UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

TUANE TONANI YAMADA

MINIMIZAÇÃO DO *TOTAL TARDINESS* EM SISTEMA DE PRODUÇÃO *NO-WAIT*
FLOWSHOP COM MANUTENÇÃO PREVENTIVA

SÃO CARLOS

2019

TUANE TONANI YAMADA

MINIMIZAÇÃO DO *TOTAL TARDINESS* EM SISTEMA DE PRODUÇÃO *NO-WAIT*
FLOWSHOP COM MANUTENÇÃO PREVENTIVA

Dissertação de mestrado apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo como parte dos requisitos para obtenção do título de mestre em Engenharia de Produção.

Área de Concentração: Gestão de Operações

Orientador: Prof. Dr. Marcelo Seido Nagano

SÃO CARLOS

2019

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da
EESC/USP com os dados inseridos pelo(a) autor(a).

Y11m

Yamada, Tuane Tonani

Minimização do total tardiness em sistema de produção no-wait
flowshop com manutenção preventiva / Tuane Tonani Yamada;
orientador Marcelo Seido Nagano. São Carlos, 2019.

Dissertação (Mestrado) - Programa de Pós-Graduação em
Engenharia de Produção e Área de concentração em Processos e
Gestão de Operações - - Escola de Engenharia\Universidade de
São Paulo, 2019.

1. No-wait flowshop. 2. Manutenção preventiva. 3. Total
tardiness. 4. Métodos heurísticos. I. Título.

FOLHA DE JULGAMENTO

Candidata: Engenheira **TUANE TONANI YAMADA**.

Título da dissertação: "Minimização do total tardiness em sistema de produção no-wait flowshop com manutenção preventiva".

Data da defesa: 15/05/2019

Comissão Julgadora:

Resultado:

Prof. Dr. **Marcelo Seido Nagano**

(Orientador)

(Escola de Engenharia de São Carlos/EESC)

Aprovado

Profa. Dra. **Juliana Keiko Sagawa**

(Universidade Federal de São Carlos/UFSCar)

aprovado

Prof. Dr. **Paulo Rogério Politano**

(Universidade Federal de São Carlos/UFSCar)

APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia de Produção:

Prof. Dr. **Marcelo Seido Nagano**

Presidente da Comissão de Pós-Graduação:

Prof. Titular **Murilo Araujo Romero**

*Dedico esse trabalho aos meus pais
pelo incondicional apoio e incentivo
para que mais essa jornada fosse
concluída. Obrigada por estarem
sempre presentes!*

AGRADECIMENTOS

Ao Prof. Dr. Marcelo Seido Nagano pela oportunidade de ter me orientado, pela paciência, por todo conhecimento transmitido e por ter acreditado em mim.

Aos membros da banca, Profa. Dra. Juliana Keiko Sagawa, Prof. Dr. Paulo Rogério Politano e Prof. Dr. Humberto Filipe de Andrade Januário Bettini pelas sugestões que enriqueceram esse trabalho.

Aos meus pais Wilma Ap. Tonani Yamada e Eduardo Yamada, e ao meu irmão Afonso Tonani Yamada por todo amor que me deram, além da educação, conselhos e apoio.

Aos meus colegas de laboratório em especial ao Hugo Miyata pelas recomendações para o desenvolvimento dessa pesquisa.

Aos meus colegas de trabalho que permitiram que essa pesquisa fosse concretizada.

Aos docentes e aos funcionários do Departamento de Engenharia de Produção da EESC – USP e do Instituto de Ciências Matemáticas e de Computação da EESC – USP.

Agradeço a todos por terem contribuído para o meu crescimento científico. Muito obrigada!

RESUMO

YAMADA, T. Y. **Minimização do *total tardiness* em sistema de produção *no-wait flowshop* com manutenção preventiva**, 2019, 114p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo. São Carlos, 2019.

Organizações eficientes são aquelas que conseguem manter equilibradas as vertentes de qualidade, custo e tempo. Em relação ao último, existem várias etapas da cadeia produtiva nas quais o tempo deve ser monitorado. Quando a programação da produção nas indústrias não é priorizado, pode-se incorrer vários efeitos negativos. Um deles, é o atraso em relação à data de entrega, no qual a corporação pode sofrer penalidades financeiras, além de uma exposição negativa para a marca, a qual pode ter sua credibilidade contestada. Dessa forma, essa pesquisa tem por objetivo propor métodos construtivos, que minimize a medida de desempenho *total tardiness* (atraso total). Para aproximar o método à realidade vivenciada pelas indústrias, será considerada a restrição de manutenção preventiva. Além disso, o ambiente de estudo será o contexto de *no-wait flowshop*, no qual as tarefas são processadas continuamente e sem que haja interrupções entre uma operação e outra de uma mesma tarefa. Além da proposição de métodos construtivos para a resolução do problema, apresenta-se uma metaheurística como forma de demonstrar como pode-se aprimorar os resultados gerado pelos métodos construtivos. Experimentações computacionais foram elaboradas e realizadas para comparação dos algoritmos. Dentre as heurísticas construtivas a que apresentou melhor desempenho foi a “EDD + NEH + LS1 + LS2”, na qual utiliza uma lógica de inserção. A metaheurística proposta é baseada no procedimento IG (*iterated greedy*), sendo que há melhora de resultado em relação as heurísticas construtivas. Assim, espera-se que essa pesquisa possa ser utilizada e aplicada pela indústria de manufatura para aumentar a efetividade da programação da produção.

Palavras-chaves: *No-wait flowshop*, manutenção preventiva, *total tardiness*, métodos heurísticos.

ABSTRACT

YAMADA, T. Y. Minimization of total tardiness in flowshop no-wait production system with preventive maintenance, 2019, 114p. Dissertation (Master's degree) - Escola de Engenharia de São Carlos, Universidade de São Paulo. São Carlos, 2019.

Efficient organizations are those that manage to keep the quality, cost and time strands balanced. With respect to the variable time, there are several stages of the production chain in which it must be monitored. When scheduling in companies is not prioritized, several negative effects incur. One of them is the delay in relation to the due date, for which the corporation can suffer financial penalties, in addition to a negative exposure to the brand, which may have its credibility challenged. Therefore, this research aims to propose constructive methods, which minimizes the performance criterion of total tardiness. In order to approximate the method to the reality of the industries, preventive maintenance constraints will be considered. And the environment of the study will be the no-wait flowshop, in which jobs are processed continuously and without interruptions between one operation and another of the same job. In addition to proposing constructive methods to solve the problem, a metaheuristic is presented as a way of demonstrating how to improve the results generated by the constructive methods. Computational experiments were elaborated and performed for comparison of the algorithms. Among the constructive heuristics that presented the best performance was "EDD + NEH + LS1 + LS2", in which it uses an insertion logic. The proposed metaheuristic is based on the IG (iterated greedy) procedure, and there is an improvement of the result in relation to the constructive heuristics. Thus, it is expected that this research can be used and applied by the manufacturing industry to increase the effectiveness of scheduling.

Keywords: No-wait flowshop, preventive maintenance, total tardiness, and heuristic methods.

LISTA DE FIGURAS

Figura 1.1 – Relação entre máquinas e ambientes.....	26
Figura 1.2 – Classificação dos métodos	30
Figura 2.1 – Modelo para condução da revisão bibliográfica sistemática	35
Figura 2.2 – Procedimento iterativo da fase de processamento	36
Figura 3.1 – Gráfico de Gantt (exemplo 1)	42
Figura 3.2 – Gráfico de Gantt para <i>no-idle</i> (exemplo 2).....	42
Figura 3.3 – Gráfico de Gantt para <i>no-wait</i> (exemplo 3).....	42
Figura 3.4 – Gráfico de Gantt com manutenção preventiva.....	43
Figura 4.1 – Gráfico de Gantt com 3 máquinas e 3 tarefas	58
Figura 4.2 – Procedimento de inserção NEH	58
Figura 4.3 – Pseudocódigo do algoritmo Ij + NEH	60
Figura 4.4 – Pseudocódigo do algoritmo MDD + NEH	60
Figura 4.5 – Pseudocódigo do algoritmo EDD + NEH + NSS1	62
Figura 4.6 – Pseudocódigo do algoritmo EDD+ NEH + LS1 + LS2	64
Figura 4.7 – Distância entre as tarefas	65
Figura 4.8 – Inserção da manutenção preventiva (exemplo 1).....	65
Figura 4.9 – Inserção da manutenção preventiva (exemplo 2).....	66
Figura 5.1 – Gráficos dos resultados das heurísticas construtivas por cenário sem política de manutenção preventiva.....	71
Figura 5.2 – Gráficos dos resultados das heurísticas construtivas sem política de manutenção preventiva	71
Figura 5.3 – Gráficos dos tempos de execução das heurísticas construtivas sem política de manutenção preventiva.....	72
Figura 5.4 – Gráficos dos resultados das heurísticas construtivas por cenário com política de manutenção preventiva 1	74
Figura 5.5 – Gráficos dos resultados das heurísticas construtivas com política de manutenção preventiva 1	75
Figura 5.6 – Gráficos dos tempos de execução das heurísticas construtivas com política de manutenção preventiva 1	76
Figura 5.7 – Gráficos dos resultados das heurísticas construtivas por cenário com política de manutenção preventiva 2.....	78

Figura 5.8 – Gráficos dos resultados das heurísticas construtivas com política de manutenção preventiva 2	78
Figura 5.9 – Gráficos dos tempos de execução das heurísticas construtivas com política de manutenção preventiva 2.....	79
Figura 5.10 – Gráficos dos resultados das heurísticas construtivas por cenário com política de manutenção preventiva 3.....	81
Figura 5.11 – Gráficos dos resultados das heurísticas construtivas com política de manutenção preventiva 3	82
Figura 5.12 – Gráficos dos tempos de execução das heurísticas construtivas com política de manutenção preventiva 3.....	83
Figura 6.1 – Pseudocódigo do algoritmo NEH + IG	87
Figura 6.2 – Gráficos dos resultados das metaheurísticas construtivas por cenário sem política de manutenção preventiva	90
Figura 6.3 – Gráficos dos resultados das metaheurísticas construtivas sem política de manutenção preventiva.....	90
Figura 6.4 – Gráficos dos tempos de execução das metaheurísticas construtivas sem política de manutenção preventiva	91
Figura 6.5 – Gráficos dos resultados das metaheurísticas construtivas por cenário com política de manutenção preventiva 1	93
Figura 6.6 – Gráficos dos resultados das metaheurísticas construtivas com política de manutenção preventiva 1.....	94
Figura 6.7 – Gráficos dos tempos de execução das metaheurísticas construtivas com política de manutenção preventiva 1	95
Figura 6.8 – Gráficos dos resultados das metaheurísticas construtivas por cenário com política de manutenção preventiva 2	97
Figura 6.9 – Gráficos dos resultados das metaheurísticas construtivas com política de manutenção preventiva 2.....	97
Figura 6.10 – Gráficos dos tempos de execução das metaheurísticas construtivas com política de manutenção preventiva 2	98
Figura 6.11 – Gráficos dos resultados das metaheurísticas construtivas por cenário com política de manutenção preventiva 3.....	100
Figura 6.12 – Gráficos dos resultados das metaheurísticas construtivas com política de manutenção preventiva 3.....	100

Figura 6.13 – Gráficos dos tempos de execução das metaheurísticas construtivas com política de manutenção preventiva 3	101
--	-----

LISTA DE TABELAS

Tabela 2.1 – Artigos pré-selecionados como referência	38
Tabela 3.1 – Tempos de processamentos (exemplos 1, 2 e 3)	41
Tabela 5.1 – Desvio relativo (%) das heurísticas construtivas por cenário sem política de manutenção preventiva.....	70
Tabela 5.2 – Desvio relativo das heurísticas construtivas sem política de manutenção preventiva	70
Tabela 5.3 – Tempos de execução das heurísticas construtivas sem política de manutenção preventiva	72
Tabela 5.4 – Desvio relativo (%) das heurísticas construtivas por cenário com política de manutenção preventiva 1.....	73
Tabela 5.5 – Desvio relativo das heurísticas construtivas com política de manutenção preventiva 1	74
Tabela 5.6 – Tempos de execução das heurísticas construtivas com política de manutenção preventiva 1	75
Tabela 5.7 – Desvio relativo (%) das heurísticas construtivas por cenário com política de manutenção preventiva 2.....	77
Tabela 5.8 – Desvio relativo das heurísticas construtivas com política de manutenção preventiva 2	77
Tabela 5.9 – Tempos de execução das heurísticas construtivas com política de manutenção preventiva 2	79
Tabela 5.10 – Desvio relativo (%) das heurísticas construtivas por cenário com política de manutenção preventiva 3.....	80
Tabela 5.11 – Desvio relativo das heurísticas construtivas com política de manutenção preventiva 3	81
Tabela 5.12 – Tempos de execução das heurísticas construtivas com política de manutenção preventiva 3	82
Tabela 5.13 – Desvio relativo (em relação ao melhor) das heurísticas sem política de manutenção preventiva.....	83
Tabela 5.14 – Desvio relativo (em relação ao melhor) das heurísticas com política de manutenção preventiva 1.....	84

Tabela 5.15 – Desvio relativo (em relação ao melhor) das heurísticas com política de manutenção preventiva 2.....	84
Tabela 5.16 – Desvio relativo (em relação ao melhor) das heurísticas com política de manutenção preventiva 3.....	85
Tabela 6.1 – Desvio relativo (%) das heurísticas construtivas por cenário sem política de manutenção preventiva.....	89
Tabela 6.2 – Desvio relativo das heurísticas construtivas sem política de manutenção preventiva	89
Tabela 6.3 – Tempos de execução das heurísticas construtivas sem política de manutenção preventiva	91
Tabela 6.4 – Desvio relativo (%) das heurísticas construtivas por cenário com política de manutenção preventiva 1.....	92
Tabela 6.5 – Desvio relativo das heurísticas construtivas com política de manutenção preventiva 1	93
Tabela 6.6 – Tempos de execução das heurísticas construtivas com política de manutenção preventiva 1	94
Tabela 6.7 – Desvio relativo (%) das heurísticas construtivas por cenário com política de manutenção preventiva 2.....	96
Tabela 6.8 – Desvio relativo das heurísticas construtivas com política de manutenção preventiva 2	96
Tabela 6.9 – Tempos de execução das heurísticas construtivas com política de manutenção preventiva 2	98
Tabela 6.10 – Desvio relativo (%) das heurísticas construtivas por cenário com política de manutenção preventiva 3.....	99
Tabela 6.11 – Desvio relativo das heurísticas construtivas com política de manutenção preventiva 3	99
Tabela 6.12 – Tempos de execução das heurísticas construtivas com política de manutenção preventiva 3	101
Tabela 6.13 – Desvio relativo (em relação ao melhor) da heurística e metaheurísticas sem política de manutenção preventiva	102
Tabela 6.14 – Desvio relativo (em relação ao melhor) da heurística e metaheurísticas com política de manutenção preventiva 1.....	102
Tabela 6.15 – Desvio relativo (em relação ao melhor) da heurística e metaheurísticas com política de manutenção preventiva 2.....	103

Tabela 6.16 – Desvio relativo (em relação ao melhor) da heurística e metaheurísticas com política de manutenção preventiva 3.....	103
---	-----

LISTA DE SIGLAS, PARÂMETROS E VARIÁVEIS

depend: tarefas dependentes

$dfolga_{zj}$ = distância entre a data de término da tarefa z e data de início da tarefa j

d_i = data de entrega da tarefa j

DI_{ij} = data de início da tarefa j na máquina i

DT_{iz} = data de término da tarefa z na máquina i

EDD = *earliest due date*

E_j = adiantamento (*earliness*) da tarefa j

eq: tempo de processamento igual para todas as tarefas

F = *flowshop*

$F, perm$ = *flowshop* permutacional

F_j = tempo de fluxo (*flow time*) da tarefa j

J = *jobshop*

k_i = k máquinas i idênticas

k -paralelo = k máquinas em paralelo

L_j = pontualidade (*lateness*) da tarefa j

LPT = *longest processing time*

m = máquinas $\{M_1, M_2, \dots, M_m\}$

Makespan (*total completion time* máximo): $C_j = C_{max}$

MTTF = tempo médio para falha (*mean time to failure*)

MTTR = tempo médio para o conserto (*mean time to repair*)

n = tarefas $\{J_1, J_2, \dots, J_n\}$

No-wait = sem espera entre as operações de uma mesma tarefa

O = *openshop*

O_{ij} = operação na máquina i da tarefa j

prec: restrições de precedência

prmt: preempção é permitida

r_j = data de liberação da tarefa j

setup = tempo de *setup* dependente da sequência

SPT = *shortest processing time*

str: sequência das tarefas

TFT = tempo total de fluxo (*total flow time*)

t_{ij} = tempo de processamento na máquina i da tarefa j

T_j = atraso (*tardiness*) da tarefa j

W_j = tempo de espera (*waiting time*) da tarefa j

β = forma

θ = escala

$\parallel$ = máquina única

SUMÁRIO

1. INTRODUÇÃO	25
1.1 PROBLEMAS DE PROGRAMAÇÃO DA PRODUÇÃO	25
1.2 CLASSIFICAÇÃO DE MÉTODOS	29
1.2.1 MÉTODOS EXATOS VS. MÉTODOS HEURÍSTICOS	30
1.2.2 MÉTODOS HEURÍSTICOS: CONSTRUTIVOS VS. MELHORATIVOS	30
1.2.3 MÉTODOS CONSTRUTIVOS: SIMPLES VS. COMPOSTOS	31
1.3 MOTIVAÇÃO E JUSTIFICATIVA DA PESQUISA	31
1.4 OBJETIVO DA PESQUISA	33
2. MÉTODO DE PESQUISA	34
2.1 ETAPAS DE PESQUISA	34
2.2 MÉTODO PARA A SELEÇÃO DO TEMA DA DISSERTAÇÃO	34
2.3 MÉTODO PARA A REVISÃO BIBLIOGRÁFICA	35
3. REVISÃO DA LITERATURA	41
3.1 CONSIDERAÇÕES INICIAIS PARA A REVISÃO BIBLIOGRÁFICA	41
3.2 <i>NO-WAIT FLOWSHOP</i>	43
3.3 MANUTENÇÃO PREVENTIVA	52
3.4 <i>NO-WAIT FLOWSHOP</i> COM TEMPO DE MANUTENÇÃO PREVENTIVA	54
4. PROPOSTA DE MÉTODOS HEURÍSTICOS CONSTRUTIVOS	57
4.1 HEURÍSTICA CONSTRUTIVA PARA COMPARAÇÃO	57
4.2 HEURÍSTICA CONSTRUTIVA Ij + NEH	59
4.3 HEURÍSTICA CONSTRUTIVA MDD + NEH	60
4.4 HEURÍSTICA CONSTRUTIVA EDD + NEH + NSS1	61
4.5 HEURÍSTICA CONSTRUTIVA EDD+ NEH + LS1 + LS2	63
5. EXPERIMENTOS COMPUTACIONAIS	67
5.1 RESULTADO DAS HEURÍSTICAS CONSTRUTIVAS SEM MANUTENÇÃO PREVENTIVA	69
5.2 RESULTADO DAS HEURÍSTICAS CONSTRUTIVAS COM POLÍTICA MANUTENÇÃO PREVENTIVA 1	73
5.3 RESULTADO DAS HEURÍSTICAS CONSTRUTIVAS COM POLÍTICA MANUTENÇÃO PREVENTIVA 2	76
5.4 RESULTADO DAS HEURÍSTICAS CONSTRUTIVAS COM POLÍTICA MANUTENÇÃO PREVENTIVA 3	80
5.5 COMPARAÇÃO DAS HEURÍSTICAS CONSTRUTIVAS	83

6. PROPOSTA DE METAHEURÍSTICA.....	86
6.1 RESULTADO DA METAHEURÍSTICA SEM POLÍTICA DE MANUTENÇÃO PREVENTIVA	88
6.2 RESULTADO DA METAHEURÍSTICA COM POLÍTICA MANUTENÇÃO PREVENTIVA 1	92
6.3 RESULTADO DA METAHEURÍSTICA COM POLÍTICA MANUTENÇÃO PREVENTIVA 2	95
6.4 RESULTADO DA METAHEURÍSTICA COM POLÍTICA MANUTENÇÃO PREVENTIVA 3	99
6.5 COMPARAÇÃO DAS HEURÍSTICAS CONSTRUTIVAS E METAHEURÍSTICA.....	101
7. CONCLUSÃO.....	104
REFERÊNCIAS BIBLIOGRÁFICAS	106

1. INTRODUÇÃO

1.1 PROBLEMAS DE PROGRAMAÇÃO DA PRODUÇÃO

A programação da produção (*scheduling*) pode ser definida como a alocação de recursos para executar algumas tarefas ao longo de um tempo determinado. Esse conceito tem recebido muita atenção de praticantes de pesquisa operacional, cientistas de gestão, profissionais responsáveis pela produção e matemáticos desde o início da década de 1950 (MACCARTHY; LIU, 1993).

A atividade de *scheduling* da produção é uma das mais importantes para o sistema de manufatura, uma vez que é responsável por programar as tarefas nas máquinas e especificar o tempo que cada uma deve ser realizada (RUIZ; DÍAZ; MATORO, 2007).

MacCarthy e Liu (1993), enunciaram um problema geral de *scheduling* da seguinte maneira: n tarefas $\{J_1, J_2, \dots, J_n\}$ têm de ser processadas em m máquinas $\{M_1, M_2, \dots, M_m\}$ disponíveis. Um subconjunto destas máquinas é necessário para completar o processamento de cada tarefa. O fluxo padrão (ordem das máquinas) pode ou não ser fixo para todas as tarefas. O processamento da tarefa J_j na máquina M_i é chamado operação, denotado por O_{ij} . Para cada operação O_{ij} há um tempo t_{ij} associado ao processamento. Além disso, pode haver uma data r_j de liberação associado a cada tarefa, em que J_j estará disponível para o processamento. Há também a variável de data de entrega d_i , em que J_j deve estar finalizada. Um *scheduling* neste contexto é uma atribuição de tarefas ao longo do tempo em máquinas. O problema de *scheduling* é encontrar uma programação que otimiza alguma medida de desempenho (MACCARTHY; LIU, 1993).

Nesse contexto, MacCarthy e Liu (1993), também enunciaram as principais premissas encontradas na literatura para a teoria de *scheduling*:

- 1) As máquinas estão sempre disponíveis e nunca quebram.
- 2) Cada máquina pode processar no máximo uma tarefa a qualquer momento.
- 3) Qualquer tarefa pode ser processado em no máximo uma máquina de cada vez.
- 4) Data de liberação r_j para todas as tarefas é zero, ou seja, todas as tarefas estão disponíveis para iniciar o processamento no tempo zero.
- 5) Preempção (*preemption*) não é permitida, uma vez que uma operação é iniciada, continua-se até estar completa.

- 6) Tempos de *setup* são independentes dos *schedulings* e estão incluídos nos tempos de processamento.
- 7) Tempos de processamento e as restrições tecnológicas são determinísticas e conhecidas, e da mesma forma para as datas de entrega, se for o caso.

Para completar a definição de um problema de *scheduling*, restrições tecnológicas das tarefas e os objetivos de *scheduling* devem ser especificados. Em relação às restrições tecnológicas, o tipo de ambiente em que são processadas as tarefas influenciam na definição das restrições (MACCARTHY; LIU, 1993). Assim, seguem os tipos de ambientes que podem ser caracterizados, conceituados por MacCarthy e Liu (1993). A Figura 1.1 ilustra esquematicamente essa relação, sendo m = máquinas, $k_i = k$ máquinas i idênticas.

Figura 1.1 – Relação entre máquinas e ambientes (MACCARTHY; LIU, 1993)

- 1) *Jobshop*: cada tarefa tem seu próprio padrão de fluxo individual ou rota específica através das máquinas e que devem ser respeitadas.
- 2) *Flowshop*: cada tarefa tem um padrão de fluxo idêntico em um conjunto de máquinas.
- 3) *Openshop*: não há nenhum padrão de fluxo especificado para todas as tarefas.
- 4) *Flowshop Permutacional*: *flowshop* em que a ordem de processamento das tarefas nas máquinas é restrita a ser a mesma em todas elas.

