados relativos ao Grupo E - RD4.

J	T	C	RD4			CSY		
			NT*	t(s)	Gap*(%)	NT*	t(s)	Gap*(%)
10	10	4	9,50	0,03	0,00	9,50	0,00	0,00
10	10	5	6,20	0,16	0,00	6,20	0,01	0,00
10	10	6	4,30	0,30	0,00	4,30	0,01	0,00
10	10	7	3,00	0,31	0,00	3,00	0,00	0,00
10	20	6	22,30	1,06	4,21	21,40	1,36	0,00
10	20	8	16,00	3,66	12,68	14,20	2,37	0,00
10	20	10	11,40	4,65	10,68	10,30	1,34	0,00
10	20	12	9,00	7,18	9,76	8,20	114,76	0,00
Média			10,21	2,17	4,66	9,64	14,98	0,00

4.3 Conclusões

Discutimos, neste capítulo, o problema da otimização das trocas de ferramentas em uma única máquina flexível de manufatura. Propomos quatro versões de um algoritmo de redução de domínio com o intuito de viabilizar a solução do MTSP por meio de métodos desenvolvidos para o problema do caixeiro viajante. Estas versões foram testadas em um conjunto de 1350 instâncias contendo até 20 tarefas e 25 ferramentas. Conforme mostrado pelos experimentos computacionais, nosso método é competitivo em instâncias de pequeno e médio portes, além de computacionalmente rápido, mas ainda apresenta, em média, um acréscimo de 13,29% no número de trocas de ferramentas para o grupo de instâncias consideradas grandes. Diante disso, como uma proposta de pesquisa futura, pretendemos investigar a fundo o conjunto de instâncias discrepantes, em busca de possíveis padrões combinatoriais ainda imunes às estratégias atuais inseridas em nossa redução de domínio.

Capítulo 5

IPMTC

Para se realizar uma boa programação para um SFM, é necessário considerar diversos elementos que estão interrelacionados: carregamento de máquinas, roteamento de tarefas, planejamento e alocação de ferramentas, carregamento e roteamento de veículos automaticamente guiados (AGV, do inglês *Automated Guided Vehicles*) e sincronização de atividades. O carregamento de máquinas consiste na atribuição de tarefas às máquinas, respeitando a limitação dos *buffers* a elas associados. O roteamento de tarefas gera a sequência de máquinas que cada tarefa deve visitar a fim de ter seu processamento finalizado. O planejamento do conjunto de ferramentas determina a quantidade necessária de cada tipo de ferramenta, com o intuito de atender às necessidades do sistema de produção, enquanto a alocação de ferramentas lida com a atribuição de ferramentas aos magazines das máquinas. O carregamento e o roteamento de AGVs estabelecem a rota a ser seguida por esse dispositivo, bem como a quantidade de ferramentas e as tarefas a serem nele transportadas. Finalmente, as decisões de sincronização especificam o início e a conclusão de cada atividade da produção.

Neste capítulo, nos concentramos nas componentes relacionadas ao carregamento de máquinas e à alocação de ferramentas em um SFM¹. Mais precisamente, o IPMTC trata do processamento de um conjunto de tarefas em um conjunto de máquinas paralelas idênticas, a fim de minimizar o *makespan*, ou seja, o tempo total exigido para o processamento de todas as tarefas. Na Seção 5.1, propomos duas modelagens matemáticas inteiras para o problema. Detalhes da meta-heurística ALNS são apresentados na Seção 5.2, como a determinação de uma solução inicial (Seção 5.2.1) e características dos operadores de remoção e inserção utilizados (Seção 5.2.2). A Seção 5.3 contém os resultados obtidos a partir de uma gama extensa de experimentos computacionais. Com a Seção 5.4, encerramos o capítulo apresentando conclusões sobre o trabalho realizado.

5.1 Formulações matemáticas

A seguir, introduzimos duas formulações matemáticas para suprir a falta de modelagens para o IPMTC e, como consequência, melhor defini-lo. O primeiro modelo, chamado M_1 , e inspirado no trabalho de Tang e Denardo (1988a) para o MTSP, utiliza três famílias de variáveis binárias. Definimos w_{jkm} igual a 1 se, e somente se, a tarefa j é processada pela máquina m na posição k , y_{tkm} igual a 1 se, e somente se, a ferramenta t está alocada na máquina m durante o processamento de sua k -ésima tarefa, e z_{tkm} igual a 1 se, e somente se, a ferramenta t é inserida na magazine da máquina m exatamente antes do início do processamento da tarefa na posição k . Sendo J_t o conjunto de tarefas que requerem a ferramenta t , Δ_m o tempo de finalização dos trabalhos em cada máquina m , e Δ o *makespan*, o modelo é dado por:

$$M1: \text{Min } \Delta \tag{5.1}$$

¹Este capítulo é fortemente baseado no artigo “*Scheduling identical parallel machines with tooling constraints*”, de autoria de Andreza Cristina Beezão, Jean-François Cordeau, Gilbert Laporte e Horacio Hideki Yanasse, aceito para publicação no *European Journal of Operational Research* e disponível *online* desde 11 de agosto de 2016.

sujeito a

$$\sum_{j=1}^n w_{jkm} \leq 1 \quad k \in J, m \in M \quad (5.2)$$

$$\sum_{m=1}^p \sum_{k=1}^n w_{jkm} = 1 \quad j \in J \quad (5.3)$$

$$\sum_{j=1}^n w_{jkm} \leq \sum_{j=1}^n w_{j(k-1)m} \quad k \in J \setminus \{1\}, m \in M \quad (5.4)$$

$$\sum_{j \in J_t} w_{jkm} \leq y_{tkm} \quad t \in T, k \in J, m \in M \quad (5.5)$$

$$\sum_{t=1}^l y_{tkm} \leq C \quad k \in J, m \in M \quad (5.6)$$

$$y_{tkm} - y_{t(k-1)m} \leq z_{tkm} \quad t \in T, k \in J \setminus \{1\}, m \in M \quad (5.7)$$

$$\sum_{k=1}^n \sum_{j=1}^n p_j w_{jkm} + \bar{t} \sum_{k=2}^n \sum_{t=1}^l z_{tkm} \leq \Delta_m \quad m \in M \quad (5.8)$$

$$\Delta_m \leq \Delta \quad m \in M \quad (5.9)$$

$$w_{jkm} \in \{0, 1\} \quad j \in J, k \in J, m \in M \quad (5.10)$$

$$y_{tkm} \in \{0, 1\} \quad t \in T, k \in J, m \in M \quad (5.11)$$

$$z_{tkm} \in \{0, 1\} \quad t \in T, k \in J, m \in M. \quad (5.12)$$

A função objetivo (5.1) minimiza o *makespan*. As restrições (5.2) e (5.3) designam as tarefas às posições nas máquinas, enquanto as restrições (5.4) impõem que uma tarefa pode ser executada em uma máquina m na posição k somente após o término da $(k - 1)$ -ésima tarefa processada em m . As restrições (5.5) e (5.6) dizem respeito às trocas de ferramentas: todas as ferramentas requeridas por uma tarefa devem estar na máquina no momento da sua execução e a capacidade do magazine de cada máquina nunca deve ser excedida. A inserção de ferramentas nos magazines é controlada pelas restrições (5.7). As restrições (5.8) calculam o instante em que as atividades de cada máquina são finalizadas e o *makespan* é definido pelas restrições (5.9). Por fim, as restrições (5.10) - (5.12) indicam as condições de integralidade das variáveis de decisão.

Para introduzir o segundo modelo, M_2 , inspirado no artigo de Laporte *et al.* (2004), definimos as variáveis binárias x_{ijm} igual a 1 se, e somente se, a tarefa i é imediatamente seguida pela tarefa j na máquina m , α_{jm} (ω_{jm}) igual a 1 se, e somente se, a tarefa j é a primeira (última) a ser processada na máquina m , β_{jm} igual a 1 se, e somente se, a tarefa j é designada à máquina m , y_{jt} igual a 1 se, e somente se, a ferramenta t está no magazine durante o processamento da tarefa j e z_{jt} igual a 1 se, e somente se, a ferramenta t é inserida no momento do processamento da tarefa j . Definindo por Δ_j o instante em que a tarefa j é finalizada, Δ o *makespan* e $M = \sum_{j \in J} p_j + \bar{t}|J|C$

uma constante suficientemente grande, o modelo segue:

$$\text{M2: Min } \Delta \quad (5.13)$$

sujeito a

$$\sum_{m=1}^p \sum_{j=1, j \neq i}^n x_{ijm} \leq 1 \quad i \in J \quad (5.14)$$

$$\sum_{m=1}^p \sum_{i=1, i \neq j}^n x_{ijm} \leq 1 \quad j \in J \quad (5.15)$$

$$\sum_{k=1}^n x_{jkm} - \sum_{i=1}^n x_{ijm} \leq \alpha_{jm} \quad j \in J, m \in M \quad (5.16)$$

$$\sum_{i=1}^n x_{ijm} - \sum_{k=1}^n x_{jkm} \leq \omega_{jm} \quad j \in J, m \in M \quad (5.17)$$

$$\sum_{t=1}^l y_{jt} \leq C \quad j \in J \quad (5.18)$$

$$x_{ijm} + y_{jt} - y_{it} \leq z_{jt} + 1 \quad i \in J, j \in J \setminus \{i\}, m \in M, t \in T \quad (5.19)$$

$$\sum_{j=1}^n \alpha_{jm} \leq 1 \quad m \in M \quad (5.20)$$

$$\sum_{j=1}^n \omega_{jm} \leq 1 \quad m \in M \quad (5.21)$$

$$\sum_{m=1}^p \alpha_{jm} \leq 1 \quad j \in J \quad (5.22)$$

$$\sum_{m=1}^p \omega_{jm} \leq 1 \quad j \in J \quad (5.23)$$

$$\sum_{m=1}^p \beta_{jm} = 1 \quad j \in J \quad (5.24)$$

$$\alpha_{jm} \leq \beta_{jm} \quad j \in J, m \in M \quad (5.25)$$

$$\omega_{jm} \leq \beta_{jm} \quad j \in J, m \in M \quad (5.26)$$

$$2x_{ijm} \leq \beta_{im} + \beta_{jm} \quad i \in J, j \in J \setminus \{i\}, m \in M \quad (5.27)$$

$$\sum_{i=1}^n \sum_{j=1, j \neq i}^n x_{ijm} + 1 = \sum_{i=1}^n \beta_{im} \quad m \in M \quad (5.28)$$

$$\Delta_j \geq \Delta_i + p_j - M(1 - x_{ijm}) + \bar{t} \sum_{t \in T_j} z_{jt} \quad i \in J, j \in J \setminus \{i\}, m \in M \quad (5.29)$$

$$\Delta_j \geq p_j + \bar{t} \sum_{t \in T_j} z_{jt} \quad j \in J \quad (5.30)$$

$$\Delta_j \leq \Delta \quad j \in J \quad (5.31)$$

$$y_{jt} = 1 \quad j \in J, t \in T_j \quad (5.32)$$

$$z_{jt} = 0 \quad j \in J, t \in T \setminus T_j \quad (5.33)$$

$$x_{ijm} \in \{0, 1\} \quad i, j \in J, m \in M \quad (5.34)$$

$$\alpha_{jm} \in \{0, 1\} \quad j \in J, m \in M \quad (5.35)$$

$$\omega_{jm} \in \{0, 1\} \quad j \in J, m \in M \quad (5.36)$$

$$\beta_{jm} \in \{0, 1\} \quad j \in J, m \in M \quad (5.37)$$

$$y_{jt} \in \{0, 1\} \quad j \in J, t \in T \quad (5.38)$$

$$z_{jt} \in \{0, 1\} \quad j \in J, t \in T. \quad (5.39)$$

A função objetivo minimiza o *makespan*. As restrições (5.14) e (5.15) asseguram que cada tarefa tenha, no máximo, um predecessor e um sucessor, enquanto as restrições (5.16) e (5.17) determinam a primeira e a última tarefa a serem processadas em cada máquina. As restrições (5.18) garantem

o respeito à limitação dos magazines. As trocas de ferramentas são representadas pelas variáveis z_{jt} , apropriadamente definidas pelas restrições (5.19). As restrições (5.20) e (5.21) afirmam que cada máquina deve ter apenas uma tarefa inicial e uma tarefa final, respectivamente, enquanto as restrições (5.22) e (5.23) estabelecem que uma tarefa não pode ser a primeira (ou última) processada em máquinas distintas. As restrições (5.24) - (5.26) são restrições de designação, garantindo que cada tarefa seja executada em exatamente uma máquina, enquanto as restrições (5.27) afirmam que um par de tarefas sucessivas devem ser processadas na mesma máquina. As restrições (5.28) relacionam as variáveis x_{ijm} e β_{jm} , e o *makespan* é definido através do conjunto de restrições (5.29) - (5.31). As restrições (5.32) garantem a disponibilidade de todas as ferramentas requeridas por uma tarefa durante o seu processamento e as restrições (5.33) estabelecem que uma troca de ferramenta seja contabilizada somente quando uma ferramenta é inserida para ser imediatamente utilizada por uma tarefa. Por fim, as restrições (5.34) - (5.39) tratam do domínio das variáveis de decisão.

Como veremos na Seção 5.3.3.1, ambas as formulações apresentaram dificuldades computacionais para a resolução de exemplares pequenos do problema, com até 15 tarefas. Motivados por este fato, na seção seguinte desenvolvemos uma meta-heurística ALNS com o objetivo de fornecer boas soluções para exemplares maiores do IPMTC, em um tempo computacional aceitável.

5.2 Meta-heurística ALNS

Nesta seção, descrevemos os detalhes de uma meta-heurística *Adaptive large neighborhood search* (ALNS), aplicada à resolução do IPMTC. Este método foi proposto por Ropke e Pisinger (2006) e tem produzido excelentes resultados para uma grande variedade de problemas. Em sua versão clássica, cada iteração da ALNS consiste na aplicação de um operador de remoção e outro de inserção (ou operadores de destruição e reparo, como preferem alguns autores). Estes, por sua vez, são escolhidos independentemente um do outro e conforme um mecanismo ponderado de seleção, que depende do sucesso de cada operador em iterações anteriores (Pisinger e Ropke, 2007). O processo de busca é dividido em um número de segmentos com α iterações. Ao final de um segmento, os pesos são atualizados conforme placares acumulados durante as últimas α iterações e, ainda, de maneira proporcional a um fator de reação r que controla a rapidez com que o ajuste dos pesos reage a alterações no sucesso dos operadores. Após a aplicação de um operador de remoção e outro de inserção, seus placares são acrescidos de um valor α_1 , α_2 ou α_3 , dependentes da pesquisa ter atingido a melhor solução encontrada até o momento, uma solução apenas melhor que a atual ou uma solução pior, mas ainda aceitável, respectivamente. Representemos por $\Delta(s)$ o *makespan* decorrente da solução s . A solução s' é sempre aceitável quando $\Delta(s') < \Delta(s)$, e aceitável com probabilidade $e^{-(\Delta(s')-\Delta(s))/\theta}$, em que θ é a temperatura, caso contrário. A temperatura inicial θ_0 é determinada por $(0,05\Delta_0)/\ln 2$, de modo que uma solução com valor 5% pior do que o *makespan* inicial Δ_0 é aceita com probabilidade 0,5. Sendo assim, dados h operadores com pesos ω_i , $i \in H = \{1, \dots, h\}$, o operador $j \in H$ é selecionado com probabilidade $\omega_j / \sum_{i \in H} \omega_i$.

Neste trabalho, implementamos nove operadores de remoção e cinco operadores de inserção considerando características tanto do PM quanto do MTSP. A estrutura principal da ALNS se baseia no mecanismo *Simulated annealing* (SA) e o critério de parada é determinado por um número máximo de iterações e uma temperatura mínima θ igual a 0,01, a qual parte de θ_0 e diminui a cada iteração de acordo com um fator de resfriamento $c \in (0, 1)$. Os passos principais da ALNS são apresentados no Algoritmo 8. Para o leitor interessado em mais detalhes a respeito de suposições e especificações da ALNS, recomendamos os trabalhos de Ropke e Pisinger (2006) e Pisinger e Ropke (2007).

5.2.1 Solução inicial

Propomos, nesta seção, um procedimento para determinação de uma solução inicial para a meta-heurística ALNS. Para tanto, combinamos a heurística construtiva desenvolvida por Chaves *et al.*

Algoritmo 8 Passos principais da estrutura de uma ALNS.

```

1: Seja  $s$  uma solução inicial;
2: Inicialize os placares  $\alpha_1, \alpha_2, \alpha_3$ ;
3:  $s^* \leftarrow s$ ;
4: repetir
5:   Escolha operadores de remoção e inserção,  $N^-$  and  $N^+$ , conforme o mecanismo ponderado de seleção;
6:   Gere uma nova solução  $s'$ , a partir de  $s$ , aplicando  $N^-$  e  $N^+$ ;
7:   se  $\Delta(s') < \Delta(s)$  então
8:      $s \leftarrow s'$ ;
9:   senão se  $\Delta(s') > \Delta(s)$  e  $s'$  é aceitável então
10:     $s \leftarrow s'$ ;
11:   fim
12:   se  $\Delta(s') < \Delta(s^*)$  então
13:      $s^* \leftarrow s'$ ;
14:   fim
15: até Critério de parada é alcançado;
16: retorne  $s^*$  (melhor solução encontrada).
  
```

(2012) com um modelo inteiro, M_3 , que minimiza o *makespan* ao dividir a solução obtida para o MTSP em p segmentos, um para cada máquina.

A heurística para o MTSP mencionada anteriormente consiste de uma fase construtiva seguida por uma fase de melhoria. A primeira fase é baseada em um grafo em que os vértices correspondem a ferramentas e existe uma aresta $j = (t_1, t_2)$ conectando os vértices t_1 e t_2 se, e somente se, as ferramentas t_1 e t_2 são requeridas pela mesma tarefa j . O algoritmo prioriza a remoção de arestas de vértices com graus pequenos, a fim de produzir vértices isolados. Em outras palavras, esta estratégia gera uma ordem de processamento de tarefas, $s = \{s_1, \dots, s_n\}$, enquanto remove do magazine as ferramentas que já não são necessárias e que não mais acarretarão trocas de ferramentas. A segunda fase é um algoritmo de melhoria inspirado na meta-heurística *Iterated Local Search* (ILS), geralmente aplicada quando a intenção é gerar novas soluções a partir da perturbação das atuais. Primeiramente, a KTNS computa o número de trocas de ferramentas necessárias na solução s . A solução ótima local $\hat{s}$ é então determinada pela troca de cada par de tarefas pertencentes a s . Na sequência, o método modifica a ordem de execução de $\beta\%$, $\beta \in [20, 40]$, pares arbitrários de tarefas de $\hat{s}$, resultando em uma nova solução s' para a busca local. As tarefas são trocadas em pares, mais uma vez, produzindo a solução ótima local $\hat{s}'$. O algoritmo continua a partir da melhor solução entre $\hat{s}'$ e $\hat{s}$, e para após um máximo de 3000 iterações.

