FERNANDO LUIS ROSSI

MÉTODOS HEURÍSTICOS PARA MINIMIZAÇÃO DA DURAÇÃO TOTAL DA
PROGRAMAÇÃO E DO TEMPO TOTAL DE FLUXO EM AMBIENTES *FLOW SHOP*
PERMUTACIONAL

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: Dr. Marcelo Seido Nagano

SÃO CARLOS

2014

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.

R831m Rossi, Fernando
Métodos Heurísticos para minimização da duração
total da programação e do tempo total de fluxo em
ambientes flow shop permutacional / Fernando Rossi;
orientador Marcelo Nagano. São Carlos, 2014.

Dissertação (Mestrado) - Programa de Pós-Graduação
em Engenharia de Produção e Área de Concentração em
Processos e Gestão de Operações -- Escola de Engenharia
de São Carlos da Universidade de São Paulo, 2014.

1. programação da produção. 2. flow shop. 3.
métodos heurísticos. 4. makespan. 5. total flowtime. I.
Título.

FOLHA DE JULGAMENTO

Candidato: Bacharel **FERNANDO LUIS ROSSI**.

Título da dissertação: "Métodos heurísticos para minimização da duração total da programação e do tempo total de fluxo em ambientes flow shop permutacional".

Data da defesa: 26/01/2015

Comissão Julgadora:

Prof. Dr. **Marcelo Seido Nagano (Orientador)**
(Escola de Engenharia de São Carlos/EESC)

Prof. Dr. **Walther Azzolini Júnior**
(Escola de Engenharia de São Carlos/EESC)

Prof. Dr. **Roberto Fernandes Tavares Neto**
(Universidade Federal de São Carlos)

Resultado:

Aprovado

Aprovado

Aprovado

Coordenadora do Programa de Pós-Graduação em Engenharia de Produção:
Profa. Associada **Daisy Aparecida do Nascimento Rebelatto**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Paulo César Lima Segantine**

AGRADECIMENTOS

A Deus.

Ao Prof. Dr. Marcelo Seido Nagano pela orientação, pelas importantes contribuições e pela confiança.

Aos meus pais Leodir e Marine que sempre me apoiaram e me incentivaram a estudar.

Ao Prof. Dr. Roberto Fernandes Tavares Neto e ao Prof. Dr. Walter Azzolini Junior pelas suas importantes contribuições para a condução deste trabalho.

Aos docentes e funcionários do Departamento de Engenharia de Produção da EESC – USP.

Resumo

ROSSI, F. L. **Métodos Heurísticos para minimização da duração total da programação e do tempo total de fluxo em ambientes *flow shop* permutacional**, 2014, 124p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2014.

Neste trabalho é abordado o problema da programação da produção em ambiente *flow shop* permutacional, tendo como objetivo analisar dois critérios de otimização: o primeiro critério é a minimização do tempo total de programação da produção (*makespan*), e o segundo é a minimização do tempo total de fluxo (*total flowtime*). Estes objetivos são relevantes para a dinâmica do ambiente de produção, porque enquanto que a minimização do *makespan* tende a melhorar a utilização dos recursos produtivos, a minimização do *total flowtime* conduz a um menor estoque em processo. Por este motivo, estes problemas tem atraído a atenção de vários pesquisadores ao longo dos anos. Neste sentido, será realizado uma revisão do estado da arte e novas heurísticas construtivas serão propostas para gerarem soluções de qualidade para ambos os problemas estudados. Com este objetivo, uma extensa experimentação computacional foi realizada para fins de comparação e avaliação dos métodos propostos com os melhores métodos heurísticos reportados da literatura.

Palavras-chave: programação da produção, *flow shop*, métodos heurísticos, *makespan*, *total flowtime*.

Abstract

ROSSI, F, L. **Heuristics Methods for the makespan and total flowtime minimization in flow shop scheduling environment**, 2014, 124p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2014.

This work addresses the scheduling problem in permutation flow shop environment, having as objective to analyze two optimization criteria: the first criterion is to minimize the makespan, and the second is to minimize the total flow time. These objectives are relevant to the dynamics of the production environment, because while minimizing the makespan tends to improve the resources utilization, minimizing the total flowtime leads to a lower in-process inventory. For this reason, these problems have attracted the attention of many researchers over the years. In this regard, will be performed a review of the state of the art and new constructive heuristics will be proposed to generate quality solutions for both problems in consideration. For this purpose, an extensive computational experiment was performed for comparison and evaluation of the proposed heuristics with the best heuristic methods reported in the literature.

Keywords : Scheduling , flow shop , heuristic methods , makespan , total *flowtime* .