- 5) Máquina Única: apenas uma máquina está disponível.
- 6) Máquinas Paralelas: k máquinas idênticas em um estágio de processamento único. Cada tarefa precisa de uma e somente uma dessas máquinas.
- 7) *Jobshop* com Máquinas Duplicadas: um *jobshop* em que há k_i máquinas idênticas em cada estágio ($i = 1, \dots, m$), e as tarefas requeridas precisam ser processados neste estágio em apenas uma dessas máquinas.

Diversos objetivos podem ser aplicados nesses ambientes de *scheduling*. Baker (1974), definiu *Completion time* como data de término de cada tarefa (C_i), no qual o *makespan* é o máximo de todos os C_i (C_{max}), Tempo de fluxo, Tempo de espera, *Lateness* e *Tardiness* como sendo esses objetivos de desempenho.

Makespan (total completion time máximo): $C_j = C_{max}$

Tempo de fluxo (*flow time*): $F_j = C_j - r_j$

Tempo de espera (*waiting time*): $W_j = C_j - r_j - \sum_{i=1}^m t_{ij}$

Lateness: $L_j = C_j - d_j$

Tardiness: $T_j = \max \{0, L_j\}$

Sendo, C = data de término da tarefa, r = data de liberação, t = tempo de processamento, d = data de entrega, j = tarefa e i = máquina.

Exemplos práticos de objetivos foram apresentados por Ackoff e Sasieni (1975), no contexto de empresas:

- Área de manufatura: maximizar o volume de bens produzidos ou minimizar o custo unitário da produção.
- Área de marketing: maximizar o volume vendido ou minimizar o custo unitário de vendas.
- Área de finanças: minimizar o capital necessário para sustentar o nível de serviço estabelecido.
- Área de RH: manter alta a produtividade dos colaboradores.

Uma das formas mais conhecidas de se estruturar um problema de *scheduling* foi definida por Conway, Maxwell e Miller (1967). A estrutura proposta por esses pesquisadores é baseado em 4 descritores A/B/C/D. O termo A se refere ao número de tarefas, geralmente caracterizada por n . Sendo que essa variável deve ser qualquer número inteiro maior que zero.

O termo B é similar ao A, porém ao invés de ser número de tarefas é o número de máquinas, cuja representação é m . Também deve-se considerar que m é um número inteiro positivo. A letra C representa o fluxo padrão e restrições de gerenciamento. Os valores mais conhecidos para esse termo C são (CONWAY; MAXWELL; MILLER, 1967):

||: Máquina Única

J: Jobshop

F: Flowshop

O: Openshop

F,perm: Flowshop permutacional

k-paralelo: k máquinas em paralelo

J, k-paralelo: Jobshop com k máquinas em paralelo em cada fase

Para caracterizar ambientes mais complexos de *scheduling*, pode-se adicionar as seguintes abreviaturas que representam restrições adicionais (CONWAY; MAXWELL; MILLER, 1967):

r_j: Tarefas possuem datas de liberação

str: Sequência das tarefas

prec: Restrições de precedência

prmt: Preempção é permitida

eq: Tempo de processamento igual para todas as tarefas

depend: Tarefas dependentes

setup = Tempo de *setup* dependente da sequência

Por fim, o carácter D representa o critério a ser otimizado. Sendo que as medidas de desempenho mais comuns são as de minimização do C_{max} (*maskepan*), F (tempo de fluxo) ou W (tempo de espera) (CONWAY; MAXWELL; MILLER, 1967).

Como exemplo de um problema básico aplicado na notação, tem-se o $n/m/J/C_{max}$, que se refere a um problema de *scheduling* em *jobshop* com n tarefas e m máquinas que tem por objetivo minimizar o *makespan*. Em relação a problemas classificados como não-básicos, existem aqueles onde alguns dos pressupostos não são válidos e/ou condições extras se aplicam, por exemplo, $r_j > 0$ para algumas tarefas j . Ao apresentar estes problemas, o descritor C fica representados por muitos termos, o que pode dificultar a interpretação e a síntese da notação (MACCARTHY; LIU, 1993).

Graham *et al.* (1979) apresentaram outra notação com base em três descritores $\alpha/\beta/\gamma$.

- O primeiro descritor α define o padrão de fluxo juntamente com o número de máquinas.
- O segundo descritor β representa outras restrições da tarefa.
- O terceiro descritor γ define o critério de *scheduling*.

Embora esta última notação possa facilitar a representação de problemas não-básicos, é comum encontrar pesquisadores de *scheduling* com a preferência para utilizar a notação definida por Conway, Maxwell e Miller em 1967 (MACCARTHY; LIU, 1993).

1.2 CLASSIFICAÇÃO DE MÉTODOS

Em ambientes de *flowshop* existem duas vertentes clássicas de métodos. Uma delas são os métodos exatos. Esses são ramificados em programação linear, *branch and bound*, geração de colunas, entre outros. A outra maneira de resolver os problemas de *flowshop* é através de métodos heurísticos, sendo que esses métodos podem ser classificados em melhorativos ou construtivos. Exemplos de métodos melhorativos são o algoritmo genético, busca tabu, PSO (*swarm optimization algorithm*), SA (*simulated annealing*), entre outros. Em relação aos métodos construtivos pode-se elencar três subdivisões, que são o método de Johnson, método simples e método composto, sendo esse último classificado em três fase (FRAMINAN; GUPTA; LEISTEN, 2004):

Fase 1: Desenvolvimento de índice;

Fase 2: Solução construtiva;

Fase 3: Solução melhorativa.

A Figura 1.2 a seguir ilustra o relacionamento e hierarquia entre os métodos.

Figura 1.2 – Classificação dos métodos

1.2.1 MÉTODOS EXATOS VS. MÉTODOS HEURÍSTICOS

Os métodos exatos são também conhecidos como métodos de solução ótima, pois retornam a solução ótima para o problema. Entretanto, o tempo para alcançar essa solução, costuma ser uma função polinomial, logo, problemas de grandes complexidades se tornam inviáveis para serem resolvidos, mesmo que computacionalmente.

Os métodos heurísticos podem ou não retornarem a solução ótima. Contudo, pode-se assumir que a solução que eles retornarão estará próxima da ótima. Muitos métodos heurísticos têm sido adotados e desenvolvidos por muitos pesquisadores, pois atingem uma solução viável (próxima da ótima) e com tempo computacional aceitável.

1.2.2 MÉTODOS HEURÍSTICOS: CONSTRUTIVOS VS. MELHORATIVOS

Em relação aos métodos melhorativos, procura-se identificar um melhor arranjo das tarefas em função de uma solução inicial, sendo que essa solução inicial utiliza um procedimento iterativo.

Em contrapartida, os métodos construtivos são caracterizados por obter a solução através da ordenação das tarefas por meio de índices de prioridade, ou a melhor sequência é escolhida a partir de sequências previamente obtidas, ou ainda através da geração de subsequências até que seja formada a sequência completa.

1.2.3 MÉTODOS CONSTRUTIVOS: SIMPLES VS. COMPOSTOS

No método composto a heurística é composta por uma fase de indexização, uma fase construtiva e uma fase melhorativa (FRAMINAN; LEISTEN; RUIZ-USANO, 2005). Dessa forma, três fases caracterizam os métodos compostos:

Fase 1 – Desenvolvimento de índice (*index development*): segundo regras de prioridade deve-se encontrar uma ordenação para as tarefas (FRAMINAN; GUPTA; LEISTEN, 2004).

Fase 2 – Solução construtiva (*solution construction*): a construção da solução é através na inserção de tarefas em uma sequência parcial, que pode ser a que foi obtida na fase 1. Deve-se levar em consideração o critério a ser otimizado (FRAMINAN; GUPTA; LEISTEN, 2004).

Fase 3 – Solução melhorativa (*solution improvement*): ao obter a sequência na fase 2, tenta-se melhorá-la através da mudança na ordenação das tarefas, que pode ser através de procedimentos de buscas locais ou metaheurísticas (FRAMINAN; GUPTA; LEISTEN, 2004).

Já o que caracteriza uma heurística ser de método simples é o fato de não haver nenhuma outra heurística nas fases de desenvolvimento de índice e de solução construtiva, além de não ocorrer a fase de melhoria da solução (FRAMINAN; LEISTEN; RUIZ-USANO, 2005).

1.3 MOTIVAÇÃO E JUSTIFICATIVA DA PESQUISA

Em nível operacional a programação da produção tem por objetivo retornar o sequenciamento com que as tarefas devem ser processadas, assim como mencionar o instante em que cada tarefa deve iniciar e terminar em cada máquina. Existe um grande interesse sobre esse tema devido à grande variedade de medidas de desempenho, ambientes de produção e características dos produtos, o que resulta em uma extensa literatura publicada (MORABITO *et al.*, 2015).

Para uma organização se manter competitiva no mercado é necessário, que investimentos sejam feitos na operação. Identificar a melhor forma de sequenciar as tarefas, bem como o melhor momento para realizar a manutenção preventiva, pode ser o diferencial para manter e prosperar no mercado. Dessa forma, essa pesquisa tem por objetivo auxiliar os profissionais responsáveis pela programação das tarefas. Para essa pesquisa foi realizado um recorte no contexto da manufatura aplicada ao ambiente de *no-wait flowshop*, no qual as tarefas são processadas continuamente e sem que haja interrupções entre uma operação e outra de uma mesma tarefa, e considerou-se paradas para manutenção preventiva, além do critério de minimização do *total tardiness*.

Os pesquisadores Arabameri e Salmasi (2013) reforçam a aplicabilidade de problemas de *no-wait* na indústria mecânica, química e em refinarias, nos quais a antecipação (*earliness*) e atrasos (*tardiness*) dos produtos podem inferir em penalidades para as indústrias. Além das aplicações já citadas para *scheduling no-wait*, Allahverdi (2016) menciona aplicações em outros contextos, tais como: programação de cirurgias, pouso de aeronaves e programação de trens.

Macchiaroli et al. (1999) ressalva que o *no-wait* é importante para a manufatura devido às restrições tecnológicas que os processos fabris exigem. E além disso, pode auxiliar a fábrica ser mais enxuta, uma vez que reduz o estoque entre processos (WIP – *work in process*).

Recentemente o interesse de adotar o critério de otimização *total tardiness* em problemas de *no-wait flowshop* aumentou, devido à sua importância para o sistema de produção e *supply chain*, uma vez que o atraso está diretamente relacionado com pagamento de multas e perda de clientes (DING et al, 2015). Liu, Song e Wu (2013), também mencionaram que o critério de otimização *total tardiness* ainda não foi muito explorado, apesar da sua importância e aplicabilidade. Por isso, a adaptação de algoritmos desenvolvidos e publicados para o critério de desempenho de minimização do *total tardiness*, parece ser uma área promissora para futura pesquisas (LAHA; GRUPTA; SAPKAL, 2014).

A maioria dos modelos propostos consideram que as máquinas sempre estão disponíveis, mas, na prática, fatores determinísticos ou aleatórios afetam a disponibilidades das máquinas, como por exemplo pode-se citar a manutenção preventiva como sendo um desses fatores (PINEDO, 2008).

A manutenção preventiva diminui as chances de que o sistema produtivo falhe inesperadamente, uma vez que atividades são programadas para que as máquinas possuam um

certo nível de confiabilidade. Isso é um fator crucial de sucesso para as indústrias (RUIZ; ALLAHVERDI, 2007).

Com mencionado por vários pesquisadores, está aumentando o interesse por pesquisas que adotem o critério de otimização *total tardiness*, muito em função do impacto financeiro e de imagem, que atrasos podem causar nas corporações. Além disso, muitos dos métodos propostos são em condições de total disponibilidades de máquinas, e isso não condiz com o que se encontrara na vida real, uma vez que as máquinas falham ou ficam indisponíveis para manutenções. Foram poucas as pesquisas que realizaram o estudo de *scheduling* de *no-wait flowshop* com manutenção preventiva. E a busca se torna ainda mais difícil, quando o critério de otimização é o *total tardiness*.

1.4 OBJETIVO DA PESQUISA

Essa pesquisa tem por objetivo desenvolver métodos heurísticos para problemas de programação da produção com o perfil de ambiente *no-wait flowshop*, minimização do *total tardiness* e com restrição de tempo de manutenção preventiva.

O método proposto poderá ser aplicado nas indústrias de manufaturas que possuem produtos nos quais o processamento não pode ser interrompido.

Também tem-se por objetivo realizar uma revisão bibliográfica sobre o recorte da pesquisa mencionada, de forma a ser uma fonte de consulta para que outros pesquisadores possam desenvolver estudos sobre o tema.

Como objetivo subsequente tem-se o desenvolvimento de uma metaheurística baseada no algoritmo construtivo de menor tempo de execução. Dessa forma, com experimento computacionais é possível comparar os métodos, e estabelecer análises e discussões.

Essa dissertação está estruturada em sete capítulos, sendo o primeiro a introdução, o segundo capítulo apresentada o detalhamento do método de pesquisa, o terceiro aborda a revisão da literatura, o quarto descreve os métodos heurísticos construtivos propostos, o quinto trata dos experimentos computacionais, o sexto inclui a metaheurística proposta, e o sétimo conclui a pesquisa.

2. MÉTODO DE PESQUISA

2.1 ETAPAS DE PESQUISA

Algumas etapas foram estruturadas para o desenvolvimento da dissertação, de modo que os resultados fossem decorrentes de uma base teórica rica em conceitos.

As fases macros foram:

Fase 1: Definição do tema da dissertação

Fase 2: Aprofundamento sobre o tema selecionado

Fase 3: Desenvolvimento de modelos matemáticos

Essa pesquisa não possui caráter de aplicação prática, ou seja, não é objetivo o desenvolvimento de um estudo de caso, ou mesmo a criação de um produto para o mercado. Entretanto, por ser uma modelagem de natureza inovadora, espera-se que possa contribuir e inspirar profissionais do mercado de otimização de operações nos seus desafios reais de gestão.

2.2 MÉTODO PARA A SELEÇÃO DO TEMA DA DISSERTAÇÃO

No universo da Gestão de Operações existem vários ramos de pesquisa, todos eles com o objetivo de auxiliar a tomada de decisão, seja ela relacionada à qualidade, processo, capacidade, estoque, força de trabalho, custo ou a combinação entre esses e outros objetivos.

O tema selecionado para essa pesquisa foi um recorte do assunto que engloba programação da produção em *flowshop*. Por ser um problema de programação da produção, logo remete-se às tomadas de decisões que os líderes de operações executam. Para que a modelagem heurística fosse mais aplicável no dia a dia dos profissionais de PCP (planejamento e controle da produção), considerou-se restrições de *no-wait* e manutenção preventiva.

A inspiração e motivação para a seleção desse tema foi principalmente a falta de discussão no meio acadêmico sobre esse assunto específico. E por englobar várias restrições, o problema se torna mais complexo de se resolver. Mas, por outro lado, aumenta sua aplicabilidade do mundo real.

Os detalhes de como o tema foi selecionado e sua importância para o mercado, estão descritos ao longo do Capítulo 1 dessa dissertação.

2.3 MÉTODO PARA A REVISÃO BIBLIOGRÁFICA

A base teórica dessa pesquisa é resultado da busca em livros, bem como acesso às bases de dados bibliográficas. Devido à grande extensão de artigos disponíveis nas bases de dados, aplicou-se um método para auxiliar a seleção de artigos para compor a revisão bibliográfica. Para isso, utilizou-se como referência métodos apresentados pelos autores Conforto, Amaral e Silva (2011) e Patil e Kant (2014).

Com o objetivo de se criar um roteiro para o método científico de busca e análise de artigos, os pesquisadores Conforto, Amaral e Silva (2011) descrevem o método de revisão bibliográfica sistemática aplicada para o contexto de “Desenvolvimento de Produtos & Gerenciamento de Projetos”. A vantagem imediata do método está associada ao aumento da confiabilidade da pesquisa, além de deixar como legado um estudo sistêmico que outros pesquisadores podem reutilizar. O roteiro intitulado de *RBS Roadmap* é caracterizado por ter repetições de ciclos contínuos, e é composto por três grandes etapas: Entrada, Processamento e Saída (Figura 2.1). Cada uma dessas etapas é detalhada a seguir:

Figura 2.1 – Modelo para condução da revisão bibliográfica sistemática (CONFORTO; AMARAL; SILVA, 2011)

1. Entrada: Definição do problema, definição dos objetivos da revisão sistemática (que devem estar alinhados com os objetivos da pesquisa), definição das fontes primárias para definição das palavras-chaves, autores e artigos importantes, definição da *string* de busca,

definição dos critérios de inclusão de artigos, definição de critérios de qualificação de artigos, definição de métodos e ferramentas de buscas cíclicas e definição de cronograma.

2. Processamento (Figura 2.2): Busca de artigos (busca de periódicos, busca de bases de dados e busca cruzada, que é a identificação de artigos relevantes através de citações), análise dos artigos (filtro 1 [leitura do título, resumo e palavras-chave], filtro 2 [leitura da introdução e conclusão] e filtro 3 [leitura completa]) e documentação (documentação das três etapas de filtros e após a seleção os artigos são catalogados).

3. Saída: Criação de alertas para novos artigos publicados, cadastro de artigos selecionados em um *software* específico para gerenciamento de referências, elaboração de um relatório que sintetiza a bibliografia e construção de um modelo teórico embasado nos resultados obtidos na RBS.

Figura 2.2 – Procedimento iterativo da fase de processamento (CONFORTO; AMARAL; SILVA, 2011)

Uma outra forma de simplificar e aumentar a eficiência na seleção de artigos também foi sugerida por Patil e Kant (2014). Os autores possuíam como motivação o tema de Gestão do Conhecimento, entretanto a metodologia utilizada pode ser aplicada a outros propósitos. A primeira etapa foi identificar todos os artigos que continham “Gestão do Conhecimento” no título. Nesse primeiro filtro, 1.654 artigos de 121 *journals* foram encontrados. O segundo passo constituiu em uma seleção mais rígida, na qual apenas artigos de *journals* com prestígio acadêmico seriam considerados na pesquisa. Como critério, o *journal* deveria possuir pontuação mínima de SJR de 100, além de índice H. Com isso, restaram 7 *journals*, que

somados totalizavam 344 artigos. Esses últimos artigos foram explorados e classificados em 5 categorias, como se segue:

- 1) Design de pesquisa aplicada: Quantitativo empírico, qualitativo empírico, conceitual quantitativo, conceitual qualitativo ou triangulação empírico.
- 2) Teste de hipótese: Binário (aplica ou não aplica).
- 3) Método de pesquisa: *Survey*, entrevistas, modelo matemático, estudo de caso, modelo conceitual ou outros.
- 4) Técnicas de análise de dados: Estatística descritiva, análise fatorial, regressão, análise de caminho, modelagem de equações estruturais, MCDM, desenvolvimento teórico e/ou outros.
- 5) Nível de análise: Propósito, uma única organização, par de organizações, cadeia de organizações, setor de organizações ou outros.

Apesar dos autores Conforto, Amaral e Silva (2011) e Patil e Kant (2014) terem desenvolvido um método para a seleção de artigos de “Desenvolvimento de Produtos & Gerenciamento de Projetos” e “Gestão do Conhecimento”, respectivamente, é possível aplicar o mesmo racional para outros temas de pesquisa. Uma vez que, é necessário apenas ajustes nos parâmetros.

Inspirados nos métodos para a seleção de artigos apresentados, foi criado o seguinte roteiro para essa dissertação:

- Etapa 1: Identificar as palavras-chaves, que devem estar alinhadas com os objetivos de pesquisa;
- Etapa 2: Selecionar bases de dados, que serão a principal fonte de artigos;
- Etapa 3: Elaborar filtros hierárquicos de buscas;
- Etapa 4: Fazer a leitura de título e palavras chaves dos artigos filtrados na etapa 3;
- Etapa 5: Fazer a leitura do resumo e conclusão dos artigos pré-selecionados na etapa 4;
- Etapa 6: Fazer a leitura dos artigos selecionados na etapa 5;
- Etapa 7: Documentar o procedimento realizado;
- Etapa 8: Avaliar se é necessário adicionar ou excluir alguma palavra-chave dos filtros, e retornar para a etapa 3;
- Etapa 9: Criar alertar para que novos artigos possam ser avaliados;
- Etapa 10: Incluir na listagem de artigos selecionados os artigos indicados por outros pesquisadores do Laboratório de Pesquisa Operacional Aplicada de pesquisa da EESP/USP.

Como resultado da aplicação das palavras chaves “*flowshop*”, “*no-wait*” e “*tardiness*” nos filtros das bases de dados, obteve-se o retorno de 33 artigos, sendo esses classificados como elegíveis ou inelegíveis para serem referências para a proposição dos algoritmos ou como referência para a revisão da literatura contida no capítulo 3.

Tabela 2.1 – Artigos pré-selecionados como referência (continua)

Elegibilidade	Representação	Função objetivo	Método	Referência
Inelegível	F <i>no-wait</i> ΣT	<i>Total tardiness</i>	Exato (<i>branch-and-bound</i>)	Ronconi e Armentano (2001)
Inelegível	F <i>no-wait</i> $C_{\max}$	<i>Makespan</i>	Heurística construtiva (MM, MME e PFE)	Ronconi (2004)
Inelegível	F <i>no-wait</i> $C_{\max} + L_{\max}$	Bicritério (<i>makespan</i> e máximo <i>lateness</i>)	Heurística melhorativa (SA, <i>simulated annealing</i> híbrido e algoritmo genético híbrido)	Allahverdi e Aldowaisan (2004)
Inelegível	F <i>no-wait</i> $\Sigma C + \Sigma T$	Bicritério (tempo total de fluxo e <i>total tardiness</i>)	Heurística melhorativa (algoritmo imune)	Tavakkoli-Moghaddam, Rahimi-Vahed e Mirzaei (2007)
Elegível	<i>Survey</i>	<i>Survey</i>	<i>Survey</i>	Allahverdi et al. (2008)
Inelegível	F <i>no-wait</i> $\Sigma C + \Sigma T$	Bicritério (tempo total de fluxo e <i>total tardiness</i>)	Heurística melhorativa (algoritmo imune)	Tavakkoli-Moghaddam, Rahimi-Vahed e Mirzaei (2008)
Inelegível	F <i>no-wait</i> ΣT	<i>Total tardiness</i>	Heurística melhorativa (algoritmo genético)	Januario et al. (2008)
Inelegível	F <i>no-wait</i> $C_{\max} + T_{\max}$	Bicritério (<i>makespan</i> e o máximo <i>tardiness</i>)	Heurística melhorativa (DDE, evolução diferencial discreta)	Pan, Wang e Qian (2009)
Elegível	F <i>no-wait, block</i> ΣT	<i>Total tardiness</i>	Heurística construtiva (FPDNEH)	Ronconi e Henriques (2009)
Inelegível	F <i>no-wait</i> ΣT	<i>Total tardiness</i>	Heurística melhorativa (algoritmo genético)	Januario, Arroyo e Moreira (2009)
Inelegível	F W $C_{\max} + \Sigma B$	Bicritério (<i>makespan</i> e custo de entrega)	Heurística construtiva (H e MH)	Gong e Tang (2009)
Inelegível	F <i>no-wait</i> $L_{\max}$	Máximo <i>lateness</i>	Heurística melhorativa (algoritmo genético)	Wang, Li, Wang (2010)
Inelegível	F $C_{\max} + \Sigma T$	Bicritério (<i>makespan</i> e <i>total tardiness</i>)	Heurística melhorativa (RVNS, <i>refreshing VNS</i>)	Perez-Gonzalez e Framinan (2010)
Inelegível	F <i>no-idle</i> $\Sigma E + \Sigma T$	Bicritério (<i>total earliness</i> e <i>total tardiness</i> ponderados)	Heurística melhorativa (DABC, colônia de abelhas artificial discreta)	Pan et al. (2011a)
Inelegível	F $\Sigma E + \Sigma T$	Bicritério (<i>total earliness</i> e <i>total tardiness</i> ponderados)	Heurística melhorativa (DLHS, algoritmo de pesquisa de harmonia local (HS) com memórias de sub-harmonia dinâmica (HM))	Pan et al. (2011b)

Tabela 2.1 – Artigos pré-selecionados como referência (continuação)