Uma vez que a solução do MTSP gerada pela heurística já é otimizada no quesito trocas de ferramentas, o *makespan* Δ a ser minimizado pelo modelo M_3 considera apenas tempos de processamento das tarefas. Essa escolha produz um ponto de partida de qualidade para a ALNS, com baixo custo computacional. Assumindo, sem perda de generalidade, que $s = \{s_1, \dots, s_n\}$ é a ordem de processamento das tarefas gerada pela heurística de Chaves *et al.* (2012) e definindo a variável binária x_{ij} igual a 1 quando há um segmento que começa com a tarefa na posição i e termina com a tarefa na posição j , sendo $i, j \in \{1, \dots, n\}$, o modelo é dado por:

$$M_3: \text{Min } \Delta \tag{5.40}$$

sujeito a

$$\sum_{i=1}^n \sum_{j=1}^n x_{ij} = p \tag{5.41}$$

$$\sum_{j=1}^n x_{1j} = 1 \quad (5.42)$$

$$\sum_{i=1}^n x_{in} = 1 \quad (5.43)$$

$$\sum_{j=1}^n x_{ij} \leq 1 \quad i = \{2, \dots, n\} \quad (5.44)$$

$$\sum_{i=1}^n x_{ij} \leq 1 \quad j = \{1, \dots, n-1\} \quad (5.45)$$

$$\sum_{k=j+1}^n x_{(j+1)k} \geq x_{ij} \quad i, j = \{1, \dots, n-1\} \quad (5.46)$$

$$\left(\sum_{k=i}^j p_{s_k} \right) x_{ij} \leq \Delta \quad i, j = \{1, \dots, n\} \quad (5.47)$$

$$x_{ij} \in \{0, 1\} \quad i, j = \{1, \dots, n\}. \quad (5.48)$$

A função objetivo (5.40) minimiza o *makespan*. A restrição (5.41) garante que o número de segmentos seja igual ao número de máquinas, p , enquanto as restrições (5.42) – (5.45) definem as extremidades possíveis de um segmento. A continuidade é assegurada pelas restrições (5.46): se existe um segmento de i a j , então deve existir $k \in \{j+1, \dots, n\}$ de tal forma que $x_{(j+1)k} = 1$. Por fim, o *makespan* é definido pelas restrições (5.47) e o domínio da variável de decisão é estabelecido nas restrições (5.48). Após a atribuição de tarefas ao conjunto de máquinas paralelas, o último passo trata da gestão de ferramentas e é solucionado pela aplicação da heurística KTNS em cada máquina, individualmente.

Denotamos por ALNS-B o algoritmo ALNS partindo da solução inicial descrita anteriormente e, por ALNS-R, o mesmo método combinado com soluções iniciais geradas de maneira randômica.

5.2.2 Operadores de remoção e inserção

No contexto do IPMTC, operadores de remoção (inserção) representam diferentes maneiras de se remover (inserir) uma quantidade q de tarefas de (em) uma ordem de processamento no conjunto de máquinas paralelas (Figura 5.1). Nesta seção, descrevemos nove operadores de remoção e cinco operadores de inserção desenvolvidos com o objetivo de intensificar e diversificar a busca por soluções.

Figura 5.1: Processo de remoção e inserção de tarefas.

5.2.2.1 Remoção randômica

Em um procedimento simples, este operador remove, randomicamente, q tarefas selecionadas da solução atual.

5.2.2.2 Remoção “Shaw”

Este operador foi inicialmente proposto por Shaw (1997). A ideia central é a remoção de tarefas consideradas similares e cujas reinserções poderiam levar a melhores soluções. Como medida de similaridade, consideramos a diferença absoluta entre a contribuição de cada tarefa à parcela da função objetivo relacionada ao tempo de processamento. A primeira tarefa j' é selecionada de forma randômica a partir de s . Então, o algoritmo calcula a *medida de parentesco* $R(j', j) = |p_{j'} - p_j|$ entre j' e todas as tarefas remanescentes $j \in \bar{J} = J \setminus \{j'\}$ a fim de escolher a próxima a ser removida. Este segundo processo de remoção também possui randomicidade, a qual é controlada por um parâmetro $\delta \geq 1$ que evita que os operadores de remoção sejam totalmente determinísticos, como explicitado nos passos 8–10 do Algoritmo 9.

Algoritmo 9 Randomicidade durante a fase de remoção de tarefas.

entradas $\delta \geq 1$;

- 1: Seja s a solução atual;
 - 2: Seja D um conjunto de tarefas removidas;
 - 3: **enquanto** $|D| < q$ **faça**
 - 4: Seja j' uma tarefa selecionada randomicamente a partir de s ;
 - 5: Seja $\bar{J}$ o conjunto de tarefas remanescentes;
 - 6: Ordene $\bar{J}$ tal que $j_1 < j_2$ quando $R(j', j_1) < R(j', j_2)$;
 - 7: Escolha um número randômico y em $[0, 1]$;
 - 8: $j'' \leftarrow \lfloor y^\delta |\bar{J}| \rfloor$;
 - 9: $D \leftarrow D \cup \{j''\}$;
 - 10: Remova a tarefa j'' da solução s ;
 - 11: **fim**
-

5.2.2.3 Remoção de tarefa crítica

Este operador remove tarefas com alto custo na solução atual s , na esperança de reinseri-las em melhores posições (Figura 5.2). Consideremos $c(s, j') = f(s) - f_{-j'}(s)$ o custo de uma tarefa j' , em que f_{-j} representa o *makespan* gerado pela solução s quando a tarefa j é considerada removida. O algoritmo remove uma tarefa de uma lista ordenada, em sentido não-crescente, de acordo com seus custos à função objetivo. A randomicidade durante a seleção de tarefas é garantida pelo parâmetro $\delta \geq 1$ (Algoritmo 9).

5.2.2.4 Remoção de maior desvio

A melhor solução do problema de designação de tarefas a máquinas não é, necessariamente, a melhor solução do IPMTC, uma vez que não são consideradas restrições de trocas de ferramentas. Baseado neste fato, este operador remove tarefas que gerem um grande desvio no valor do *makespan* quando avaliamos a soma dos tempos de processamento com o tempo gasto com trocas de ferramentas, em comparação ao considerarmos apenas os tempos de processamento. O número de trocas entre duas tarefas j_1 e j_2 é estimado através do *lower bound* $LB = \max\{0, |T_{j_1} \cup T_{j_2}| - C\}$, sendo T_j o conjunto das ferramentas requeridas pela tarefa j .

5.2.2.5 Remoção randômica de máquina

Com o intuito de diversificar ainda mais o processo de busca, este operador remove tarefas de uma máquina m selecionada randomicamente. Se o número de tarefas atribuídas a m for menor que q , o método procede para uma nova máquina, também selecionada de forma randômica.

5.2.2.6 Remoção pseudo-randômica de máquina

A primeira máquina e a primeira tarefa removida são selecionadas randomicamente. As próximas remoções, contudo, priorizam tarefas atribuídas a esta máquina e que sejam mais similares àquelas

Figura 5.2: Remoção de tarefas consideradas críticas.

tarefas já removidas. O conceito de parentesco é o mesmo utilizado no operador *Remoção “Shaw”*.

5.2.2.7 Remoção de máquina crítica

Este operador escolhe, primeiramente, a máquina m responsável pelo valor do *makespan*, isto é, tal que $\Delta_m = \Delta$. Em seguida, segue o mesmo processo de remoção descrito no operador anterior. O objetivo, aqui, é impor a remoção de variáveis que “atrapalhem” a estrutura da solução atual, ou seja, que aumentem consideravelmente o valor da função objetivo.

5.2.2.8 Remoção - Grafo de vizinhança

A ideia principal deste operador é a remoção de tarefas atribuídas a máquinas impróprias. Como ilustrado na Figura 5.3, definimos um grafo completo em que o peso de cada aresta (j_1, j_2) representa o melhor valor de função objetivo obtido em uma solução em que j_1 e j_2 são processadas consecutivamente e em uma mesma máquina. O placar de um nó (tarefa) é dado pela soma dos pesos das arestas nele incidentes e as tarefas com placares altos são mais prováveis de serem selecionadas para remoção. O parâmetro $\delta \geq 1$ controla a randomicidade no processo de remoção de tarefas (Algoritmo 9).

5.2.2.9 Remoção - Grafo de requisições

Similarmente ao método anterior, o custo de uma aresta (j_1, j_2) pertencente ao grafo de requisições é dado pelo número de vezes em que as tarefas j_1 and j_2 são processadas em uma mesma máquina, considerando as 100 melhores soluções encontradas no processo de busca. As tarefas com pequenos placares são mais suscetíveis à remoção, pois são priorizadas as remoções de arestas pouco promissoras (Figura 5.4). As remoções são randômicas e controladas pelo parâmetro $\delta \geq 1$ (Algoritmo 9).

5.2.2.10 Inserção gulosa_p

Neste operador, q tarefas são inseridas, conforme a ordem de remoção, em posições que gerem o mínimo aumento possível no tempo total de processamento em cada máquina.

Figura 5.3: Ilustração do funcionamento do operador “Remoção - Grafo de vizinhança”.

Figura 5.4: Ilustração do funcionamento do operador “Remoção - Grafo de requisições”.

5.2.2.11 Inserção gulosa_Δ

Este operador possui uma diferença sutil do operador anterior. As tarefas são inseridas respeitando a ordem de remoção e em posições que menos aumentem o valor total do *makespan* Δ da solução atual.

5.2.2.12 Inserção gulosa_p - Look-ahead

Neste operador, a tarefa j escolhida para ser inserida possui a posição com menor custo global, considerando apenas os tempos de processamento. Mais especificamente, seja $\epsilon_p^+(j)$ a mudança no tempo total de processamento devido à inserção de tal tarefa j . A próxima inserção procura pela tarefa $\tilde{j}$ tal que $\epsilon_p^+(\tilde{j}) = \min_{j \in Q} \epsilon_p^+(j)$, sendo Q é o conjunto de tarefas removidas anteriormente.

5.2.2.13 Inserção gulosa_Δ - Look-ahead

Este operador generaliza a *Inserção gulosa_p* anterior, no sentido de inserir a tarefa que possui a posição com menor custo global quando consideramos o *makespan* Δ como função objetivo.

5.2.2.14 Inserção 2-Regret

Movimentos gulosos costumam postergar a alocação de tarefas críticas para as iterações finais, justamente quando não há muita liberdade para movimentação (Pisinger e Ropke, 2007). Para lidar com esta questão, o movimento de “arrependimento” escolhe tarefas com maiores diferenças entre suas inserções na primeira e na segunda posições menos custosas à função objetivo.

5.3 Experimentos computacionais

Nesta seção, apresentamos todos os experimentos computacionais realizados para avaliar a performance do algoritmo ALNS desenvolvido, assim como para analisar os limites das formulações matemáticas M_1 e M_2 . Inicialmente, descrevemos o processo de geração de dois conjuntos de instâncias, IPMTC-I e IPMTC-II (Seção 5.3.1). Seguem, então, a etapa de calibração dos parâmetros da ALNS (Seção 5.3.2) e os resultados obtidos através dos modelos e do método de solução proposto (Seção 5.3.3).

5.3.1 Instâncias teste

Uma vez que o conjunto de 37 instâncias utilizadas por Fathi e Barnette (2002) não estava disponível, testamos nosso método em dois novos conjuntos de problemas. Primeiramente, geramos um conjunto de 1440 instâncias denominado IPMTC-I e adaptado de Yanasse *et al.* (2009). O primeiro passo desta adaptação se trata da geração de tempos de processamento aleatórios p_j ($j \in J$), por meio de uma distribuição uniforme no intervalo $[1, 59]$. O número de máquinas varia de 2 a 4 e o número de tarefas é 8, 15 ou 25 para $p = 2$, 15 ou 25 para $p = 3$, e 25 para $p = 4$. Cada configuração máquina-tarefa resulta em duas configurações máquina-tarefa-ferramenta, dependendo do número de ferramentas requeridas ser igual a 15 ou 20. Para seguir o padrão utilizado por Fathi e Barnette (2002), definimos três valores de tempos de troca dados por $s_0 = \text{rand}(p_{\min}, p_{\max})$, $s_1 = \lceil 1, 15p_{\max} \rceil$ e $s_2 = \lfloor 0, 75p_{\min} \rfloor$, em que $p_{\min}$ e $p_{\max}$ representam os valores mínimo e máximo dentre os tempos de processamento estabelecidos anteriormente e $\text{rand}(a, b)$ retorna um número uniformemente distribuído no intervalo $[a, b]$. Em seguida, tomando quatro casos para cada uma dessas combinações, totalizamos uma amostra de 144 configurações. Finalmente, geramos dez instâncias para cada configuração e obtemos o total de 1440 instâncias.

Em uma segunda etapa e com a intenção de explorar os limites da meta-heurística ALNS proposta, aumentamos as dimensões do problema propondo um conjunto de 1440 instâncias grandes, com 50, 100 e 200 tarefas, denominado IPMTC-II. O número de máquinas é 3, 4 ou 5 para $n = 50$, 4, 5, 6 ou 7 para $n = 100$, e 6, 7, 8, 9 ou 10 para $n = 200$. O número de ferramentas necessárias agora é dado por 30 ou 40 e a razão de proporcionalidade com a capacidade do magazine, C , é garantido pelo esquema descrito em Crama *et al.* (1994). Os processos de geração dos tempos de processamento e de troca continuam os mesmos. Obtemos, assim, um total de 1440 problemas através da geração de dez instâncias para cada duas combinações máquina-tarefa-ferramenta.

5.3.2 Etapa de calibração da ALNS

Para a calibragem dos parâmetros da ALNS, escolhemos um subconjunto de 30 instâncias representativas do conjunto IPMTC-I. Sejam it o número total de iterações, α o número de iterações consecutivas em um segmento, γ a percentagem de tarefas removidas a cada iteração da ALNS, δ o coeficiente que evita o determinismo na fase de remoção, c e r os fatores de resfriamento e reação relacionados ao SA, respectivamente, e a tupla $(\alpha_1, \alpha_2, \alpha_3)$ os parâmetros responsáveis pelo controle do mecanismo de ajuste dos pesos dos operadores. A fim de definir todos estes parâmetros, cada um assume vários valores determinados com antecedência (Tabela 5.1), enquanto os outros são mantidos fixos. Através de um processo iterativo, a meta-heurística ALNS é executada cinco vezes para cada configuração de parâmetros e a configuração relacionada às melhores médias de resultados é escolhida. Os valores finais de todos os parâmetros são apresentados na Tabela 5.2.

Tabela 5.1: Combinação de parâmetros analisados na fase de calibração da ALNS.

Parâmetro	Valores
it/α	10k/50; 10k/100; 25k/50; 25k/100; 25k/250; 100k/50; 100k/100; 100k/200; 100k/400; 100k/1000
γ	5%; 10%; 15%; 25%; 40%
δ	1, 0; 1, 4; 1, 8; 2, 2; 2, 5
c	0, 992; 0, 995; 0, 997; 0, 999; 0, 99975
r	0, 01; 0, 04; 0, 07; 0, 09; 0, 11
$(\alpha_1, \alpha_2, \alpha_3)$	(30, 10, 40); (50, 10, 30); (50, 15, 20); (50, 30, 5); (50, 30, 10); (50, 30, 20)

Nota: A abreviação k representa 1000 iterações.

Tabela 5.2: Valores finais dos parâmetros da ALNS.

Parâmetro	Valor
it	10k, 25k ou 100k
α	1% it
γ	10% ou 25%
δ	1, 8
c	0, 995
r	0, 01
$(\alpha_1, \alpha_2, \alpha_3)$	(50, 30, 20)

Nota: A abreviação k representa 1000 iterações.

5.3.3 Resultados

Os algoritmos foram implementados em C++ e os experimentos conduzidos em um computador 2.67 GHz Intel Xeon X55 com 12 Gb de memória. Utilizamos o *software* IBM CPLEX versão 12.6.2, com uma *thread* e tamanho máximo da árvore de busca igual a 6 Gb, para a solução dos modelos M_1 e M_2 , e o CPLEX versão 12.6, com as configurações padrão, para resolver o modelo inteiro-misto M_3 . O tempo limite imposto para a solução de cada instância foi de 4200 segundos.

Com o intuito de promover uma base de comparação, implementamos os dois melhores métodos de busca local propostos por Fathi e Barnette (2002), nomeados CLIP1 e CLIP2. Seus algoritmos começam por uma solução gerada randomicamente através de uma heurística baseada em listas de processamento. Analisamos, também, a aplicação da heurística de Chaves *et al.* (2012) combinada unicamente com o modelo M_3 . Este método é chamado Ch-B e é bastante útil para a avaliação da qualidade das buscas realizadas pelos operadores de remoção e inserção propostos.

Ainda, uma vez que o número de tarefas removidas de uma solução tem um impacto significativo sobre a precisão da ALNS, propomos seis cenários para testarmos os nossos métodos, os quais são obtidos pela variação do número de iterações entre 10, 25 e 100 mil e o número de remoções q igual a 10% ou 25% do número total de tarefas:

- Cenário 1 (c1): 10.000 iterações e 10% das tarefas removidas;
- Cenário 2 (c2): 10.000 iterações e 25% das tarefas removidas;
- Cenário 3 (c3): 25.000 iterações e 10% das tarefas removidas;
- Cenário 4 (c4): 25.000 iterações e 25% das tarefas removidas;
- Cenário 5 (c5): 100.000 iterações e 10% das tarefas removidas;
- Cenário 6 (c6): 100.000 iterações e 25% das tarefas removidas.

5.3.3.1 Experimento 1: formulações matemáticas

Prevendo dificuldades computacionais por parte das modelagens matemáticas M_1 e M_2 , optamos por testá-las no conjunto restrito de 30 instâncias representativas retiradas do conjunto

IPMTC-I e utilizadas para a calibração dos parâmetros da ALNS. A Tabela 5.3 sumariza os experimentos computacionais realizados. A primeira coluna é o número de tarefas ($|J|$) e a segunda mostra o percentual do *gap* de otimalidade (*Gap*). As próximas colunas representam o tempo computacional em segundos ($t(s)$), o número de nós explorados (*Nós*) e a porcentagem de soluções ótimas obtidas (*Opt*(%)). O modelo M_1 gera resultados de melhor qualidade, porém, como esperado, ambas formulações apresentam dificuldades já na resolução de instâncias com 15 tarefas. Isso reforça a necessidade do desenvolvimento de meta-heurísticas que produzam boas soluções em um tempo computacional razoável.