LISTA DE FIGURAS

<i>Figura 1 - PCP e as suas atividades.</i>	22
<i>Figura 2 - Relação hierárquica entre os ambientes de produção. Fonte: adaptado de MacCarthy e Liu (1993).</i>	23
<i>Figura 3 – Estrutura do Trabalho.</i>	26
<i>Figura 4 - Exemplo de um Diagrama de Gantt. Fonte: Autor, 2014.</i>	29
<i>Figura 5 - Ilustração do método: inserção da tarefa 5 na 3a posição. Fonte: adaptado de Taillard, 1990.</i>	34
<i>Figura 6 – Heurística NEH.</i>	49
<i>Figura 7 – Heurística NEH NM.</i>	50
<i>Figura 8 – Heurística NEH KK.</i>	50
<i>Figura 9 – Heurística NEH KK1.</i>	51
<i>Figura 10 – Heurística NEH D.</i>	52
<i>Figura 11 – Heurística NEH KK2.</i>	52
<i>Figura 12 – Heurística FRB3.</i>	53
<i>Figura 13 – Heurística INEH KK2D.</i>	54
<i>Figura 14 - Procedimento NEH FF.</i>	55
<i>Figura 15 – Pseudocódigo da heurística IG_RS. Fonte: adaptado de Ruiz e Stützle, 2007....</i>	56
<i>Figura 16 - Ilustração do procedimento IG_RS.</i>	57
<i>Figura 17 - Mecanismo de destruição e reconstrução da heurística G8.</i>	59
<i>Figura 18 – Pseudocódigo da heurística G8.</i>	60
<i>Figura 19 - DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes do Taillard.</i>	65
<i>Figura 20 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes do Taillard.</i>	66
<i>Figura 21 – Gráfico com os intervalos das médias das heurísticas com nível de confiança de 95%.</i>	67
<i>Figura 22 - Tempo computacional médio e TRPM das heurística nos problemas testes do Taillard.</i>	69
<i>Figura 23 – DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes do Taillard.</i>	72
<i>Figura 24 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes do Taillard.</i>	73

<i>Figura 25 – Gráfico com o intervalo das médias das novas heurísticas propostas com um nível de confiança de 95%.....</i>	<i>75</i>
<i>Figura 26 - Tempo computacional médio e TRPM das heurísticas propostas nos problemas testes do Taillard.</i>	<i>76</i>
<i>Figura 27 – Gráfico do intervalo das médias das heurísticas comparadas nos problemas testes do Taillard com um grau de confiança de 95%.</i>	<i>76</i>
<i>Figura 28 – DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes de pequeno porte do VFR.</i>	<i>79</i>
<i>Figura 29 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes de pequeno porte do VFR.....</i>	<i>79</i>
<i>Figura 30 - Gráfico com os intervalos das médias das heurísticas nos problemas testes de pequeno porte do VFR com nível de confiança de 95%.....</i>	<i>80</i>
<i>Figura 31 - Tempo computacional médio e TRPM das heurísticas no conjunto de problemas testes VFR de pequeno porte.</i>	<i>81</i>
<i>Figura 32 - DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes de grande porte do VFR.</i>	<i>83</i>
<i>Figura 33 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes de grande porte do VFR.....</i>	<i>84</i>
<i>Figura 34 – Gráfico com os intervalos de média das heurísticas para os problemas de grande porte do VFR com nível de confiança de 95%.</i>	<i>84</i>
<i>Figura 35 - Tempo computacional médio e TRPM das heurísticas nos problemas testes do VFR de grande porte.</i>	<i>85</i>
<i>Figura 36 – Classificação das heurísticas.</i>	<i>90</i>
<i>Figura 37 – Heurística FL.</i>	<i>102</i>
<i>Figura 38 – Heurística FL-LS.....</i>	<i>103</i>
<i>Figura 39 – Heurística LR-NEH(x). Fonte: adaptado de Pan e Ruiz, 2013.....</i>	<i>104</i>
<i>Figura 40 – Heurística FF-NEH(x). Fonte: adaptado de Fernandez-Viagas e Framinan (2015).</i>	<i>105</i>
<i>Figura 41 – Heurística FF-RN(x, y)</i>	<i>108</i>
<i>Figura 42 – Exemplo ilustrativo do funcionamento da heurística FF-RN(x, y).</i>	<i>109</i>
<i>Figura 43 – DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes do Taillard.</i>	<i>112</i>
<i>Figura 44 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes do Taillard.....</i>	<i>113</i>

<i>Figura 45 – Gráfico com o intervalo das médias das heurísticas comparadas nos problemas testes do Taillard com um nível de confiança de 95%.</i>	<i>114</i>
<i>Figura 46 - Tempo computacional média e TRPM de cada método nos problemas testes do Taillard.</i>	<i>115</i>
<i>Figura 47 – DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes de pequeno porte do VFR.</i>	<i>118</i>
<i>Figura 48 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes de pequeno porte do VFR.....</i>	<i>119</i>
<i>Figura 49 – Gráfico com os intervalos das médias das heurísticas comparadas nos problemas testes de pequeno porte do VFR.</i>	<i>120</i>
<i>Figura 50 - Tempo médio computacional ou TRPM dos métodos comparados nos problemas testes de pequeno porte do VFR.</i>	<i>121</i>
<i>Figura 51 – DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes de grande porte do VFR</i>	<i>123</i>
<i>Figura 52 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes de grande porte do VFR.....</i>	<i>123</i>
<i>Figura 53 – Gráfico com os intervalos das médias das heurísticas com grau de confiança de 95% nos problemas testes de grande porte do VFR.....</i>	<i>124</i>
<i>Figura 54 - Tempo computacional médio e TRPM das heurísticas para os problemas de grande porte do VFR.</i>	<i>126</i>