Elegibilidade	Representação	Função objetivo	Método	Referência
Inelegível	F <i>no-wait</i> $C_{\max}$	<i>Makespan</i>	Heurística melhorativa (algoritmo genético híbrido)	Jarboui, Eddaly e Siarry (2011)
Inelegível	F <i>no-idle</i> ΣT	<i>Total tardiness</i>	Heurística melhorativa (DE, algoritmo de evolução diferencial)	Tasgetiren et al. (2011)
Inelegível	F <i>no-wait, setup</i> $C_{\max}$	<i>Makespan</i>	Heurística melhorativa (SA, <i>simulated annealing</i>)	Ying et al. (2012)
Elegível	F <i>no-wait</i> ΣT	<i>Total tardiness</i>	Heurística melhorativa (SA-IP)	Aldowaisan e Allahverdi (2012b)
Inelegível	F <i>no-wait</i> $C_{\max} + \Sigma T$	Bicritério (<i>makespan</i> e <i>total tardiness</i>)	Heurística melhorativa (GILS + MILS + CILS)	Naderi et al. (2012)
Inelegível	F <i>no-idle</i> ΣT	<i>Total tardiness</i>	Heurística melhorativa (DABC, colônia de abelhas artificial discreta)	Tasgetiren et al. (2013)
Inelegível	F <i>no-wait, setup</i> $\Sigma E + \Sigma T$	Bicritério (<i>total earliness</i> e <i>total tardiness</i> ponderados)	Heurística melhorativa (PSO, <i>particle swarm optimization</i>)	Arabameri e Salmasi (2013)
Elegível	F <i>no-wait</i> ΣT	<i>Total tardiness</i>	Heurística construtiva (MNEH, NEH modificado)	Liu, Song e Wu, (2013)
Inelegível	F <i>time lag</i> $\Sigma B + \Sigma T$	Bicritério (maximizar o lucro total e minimizar o desvio da data de entrega)	Heurística melhorativa (algoritmo genético)	Sheikh (2013)
Inelegível	F <i>no-wait</i> $\Sigma E + \Sigma T$	Bicritério (<i>total earliness</i> e <i>total tardiness</i> ponderados)	Heurística melhorativa (MIP)	Mao et al. (2014)
Inelegível	F <i>no-wait, setup</i> $C_{\max} + \Sigma T$	Bicritério (<i>makespan</i> e <i>total tardiness</i>)	Heurística melhorativa (NSGA-II + VNS)	Asefi et al. (2014)
Inelegível	F <i>no-wait</i> $C_{\max} + \Sigma T$	Bicritério (<i>makespan</i> e <i>total tardiness</i>)	Heurística melhorativa (MOEA, algoritmo multi-objetivo de eletromagnetismo)	Khalili (2014)
Inelegível	F <i>no-wait, paralela</i> $C_{\max}$	<i>Makespan</i>	Heurística melhorativa (HSA, algoritmo metaheurístico de busca de caça)	Naderi, Khalili e Khamseh (2014)
Elegível	F <i>no-wait</i> ΣT	<i>Total tardiness</i>	Heurística melhorativa (NEH acelerado com fatores aleatórios)	Ding et al. (2015)
Inelegível	F <i>no-wait</i> $C_{\max}$	<i>Makespan</i>	Exato (programação inteira mista, programação quadrática mista e programação com restrições)	Samarghandi e Behrooz (2017)
Inelegível	F <i>no-wait</i> $C_{\max} + \Sigma T$	Bicritério (<i>makespan</i> e <i>total tardiness</i>)	Heurística melhorativa (AA, <i>simulated annealing</i> com algoritmo de inserção)	Allahverdi, Aydilek, H. e Aydilek, A. (2018)
Inelegível	F <i>no-wait</i> $C_{\max} + \Sigma T$	Bicritério (<i>makespan</i> e <i>total tardiness</i>)	Heurística melhorativa (CTIG, guloso iterado baseado em teoria de nuvem)	Lin, Lu e Ying (2018)

Tabela 2.1 – Artigos pré-selecionados como referência (conclusão)

Elegibilidade	Representação	Função objetivo	Método	Referência
Inelegível	F <i>no-wait</i> ΣC	Tempo total de fluxo	Estudo de várias heurísticas construtivas	Miyata, Nagano e Gupta (2019)

Dos artigos pré-selecionados apenas 5 foram selecionados para serem estudados com maior profundidade. Os outros artigos não são elegíveis, pois não cumpriam os critérios de aceitação. Além dos artigos selecionados, outros foram utilizados para a composição da referência bibliográfica, dentre eles: Ruiz e Allahverdi (2006), Moccellini e Nagano (2007), Ruiz, Díaz e Matoro (2007), Laha e chakraborty (2009), Allahverdi (2012), Naderi e Arshadi (2013), Liu et al. (2013), Ribas, Companys e Tort-Martorell (2013), Arabameri e Salmasi (2014), Ding et al. (2015), Cheng, Zhou e Li (2017), entre outros.

Nota-se que nenhum artigo encontrado menciona a restrição de manutenção preventiva, o que torna a combinação de *flowshop*, *no-wait*, *total tardiness* e manutenção preventiva incomum.

3. REVISÃO DA LITERATURA

Esta revisão enfatiza os principais acontecimentos e tendências. Não tem por objetivo atingir o estado da arte.

3.1 CONSIDERAÇÕES INICIAIS PARA A REVISÃO BIBLIOGRÁFICA

Medidas de desempenho são os critérios que se deseja otimizar, seja para minimizar ou maximizar. As funções objetivo podem ser de várias naturezas, sua motivação pode ser financeira, operacional, estratégica ou mesmo de limitações físicas de uma fábrica.

As principais medidas de desempenho para avaliar a qualidade do programa desenvolvido para a produção na visão de Morabito et al. (2015) são: *makespan*, tempo total de fluxo, *tardiness* máximo, *total tardiness*, *lateness* e número de tarefas atrasadas.

Os atrasos são mais impactantes do que os adiantamentos, em muitos casos. Dessa forma, estudar a minimização da soma dos atrasos (*total tardiness*) de todas as tarefas torna-se objeto de interesse para algumas indústrias.

$$\text{Representação: } \min \sum_{j=1}^n T_j \quad \text{Eq. (3.1)}$$

Sendo, $T_j = \max \{C_j - d_j, 0\} = \text{tardiness (atraso) da tarefa } j$

Em um problema padrão de *scheduling* existem n tarefas para serem dispostas em m máquinas, o que gera $n! \cdot m!$ possibilidades de organizar as tarefas, no caso das n tarefas serem processadas por todas as m máquinas (MACCARTHY; LIU, 1993). A seguir, encontra-se um exemplo de disposição para 4 tarefas (j_1, j_2, j_3 e j_4) em 2 máquinas (M_1 e M_2) (Figura 3.1), que contenham os tempos de processamentos mostrados na Tabela 3.1. A sequência gerada no exemplo foi j_2, j_4, j_3 e j_1 .

	j_1	j_2	j_3	j_4
M1	1	2	3	2
M2	3	1	2	2

Tabela 3.1 – Tempos de processamentos (exemplos 1, 2 e 3)

Figura 3.1 – Gráfico de Gantt (exemplo 1)

Em algumas situações práticas não é viável que a máquina fique ociosa ou que haja estoque em processo. Razões financeiras e de propriedades dos materiais dos produtos reforçam esse cenário. Ou seja, existem indústrias que utilizam máquinas nas quais é muito caro deixá-las ligadas; por isso, otimizar suas operações se torna prioridade. Logo, reduzir o tempo ocioso é fundamental. Por outro lado, existem alguns processos industriais, nos quais o produto que está sendo processado não pode “ficar parado”. Por exemplo, nas indústrias de fundição, se o ferro esfriar, ele irá solidificar e isso inviabiliza todas as etapas subsequentes (ARABAMERI E SALMASI, 2013).

Dessa forma, uma maneira de evitar a primeira situação, é zerando todos os intervalos entre as tarefas de uma máquinas, como demonstra a ilustração abaixo (Figura 3.2), na qual corresponde a sequência j_2, j_4, j_3 e j_1 .

Figura 3.2 – Gráfico de Gantt para *no-idle* (exemplo 2)

Em relação ao problema mencionados na segunda situação, o ideal é sequenciar as tarefas de modo que não haja espera entre o processamento das mesmas nas máquinas. Portanto, trata-se de um problema de *no-wait*, como demonstrado na Figura 3.3.

Figura 3.3 – Gráfico de Gantt para *no-wait* (exemplo 3)

Os exemplos demonstram o impacto do *scheduling*, ou seja, um correto sequenciamento das tarefas pode gerar vantagem competitiva. Mas, para que o sequenciamento das tarefas não sofra impacto de possíveis falhas nas máquinas, é fundamental que um plano de manutenção preventiva seja implantado pelas indústrias, e considerado no *scheduling*. Fusco (2010) mencionou que empresas mais preparadas, que conseguem competir em ambientes de alta concorrência, são aquelas que investem em metodologias e técnicas que melhorem a gestão.

A Figura 3.4 ilustra como a manutenção preventiva poderia ser inserida no *scheduling*. Existem várias políticas para a implementação da manutenção preventiva. Dentre elas pode-se elencar as seguintes:

- Intervalos de tempos predefinidos (por exemplo, todo final de mês)
- Intervalos de tempos dependente da utilização da máquina (por exemplo, a cada ciclo de 300 horas de uso)
- Intervalos de tempos considerando o misto de utilização da máquina e ordens de produção, de forma a evitar que um pedido seja interrompido para efetuar uma manutenção preventiva.

Figura 3.4 – Gráfico de Gantt com manutenção preventiva

3.2 NO-WAIT FLOWSHOP

Problemas de *scheduling flowshop* são aqueles nos quais um conjunto de máquinas distintas processam n tarefas na mesma sequência, de modo a otimizar um critério de desempenho (MOCCELLIN; NAGANO, 2007). Nessa situação, as tarefas podem ter um tempo de espera para serem processadas na próxima máquina, assim como pode haver tempo inativo de máquina. Quando não é permitido que as tarefas sejam interrompidas entre uma máquina e outra, o problema é tratado como *no-wait*. Existem muitas aplicações para *no-wait*, tais como: produção de aço, indústria química e processamento de alimentos (LAHA; CHAKRABORTY, 2009). Outros exemplos de aplicação são as indústrias de plásticos, farmacêuticas e de metal aquecido (RUIZ; ALLAHVERDI, 2006).

Não haver ociosidade entre operações de uma mesma tarefa é denominado *no-wait* (RUIZ; ALLAHVERDI, 2006). Se necessário, o início do processamento de uma tarefa em uma máquina é atrasado para que a conclusão da mesma coincida com o início na próxima máquina, dessa forma é garantida a condição de *no-wait* (ALDOWAISAN; ALLAHVERDI, 2012a).

O estudo de soluções para problemas de *scheduling no-wait* é datado na década de 70. Desde então, vários métodos foram propostos (ALLAHVERDI, 2016). A seguir, são relatados alguns métodos disponíveis na literatura sobre o tema *no-wait flowshop*. Priorizaram-se os artigos que mencionavam algo sobre o critério de desempenho *tardiness*. Também foram organizados em ordem cronológica de publicação.

Reddi e Ramamoorthy (1972) estudaram o problema de *no-wait* em ambiente de *flowshop*, no qual n tarefas têm que ser processados sem interrupção em m máquinas ($F | \textit{no-wait} | \Sigma C$). Os autores consideraram o tempo de espera que uma tarefa deve ter para iniciar o processamento, de modo que, uma vez iniciada, seu processamento não seja interrompido. Também foram consideradas as restrições para que uma tarefa não sobreponha outra tarefa, já que uma máquina só pode processar uma tarefa de cada vez.

Deve-se considerar o motivo que originou o desejo de realizar uma otimização na produção, ou seja, nem sempre é desejável minimizar o tempo total de fluxo. Se as máquinas forem alugadas por custos diferentes, talvez, o desejável seja minimizar o tempo inativo das máquinas mais caras. Dessa forma, uma ponderação dos tempos ociosos pode ter utilizada na função objetivo (REDDI; RAMAMOORTHY, 1972).

Os pesquisadores, Aldowaisan e Allahverdi (1998), estudaram o problema de 2 máquinas em um contexto de *no-wait flowshop* com tempos de setups separados dos tempos de processamento. Com o objetivo de minimizar o tempo total de fluxo ($F | \textit{no-wait}, | \Sigma C$), uma heurística foi desenvolvida para encontrar soluções. Utilizou-se também um critério de eliminação. Basicamente, uma ordem de preferência é estabelecida, verifica-se se permutações podem ocorrer e calcula-se o tempo total de fluxo.

Com o objetivo de minimizar o *lateness* máximo em problemas de *no-wait flowshop* e com tempos de setups independentes dos tempos de processamento ($F | \textit{no-wait}, \textit{setup} | L_{\max}$), Ruiz e Allahverdi (2006) propuseram para o problema de 3 máquinas várias heurísticas e 4 novos algoritmos genéticos. Os métodos propostos apresentaram bom desempenho computacional, se comparado com outros métodos da literatura.

A fórmula do *lateness* máximo pode ser representada como segue na equação Eq.(3.2), no qual o *lateness* é a diferença entre a data de término e a data de entrega. Logo em seguida,

equação Eq.(3.3) representa o que se deseja minimizar, o *lateness* máximo (RUIZ; ALLAHVERDI, 2006)

$$T[j] = C[j] - d[j] \quad \text{Eq. (3.2)}$$

$$L[\text{máx}] = \max_{j=1, \dots, n} \{L[j]\} \quad \text{Eq. (3.3)}$$

Sendo, $L = \text{lateness}$, $C[j]$ = data de término da tarefa j e $d[j]$ = data de entrega da tarefa j .

O algoritmo proposto foi desenvolvido em duas etapas, sendo a primeira composta por métodos heurísticos e a segunda inclui algoritmo genético (RUIZ; ALLAHVERDI, 2006):

- Métodos heurísticos: Utilização de regras de despachos já existentes e criação de novas. Por exemplo, EDD - *Earliest Due Date* (tarefas ordenadas pelas datas de entrega de forma crescentes), EDDP - *Earliest Due Dates with Processing* (tarefas ordenadas de forma crescente segundo um índice), EDDSP - *Earliest Due Dates with Processing and Setups* (tarefas ordenadas de forma crescente segundo um índice com tempo de *setups*), SLACK (regra dinâmica), SLACKRW (SLACK com tempo total de processamento), MDD - *Modified Due Date* (regra dinâmica, no qual a próxima tarefa ordenada é aquele com a data de entrega mínima modificada) e heurística NEH modificada.

- Algoritmo genético: Esse método se inspira na genética humana, ou seja, os indivíduos sobrem mutações, e a população é avaliada. E um critério de parada deve ser estabelecido. Após feita a inicialização da população, é utilizado um processo de *crossover* para melhorar as soluções geradas. Uma hibridação com busca local é implementada para aumentar a velocidade de processamento. Por fim é realizada a calibração dos algoritmos.

Ruiz e Allahverdi (2007) propuseram uma regra de dominância efetiva para problemas de 4 máquinas, mas com possibilidade para ser expandida para problemas de m máquinas. Os autores limitaram-se em problemas de *flowshop*, que consideravam ao mesmo tempo o ambiente *no-wait* e tempo de *setups* separados. Além disso, o critério de otimização utilizado foi o de minimizar o tempo total de fluxo ($F \mid \text{no-wait, setup} \mid \Sigma C$).

As regras de dominâncias são composto por 5 heurísticas simples e 2 métodos de busca locais estocásticos. Essas regras são importantes, pois podem melhorar a qualidade das soluções, uma vez que reordenam a sequência das tarefas. As heurísticas determinísticas (heurísticas 1 até 5) e os métodos de buscas locais estocásticos (heurísticas 6 e 7), estão resumidos abaixo (RUIZ; ALLAHVERDI, 2007):

- Heurística 1: inicialização da sequência (remoção de algumas tarefas).
- Heurística 2: apenas uma chamada limitada de busca local sem reiniciar.

- Heurística 3: uma chamada de busca local sem reiniciar.
- Heurística 4: duas chamadas de busca local sem reiniciar.
- Heurística 5: doze chamadas de busca local sem reiniciar.
- Heurística 6: iterações de buscas locais completas até que o critério de parada seja atingido, há reinício a cada 10 iterações sem melhora na solução.
- Heurística 7: adaptação do método ILS (*iterated local procedure*).

Foi estudado por Laha e Chakraborty (2009) o problema de *no-wait flowshop* para minimizar o *makespan* ($F \mid \textit{no-wait} \mid C_{\max}$). Os autores desenvolveram uma heurística construtiva baseada no princípio de inserção de tarefas. Os autores se inspiraram na heurística NEH para elaborarem o método, na qual começa com uma sequência inicial de tarefas organizadas em ordem decrescente de tempo de processamento. As duas primeiras tarefas são selecionadas, e a melhor sequência parcial dessas duas tarefas é selecionada. As outras tarefas são então inseridas na sequência inicial, de forma a se ter uma melhor configuração. O método proposto pelos autores utiliza um mecanismo de vizinhança para geração da solução inicial, como é descrito abaixo:

- Ordenar os n tarefas em ordem decrescente em tempo de processamento.
- Gerar $2*(n - 1)$ sequências ao aplicar os métodos de selecionar tarefas para mudarem de lugar na sequência, algumas tarefas do início da sequência vão para o final e vice e versa. Selecionar a sequência de menor *makespan*.
- Definir $k = 1$, e selecionar as duas primeiras tarefas da sequência gerada na etapa anterior para compor a sequência atual.
- Definir $k = k + 1$, e gerar soluções parciais, ou seja, cada uma com $2*k$ tarefas, e inserir as duas próximas tarefas como um bloco na sequência. Selecionar a melhor sequência parcial. Em seguida, o primeiro bloco de duas tarefas é testado em todas as posições possíveis na sequência. Verifica-se se a solução melhora. O segundo bloco de duas tarefas também é testado em outras posições.
- O passo anterior é repetido até que todas as tarefas tenham sido programadas.

Aldowaisan e Allahverdi (2012b) estudaram as regras de despacho existentes para melhorar a solução inicial do método que propuseram, para o problema de *no-wait flowshop* com o objetivo de minimizar o *total tardiness* ($F \mid \textit{no-wait} \mid \Sigma T$). O método utilizado foi o *simulated annealing* (SA) e algoritmo genético melhorado (GA).

Como solução inicial foi utilizado o MDD (*modified due date*) para o *simulated annealing* (SA). E para o algoritmo genético considerou-se como solução inicial as regras MDD (*modified due date*), SMPT (*single machine processing time*) e EDDP (*earliest due date with processing time*) (ALDOWAISAN, ALLAHVERDI; 2012b).

Para melhorar os algoritmos SA e GA os pesquisadores aplicaram a heurística PAAH, na qual é constituída por três etapas. A primeira é obter uma sequência inicial, a outra é melhorar a essa sequência por meio de um procedimento de inserção, e por fim melhorar ainda mais a solução através de um procedimento de troca (ALDOWAISAN, ALLAHVERDI; 2012b).

Liu et al. (2013) propuseram algumas heurísticas para *no-wait flowshop* com o critério de otimização *tardiness* ($F | \text{no-wait} | \Sigma T$). O método que melhor apresentou resultado foi o NEH modificado. Os pesquisadores testaram algumas regras de despacho, pois elas utilizam índices simples para gerar uma solução inicial, como por exemplo o tempo de processamento ou data de entrega. Dessa forma, as regras SPT (*smallest processing time*) e EDD (*earliest due date*) foram consideradas nos estudos. Contudo, também foram consideradas heurísticas construtivas simples, que geram progressivamente a sequência:

- SLACK: Para as tarefas ainda não programadas, calcula-se a folga entre a data de entrega e a data de término da tarefa. A tarefa com menor folga é adiciona à sequência.

- SLACKRW (SLACK com *remaining work*): O processo é parecido com o SLACK, porém, considera-se o tempo total de processamento.

- MDD (*modified due date*): Adiciona-se à sequência a tarefa com a menor data de entrega modificada, ou seja, é o máximo entre a data de entrega e o tempo total de processamento.

Os autores então propuseram um algoritmo NEH modificado. Primeiro, gera-se a solução inicial com a regra EDD, que obteve o melhor desempenho para ser a solução inicial. Depois seleciona-se as duas primeiras tarefas e avalia-se o *total tardiness* das sequências parciais. A sequência com menor *total tardiness* é selecionada. As tarefas não programadas são inseridos e novamente seleciona-se a programação com o menor *total tardiness* (LIU et al., 2013).

Os pesquisadores Arabameri e Salmasi (2013) propuseram um modelo de programação linear inteira mista para o problema de *no wait flowshop* com tempos de *setups* dependentes e função objetivo de minimizar o *earliness* e *tardiness* ponderado ($F | \text{no-wait, setup} | wE + wT$). Para essa categoria de problema os pesquisadores estudaram os métodos de Busca Tabu (TS) e

Particle Swarm Optimization (PSO), sendo que o último apresentou melhores resultados que o primeiro.

Naderi e Arshadi (2013) estudaram o problema de *no-wait* em *flowshop* para minimizar o *total tardiness*. A solução proposta foi uma metaheurística composta por busca de vizinhança variável (VNS) de dois níveis, NSS1 e o NSS2. Com a sequência gerada no procedimento VNS aplicou-se o *simulated annealing* (SA).

A primeira etapa do VNS intitulada de NSS1 é responsável por remover aleatoriamente uma tarefa por vez, e realizar a inserção dessa tarefa em todas as posições da sequência, a melhor sequência parcial é aceita. A segunda etapa, NSS2, segue o mesmo procedimento da descrita para o NSS1, porém são removidos duas tarefas por vez.

A metaheurística Busca Tabu (TS) foi desenvolvida para resolver problemas de otimização combinatória, e sua vantagem é que não fica presa em ótimos locais. O algoritmo começa a partir de uma solução inicial (pode ser gerada por EDD – *earliest due date*, LTER – *longest tardiness/earliness rate* ou ainda por uma sequência randômica), e sua vizinhança é avaliada ao realizar perturbações. As possíveis soluções ficam armazenadas na chamada lista Tabu. Essa lista é avaliada de forma a evitar que o problema fique preso em ótimos locais (ARABAMERI; SALMASI, 2014).

O algoritmo *Particle Swarm Optimization* (PSO) é utilizado para resolver problemas com funções contínuas não lineares, e inicia com várias soluções iniciais denominadas partículas, sendo que cada partícula apresenta duas dimensões, posição e velocidade. O espaço de busca aplicado para o problema são as tarefas, assim as partículas se movem (em enxames) nesse espaço em busca dos melhores valores da função objetivo. Os algoritmos PSO podem se associar a outras metaheurísticas, como buscas locais, para melhorar o resultado (ARABAMERI; SALMASI, 2014).

Com uma técnica de melhorar a sequência inicial gerada, a partir de um método de inserção de tarefas, os autores Sapkal e Laha (2013) e Laha e Sapkal (2014) desenvolveram uma heurística construtiva para o problema de *scheduling* de *no-wait flowshop* com o critério de otimização do tempo total de fluxo ($F | \text{no-wait} | \Sigma C$). Para isso, se basearam no pressuposto que as tarefas são organizadas baseadas pela prioridade da soma dos tempos de processamento das máquinas gargalos.

Foram estudadas várias heurísticas até que os autores elaborassem o seguinte método (LAHA; SAPKAL, 2014):

- Passo 1: Considerar a máquina gargalo para organizar as tarefas em ordem crescente dos tempos de processamento.
- Passo 2: Considerar as máquinas adjacentes como gargalo, e fazer a ordenação das tarefas em ordem crescente dos tempos de processamento para as máquinas individualmente.
- Passo 3: Das sequências geradas nos passos 1 e 2, a melhor é selecionada como sendo a sequência inicial.
- Passo 4: Definir $k = 1$, selecionar as duas primeiras tarefas da sequência inicial, e selecionar a melhor das duas sequências parciais de duas tarefas, considerar como sequência atual.
- Passo 5: Definir $k = k + 1$, e gerar soluções parciais com $2*k$ tarefas (inserção de bloco de duas tarefas). Selecionar a melhor sequência parcial como atual. Em seguida, o primeiro par de tarefas é colocado em todas as posições possíveis da sequência atual, se a solução gerada for melhor, essa passa a configurar a sequência atual. O mesmo procedimento é efetuado para o segundo bloco de duas tarefas.
- Passo 6: Repetir o passo 5 até que todos as tarefas estejam programados.
- Passo 7: Se ao final houver apenas uma tarefa, insira essa última tarefa na sequência atual em todas as posições possíveis.
- Passo 8: Das soluções geradas considerar a que tiver menor tempo total de fluxo como solução atual.