Tabela 5.3: Análise comparativa do desempenho das formulações matemáticas

$ J $	M_1				M_2			
	Gap(%)	t(s)	Nós	Opt(%)	Gap(%)	t(s)	Nós	Opt(%)
8	0	43	14692	100	0	220	44103	100
15	32	3600	341295	0	88	3536	79118	0
25	18	3284	104915	10	89	3600	12749	0
Média	17	2309	153634		59	2452	45323	

5.3.3.2 Experimento 2: ALNS no conjunto de instâncias IPMTC-I

Para garantir comparações justas, todos os métodos mencionados anteriormente foram executados dez vezes para cada instância do conjunto IPMTC-I, sob às mesmas configurações de máquina. Os resultados computacionais são resumidos nas Tabelas 5.4 a 5.8. As Tabelas 5.4 e 5.5 apresentam os *gaps* relativos à melhor solução encontrada por qualquer um dos quatro métodos CLIP1, CLIP2, ALNS-B e ALNS-R para uma dada instância, considerando nosso pior e nosso melhor cenário, c1 e c4, respectivamente. As três primeiras colunas são o número de máquinas ($|M|$), tarefas ($|J|$) e ferramentas requeridas ($|T|$). O desvio percentual médio é relatado nas colunas *Gap* e o desvio percentual em relação à melhor solução encontrada é apresentado nas colunas *Gap**. Nosso algoritmo produz melhores resultados que CLIP1 e CLIP2 em todas as classes de instâncias, sendo obtidas as melhores médias em cenários em que a porcentagem de tarefas removidas é igual a 25% e o método tem mais liberdade de ação. O uso de tempos de troca distintos não parece afetar a qualidade da solução final obtida.

Tabela 5.4: Gap relativo à melhor solução encontrada considerando os métodos CLIP1, CLIP2, ALNS-B e ALNS-R - Cenário 1.

$ M $	$ J $	$ T $	CLIP1		CLIP2		ALNS-B		ALNS-R	
			Gap	Gap*	Gap	Gap*	Gap	Gap*	Gap	Gap*
2	8	15	12,23	2,20	10,55	1,90	2,94	0,65	4,51	0,41
2	8	20	13,50	2,08	12,10	1,77	2,66	0,47	6,45	0,13
2	15	15	18,75	8,94	16,77	7,74	0,41	0,01	0,43	0,02
2	15	20	16,64	7,21	14,29	5,45	0,34	0,01	0,38	0,00
2	25	15	17,83	8,97	16,28	6,48	0,16	0,01	0,19	0,02
2	25	20	19,51	10,46	17,76	8,60	0,26	0,02	0,28	0,03
3	15	15	18,62	6,56	14,04	4,30	0,69	0,05	0,70	0,04
3	15	20	14,57	5,61	11,77	3,60	0,71	0,04	0,68	0,02
3	25	15	20,65	9,69	19,61	7,35	1,71	0,08	1,83	0,21
3	25	20	20,97	10,58	18,36	7,05	1,31	0,11	1,43	0,09
4	25	15	15,32	4,57	13,58	3,10	0,60	0,06	0,65	0,10
4	25	20	25,44	11,20	23,18	8,80	2,55	0,18	2,81	0,29
Média			17,84	7,34	15,69	5,51	1,19	0,14	1,70	0,11

No que diz respeito ao tempo de execução, a ALNS é mais lenta que os algoritmos CLIP1 e CLIP2, uma vez que seus componentes requerem mais tempo do que um procedimento de busca local. Apesar disso, o maior tempo de processamento registrado pela ALNS-B é bastante razoável: menos de dez minutos, em média, considerando os seis cenários.

Como citado anteriormente, o critério de parada da ALNS é o número de iterações realizadas

Tabela 5.5: Gap relativo à melhor solução encontrada considerando os métodos CLIP1, CLIP2, ALNS-B e ALNS-R - Cenário 4.

M	J	T	CLIP1		CLIP2		ALNS-B		ALNS-R	
			Gap	Gap*	Gap	Gap*	Gap	Gap*	Gap	Gap*
2	8	15	12,25	2,22	10,58	1,92	0,00	0,00	0,00	0,00
2	8	20	13,52	2,10	12,12	1,78	0,00	0,00	0,00	0,00
2	15	15	18,76	8,95	16,78	7,74	0,02	0,00	0,02	0,00
2	15	20	16,64	7,22	14,29	5,45	0,05	0,00	0,04	0,00
2	25	15	17,86	9,00	16,31	6,51	0,07	0,01	0,08	0,01
2	25	20	19,52	10,48	17,77	8,61	0,11	0,02	0,11	0,02
3	15	15	18,63	6,57	14,05	4,31	0,10	0,00	0,09	0,00
3	15	20	14,57	5,61	11,78	3,60	0,06	0,00	0,06	0,00
3	25	15	20,75	9,78	19,71	7,44	0,40	0,04	0,41	0,04
3	25	20	21,12	10,72	18,51	7,19	0,30	0,02	0,32	0,05
4	25	15	15,47	4,71	13,72	3,24	0,34	0,05	0,34	0,06
4	25	20	25,76	11,48	23,50	9,08	0,70	0,06	0,66	0,05
Média			17,90	7,40	15,76	5,57	0,18	0,02	0,18	0,02

e a temperatura do mecanismo de SA. Testes diversos mostraram que 25.000 iterações geralmente fornecem um bom *trade-off* entre o tempo computacional e a qualidade da solução obtida. Como indicado na Tabela 5.2, o número de iterações consecutivas em um segmento da meta-heurística é proporcional ao número total de iterações realizadas. Isto também é verdade quanto ao número de vezes que os placares, os pesos e as probabilidades são ajustados durante o processo de seleção dos operadores de remoção e inserção. Portanto, em geral, um grande número de iterações não necessariamente conduz aos melhores resultados.

Tabela 5.6: Comparação do desempenho de todos os métodos e cenários analisados.

Método	Gap(%)	Gap*(%)	t(s)	t*(s)	Imp
ALNS-B - c1	9,82	8,55	33	11	5,58
ALNS-B - c2	8,68	8,40	84	28	5,72
ALNS-B - c3	9,39	8,49	69	22	5,90
ALNS-B - c4	8,53	8,36	196	58	5,87
ALNS-B - c5	9,10	8,44	167	45	6,07
ALNS-B - c6	8,51	8,36	534	123	5,88
ALNS-R - c1	10,48	8,52	115	50	11,39
ALNS-R - c2	8,69	8,40	346	135	9,85
ALNS-R - c3	9,95	8,43	285	102	11,71
ALNS-R - c4	8,53	8,37	867	283	10,08
ALNS-R - c5	9,70	8,40	738	220	11,93
ALNS-R - c6	8,51	8,36	1198	332	10,07
Ch-B	30,28	23,72	35	-	-
CLIP1	27,86	16,22	0	0	4,06
CLIP2	25,47	14,30	0	0	4,47

Nota: Números em negrito indicam a porcentagem mínima de desvio entre os métodos.

Nosso método ainda se mostrou robusto com relação a diferentes soluções iniciais. Com efeito, a semelhança dos *gaps* entre as variantes ALNS-B e ALNS-R confirmam a habilidade de nossos algoritmos na produção de bons resultados independentemente do ponto de partida.

Sejam $t(s)$ e $t^*(s)$ o tempo total de execução e o tempo necessário para a determinação da melhor solução, respectivamente. A Tabela 5.6 mostra que a heurística que produz a solução inicial Ch-B tem o pior desempenho dentre todos os algoritmos considerados em nossos experimentos. Tais resultados salientam a necessidade de uma fase de pós-otimização e confirmam o bom desempenho dos operadores de remoção e inserção.

Embora esteja claro que a ALNS-B e a ALNS-R são comparáveis quanto à qualidade das soluções (o que é confirmado pelo teste de Wilcoxon² com nível de significância $p < 0,01$), é interessante notar que a ALNS-B gasta menos de 31% do tempo de execução exigido pela ALNS-R.

²O teste de Wilcoxon é um teste de hipótese estatística, não-paramétrico, utilizado quando se pretende comparar duas amostras relacionadas, pareadas ou medições repetidas em uma única amostra para avaliar se suas distribuições populacionais de dados são idênticas ou não.

Figura 5.5: Comportamento dos métodos ALNS-B e ALNS-R para um problema com 4 máquinas e 25 tarefas (instância 1262 do conjunto IPMTC-I).

Figura 5.6: Comportamento dos métodos ALNS-B e ALNS-R para um problema com 4 máquinas e 25 tarefas (instância 1301 do conjunto IPMTC-I).

Finalmente, o número de melhorias aplicadas para alcançar a melhor solução utilizando a ALNS-R é um pouco maior, como indicado na coluna *Imp*, o que se justifica pela baixa qualidade de seus pontos de partida. As Figuras 5.5 e 5.6 confirmam o comportamento da ALNS. Elas exibem como as soluções novas (*New*), atuais (*Current*) e melhores (*Best*) mudam ao longo das 25.000 iterações. As estatísticas à esquerda correspondem à variante ALNS-B, enquanto os resultados relativos à ALNS-R encontram-se resumidos no lado direito.

As Tabelas 5.7 e 5.8 tratam do desempenho dos operadores de remoção e inserção da ALNS, respectivamente. O segundo e o terceiro conjuntos de colunas fornecem a percentagem média de chamadas nas quais uma nova melhor solução foi encontrada através da utilização de cada um dos operadores. Entre os métodos de remoção, o operador REMOÇÃO DE MÁQUINA CRÍTICA (R7) foi responsável pela maior frequência de melhorias, imediatamente seguido por REMOÇÃO DE MAIOR DESVIO (R4). Ambos são baseados em movimentos simples “gulosos” e de intensificação, mas são igualmente importantes para a diversificação do processo de busca. Dentre os operadores de inserção, INSERÇÃO 2-REGRET (I5) é o mais bem sucedido. É interessante notarmos a superioridade do operador INSERÇÃO GULOSA Δ (I2) em comparação com INSERÇÃO GULOSA p (I1). Embora ambos se baseiem em estratégias “gulosas”, o segundo efetua uma análise mais aprofundada considerando não apenas tempos de processamento de tarefas, mas também a contribuição do número de trocas de ferramentas à função objetivo.

Finalmente, a quarta coluna apresenta a degradação da média das soluções encontradas com e sem a aplicação de cada operador. Através do mesmo subconjunto de 30 instâncias do conjunto IPMTC-I utilizadas durante a fase de calibração, a ALNS-B foi executada excluindo um operador de cada vez, enquanto os demais eram mantidos fixos. A degradação das soluções é quase constante,

Tabela 5.7: Estatísticas dos operadores de remoção.

Operador	Média % de melhorias (Por cenário)						Média % de melhorias	Degradação de soluções
	c1	c2	c3	c4	c5	c6		
R1	7,78	4,43	6,95	2,69	4,63	2,29	4,80	0,10
R2	8,60	4,37	6,78	2,64	4,73	1,55	4,78	0,10
R3	8,90	4,30	6,95	2,65	6,14	1,84	5,13	0,10
R4	14,45	8,31	10,06	4,64	7,33	2,32	7,85	0,11
R5	12,60	7,30	7,57	4,12	5,02	2,14	6,46	0,10
R6	12,50	7,12	8,22	4,37	4,78	2,60	6,60	0,11
R7	13,85	6,74	11,32	3,92	8,74	2,62	7,87	0,11
R8	9,59	4,26	8,23	2,71	5,85	1,55	5,37	0,10
R9	9,01	4,41	6,98	2,69	4,96	1,54	4,93	0,10

Nota: Números em negrito indicam a porcentagem média máxima de chamadas e a degradação máxima entre os diferentes operadores.

Tabela 5.8: Estatísticas dos operadores de inserção.

Operador	Média % de melhorias (Por cenário)						Média % de melhorias	Degradação de soluções
	c1	c2	c3	c4	c5	c6		
I1	6,32	3,36	6,45	1,74	4,86	0,95	3,95	0,11
I2	15,16	5,05	12,67	3,16	10,68	1,87	8,10	0,10
I3	6,53	2,59	5,66	1,51	3,97	0,78	3,51	0,10
I4	14,85	4,46	12,72	2,88	10,44	1,89	7,87	0,11
I5	16,72	5,12	13,63	3,24	11,13	2,65	8,75	0,11

Nota: Números em negrito indicam a porcentagem média máxima de chamadas e a degradação máxima entre os diferentes operadores.

variando entre 10% e 11%.

5.3.3.3 Experimento 3: ALNS no conjunto de instâncias IPMTC-II

Enfatizamos, nesta seção, os resultados relativos às instâncias grandes pertencentes ao conjunto IPMTC-II. Testamos os limites do nosso método através de experimentos e comparações entre os algoritmos ALNS-B, ALNS-R, CLIP1 e CLIP2. Com respeito ao algoritmo ALNS-B e, como esperado, testes iniciais evidenciaram dificuldades já na solução das instâncias com 50 tarefas. A fim de melhor explicar este comportamento, realizamos, então, testes em 15 instâncias representativas do conjunto IPMTC-II, variando o número de tarefas entre 50, 100 e 200 e fixando o número de iterações e remoções iguais a 25.000 e 25% (Cenário 4), respectivamente. A ALNS-B foi executada duas vezes para cada problema, totalizando 30 experimentos. Para instâncias com 100 e 200 tarefas, o método atingiu o tempo limite permitido de 3600 segundos antes mesmo de finalizar a determinação da solução inicial. No caso de instâncias com 50 tarefas, a ALNS-B foi completamente executada, mas o tempo computacional gasto durante a fase da solução inicial foi, em média, 83,76% do tempo total de execução, como vemos na coluna *Run(%)* da Tabela 5.9. A quarta e a quinta colunas, $t(s)$ e $t_i(s)$, fornecem o tempo total de execução e o tempo necessário para a determinação da solução inicial, respectivamente.

Para analisar o impacto do tamanho das instâncias do conjunto IPMTC-II na meta-heurística ALNS proposta, a Tabela 5.10 apresenta aos *gaps* relativos às melhores soluções obtidas pelos métodos ALNS-R, CLIP1 e CLIP2 para cada instância. Consideramos o Cenário 4, por este ter sido o melhor cenário identificado durante os experimentos no conjunto de instâncias IPMTC-I. Os cabeçalhos das colunas são os mesmos explicados na Seção 5.3.3.2, com exceção da coluna adicional *Exc(%)*, que relata a parcela de instâncias de cada combinação máquina-tarefa-ferramenta que atingem o tempo limite de 4 horas (14.400 segundos) antes do término do número total de iterações. Enfatizamos, entretanto, que mesmo nosso algoritmo sendo parado por tempo limite em 86,52% dos casos, ainda assim alcançamos melhores resultados que os métodos CLIP1 e CLIP2 de Fathi e Barnette (2002) em todas as classes de instâncias. Com efeito, a ALNS-R gera *gaps* médios iguais a 1,7%, 2,2%, e 2,32% para 50, 100 e 200 tarefas, respectivamente, em contraste

Tabela 5.9: *Desempenho da ALNS-B - Cenário 4.*

$ M $	$ J $	$ T $	$t(s)$	$t_i(s)$	Run(%)
3	50	30	1000	861	86,10
3	50	30	957	820	85,68
3	50	30	1159	975	84,12
3	50	30	1135	943	83,08
4	50	30	1294	1092	84,39
4	50	30	1276	1067	83,62
5	50	40	924	719	77,81
5	50	40	907	701	77,29
5	50	40	3202	2813	87,85
5	50	40	3161	2772	87,69
Média			1501,50	1276,30	83,76

Tabela 5.10: *Gap relativo à melhor solução encontrada considerando os métodos CLIP1, CLIP2 e ALNS-R - Cenário 4.*

$ M $	$ J $	$ T $	CLIP1		CLIP2		ALNS-R		Exc(%)
			Gap(%)	Gap*(%)	Gap(%)	Gap*(%)	Gap(%)	Gap*(%)	
3	50	30	23,79	15,16	22,56	14,73	1,62	0,00	9,33
3	50	40	18,89	11,86	18,41	10,94	1,29	0,00	74,17
4	50	30	20,84	11,75	18,57	11,23	1,67	0,00	31,88
4	50	40	20,27	11,54	20,46	11,20	1,89	0,00	82,33
4	100	30	18,31	7,69	15,28	6,37	1,96	0,00	100,00
4	100	40	13,77	4,45	10,93	3,49	1,97	0,12	100,00
5	50	30	18,66	9,99	15,90	7,93	1,62	0,00	15,83
5	50	40	21,69	12,28	21,69	12,71	2,08	0,00	63,00
5	100	30	15,12	5,14	11,88	4,24	1,87	0,02	100,00
5	100	40	12,40	3,38	9,78	2,33	2,37	0,48	100,00
6	100	30	16,51	5,95	12,14	2,98	2,41	0,46	100,00
6	100	40	13,04	2,89	9,63	1,41	3,12	0,89	100,00
6	200	30	22,25	12,03	17,06	9,51	1,84	0,00	100,00
6	200	40	18,79	9,37	14,99	7,81	2,54	0,03	100,00
7	100	30	14,63	5,13	10,82	3,35	2,06	0,37	100,00
7	100	40	15,08	5,09	11,04	3,22	1,85	0,06	100,00
7	200	30	21,38	12,87	15,90	8,60	2,29	0,04	100,00
7	200	40	18,82	9,39	13,88	7,23	2,64	0,05	100,00
8	200	30	21,52	10,98	16,29	9,40	2,17	0,00	100,00
8	200	40	17,57	7,65	12,38	5,04	2,38	0,08	100,00
9	200	30	18,81	9,51	13,96	7,35	2,17	0,01	100,00
9	200	40	16,77	7,54	12,52	6,24	2,73	0,07	100,00
10	200	30	19,72	10,43	14,27	7,84	2,08	0,05	100,00
10	200	40	17,24	8,20	11,51	4,87	2,38	0,05	100,00
Média			18,16	8,76	14,66	7,08	2,13	0,12	86,52

com os valores 19,6%, 11,44%, e 14,27% alcançados pelo CLIP2.

5.4 Conclusões

Neste capítulo, modelamos e resolvemos o problema de sequenciamento de tarefas em um conjunto de máquinas paralelas idênticas, com foco na incorporação de restrições relacionadas a trocas de ferramentas em sistemas flexíveis de manufatura (IPMTC). Como principais contribuições, foram propostas duas formulações matemáticas e uma meta-heurística do tipo ALNS. Nosso algoritmo foi testado com dois métodos distintos para a obtenção de soluções iniciais e em seis cenários diferentes, a fim de atestarmos a eficiência dos operadores de remoção e inserção que compõem a ALNS. Para avaliarmos, plenamente, a eficácia e a precisão de nossa meta-heurística, geramos dois novos conjuntos de instâncias contendo até 10 máquinas, 200 tarefas e 40 ferramentas. Os resultados computacionais mostram que nosso método supera os algoritmos existentes em todas as 2880 instâncias que compõem os dois *benchmarks* propostos.

Capítulo 6

JSSPTC

Neste capítulo, estudamos a otimização do manejo de ferramentas utilizadas durante o processamento de tarefas em um conjunto de máquinas paralelas e idênticas inseridas em um sistema de manufatura do tipo *job shop* (JSSPTC).