LISTA DE QUADROS

<i>Quadro 1 - Variações da heurística NEH e suas composições (Parte 1).</i>	46
<i>Quadro 2 - Diferentes formas de seleção dos pares de tarefas.</i>	58
<i>Quadro 3 – Combinações entre mecanismos de desempate e ordenações iniciais.</i>	61
<i>Quadro 4 - Composição das fases das heurística selecionadas para avaliação.</i>	71
<i>Quadro 5 – Resumo da composição das heurísticas simples e compostas</i>	100

LISTA DE TABELAS

<i>Tabela 1 – DRM para cada classe de problemas e média geral obtidas pelos métodos heurísticos nos problemas testes do Taillard.</i>	<i>65</i>
<i>Tabela 2 – DRP médio obtido pelas heurísticas nos problemas testes de Taillard.</i>	<i>68</i>
<i>Tabela 3 - Eficiência computacional das heurísticas nos problemas testes do Taillard.</i>	<i>68</i>
<i>Tabela 4 – Heurísticas construtivas em ordem crescente de DRM nos problemas testes do Taillard.</i>	<i>70</i>
<i>Tabela 5 - DRM para cada classe de problemas e média geral obtidas pelos métodos heurísticos nos problemas testes do Taillard.</i>	<i>71</i>
<i>Tabela 6 - DRP médio obtido pelas novas heurísticas propostas nos problemas testes de Taillard.</i>	<i>74</i>
<i>Tabela 7 -Tempo computacional dos métodos propostos nos problemas testes do Taillard. ..</i>	<i>75</i>
<i>Tabela 8 - DRM para cada classe de problemas e média geral obtidas pelos novos métodos propostos nos problemas testes do VFR.</i>	<i>78</i>
<i>Tabela 9 – Eficiência computacional das heurísticas para os problemas testes de pequeno porte do VFR.</i>	<i>81</i>
<i>Tabela 10 – DRM para cada classe de problemas e média geral obtidas pelos novos métodos propostos nos problemas testes de grande porte do VFR.</i>	<i>82</i>
<i>Tabela 11 - Eficiência computacional médio para problemas de grande porte do VFR.</i>	<i>85</i>
<i>Tabela 12 – Resultados consolidados.</i>	<i>86</i>
<i>Tabela 13–Resultados obtidos nas experimentações computacionais de Pan e Ruiz (2013).</i>	<i>101</i>
<i>Tabela 14 – DRM para cada classe de problemas e média geral obtidas pelos novos métodos heurísticos nos problemas testes do Taillard.</i>	<i>111</i>
<i>Tabela 15 - Continuação da Tabela 14.</i>	<i>111</i>
<i>Tabela 16 - Tempo computacional das heurísticas nos problemas testes do Taillard.</i>	<i>114</i>
<i>Tabela 17 - Continuação da Tabela 16.</i>	<i>115</i>
<i>Tabela 18 – Impacto da geração da lista W' no desempenho da heurística FF-RN(x; y).</i>	<i>116</i>
<i>Tabela 19 – DRM para cada classe de problemas e média geral obtidas pelos novos métodos heurísticos nos problemas testes de pequeno porte do VFR.</i>	<i>117</i>
<i>Tabela 20 – Tempo médio computacional e TRPM das heurísticas nos problemas testes de pequeno porte do VFR.</i>	<i>121</i>
<i>Tabela 21 – DRM para cada classe de problemas e média geral obtidas pelos novos métodos heurísticos nos problemas testes de grande porte do VFR.</i>	<i>122</i>
<i>Tabela 22 - Eficiência das heurísticas para problemas de grande porte do VFR.</i>	<i>125</i>

SUMÁRIO

1. INTRODUÇÃO.....	21
1.1 PROGRAMAÇÃO DA PRODUÇÃO.....	21
1.2 MOTIVAÇÃO E JUSTIFICATIVA DA PESQUISA.....	25
1.3 OBJETIVO DA PESQUISA.....	25
1.4 ESTRUTURA DO TRABALHO.....	26
1.5 METODOLOGIA DE REVISÃO DA LITERATURA.....	27
2. PROBLEMA DE PROGRAMAÇÃO <i>FLOW SHOP</i>.....	28
3. PROBLEMA DE PROGRAMAÇÃO <i>FLOW SHOP</i> PERMUTACIONAL COM DE MINIMIZAÇÃO DO <i>MAKESPAN</i>	32
3.1 DEFINIÇÃO DO PROBLEMA.....	32
3.2 REVISÃO DA LITERATURA.....	34
3.2.1 Ordenação inicial.....	39
3.2.2 Geração da sequência	41
3.2.3 Mecanismo de desempate.....	42
3.3 HEURÍSTICAS CONSTRUTIVAS PARA O PROBLEMA $F prmu / C_{max}$	49
3.4 NOVOS MÉTODOS HEURÍSTICOS PARA O PROBLEMA $F prmu / C_{max}$	56
3.5 EXPERIMENTAÇÃO COMPUTACIONAL E ANÁLISE DOS RESULTADOS DO PROBLEMA $F prmu / C_{max}$	63
3.6 CONSIDERAÇÕES FINAIS DO PROBLEMA $F prmu / C_{max}$	87
4. PROBLEMA DE PROGRAMAÇÃO <i>FLOW SHOP</i> COM CRITÉRIO DE MINIMIZAÇÃO DO <i>TOTAL FLOWTIME</i>	88
4.1 DEFINIÇÃO DO PROBLEMA.....	88
4.2 REVISÃO DA LITERATURA.....	89
4.2.1 Heurísticas simples	90
4.2.2 Heurísticas compostas	94
4.3 HEURÍSTICAS SIMPLES PARA O PROBLEMA $F prmu / Ci$	101