Para resolver problemas de *flowshop* em condições de *no-wait* foi desenvolvido uma heurística construtiva para minimizar o tempo total de fluxo ($F \mid \text{no-wait} \mid \Sigma C$). Em uma primeira etapa foi elaborada uma heurística baseada em penalidades para gerar o sequenciamento inicial, na qual derivou do método de aproximação de Vogel. Na segunda etapa a solução é melhorada por meio da heurística de *forward shift heuristic*, na qual insere tarefas em posições relativas à frente da atual. E na terceira etapa, a solução sofre uma melhora através de heurísticas de inserção (par de tarefas ou tarefas únicas), a qual aceleram a procura de uma sequência melhor (LAHA; GRUPTA; SAPKAL, 2014).

Samarghandi (2015) estudou o problema de *scheduling* em ambiente de *no-wait flowshop* com o objetivo de minimizar o *makespan*, e restrição de data de entrega (*due date*) ($F \mid \text{no-wait}, d \mid C_{\max}$). O problema é classificado como sendo *NP-Hard*, uma vez que deriva do clássico *NP-Hard* $F \mid \text{no-wait} \mid C_{\max}$. Um algoritmo caracterizado por uma otimização por enxame de partículas (*particle swarm optimization* – PSO) foi proposto, no qual consiste em associar a

uma partícula i uma posição $xi(t)$ e uma velocidade $vi(t)$ no tempo t . Basicamente, o algoritmo move sistematicamente as partículas e armazena em memória a melhor posição da partícula i em relação a função objetivo. Assim como, armazena as posições que resultam no melhor resultado global. Para as soluções iniciais várias regras de despacho foram testadas, sendo a EDD a de melhor resultado. E a abordagem baseada em delineamento de experimentos de Taguchi (*Taguchi-based design of experience approach*) foi utilizada para calibração dos parâmetros do algoritmo, e assim permitir alcançar a solução final mais rapidamente.

Com o objetivo de investigar o problema de *no-wait flowshop* para minimizar *o total tardiness* com tempos de *setups* separados ($\mathbf{F} \mid \text{no-wait, setup} \mid \Sigma \mathbf{T}_i$), os autores Aldowaisan e Allahverdi (2015) estudaram algumas regras de despacho, sendo que três responderam de maneira melhor. Além disso, propuseram melhorias nas heurísticas de *simulated annealing* (SA) e algoritmo genético (GA).

Foi utilizado a regra de despacho MDD para melhorar a solução inicial da heurística de *recozimento simulado* (SA). E a intervenção no algoritmo genético (GA) foi através de três regras de despacho (MDD, EDDP e TOTA) que ajudaram na constituição da população inicial do GA. Além disso, um algoritmo de inserção foi proposto com o objetivo de desenvolver versões melhoradas para SA e GA (ALDOWAISAN; ALLAHVERDI, 2015).

Os autores também fizeram modificações em vários parâmetros de entrada das heurísticas de SA e GA. As mesmas foram necessárias para adaptar as heurísticas para incluam os tempos de *setups*.

A seguir encontram-se as regras de despacho utilizadas no algoritmo (ALDOWAISAN; ALLAHVERDI, 2015):

- Sequência de tarefas com a data de entrega antecipada com tempos de processamento e *setups* (EDDP), onde as tarefas são sequenciados em ordem não-decrescentes da relação entre data de entrega, tempo de processamento e tempo de *setup* (Eq.(3.4)):

$$d[j] / \sum_{i=1}^m (S_{ij} + P_{ij}) \quad \text{Eq. (3.4)}$$

Sendo, d_j = data de entrega da tarefa j , P_{ij} = tempo de processamento da tarefa j na máquina i e S_{ij} = tempo de *setup* da tarefa j na máquina i .

- Sequência de tarefas com a data de entrega modificada (MDD), onde as tarefas são sequenciados em ordem não decrescente de máximo entre a data de entrega e a data de término da tarefa.
- Sequência de tarefas com o *total tardiness* mínimo das tarefas com base na média dos tempos de *setup* e processamento para cada tarefa em todas as máquinas (TOTA), onde

a média dos tempos de *setups* e de processamentos de cada tarefa em todas as máquinas é calculado.

Ding et al. (2015) estudaram problemas para minimizar o *total tardiness* no contexto de *no-wait flowshop* ($\mathbf{F} \mid \textit{no-wait} \mid \Sigma T$). Para isso, introduziram o conceito de tarefas sensíveis. Dessa forma, ao invés de avaliarem o *tardiness* de cada tarefa, avaliaram o *tardiness* das tarefas sensíveis. Assim, houve um ganho computacional de processamento. Foram desenhados um algoritmo NEH acelerado e uma heurística gulosa.

As tarefas sensíveis são aquelas que se deseja inserir em uma sequência. Ou seja, as tarefas sensíveis são aquelas que podem ser seus *tardiness* modificados pela inserção de alguma tarefa na sequência (DING et al, 2015).

Foram elaboradas 2 heurísticas aceleradas, de modo a melhorar o ganho computacional. Elas fazem referência à inserção de tarefas NEH. Dessa forma, a primeira heurística elaborada foi sobre acelerar o algoritmo NEH. Nesse caso, gera-se uma solução inicial através da regra de despacho EDD. Depois avalia-se os duas primeiras tarefas, e seleciona-se a sequência com menor *total tardiness*. Em seguida é utilizado o procedimento de inserção NEH. O segundo método estudado foi em relação a aceleração do algoritmo guloso, no qual é feito por uma fase de destruição e outra de construção. Na primeira fase, tarefas aleatoriamente são removidas da sequência, assim uma sequência parcial é obtida. As tarefas removidas são então adicionados na sequência através do procedimento de inserção NEH (DING et al, 2015).

O problema de *no-wait flowshop* com restrição de data de entrega ($\mathbf{F} \mid \textit{no-wait}, \mathbf{d} \mid \mathbf{C}_{\max}$) foi resolvido por Samarghandi e Behroozi (2016) através da técnica de otimização por algoritmo de enumeração. A função objetivo considerada foi a de minimizar o *makespan*. As características desse problema o tornam fortemente *NP-Hard*. Segundo os autores esse problema foi raramente estudado, devido à dificuldade de demonstrar uma solução viável ou falta dela.

Os pesquisadores Lin e Ying (2016) estudaram o problema de *flowshop* permutacional com *no-wait* e objetivo de minimizar o *makespan* ($\mathbf{Fperm} \mid \textit{no-wait} \mid \mathbf{C}_{\max}$). Como técnica utilizaram a programação inteira mista (MIP) e um algoritmo guloso iterado com dois mecanismos de auto ajuste e um mecanismo de destruição.

O algoritmo desenvolvido possui algumas fases: geração da solução inicial, fase de destruição, fase de construção e critério de parada. Na fase de construção foram criados dois mecanismos de auto ajuste, um que ajusta dinamicamente o tamanho e o alcance da vizinhança

de busca e um outro relacionado com as probabilidades de seleção das regras de destruição durante o processo de busca. Na fase de destruição, um mecanismo é utilizado para escolher a regra de destruição mais adequada, uma vez que considera o critério de desempenho que se deseja otimizar. O processo é iterativo até que se alcance o critério de parada (LIN; YING, 2016).

Samarghandi e Behroozi (2017) propuseram métodos exatos para o problema de *no-wait flowshop* com restrição de data de entrega e com critério de desempenho a minimização do *makespan* ($F \mid \text{no-wait}, d \mid C_{\max}$). As tarefas devem ser finalizadas antes das suas respectivas datas de entrega, por isso considera-se a restrição *hard*. Os métodos desenvolvidos foram um modelo de programação inteira mista (MIP), dois modelos de programação inteira mista quadrática (*quadratic MIP*) e dois modelos de programação de restrição (CP).

3.3 MANUTENÇÃO PREVENTIVA

Uma atividade com muita relação com o *scheduling* da produção é a manutenção, que consiste em uma técnica ou operação que permite manter máquinas em serviço. Tanto a programação das tarefas e a manutenção consomem horas produtivas. Isso implica em uma concorrência de horas, que devem ser balanceadas para que a planta industrial tenha um bom funcionamento e seja confiável (RUIZ; DÍAZ; MATORO, 2007).

A manutenção pode ser dividida em dois grandes grupos. O primeiro é sobre a manutenção corretiva (CM – *corrective maintenance*) e o outro é a manutenção preventiva (PM – *preventive maintenance*). A manutenção corretiva são as atividades de manutenção realizadas quando o equipamento já falhou. E a manutenção preventiva ocorre quando o sistema ainda está em operação, ou seja, são realizadas ações antes que as máquinas falhem. Dentre alguns objetivos da PM, pode-se elencar (RUIZ; DÍAZ; MATORO, 2007):

- Aumento da confiabilidade nas máquinas, que implica em redução de custos, pois as máquinas tendem a falharem menos.
- Aumento da vida operacional do equipamento.
- Melhorias no planejamento e gestão da produção.
- Melhoria na segurança.

Para melhorar a confiabilidade de um sistema é necessário definir uma política de manutenção preventiva para ser aplicada, visto que as máquinas estão expostas a corrosão,

fadiga, desgaste, entre outras variáveis (CHENG; ZHOU; LI, 2017). Existe ainda a manutenção preditiva, que ocorre quando as políticas de manutenção preventiva incorporam informações do estado atual da máquina. Dessa forma, a previsão de falha é mais precisa (HASENBEIN; KUTANOGLU; LIAO, 2013).

Segundo Ruiz, Díaz e Matoro (2007) as políticas de manutenção preventivas são baseadas em teorias estatísticas de confiabilidade. A distribuição de Weibull é uma das mais utilizadas para retratar o tempo de falha das máquinas, já que as taxas de falhas não são constantes nessa distribuição. A seguir está expressa a função de confiabilidade de Weibull ($R(t)$) em dois parâmetros (β = forma, θ = escala), e t representa a vida útil da máquina.

$$R(t) = \exp \left[- \left(\frac{t}{\theta} \right)^\beta \right], \quad t > 0 \quad \text{Eq. (3.5)}$$

Outra medida utilizada é a taxa de falha ($Z(t)$), expressa pela seguinte equação (CASSADY; KUTANOGLU, 2003):

$$Z(t) = \frac{\beta}{\theta^\beta} * t^{\beta-1} \quad \text{Eq. (3.6)}$$

O tempo médio entre falhas MTTF (*mean time to failure*) encontra-se abaixo, no qual há os parâmetros θ e Γ tabulados (RUIZ et al., 2007):

$$MTTF = \theta * \Gamma * \left[1 + \frac{1}{\beta} \right] \quad \text{Eq. (3.7)}$$

A disponibilidade de uma máquina (A) pode ser calculada por uma equação composta por $MTTF$ e $MTTR$ (*mean time to repair*), sendo o último o tempo médio previsto para que o sistema seja reparado, e esteja operando como em condições normais.

$$A = \frac{MTTF}{MTTF + MTTR} \quad \text{Eq. (3.8)}$$

Existem algumas políticas sobre manutenção preventiva, que podem ser aplicadas em problemas de *flowshop*. Ruiz, Díaz e Matoro (2007) citam três dessas políticas, que tem por objetivo maximizar a disponibilidade das máquinas ou manter um nível mínimo de confiabilidade na produção. Para isso criaram um critério para considerar a manutenção preventiva em problemas de *flowshop* com o objetivo de minimizar o *makespan*. Dos métodos adaptados, os que se mostraram mais eficientes foram o algoritmo de colônia de formigas e o algoritmo genético.

- Política 1 (modelo de manutenção preventiva em intervalos de tempo fixos): Muitas indústrias utilizam a estratégia de definirem quando serão as paradas para realizar a manutenção. Nesse caso, não se utilizam modelos probabilísticos de falhas. Os programadores ajustam o cronograma para que existam paradas entre ciclos de produção para haver a manutenção, de forma a não interromper uma ou um conjunto de ordens de produção.

- Política 2 (modelo de manutenção preventiva com maximização da disponibilidade das máquinas): O objetivo dessa política é maximizar a disponibilidade do sistema, uma vez que ele pode ser reparado, falhar ou ter paradas para a manutenção preventiva. Dessa forma, o intervalo ideal entre as manutenções preventivas (T_{PM2}) pode ser planejado segundo a equação abaixo, no qual o parâmetro tr é o tempo de reparo e tp é o tempo gasto na manutenção preventiva.

$$T_{PM2} = \theta * \left[\frac{tp}{tr * (\beta - 1)} \right]^{1/\beta} \quad \text{Eq. (3.9)}$$

- Política 3 (modelo de manutenção preventiva que mantém uma confiabilidade mínima para um determinado período de produção): Essa política visa fazer uma manutenção preventiva sistemática, ou seja, em intervalos regulares, uma vez que, considera-se que a manutenção preventiva restaura as máquinas como se fossem novas. A equação utilizada para essa política (T_{PM3}) encontra-se logo em seguida:

$$T_{PM3} = \left[-\theta^\beta * \frac{\ln Ro(t)}{t} \right]^{1/(\beta-1)} \quad \text{Eq. (3.10)}$$

Sendo $Ro(t)$ uma função de confiabilidade.

3.4 NO-WAIT FLOWSHOP COM TEMPO DE MANUTENÇÃO PREVENTIVA

A programação das tarefas está intimamente correlacionada com a manutenção preventiva, já que o processamento das tarefas utiliza as máquinas, sendo necessário que se faça a manutenção nas mesmas para evitar quebras. Além disso, as ações para realizar a manutenção preventiva fazem com que a máquina fique indisponível, logo, isso afeta o sequenciamento da produção (RUIZ; DÍAZ; MATORO, 2007). O que torna especialmente crítico para problemas envolvendo *no-wait*.

Ng e Kovalyov (2004) estudaram um problema de *flowshop* de 2 máquinas e consideraram que uma das duas máquinas não estava sempre disponível, seja por falha, manutenção preventiva ou tempo destinado para processamento de outros produtos inacabados. O problema estudado não estava restrito a condição de *no-wait*, mas a solução encontrada foi desenvolver um algoritmo de aproximação de tempo polinomial.

Os autores Mokhtaria, Mozdgir e Abadi (2012) também não consideraram a condição de *no-wait* nos seus estudos, porém ressaltaram a importância de modelar a manutenção preventiva, no contexto de máquinas paralelas. O método desenvolvido foi um modelo de programação não linear, e um algoritmo de busca de vizinhança adaptado foi implementado, no qual são geradas múltiplas soluções de teste para se registrar a melhor. E dessa forma, encontrar a solução final a partir da sequência com melhor resultado.

Os pesquisadores Ma, Chu e Zuo (2010) escreveram um artigo sobre *scheduling* com restrição determinística de disponibilidade, uma vez que não identificaram artigos sobre o tema. Restrições determinísticas de disponibilidade são aquelas em que a manutenção é fixada antecipadamente, ou seja, não é um processo estocástico ou aleatório. Um exemplo de indisponibilidade por fator determinístico ocorre quando a máquina não está disponível para iniciar o processamento, pois ainda não terminou de processar tarefas de agendamentos anteriores. Porém, no *survey* realizado pelos pesquisadores não foi identificado nenhum artigo com o critério de otimização *total tardiness*. Os problemas possuíam como medida de desempenho o *makespan*. Os autores citados foram Espinouse, Formanowicz e Penz (1999, 2001), Wang and Cheng (2001), Cheng e Liu (2003a, 2003b) e Kubzin e Strusevich (2004).

Espinouse, Formanowicz e Penz (2001) estudaram o problema de 2 máquinas para *no-wait flowshop* e a otimização considerada foi o *makespan*. As tarefas foram programadas de forma que a sequência fosse primeiro processada pela máquina 1, e depois pela máquina 2. Quando o tempo de manutenção preventiva foi considerado na máquina 1, o problema foi resolvido por um algoritmo de tempo polinomial, e quando o tempo de manutenção preventiva foi considerado na máquina 2, a solução foi elaborar um método baseado no algoritmo de Gilmore Gomory (GGA).

Cheng e Liu (2013a) apresentaram um algoritmo de tempo polinomial para duas máquinas para o problema de *no-wait flowshop* com objetivo de minimizar o *makespan*. Eles consideraram um intervalo de indisponibilidade para apenas uma máquina, ou mesmo intervalos de indisponibilidade que se sobrepõem nas duas máquinas.

Wang e Cheng (2001) propuseram uma heurística baseada no algoritmo de Gilmore Gomory para o problema *no-wait flowshop* de 2 máquinas com restrição de disponibilidade em uma das máquinas e função objetivo de minimização do *makespan*. Dessa forma, os pesquisadores utilizaram alguns procedimentos do GGA, para então melhorarem a programação através da ordenação de algumas tarefas críticas.

Kubzin e Strusevich (2004) estudaram o problema de *no-wait flowshop* com 2 máquinas, no qual uma das máquinas não está disponível em um determinado intervalo. O critério de desempenho avaliado foi o *makespan*. As heurísticas desenvolvidas são $3/2$ e $4/3$ vezes o valor ótimo. A primeira heurística inicia com a permutação de Gilmore Gomory para as duas máquinas disponíveis continuamente. Em seguida, insere-se o *slot* da manutenção preventiva e avalia-se qual tarefas foi afetada, remove-se a tarefa afetada e as duas outras tarefas próximas, e encontra-se a posição ideal para essas três tarefas na sequência com manutenção preventiva. A segunda heurística é similar à primeira, porém é realizado um tratamento especial na tarefa com um tempo de processamento grande e uma enumeração diferente das tarefas, quando há o tempo da manutenção preventiva.

4. PROPOSTA DE MÉTODOS HEURÍSTICOS CONSTRUTIVOS

4.1 HEURÍSTICA CONSTRUTIVA PARA COMPARAÇÃO

Não foram encontrados métodos na literatura que abordavam ao mesmo tempo as características de: *no-wait flowshop*, *total tardiness*, **manutenção preventiva** e **heurística construtiva**. Dessa forma, foi selecionado um método que possui algumas das características desejadas com exceção da manutenção preventiva para ser a heurística a ser comparada com os métodos propostos. Assim, a adaptação foi de baixo impacto, uma vez que não implicou em uma alteração da metodologia proposta pelos autores originais. Essa heurística construtiva que foi utilizada para comparação do desempenho entre as heurísticas proposta foi desenvolvida pelos pesquisadores Liu, Song e Wu (2013).

Liu, Song e Wu (2013) propuseram uma heurística construtiva para o problema de *no-wait flowshop* com critério de *total tardiness*, na qual os autores denominaram de MNEH, pois utilizam o algoritmo NEH modificado.

Passo 1: Ordenar as tarefas sob a regra EDD para gerar uma sequência inicial de tarefas.

Passo 2: Ordenar as duas primeiras tarefas, e selecionar a sequência de menor *total tardiness* entre elas. Caso haja empate, selecionar o de menor *makespan*.

Passo 3: Executar o algoritmo NEH em relação ao critério de menor *total tardiness*. Caso haja empate, selecionar o de menor *makespan*.

Para o cálculo do *total tardiness* foi utilizada uma equação de distância (D_{jk}). Ou seja, distância entre o início da tarefa j e o início da tarefa k (LIU; SONG; WU, 2013):

$$D_{jk} = \max_{1 \leq i \leq m} (\sum_{h=1}^i p_{hj} - \sum_{h=1}^{i-1} p_{hk}) \quad \text{Eq. (4.1)}$$

Sendo, p_{ij} tempo de processamento da tarefa i na máquina i .

Essa distância é então utilizada para o cálculo do tempo de conclusão da tarefa j em uma determinada sequência π ($C[j]/\pi$) (LIU; SONG; WU, 2013):

$$C_{[j]}(\pi) = \sum_{l=2}^i D_{[l-1,l]}(\pi) + P_{[j]}(\pi) \quad \text{Eq. (4.2)}$$

Sendo, P_j o tempo total de processamento da tarefa j em todas as máquinas.

Dessa forma, é possível calcular o tempo do *total tardiness* da sequência (LIU; SONG; WU, 2013):

$$T(\pi) = \sum \max\{C_{[j]}(\pi) - d_{[j]}(\pi), 0\} \quad \text{Eq. (4.3)}$$

Sendo, d_j a data de entrega da tarefa j .

A Figura 4.1 ilustra um Gantt com 3 máquinas e 3 tarefas para o problema de *no-wait, flowshop* assim como as variáveis de interesse D_{jk} , P_j , C_j , d_j e T_j .

Figura 4.1 – Gráfico de Gantt com 3 máquinas e 3 tarefas (LIU; SONG; WU, 2013)

O procedimento de inserção NEH aplicado nesse algoritmo consiste em primeiro possuir uma sequência inicial. Em seguida é criada uma sequência parcial com as duas primeiras tarefas, e a terceira tarefa é então testada na posição um, dois e três, sem que a posição relativa das tarefas 1 e 2 se alterem. Das três sequências geradas é selecionada a de menor *total tardiness*. O procedimento continua com a inserção da quarta tarefa, e até que todas as tarefas tenham sido inseridas. A Figura 4.2 demonstra como o algoritmo NEH executa, ou seja, a tarefa k é posicionada em todas as posições para que seja verificada a sequência candidata de menor *total tardiness* (LIU; SONG; WU, 2013).

Figura 4.2 – Procedimento de inserção NEH (LIU; SONG; WU, 2013)

Os autores, Liu, Song e Wu (2013), ressaltam a baixa publicação de artigos que utilizam o critério de *total tardiness*. E por isso, propuseram algumas heurísticas na qual a de melhor

desempenho a do MNEH, sendo essa heurística utilizada nesse estudo como algoritmo de comparação.

Para a elaboração de heurísticas construtivas propostas com as características de critérios desejados foram utilizados alguns algoritmos como inspiração. Ao total foram desenvolvidos 4 modelos, que estão detalhados a seguir.

Todos os algoritmos propostos utilizam a formulação descrita no artigo do Liu, Song e Wu (2013) para o cálculo do *total tardiness*. E o detalhamento da etapa de manutenção preventiva será abordada no capítulo 5, uma vez que todos as heurísticas proposta utilizam da mesma técnica.

4.2 HEURÍSTICA CONSTRUTIVA I_j + NEH

A primeira heurística foi inspirada no artigo na Ronconi e Henriques (2009). O problema estudado pelos pesquisadores foi o de *flowshop* para minimização do *total tardiness*, porém associados a problemas de bloqueio, ou seja, não é permitido *buffers* entre máquinas sucessivas. A proposta dos autores foi a criação de uma heurística construtiva denominada FPDNEH e uma metaheurística baseada em GRASP.

A inspiração para a proposição do primeiro algoritmo foi baseada na forma em que é gerada a solução inicial na heurística FPDNEH dos pesquisadores Ronconi e Henriques (2009). A primeira etapa do algoritmo FPDNEH é o cálculo do termo I_j para todas as tarefas j , após esse cálculo a tarefa com menor I_j é selecionada para a primeira posição. Para a heurística proposta I_j + NEH todas as tarefas foram sequenciadas de forma crescente em relação ao I_j , e essa foi identificada como solução inicial. O termo I_j é uma composição das datas de entrega e tempos de processamento na primeira máquina, uma vez que afeta diretamente o valor do *total tardiness*. Ou seja:

$$I_j = d_j + p_{j1} \quad \text{Eq. (4.4)}$$

Sendo, d_j a data de entrega da tarefa j e p_{j1} o tempo de processamento da tarefa j na primeira máquina.

O passo seguinte foi aplicar o algoritmo NEH considerando o menor *total tardiness* gerados nas sequências candidatas. Abaixo segue-se um pseudocódigo.

Dado um conjunto $J = \{j_1, j_2, j_3, \dots, j_n\}$ de n tarefas, seja S o conjunto de tarefas programadas, R o conjunto de tarefas não programadas.

Passo 1: $R \leftarrow J; S \leftarrow \emptyset;$

Passo 2: Ordene as tarefas de R em ordem crescente de I_j ($I_j = d_j + p_{jl}$);

Passo 3: Crie uma subsequência com a tarefa de menor I_j , chame essa subsequência de S ;

Passo 4: Inclua no conjunto R as tarefas que não estão no conjunto S ;

Passo 5: Selecione a primeira tarefa do conjunto R e teste em todas as posições possíveis no conjunto S , sem que as posições relativas das tarefas já pertencentes ao conjunto S se alterem (mecanismo NEH). Selecione a subsequência com menor *total tardiness* (T), sendo $T = \sum \max \{C_j - d_j, 0\}$. Caso haja empate no valor do *total tardiness*, selecione a subsequência com menor *makespan*;

Passo 6: Repita o passo 5 até $R \leftarrow \emptyset;$

Passo 7: Calcular o *total tardiness* (T) da sequência final S .