Na Seção 6.1, o JSSPTC é definido matematicamente através de uma formulação inteira que visa a minimização do *makespan* ao final da produção. Em seguida, na Seção 6.2, apresentamos os detalhes de uma heurística construtiva, dotada de um método de determinação de soluções iniciais baseado na utilização de diferentes regras de despacho, com critérios de escolha determinístico ou estocástico. As soluções advindas deste passo são submetidas a um procedimento de melhoria na Seção 6.3. A descrição do conjunto de instâncias utilizado nos experimentos computacionais, bem como os resultados deles obtidos, são apresentados na Seção 6.4. Por fim, na Seção 6.5, concluímos o capítulo com uma breve avaliação dos avanços realizados.

6.1 Formulação matemática

A seguir, introduzimos o que, de nosso conhecimento, é o primeiro modelo matemático desenvolvido para o JSSPTC. Neste trabalho, consideramos que o tempo incorrido para as trocas é proporcional ao número de trocas de ferramentas realizadas. As considerações iniciais de distribuição das operações de cada tarefa às máquinas são garantidas por restrições clássicas associadas ao ambiente *job shop*. Então, as restrições relacionadas ao manejo de ferramentas nos magazines das máquinas definem a ordem de processamento das tarefas até que todas as operações sejam executadas.

Para facilitar a compreensão da modelagem matemática, relembramos que cada tarefa $j \in J$ é composta por um conjunto de operações $\sigma^j = (\sigma_1^j \sigma_2^j \dots \sigma_p^j)$, previamente definido, em que $\sigma_u^j = m$, $u \in \{1, \dots, p\}$, indica que a u -ésima operação da tarefa j deve ser executada na máquina m .

6.1.1 Minimização do número de trocas de ferramentas

Denominamos a formulação matemática do JSSPTC como M_4 e, para seu desenvolvimento, utilizamos quatro famílias de variáveis, sendo três binárias. Seja x_{jm} a variável que representa o tempo de início do processamento da tarefa $j \in J$ na máquina m . Definimos também y_{ijm} igual a 1 se, e somente se, a tarefa i precede imediatamente o processamento da tarefa j na máquina m , ω_{tjm} igual a 1 se, e somente se, a ferramenta t é inserida para o processamento da tarefa j na máquina m e α_{tjm} igual a 1 se, e somente se, a ferramenta t está alocada na máquina m durante o processamento da tarefa j . Sendo K uma constante suficientemente grande e Δ o *makespan*, segue o modelo M_4 para o JSSPTC:

$$M4: \text{Min } \Delta \tag{6.1}$$

sujeito a

$$\Delta \geq x_{j\sigma_p^j} + p_{j\sigma_p^j} \quad j \in J \quad (6.2)$$

$$x_{j\sigma_{k+1}^j} \geq x_{j\sigma_k^j} + p_{j\sigma_k^j} + \bar{t} \sum_{t=1}^l \omega_{tj\sigma_{k+1}^j} \quad j \in J, k \in M \setminus \{p\} \quad (6.3)$$

$$x_{jm} \geq x_{im} + p_{is} - K(1 - y_{ijm}) + \bar{t} \sum_{t=1}^l \omega_{tjm} \quad i, j \in J, m \in M \quad (6.4)$$

$$\sum_{t=1}^l \alpha_{tjm} \leq C \quad j \in J, m \in M \quad (6.5)$$

$$y_{ijm} + \alpha_{tjm} - \alpha_{tim} \leq \omega_{tjm} + 1 \quad i, j \in J, t \in T, m \in M \quad (6.6)$$

$$\alpha_{tjm} = 1 \quad j \in J, t \in T_{jm}, m \in M \quad (6.7)$$

$$\omega_{tjm} \leq \sum_{i=1}^n y_{ijm} \quad j \in J, m \in M \quad (6.8)$$

$$\sum_{i \in J \setminus \{j\}} y_{ijm} \leq 1 \quad j \in J, m \in M \quad (6.9)$$

$$\sum_{j \in J \setminus \{i\}} y_{ijm} \leq 1 \quad i \in J, m \in M \quad (6.10)$$

$$y_{ijm} + y_{jim} \leq 1 \quad i, j \in J, i \neq j, m \in M \quad (6.11)$$

$$\sum_{j \in J \setminus \{i\}} y_{ijm} = n - 1 \quad i \in J, m \in M \quad (6.12)$$

$$x_{jm} \in \mathbb{Z}_+ \quad j \in J, m \in M \quad (6.13)$$

$$y_{ijm} \in \{0, 1\} \quad i, j \in J, m \in M \quad (6.14)$$

$$\omega_{tjm} \in \{0, 1\} \quad j \in J, t \in T, m \in M \quad (6.15)$$

$$\alpha_{tjm} \in \{0, 1\} \quad j \in J, t \in T, m \in M. \quad (6.16)$$

A função objetivo (6.1) minimiza o *makespan*, isto é, a soma dos tempos de processamento de operações e do tempo gasto com trocas de ferramentas na última máquina a ser finalizada (restrições (6.2)). As restrições (6.3) garantem que duas operações de uma mesma tarefa não sejam processadas simultaneamente em máquinas distintas, enquanto as restrições (6.4) asseguram que duas operações de diferentes tarefas não sejam executadas, ao mesmo tempo, em uma dada máquina. Adotamos $K = \sum_{j \in J} \sum_{m \in M} p_{jm} + \bar{t}l$. A limitação dos magazines das máquinas é respeitada pela restrição (6.5) e a situação de trocas de ferramentas é abordada nas restrições (6.6). Com efeito, se uma ferramenta não está na máquina para o processamento da i -ésima operação e nem foi inserida para o processamento da operação j , então esta não pode seguir aquela e requerer a ferramenta t ao mesmo tempo. As restrições (6.7) garantem a alocação de ferramentas nos magazines sempre que requeridas por uma operação e as restrições (6.8) impõem que se não há uma tarefa que preceda a tarefa j , então nenhuma troca é contabilizada neste instante. As restrições (6.9) e (6.10) estabelecem que cada tarefa possua, respectivamente e no máximo, um único predecessor e um único sucessor. Com o intuito de fortalecer o valor da relaxação linear deste modelo, propomos as restrições (6.11) e (6.12): primeiro eliminamos soluções simétricas quanto à precedência de tarefas em uma máquina e, na segunda, determinamos que o número máximo de pares de tarefas precedentes seja igual ao número de tarefas, n , menos uma unidade. Por último, as restrições (6.13) a (6.16) definem os domínios das variáveis.

6.2 Heurística construtiva

Nesta seção, descrevemos os passos da heurística construtiva (HC) proposta para a resolução do JSSPTC e apresentada no Algoritmo 10. Basicamente, para cada regra de despacho, aplicamos a heurística (CRS) para construção de soluções para o *job shop* desenvolvida por [Scrich \(1997\)](#) e obtemos uma solução s e seu *makespan* correspondente Δ (Passo 3). Com o intuito de acrescentar estocasticidade ao nosso método, repetimos este último passo por outras cinco vezes, para a mesma regra de despacho, porém utilizando um elemento randômico, denotado por ν , durante o processo de escolha de operações consideradas livres¹ para serem processadas (Algoritmo 12). Em cada uma das seis soluções obtidas, digamos $s_1, \dots, s_6$, aplicamos, então, a heurística KTNS de [Tang e Denardo \(1988a\)](#) para otimizar as trocas de ferramentas nas diferentes máquinas (Passo 4). Isso é necessário, uma vez que o método (CRS) fornece apenas uma estimativa deste número de trocas. Como consequência da utilização da KTNS, obtemos novas soluções $s'_1, \dots, s'_6$, com intervalos de troca distintos e, portanto, novos valores de *makespan* $\Delta'_1, \dots, \Delta'_6$, $\Delta'_i \leq \Delta_i$, $i \in \{1, \dots, 6\}$, os quais são determinados por meio do algoritmo de caminho crítico (Passo 5). No Passo 6, atualizamos o conjunto de soluções S através da inserção de $(s'_1, \Delta'_1), \dots, (s'_6, \Delta'_6)$. Notemos que, ao final, teremos 48 soluções, obtidas pela variação das oito regras de despacho.

Algoritmo 10 Heurística construtiva CH.

entradas Rd (Conjunto de regras de despacho);

- 1: Seja S um conjunto de pares representando soluções e seus respectivos *makespans*; $S \leftarrow \emptyset$;
 - 2: **para todo** $r \in Rd$ **faça**
 - 3: $(s, \Delta) \leftarrow CRS(r)$;
 - 4: $s' \leftarrow KTNS(s)$;
 - 5: $\Delta' \leftarrow CritPath(s')$;
 - 6: $S \leftarrow S \cup \{(s', \Delta')\}$;
 - 7: **fim**
 - 8: **retorne** S (conjunto de soluções e *makespan* encontrados).
-

Na sequência, explicamos os detalhes de cada um dos procedimentos utilizados no Algoritmo 10. Na Seção 6.2.1, começamos pela solução inicial gerada pela (CRS), passamos pelas fórmulas das regras de despacho propostas (Seção 6.2.1.1) e, por último, pelos passos do algoritmo de determinação do caminho crítico (Seção 6.2.1.2).

6.2.1 Solução inicial

O Algoritmo 11 descreve a heurística (CRS) de [Scrich \(1997\)](#). No Passo 10, verificamos a disponibilidade de uma dada máquina m no instante t_{aloc} em que se considera a alocação de uma operação no sistema de produção. Em caso afirmativo, aplicamos a função *OperAvailable* para determinar o conjunto X das operações passíveis de serem processadas por m (Passo 11). Através da função *BestRD*, avaliamos as operações de X conforme a regra de despacho r passada como parâmetro e selecionamos a operação op com maior prioridade (Passo 15). A operação op é, então, alocada à máquina m , a qual tem seu instante de disponibilidade atualizado pelo acréscimo do tempo gasto com o processamento da operação op , somado ao tempo utilizado durante as trocas de ferramentas geradas (Passos 15–17). Para esta estimativa do número de trocas, a função *Switches* utiliza o *lower bound* $LB = \max\{0, |T_{op,m} \cup T_{op',m}| - C\}$, sendo op' a operação mais recentemente executada na máquina m , e prioriza que permaneçam no magazine da máquina as ferramentas consideradas mais populares.

Ainda dentro da função *BestRD*, faremos também as escolhas randômicas mencionadas no início da Seção 6.2. Como pode ser visto no Algoritmo 12, basicamente, em cada uma das cinco repetições, reorganizamos o conjunto X em ordem não-crescente ou não-decrescente (dependendo da regra de despacho em questão, Seção 6.2.1.1) de seus valores de prioridade, donde obtemos o

¹Dizemos que uma operação está *livre* quando o início de seu processamento não viola as restrições clássicas do ambiente *job shop*.

Algoritmo 11 Heurística CRS de [Scrich \(1997\)](#).

entradas p_{jm} (Tempo de processamento da tarefa j na máquina m , $j \in J$, $m \in M$);
 K' (Constante suficientemente grande).
 r (Regra de despacho);

- 1: Sejam O e X conjuntos de operações; $O \leftarrow \emptyset$; $X \leftarrow \emptyset$;
- 2: Seja m a máquina atual;
- 3: Seja op uma operação;
- 4: Seja $time[m]$ o instante em que uma máquina m se torna disponível; $time[m] \leftarrow 0$, $m \in M$;
- 5: Seja t_{aloc} o instante em que se considera a alocação de uma operação à uma dada máquina; $t_{aloc} \leftarrow 0$;
- 6: Seja s uma solução do JSSPTC;
- 7: Seja Δ o *makespan*; $\Delta \leftarrow 0$;
- 8: $m \leftarrow 1$;
- 9: **enquanto** $|O| < pn$ **faça**
- 10: **se** $time[m] \leq t_{aloc}$ ou m está marcada **então**
- 11: $X \leftarrow OperAvailable(m, t_{aloc})$;
- 12: **se** $X = \emptyset$ **então**
- 13: GOTO MARCA $_m$;
- 14: **fim**
- 15: $op \leftarrow BestRD(X, r)$;
- 16: Na solução s , aloque a operação op à máquina m ;
- 17: Atualize $time[m] = t_{aloc} + p_{op,m} + \bar{t}Switches(m, op)$;
- 18: **senão**
- 19: MARCA $_m$:
- 20: Marque a máquina m ;
- 21: $time[m] = K'$;
- 22: **fim**
- 23: $m \leftarrow m + 1(\text{mod } p)$;
- 24: $t_{aloc} \leftarrow GetMin(time, p)$;
- 25: $O \leftarrow O \cup \{op\}$;
- 26: **fim**
- 27: $\Delta \leftarrow GetMax(time, p)$;
- 28: **retorne** s (solução do JSSPTC) e Δ (*makespan*).

conjunto ordenado $\bar{X}$ (Passo 5). Então, alocamos em Y apenas os índices associados às operações de $\bar{X}$ cujas prioridades não excedem o valor de *corte* (Passos 7–11). Nos Passos 12 e 13, escolhemos um índice aleatório ν do conjunto Y e a ν -ésima operação de $\bar{X}$ para ser processada.

Algoritmo 12 Estocasticidade no processo de escolha de operações.

entradas r (Regra de despacho);
 X (Conjunto de operações livres para serem processadas);

- 1: Seja Y um conjunto de índices; $Y \leftarrow \emptyset$;
- 2: Seja $\bar{X}$ um conjunto de operações; $\bar{X} \leftarrow \emptyset$;
- 3: Seja *corte* um número positivo;
- 4: Seja *op* uma operação;
- 5: $\bar{X} \leftarrow \text{Sort}(X, r)$; digamos $\bar{X} = \{\bar{op}_1, \dots, \bar{op}_{|X|}\}$
- 6: $\text{corte} \leftarrow \frac{|z_{\bar{op}_{|X|}} - z_{\bar{op}_1}|}{2}$;
- 7: **para todo** $i = 1, \dots, |X|$ **faça**
- 8: **se** $\bar{op}_i \leq \text{corte}$ **então**
- 9: $Y \leftarrow Y \cup \{i\}$;
- 10: **fim**
- 11: **fim**
- 12: Escolha um elemento randômico, ν , em Y ;
- 13: $op \leftarrow \bar{op}_\nu$;
- 14: **retorne** *op* (operação escolhida).

Dando continuidade ao Algoritmo 11, consideramos que uma máquina m é marcada nos casos em que não está disponível (Passos 18–22) ou em que o conjunto X é vazio, isto é, não há operações para serem nela processadas (Passos 12–14). Para garantir que esta máquina fique ociosa durante um ciclo do algoritmo, utilizamos a constante K' no Passo 21 para evitar que m seja escolhida pela função *GetMin*, a qual determina o valor mínimo do conjunto $\{time[m]; m \in M\}$ (Passo 24). Nos Passos 23 a 25, passamos à próxima máquina, atualizamos o instante de alocação e adicionamos a operação *op* ao conjunto O das operações já processadas. O algoritmo se repete até que todas as (pn) operações sejam executadas no conjunto de p máquinas. Como dado de saída, retorna uma solução s para o JSSPTC e o respectivo valor de seu *makespan* Δ , contabilizado no Passo 27 através da função *GetMax*.

6.2.1.1 Regras de despacho

Quanto ao mecanismo das regras de despacho, este calcula, sempre que uma máquina se torna disponível, um valor de prioridade para cada uma das operações livres para serem processadas nesta máquina. A operação com melhor prioridade é, então, escolhida. O cálculo de cada valor de prioridade varia conforme a semântica da regra em questão. Neste trabalho, implementamos oito regras de despacho que consideram tanto elementos do *job shop* quanto da minimização de trocas de ferramentas.

Dado o conjunto X das operações livres e m a máquina disponível, definamos z_j como o valor de prioridade da operação de X associada à tarefa $j \in J$. As regras 1, 3, 5 e 6 foram selecionadas da literatura do *job shop* (Holthaus e Rajendran, 1997; Jayamohan e Rajendran, 2000; Scrich, 1997). As demais foram adaptadas das anteriores (regras 2 e 4) ou criadas especificamente para o JSSPTC (regras 7 e 8), todas considerando trocas de ferramentas no cálculo das respectivas prioridades. Para a estimativa do número de trocas, utilizamos o mesmo critério da função *BestRD*, da seção anterior.

1. *Shortest processing time* (SPT)

O valor de prioridade z_j é dado pelo tempo de processamento da operação relacionada a uma dada tarefa j na máquina m :

$$z_j = p_{jm}.$$

A operação com *menor* valor de z_j é escolhida.

2. ***Shortest processing time com trocas de ferramentas (SPT_Trocas)***

O valor de prioridade z_j é dado pela soma do tempo de processamento da operação relacionada a uma dada tarefa j na máquina m , com o tempo de troca $\bar{t}$ multiplicado por uma estimativa do acréscimo gerado no número de trocas de ferramentas, digamos τ :

$$z_j = p_{jm} + \bar{t}\tau.$$

A operação com *menor* valor de z_j é escolhida.

3. ***Longest processing time (LPT)***

O valor de prioridade z_j é dado pelo tempo de processamento da operação relacionada a uma dada tarefa j na máquina m :

$$z_j = p_{jm}.$$

A operação com *maior* valor de z_j é escolhida.

4. ***Longest processing time com trocas de ferramentas (LPT_Trocas)***

O valor de prioridade z_j é dado pela soma do tempo de processamento da operação relacionada a uma dada tarefa j na máquina m , com o tempo de troca $\bar{t}$ multiplicado por uma estimativa do acréscimo τ gerado no número de trocas de ferramentas:

$$z_j = p_{jm} + \bar{t}\tau.$$

A operação com *maior* valor de z_j é escolhida.

5. ***Processing time and present waiting time (PTPW)***

O valor de prioridade z_j é dado pela subtração entre o tempo de processamento da operação u relacionada a uma dada tarefa i na máquina m , e o instante t_{u-1} em que foi finalizada a execução da sua operação precedente ($u-1$), conforme consta no plano de processamento da tarefa j :

$$z_j = p_{im} - t_{u-1}.$$

A operação com *menor* valor de z_j é escolhida.

6. ***Flow due-date (FDD)***

O valor de prioridade z_j de uma operação u relacionada a uma dada tarefa j na máquina m é dado pela soma do instante t^j em que foi iniciado o processamento da primeira operação de j com o somatório dos tempos de processamento requeridos pelas operações precedentes à operação u em suas respectivas máquinas, digamos m_μ :

$$FDD_{j,u} = t^j + \sum_{q=1}^u p_{m_\mu,q},$$

$$z_j = FDD_{j,u}.$$

A operação com *menor* valor de z_j é escolhida.

7. *Mínimo de trocas* (MIN_Trocas)

O valor de prioridade z_j é dado pela multiplicação entre o tempo de troca $\bar{t}$ e uma estimativa do acréscimo τ gerado no número de trocas de ferramentas ao processarmos uma operação relacionada a uma dada tarefa j na máquina m :

$$z_j = \bar{t}\tau.$$

A operação com *menor* valor de z_j é escolhida.