4.4	NOVOS MÉTODOS HEURÍSTICOS PARA O PROBLEMA $F \mid prmu / \mathbf{Cj}$	106
4.5	EXPERIMENTAÇÃO COMPUTACIONAL E ANÁLISE DOS RESULTADOS PARA O PROBLEMA $F \mid prmu / \mathbf{Ci}$	110
4.6	CONSIDERAÇÕES FINAIS DO PROBLEMA $F \mid prmu / \mathbf{Ci}$	127
5.	CONSIDERAÇÕES FINAIS	128
	REFERÊNCIAS	130

1 INTRODUÇÃO

A programação da produção é uma decisão que desempenha um papel importante na indústria de manufatura e serviços. No atual ambiente competitivo, uma programação eficiente se tornou um critério de sobrevivência das empresas no mercado. Estas devem atender a prazos de entregas prometidos, e não atendê-los pode resultar em perdas significativas de clientela. Também se faz necessário programar as atividades de uma forma com que os recursos disponíveis sejam utilizados de maneira eficiente (PINEDO, 2008). A programação da produção é um processo de decisão que lida com a alocação de recursos às tarefas em um determinado período de tempo com a finalidade de otimizar um ou mais objetivos (PINEDO, 2008).

O ambiente de produção *flow shop* está presente em vários segmentos relevantes da indústria como, por exemplo, as indústrias metalúrgicas, de produtos químicos, e farmacêuticos. Em determinadas situações nas quais se objetiva melhorar a lucratividade ou por meio da maximização da utilização dos recursos ou por meio da redução do estoque em processo, pode-se recorrer, respectivamente, a métodos de programação da produção com critério de minimização da duração total da programação (*makespan*) ou do tempo total de fluxo (*total flowtime*) (PINEDO, 2008).

Nesse contexto, este projeto tem por objetivo desenvolver métodos heurísticos para auxiliar o processo da programação da produção. Será abordado, mais especificamente, o problema de programação da produção para minimização do *makespan* e do *total flowtime* em ambiente *flow shop* permutacional. Portanto, são dois problemas no mesmo ambiente de produção, mas cada um com um objetivo diferente a ser minimizado.

1.1 PROGRAMAÇÃO DA PRODUÇÃO

A gestão da produção é a atividade de gerenciamento de recursos escassos e processos que produzem e entregam bens e serviços, visando atender às necessidades do cliente. Para esse fim, o processo de gestão das operações de um sistema de produção é

composto por três funções centrais: marketing, desenvolvimento de produto/serviço e produção. O Planejamento e Controle da Produção (PCP) é a área responsável por gerir a função da produção dentro de uma empresa, administrando as atividades de operação produtiva de modo a satisfazer continuamente a demanda dos consumidores (SLACK, 2009).

As atividades de Planejamento e Controle da Produção envolvem uma série de decisões com o objetivo de definir o que, quando e quanto produzir, comprar e entregar, além de quem e/ou como produzir (SLACK, 2009). Essas decisões seguem uma estruturação hierárquica, na qual a programação da produção é um de seus componentes que lida com decisões de curto e médio prazo. A relação entre as áreas chaves de uma organização e as atividades do PCP é ilustrada pela Figura 1.

Figura 1 - PCP e as suas atividades.

Na programação da produção, um dos primeiros passos é definir qual o sistema produtivo empregado na produção dos produtos e/ou serviços. Com base no tipo de produto e no tipo de processo, Johnson e Montgomery (1974) classificam os sistemas de produção em: sistema contínuo, sistema grande projeto e sistema intermitente.

No sistema contínuo, poucos tipos de produtos similares são produzidos em grandes volumes. No sistema grande projeto, são feitos produtos muitas vezes únicos altamente complexos e variados. Já no sistema intermitente, nos estágios produtivos ocorrem mudanças de um produto para outro, como consequência da alta variedade de produtos. Dentro desta última categoria se enquadram os ambientes máquina única, máquinas em paralelo, *flow shop*,

job shop, *open shop*. A Figura 2 mostra a relação hierárquica entre os ambientes de produção proposta por MacCarthy e Liu (1993).

Figura 2 - Relação hierárquica entre os ambientes de produção. Fonte: adaptado de MacCarthy e Liu (1993).

Nesse sentido, vários métodos de programação foram desenvolvidos para este fim. Segundo MacCarthy e Liu (1993) um problema geral de programação da produção pode ser enunciado como n tarefas $\{J_1, J_2, \dots, J_n\}$ que têm de serem processadas em m máquinas $\{M_1, M_2, \dots, M_m\}$ disponíveis. Uma tarefa é definida por uma sequência de operações que devem ser executadas em uma dada ordem. Sendo que cada uma destas operações é realizada em uma máquina por um período de tempo.