Figura 4.3 – Pseudocódigo do algoritmo I_j + NEH

4.3 HEURÍSTICA CONSTRUTIVA MDD + NEH

Os pesquisadores Aldowaisam e Allahverdi (2012) abordaram o problema de *scheduling* de *no-wait flowshop* com o objetivo de minimizar o *total tardiness*. Em um primeiro momento os autores estudaram várias regras de despacho (EDD, EDDP, MDD, SLACK, SRMWK, SMPT, ASMPT), no qual a MDD se sobressaiu. Logo após utilizaram de metheurísticas de *Simulated Annealing* (SA) e algoritmos genéticos (GA) para resolverem o problema.

Como o principal interesse dessa pesquisa é o estudo de heurísticas construtivas, o método proposto é sequenciar as tarefas segundo a regra MDD, avaliar as duas primeiras tarefas, e em seguida aplicar o algoritmo NEH. Dessa forma o pseudocódigo segue abaixo na Figura 4.4.

Dado um conjunto $J = \{j_1, j_2, j_3, \dots, j_n\}$ de n tarefas, seja S o conjunto de tarefas programadas, R o conjunto de tarefas não programadas.

Figura 4.4 – Pseudocódigo do algoritmo MDD + NEH (continua)

Passo 1: $R \leftarrow J$; $S \leftarrow \emptyset$;

Passo 2: Ordene as tarefas de R em ordem crescente de MDD, sendo $MDD = \max\{C_j, d_j\}$;

Passo 3: Crie duas subsequências de duas tarefas. A primeira subsequência composta pelas duas primeiras tarefas de R ($S' \leftarrow R_1, R_2$), e a segunda subsequência composta pelas duas primeiras tarefas de R , porém na ordem inversa ($S'' \leftarrow R_2, R_1$);

Passo 4: Calcule o *total tardiness* (T) para as sequências S' e S'' , e selecionar a sequência com menor T , sendo $T = \sum \max\{C_j - d_j, 0\}$, e chame de S ;

Passo 5: Inclua no conjunto R as tarefas que não estão no conjunto S ;

Passo 6: Selecione a primeira tarefa do conjunto R e teste em todas as posições possíveis no conjunto S , sem que as posições relativas das tarefas já pertencentes ao conjunto S se alterem (mecanismo NEH). Selecione a subsequência com menor *total tardiness*. Caso haja empate no valor do *total tardiness*, selecione a subsequência com menor *makespan*;

Passo 7: Repita o passo 6 até $R \leftarrow \emptyset$;

Passo 8: Calcular o *total tardiness* (T) da sequência final S .

Figura 4.4 – Pseudocódigo do algoritmo MDD + NEH (conclusão)

4.4 HEURÍSTICA CONSTRUTIVA **EDD + NEH + NSS1**

Utilizou-se o artigo publicado pelos pesquisadores Naderi e Arshadi (2013) e Ding et al. (2015) como inspiração para a proposição do próximo método.

O problema estudado pelos autores Naderi e Arshadi (2013) foram um *no-wait flowshop* para a minimização do *total tardiness*. E a técnica utilizada para a resolução foi busca de vizinhança variável (VNS) e algoritmo de *simulated annealing* (SA). Sendo que o VNS foi subdividido em 2 etapas, o NSS1 e o NSS2.

NSS1: Remove-se aleatoriamente uma tarefa por vez (sem repetição), reinsere-se em todas as posições, e testa-se se o *total tardiness* diminuiu. Se melhorar a sequência gerada pelo NSS1 é aceita, e outra tarefa é selecionada aleatoriamente para executar o mesmo processo.

NSS2: Selecionam-se duas tarefas aleatoriamente e inserem-se aleatoriamente em outra posição. E esse procedimento é testado n (parâmetro) vezes até que outras duas tarefas são selecionadas aleatoriamente para executarem o mesmo processo. A solução melhor é aceita, mesmo que seja inferior a solução atual. Dessa forma, ótimos locais podem ser combatidos.

O problema abordado por Ding et al. (2015) foi uma minimização do *total tardiness* no contexto de *no-wait flowshop*, e a técnica utilizada englobavam 2 heurísticas aceleradas, ou

seja, primeiro obtém-se uma solução inicial por meio da regra de despacho EDD. Em seguida avaliam-se as duas primeiras tarefas, seleciona-se a de menor *total tardiness*, e completa-se a sequência com o procedimento de inserção de tarefas NEH. A segunda heurística destrói e constrói a sequência anterior obtida. Dessa forma, tarefas são removidas aleatoriamente e adicionadas na sequência parcial por meio do procedimento NEH.

Assim, o algoritmo proposto EDD + NEH + NSS1 foi inspirado nos artigos de Naderi e Arshadi (2013) e Ding et al. (2015), e possui os seguintes passos. Primeiro gera-se uma solução inicial por meio da regra de despacho EDD, avaliam-se as duas primeiras tarefas, e em seguida aplica-se o algoritmo NEH. Por fim aplica-se o NSS1, porém sem os fatores aleatórios, ou seja, tarefa a tarefa (primeira a tarefa número 1, depois a tarefa número 2, e assim consecutivamente até que todas as tarefas tenham sido verificadas) é removida e testada nas outras posições se o *total tardiness* diminui, se diminuir a sequência é aceita. A remoção das tarefas é em relação a posição que a tarefa ocupava no conjunto original, que os dados foram disponibilizados, e não em relação a posição da última sequência gerada. Em seguida encontra-se a Figura 4.5 com o pseudocódigo para uma melhor visualização.

Dado um conjunto $J = \{j_1, j_2, j_3, \dots, j_n\}$ de n tarefas, seja S o conjunto de tarefas programadas, R o conjunto de tarefas não programadas.

Passo 1: $R \leftarrow J; S \leftarrow \emptyset;$

Passo 2: Ordene as tarefas de R em ordem crescente de data de entrega (regra EDD);

Passo 3: Crie duas subsequências de duas tarefas. A primeira subsequência composta pelas duas primeiras tarefas de R ($S' \leftarrow R_1, R_2$), e a segunda subsequência composta pelas duas primeiras tarefas de R , porém na ordem inversa ($S'' \leftarrow R_2, R_1$);

Passo 4: Calcule o *total tardiness* (T) para as sequências S' e S'' , e selecionar a sequência com menor T , sendo $T = \sum \max \{C_j - d_j, 0\}$, e chame de S ;

Passo 5: Inclua no conjunto R as tarefas que não estão no conjunto S ;

Passo 6: Selecione a primeira tarefa do conjunto R e teste em todas as posições possíveis no conjunto S , sem que as posições relativas das tarefas já pertencentes ao conjunto S se alterem (mecanismo NEH). Selecione a subsequência com menor *total tardiness*. Caso haja empate no valor do *total tardiness*, selecione a subsequência com menor *makespan*;

Passo 7: Repita o passo 6 até $R \leftarrow \emptyset$;

Passo 8: Considere $j = 1$;

Figura 4.5 – Pseudocódigo do algoritmo EDD + NEH + NSS1 (continua)

<p>Passo 9: Remova a tarefa j e aplique o procedimento NEH. Selecione a sequência de menor <i>total tardiness</i> e chame-a de S.</p> <p>Passo 10: Caso $j \neq n$, considere $j = j + 1$ e vá para o passo 9, caso contrário siga para o passo 11;</p> <p>Passo 11: Calcular o <i>total tardiness</i> (T) da sequência final S.</p>
--

Figura 4.5 – Pseudocódigo do algoritmo EDD + NEH + NSS1 (conclusão)

4.5 HEURÍSTICA CONSTRUTIVA **EDD+ NEH + LS1 + LS2**

A próxima heurística proposta foi baseada na metaheurística desenvolvida pelos autores Ribas, Companys, Tort-Martorell (2013). O problema estudado pelos pesquisadores foi um *scheduling* de bloqueio para *flowshop* com critério de *total tardiness*. O procedimento inicia-se com um NEH, sendo a solução inicial uma sequência gerada por EDD ou FPD. Segundo os autores, ambas regras de despacho para a geração da solução inicial possuem desempenho similar. Então, a sequência gerada é melhorada com algoritmos de busca local, sendo o primeiro, LS1, com o objetivo de realizar troca, e o segundo, LS2, com o objetivo de inserção. O LS1 seleciona aleatoriamente uma tarefa e troca de posição com uma tarefa que está a sua frente na sequência. Já o LS2, seleciona aleatoriamente uma tarefa e insere em todas as posições independente se está localizada antes ou depois. Para ambos a nova sequência candidata só é aceita se o *total tardiness* foi menor que o da atual sequência. A última etapa apresentada pelos autores foi um mecanismo de perturbação para permitir a fuga de ótimos locais.

Como proposta de heurística construtiva baseada no artigo de Ribas, Companys, Tort-Martorell (2013) tem-se o algoritmo NEH + LS1 + LS2, que é composto por uma fase de NEH a partir da solução inicial gerada pela regra EDD, e avaliação das duas primeiras tarefas. As fases seguintes aplicadas foram de inserção, LS1 e LS2 respectivamente, como descrita no artigo de Ribas, Companys, Tort-Martorell (2013), porém sem os fatores aleatórios. Além disso, na etapa LS1, aplicou-se a inserção ao invés da troca. Ou seja, as tarefas selecionadas para inserção seguem uma ordem crescente, primeiro testa-se a tarefa número 1, depois a tarefa número 2 e o processo de repete até que todas as tarefas sejam testadas. A seguir encontra-se o pseudocódigo.

Dado um conjunto $J = \{j_1, j_2, j_3, \dots, j_n\}$ de n tarefas, seja S o conjunto de tarefas programadas, R o conjunto de tarefas não programadas.

Passo 1: $R \leftarrow J; S \leftarrow \emptyset$;

Passo 2: Ordene as tarefas de R em ordem crescente de data de entrega (regra EDD);

Passo 3: Crie duas subsequências de duas tarefas. A primeira subsequência composta pelas duas primeiras tarefas de R ($S' \leftarrow R_1, R_2$), e a segunda subsequência composta pelas duas primeiras tarefas de R , porém na ordem inversa ($S'' \leftarrow R_2, R_1$);

Passo 4: Calcule o *total tardiness* (T) para as sequências S' e S'' , e selecionar a sequência com menor T , sendo $T = \sum \max \{C_j - d_j, 0\}$, e chame de S ;

Passo 5: Inclua no conjunto R as tarefas que não estão no conjunto S ;

Passo 6: Selecione a primeira tarefa do conjunto R e teste em todas as posições possíveis no conjunto S , sem que as posições relativas das tarefas já pertencentes ao conjunto S se alterem (mecanismo NEH). Selecione a subsequência com menor *total tardiness*. Caso haja empate no valor do *total tardiness*, selecione a subsequência com menor *makespan*;

Passo 7: Repita o passo 6 até $R \leftarrow \emptyset$;

Passo 8: Considere $j = 1$;

Passo 9: Remova a tarefa j e aplique o procedimento NEH, porém a posição de teste deve ser maior que a posição original de j . Selecione a sequência de menor *total tardiness* e chame-a de S .

Passo 10: Caso $j \neq n$, considere $j = j + 1$ e vá para o passo 9, caso contrário siga para o passo 11;

Passo 12: Considere $j = 1$;

Passo 13: Remova a tarefa j e aplique o procedimento NEH. Selecione a sequência de menor *total tardiness* e chame-a de S .

Passo 14: Caso $j \neq n$, considere $j = j + 1$ e vá para o passo 13, caso contrário siga para o passo 15;

Passo 15: Calcular o *total tardiness* (T) da sequência final S .

Figura 4.6 – Pseudocódigo do algoritmo EDD+ NEH + LS1 + LS2 (conclusão)

4.6 HEURÍSTICA CONSTRUTIVA COM MANUTENÇÃO PREVENTIVA

Com o objetivo de incluir a característica de manutenção preventiva nos métodos analisados (comparativo ou propostos) executou-se primeiro os algoritmos sem essa

consideração, e após a sequência final com o menor *total tardiness* ter sido gerada, adicionou-se essa restrição.

Para a inclusão das paradas para a manutenção preventiva é necessário calcular a distância entre a data de término da tarefa z e data de início da tarefa j em cada máquina, essa distância é chamada de folga ($dfolga_{izj}$). A Figura 4.7 demonstra graficamente essas distâncias.

$$Dfolga_{izj} = DI_{ij} - DT_{iz} \quad \text{Eq. (4.5)}$$

sendo, DI = data de início, DT = data de término, j e z = tarefas (onde $j = z + 1$), i = máquina

Figura 4.7 – Distância entre as tarefas

Com essas distâncias calculadas entre todas as tarefas em todas as máquinas, verifica-se em qual momento a manutenção preventiva precisa ser realizada. Se a manutenção preventiva precisar ser alocada em um slot ocupado por uma tarefa, verifica-se a quantidade de *slots* de tempo que a tarefa deve ser deslocado para que a manutenção preventiva seja realizada sem que a característica de *no-wait* se perca. A Figura 4.8 ilustra essa situação.

Figura 4.8 – Inserção da manutenção preventiva (exemplo 1)

Outra possibilidade é que a manutenção preventiva precise ser alocada em um *slot* livre, nessa situação é necessário utilizar os dados da matriz de folga para descobrir se é necessário algum deslocamento das tarefas subsequentes, ou se a folga permite que a manutenção preventiva seja realizada. Na Figura 4.9 é possível verificar duas situações, a primeira quando não é necessário deslocamento, e a outra quando é necessário calcular o deslocamento das tarefas subsequentes.

Figura 4.9 – Inserção da manutenção preventiva (exemplo 2)

Após todas as considerações sobre a manutenção preventiva, atualiza-se o valor do *total tardiness*. Os planos de paradas para as manutenções preventivas foram gerados a partir das políticas descritas no capítulo 3 (Revisão da literatura), e no capítulo 5 (Experimentos computacionais) podem-se consultar os parâmetros aplicados.

5. EXPERIMENTOS COMPUTACIONAIS

Todos os métodos foram desenvolvidos em linguagem VBA, e as instâncias foram gerados em um PC com CPU Intel(R) Core(TM) i7-7500U (2,70 GHz e 2,90 GHz) e 8GB de RAM.

Para os cenários foram utilizadas as bases disponíveis e reconhecidas na literatura. Ronconi e Henriques (2007) disponibilizaram instâncias cujos tempos de processamento foram distribuídos uniformemente entre 1 e 99, e as datas de entrega foram uniformemente distribuídas entre $P(1 - T - R/2)$ e $P(1 - T + R/2)$, onde T e R são o fator de *tardiness* das tarefas e intervalo de dispersão das datas de entrega, e P está relacionado com as características de um *flowshop* com *buffer* ilimitado.

Como o problema de interesse é um *flowshop* com manutenção preventiva foi necessário adaptar as datas de entrega, fator P, para contemplar essas restrições. Assim, define-se P como:

$$P = \max \left\{ \max_{1 \leq v \leq m} \left[\sum_{j=1}^n t_{vj} + \min_j \sum_{q=1}^{v-1} t_{qj} + \min_j \sum_{q=v+1}^m t_{qj} + \sum_{x=1}^w r_{vx} \right], \max_j \sum_{v=1}^m t_{vj} \right\} \text{ Eq. (5.1)}$$

Sendo, t_{vj} = tempo de processamento na máquina v da tarefa j , t_{qj} = tempo de processamento na máquina q da tarefa j , m = máquina, r_{vx} = tempo da manutenção preventiva número x na máquina v .

Dessa forma, o termo r_{vx} inclui a condição de manutenção preventiva para a geração das datas de entrega, uma vez que leva em consideração o número máximo de manutenções em cada máquina. A quantidade de manutenções por máquinas foram definidas segundo as políticas 1, 2 e 3.

Foram geradas 120 instâncias com 4 cenários de data de entrega cada, ou seja, variaram-se os parâmetros T e R das seguintes maneiras:

- Cenário 1 (c1): baixo fator de *tardiness* ($T = 0,2$) e pequeno intervalo de datas de entrega ($R = 0,6$).
- Cenário 2 (c2): baixo fator de *tardiness* ($T = 0,2$) e amplo intervalo de datas de entrega ($R = 0,8$).
- Cenário 3 (c3): elevado fator de *tardiness* ($T = 0,4$) e pequeno intervalo de datas de entrega ($R = 0,6$).

- Cenário 4 (c4): elevado fator de *tardiness* ($T = 0,4$) e amplo intervalo de datas de entrega ($R = 0,8$).

Quanto maior o parâmetro T , menores valores de *tardiness* deverão ser respeitados. E quanto maior o parâmetro R , maior a amplitude do intervalo que os valores de *tardiness* deverão respeitar.

Cada heurística foi testada nas 480 instâncias (120 problemas com 4 cenários de data de entrega cada um), que variam de 20 tarefas e 5 máquinas até 500 tarefas e 200 máquinas. Além disso, foram aplicados 3 cenários de manutenção preventiva e 1 cenário sem manutenção, o que totalizam 1.920 instâncias para cada heurística que foi testada.

Para a geração dos valores das manutenções preventivas utilizou-se a seguinte lógica:

- Política 1 (modelo de manutenção preventiva em intervalos de tempo fixos): A cada 100 unidades de tempo processadas é realizada a parada para a manutenção preventiva, sendo as durações das manutenções geradas a partir de uma distribuição uniforme com variação de 1 a 25 unidade de tempo.

- Política 2 (modelo de manutenção preventiva com maximização da disponibilidade das máquinas): Aplicou-se a equação abaixo, na qual foi definida no capítulo 3 (Revisão da literatura), sendo os parâmetros utilizados $\theta = 1500$, $\beta = 2$, $tp = 1$ e $tr = 8$, que resulta em uma parada a cada 530 unidade de tempo processadas por máquina (T_{PM2}). Os valores dos parâmetros foram estabelecidos segundo referências utilizadas pelos pesquisadores do laboratório de operações da EESC/USP. Foi utilizada uma distribuição uniforme com variação de 1 a 25 unidade de tempo para a geração das durações das manutenções preventivas.

$$T_{PM2} = \theta * \left[\frac{tp}{tr * (\beta - 1)} \right]^{1/\beta}$$

- Política 3 (modelo de manutenção preventiva que mantém uma confiabilidade mínima para um determinado período de produção): Aplicou-se a equação abaixo, na qual foi definida no capítulo 3 (Revisão da literatura), sendo os parâmetros utilizados $\theta = 1500$, $\beta = 2$, $t = 800$ e $Ro(t) = 0,85$, que resulta em uma parada a cada 457 unidade de tempo processadas por máquina (T_{PM3}). Também foi utilizada uma distribuição uniforme com variação de 1 a 25 unidade de tempo para a geração das durações das manutenções preventivas.

$$T_{PM3} = \left[-\theta^\beta * \frac{\ln Ro(t)}{t} \right]^{1/(\beta-1)}$$

Para a comparação dos resultados das heurísticas propostas utilizou-se o desvio relativo como método. A equação utilizada segue abaixo:

$$Desvio\ relativo_{Ref} = \frac{R_{ref} - R_{prop}}{R_{ref}} * 100\% \quad Eq. (5.2)$$

Sendo, R_{ref} = resultado do algoritmo referência e R_{prop} = resultado do algoritmo proposto.

Com esse método é possível calcular a melhora ou piora do algoritmo proposto em relação ao algoritmo de referência. Quando o valor de $R_{ref} = R_{prop}$ a melhora do algoritmo proposto em relação ao de referência é zero. Quando $R_{ref} = 0$ e $R_{prop} > 0$ temos um resultado de $-\infty$, nessa situação o exemplo não foi considerado para fim de cálculo de média dos resultados.

Outra medida de comparação utilizada foi o desvio relativo em função do melhor algoritmo. Dessa forma, calcula-se a o quanto distante o método está do melhor resultado. A equação abaixo expressa a equação utilizada.

$$Desvio\ relativo_{Melhor} = \frac{R_{prop} - R_{melhor}}{R_{melhor}} * 100\% \quad Eq. (5.3)$$

Sendo, R_{melhor} = resultado do melhor algoritmo e R_{prop} = resultado do algoritmo proposto.

5.1 RESULTADO DAS HEURÍSTICAS CONSTRUTIVAS SEM MANUTENÇÃO PREVENTIVA

Quatro heurísticas construtivas foram propostas e comparadas com o algoritmo MNEH dos pesquisados Liu, Song e Wu (2013). Em uma primeira perspectiva não foi considerada a restrição de manutenção preventiva.

Os resultados demonstrados nas tabelas seguintes (Tabela 5.1 e Tabela 5.2) mostram que o melhor algoritmo proposto, e que na maioria das situações possui os melhores resultados entre todos os métodos, é o algoritmo EDD + NEH + LS1 + LS2. Nas Tabelas 5.1 e 5.2 os valores em vermelho significam que houve uma piora do resultado gerado pelo algoritmo em relação ao algoritmo referência (Eq. (5.2)), que é o MNEH dos autores Liu, Song e Wu (2013). Os dados em verde significam que foi o algoritmo de melhor resultado para o problema estudado dentre os que estavam em teste. Os valores contidos nas tabelas são resultado de uma média de 10 exemplos cada.

A análise da tabela indica que o algoritmo Ij + NEH não desempenha bem em cenários com datas de entrega “frouxas”, uma vez que os piores resultados estão no cenário 1 e 2. O método MDD + NEH não desempenhou bem em um cenário de datas de entrega “apertadas” e dispersas.

	Ij + NEH				MDD + NEH				EDD + NEH + NSS1				EDD + NEH + LS1 + LS2			
Problema	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
j20 m5	-6,3	-1,1	-1,0	1,5	0,0	0,0	0,0	-2,9	17,5	37,2	11,2	10,4	18,3	36,5	13,2	11,0
j20 m10	1,9	0,1	0,6	-0,2	0,0	-2,4	0,8	0,8	9,3	12,4	5,8	5,6	8,4	13,5	6,9	6,4
j20 m20	-2,4	-3,1	1,2	0,1	-0,4	-2,5	1,1	0,8	3,3	1,9	3,5	2,9	3,2	2,2	3,4	3,1
j50 m5	2,6	-39,7	2,2	0,4	0,0	0,0	0,0	2,2	23,8	63,3	10,5	12,4	22,8	70,1	12,8	14,5
j50 m10	-1,4	-4,0	1,1	1,3	0,0	0,0	0,0	-3,5	12,4	22,1	8,8	5,9	13,7	23,5	11,1	6,9
j50 m20	-1,0	0,7	-0,4	-0,8	0,0	0,5	0,0	-1,6	7,5	6,3	4,7	2,8	9,7	7,2	5,2	3,4
j100 m5	-2,1	-0,8	0,1	1,2	0,0	0,0	0,0	-0,4	28,8	78,7	13,1	17,3	32,2	83,5	16,8	19,5
j100 m10	2,3	-3,2	1,8	-0,2	0,0	0,0	0,0	1,5	12,2	26,5	7,5	9,1	15,3	31,4	9,6	10,5
j100 m20	0,3	-0,9	-0,2	-0,8	0,0	0,0	0,0	-0,1	8,6	10,7	3,8	3,5	10,1	12,9	4,3	4,4
j200 m10	0,0	0,0	0,1	0,1	0,0	0,0	0,0	-0,7	12,6	27,2	7,6	7,5	15,2	33,2	9,5	9,1
j200 m20	1,1	0,1	0,0	-0,7	0,0	0,0	0,0	-0,3	7,6	11,0	4,7	4,2	8,4	14,0	5,6	5,0
j500 m20	-0,8	1,2	-0,3	0,4	0,0	0,0	0,0	0,1	7,2	12,4	4,7	4,7	8,6	15,1	5,8	5,9
Média	-0,5	-4,2	0,4	0,2	0,0	-0,4	0,2	-0,3	12,6	25,8	7,2	7,2	13,8	28,6	8,7	8,3

Tabela 5.1 – Desvio relativo (%) das heurísticas construtivas por cenário sem política de manutenção preventiva

	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	-1,8%	-0,7%	19,1%	19,7%
j20 m10	0,6%	-0,2%	8,3%	8,8%
j20 m20	-1,0%	-0,3%	2,9%	3,0%
j50 m5	-8,6%	0,6%	27,5%	30,0%
j50 m10	-0,7%	-0,9%	12,3%	13,8%
j50 m20	-0,4%	-0,3%	5,3%	6,4%
j100 m5	-0,4%	-0,1%	34,5%	38,0%
j100 m10	0,2%	0,4%	13,8%	16,7%
j100 m20	-0,4%	0,0%	6,6%	7,9%
j200 m10	0,0%	-0,2%	13,7%	16,7%
j200 m20	0,1%	-0,1%	6,9%	8,3%
j500 m20	0,1%	0,0%	7,3%	8,8%
Média	-1,0%	-0,1%	13,2%	14,9%

Tabela 5.2 – Desvio relativo das heurísticas construtivas sem política de manutenção preventiva

Graficamente é possível observar os resultados nas Figuras 5.1 e 5.2. Dados plotados acima do eixo zero indicam que o algoritmo foi melhor que o de referência.