8. *Contribuição à função objetivo* (CONT_Trocas)

O valor de prioridade z_j é dado pela contribuição da operação u relacionada a uma dada tarefa j à função objetivo, ou seja, a diferença no valor do *makespan* parcial Δ^{parc} obtido ao considerarmos ou não a execução de u :

$$z_j = \Delta^{parc}(j, u) - \Delta_{-u}^{parc}(j, u),$$

onde Δ_{-u}^{parc} representa o cálculo do *makespan* na ausência da operação u . A operação com *menor* valor de z_j é escolhida.

6.2.1.2 Algoritmo do caminho crítico

Nesta seção, descrevemos o algoritmo de determinação do caminho crítico (ou mais longo) de um dado grafo. Antes disso, entretanto, introduzimos uma representação gráfica do *job shop* (JSSP) proposta por Taillard (1994) e, posteriormente, adaptada ao JSSPTC por Hertz e Widmer (1996).

Para o grafo do JSSP, criamos um nó para cada operação de cada tarefa, cujo peso é dado pelo seu tempo de processamento. Existe um arco (u_1, u_2) conectando as operações u_1 e u_2 se elas pertencem à mesma tarefa e u_1 deve ser processada exatamente antes de u_2 . Adicionamos dois nós fictícios 0 e $(pn + 1)$, representando o início e o fim do processamento das operações. O nó fictício 0 é conectado por arcos de comprimentos nulos à operação inicial de cada tarefa, da mesma forma que existem nós de comprimentos também nulos ligando as operações finais de cada tarefa ao nó fictício $(pn + 1)$. Os arcos tratam das restrições de precedência entre as operações de uma mesma tarefa, especificadas pelo plano de processamento do *job shop*. Nós relacionados a operações executadas em uma mesma máquina são unidos por arcos disjuntivos, isto é, com direção ainda não conhecida. Notemos que, estabelecidas tais direções, obtemos um grafo direcionado, a ordem de processamento em cada máquina e, assim, o *makespan*.

Figura 6.1: Grafo do JSSP para instância com 3 máquinas e 3 tarefas.

Figura 6.2: Solução factível, conforme o caminho crítico do grafo JSSP.

O caminho crítico de cada tarefa é o caminho mais longo entre 0 e seu último nó e o caminho crítico do grafo é o caminho mais longo entre 0 e $(pn + 1)$. Se o grafo direcionado obtido é acíclico, sua configuração caracteriza uma solução factível para o JSSP. Buscamos, portanto, pelo caminho crítico de menor comprimento.

A Figura 6.1 ilustra o grafo disjuntivo relativo à versão *job shop* da instância com 3 máquinas e 3 tarefas descrita na Tabela 2.6. O caminho crítico do grafo é dado por $0 \rightarrow 4 \rightarrow 2 \rightarrow 8 \rightarrow 9 \rightarrow 10$ e gera a solução factível da Figura 6.2, com *makespan* igual a 6 segundos.

A adaptação deste grafo para o contexto do JSSPTC é simples. Um instante de troca entre duas operações u_1 e u_2 é considerado um nó artificial, inserido no grafo entre os nós relativos às operações em questão e com peso igual ao tempo gasto com as trocas de ferramentas entre u_1 e u_2 , ou seja, $\bar{t}\tau_{(u_1, u_2)}$. As demais estruturas se mantêm como no grafo do JSSP. Por exemplo, na Figura 6.3, ao supormos a necessidade de 3 trocas entre as operações 4 e 2, e 2 trocas entre as operações 3 e 6, obteríamos o grafo do JSSPTC com 12 vértices apresentado abaixo, cujos nós artificiais são realçados na cor verde.

Figura 6.3: Grafo do JSSPTC para instância com 3 máquinas e 3 tarefas.

Notemos que, de fato, a inserção de restrições de trocas de ferramentas afetam a ordem de processamento anteriormente estabelecida para o *job shop*. Isto justifica nossa estratégia de sempre aplicarmos o método do caminho crítico (Algoritmo 13), de ordem polinomial (Taillard, 1994), após a otimização dos magazines das máquinas feita pela KTNS. Para uma melhor compreensão dos passos de tal algoritmo, tomemos $G = (V, A)$ o grafo JSSPTC a ser otimizado e λ seu respectivo número de nós artificiais. Dada op uma de suas operações, sejam j_{op} a tarefa à qual a operação op pertence, m_{op} a máquina responsável por seu processamento, α_j^{op} (ω_j^{op}) a operação pertencente à tarefa j_{op} predecessora (sucessora) da operação op , α_M^{op} (ω_M^{op}) a operação que precede (sucede) op na máquina m_{op} , t^{op} o menor instante em que o processamento da operação op é iniciado, l^{op} o comprimento do caminho crítico entre a operação op e o nó fictício $(pn + \lambda + 1)$ e, por fim, $p_{m_{op}, op}$ o tempo requerido para a execução de op em sua respectiva máquina. As regras de precedência entre as operações influenciam tanto o sequenciamento em cada uma das máquinas, quanto entre as diferentes máquinas do problema, motivo pelo qual os cálculos de t^{op} e l^{op} são dados por:

$$t^{op} = \max\{t_M^{\alpha_M^{op}} + p_{m_{op}, \alpha_M^{op}}, t_j^{\alpha_j^{op}} + p_{m_{op}, \alpha_j^{op}}\}; \quad (6.17)$$

$$l^{op} = \max\{l_M^{\omega_M^{op}}, l_j^{\omega_j^{op}}\} + p_{op, m_{op}}; \quad (6.18)$$

onde utilizamos m_μ para representar a máquina onde é processada a operação que precede op no plano de processamento da tarefa j .

Algoritmo 13 Determinação do caminho crítico.

entradas Grafo direcionado do JSSPTC, $G = (V, A)$.

- 1: Seja l_{crit} um número positivo; $l_{crit} \leftarrow 0$;
 - 2: $t^{op} \leftarrow 0$, $l^{op} \leftarrow 0$, $op \in V$;
 - 3: **para todo** $op \in V$ **faça**
 - 4: $Calculate_t(op)$;
 - 5: $Calculate_l(op)$;
 - 6: **fim**
 - 7: $l_{crit} \leftarrow \max_{op \in V}\{t^{op} + l^{op}\}$;
 - 8: **retorne** l_{crit} (comprimento do caminho crítico do grafo G).
-

O laço de repetição central do algoritmo calcula os valores de t^{op} e l^{op} para cada operação do grafo (Passos 3–6). Então, no Passo 7, tomamos como comprimento do caminho crítico de G a maior soma entre o início do processamento de uma operação e o tempo gasto para que esta alcance o nó fictício $(pn + \lambda + 1)$.

6.3 Heurística de pós-otimização

Nesta seção, apresentamos a busca local aplicada à solução obtida pela heurística construtiva CH e as soluções origem e destino de um posterior *path relinking*. Em suma, a ideia consiste, primeiramente, em explorar as vizinhanças da ordem de processamento de operações alocadas à máquina crítica, mantendo fixas a configuração das demais máquinas. Então, através do *path relinking*, propomos movimentos exploratórios capazes de aprimorar a busca por melhores soluções.

6.3.1 Busca local (BL)

Dada a solução s proveniente da heurística CH, Δ seu *makespan* e a máquina crítica m^* , consideremos $(op_1^*, \dots, op_n^*)$ a sequência de operações executadas em m^* . Propomos uma busca local que troca a ordem de execução de todos os pares de operações (op_i^*, op_j^*) , sendo $i, j \in J$, $i \neq j$, lembrando que são desconsideradas as trocas que causem ciclos no grafo. Após cada permuta, aplicamos a heurística KTNS para a reorganização ótima dos magazines e, então, o algoritmo de caminho crítico para determinar o novo *makespan*. Ao final do processo de trocas, buscamos

pela solução s' responsável pelo menor entre os novos *makespans* encontrados, denotado por Δ' . Se $\Delta' < \Delta$, iniciamos uma nova busca local a partir de s' . Caso contrário, mantemos a solução s anterior. O algoritmo para quando esgota todas as possibilidades de trocas com potencial de melhoria das soluções, ou seja, após verificar toda a vizinhança de s .

6.3.2 Path relinking

O Algoritmo 14 descreve uma melhoria desenvolvida para a heurística construtiva CH. Através da função *Best*, nos Passos 7 e 8 determinamos as melhores soluções obtidas pela heurística CH e pela busca local BL. Então, aplicamos o método *Path relinking* (PR) utilizando-as como origem e destino, respectivamente.

Algoritmo 14 Heurística construtiva final: após a busca local e o algoritmo de *path relinking*.

entradas $S \leftarrow \text{HC}(\text{regra})$;

1: Seja S' um conjunto de soluções; $S' \leftarrow \emptyset$;

2: Sejam (s_1, Δ_1) , (s_2, Δ_2) e (s^*, Δ^*) soluções do JSSPTC, acompanhadas de seus *makespans*;

3: **para todo** $(s, \Delta) \in S$ **faça**

4: $(s', \Delta') \leftarrow \text{BL}(s, \Delta)$;

5: $S' \leftarrow S' \cup \{(s', \Delta')\}$;

6: **fim**

7: $(s_1, \Delta_1) \leftarrow \text{Best}(S)$;

8: $(s_2, \Delta_2) \leftarrow \text{Best}(S')$;

9: $(s^*, \Delta^*) \leftarrow \text{PR}(s_1, s_2)$;

10: **retorne** (s^*, Δ^*) (melhor solução obtida pelo *path relinking*).

6.4 Experimentos computacionais

Nesta seção, apresentamos aspectos relativos aos experimentos computacionais realizados. Na Seção 6.4.1, introduzimos a descrição do conjunto de instâncias teste utilizado. Em seguida, resumizamos os resultados referentes à modelagem matemática e à heurística construtiva desenvolvidas para o JSSPTC.

6.4.1 Instâncias teste

Para a avaliação dos métodos desenvolvidos, geramos um novo conjunto de instâncias para o JSSPTC. Utilizamos 46 instâncias do *job shop* clássico, retiradas da base de dados da *OR-Library*² e contendo entre 36 e 300 operações. Para adaptar ao contexto de trocas de ferramentas, inserimos conjuntos de requerimentos das operações, capacidade dos magazines e tempos de troca distintos, conforme o esquema adotado por Yanasse *et al.* (2009).

A instância de menor porte gerada contém 6 tarefas, 6 máquinas e 15 ferramentas. Ao variarmos $C = 5$ ou $C = 10$, obtemos duas configurações máquina-tarefa-ferramenta-capacidade, que se tornam seis ao considerarmos três valores de tempo de troca. Tomando 10 casos para cada uma destas combinações, chegamos a um total de 60 exemplares.

Quanto às demais instâncias, cada configuração máquina-tarefa resulta em três combinações máquina-tarefa-ferramenta, dependendo do número de ferramentas requeridas $|T|$ ser igual a 15, 20 ou 25. A capacidade dos magazines assume valores do conjunto $\{5, 10\}$ quando $|T| = 15$, $\{5, 10, 15\}$ se $|T| = 20$ e $\{5, 10, 15, 20\}$ quando $|T| = 25$, donde obtemos um total de 9 combinações máquina-tarefa-ferramenta-capacidade. Ao considerarmos três fórmulas para determinação do tempo de troca, chegamos a 27 combinações. Ao final, obtemos $(60 + 45 \times 27) = 1275$ novas instâncias para o JSSPTC.

²<http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html>

6.4.2 Resultados

Utilizamos a linguagem C++ para implementação dos algoritmos e os experimentos foram realizados em um computador 2.67 GHz Intel Xeon X55 com 12 Gb de memória. Para a resolução do modelo M_4 , utilizamos o *software* IBM CPLEX versão 12.6.2, com uma *thread*, tamanho máximo da árvore de busca igual a 6 Gb e tempo limite de 3600 segundos para a solução de cada instância.

Em termos de notação, utilizamos as siglas CH para a versão determinística da heurística construtiva vista no Algoritmo 10, e CH-R para a versão estocástica da mesma. O mesmo ocorre para a busca local BL abordada na Seção 6.3.1. Por último, escrevemos Final e Final-R como referência à heurística CH após o procedimento de busca local e a aplicação do *path relinking*, também em suas versões determinística e estocástica. Como dito anteriormente, não há modelos nem heurísticas construtivas propostos na literatura do JSSPTC. Por esse motivo, com o intuito de bem avaliar os nossos métodos, comparamos a solução obtida por cada um com a melhor solução sabida quando consideramos todos eles.

6.4.2.1 Experimento 1: formulação matemática

A Tabela 6.1 sumariza os resultados alcançados ao resolvermos o modelo M_4 . As três primeiras colunas representam o número de tarefas ($|J|$), máquinas ($|M|$) e ferramentas ($|T|$) e o *gap* de otimalidade obtido pelo CPLEX pode ser visto na coluna *Gap*(%). A quinta coluna, $t(s)$, mostra o tempo computacional, em segundos, *Opt*(%) retrata a porcentagem de instâncias resolvidas de forma ótima. Os casos em que nem mesmo uma solução factível foi encontrada são abordados na coluna *Unsolved*(%).

Tabela 6.1: Análise do desempenho da formulação matemática.

$ J $	$ M $	$ T $	M_4					
			Gap(%)	t(s)	Opt(%)	Unsolved(%)		
6	6	15	14,2	2190	55,0	0,0		
10	5	15	54,5	3148	0,0	16,7		
		20	48,5	3282	0,0	26,7		
		25	42,9	3237	0,0	35,0		
		10	15	2,8	3600	0,0	95,2	
15	5	20	1,3	3600	0,0	96,8		
		25	-	3600	0,0	100,0		
		15	-	3600	0,0	100,0		
		20	-	3600	0,0	100,0		
		25	1,2	3600	0,0	98,3		
		10	15	-	3600	0,0	100,0	
	10	20	-	3600	0,0	100,0		
		25	-	3600	0,0	100,0		
		15	15	-	3600	0,0	100,0	
		20	-	3600	0,0	100,0		
		25	-	3600	0,0	100,0		
		20	5	15	-	3600	0,0	100,0
20	5	20	-	3600	0,0	100,0		
		25	-	3600	0,0	100,0		
		10	15	-	3600	0,0	100,0	
		20	-	3600	0,0	100,0		
		25	-	3600	0,0	100,0		
		15	15	-	3600	0,0	100,0	
	10	20	-	3600	0,0	100,0		
		25	-	3600	0,0	100,0		
		30	10	15	-	3600	0,0	100,0
		20		-	3600	0,0	100,0	
		25		-	3600	0,0	100,0	
		25		-	3600	0,0	100,0	
Média			23,6	3509	2,0	88,2		

Nota: Utilizamos o símbolo - quando mesmo uma solução factível não é encontrada. A média dos *gaps* de otimalidade não considera estes casos.

No primeiro grupo, referente às instâncias de menor porte, o modelo sempre encontra ao menos uma solução factível para o problema. Isso requer, entretanto, um tempo computacional elevado, em média, 2190 segundos. No segundo grupo de exemplares, não são encontradas soluções ótimas e a

quantidade de soluções factíveis diminui consideravelmente. A partir dos problemas com dimensões $|J| = 15$ e $|M| = 10$, o modelo se torna incapaz de determinar soluções factíveis em um tempo limite de 3600 segundos.

6.4.2.2 Experimento 2: heurística construtiva

Para avaliarmos a eficácia da heurística construtiva CH, comparamos sua versão determinística com cinco execuções de sua variante estocástica. Na Tabela 6.2, os tempos computacionais são reportados nas colunas $t(s)$, enquanto a quarta e a quinta colunas se referem a desvios médios obtidos ao considerarmos as oito regras de despacho. $Gap(\%)$ trata da média dentre todas as soluções encontradas e $Gap^*(\%)$ se restringe às melhores soluções para cada instância de uma combinação tarefa-máquina-ferramenta. No que se refere à heurística CH-R, as colunas $Gap(\%)$ e $Gap^*(\%)$ seguem a mesma semântica, porém considerando as oito regras de despacho para cada uma das cinco execuções realizadas.

Para uma melhor compreensão das tabelas, consideremos $Inst$ o conjunto de instâncias, C_{Inst} uma combinação tarefa-máquina-ferramenta, $r = \{1, \dots, 8\}$ o conjunto de regras de despacho, $\kappa = \{1, \dots, 5\}$ os índices referentes às cinco repetições relacionadas às versões estocásticas dos métodos e $h = \{1, 2, 3\}$ as três heurísticas analisadas, isto é, CH, BL e PR. Definindo s_ι^r uma solução encontrada para a instância ι do grupo de instâncias C_{inst} com relação à regra de despacho r , $s_{\iota\kappa}^r$ uma solução encontrada para a instância ι do grupo de instâncias C_{inst} com relação à regra de despacho r e à repetição κ , e s_ι^* a melhor solução encontrada para a instância ι do grupo de instâncias C_{inst} , dentre todos os métodos CH, BL e PR, obtemos, abaixo, as definições analíticas dos desvios:

$$\text{Determinístico: } Gap_h(\%) = \frac{\sum_{\iota \in C_{inst}} \sum_{r=1}^8 \frac{s_\iota^r - s_\iota^*}{s_\iota^*}}{|C_{inst}| \times 8} \times 100, C_{Inst} \in Inst, h \in \{1, 2, 3\}. \quad (6.19)$$

$$\text{Estocástico: } Gap_h(\%) = \frac{\sum_{\iota \in C_{inst}} \sum_{r=1}^8 \sum_{\kappa=1}^5 \frac{s_{\iota\kappa}^r - s_\iota^*}{s_\iota^*}}{|C_{inst}| \times 8 \times 5} \times 100, C_{Inst} \in Inst, h \in \{1, 2, 3\}. \quad (6.20)$$

$$\text{Determinístico: } Gap_h^*(\%) = \frac{\sum_{\iota \in C_{inst}} \frac{\min_{r \in \{1, \dots, 8\}} \{s_\iota^r\} - s_\iota^*}{s_\iota^*}}{|C_{inst}|} \times 100, C_{Inst} \in Inst, h \in \{1, 2, 3\}. \quad (6.21)$$

$$\text{Estocástico: } Gap_h^*(\%) = \frac{\sum_{\iota \in C_{inst}} \frac{\min_{\substack{r \in \{1, \dots, 8\} \\ \kappa \in \{1, \dots, 5\}}} \{s_{\iota\kappa}^r\} - s_\iota^*}{s_\iota^*}}{|C_{inst}|} \times 100, C_{Inst} \in Inst, h \in \{1, 2, 3\}. \quad (6.22)$$

Vemos que os valores de $Gap(\%)$ e $t(s)$ pouco variam ao inserirmos a característica randômica à heurística CH. O mesmo não acontece com os valores de $Gap^*(\%)$. De fato, como a heurística CH-R é executada cinco vezes, obtém-se soluções de melhor qualidade mais frequentemente que a variante determinística CH, repetida apenas uma vez.

Tabela 6.2: Análise comparativa do desempenho das versões determinística e estocástica da heurística construtiva CH.