É também geralmente assumido que cada máquina pode realizar apenas uma operação ao mesmo tempo. No caso mais geral, pode haver mais de uma máquina do mesmo tipo formando centros produtivos (PINEDO, 2008).

Um problema de programação da produção pode ser descrito pelo terceto $\alpha \mid \beta \mid \gamma$. O campo α descreve o ambiente de produção. O campo β fornece os detalhes das características de processamento e restrições. O campo γ descreve o objetivo a ser minimizado (GRAHAM *et al.*, 1979).

Alguns dos ambientes produtivos especificados pelo campo α são:

- i. Máquina Única (1): existe uma única máquina disponível para o processamento das tarefas;
- ii. Máquinas Paralelas Idênticas (P_m): existem m máquinas idênticas em paralelo. A tarefa j requer uma única operação e pode ser processada em qualquer uma das m máquinas;
- iii. *Job shop* (J_m): cada tarefa tem seu próprio roteiro de processamento nas máquinas;
- iv. *Flow shop* (F_m): todas as tarefas têm a mesma sequência de processamento nas máquinas;
- v. *Open Shop* (O_m): não há uma sequência específica ou preestabelecida de processamento das tarefas;
- vi. *Flexible job shop* (FJc): é um *job shop* no qual existe um conjunto de máquinas paralelas em cada estágio de produção;
- vii. *Flexible flow shop* (FFc): é um *flow shop* no qual existe um conjunto de máquinas paralelas em cada estágio de produção.

As restrições de processamento especificadas no campo β podem incluir múltiplas entradas. Algumas das entradas possíveis no campo β são:

- i. *Release dates* (r_j): a tarefa j não pode começar o processamento antes da data r_j ;
- ii. *Setup* dependente de sequência (s_{jk}): s_{jk} representa o tempo de *setup* entre o processamento das tarefas j e k ;
- iii. Permutação ($prmu$): esta restrição impõe que a ordem de processamento das tarefas na primeira máquina permaneça nas máquinas subsequentes;
- iv. *No-wait* (nwt): não se permite que as tarefas fiquem em espera entre máquinas sucessivas.

No campo γ , um usual objetivo a ser minimizado é sempre em função do tempo de conclusão das tarefas. O tempo de conclusão da tarefa j na máquina i é dado por C_{ij} , e C_j determina o tempo de conclusão da tarefa j . Baker (1974) observou três tipos de objetivos predominantes na programação da produção e apontou as medidas de desempenho mais comuns associadas a estes:

- i. *Makespan* (C_{max}): utilização eficiente dos recursos, definido como o $\max(C_1, \dots, C_n)$ e equivalente ao tempo de conclusão da última tarefa do sistema.
- ii. *Maximum Lateness* (L_{max}): conformidade com os prazos pré-estabelecidos, definido como o $\max(L_1, \dots, L_n)$ e mede a maior violação dos prazos de entregas;
- iii. *Total flowtime* ($\sum C_j$): resposta rápida a demanda, mede o tempo de fluxo ($\bar{F}$) e é definido como a soma dos tempos de conclusão das n tarefas. Fornece um indicador do estoques incorridos pela programação.

No presente trabalho, seguindo a notação $\alpha \mid \beta \mid \gamma$, serão estudados os problemas $F_m \mid prmu \mid C_{max}$ e $F_m \mid prmu \mid \sum C_j$, nesta ordem respectivamente.

1.2 MOTIVAÇÃO E JUSTIFICATIVA DA PESQUISA

A motivação para desenvolver a pesquisa proposta neste projeto é originária da investigação do problema proposto na literatura, em que se constatou que a programação é um componente importante no processo de administração da produção dentro das empresas, bem como no reconhecimento da oportunidade de colaborar com o corpo de conhecimento acadêmico acerca do assunto.

1.3 OBJETIVO DA PESQUISA

Alicerçado pela revisão da literatura sobre o problema objetivo de pesquisa, os objetivos propostos são:

- i. Estudar os problemas de programações da produção em ambiente *flow shop* permutacional com critério de minimização do *makespan* e do *total flowtime*;
- ii. Desenvolver heurísticas que forneçam soluções de qualidade e que sejam eficientes computacionalmente;

- iii. Promover a divulgação dos resultados obtidos junto à comunidade científica nacional e internacional.

1.4 ESTRUTURA DO TRABALHO

Este trabalho é organizado da seguinte forma. A Seção 2 contextualiza de forma geral a utilização de métodos quantitativos no tratamento de problemas de programação de produção. Na Seção 3, será abordado o problema *flow shop* permutacional com critério de minimização do *makespan*. Na Seção 4, será discutido o problema com critério de minimização do *total flowtime*. Cada uma destas seções conterà sua própria revisão da literatura do problema em questão, detalhamento das principais heurísticas, apresentação das heurísticas propostas, experimentação computacional e análise estatística e resultados obtidos. Na Seção 5, o trabalho encerra-se com as considerações finais. A Figura 3 ilustra de forma resumida a estrutura deste trabalho.