Figura 5.1 – Gráficos dos resultados das heurísticas construtivas por cenário sem política de manutenção preventiva

Figura 5.2 – Gráficos dos resultados das heurísticas construtivas sem política de manutenção preventiva

Em relação ao tempo computacional o algoritmo de melhor resultado foi o que gastou mais tempo de processamento, como demonstrado na Tabela 5.3 e no gráfico 5.3.

segundos	MNEH	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	0,04	0,02	0,03	0,08	0,12
j20 m10	0,04	0,02	0,03	0,08	0,12
j20 m20	0,05	0,03	0,03	0,08	0,12
j50 m5	0,28	0,28	0,28	1,00	1,37
j50 m10	0,28	0,28	0,29	1,00	1,35
j50 m20	0,31	0,31	0,32	1,04	1,38
j100 m5	1,92	2,03	2,04	7,78	10,47
j100 m10	2,19	2,05	2,14	8,30	11,13
j100 m20	2,31	2,14	2,22	8,34	10,59
j200 m10	15,14	15,51	15,66	61,66	84,94
j200 m20	15,35	16,94	15,66	59,84	88,63
j500 m20	237,28	240,02	237,61	942,82	1.299,94

Tabela 5.3 – Tempos de execução das heurísticas construtivas sem política de manutenção preventiva

Figura 5.3 – Gráficos dos tempos de execução das heurísticas construtivas sem política de manutenção preventiva

5.2 RESULTADO DAS HEURÍSTICAS CONSTRUTIVAS COM POLÍTICA MANUTENÇÃO PREVENTIVA 1

Essa seção é dedicada aos resultados dos algoritmos, que depois de gerados é aplicada a Política 1 de manutenção preventiva, na qual utiliza intervalos fixos, sem estudos estáticos ou probabilísticos para definir o intervalo que deve ocorrer as paradas. Dessa forma, arbitrou-se o valor de 100 unidade de tempo de processamento para esse intervalo. Esse valor de intervalo é o mais apertado entre as três políticas, logo, um maior número de paradas ocorrerá.

Dos resultados obtidos, a heurística EDD + NEH + LS1 + LS2 continua com os melhores resultados, como demonstrado nas Tabelas 5.4 e 5.5. O algoritmo MDD + NEH teve uma melhora relativa em relação aos resultados dos testes sem política de manutenção preventiva. Contudo, a geração da solução inicial pelas regras MDD ou Ij não possuem o mesmo desempenho da regra EDD.

As Figuras 5.5 e 5.6 mostram a superioridade dos algoritmos construtivos EDD + NEH + NSS1 e EDD + NEH + LS1 + LS2.

	Ij + NEH				MDD + NEH				EDD + NEH + NSS1				EDD + NEH + LS1 + LS2			
Problema	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
j20 m5	-4,9	-17,5	-0,2	-0,2	0,0	0,0	0,0	0,0	16,0	24,5	11,7	8,2	16,5	26,3	12,4	11,2
j20 m10	0,3	-0,2	-0,4	0,3	0,0	0,0	-0,7	1,4	6,8	11,6	2,9	4,1	7,6	12,3	3,1	4,7
j20 m20	0,8	0,8	0,4	-0,2	0,0	2,9	-0,1	-0,5	5,7	4,3	2,7	1,6	7,5	5,5	2,6	1,9
j50 m5	-3,3	-11,1	1,3	-0,3	0,0	0,0	0,0	0,0	25,0	33,6	14,3	12,4	27,8	33,7	13,8	16,0
j50 m10	0,9	-2,8	-2,5	0,8	0,0	0,0	0,0	0,0	11,3	13,3	8,1	7,1	13,1	12,8	11,2	8,9
j50 m20	-1,4	0,5	0,3	0,1	0,0	0,0	-0,2	1,0	7,0	6,7	3,4	4,0	7,9	8,9	5,0	4,1
j100 m5	-0,9	-5,8	0,8	-1,4	0,0	0,0	0,0	0,0	30,9	37,2	15,2	16,0	32,8	42,3	16,4	18,1
j100 m10	-1,0	2,0	-0,3	-1,3	0,0	0,0	0,0	0,0	13,1	16,6	8,0	8,3	16,2	21,3	9,8	10,3
j100 m20	0,1	0,2	-0,4	-0,6	0,0	0,0	0,0	0,0	8,0	8,2	4,1	3,7	8,5	11,1	4,9	5,1
j200 m10	1,6	-0,4	-0,4	-0,4	0,0	0,0	0,0	0,0	13,4	17,8	7,3	10,1	16,1	20,5	8,5	12,3
j200 m20	0,0	-0,9	1,0	-0,5	0,0	0,0	0,0	0,0	7,6	9,7	4,3	4,1	8,4	10,9	5,0	5,8
j500 m20	-0,1	0,3	-0,3	-0,2	0,0	0,0	0,0	0,0	6,4	8,5	4,1	5,1	7,5	10,6	4,9	7,2
Média	-0,7	-2,9	-0,1	-0,3	0,0	0,2	-0,1	0,1	12,6	16,0	7,2	7,1	14,2	18,0	8,1	8,8

Tabela 5.4 – Desvio relativo (%) das heurísticas construtivas por cenário com política de manutenção preventiva 1

	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	-5,7%	0,0%	15,1%	16,6%
j20 m10	0,0%	0,2%	6,3%	6,9%
j20 m20	0,5%	0,6%	3,6%	4,4%
j50 m5	-3,4%	0,0%	21,3%	22,8%
j50 m10	-0,9%	0,0%	10,0%	11,5%
j50 m20	-0,1%	0,2%	5,3%	6,5%
j100 m5	-1,8%	0,0%	24,8%	27,4%
j100 m10	-0,1%	0,0%	11,5%	14,4%
j100 m20	-0,2%	0,0%	6,0%	7,4%
j200 m10	0,1%	0,0%	12,1%	14,4%
j200 m20	-0,1%	0,0%	6,4%	7,5%
j500 m20	-0,1%	0,0%	6,0%	7,5%
Média	-1,0%	0,1%	10,7%	12,3%

Tabela 5.5 – Desvio relativo das heurísticas construtivas com política de manutenção preventiva 1

Figura 5.4 – Gráficos dos resultados das heurísticas construtivas por cenário com política de manutenção preventiva 1

Figura 5.5 – Gráficos dos resultados das heurísticas construtivas com política de manutenção preventiva 1

Os tempos de execução das heurísticas permanecem na mesma ordem de grandeza, o que indica que a adição do procedimento para inserção da manutenção preventiva não prejudica o desempenho computacional dos algoritmos. A Tabela 5.6 e a Figura 5.6 possuem informações sobre esses tempos de execução.

segundos	MNEH	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	0,25	0,29	0,31	0,36	0,34
j20 m10	0,26	0,31	0,33	0,38	0,34
j20 m20	0,27	0,32	0,34	0,39	0,36
j50 m5	0,51	0,59	0,60	1,35	1,62
j50 m10	0,50	0,54	0,57	1,27	1,71
j50 m20	0,52	0,58	0,60	1,35	1,73
j100 m5	2,19	2,41	2,34	8,92	10,86
j100 m10	2,45	2,33	2,51	8,66	10,93
j100 m20	2,53	2,45	2,66	8,82	11,10
j200 m10	14,80	16,84	15,95	60,14	84,46
j200 m20	15,18	16,31	16,63	62,81	85,34
j500 m20	234,45	242,81	236,03	943,22	1.294,45

Tabela 5.6 – Tempos de execução das heurísticas construtivas com política de manutenção preventiva 1

Figura 5.6 – Gráficos dos tempos de execução das heurísticas construtivas com política de manutenção preventiva 1

5.3 RESULTADO DAS HEURÍSTICAS CONSTRUTIVAS COM POLÍTICA MANUTENÇÃO PREVENTIVA 2

A Política 2 tem por objetivo maximizar a disponibilidade da máquina, por isso a partir de parâmetros é determinado o intervalo ideal que maximize tal interesse. Dentre todas as políticas é a que possui maior intervalo entre uma parada e outra para a realização da manutenção.

Os algoritmos EDD + NEH + NSS1 e EDD + NEH + LS1 + LS2 possuem resultados sempre superiores ao algoritmo de referência MNEH, sendo que para problemas de baixa complexidade (até 20 tarefas) ambos os algoritmos possuem resultados bons. Entretanto, quando a complexidade aumenta para média e alta o algoritmo EDD + NEH + LS1 + LS2 possui desempenho inquestionável, como demonstrado nas Tabelas 5.7 e 5.8.

Os gráficos dos resultados das heurísticas se encontram nas Figuras 5.7 e 5.8. O detalhamento dos tempos de processamento e comparação entre os métodos podem ser consultados na Tabela 5.9 e Figura 5.9.

Problema	Ij + NEH				MDD + NEH				EDD + NEH + NSS1				EDD + NEH + LS1 + LS2			
	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
j20 m5	-2,9	-4,8	-4,5	-2,9	0,0	0,0	0,0	-1,1	17,1	22,4	9,1	8,7	20,9	27,5	8,7	8,8
j20 m10	-2,4	-6,1	0,2	-0,1	0,0	0,0	-0,6	-0,8	10,1	14,5	3,8	3,8	10,7	16,9	3,9	3,6
j20 m20	-1,4	-0,3	1,8	-0,7	-1,1	-0,1	0,1	-1,7	5,1	4,3	1,2	1,6	5,1	4,0	1,6	1,7
j50 m5	-1,5	-4,1	-3,8	0,7	0,0	0,0	0,0	0,0	27,5	29,3	13,6	12,3	28,9	33,8	14,4	13,7
j50 m10	-0,6	3,2	1,6	0,1	0,0	0,0	0,0	0,0	13,0	14,3	6,8	9,1	14,2	18,0	9,4	11,1
j50 m20	2,1	0,0	-1,5	0,4	0,0	0,0	0,0	-0,5	8,1	7,7	2,4	3,9	9,9	9,0	2,9	4,9
j100 m5	1,1	-9,5	-2,0	-0,5	0,0	0,0	0,0	0,0	25,5	34,8	10,8	16,7	28,2	36,5	14,5	18,4
j100 m10	2,3	0,4	-0,5	1,2	0,0	0,0	0,0	0,0	12,2	18,2	8,1	9,4	14,0	19,0	10,0	10,8
j100 m20	0,1	-1,7	1,0	-0,7	0,0	0,0	0,0	0,0	7,4	9,4	5,6	4,6	7,9	10,6	6,9	6,4
j200 m10	0,8	0,6	-0,9	-1,0	0,0	0,0	0,0	0,0	14,1	15,5	7,1	8,9	16,4	18,3	8,6	11,0
j200 m20	0,2	0,1	0,2	-0,3	0,0	0,0	0,0	0,0	6,1	8,2	4,5	4,6	8,5	9,2	5,5	5,2
j500 m20	-0,5	0,1	0,0	0,3	0,0	0,0	0,0	0,0	5,1	8,9	4,8	4,8	7,2	11,0	5,8	5,7
Média	-0,2	-1,8	-0,7	-0,3	-0,1	0,0	0,0	-0,3	12,6	15,6	6,5	7,4	14,3	17,8	7,7	8,5

Tabela 5.7 – Desvio relativo (%) das heurísticas construtivas por cenário com política de manutenção preventiva 2

	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	-3,8%	-0,3%	14,3%	16,5%
j20 m10	-2,1%	-0,4%	8,1%	8,8%
j20 m20	-0,2%	-0,7%	3,0%	3,1%
j50 m5	-2,2%	0,0%	20,7%	22,7%
j50 m10	1,1%	0,0%	10,8%	13,2%
j50 m20	0,3%	-0,1%	5,5%	6,7%
j100 m5	-2,7%	0,0%	22,0%	24,4%
j100 m10	0,9%	0,0%	12,0%	13,5%
j100 m20	-0,3%	0,0%	6,8%	7,9%
j200 m10	-0,1%	0,0%	11,4%	13,6%
j200 m20	0,1%	0,0%	5,8%	7,1%
j500 m20	0,0%	0,0%	5,9%	7,4%
Média	-0,8%	-0,1%	10,5%	12,1%

Tabela 5.8 – Desvio relativo das heurísticas construtivas com política de manutenção preventiva 2

Figura 5.7 – Gráficos dos resultados das heurísticas construtivas por cenário com política de manutenção preventiva 2

Figura 5.8 – Gráficos dos resultados das heurísticas construtivas com política de manutenção preventiva 2

segundos	MNEH	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	0,29	0,29	0,31	0,41	0,46
j20 m10	0,35	0,31	0,32	0,43	0,47
j20 m20	0,37	0,33	0,35	0,48	0,48
j50 m5	0,60	0,59	0,60	1,56	1,74
j50 m10	0,59	0,55	0,57	1,47	1,71
j50 m20	0,62	0,58	0,60	1,52	1,75
j100 m5	2,21	2,30	2,36	8,85	10,88
j100 m10	2,44	2,28	2,71	8,46	10,87
j100 m20	2,55	2,38	2,66	8,63	10,94
j200 m10	14,99	16,87	16,11	61,40	84,97
j200 m20	15,17	16,20	15,59	61,68	87,79
j500 m20	230,39	239,60	239,45	956,49	1.341,91

Tabela 5.9 – Tempos de execução das heurísticas construtivas com política de manutenção preventiva 2

Figura 5.9 – Gráficos dos tempos de execução das heurísticas construtivas com política de manutenção preventiva 2

5.4 RESULTADO DAS HEURÍSTICAS CONSTRUTIVAS COM POLÍTICA MANUTENÇÃO PREVENTIVA 3

A última política estudada mantém uma confiabilidade mínima para um determinado período de produção, por isso um dos parâmetros é uma função de confiabilidade. A superioridade do algoritmo EDD + NEH + LS1 + LS2 se mantém, enquanto a inferioridade do algoritmo Ij + NEH também permanece.

Pode-se inferir que os métodos de inserção LS1 e LS2 combinados são superiores ao NSS1, uma vez que se potencializa a melhora do resultado quando mais de um método de inserção é aplicado. No caso do algoritmo EDD + NEH + LS1 + LS2 são aplicados 3 métodos de inserção, apesar do aumento do tempo computacional, a melhora percentual pode ser vantajosa.

Todos os resultados de desvio relativo e tempo de computação dos algoritmos em todos os cenários encontram-se nas Figuras 5.10, 5.11 e 5.12 e nas Tabelas 5.10, 5.11 e 5.12.

	Ij + NEH				MDD + NEH				EDD + NEH + NSS1				EDD + NEH + LS1 + LS2			
Problema	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
j20 m5	-2,5	0,4	1,3	-1,3	0,0	0,0	0,0	0,0	23,6	23,3	12,6	8,9	24,4	24,5	12,2	9,9
j20 m10	-0,5	-0,2	-0,6	0,2	0,0	0,0	0,1	0,0	9,2	13,9	4,4	4,3	9,4	17,0	3,9	4,5
j20 m20	-0,5	-3,4	0,5	-0,6	-0,8	-1,1	1,7	0,0	3,8	3,7	2,0	2,7	4,1	5,3	2,2	2,8
j50 m5	4,0	-4,9	-2,2	3,1	0,0	0,0	0,0	0,0	26,5	29,8	13,1	15,8	24,6	34,0	13,1	18,5
j50 m10	2,3	-2,3	-3,4	-1,1	0,0	0,0	0,0	0,0	12,9	13,5	6,8	7,6	16,0	17,8	7,2	9,5
j50 m20	1,3	-1,8	0,5	-0,1	0,0	0,0	-0,6	0,5	6,7	6,4	4,5	2,4	7,2	6,9	5,2	2,5
j100 m5	-3,6	1,7	1,3	-3,1	0,0	0,0	0,0	0,0	25,3	37,6	12,9	17,5	28,6	40,7	15,1	18,9
j100 m10	-0,9	2,9	0,9	0,4	0,0	0,0	0,0	0,0	14,8	19,8	8,3	9,9	16,0	21,5	9,5	12,0
j100 m20	0,6	-0,4	0,1	0,1	0,0	0,0	0,0	0,0	8,6	8,3	4,7	5,5	8,1	10,1	5,8	5,8
j200 m10	-2,9	-1,4	-1,1	0,5	0,0	0,0	0,0	0,0	12,1	16,7	7,3	8,2	14,1	19,0	8,3	9,8
j200 m20	-1,3	1,0	1,3	-0,1	0,0	0,0	0,0	0,0	6,9	7,4	4,0	4,9	8,4	9,6	5,1	5,7
j500 m20	0,7	0,1	0,0	-0,5	0,0	0,0	0,0	0,0	6,9	7,6	4,6	4,9	8,3	10,8	5,8	6,5
Média	-0,3	-0,7	-0,1	-0,2	-0,1	-0,1	0,1	0,0	13,1	15,7	7,1	7,7	14,1	18,1	7,8	8,9

Tabela 5.10 – Desvio relativo (%) das heurísticas construtivas por cenário com política de manutenção preventiva 3

	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	-0,5%	0,0%	17,1%	17,7%
j20 m10	-0,3%	0,0%	7,9%	8,7%
j20 m20	-1,0%	0,0%	3,1%	3,6%
j50 m5	0,0%	0,0%	21,3%	22,5%
j50 m10	-1,1%	0,0%	10,2%	12,6%
j50 m20	0,0%	0,0%	5,0%	5,5%
j100 m5	-0,9%	0,0%	23,3%	25,8%
j100 m10	0,8%	0,0%	13,2%	14,8%
j100 m20	0,1%	0,0%	6,7%	7,4%
j200 m10	-1,2%	0,0%	11,1%	12,8%
j200 m20	0,2%	0,0%	5,8%	7,2%
j500 m20	0,1%	0,0%	6,0%	7,9%
Média	-0,3%	0,0%	10,9%	12,2%

Tabela 5.11 – Desvio relativo das heurísticas construtivas com política de manutenção preventiva 3

Figura 5.10 – Gráficos dos resultados das heurísticas construtivas por cenário com política de manutenção preventiva 3

Figura 5.11 – Gráficos dos resultados das heurísticas construtivas com política de manutenção preventiva 3

Segundos	MNEH	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	0,25	0,29	0,31	0,41	0,47
j20 m10	0,26	0,30	0,32	0,44	0,48
j20 m20	0,27	0,33	0,34	0,41	0,50
j50 m5	0,51	0,59	0,59	1,33	1,84
j50 m10	0,52	0,55	0,56	1,28	1,73
j50 m20	0,55	0,59	0,59	1,33	1,75
j100 m5	2,14	2,33	2,28	8,07	10,88
j100 m10	2,47	2,26	2,46	8,65	10,83
j100 m20	2,51	2,38	2,52	8,09	10,85
j200 m10	14,81	15,84	15,44	61,99	84,83
j200 m20	15,21	16,38	15,77	62,43	89,02
j500 m20	233,04	238,36	237,35	935,60	1.355,76

Tabela 5.12 – Tempos de execução das heurísticas construtivas com política de manutenção preventiva 3

Figura 5.12 – Gráficos dos tempos de execução das heurísticas construtivas com política de manutenção preventiva 3

5.5 COMPARAÇÃO DAS HEURÍSTICAS CONSTRUTIVAS

Ao aplicar a equação Eq(5.3) de desvio relativo em relação ao melhor algoritmo (menor *total tardiness*), verifica-se que o algoritmo EDD + NEH + LS1 + LS2 é o de menor desvio. A seguir encontram-se os resultados consolidados dos cenários por política e heurísticas.

Problema	MNEH	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	33,7%	35,2%	34,5%	2,5%	2,0%
j20 m10	11,5%	10,9%	11,7%	1,3%	0,5%
j20 m20	3,6%	4,7%	3,9%	0,6%	0,5%
j50 m5	93,4%	106,5%	92,7%	8,4%	0,9%
j50 m10	17,9%	18,8%	18,9%	2,2%	0,4%
j50 m20	7,2%	7,6%	7,5%	1,4%	0,2%
j100 m5	4483,2%	2955,3%	4483,3%	104,2%	0,6%
j100 m10	22,5%	22,3%	22,1%	4,0%	0,2%
j100 m20	9,0%	9,4%	9,0%	1,5%	0,1%
j200 m10	22,3%	22,2%	22,5%	4,1%	0,0%
j200 m20	9,3%	9,2%	9,4%	1,7%	0,1%
j500 m20	9,9%	9,7%	9,9%	1,8%	0,0%
Média	393,6%	267,6%	393,8%	11,1%	0,5%

Tabela 5.13 – Desvio relativo (em relação ao melhor) das heurísticas sem política de manutenção preventiva

Problema	MNEH	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	24,3%	32,0%	24,3%	3,3%	1,3%
j20 m10	8,3%	8,1%	8,1%	1,0%	0,3%
j20 m20	5,6%	5,1%	4,9%	1,7%	0,8%
j50 m5	36,9%	42,8%	36,9%	4,7%	2,6%
j50 m10	14,0%	14,9%	14,0%	2,4%	0,6%
j50 m20	7,4%	7,5%	7,1%	1,6%	0,3%
j100 m5	44,0%	46,8%	44,0%	5,8%	1,5%
j100 m10	17,7%	17,7%	17,7%	3,8%	0,2%
j100 m20	8,6%	8,7%	8,6%	1,9%	0,3%
j200 m10	17,4%	17,3%	17,4%	2,9%	0,2%
j200 m20	8,6%	8,7%	8,6%	1,5%	0,3%
j500 m20	8,2%	8,3%	8,2%	1,6%	0,0%
Média	16,8%	18,2%	16,7%	2,7%	0,7%

Tabela 5.14 – Desvio relativo (em relação ao melhor) das heurísticas com política de manutenção preventiva 1

Problema	MNEH	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	23,7%	27,5%	24,0%	3,8%	0,8%
j20 m10	12,3%	14,5%	12,7%	2,4%	1,3%
j20 m20	4,3%	4,3%	4,9%	0,9%	0,8%
j50 m5	34,9%	37,0%	34,9%	4,9%	1,9%
j50 m10	16,5%	14,9%	16,5%	3,3%	0,4%
j50 m20	7,7%	7,3%	7,8%	1,6%	0,3%
j100 m5	39,3%	42,9%	39,3%	6,2%	2,7%
j100 m10	16,8%	15,6%	16,8%	2,4%	0,6%
j100 m20	9,1%	9,4%	9,1%	1,6%	0,3%
j200 m10	16,3%	16,4%	16,3%	2,8%	0,2%
j200 m20	8,0%	7,9%	8,0%	1,6%	0,3%
j500 m20	8,2%	8,2%	8,2%	1,8%	0,1%
Média	16,4%	17,2%	16,5%	2,8%	0,8%

Tabela 5.15 – Desvio relativo (em relação ao melhor) das heurísticas com política de manutenção preventiva 2

Problema	MNEH	Ij + NEH	MDD + NEH	EDD + NEH + NSS1	EDD + NEH + LS1 + LS2
j20 m5	30,7%	30,0%	30,7%	4,6%	3,2%
j20 m10	11,7%	11,7%	11,7%	2,0%	1,0%
j20 m20	4,4%	5,4%	4,4%	1,1%	0,5%
j50 m5	35,3%	35,6%	35,3%	4,4%	2,8%
j50 m10	16,0%	17,1%	16,0%	3,6%	0,5%
j50 m20	6,8%	6,8%	6,9%	1,4%	0,8%
j100 m5	40,7%	41,3%	40,7%	5,0%	1,2%
j100 m10	19,0%	17,8%	19,0%	2,6%	0,8%
j100 m20	8,8%	8,7%	8,8%	1,4%	0,6%
j200 m10	15,7%	17,2%	15,7%	2,6%	0,5%
j200 m20	8,0%	7,8%	8,0%	1,7%	0,2%
j500 m20	8,7%	8,6%	8,7%	2,1%	0,1%
Média	17,1%	17,3%	17,1%	2,7%	1,0%

Tabela 5.16 – Desvio relativo (em relação ao melhor) das heurísticas com política de manutenção preventiva 3

6. PROPOSTA DE METAHEURÍSTICA

Soluções para a resolução de problemas de *flowshop no-wait* com manutenção preventiva e com o objetivo de minimizar o *total tardiness* são pouco exploradas, apesar de vários segmentos de indústrias terem esse desafio. Como mencionado em capítulos anteriores, não foram encontradas referências para esse problema em específico. Por isso, estudou-se nessa dissertação possíveis soluções com heurísticas construtivas, sendo o métodos com melhor desempenho o do algoritmo EDD + NEH + LS1 + LS2.