$ J $	$ M $	$ T $	CH			CH-R			
			Gap(%)	Gap*(%)	t(s)	Gap(%)	Gap*(%)	t(s)	
6	6	15	35,6	16,5	0,0	32,5	7,9	0,0	
10	5	15	23,1	7,6	0,0	22,3	4,5	0,0	
		20	23,8	9,3	0,0	22,7	5,6	0,0	
		25	23,0	9,2	0,0	23,1	5,2	0,0	
		10	15	19,7	8,0	0,0	19,0	3,7	0,0
15	5	20	20,2	8,0	0,0	20,5	3,8	0,0	
		25	20,3	8,8	0,0	20,1	4,0	0,0	
		15	21,8	11,1	0,0	20,5	5,3	0,0	
		20	20,0	9,4	0,0	19,1	4,7	0,0	
	10	25	19,9	9,5	0,0	20,2	5,6	0,0	
		15	19,2	8,3	0,0	17,6	2,5	0,0	
		20	17,6	7,3	0,0	17,5	3,2	0,0	
		25	18,0	6,9	0,0	17,6	3,1	0,0	
	15	15	16,5	5,5	0,0	16,3	3,0	0,0	
		20	18,1	7,2	0,0	17,5	3,4	0,0	
		25	16,5	7,0	0,0	16,5	2,9	0,0	
		20	5	15	17,2	6,6	0,0	15,8	3,4
20	18,7			8,4	0,0	17,1	4,6	0,0	
25	18,5			8,6	0,0	17,8	5,3	0,0	
10	15			15,6	6,7	0,0	14,6	2,6	0,0
10	20		15,4	6,7	0,0	15,5	3,2	0,0	
	25		15,5	6,4	0,0	14,9	2,7	0,0	
	15		15	14,3	5,3	0,0	14,1	2,6	0,0
			20	14,4	6,2	0,0	14,2	2,8	0,0
25			14,5	6,3	0,0	14,8	2,9	0,0	
30			10	15	14,2	5,6	0,0	13,6	2,6
	20			13,4	4,7	0,0	12,9	2,0	0,0
	25			13,7	5,5	0,0	13,0	2,1	0,0
	Média			18,5	7,7	0,0	17,9	3,8	0,0

6.4.2.3 Experimento 3: heurística construtiva pós-otimizada

Nas Tabelas 6.3 e 6.4, vemos os resultados decorrentes da aplicação da busca local BL e, posteriormente, do método *Path relinking*, respectivamente. Os significados das colunas são os mesmos explicados na tabela anterior. Observamos que, como o PR é executado apenas uma vez, sem variação dentre as diferentes regras de despacho, não consideramos o desvio médio $Gap(\%)$.

Conforme exposto na Tabela 6.3, obtemos melhorias em todas as classes de instâncias, em comparação às soluções advindas da heurística CH. Com efeito, o desvio $Gap^*(\%)$ foi reduzido de 7,7% a 5,5% no caso determinístico e de 3,8% para 1,6% ao considerarmos o fator estocástico. Os tempos computacionais gastos foram bastante semelhantes e a BL-R também obteve as melhores soluções em um número maior de vezes.

Os resultados finais, alcançados após a aplicação do *path relinking* e sumarizados na Tabela 6.4, seguem o mesmo padrão de comportamento. As soluções da abordagem estocástica Final-R constituem, portanto, o *benchmark* do JSSPTC. Em comparação às soluções determinadas pela busca local BL, o PR foi capaz de reduzir os valores de $Gap^*(\%)$ de 5,5% para 4,5% na versão determinística e 1,6% para 0,5% no caso estocástico.

Nas Figuras 6.4 e 6.5, analisamos o comportamento médio ($Gap(\%)$) das oito regras de despacho testadas. Isso é feito considerando tanto o número de tarefas ($|J|$) quanto o número de máquinas ($|M|$) das instâncias em questão. As linhas dos gráficos nos mostram que não há uma diferença bastante significativa entre as regras. Em uma análise mais microscópica, entretanto, vemos que a regra FDD (do inglês, *Flow due-date*) se sobressai em todas as situações. Percebemos também que o contraste entre as diferentes regras diminui conforme elevamos a complexidade do problema, ou seja, aumentamos o número de tarefas ou máquinas.

Tabela 6.3: Análise comparativa do desempenho das versões determinística e estocástica da busca local BL.

$ J $	$ M $	$ T $	BL			BL-R		
			Gap(%)	Gap*(%)	t(s)	Gap(%)	Gap*(%)	t(s)
6	6	15	32,3	12,3	0,0	29,0	5,2	0,0
10	5	15	19,0	4,3	0,1	19,1	1,6	0,1
		20	19,8	5,4	0,2	19,2	2,0	0,2
		25	18,6	3,8	0,2	19,2	1,1	0,2
		10	17,9	6,4	0,4	17,4	1,9	0,3
15	5	20	17,9	5,6	0,5	18,3	1,9	0,5
		25	18,3	6,5	0,6	18,0	2,2	0,6
		15	19,1	8,6	0,8	17,4	1,2	0,9
		20	16,8	5,9	1,1	16,4	0,9	1,0
	10	25	16,8	5,3	1,4	17,2	0,6	1,4
		15	17,8	6,2	2,3	16,0	1,0	2,4
		20	16,3	5,8	3,0	15,9	1,2	3,1
		25	16,3	5,3	4,0	16,0	1,4	3,8
	15	15	15,1	4,7	5,8	15,2	2,0	5,6
		20	17,0	6,4	6,2	16,3	1,8	6,6
		25	15,2	5,7	8,9	15,4	1,9	8,4
		20	15,1	5,0	3,4	16,3	0,9	3,5
20	5	20	15,5	4,8	5,0	14,3	0,8	4,7
		25	15,5	4,1	6,2	14,9	1,1	5,9
		10	14,8	5,5	10,5	13,6	1,3	10,6
		20	14,2	4,9	13,9	14,2	1,8	13,7
	10	25	14,1	5,2	16,6	13,7	1,1	16,2
		15	13,4	3,6	23,2	13,3	1,9	22,2
		20	13,6	5,0	25,5	13,4	2,0	26,5
		25	13,4	5,6	36,7	13,7	1,5	35,8
	15	15	13,2	4,5	78,3	12,7	2,0	77,6
		20	12,4	3,8	96,9	12,1	0,7	94,4
		25	12,5	4,4	128,8	11,9	0,6	127,4
		Média			16,5	5,5	17,2	16,0

Tabela 6.4: Análise comparativa do desempenho das versões finais, determinística e estocástica, da heurística construtiva CH combinada ao Path relinking PR.

$ J $	$ M $	$ T $	Final, com PR		Final-R, com PR		
			Gap*(%)	t(s)	Gap(%)	Gap*(%)	t(s)
6	6	15	10,7	0,2	8,6	3,6	0,2
10	5	15	3,9	1,4	5,1	0,7	1,5
		20	4,1	1,9	5,2	1,0	2,0
		25	3,5	2,4	4,7	0,6	2,3
		10	4,6	7,4	4,4	0,5	6,2
15	5	20	4,5	7,9	4,7	0,3	9,0
		25	5,3	10,7	4,8	0,5	11,3
		15	7,9	10,2	4,9	0,0	10,7
		20	5,1	12,7	4,5	0,2	12,9
	10	25	4,5	16,2	5,1	0,3	16,5
		15	5,1	49,8	4,3	0,2	53,1
		20	4,6	70,4	4,0	0,2	73,5
		25	4,3	81,8	4,0	0,3	89,6
	15	15	3,6	207,9	4,1	0,5	210,9
		20	5,2	206,0	4,4	0,3	243,6
		25	3,9	329,8	4,3	0,4	315,6
		20	3,6	44,3	3,7	0,4	41,8
20	5	20	4,0	62,2	4,1	0,3	56,8
		25	3,6	70,7	4,2	0,5	70,4
		10	4,4	265,7	3,7	0,1	279,7
		20	4,1	399,6	4,4	0,7	387,2
	10	25	3,9	412,1	3,7	0,2	436,3
		15	3,4	1063,4	3,4	0,3	1406,7
		20	2,9	1590,2	3,9	0,5	1593,6
		25	3,7	2236,4	3,9	0,5	2056,1
	15	15	3,6	1851,6	4,0	0,7	2175,6
		20	3,1	2484,4	3,5	0,2	2630,6
		25	3,5	3768,9	3,3	0,2	3332,7
Média			4,5	545,2	4,4	0,5	554,5

Figura 6.4: Comparação do desempenho das regras de despacho - Heurística Construtiva Determinística.

Figura 6.5: Comparação do desempenho das regras de despacho - Heurística Construtiva Estocástica.

6.5 Conclusões

Neste capítulo, abordamos o problema de sequenciamento de tarefas em um conjunto de máquinas paralelas inseridas em um ambiente do tipo *job shop*, com o objetivo de minimizar o *makespan* (JSSPTC). Este é um problema muito pouco estudado na literatura, por isso propomos, inicialmente, uma formulação matemática que melhor o definisse. Em seguida, como método de solução, implementamos uma heurística construtiva baseada em regras de despacho com escolhas tanto determinísticas quanto estocásticas, a qual foi combinada a procedimentos de melhoria como busca local e *path relinking*. A fim de melhor avaliar os métodos propostos, desenvolvemos também um novo conjunto de 1275 instâncias, adaptadas de exemplares retirados da *OR-Library* e contendo até 300 operações. Os resultados computacionais obtidos atestam a dificuldade em se resolver o JSSPTC de forma exata. Isto, em si, justifica o desenvolvimento do método heurístico que trata de maneira eficiente os obstáculos encontrados pelo modelo em conjunto com o software CPLEX.

Capítulo 7

Conclusões

Neste trabalho, efetuamos um estudo acerca do sequenciamento de tarefas em um sistema flexível de manufatura (SFM), de modo a otimizar o manejo de ferramentas inseridas e removidas nos magazines das máquinas. Esta é uma temática reconhecidamente crucial para a implementação eficaz de um SFM, uma vez que a parcela de tempo relacionada à gestão das ferramentas influencia de forma significativa o tempo total gasto ao final do processamento de todas as tarefas.

Nosso estudo se concentra em três abordagens distintas, porém em níveis crescentes de abrangência, para as quais foram implementados diversos métodos de resolução.

Primeiramente, no Capítulo 4, desenvolvemos quatro versões de um algoritmo de redução de domínio baseado em transformações do problema da minimização de trocas de ferramentas em uma única máquina flexível (MTSP, do inglês *Minimization of tool switches problem*) em extensões do problema do caixeiro viajante (TSP, do inglês *Traveling salesman problem*), com o intuito de nos beneficiarmos da grande variedade de métodos heurísticos eficientes já estabelecidos para a resolução do TSP. Por meio de uma extensa gama de experimentos computacionais, concluímos a competitividade de nossos métodos no caso de instâncias pequenas e médias. No caso de instâncias grandes e críticas, onde existe uma distância considerável entre o número total de ferramentas (l) e a capacidade da máquina (C), a heurística de [Chaves et al. \(2012\)](#) se sobressai na minimização do número de trocas de ferramentas, apesar de exigir um tempo computacional mais elevado.

Em seguida, no Capítulo 5, tratamos do sequenciamento de tarefas em um conjunto de máquinas flexíveis, paralelas e idênticas (IPMTC, do inglês *Identical parallel machines problem with tooling constraints*). Para sua resolução, propomos duas modelagens matemáticas inspiradas em modelos proeminentes do MTSP, além de uma meta-heurística ALNS (do inglês, *Adaptive large neighborhood search*). Experimentamos nossa heurística através de dois métodos para determinação de soluções iniciais, além de em seis cenários decorrentes da combinação de parâmetros calibrados para a ALNS. Os resultados computacionais atestam a superioridade de nosso método em todas as classes de instâncias geradas, em comparação com o mais recente *benchmark* disponível na literatura do IPMTC. Este capítulo resultou no seguinte artigo científico:

- Beezão, A. C., Cordeau, J.-F., Laporte, G., Yanasse, H. H., *Scheduling identical parallel machines with tooling constraints*, European Journal of Operational Research. Disponível online desde 11 de agosto de 2016.

Por último, no Capítulo 6 abordamos a minimização do *makespan* gerado pelo sequenciamento de tarefas em um conjunto de máquinas paralelas dispostas em um ambiente de manufatura do tipo *job shop* (JSSPTC, do inglês *Job shop scheduling problem with tooling constraints*). Desenvolvemos, inicialmente, uma modelagem matemática combinando as restrições clássicas do *job shop* a restrições que controlam a movimentação de ferramentas nos magazines das máquinas. Devido às dificuldades encontradas na solução exata do modelo, implementamos oito regras de despacho como parte de uma heurística construtiva, denominada CH. Por meio dos experimentos computacionais, observamos que a heurística proposta apresentou um comportamento eficiente e eficaz na

maioria dos grupos de instâncias testados. Esperamos, portanto, publicar os avanços decorrentes desta abordagem em um periódico internacional com seletiva política editorial.

7.1 Perspectivas de trabalho

Durante o desenvolvimento dos métodos descritos nesta tese e, posteriormente, a análise dos resultados obtidos, identificamos oportunidades de pesquisa promissoras ligadas aos capítulos relacionados ao MTSP, ao IPMTC e ao JSSPTC, as quais descrevemos a seguir.

No Capítulo 4, apesar de todos os esforços para com a determinação de nós de qualidade, observamos um *gap* entre nossos resultados e o *benchmark* da literatura de métodos heurísticos propostos ao MTSP. Pretendemos, então, elaborar um novo conceito de nós cruciais, de modo a diminuir os valores das arestas críticas do grafo MTSP/GTSP.

O Capítulo 5 sugere algumas extensões relevantes para o IPMTC. Podemos citar, por exemplo, o caso em que as máquinas não sejam idênticas, no sentido de possuírem magazines com capacidades distintas e/ou praticarem o compartilhamento de ferramentas. Considerando, também, os bons resultados alcançados pela ALNS, pretendemos, em um futuro próximo, estudar a aplicabilidade desta meta-heurística a problemas correlatos aos abordados nesta tese.

Quanto ao Capítulo 6, consideramos utilizar regras de eliminação de simetrias durante o processamento de tarefas, com o objetivo de tornar mais eficiente a adaptação, ao nosso contexto, do algoritmo TOMATO, proposto por [Hertz e Widmer \(1996\)](#) para a minimização do número de instantes de troca em um ambiente do tipo *job shop*.

Referências Bibliográficas

- Adams et al. (1988)** J. Adams, E. Balas e D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34:391–401. Citado na(s) pág(s). [30](#)
- Agnetis et al. (1997)** A. Agnetis, A. Alfieri, P. Brandimarte e P. Prinsecchi. Joint job/tool scheduling in a flexible manufacturing cell with no on-board tool magazine. *Computer Integrated Manufacturing Systems*, 10:61–68. Citado na(s) pág(s). [28](#)
- Aiex et al. (2003)** R. M. Aiex, S. Binato e M. G. C. Resende. Parallel grasp with path-relinking for job shop scheduling. *Parallel Computing*, 29:393–430. Citado na(s) pág(s). [31](#)
- Al-Fawzan e Al-Sultan (2002)** M. A. Al-Fawzan e K. S. Al-Sultan. A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. *Computers and Industrial Engineering*, 44:35–47. Citado na(s) pág(s). [23](#)
- Allahverdi (2015)** A. Allahverdi. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246:345–378. Citado na(s) pág(s). [16](#), [17](#), [29](#)
- Allahverdi et al. (2008)** A. Allahverdi, C. T. Ng, T. C. E. Cheng e M. Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187: 985–1032. Citado na(s) pág(s). [29](#)
- Amaya et al. (2008)** J. E. Amaya, C. Cotta e A. J. Fernández. A memetic algorithm for the tool switching problem. *Lecture Notes in Computer Science*, 5296:190–202. Citado na(s) pág(s). [1](#)
- Applegate e Cook (1991)** D. Applegate e W. Cook. A computational study of the job-shop scheduling problem. *ORSA Journal on Computing*, 3:149–156. Citado na(s) pág(s). [31](#)
- Armentano e Scrich (2000)** V. A. Armentano e C. R. Scrich. Tabu search for minimizing total tardiness in a job shop. *International Journal of Production Economics*, 63:131–140. Citado na(s) pág(s). [31](#)
- Atan e Pandit (1996)** T. S. Atan e R. Pandit. Auxiliary tool allocation in flexible manufacturing systems. *European Journal of Operational Research*, 89:642–659. Citado na(s) pág(s). [30](#)
- Avci e Akturk (1996)** S. Avci e M. S. Akturk. Tool magazine arrangement and operations sequencing on cnc machines. *Computers & Operations Research*, 23:1069–1081. Citado na(s) pág(s). [26](#)
- Ayres (1988)** R. U. Ayres. Future trends in factory automation. *Manufacturing review*, 1:93–103. Citado na(s) pág(s). [2](#)
- Balakrishnan e Chakravarty (2001)** N. Balakrishnan e A.K. Chakravarty. Opportunistic re-tooling of a flexible machine subject to failure. *Naval Research Logistics*, 48:79–97. Citado na(s) pág(s). [26](#)
- Balas (1969)** E. Balas. Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. *Operations Research*, 17:941–957. Citado na(s) pág(s). [30](#), [31](#), [32](#)