Figura 3 – Estrutura do Trabalho.

1.5 METODOLOGIA DE REVISÃO DA LITERATURA

A condução das pesquisas se deu pelos seguintes passos: delimitação da pesquisa, processo de busca e filtragem das publicações.

A pesquisa foi delimitada em heurísticas construtivas para o os problemas $F_m \mid prmu \mid C_{max}$ e $F_m \mid prmu \mid \sum C_j$; publicadas em artigos em periódicos indexados ao *Institute for Scientific Information* (ISI) com fator de impacto *Journal Citation Reports* (JCR);

No processo de busca, realizou-se a pesquisa por publicações no idioma inglês, primeiramente, nas seguintes bases de dados eletrônicas:

- ISI *Web of Knowledge*: <http://apps.webofknowledge.com>;
- *Science Direct* da Elsevier: <http://www.sciencedirect.com>;
- Scopus: <http://www.scopus.com>;

Posteriormente, a medida em que as publicações foram sendo encontradas, o processo de busca se deu em bibliotecas on-line de periódicos referenciados nos artigos encontrados. Nesta pesquisa, não se considerou anais de congressos, de conferências e de eventos. A busca ocorreu entre Agosto de 2013 até Outubro de 2014.

As *strings* utilizadas para buscas foram “*permutation flow shop flowshop flow-shop makespan*”, “*permutation flow shop flowshop flow-shop total flowtime mean*”, que poderiam estar contidos no título do artigo.

Por último, a seleção das publicações passou por um processos contendo três filtragens. A primeira filtragem foi um processo de análise e identificação de publicações pertinentes ao escopo da pesquisa: Uma leitura preliminar do título das publicações foi realizada, filtrando trabalhos que tratavam de outros problemas. Na segunda filtragem, foi realizada a leitura do resumo das publicações, determinando quais abordagem foram utilizadas nos trabalhos. Na terceira e última filtragem, foram selecionados apenas os trabalhos que abordavam da proposição de métodos heurísticos construtivos para os problemas $F_m \mid prmu \mid C_{max}$ e $F_m \mid prmu \mid \sum C_j$.

2 PROBLEMA DE PROGRAMAÇÃO *FLOW SHOP*

Problemas de programação de operações *flow shop* nos ambientes industriais são, frequentemente, complexos e apresentam características peculiares a cada empresa.

No âmbito da Pesquisa Operacional, tais problemas são representados por modelos, na maioria dos casos matemáticos, devendo conter as variáveis e parâmetros que explicam significativamente o comportamento dos problemas tratados.

Para o problema *flow shop*, as principais hipóteses simplificadoras do problema são:

- i. o ambiente de *flow shop* é permutacional, ou seja a mesma ordem de processamento das tarefas permanece em todas as máquinas;
- ii. os tempos de processamento das tarefas nas diversas máquinas são determinados e fixos;
- iii. as tarefas têm a mesma data de liberação, a partir da qual qualquer uma pode ser programada e executada;
- iv. os tempos de preparação das operações nas diversas máquinas são incluídos nos tempos de processamento e independem da sequência de operações em cada máquina;
- v. preempção (*preemption*): não se admite a interrupção de uma ordem para processamento de outra;
- vi. estoques intermediários podem ser formados.

Como já mencionado no capítulo anterior, o problema de programação de operações *flow shop* permutacional é um problema no qual cada tarefa de um conjunto de n tarefas deve ser processada, na mesma sequência, em cada máquina de um conjunto de m máquinas distintas tendo como objetivo a otimização de um determinado critério.

Usualmente, a solução do problema de programação *flow shop* permutacional consiste em determinar, dentre as $(n!)$ sequências possíveis das tarefas, aquela que minimiza o intervalo de tempo entre o início de execução da primeira tarefa na primeira máquina e o término de execução da última tarefa na última máquina, ou seja, a duração total da programação (*makespan*). Outro objetivo que pode ser minimizado é o tempo de total de fluxo

(*total flowtime*). Ambos os objetivos, *makespan* e *total flowtime*, são importantes nas indústrias, pois lidam com decisões relevantes que afetam o desempenho dos sistemas produtivos. Uma forma de representar a solução de um problema de programação da produção é por meio de um Diagrama de Gantt. A Figura 4 exemplifica um Diagrama de Gantt de uma possível solução para um problema de programação da produção *flow shop* permutacional com três máquinas e três tarefas.

Figura 4 - Exemplo de um Diagrama de Gantt. Fonte: Autor, 2014.

Para auxiliar a programação da produção muitos métodos de solução foram propostos. Segundo MacCarthy e Liu (1993), estes métodos são majoritariamente de três tipos: métodos eficientes ótimos, métodos enumerativos ótimos e métodos heurísticos. Dentre os métodos ótimos eficientes está o método de Johnson (1954) que determina a solução ótima para um problema de programação *flow shop* permutacional com duas máquinas e n tarefas em um tempo polinomial. Os métodos ótimos enumerativos são aqueles que determinam a solução ótima do problema, porém a um custo computacional maior do que os métodos eficientes ótimos. Dentre estes podem ser citados os procedimentos *branch & bound* e a programação dinâmica. O último tipo de método são os métodos heurísticos, que fornecem soluções de qualidade a um tempo computacional razoável.