Como forma de propor uma melhoria nas heurísticas construtivas estudadas, propõem-se a elaboração de uma metaheurística. Dessa forma, a proposta foi adaptar a heurística construtiva com o menor tempo computacional para se tornar uma metaheurística, e assim realizar comparações com a melhor heurística construtiva. Foi selecionada a heurística com menor tempo computacional uma vez que metaheurísticas tendem a consumir mais tempo de execução.

A metaheurística proposta foi baseada no método proposto por Ruiz e Stutzle (2007), na qual a solução inicial é composta por uma ordenação decrescente de tempos de processamento. Em seguida, são selecionadas as duas primeiras tarefas para que as duas possíveis sequências parciais sejam avaliadas. Com o procedimento NEH são inseridas as outras tarefas de acordo com a sequência inicial das mesmas. A sequência gerada alimenta o algoritmo IG (*iterated greedy*) que é constituído por uma fase de destruição onde alguns tarefas são eliminadas, e outra fase de construção onde as tarefas eliminadas são reinseridas na sequência uma a uma e na ordem que foram eliminadas por meio do algoritmo NEH. A quantidade de tarefas eliminadas é um parâmetro, e o algoritmo só finaliza quando um critério de aceitação é atingido.

A metaheurística proposta é inspirada no procedimento IG (*iterated greedy*), e é composta pela solução gerada pelo algoritmo MNEH acrescido do IG. O método intitula-se NEH + IG sendo que 3 configurações de parâmetros foram testados:

NEH + IG[1]: Possui quantidade parametrizável de tarefas removidas na fase de destruição do IG. Essa quantidade de tarefas (*Rem*) é função do número de máquinas e tarefas do problema, e segue a seguinte equação:

$$Rem = arr.cima \left(\frac{j}{m/2} \right) \quad \text{Eq. (6.1)}$$

Sendo, Rem = quantidade de tarefas removidas, j = quantidade de tarefas, m = quantidade de máquinas.

Além da quantidade parametrizável de tarefas removidas, a configuração [1] possui critério de parada com $fator = 1$, sendo o $fator$ um componente da equação do *loop*:

$$Loop = arr.cima \left(\frac{j}{Rem} \right) * fator \quad \text{Eq. (6.2)}$$

Ou seja, se $Rem = 2$ e a quantidade de tarefas do problema é 10, tem-se $loop = 5$. Dessa forma, executa-se o procedimento do IG 5 vezes, uma vez que o $fator = 1$. Para esse mesmo exemplo, se $fator$ fosse igual a 10, executar-se-ia o procedimento IG 50 vezes.

NEH + IG[2]: Possui quantidade fixa de tarefas removidas na fase de destruição, sendo 3 essa quantidade. Além disso, $fator = 1$.

NEH + IG[3]: Possui quantidade fixa de tarefas removidas na fase de destruição, sendo 3 essa quantidade. Além disso, $fator = 10$.

O objetivo dessas configurações é testar a influência da parametrização (em função do número de tarefas e máquinas) da quantidade de tarefas removidas e o impacto do aumento do $fator$. Abaixo segue pseudocódigo do método NEH + IG.

Dado um conjunto $J = \{j_1, j_2, j_3, \dots, j_n\}$ de n tarefas, seja S o conjunto de tarefas programadas, R o conjunto de tarefas não programadas.

Passo 1: $R \leftarrow J$; $S \leftarrow \emptyset$;

Passo 2: Ordene as tarefas de R em ordem crescente de data de entrega (regra EDD);

Passo 3: Crie duas subsequências de duas tarefas. A primeira subsequência composta pelas duas primeiras tarefas de R ($S' \leftarrow R_1, R_2$), e a segunda subsequência composta pelas duas primeiras tarefas de R , porém na ordem inversa ($S'' \leftarrow R_2, R_1$);

Passo 4: Calcule o *total tardiness* (T) para as sequências S' e S'' , e selecione a sequência com menor T , sendo $T = \sum \max \{C_j - d_j, 0\}$, e chame de S ;

Passo 5: Inclua no conjunto R as tarefas que não estão no conjunto S ;

Figura 6.1 – Pseudocódigo do algoritmo NEH + IG (continua)

Passo 6: Selecione a primeira tarefa do conjunto R e teste em todas as posições possíveis no conjunto S , sem que as posições relativas das tarefas já pertencentes ao conjunto S se alterem (mecanismo NEH). Selecione a subsequência com menor *total tardiness*. Caso haja empate no valor do *total tardiness*, selecione a subsequência com menor *makespan*;

Passo 7: Repita o passo 6 até $R \leftarrow \emptyset$;

Passo 8: Calcule a quantidade de tarefas que serão removidas (Rem) e a quantidade de *loops* ($Loop$);

Passo 9: Considere $QteLoop = 1$

Passo 10: Remova Rem tarefas aleatoriamente da sequência S e coloque na sequência R . Aplique o procedimento NEH até $R \leftarrow \emptyset$. Selecione a sequência de menor *total tardiness* e chame-a de S .

Passo 11: Caso $QteLoop \neq Loop$, considere $QteLoop = QteLoop + 1$ e vá para o passo 10, caso contrário siga para o passo 12;

Passo 12: Calcular o *total tardiness* (T) da sequência final S .

Figura 6.1 – Pseudocódigo do algoritmo NEH + IG (conclusão)

6.1 RESULTADO DA METAHEURÍSTICA SEM POLÍTICA DE MANUTENÇÃO PREVENTIVA

Os primeiros resultados foram no cenário sem manutenção preventiva. Nessa situação os algoritmos NEH + IG[1] e NEH + IG[2] foram piores que a heurística construtiva de melhor resultado EDD + NEH + LS1 + LS2 (utilizada como referência). Entretanto, a metaheurística NEH + IG[3] demonstrou possuir bom desempenho para problemas de médio e grande porte. Os dados que comprovam o desempenho relativo entre o método de referência podem ser consultados nas Tabelas 6.1 e 6.2 e nas Figuras 6.2 e 6.3.

Alguns gráficos contidos nas Figuras 6.2 e 6.3 tiveram o eixo y cortado, uma vez que o não corte prejudicava a visualização dos mesmos. E os valores gerados por cada algoritmo e cenário podem ser consultados nas Tabelas 6.1 e 6.2.

	NEH + IG [1]				NEH + IG [2]				NEH + IG [3]			
Problema	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
j20 m5	-24,6	-48,4	-7,5	-11,6	-19,0	-17,5	-5,7	-4,8	-7,0	-11,3	0,8	1,1
j20 m10	-10,1	-11,4	-6,3	-6,6	-1,7	-7,1	-4,5	-3,1	-1,6	-0,7	-0,2	-0,5
j20 m20	-0,7	-0,8	-1,1	-1,8	-2,1	-0,3	-1,9	-1,3	-1,1	-1,2	-0,6	0,6
j50 m5	-33,7	-170,5	-14,8	-18,1	-14,5	-45,6	-5,5	-7,1	6,1	7,5	2,1	1,9
j50 m10	-11,3	-14,7	-6,8	-5,0	-7,5	-11,6	-7,3	-2,7	2,2	4,4	1,7	0,4
j50 m20	-5,1	-3,0	-2,8	-2,6	-5,4	-2,4	-2,1	-1,6	1,2	1,3	2,6	-0,2
j100 m5	-56,1	-346,6	-17,7	-23,0	-18,1	-66,8	-8,3	-8,9	2,9	47,6	-1,2	1,1
j100 m10	-11,3	-23,0	-7,1	-8,6	-7,3	-15,9	-4,8	-4,4	3,6	3,8	2,5	1,4
j100 m20	-5,1	-6,6	-2,4	-3,4	-4,7	-7,1	-1,3	-2,0	1,1	1,8	2,9	0,7
j200 m10	-15,0	-23,5	-8,0	-9,3	-8,0	-16,8	-4,7	-4,2	3,4	9,4	1,8	0,7
j200 m20	-4,5	-8,7	-3,7	-3,8	-3,7	-6,5	-3,0	-2,4	3,2	3,1	1,3	1,5
j500 m20	-6,4	-10,4	-5,2	-5,1	-3,8	-7,2	-2,7	-3,2	2,0	3,2	0,7	1,3
Média	-15,3	-55,6	-7,0	-8,2	-8,0	-17,1	-4,3	-3,8	1,3	5,7	1,2	0,8

Tabela 6.1 – Desvio relativo (%) das metaheurísticas por cenário sem política de manutenção preventiva

	NEH + IG [1]	NEH + IG [2]	NEH + IG [3]
j20 m5	-23,0%	-11,7%	-4,1%
j20 m10	-8,6%	-4,1%	-0,8%
j20 m20	-1,1%	-1,4%	-0,6%
j50 m5	-59,3%	-18,2%	4,4%
j50 m10	-9,4%	-7,3%	2,2%
j50 m20	-3,4%	-2,9%	1,2%
j100 m5	-110,8%	-25,5%	12,6%
j100 m10	-12,5%	-8,1%	2,8%
j100 m20	-4,4%	-3,8%	1,6%
j200 m10	-13,9%	-8,4%	3,8%
j200 m20	-5,2%	-3,9%	2,3%
j500 m20	-6,8%	-4,2%	1,8%
Média	-21,5%	-8,3%	2,3%

Tabela 6.2 – Desvio relativo das metaheurísticas sem política de manutenção preventiva

Figura 6.2 – Gráficos dos resultados das metaheurísticas por cenário sem política de manutenção preventiva

Figura 6.3 – Gráficos dos resultados das metaheurísticas sem política de manutenção preventiva

Em relação ao tempo de execução, os algoritmos NEH + IG[1] e NEH + IG[2] são mais rápidos que os outros algoritmos, em contrapartida possuem os piores resultados, como pode ser visualizado na Tabela 6.3 e Figura 6.4.

segundos	EDD + NEH + LS1 + LS2	NEH + IG [1]	NEH + IG [2]	NEH + IG [3]
j20 m5	0,12	0,08	0,07	0,48
j20 m10	0,12	0,08	0,07	0,46
j20 m20	0,12	0,10	0,08	0,47
j50 m5	1,37	0,90	0,97	7,09
j50 m10	1,35	0,89	0,98	7,19
j50 m20	1,38	0,98	1,00	7,29
j100 m5	10,47	6,61	7,61	58,47
j100 m10	11,13	6,68	7,51	57,94
j100 m20	10,59	7,32	7,56	58,90
j200 m10	84,94	53,79	60,81	462,89
j200 m20	88,63	58,66	61,37	462,05
j500 m20	1.299,94	873,76	946,89	7.408,50

Tabela 6.3 – Tempos de execução das metaheurísticas sem política de manutenção preventiva

Figura 6.4 – Gráficos dos tempos de execução das metheurísticas sem política de manutenção preventiva

6.2 RESULTADO DA METAHEURÍSTICA COM POLÍTICA MANUTENÇÃO PREVENTIVA 1

Assim como no estudo sem manutenção, o algoritmo NEH + IG[3] foi o de melhor resultado geral apesar do elevado tempo de execução. Para o problema de 500 tarefas e 20 máquinas o algoritmo demora em média 7.400 segundos para resolver, muito em função da linguagem de programação utilizada.

Abaixo seguem os resultados para as metaheurísticas com política de manutenção preventiva 1, Figura 6.5, 6.6 e Tabelas 6.4, 6.5. O detalhamento do tempo computacional pode ser consultado na Tabela 6.6 e na Figura 6.7.

	NEH + IG [1]				NEH + IG [2]				NEH + IG [3]			
Problema	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
j20 m5	-34,6	-46,7	-4,9	-10,4	-13,4	-18,7	-10,2	-7,7	0,1	-7,0	2,5	0,8
j20 m10	-4,8	-7,9	-2,1	-1,6	-3,1	-4,3	-3,6	-3,6	-0,1	2,7	-1,0	1,6
j20 m20	-4,1	-3,4	-1,3	-0,3	-4,2	-2,4	-1,1	-0,6	1,5	0,8	0,8	0,3
j50 m5	-37,5	-69,2	-12,7	-18,1	-14,6	-16,8	-5,7	-7,4	8,8	2,1	1,9	-0,1
j50 m10	-10,8	-8,1	-7,9	-6,1	-6,6	-3,8	-6,8	-3,2	2,8	7,5	1,8	3,6
j50 m20	-5,5	-5,5	-2,7	-2,0	-4,4	-3,1	-1,1	-1,9	0,5	0,4	0,7	1,7
j100 m5	-60,8	-97,6	-18,4	-18,7	-24,4	-35,1	-7,5	-6,8	-3,1	-2,8	-2,6	3,3
j100 m10	-14,9	-16,6	-7,3	-6,7	-7,4	-9,5	-4,1	-6,0	2,0	4,3	3,0	2,5
j100 m20	-5,2	-5,6	-3,3	-3,8	-3,3	-4,5	-2,6	-2,6	2,8	1,7	2,8	2,9
j200 m10	-13,5	-16,1	-8,5	-8,7	-6,0	-10,1	-3,4	-5,1	5,0	3,2	2,0	2,9
j200 m20	-5,0	-5,2	-3,9	-3,6	-3,4	-5,3	-2,4	-2,7	1,9	3,3	1,6	1,3
j500 m20	-5,7	-7,5	-4,4	-5,8	-4,0	-4,5	-2,6	-3,8	1,3	1,2	1,7	0,4
Média	-16,9	-24,1	-6,4	-7,2	-7,9	-9,8	-4,2	-4,3	1,9	1,5	1,3	1,8

Tabela 6.4 – Desvio relativo (%) das metaheurísticas por cenário com política de manutenção preventiva 1

	NEH + IG [1]	NEH + IG [2]	NEH + IG [3]
j20 m5	-24,2%	-12,5%	-0,9%
j20 m10	-4,1%	-3,6%	0,8%
j20 m20	-2,3%	-2,1%	0,8%
j50 m5	-34,4%	-11,1%	3,2%
j50 m10	-8,2%	-5,1%	3,9%
j50 m20	-3,9%	-2,6%	0,8%
j100 m5	-48,9%	-18,4%	-1,3%
j100 m10	-11,4%	-6,7%	2,9%
j100 m20	-4,5%	-3,2%	2,5%
j200 m10	-11,7%	-6,1%	3,3%
j200 m20	-4,4%	-3,5%	2,0%
j500 m20	-5,8%	-3,7%	1,1%
Média	-13,6%	-6,6%	1,6%

Tabela 6.5 – Desvio relativo das metaheurísticas com política de manutenção preventiva 1

Figura 6.5 – Gráficos dos resultados das metaheurísticas por cenário com política de manutenção preventiva 1

Figura 6.6 – Gráficos dos resultados das metaheurísticas com política de manutenção preventiva 1

segundos	EDD + NEH + LS1 + LS2	NEH + IG [1]	NEH + IG [2]	NEH + IG [3]
j20 m5	0,34	0,39	0,24	0,63
j20 m10	0,34	0,40	0,24	0,64
j20 m20	0,36	0,44	0,26	0,65
j50 m5	1,62	1,26	1,16	7,29
j50 m10	1,71	1,30	1,14	7,61
j50 m20	1,73	1,33	1,17	7,63
j100 m5	10,86	7,00	7,94	59,18
j100 m10	10,93	7,10	7,81	63,07
j100 m20	11,10	7,71	7,94	63,17
j200 m10	84,46	56,71	61,14	477,15
j200 m20	85,34	60,76	62,15	473,11
j500 m20	1.294,45	875,71	961,32	7.420,51

Tabela 6.6 – Tempos de execução das metaheurísticas com política de manutenção preventiva 1

Figura 6.7 – Gráficos dos tempos de execução das metheurísticas com política de manutenção preventiva 1

6.3 RESULTADO DA METAHEURÍSTICA COM POLÍTICA MANUTENÇÃO PREVENTIVA 2

Os resultados demonstram que deixar parametrizável o cálculo para a quantidade de tarefas a serem removidas não gera bons resultados, ao passo que uma quantidade fixa e pequena melhora os resultados. Mas, a melhora efetiva só é alcançada quando uma quantidade maior de *loops* são empregadas, que é o caso do algoritmo NEH + IG[3], uma vez que mais soluções candidatas são testadas.

Em relação as políticas de manutenção preventiva empregadas, a política 2 continua desempenhando o mesmo comportamento que a política 1, ou seja, existe uma melhora no algoritmo NEH + IG[3] e uma piora dos outros dois algoritmos (NEH + IG[1] e NEH[2]). Abaixo seguem os resultados para as metaheurísticas com política de manutenção preventiva 2, Figura 6.8, 6.9 e 6.10 e Tabelas 6.7, 6.8 e 6.9.

Problema	NEH + IG [1]				NEH + IG [2]				NEH + IG [3]			
	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
j20 m5	-27,5	-35,3	-8,1	-7,8	-17,0	-16,3	-2,8	0,1	-9,1	-4,6	0,4	0,3
j20 m10	-4,0	-10,7	-0,6	-3,1	-5,4	-13,6	-0,2	-2,2	2,2	-0,2	1,6	0,7
j20 m20	-1,3	-3,2	-1,1	-0,7	-4,1	-2,6	-0,8	-1,1	0,8	0,9	1,4	0,1
j50 m5	-40,9	-78,7	-15,4	-17,7	-25,9	-18,9	-5,6	-5,2	-5,5	-7,1	4,8	0,9
j50 m10	-10,6	-11,8	-7,1	-9,2	-8,2	-8,3	-4,9	-5,6	6,3	7,2	1,4	0,7
j50 m20	-4,9	-4,6	-2,9	-3,0	-4,1	-4,7	-2,3	-2,9	-0,7	2,9	0,4	1,0
j100 m5	-40,8	-79,9	-18,2	-19,6	-15,0	-19,9	-8,2	-9,1	1,4	4,1	-2,9	1,1
j100 m10	-10,8	-13,3	-7,1	-7,1	-7,1	-6,7	-5,7	-3,9	4,6	8,9	0,6	0,6
j100 m20	-3,7	-7,3	-3,0	-4,9	-2,5	-3,6	-3,8	-4,0	3,4	1,8	1,6	1,7
j200 m10	-14,9	-16,2	-7,3	-8,7	-7,3	-6,7	-4,5	-5,2	2,8	5,4	2,3	0,4
j200 m20	-7,2	-5,8	-3,5	-4,2	-5,0	-4,3	-1,8	-2,3	0,5	3,4	1,8	1,6
j500 m20	-5,9	-8,5	-4,6	-4,1	-3,1	-5,7	-2,7	-2,5	1,6	0,7	1,3	1,7
Média	-14,4	-22,9	-6,6	-7,5	-8,7	-9,3	-3,6	-3,7	0,7	1,9	1,2	0,9

Tabela 6.7 – Desvio relativo (%) das metaheurísticas por cenário com política de manutenção preventiva 2

	NEH + IG [1]	NEH + IG [2]	NEH + IG [3]
j20 m5	-19,7%	-9,0%	-3,3%
j20 m10	-4,6%	-5,3%	1,1%
j20 m20	-1,6%	-2,2%	0,8%
j50 m5	-38,2%	-13,9%	-1,7%
j50 m10	-9,7%	-6,8%	3,9%
j50 m20	-3,8%	-3,5%	0,9%
j100 m5	-39,6%	-13,0%	0,9%
j100 m10	-9,6%	-5,9%	3,7%
j100 m20	-4,7%	-3,5%	2,1%
j200 m10	-11,8%	-5,9%	2,7%
j200 m20	-5,2%	-3,3%	1,8%
j500 m20	-5,8%	-3,5%	1,3%
Média	-12,8%	-6,3%	1,2%

Tabela 6.8 – Desvio relativo das metaheurísticas com política de manutenção preventiva 2

Figura 6.8 – Gráficos dos resultados das metaheurísticas por cenário com política de manutenção preventiva 2

Figura 6.9 – Gráficos dos resultados das metaheurísticas com política de manutenção preventiva 2

segundos	EDD + NEH + LS1 + LS2	NEH + IG [1]	NEH + IG [2]	NEH + IG [3]
j20 m5	0,46	0,39	0,24	0,64
j20 m10	0,47	0,40	0,25	0,67
j20 m20	0,48	0,44	0,26	0,66
j50 m5	1,74	1,25	1,17	7,35
j50 m10	1,71	1,24	1,29	7,51
j50 m20	1,75	1,33	1,34	7,46
j100 m5	10,88	7,09	8,26	59,33
j100 m10	10,87	6,90	7,78	58,16
j100 m20	10,94	7,61	7,83	58,40
j200 m10	84,97	59,29	61,22	475,10
j200 m20	87,79	64,13	63,08	476,52
j500 m20	1.341,91	886,39	945,88	7.441,77

Tabela 6.9 – Tempos de execução das metaheurísticas com política de manutenção preventiva 2

Figura 6.10 – Gráficos dos tempos de execução das metheurísticas com política de manutenção preventiva 2

6.4 RESULTADO DA METAHEURÍSTICA COM POLÍTICA MANUTENÇÃO PREVENTIVA 3

Para a política 3 de manutenção preventiva os resultados seguem os mesmos padrões observados para os outros cenários de política. Logo em seguida encontram-se os resultados (Figura 6.11, 6.12 e 6.13 e Tabelas 6.10, 6.11 e 6.12).

	NEH + IG [1]				NEH + IG [2]				NEH + IG [3]			
Problema	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
j20 m5	-39,1	-37,3	-6,0	-8,1	-18,4	-12,8	-9,1	-5,8	-7,0	-1,1	3,1	2,6
j20 m10	-5,1	-13,0	-0,9	-4,2	-3,4	-8,1	-1,4	-2,2	4,7	-2,7	1,8	5,0
j20 m20	-1,5	-3,5	-1,5	-1,3	-3,5	-4,0	-1,9	0,2	0,7	-1,8	1,4	-0,8
j50 m5	-48,3	-65,9	-18,6	-18,7	-13,5	-21,5	-6,0	-8,1	0,1	-12,7	0,2	-1,4
j50 m10	-10,4	-12,5	-3,2	-6,2	-9,0	-7,8	-0,9	-4,8	4,4	3,1	2,7	0,4
j50 m20	-2,4	-3,7	-2,3	-2,1	-3,3	-3,5	-3,3	-1,6	2,7	3,8	-0,5	1,3
j100 m5	-56,3	-93,8	-17,0	-26,5	-15,5	-34,9	-6,3	-10,0	-1,1	-8,7	0,4	2,3
j100 m10	-12,2	-15,7	-5,2	-6,9	-6,7	-6,0	-4,2	-7,6	4,7	7,1	2,7	3,4
j100 m20	-3,5	-6,4	-3,5	-3,7	-4,3	-3,6	-3,2	-3,1	2,1	1,9	0,8	2,0
j200 m10	-13,1	-18,3	-6,4	-8,6	-7,8	-9,2	-4,4	-4,1	0,0	3,8	3,9	3,1
j200 m20	-6,0	-7,3	-3,6	-3,9	-4,1	-4,6	-2,5	-2,3	1,6	1,7	2,0	1,3
j500 m20	-6,0	-7,4	-5,1	-6,0	-4,1	-4,6	-3,0	-3,4	2,0	1,1	0,4	0,8
Média	-17,0	-23,7	-6,1	-8,0	-7,8	-10,1	-3,8	-4,4	1,2	-0,4	1,6	1,7

Tabela 6.10 – Desvio relativo (%) das metaheurísticas por cenário com política de manutenção preventiva 3

	NEH + IG [1]	NEH + IG [2]	NEH + IG [3]
j20 m5	-22,7%	-11,5%	-0,6%
j20 m10	-5,8%	-3,8%	2,2%
j20 m20	-1,9%	-2,3%	-0,1%
j50 m5	-37,9%	-12,3%	-3,4%
j50 m10	-8,1%	-5,6%	2,7%
j50 m20	-2,6%	-2,9%	1,8%
j100 m5	-48,4%	-16,7%	-1,8%
j100 m10	-10,0%	-6,1%	4,5%
j100 m20	-4,3%	-3,6%	1,7%
j200 m10	-11,6%	-6,4%	2,7%
j200 m20	-5,2%	-3,4%	1,6%
j500 m20	-6,1%	-3,8%	1,1%
Média	-13,7%	-6,5%	1,0%

Tabela 6.11 – Desvio relativo das metaheurísticas com política de manutenção preventiva 3

Figura 6.11 – Gráficos dos resultados das metaheurísticas por cenário com política de manutenção preventiva 3

Figura 6.12 – Gráficos dos resultados das metaheurísticas com política de manutenção preventiva 3

segundos	EDD + NEH + LS1 + LS2	NEH + IG [1]	NEH + IG [2]	NEH + IG [3]
j20 m5	0,47	0,39	0,24	0,64
j20 m10	0,48	0,40	0,25	0,65
j20 m20	0,50	0,44	0,26	0,65
j50 m5	1,84	1,25	1,24	7,37
j50 m10	1,73	1,24	1,15	7,69
j50 m20	1,75	1,31	1,18	7,61
j100 m5	10,88	6,95	7,82	59,30
j100 m10	10,83	6,91	7,67	61,13
j100 m20	10,85	7,54	7,77	63,49
j200 m10	84,83	53,59	61,29	479,32
j200 m20	89,02	57,38	61,78	476,84
j500 m20	1.355,76	870,67	948,45	7.405,59

Tabela 6.12 – Tempos de execução das metaheurísticas com política de manutenção preventiva 3

Figura 6.13 – Gráficos dos tempos de execução das metheurísticas com política de manutenção preventiva 3

6.5 COMPARAÇÃO DAS HEURÍSTICAS CONSTRUTIVAS E METAHEURÍSTICA

O algoritmo com melhor desempenho a partir de 50 tarefas é o NEH + IG[3], como pode-se observar na aplicação do desvio relativo em relação ao melhor método. As tabelas a

seguir demonstram o valor do desvio consolidado dos cenários e detalhados por política e métodos, Tabelas 6.13, 6.14, 6.15 e 6.16.