- Balas e Vazacopoulos (1998)** E. Balas e A. Vazacopoulos. Guided local search with shifting bottleneck for job shop scheduling. *Management Science*, 44:262–275. Citado na(s) pág(s). [31](#)
- Bard (1988)** J. F. Bard. A heuristic for minimizing the number of tool switches on flexible machine. *IIE Transactions*, 20:382–391. Citado na(s) pág(s). [2](#), [5](#), [22](#), [26](#)
- Barnea e Sipper (1993)** A. Barnea e D. Sipper. Setup reduction in pcb automated assembly. *Computer Integrated Manufacturing Systems*, 6:18–26. Citado na(s) pág(s). [27](#)
- Becceneri et al. (2004)** J.C. Becceneri, H.H. Yanasse e N.Y. Soma. A method for solving the minimization of the maximum number of open stacks problem within a cutting process. *Computers & Operations Research*, 31:2315–2332. Citado na(s) pág(s). [27](#)
- Behnamian et al. (2009)** J. Behnamian, M. Zandieh e S. M. T. Fatemi Ghomi. Parallel-machine scheduling problems with sequence-dependent setup times using an aco, sa and vns hybrid algorithm. *Expert Systems with Applications*, 36:9637–9644. Citado na(s) pág(s). [29](#)
- Behzad e Modarres (2002)** A. Behzad e M. Modarres. A new efficient transformation of the generalized traveling salesman problem into traveling salesman problem. Em *Proceedings of the 15th International Conference of Systems Engineering (ICSE)*, 1–5, Las Vegas, USA. Citado na(s) pág(s). [23](#), [33](#), [38](#)
- Belady (1966)** L. A. Belady. A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, 5:78–101. Citado na(s) pág(s). [27](#)
- Berrada e Stecké (1986)** M. Berrada e K. E. Stecké. A branch and bound approach for machine load balancing in flexible manufacturing systems. *Management Science*, 32:1316–1335. Citado na(s) pág(s). [29](#)
- Binato et al. (2002)** S. Binato, W. J. Hery, D. M. Loewenstern e M. G. C. Resende. *Essays and Surveys in Metaheuristics*, chapter A Grasp for Job Shop Scheduling, 59–79. Springer US. Citado na(s) pág(s). [31](#)
- Birgin et al. (2015)** E. G. Birgin, J. E. Ferreira e D. P. Ronconi. List scheduling and beam search methods for the flexible job shop scheduling problem with sequencing flexibility. *European Journal of Operational Research*. Citado na(s) pág(s). [31](#)
- Blazewicz e Finke (1994)** J. Blazewicz e G. Finke. Scheduling with resource management in manufacturing systems. *European Journal of Operational Research*, 76:1–14. Citado na(s) pág(s). [6](#), [25](#), [27](#)
- Borenstein e Becker (1994)** D. Borenstein e J.L. Becker. Simflex: um avaliador de sistemas flexíveis de manufatura. *Revista de Administração*, 29:77–84. Citado na(s) pág(s). [21](#)
- Brandimarte (1993)** P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41:157–183. Citado na(s) pág(s). [31](#)
- Brucker et al. (1994)** P. Brucker, B. Jurisch e B. Sievers. A branch and bound algorithm for job-shop scheduling problem. *Discrete Applied Mathematics*, 49:105–127. Citado na(s) pág(s). [31](#)
- Burger e Jacobs (2015)** A. P. Burger e C. G. Jacobs. Scheduling multi-colour print jobs with sequence-dependent setup times. *J Sched*, 18:131–145. Citado na(s) pág(s). [27](#)
- Buyurgana et al. (2004)** N. Buyurgana, C. Saygin e S. E. Kilic. Tool allocation in flexible manufacturing systems with tool alternatives. *Robotics and Computer-Integrated Manufacturing*, 20:341–349. Citado na(s) pág(s). [30](#)
- Carlier e Pinson (1989)** J. Carlier e E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35:164–176. Citado na(s) pág(s). [31](#)

- Carpaneto et al. (1995)** G. Carpaneto, M. Dell’Amico e P. Toth. Algorithm 750: Cdt: A subroutine for the exact solution of large-scale, asymmetric traveling salesman problems. *ACM Transactions on Mathematical Software*, 21(4):410–415. Citado na(s) pág(s). [38](#)
- Catanzaro et al. (2015)** D. Catanzaro, L. E. N. Gouveia e M. Labbé. Improved integer linear programming formulations for the job sequencing and tool switching problem. *European Journal of Operational Research*, 244:766–777. Citado na(s) pág(s). [8](#), [14](#), [15](#), [24](#)
- Caumond et al. (2008)** A. Caumond, P. Lacomme e N. Tchernev. A memetic algorithm for the job-shop with time-lags. *Computers & Operations Research*, 35:2331–2356. Citado na(s) pág(s). [31](#)
- Chan e Swarnkar (2006)** F. T. S. Chan e R. Swarnkar. Ant colony optimization approach to a fuzzy goal programming model for a machine tool selection and operation allocation problem in an fms. *Robotics and Computer-Integrated Manufacturing*, 22:353–362. Citado na(s) pág(s). [30](#)
- Chaves et al. (2012)** A. A. Chaves, E. L. F. Senne e H. H. Yanasse. Uma nova heurística para o problema de minimização de trocas de ferramentas. *Gestão e Produção*, 19:17–30. Citado na(s) pág(s). [7](#), [23](#), [36](#), [48](#), [51](#), [53](#), [60](#), [61](#), [67](#), [91](#)
- Chaves et al. (2016)** A. A. Chaves, L. A. N. Lorena, E. L. F. Senne e M.G.C. Resende. Hybrid method with cs and brkga applied to the minimization of tool switches problem. *Computers & Operations Research*, 67:174–183. Citado na(s) pág(s). [23](#)
- Cheng et al. (2013)** T. C. E. Cheng, B. Peng e Z. Lü. A hybrid evolutionary algorithm to solve the job shop scheduling problem. *Annals of Operations Research*, 1–15. Citado na(s) pág(s). [31](#)
- Coffman e Denning (1973)** E. G. Coffman e J. P. Denning. *Operating systems theory*. Englewood Cliffs: Prentice Hall. Citado na(s) pág(s). [21](#), [27](#)
- Coffman et al. (1978)** E. G. Coffman, M. R. Garey e D. S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, 7:1–17. Citado na(s) pág(s). [28](#)
- Crama (1997)** Y. Crama. Combinatorial optimization models for production scheduling in automated manufacturing systems. *European Journal of Operational Research*, 99:136–153. Citado na(s) pág(s). [21](#), [27](#)
- Crama e Oerlemans (1996)** Y. Crama e A.G. Oerlemans. *Production planning in automated manufacturing*. Springer Verlag. Citado na(s) pág(s). [21](#)
- Crama et al. (1994)** Y. Crama, A. W. J. Kolen, A. G. Oelermans e F. C. R. Spieksma. Minimizing the number of tool switches on a flexible machine. *International Journal of Flexible Manufacturing Systems*, 6:33–54. Citado na(s) pág(s). [2](#), [6](#), [16](#), [22](#), [26](#), [28](#), [47](#), [66](#)
- Crama et al. (2007)** Y. Crama, L. S. Moonen, F. C. R. Spieksma e E. Talloen. The tool switching problem revisited. *European Journal of Operational Research*, 182:952–957. Citado na(s) pág(s). [26](#), [27](#)
- Crowder et al. (1983)** H. Crowder, E.L. Johnson e M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31:803–834. Citado na(s) pág(s). [22](#)
- Cumings (1986)** S. Cumings. Developing integrated tooling systems: a case study at garret turbine engine company. Em *Proceedings of the Fall Industrial Engineering Conference (Institute of Industrial Engineers, Norcross, GA)*, 21–26, Boston, USA. Citado na(s) pág(s). [2](#)
- da Cunha Jr. (2012)** J. J. da Cunha Jr. Programação de produção em máquinas cnc para o curto prazo. Dissertação de Mestrado, Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, MG. Citado na(s) pág(s). [xvii](#), [7](#)

- Dantzig et al. (1954)** G. B. Dantzig, D. R. Fulkerson e S. M. Johnson. Solution of a large-scale traveling salesman problem. *Operations Research*, 2:393–410. Citado na(s) pág(s). [10](#)
- Das et al. (2009)** K. Das, M. Baki e X. Li. Optimization of operation and changeover time for production planning and scheduling in a flexible manufacturing system. *Computers and Industrial Engineering*, 56:283–293. Citado na(s) pág(s). [30](#)
- de la Banda e Stuckey (2007)** M.G. de la Banda e P.J. Stuckey. Dynamic programming to minimize the maximum number of open stacks. *INFORMS Journal on Computing*, 19:607–617. Citado na(s) pág(s). [27](#)
- Dell’Amico e Trubian (1993)** M. Dell’Amico e M. Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41:231–252. Citado na(s) pág(s). [31](#)
- Denizel (2003)** M. Denizel. Minimization of the number of tool magazine setups on automated machines: a lagrangean decomposition approach. *Operations Research*, 51:309–320. Citado na(s) pág(s). [6](#), [25](#), [26](#)
- Desrochers e Laporte (1991)** M. Desrochers e G. Laporte. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36. Citado na(s) pág(s). [11](#)
- Djellab et al. (2000)** H. Djellab, K. Djellab e M. Gourgand. A new heuristic based on a hypergraph representation for the tool switching problem. *International Journal of Production Economics*, 64:165–176. Citado na(s) pág(s). [22](#)
- Du e Leung (1990)** J. Du e J. Y.-T. Leung. Minimizing total tardiness on one machine is np-hard. *Mathematics of Operations Research*, 15:483–495. Citado na(s) pág(s). [31](#)
- Faggioli e Bentivoglio (1998)** E. Faggioli e C.A. Bentivoglio. Heuristic and exact methods for the cutting sequencing problem. *European Journal of Operational Research*, 110:564–575. Citado na(s) pág(s). [27](#)
- Fathi e Barnette (2002)** Y. Fathi e K. W. Barnette. Heuristic procedures for the parallel machine problem with tool switches. *International Journal of Production Research*, 40:151–164. Citado na(s) pág(s). [26](#), [28](#), [29](#), [66](#), [67](#), [71](#)
- Fathi e Taheri (1989)** Y. Fathi e J. Taheri. A mathematical model for loading the sequencers in a printed circuit pack manufacturing environment. *International Journal of Production Research*, 27:1305–1316. Citado na(s) pág(s). [27](#)
- Fattahi et al. (2007)** P. Fattahi, M. S. Mehrabad e F. Jolai. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligente Manufacturing*, 18:331–342. Citado na(s) pág(s). [31](#)
- Fisher e Thompson (1963)** H. Fisher e G. L. Thompson. *Industrial Scheduling*, chapter Probabilistic Learning Combinations of Local Job Shop Scheduling Rules. Prentice-Hall. Citado na(s) pág(s). [31](#), [32](#)
- Ford (1922)** H. Ford. *My life and work - an autobiography of Henry Ford*. Citado na(s) pág(s). [1](#)
- Fowler et al. (2003)** J. W. Fowler, S. M. Horng e J. K. Cochran. A hybridized genetic algorithm to solve parallel machine scheduling problems with sequence-dependent setups. *International Journal of Industrial Engineering: Theory Applications and Practice*, 10:232–243. Citado na(s) pág(s). [28](#)
- Gaalman e Nawijn (1996)** G.J.C. Gaalman e W.M. Nawijn. Tool sharing in parallel part production. *International Journal of Production Economics*, 46/47:521–533. Citado na(s) pág(s). [26](#)

- Garey e Johnson (1979)** M. Garey e D. S. Johnson. *Computers and Intractability*. Freeman, New York. Citado na(s) pág(s). 16, 28
- Gendreau et al. (1992)** M. Gendreau, A. Hertz e G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40:1086–1094. Citado na(s) pág(s). 22
- Gendreau et al. (2001)** M. Gendreau, G. Laporte e E. M. Guimarães. A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 133:183–189. Citado na(s) pág(s). 30
- Ghiani et al. (2007)** G. Ghiani, A. Grieco e E. Guerriero. An exact solution to the tlp problem in a nc machine. *Robotics and Computer-Integrated Manufacturing*, 23:645–649. Citado na(s) pág(s). 24
- Ghiani et al. (2010)** G. Ghiani, A. Grieco e E. Guerriero. Solving the job sequencing and tool switching problem as a nonlinear least cost hamiltonian cycle problem. *Networks*, 55:379–385. Citado na(s) pág(s). 24
- Ghrayeb et al. (2003)** O. A. Ghrayeb, N. Phojanamongkolkij e P. R. Finch. A mathematical model and heuristic procedure to schedule printed circuit packs on sequence. *International Journal of Production Research*, 41:3849–3860. Citado na(s) pág(s). 27
- Glover (1999)** F. Glover. *Interfaces in Computer Science and Operations Research*, chapter Tabu Search and Adaptive Memory Programming — Advances, Applications and Challenges, 1–75. Springer US. Citado na(s) pág(s). 46
- Gómez e Lorena (1998)** A. T. Gómez e L. A. N. Lorena. Modelagem de sistemas de manufatura flexíveis considerando restrições temporais e a capacidade do magazine. *Gestão e Produção*, 5: 69–80. Citado na(s) pág(s). 5, 23, 25
- Gonçalves et al. (2005)** J. F. Gonçalves, J. J. de M. Mendes e M. G. C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167: 77–95. Citado na(s) pág(s). 31
- González et al. (2015)** M. A. González, A. Oddi, R. Rasconi e R. Varela. Scatter search with path relinking for the job shop with time lags and setup times. *Computers & Operations Research*, 60:37–54. Citado na(s) pág(s). 31, 32
- Gray et al. (1993)** A. E. Gray, A. Seidmann e K. E. Stecke. A synthesis of decision models for tool management in automated manufacturing. *Management Science*, 39:549–567. Citado na(s) pág(s). 2, 21
- Gray et al. (1988)** A.E. Gray, A. Seidmann e K.E. Stecke. A synthesis of tool-management issues and decision problems in automated manufacturing. Relatório Técnico 590-b, School of Business Administration - University of Michigan. Citado na(s) pág(s). 21
- Grimes e Hebrard (2015)** D. Grimes e E. Hebrard. Solving variants of the job shop scheduling problem through conflict-directed search. *INFORMS Journal on Computing*, 27:268–284. Citado na(s) pág(s). 31
- Gronalt et al. (1997)** M. Gronalt, M. Grunow, H.O. Günther e R. Zeller. A heuristic for component switching on smt placement machine. *International Journal of Production Economics*, 53: 181–190. Citado na(s) pág(s). 26
- Gómez et al. (2012)** A. T. Gómez, A. G. Rodrigues e R. da R. Righi. *Production Scheduling*, chapter Analyzing Different Production Times Applied to the Job Shop Scheduling Problem, 99–112. InTech, New York. Citado na(s) pág(s). 32

- He et al. (1993)** Z. He, T. Yang e D. E. Deal. A multiple-pass heuristic rule for job shop scheduling with due dates. *International Journal of Production Research*, 31:2677–2692. Citado na(s) pág(s). [31](#)
- Henry-Labordere (1969)** A.L. Henry-Labordere. The record balancing problem: a dynamic programming solution of a generalized traveling salesman problem. *RIBO*, B-2:736–743. Citado na(s) pág(s). [33](#)
- Hertz e Widmer (1996)** A. Hertz e M. Widmer. An improved tabu search approach for solving the job shop scheduling problem with tooling constraints. *Discrete Applied Mathematics*, 65: 319–345. Citado na(s) pág(s). [26](#), [32](#), [81](#), [92](#)
- Hertz et al. (1998)** A. Hertz, G. Laporte, M. Mittaz e K. Stecke. Heuristics for minimizing tool switches when scheduling part types on a flexible machine. *IEE Transactions*, 30:689–694. Citado na(s) pág(s). [12](#), [22](#)
- Hirabayashi et al. (1984)** R. Hirabayashi, H. Suzuki e N. Tsuchiya. Optimal tool module design problem for nc machine tools. *Journal of the Operations Research Society of Japan*, 27:205–228. Citado na(s) pág(s). [25](#)
- Hoh e Kim (1997)** H.-K. Hoh e Y.-D. Kim. Due-date based loading and scheduling methods for a flexible manufacturing system with an automatic tool transporter. *International Journal of Production Research*, 35:2989–3003. Citado na(s) pág(s). [28](#)
- Holthaus e Rajendran (1997)** O. Holthaus e C. Rajendran. Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics*, 48:87–105. Citado na(s) pág(s). [31](#), [79](#)
- Hop e Nagarur (2004a)** N. V. Hop e N. N. Nagarur. The scheduling problem of pcbs for multiple non-identical parallel machines. *European Journal of Operational Research*, 158:577–594. Citado na(s) pág(s). [2](#)
- Hop e Nagarur (2004b)** N. Van Hop e N. N. Nagarur. The scheduling problem of pcbs for multiple non-identical parallel machines. *European Journal of Operational Research*, 158:577–594. Citado na(s) pág(s). [29](#)
- Hu e Yao (2011)** D. Hu e Z. Yao. Parallel machines scheduling with sequence-dependent setup times constraints. *Advanced Science Letters*, 4:2528–2531. Citado na(s) pág(s). [29](#)
- Hwang et al. (2005)** H.-C. Hwang, K. Lee e S. Y. Chang. The effect of machine availability on the worst-case performance of lpt. *Discrete Applied Mathematics*, 148:49–61. Citado na(s) pág(s). [28](#)
- Hwang e Shogan (1989)** S. Hwang e A.W. Shogan. Modelling and solving an fms part selection problem. *International Journal of Production Research*, 27:1349–1366. Citado na(s) pág(s). [25](#)
- Ingersoll (1984)** Engineers Ingersoll. *The FMS Report*. IFS Publications Ltd. Citado na(s) pág(s). [21](#)
- Jayamohan e Rajendran (2000)** M. S. Jayamohan e C. Rajendran. New dispatching rules for shop scheduling: a step forward. *International Journal of Production Research*, 38:563–586. Citado na(s) pág(s). [31](#), [79](#)
- Jonker e Volgenant (1983)** R. Jonker e T. Volgenant. Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, 2:161–163. Citado na(s) pág(s). [38](#)
- Keung et al. (2001)** K. W. Keung, W. H. Ip e T. C. Lee. A genetic algorithm approach to the multiple machine tool selection problem. *Journal of Intelligent Manufacturing*, 12:331–342. Citado na(s) pág(s). [30](#)

- Konak et al. (2008)** A. Konak, S. Kulturel-Konak e M. Azizoglu. Minimizing the number of tool switching instants in flexible manufacturing systems. *International Journal of Production Economics*, 116:298–307. Citado na(s) pág(s). [6](#), [25](#)
- Koulamas e Kypariris (2009)** C. P. Koulamas e G. J. Kypariris. A modified lpt algorithm for the two uniform parallel machine makespan minimization problem. *European Journal of Operational Research*, 96:61–68. Citado na(s) pág(s). [29](#)
- Koulamas (1992)** C.P. Koulamas. Design of tool magazine and operational areas for a multi-tool machine. *European Journal of Operational Research*, 63:240–253. Citado na(s) pág(s). [26](#)
- Kuhpfahl e Bierwirth (2016)** J. Kuhpfahl e C. Bierwirth. A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. *Computers & Operations Research*, 66:44–57. Citado na(s) pág(s). [31](#)
- Kumar e Sridharan (2009)** N. S. Kumar e R. Sridharan. Simulation modeling and analysis of part and tool flow control decisions in a flexible manufacturing system. *Robotics and Computer-Integrated Manufacturing*, 25:829–838. Citado na(s) pág(s). [30](#)
- Kurz e Askin (2001)** M. E. Kurz e R. G. Askin. Heuristic scheduling of parallel machines with sequence-dependent setup times. *International Journal of Production Research*, 39:3747–3769. Citado na(s) pág(s). [28](#)
- Kusiak e Chow (1986)** A. Kusiak e W.S. Chow. Efficient solving of the group technology problem. Relatório Técnico 686, Department of mechanical engineering - University of Manitoba. Citado na(s) pág(s). [23](#)
- Laarhoven et al. (1992)** P. J. M. Van Laarhoven, E. H. L. Aarts e J. K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40:113–125. Citado na(s) pág(s). [30](#)
- Lamond e Sodhi (2003)** B.F. Lamond e M.S. Sodhi. Minimizing the expected processing time on a flexible machine with random tool lives. Relatório técnico, Université Laval, Québec, Canada. Citado na(s) pág(s). [26](#)
- Laporte et al. (1996)** G. Laporte, A. Asef-Varizi e C. Sriskandarajah. Some applications of the generalized travelling salesman problem. *The Journal of the Operational Research Society*, 47:1461–1467. Citado na(s) pág(s). [33](#)
- Laporte et al. (2004)** G. Laporte, J. J. Salazar e F. Semet. Exact algorithms for the job sequencing and tool switching problem. *IIE Transactions*, 36:37–45. Citado na(s) pág(s). [7](#), [8](#), [9](#), [10](#), [11](#), [12](#), [14](#), [22](#), [24](#), [25](#), [28](#), [58](#)
- Lawler et al. (1993)** E. L. Lawler, A. H. G. R. Kan e D. B. Shmoys. *Logistics of Production and Inventory*, chapter Sequencing and scheduling: Algorithms and complexity, 445–522. Elsevier. Citado na(s) pág(s). [30](#)
- Lee (1991)** C.-Y. Lee. Parallel machines scheduling with non-simultaneous machine available time. *Discrete Applied Mathematics*, 30:53–61. Citado na(s) pág(s). [28](#)
- Lee e Massey (1988)** C.-Y. Lee e J. D. Massey. Multiprocessor scheduling: Combining lpt and multifit. *Discrete Applied Mathematics*, 20:233–242. Citado na(s) pág(s). [29](#)
- Lee et al. (2000)** C.-Y. Lee, Y. He e G. Tang. A note on “parallel machine scheduling with non-simultaneous machine available time”. *Discrete Applied Mathematics*, 100:133–135. Citado na(s) pág(s). [28](#)
- Lee et al. (2010)** Z.-J. Lee, S.-W. Lin e K.-C. Ying. Scheduling jobs on dynamic parallel machines with sequence-dependent setup times. *International Journal of Advanced Manufacturing Technology*, 47:773–781. Citado na(s) pág(s). [29](#)