Os métodos heurísticos podem ser classificados de diversas formas, e uma delas classifica-os em construtivos ou melhorativos. No caso dos métodos construtivos, a sequência adotada como solução do problema é obtida:

- i. diretamente a partir da ordenação das tarefas segundo índices de prioridade calculados em função dos tempos de processamento das tarefas, como por exemplo: Palmer (1965) e Koulamas (1998);
- ii. escolhendo-se a melhor sequência das tarefas a partir de um conjunto de sequências também obtidas utilizando-se índices de prioridade associados às tarefas: Campbell, Dudek & Smith (1970) e Hundal & Rajgopal (1988);
- iii. ou ainda, a partir da geração sucessiva de sequências parciais das tarefas (subsequências) até a obtenção de uma sequência completa através de algum critério de inserção de tarefas (por exemplo, NEH de Nawaz, Enscore & Ham, 1983).

No caso dos métodos melhorativos, obtém-se uma solução inicial e, posteriormente, por meio de algum procedimento iterativo (geralmente envolvendo trocas de posições das tarefas na sequência) busca-se conseguir uma solução melhor que a atual em relação à medida de desempenho adotada.

Recentemente, um extenso esforço de pesquisa tem sido dedicado tanto à solução do problema de minimização do *makespan* quanto do *total flowtime* (FRAMINAN; GUPTA; LEISTEN, 2004).

Assim, muitos métodos heurísticos têm sido propostos para a solução desse problema. Para melhor descrever-los, Framinan, Gupta e Leisten (2004) propôs que o desenvolvimento do método heurístico construtivo consiste de três fases:

- i. Fase I: desenvolvimento de um índice;
- ii. Fase II: construção da solução;
- iii. Fase III: melhoria da solução.

Na Fase I, para desenvolver um índice pode-se recorrer a uma lógica de priorização das tarefas derivada de alguma propriedade dos dados do problema. Por exemplo, usualmente, em um problema de minimização do *makespan*, as tarefas são ordenadas em ordem não crescente das somas dos tempos do processamento. Pode-se também utilizar algum tipo de analogia. Por analogia se subentende utilizar os dados do problema para construir e resolver um problema diferente (problema do caixeiro viajante). Algumas heurísticas que utilizam de

analogias para criar índices são os métodos de Palmer (1965), Gupta (1971), Campbell, Dudek e Smith (1970) e Koulamas (1998).

Na construção da solução, Fase II, uma solução é construída iterativamente por meio da inserção de tarefas não sequenciadas em uma ou mais posições da sequência parcial até que a sequência esteja completa. A cada iteração, seleciona-se a tarefa que será inserida e a posição na sequência parcial.

Por último, na Fase III, a solução gerada na Fase II é melhorada por meio de algum procedimento de melhoria, podendo ser uma busca local ou uma metaheurística.

Como já mencionado, neste trabalho serão propostos métodos heurísticos construtivos para dois problemas de programação de operações *flow shop* permutacional, o primeiro com critério de minimização do *makespan* e o segundo com a minimização do *total flowtime*. A partir do exame da literatura, os principais métodos heurísticos construtivos para cada um dos problemas serão, a seguir, sucintamente descritos.

3 PROBLEMA DE PROGRAMAÇÃO *FLOW SHOP* PERMUTACIONAL COM DE MINIMIZAÇÃO DO *MAKESPAN*

O primeiro problema ser estudado nesta dissertação é o problema *flow shop* permutacional com minimização do *makespan*. Nesta seção as principais heurísticas construtivas para o problema *flow shop* permutacional com critério de minimização do *makespan* serão revisadas. Também serão propostas heurísticas para o problema, que serão comparadas com os melhores métodos existentes para o problema.

3.1 DEFINIÇÃO DO PROBLEMA

O problema em consideração pode ser definido da seguinte maneira. Seja $\pi = \{ J_1, J_2, \dots, J_j, \dots, J_n \}$ uma sequência com um conjunto de n tarefas que devem ser processadas, na mesma ordem, por um conjunto de m máquinas distintas. O tempo de processamento da tarefa j na máquina i é p_{ij} ($j = 1, 2, \dots, n; i = 1, 2, \dots, m$). Se uma determinada tarefa não tiver operação em uma certa máquina, seu correspondente tempo de processamento é assumido como igual a zero. Uma determinada tarefa na posição k de uma sequência π também pode ser denotada por $\pi(k)$. Seja C_{ij} o tempo de conclusão da tarefa j na máquina i , o problema em questão consiste em minimizar o valor do C_{max} (*makespan*). Os valores de C_{ij} e do *makespan* podem ser calculados de forma recursiva por meio da Equação 1. Na teoria que estuda a complexidade dos problemas de natureza combinatória, o problema em questão é classificado como NP-hard (GAREY; JOHNSON; SETHI, 1976), de forma que podem ser resolvido eficientemente de maneira ótima somente em casos de pequeno porte.

$$\begin{aligned} C_{ij} &= \max(C_{i-1j}, C_{ij-1}) + p_{ij} \\ (i &= 1, \dots, m) \quad (j = 1, \dots, n) \\ C_{0j} &= C_{i0} = 0 \\ C_{max} &= C_{mn} \end{aligned} \tag{1}$$

Como pode ser observado, a complexidade do cálculo do *makespan* é $O(nm)$.