Problema	EDD + NEH + LS1 + LS2	NEH + IG[1]	NEH + IG[2]	NEH + IG[3]
j20 m5	2,4%	25,8%	14,3%	6,4%
j20 m10	1,7%	10,5%	5,9%	2,4%
j20 m20	0,9%	2,0%	2,3%	1,5%
j50 m5	11,6%	85,6%	30,6%	2,7%
j50 m10	3,5%	13,2%	11,0%	1,0%
j50 m20	2,2%	5,7%	5,2%	0,9%
j100 m5	23,6%	218,3%	75,0%	1,3%
j100 m10	4,4%	17,6%	13,0%	1,0%
j100 m20	2,2%	6,6%	6,0%	0,4%
j200 m10	4,4%	19,1%	13,3%	0,2%
j200 m20	2,5%	7,8%	6,5%	0,1%
j500 m20	1,8%	8,8%	6,2%	0,0%
Média	5,1%	35,1%	15,8%	1,5%

Tabela 6.13 – Desvio relativo (em relação ao melhor) da heurística e metaheurísticas sem política de manutenção preventiva

Problema	EDD + NEH + LS1 + LS2	NEH + IG[1]	NEH + IG[2]	NEH + IG[3]
j20 m5	4,5%	29,2%	17,4%	4,9%
j20 m10	3,6%	7,8%	7,3%	2,6%
j20 m20	2,2%	4,5%	4,3%	1,2%
j50 m5	7,4%	44,0%	19,0%	2,8%
j50 m10	5,6%	14,3%	10,9%	1,2%
j50 m20	2,3%	6,3%	4,9%	1,3%
j100 m5	2,9%	53,0%	21,7%	4,0%
j100 m10	3,7%	15,5%	10,7%	0,5%
j100 m20	2,9%	7,5%	6,3%	0,3%
j200 m10	3,5%	15,6%	9,8%	0,1%
j200 m20	2,3%	6,8%	5,8%	0,1%
j500 m20	1,3%	7,2%	5,1%	0,1%
Média	3,5%	17,6%	10,3%	1,6%

Tabela 6.14 – Desvio relativo (em relação ao melhor) da heurística e metaheurísticas com política de manutenção preventiva 1

Problema	EDD + NEH + LS1 + LS2	NEH + IG[1]	NEH + IG[2]	NEH + IG[3]
j20 m5	3,8%	24,2%	13,0%	6,5%
j20 m10	3,4%	8,1%	8,9%	2,1%
j20 m20	2,3%	3,9%	4,5%	1,4%
j50 m5	4,8%	44,2%	18,9%	5,9%
j50 m10	5,4%	15,6%	12,5%	1,0%
j50 m20	1,8%	5,7%	5,4%	0,8%
j100 m5	7,0%	50,8%	20,2%	4,3%
j100 m10	4,9%	15,0%	11,0%	0,8%
j100 m20	2,6%	7,4%	6,2%	0,3%
j200 m10	3,4%	15,6%	9,5%	0,5%
j200 m20	2,2%	7,4%	5,6%	0,3%
j500 m20	1,4%	7,3%	4,9%	0,1%
Média	3,6%	17,1%	10,0%	2,0%

Tabela 6.15 – Desvio relativo (em relação ao melhor) da heurística e metaheurísticas com política de manutenção preventiva 2

Problema	EDD + NEH + LS1 + LS2	NEH + IG[1]	NEH + IG[2]	NEH + IG[3]
j20 m5	6,9%	30,6%	18,1%	6,2%
j20 m10	4,9%	10,9%	8,7%	2,1%
j20 m20	2,2%	4,2%	4,5%	2,3%
j50 m5	4,1%	43,6%	16,7%	7,0%
j50 m10	4,8%	13,1%	10,6%	1,7%
j50 m20	2,9%	5,6%	5,8%	0,9%
j100 m5	3,7%	53,8%	20,9%	5,2%
j100 m10	5,2%	15,8%	11,6%	0,3%
j100 m20	2,4%	6,7%	6,0%	0,6%
j200 m10	3,4%	15,3%	10,0%	0,5%
j200 m20	1,9%	7,2%	5,3%	0,2%
j500 m20	1,3%	7,5%	5,1%	0,2%
Média	3,6%	17,9%	10,3%	2,3%

Tabela 6.16 – Desvio relativo (em relação ao melhor) da heurística e metaheurísticas com política de manutenção preventiva 3

7. CONCLUSÃO

Resolver problemas de *flowshop no-wait* com manutenção preventiva e com o objetivo de minimizar o *total tardiness* tem sido o desafio de várias indústrias, uma vez que siderurgias, indústrias alimentícias, indústrias de remédios e outras precisam resolver esse problema para melhorarem sua eficiência operacional.

Essa pesquisa propôs heurísticas para resolver o problema estudado. Dentre as heurísticas construtivas propostas, a intitulada EDD + NEH + LS1 + LS2 foi a que obteve melhor resultado. O desempenho foi alcançado devido aos mecanismos de inserção que o algoritmo executa. A inserção permite obter novas sequências, porém preserva a posição relativa das tarefas, com exceção da tarefa que será inserida. Dessa forma, não há uma destruição total da sequência anterior gerada. A implicação que existe é que a solução inicial deve ser a melhor possível, pois isso afetará o desempenho de todo o algoritmo. A solução inicial utilizada (EDD) demonstra ser uma boa solução para problemas que desejam minimizar o *total tardiness*, uma vez que considera a data de entrega. Ou seja, tarefas com datas de entrega mais recentes são programadas primeiro.

Além da heurística construtiva, foi proposta uma metaheurística, já que não havia referências para o problema estudado. Dessa forma, a melhor heurística construtiva foi comparada com a heurística NEH + IG. Os resultados demonstram que dependendo das configurações da metaheurística, essa pode ser superior ao EDD + NEH + LS1 + LS2. Apesar do aumento do tempo computacional, o benefício na redução do *total tardiness* é aceitável. Notou-se que a remoção de uma quantidade menor de tarefas afeta positivamente o desempenho do NEH + IG. Dessa forma, pressupõem-se que a remoção de uma quantidade grande de tarefas destrói muito a ordenação previamente realizada, e com isso não há melhora no *total tardiness*. Outro fator diretamente positivo é a quantidade de *loops* de inserções que serão executados. Já era esperado essa melhora diretamente proporcional, pois mais *loops* significam mais soluções geradas, e assim maior a probabilidade de uma sequência com menor *total tardiness* ser identificada.

Em relação as políticas de manutenção preventiva não foi encontrado nenhum comportamento específico, que indique alguma correlação entre os resultados e as políticas de manutenção preventiva. A forma que a manutenção preventiva foi considerada no algoritmo contribuiu para esse resultado. Ou seja, a manutenção preventiva foi programada depois que a sequência final foi identificada. Caso fosse considerada a restrição de manutenção preventiva

na ordenação das tarefas desde a concepção da solução inicial, a correlação entre políticas de manutenção e resultado do *total tardiness* poderia ser mais clara.

Por fim, sugere-se para futuras pesquisas incluir na concepção das heurísticas a variável de manutenção preventiva, além da elaboração de outras metaheurísticas, sendo uma sugestão utilizar a heurística EDD + NEH + LS1 + LS2 como solução inicial.

REFERÊNCIAS BIBLIOGRÁFICAS

ACKOFF, R. L; SASIENI, M. W. **Pesquisa operacional**, Livros técnicos e científicos editora s.a., Rio de Janeiro/RJ, 1975.

ALDOWAISAN, T. A.; ALLAHVERDI, A. No-wait flowshop scheduling problem to minimize the number of tardy jobs. **International Journal of Advanced Manufacturing Technology**, vol. 61, pp. 311-323, 2012a.

ALDOWAISAN, T.; ALLAHVERDI, A. Minimizing total tardiness in no-wait flowshops. **Foundations of Computing and Decision Sciences**, vol. 37, n.3, pp. 149-162, 2012b.

ALDOWAISAN, T.; ALLAHVERDI, A. No-wait flowshops to minimize total tardiness with setup times. **Intelligent Control and Automation**, vol. 6, pp. 38-44, 2015.

ALDOWAISAN, T.; ALLAHVERDI, A. Total flowtime in no-wait flowshops with separated setup times. **Computers & Operations Research**, vol. 25, pp. 757-765, 1998.

ALLAHVERDI, A. A survey of scheduling problems with no-wait in process. **European Journal of Operational Research**, 2016.

ALLAHVERDI, A. et al. A survey of scheduling problems with setup times or costs. **European Journal of Operational Research**, vol. 187, pp. 985-1032, 2008.

ALLAHVERDI, A.; ALDOWAISAN, T. No-wait flowshops with bicriteria of makespan and maximum lateness. **European Journal of Operational Research**, vol. 152, pp. 132-147, 2004.

ALLAHVERDI, A.; AYDILEK, H.; AYDILEK, A. No-wait flowshop scheduling problem with two criteria: total tardiness and makespan. **European Journal of Operational Research**, vol. 269, pp. 590-601, 2018.

ARABAMERI, S.; SALMASI, N. Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem. **Computers & Industrial Engineering**, vol. 64, pp. 902-916, 2013.

ASEFI, H. et al. A hybrid NSGA-II and VNS for solving a bi-objective no-wait flexible flowshop scheduling problem. **International Journal of Advanced Manufacturing Technology**, 2014.

BAKER, K. P. **Introduction to sequencing and scheduling**. New York: John Wiley, 1974.

CASSADY, C. R.; KUTANOGLU, E. Minimizing job tardiness using integrated preventive maintenance planning and production scheduling. **IEEE Transaction**, vol. 35, pp. 503-513, 2003.

CHENG, G. Q.; ZHOU, B. H.; LI, L. Joint optimization of lot sizing and condition-based maintenance for multi-component production systems. **Computers & Industrial Engineering**, vol. 110, pp. 538-549, 2017.

CHENG, T. C. E.; LIU, Z. $3/2$ -approximation for two-machine no-wait flowshop scheduling with availability constraints. **Information Processing Letters**, vol. 88, pp. 161-165, 2003b.

CHENG, T. C. E.; LIU, Z. Approximability of two-machine no-wait flowshop scheduling with availability constraints. **Operations Research Letters**, vol. 31, pp. 319-322, 2003a.

CONFORTO, E. C.; AMARAL, D. C.; SILVA, S. L. Roteiro para revisão bibliográfica sistemática: aplicação no desenvolvimento de produtos e gerenciamento de projetos. **In: Congresso Brasileiro de Gestão de Desenvolvimento de Produtos, 2011, Porto Alegre. Congresso Brasileiro de Gestão de Desenvolvimento de Produtos, 2011.**

CONWAY, R. W.; MAXWELL, W. L.; MILLER, L. W. **Theory of scheduling**. Addison-Wesley, 1967.

DING, J. et al. Accelerated methods for total tardiness minimisation in no-wait flowshops. **International Journal of Production Research**, vol. 53, n. 4, pp. 1002-1018, 2015.

ESPINOUSE, M. L.; FORMANOWLCZ, P.; PENZ, B. Complexity results and approximation algorithms for the two-machine no-wait flow-shop with limited machine availability. **Journal of the Operational Research Society**, vol. 52, pp. 116-121, 2001.

ESPINOUSE, M. L.; FORMANOWLCZ, P.; PENZ, B. Minimizing the makespan in the two-machine no-wait flow-shop with limited machine availability. **Computers & Industrial Engineering**, vol. 37, pp. 497-500, 1999.

FRAMINAN, J. M.; GUPTA, J. N. D.; LEISTEN, R. A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. **Journal of Operational Research Society**, vol. 55, pp. 1243-1255, 2004.

FRAMINAN, J. M.; LEISTEN, R.; RUIZ-USANO, R. Comparison of heuristics for flowtime minimization in permutation flowshop. **Computers & Operation Research**, n. 32, pp. 1237-1254, 2005.

FUSCO, J. P. A. Gestão de operações na prática: as empresas e os desafios de contexto concorrencial. **Produção**, vol. 9, n. 2, pp. 19-36, 2010.

GONG, H.; TANG, T. Batch Delivery Scheduling with Limited Waiting Time Constraint on a Single Machine. **Chinese Control and Decision Conference**, pp. 2755-2759, 2009.

GRAHAN, R. L.; LAWLER, E. L.; LENSTRA, J. K.; KAN, A. H. G. R. Optimisation and approximation in deterministic sequencing and scheduling: a survey. **Annals of Discrete Mathematics**, vol. 5, pp. 287-326, 1979.

HASENBEIN, J. J.; KUTANOGLU, E.; LIAO, M. Single-machine multiple-recipe predictive maintenance. **Probability in the Engineering and Informational Sciences**, vol. 27, pp. 209-235, 2013.

JANUARIO, T. O. et al. Genetic Local Search Algorithm for the Minimum Total Tardiness Permutation Flowshop Problem. **The 11th IEEE International Conference on Computational Science and Engineering - Workshops**, 2008.

JANUARIO, T. O.; ARROYO, J. E. C.; MOREIRA, M. C. O. Genetic Algorithm for Tardiness Minimization in Flowshop with Blocking. **Springer-Verlag Berlin Heidelberg**, vol. 236, pp. 153-164, 2009.

JARBOUI, B.; EDDALY, M.; SIARRY, P. A hybrid genetic algorithm for solving no-wait flowshop scheduling problems. **International Journal of Advanced Manufacturing Technology**, vol. 54, pp. 1129-1143, 2011.

KHALILI, M. A multi-objective electromagnetism algorithm for a bi-objective hybrid no-wait flowshop scheduling problem. **International Journal of Advanced Manufacturing Technology**, vol. 70, pp. 1591-1601, 2014.

KUBZIN, M. A., STRUSEVICH, V. A. Two-machine flow shop no-wait scheduling with a nonavailability interval. **Naval Research Logistics**, v. 51, pp. 613-631, 2004.

LAHA, D.; CHAKRABORTY, U. K. A constructive heuristic for minimizing makespan in no-wait flow shop scheduling. **The International Journal of Advanced Manufacturing Technology**, vol. 41, pp. 97–109, 2009.

LAHA, D.; GRUPTA, J. N. D.; SAPKAL, S. U. A penalty-shift-insertion-based algorithm to minimize total flow time in no-wait flow shops. **Journal of the Operational Research Society**, vol. 64, n.10, pp. 1611–1624, 2014.

LAHA, D.; SAPKAL, S. U. An improved heuristic to minimize total flow time for scheduling in the m-machine no-wait flow shop. **Computers & Industrial Engineering**, vol. 67, pp. 36-43, 2014.

LIN, S. W.; LU, C. C.; YING, K. C. Minimizing the sum of makespan and total weighted tardiness in a no-wait flowshop. **IEEE ACCESS**, vol. 6, pp. 78666-78677, 2018.

LIN, S. W.; YING, K. C. Minimizing makespan for solving the distributed no-wait flowshop scheduling problem. **Computers & Industrial Engineering**, vol. 99, pp. 202-208, 2016.

LIU, G.; SONG, S.; WU, C. Some heuristics for no-wait flowshops with total tardiness criterion. **Computers and Operations Research**, vol. 40, n. 2, pp. 521-525, 2013.

MA, Y.; CHU, C.; ZUO, C. A survey of scheduling with deterministic machine availability constraints. **Computers & Industrial Engineering**, vol. 58, pp. 199-211, 2010.

MACCARTHY, B. L.; LIU, J. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. **International Journal of Production Research**, vol. 31, n.1, pp. 59-79, 1993.

MACCHIAROLI, R.; MOLE, S.; RIEMMA, S. Modelling and optimization of industrial manufacturing processes subject to no-wait constraints. **International Journal of Production Research**, vol. 37, n.11, pp. 2585-2607, 1999.

MAO, K. et al. A novel Lagrangian relaxation approach for a hybrid flowshop scheduling problem in the steelmaking-continuous casting process. **European Journal of Operational Research**, vol. 23, pp. 651-60, 2014.

MIYATA, H. H.; NAGANO, M. S.; GUPTA, J. N. D. Incorporating preventive maintenance into the m-machine no-wait flow-shop scheduling problem with total flow-time minimization: a computational study. **Engineering Optimization**, vol. 51, n. 4, pp. 680-698, 2019.

MOCCELLIN, J. V.; NAGANO, M. S. Uma propriedade estrutural do problema de programação da produção flow shop permutacional. **Pesquisa Operacional**, vol. 27, n.3, pp.487-515, 2007.

MOKHTARI, H.; MOZDGIR, A.; ABADI, I. N. K. A reliability/availability approach to joint production and maintenance scheduling with multiple preventive maintenance services. **International Journal of Production Research**, vol. 50, n. 20, pp. 5906-5925, 2012.

MORABITO, R.; ARENALES, M.; ARMENTANO, V. A.; YANASSE, H. H. **Pesquisa operacional** – para cursos de engenharia, 2. ed., Elsevier, Rio de Janeiro, 2015.

NADERI, B. et al. Multi-objective no-wait flowshop scheduling problems: models and algorithms. **International Journal of Production Research**, vol. 50, n. 10, pp. 2592-2608, 2012.

NADERI, B.; ARSHADI, A. Scheduling a variant of flowshop problems to minimize total tardiness. **Technical Journal of Engineering and Applied Sciences**, vol. 3, n.22, pp. 3142-3149, 2013.

NADERI, B.; KHALILI, M.; KHAMSEH, A. A. Mathematical models and a hunting search algorithm for the no-wait flowshop scheduling with parallel machines. **International Journal of Production Research**, vol. 52, n. 9, pp. 2667-2681, 2014.

NG, C. T.; KOVALYOV, M. Y. An FPTAS for scheduling a two-machine flowshop with one unavailability interval. **Naval Research Logistics**, vol. 51, pp. 307-315, 2004.

PAN, Q. et al. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. **Information Sciences**, vol. 181, pp. 2455-2468, 2011a.

PAN, Q. et al. A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem. **Expert Systems with Applications**, vol. 38, pp. 3252-3259, 2011b.

PAN, Q.; WANG, L.; QIAN, B. A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. **Computers & Operations Research**, vol. 36, pp. 2498-2511, 2009.

PATIL, S. K.; KANT, R. Methodological literature review of knowledge management research. **TÉKHNE - Review of Applied Management Studies**, vol.12, pp. 3-14, 2014.

PEREZ-GONZALEZ, P.; FRAMINAN, J. M. Setting a common due date in a constrained flowshop: A variable neighbourhood search approach. **Computers & Operations Research**, vol. 37, pp. 1740-1748, 2010.

PINEDO, M. L. **Scheduling: Theory, algorithms and systems**, 3. Ed., Springer, New York, 2008.

REDDI, S. S.; RAMAMOORTHY, C. V. On the flow-shop sequencing problem with no wait process. **Operational Research Quarterly**, vol. 72, pp. 323-331, 1972.

RIBAS, I.; COMPANYS, R.; TORT-MARTORELL, X. An efficient iterated local search algorithm for the total tardiness blocking flow shop problem. **International Journal of Production Research**, vol. 51, n. 17, pp. 5238-5252, 2013.

RONCONI, D. P. A note on constructive heuristics for the flowshop problem with blocking. **Int. J. Production Economics**, vol. 87, pp. 39-48, 2004.

RONCONI, D. P.; ARMENTANO, V. A. Lower bounding schemes for flowshops with blocking in-process. **Journal of the Operational Research Society**, vol. 52, pp. 1289-1297, 2001.

RONCONI, D. P.; HENRIQUES, L. R. S. Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. **Omega**, vol. 37, pp. 272-281, 2009.

RUIZ, R.; ALLAHVERDI, A. No-wait flowshop with separate setup times to minimize maximum lateness. **The International Journal of Advanced Manufacturing Technology**, vol. 35, pp. 551-565, 2006.

RUIZ, R.; DÍAZ, C. G.; MATORO, C. Considering scheduling and preventive maintenance in the flowshop sequencing problem. **Computers and Operations Research**, vol. 34, pp. 3314-3330, 2007.

RUIZ, R.; STUTZLE, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. **European Journal of Operational Research**, vol. 177, pp. 2033-2049, 2007.

RUIZ, R; ALLAHVERDI, A. Some effective heuristics for no-wait flowshops with setup times to minimize total completion time. **Annals of Operations Research**, vol. 156, pp. 143-171, 2007.

SAMARGHANDI, H. A particle swarm optimisation for the no-wait flow shop problem with due date constraints. **International Journal of Production Research**, vol. 53, n. 9, pp. 2853-2870, 2015.

SAMARGHANDI, H.; BEHROOZI, M. An enumeration algorithm for the no-wait flow shop problem with due date constraints. **IFAC-PapersOnLine**, vol. 49, n. 12, pp. 1803-1808, 2016.

SAMARGHANDI, H.; BEHROOZI, M. On the exact solution of the no-wait flow shop problem with due date constraints. **Computers and Operations Research**, vol. 81, pp. 141-159, 2017.

SAMARGHANDI, H.; BEHROOZI, M. On the exact solution of the no-wait flow shop problem with due date constraints. **Computers and Operations Research**, vol. 81, pp. 141-159, 2017.

SAPKAL, S. U.; LAHA, D. A heuristic for no-wait flow shop scheduling. **The International Journal of Advanced Manufacturing Technology**, vol. 68, n. 12, pp. 1327-1338, 2013.

SHEIKH, S. Multi-objective flexible flow lines with due window, time lag, and job rejection. **International Journal of Advanced Manufacturing Technology**, vol. 64, pp. 1423-1433, 2013.

TASGETIREN, M. F. et al. A differential evolution algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. **International Journal of Production Research**, vol. 49, n. 16, pp. 5033-5050, 2011.

TASGETIREN, M. F. et al. A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion. **Applied Mathematical Modelling**, vol. 37, pp. 6758-6779, 2013.

TAVAKKOLI-MOGHADDAM, R.; RAHIMI-VAHED, A. R.; MIRZAEI, A. H. **International Journal of Advanced Manufacturing Technology**, vol. 36, pp. 969-981, 2008.

TAVAKKOLI-MOGHADDAM, R.; RAHIMI-VAHED, A.; MIRZAEI, A. H. hybrid multi-objective immune algorithm for a flowshop scheduling problem with bi-objectives: Weighted mean completion time and weighted mean tardiness. **Information Sciences**, vol. 177, pp. 5072-5090, 2007.

WANG, C.; LI, X; WANG, Q. Accelerated tabu search for no-wait flowshop scheduling problem with maximum lateness criterion. **European Journal of Operational Research**, vol. 206, pp. 64-72, 2010.

WANG, G.; CHENG, T. C. E. Heuristics for two-machine no-wait flowshop scheduling with an availability constraint. **Information Processing Letters**, v. 80, pp. 305-309, 2001.

YING, K. et al. Metaheuristics for scheduling a no-wait flowshop manufacturing cell with sequence-dependent family setups. **International Journal of Advanced Manufacturing Technology**, vol. 58, pp. 671-682, 2012.