- Leon e Peters (1998)** V.J. Leon e A. Peters. A comparison of setup strategies for printed circuit board assembly. *Computers and Industrial Engineering*, 34:219–234. Citado na(s) pág(s). [27](#)
- Lien et al. (1993)** Y.-N. Lien, E. Ma e B. W.-S. Wah. Transformation of the generalized traveling salesman problem into the standard traveling salesman problem. *Information Sciences*, 74:177–189. Citado na(s) pág(s). [33](#), [38](#)
- Lin e Kernighan (1973)** S. Lin e B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516. Citado na(s) pág(s). [38](#)
- Linhares e Yanasse (2002)** A. Linhares e H.H. Yanasse. Connections between cutting-pattern sequencing, vlsi design, and exible machines. *Computers & Operations Research*, 29:1759–1772. Citado na(s) pág(s). [27](#)
- Madsen (1988)** O.B.G. Madsen. An application of travelling-salesman routines to solve pattern-allocation problems in the glass industry. *Journal of the Operational Research Society*, 39:249–256. Citado na(s) pág(s). [27](#)
- Marvizadeh e Choobineh (2013)** S. Z. Marvizadeh e F.F. Choobineh. Reducing the number of setups for cnc punch presses. *Omega*, 41:226–235. Citado na(s) pág(s). [25](#)
- Matzliach (1998)** B. Matzliach. The online tool switching problem with non-uniform tool size. *International Journal of Production Research*, 36:3407–3420. Citado na(s) pág(s). [26](#)
- Matzliach e Tzur (2000)** B. Matzliach e M. Tzur. Storage management of items in two levels of availability. *European Journal of Operational Research*, 121:363–379. Citado na(s) pág(s). [26](#)
- Mcgeoch e Sleator (1991)** L. A. Mcgeoch e D. D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6:816–825. Citado na(s) pág(s). [27](#)
- Meeran e Morshed (2012)** S. Meeran e M. S. Morshed. A hybrid genetic tabu search algorithm for solving job shop scheduling problems: a case study. *Journal of Intelligent Manufacturing*, 23:1063–1078. Citado na(s) pág(s). [31](#)
- Melnyk et al. (1989)** S.A. Melnyk, S. Ghosh e G.L. Ragatz. Tooling constrains and shop floor scheduling: a simulation study. *Journal of Operations Management*, 8:69–89. Citado na(s) pág(s). [21](#)
- Mendes et al. (2002)** A. S. Mendes, F. M. Muller, P. M. França e P. Moscato. Comparing meta-heuristic approaches for parallel machine scheduling problems. *Production Planning and Control*, 13:143–154. Citado na(s) pág(s). [29](#)
- Miller et al. (1960)** C. E. Miller, A. W. Tucker e R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery*, 7:326–329. Citado na(s) pág(s). [11](#)
- Mohamed et al. (1999)** Z. M. Mohamed, A. Kumar e J. Motwani. An improved part grouping model for minimizing makespan in fms. *European Journal of Operational Research*, 116:171–182. Citado na(s) pág(s). [29](#)
- Montazeri e Wassenhove (1990)** M. Montazeri e L. N. Van Wassenhove. Analysis of scheduling rules for an fms. *International Journal of Production Research Volume 28, Issue 4, 1990*, 28:785–802. Citado na(s) pág(s). [31](#)
- Morales (2009)** Roy Andres Gomez Morales. Modelagem e análise de sistemas flexíveis de manufatura tolerantes à falhas baseado em rede bayesiana e rede de petri. Dissertação de Mestrado, Escola Politécnica da Universidade de São Paulo. Citado na(s) pág(s). [2](#)

- Novas e Henning (2014)** J. M. Novas e G. P. Henning. Integrated scheduling of resource-constrained flexible manufacturing systems using constraint programming. *Expert Systems with Applications*, 41:2286–2299. Citado na(s) pág(s). [30](#)
- Novicki e Smutnicki (1996)** E. Novicki e C. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42:797–813. Citado na(s) pág(s). [31](#), [32](#)
- Oerlemans (1992)** A. G. Oerlemans. *Production planning for flexible manufacturing systems*. Tese de Doutorado, University of Limburg, Maastricht, Holland. Citado na(s) pág(s). [2](#), [6](#), [16](#)
- Özpeynirci (2015)** S. Özpeynirci. A heuristic approach based on time-indexed modelling for scheduling and tool loading in flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology*, 47:1269–1274. Citado na(s) pág(s). [30](#)
- Panwalkar e Islander (1977)** S. S. Panwalkar e W. Islander. A survey of scheduling rules. *Operations Research*, 25:45–61. Citado na(s) pág(s). [28](#)
- Parker (1995)** R. G. Parker. *Deterministic Scheduling Theory*. Citado na(s) pág(s). [28](#)
- Peng et al. (2015)** B. Peng, Z. Lu e T. C. E. Cheng. A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research*, 53:154–164. Citado na(s) pág(s). [31](#)
- Persi et al. (1999)** P. Persi, W. Ukovich, R. Pesenti e M. Nicolich. A hierarchic approach to production planning and scheduling of a flexible manufacturing system. *Robotics and Computer Integrated Manufacturing*, 15:373–385. Citado na(s) pág(s). [29](#)
- Pinto (2004)** M. J. Pinto. *Algumas contribuições à resolução do problema de corte integrado ao problema de sequenciamento dos padrões*. Tese de Doutorado, Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP. Citado na(s) pág(s). [27](#)
- Pinto e Yanasse (2001)** M.J. Pinto e H.H. Yanasse. O problema de corte e sequenciamento de padrões: uma abordagem integrada. Em *Anais do I Workshop de Computação Aplicada*, 109–111, São José dos Campos, Brasil. Citado na(s) pág(s). [27](#)
- Pisinger e Ropke (2007)** D. Pisinger e S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435. Citado na(s) pág(s). [60](#), [66](#)
- Privault e Finke (1995)** C. Privault e G. Finke. Modelling a tool switching problem on a single nc-machine. *Journal of Intelligent Manufacturing*, 6:87–94. Citado na(s) pág(s). [26](#)
- Privault e Finke (2000)** C. Privault e G. Finke. k-server problems with bulk requests: an application to tool switching in manufacturing. Em *Annals of Operations Research*, volume 96, 255–269, Dresden, Germany. Citado na(s) pág(s). [22](#), [27](#)
- Qing-dao-er-ji e Wang (2012)** R. Qing-dao-er-ji e Y. Wang. A new hybrid genetic algorithm for job shop scheduling problem. *Computers & Operations Research*, 39:2291–2299. Citado na(s) pág(s). [31](#)
- Rajagopalan (1986)** S. Rajagopalan. Formulation and heuristic solutions for parts grouping and tool loading in flexible manufacturing systems. Em *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems (1986)*, 311–320, Amsterdam, Holland. Elsevier Science Publishers. Citado na(s) pág(s). [28](#)
- Renaud et al. (1996)** J. Renaud, F. F. Boctor e G. Laporte. A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on Computing*, 8:134–143. Citado na(s) pág(s). [45](#), [46](#), [51](#)

- Ropke e Pisinger (2006)** S. Ropke e D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40: 455–472. Citado na(s) pág(s). [60](#)
- Rupe e Kuo (1997)** J. Rupe e W. Kuo. Solutions to a modified tool loading problem for a single fmm. *International Journal of Production Research*, 35:2253–2268. Citado na(s) pág(s). [26](#)
- Sabah et al. (1985)** U.R. Sabah, D.M. Edward e S. Farouqui. An integer programming application to solve sequencer mix problems in printed circuit board production. *International Journal of Production Research*, 23:543–561. Citado na(s) pág(s). [27](#)
- Salonen et al. (2006)** K. Salonen, C. Raduly-Baka e O. S. Nevalainen. A note on the tool switching problem of a flexible machine. *Computers and Industrial Engineering*, 50:458–465. Citado na(s) pág(s). [23](#)
- Sarin e Chen (1987)** S.C. Sarin e C.S. Chen. The machine loading and tool allocation problem in a flexible manufacturing system. *International Journal of Production Research*, 25:1081–1094. Citado na(s) pág(s). [26](#)
- Scrich (1997)** C. R. Scrich. *Busca Tabu para a programação de tarefas em job shop com datas de entrega*. Tese de Doutorado, Universidade de Campinas (UNICAMP), Campinas, SP. Citado na(s) pág(s). [30](#), [31](#), [77](#), [78](#), [79](#)
- Scrich et al. (2004)** C. R. Scrich, V. A. Armentano e M. Laguna. Tardiness minimization in a flexible job shop: A tabu search approach. *Journal of Intelligent Manufacturing*, 15:103–115. Citado na(s) pág(s). [31](#)
- Senne e Yanasse (2009)** E. L. F. Senne e H. H. Yanasse. Beam search algorithms for minimizing tool switches on a flexible manufacturing system. Em *Proceedings of the XI WSEAS International Conference on Mathematical and Computational Methods in Science and Engineering*, volume 1, 68–72, Baltimore, USA. WSEAS Press. Citado na(s) pág(s). [23](#)
- Shaw (1997)** P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. Relatório técnico, Department of Computer Science, University of Strathclyde, United Kingdom. Citado na(s) pág(s). [63](#)
- Shirazi e Frizelle (2001)** R. Shirazi e G. D. M. Frizelle. Minimizing the number of tool switches on a flexible machine: an empirical study. *International Journal of Production Research*, 39: 3547–3560. Citado na(s) pág(s). [25](#)
- Song e Hwang (2002)** C.-Y. Song e H. Hwang. Optimal tooling policy for a tool switching problem of a flexible machine with automatic tool transporter. *International Journal of Production Research*, 40:873–883. Citado na(s) pág(s). [26](#)
- Srivastava e Chen (1996)** B. Srivastava e W-H. Chen. Batching in production planning for flexible manufacturing systems. *International Journal of Production Economics*, 43:127–137. Citado na(s) pág(s). [25](#)
- Stecke (1983)** K.E. Stecke. Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems. *Management Science*, 29:273–288. Citado na(s) pág(s). [21](#), [25](#)
- Stecke (1985)** K.E. Stecke. Design, planning, scheduling and control problems of flexible manufacturing systems. *Annals of Operations Research*, 3:3–12. Citado na(s) pág(s). [21](#)
- Sule (1992)** D.R. Sule. A heuristic procedure for component scheduling in printed circuit pack sequencers. *International Journal of Production Research*, 30:1191–1208. Citado na(s) pág(s). [27](#)

- Tahar et al. (2006)** D. N. Tahar, F. Yalaoui, C. Chu e L. Amodeo. A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times. *International Journal of Production Economics*, 99:63–73. Citado na(s) pág(s). [29](#)
- Taillard (1989)** E. Taillard. Parallel taboo search technique for the job shop scheduling problem. Relatório técnico, DMA, Ecole Polytechnique Fédérale de Lausanne, Switzerland. Citado na(s) pág(s). [30](#)
- Taillard (1994)** E. D. Taillard. Parallel taboo search techniques for the job shop scheduling problem. *ORSA Journal on Computing*, 6:108–117. Citado na(s) pág(s). [30](#), [32](#), [81](#), [83](#)
- Tang e Denardo (1988a)** C. S. Tang e E. V. Denardo. Models arising from a flexible manufacturing machine, part i: Minimization of the number of tool switches. *Operations Research*, 36:767–777. Citado na(s) pág(s). [2](#), [6](#), [7](#), [8](#), [9](#), [10](#), [12](#), [22](#), [24](#), [26](#), [28](#), [30](#), [32](#), [47](#), [57](#), [77](#)
- Tang e Denardo (1988b)** C. S. Tang e E. V. Denardo. Models arising from a flexible manufacturing machine - part ii: Minimization of the number of switching instants. *Operations Research*, 36:778–784. Citado na(s) pág(s). [6](#), [25](#), [28](#)
- Ten Eikelder et al. (1999)** H. M. M. Ten Eikelder, B. J. M. Aarts, M. G. A. Verhoeven e E. H. L. Aarts. *Meta-Heuristic: Advances and Trends in Local Search Paradigms for Optimization*, chapter Sequential and parallel local search algorithms for job shop scheduling, 359–371. Kluwer Academic Publishers. Citado na(s) pág(s). [31](#)
- Tomek (1986)** P. Tomek. Tooling strategies related to fms management. *The FMS Magazine*, 5:102–107. Citado na(s) pág(s). [2](#)
- Turkcan et al. (2007)** A. Turkcan, S. Akturk e R. H. Storer. Due date and cost-based fms loading, scheduling and tool management. *International Journal of Production Research*, 45:1183–1213. Citado na(s) pág(s). [28](#)
- Tzur e Altman (2004)** M. Tzur e A. Altman. Minimization of tool switches for a flexible manufacturing machine with slot assignment of different tool sizes. *IIE Transactions*, 36:95–110. Citado na(s) pág(s). [26](#)
- Veeramani et al. (1992)** D. Veeramani, D.M. Upton e M.M. Barash. Cutting-tool management in computer-integrated manufacturing. *International Journal of Flexible Manufacturing Systems*, 3/4:237–265. Citado na(s) pág(s). [21](#)
- Wang e Zou (2003)** S. F. Wang e Y. R. Zou. Techniques for the job shop scheduling problem: a survey. *Theory & Practice*, 23:49–55. Citado na(s) pág(s). [31](#)
- Widmer (1991)** M. Widmer. Job shop scheduling with tooling constraints: A tabu search approach. *The Journal of the Operational Research Society*, 42:75–82. Citado na(s) pág(s). [6](#), [17](#), [25](#), [32](#)
- Yanasse (2008)** H. H. Yanasse. Reducing symmetries in the minimization of open stacks problem. Em *Book of Extended Abstracts of XIV Latin Ibero-American Congress on Operations Research*, 1–3, Cartagena, Colombia. Citado na(s) pág(s). [24](#)
- Yanasse (1997)** H. H. Yanasse. On a pattern sequencing problem to minimize the maximum number of open stacks. *European Journal of Operational Research*, 100:454–463. Citado na(s) pág(s). [24](#), [27](#)
- Yanasse e Lamosa (2005)** H. H. Yanasse e M. J. P. Lamosa. An application of the generalized travelling salesman problem: the minimization of tool switches problem. Em *International Annual Scientific Conference of the German Operations Research Society*, 90–100, Bremen, Germany. Citado na(s) pág(s). [22](#), [33](#)

- Yanasse e Rodrigues (2007)** H. H. Yanasse e R. C. M. Rodrigues. A partial ordering enumeration scheme for solving the minimization of tool switches problem. Em *Book of Abstracts - INFORMS Annual Meeting Seattle*, 299–299, Seattle, USA. Citado na(s) pág(s). [8](#), [24](#)
- Yanasse et al. (2009)** H. H. Yanasse, R. C. M. Rodrigues e E. L. F. Senne. Um algoritmo enumerativo baseado em ordenamento parcial para resolução do problema de minimização de trocas de ferramentas. *Gestão e Produção*, 16:370–381. Citado na(s) pág(s). [23](#), [24](#), [47](#), [51](#), [66](#), [84](#)
- Yanasse (2009)** H.H. Yanasse. Um novo limitante inferior para o problema de minimização de trocas de ferramentas. Em *Anais do XLI Simpósio Brasileiro de Pesquisa Operacional*, 2841–2848, Porto Seguro, Brasil. Citado na(s) pág(s). [8](#), [24](#)
- Yanasse (2011)** H.H. Yanasse. Sobre o problema de minimização de trocas de ferramentas: um caso especial. Em *Anais do XLIII Simpósio Brasileiro de Pesquisa Operacional*, 72–72, Ubatuba, Brasil. Citado na(s) pág(s). [24](#)
- Yanasse (2012)** H.H. Yanasse. Um algoritmo polinomial para o caso de árvores do problema de minimização de trocas de ferramentas. Em *Anais do XLIV Simpósio Brasileiro de Pesquisa Operacional*, 1–7, Rio de Janeiro, Brasil. Citado na(s) pág(s). [24](#)
- Yanasse e Pinto (2002)** H.H. Yanasse e M.J. Pinto. The minimization of tool switches problem as a network flow problem with side constraints. Em *Anais do XXXIV Simpósio Brasileiro de Pesquisa Operacional*, Rio de Janeiro, Brasil. Published in CD-ROM, Ref. arq0223. Citado na(s) pág(s). [22](#), [33](#)
- Zahmani et al. (2015)** M. H. Zahmani, B. Atmani, A. Bekrar e N. Aissani. Multiple priority dispatching rules for the job shop scheduling problem. Em *Proceedings of the 3rd International Conference on Control, Engineering & Information Technology (CEIT)*, 1–6, Tlemcen, Algeria. Citado na(s) pág(s). [31](#)
- Zeballos (2010)** L. J. Zeballos. A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 26:725–743. Citado na(s) pág(s). [xvii](#), [1](#), [23](#), [26](#), [30](#)
- Zhang et al. (2007)** C. Zhang, P. Li, Z. Guan e Y. Rao. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, 34:3229–3242. Citado na(s) pág(s). [31](#)
- Zhou et al. (2005)** B. H. Zhou, L. F. Xi e Y. S. Cao. A beam-search-based algorithm for the tool switching problem on a flexible machine. *International Journal of Advanced Manufacturing Technology*, 25:876–882. Citado na(s) pág(s). [5](#), [23](#)