Em alguns casos, quando houver inserção de uma tarefa nas posições da sequência, também é possível calcular o valor de C_{max} por meio do método de aceleração de Taillard (1990). Para se determinar o C_{max} da sequência após a inserção da tarefa j na k -ésima posição devem ser executados os passos a seguir.

(1) Calcule o tempo de conclusão mais cedo e_{ik} da k -ésima tarefa na máquina i ; o tempo de início da primeira tarefa na primeira máquina é 0, conforme a Equação 2 (Figura 5a).

$$\begin{aligned} e_{ik} &= \max(e_{i-1,k}, e_{i,k-1}) + p_{ik} \\ (k &= 1, \dots, j-1) \quad (i = 1, \dots, m) \\ e_{i0} &= 0, e_{0k} = 0 \end{aligned} \quad (2)$$

(2) Calcule q_{ik} , que é a duração entre o tempo de início da k -ésima tarefa na máquina i e o fim das operações das tarefas da sequência, conforme a Equação 3 (Figura 5b).

$$\begin{aligned} q_{ik} &= \max(q_{i+1,k}, q_{i,k+1}) + p_{ik} \\ (k &= j-1, \dots, 1) \quad (i = m, \dots, 1) \\ q_{ij} &= 0, q_{m+1,k} = 0 \end{aligned} \quad (3)$$

(3) Calcule o tempo relativo de conclusão mais cedo f_{ik} , na máquina i da tarefa j , inserida na posição k , conforme a Equação 4 (Figura 5c).

$$\begin{aligned} f_{ik} &= \max(f_{i-1,k}, e_{i,k-1}) + p_{ij} \\ (k &= 1, \dots, j) \quad (i = 1, \dots, m) \\ f_{0k} &= 0 \end{aligned} \quad (4)$$

(4) Calcule o valor de C_{max} quando a tarefa j é adicionada na posição k , conforme a Equação 5.

$$C_{max} = \max(f_{ik}, q_{ik}) \quad (5)$$

$$(k = 1, \dots, j) \quad (i = 1, \dots, m)$$

Figura 5 - Ilustração do método: inserção da tarefa 5 na 3ª posição. Fonte: adaptado de Taillard, 1990.

Todos estes passos podem ser executados em $O(jm)$. Ou seja, é possível avaliar a inserção de uma tarefa em todas as posições de uma sequência com n tarefas em um tempo $O(nm)$. Consequentemente, o algoritmo NEH de Nawaz, Enscore e Ham (1983) pode ser executado em $O(n^2m)$, reduzindo sua complexidade (TAILLARD, 1990). Este método de aceleração também pode ser adaptado a outros algoritmos que utilizam procedimentos de inserção de tarefas semelhantes ao método NEH.

3.2 REVISÃO DA LITERATURA

Para o problema de programação *flow shop* com critério de minimização do *makespan*, desde que Johnson (1954) propôs uma solução ótima para o problema de n tarefas processadas em duas máquinas, vários algoritmos heurísticos foram desenvolvidos para resolver o problema em sistemas *flow shop* com n tarefas e mais de duas máquinas, buscando minimizar o *makespan*.

Palmer (1965) propôs um índice denominado *slope index*, a partir do qual se estabelece a sequência de processamento das tarefas nas máquinas. Tal índice é calculado de forma que as tarefas cujos tempos de processamento tendem a crescer na sequência das máquinas tenham prioridade na programação, ou seja, devem ocupar as primeiras posições na ordem de execução. O *slope index* para uma tarefa j é dado pela Equação 6.

$$H_j = \sum_{i=1}^m (2i - m - 1)p_{ij} \quad \text{para } j = 1, 2, 3, \dots, n \quad (6)$$

A partir dos valores de H_j , estabelece-se a sequência de programação das tarefas, de acordo com a ordenação não crescente dos índices.

Campbell, Dudek & Smith (1970) propuseram um procedimento conhecido por CDS, que é uma generalização do algoritmo de Johnson (1954) para a solução do problema com duas máquinas com critério de minimização do *makespan*. Sua eficiência é atribuída a duas razões básicas: (i) origina $m-1$ subproblemas artificiais de duas máquinas a partir do problema original de m máquinas; (ii) utiliza a técnica de Johnson (1954) para resolvê-los de forma heurística, ou seja, corresponde à utilização da Regra de Johnson em $(m-1)$ estágios, em cada um dos quais é obtido um problema com apenas duas máquinas, com tempos de processamento “artificiais” p'_{1j} e p'_{2j} ($j = 1, 2, \dots, n$). No primeiro estágio, $p'_{1j} = p_{1j}$ e $p'_{2j} = p_{mj}$, ou seja, a Regra de Johnson é aplicada somente quando se consideram a primeira e a última máquinas, não sendo as demais consideradas. No segundo estágio, $p'_{1j} = p_{1j} + p_{2j}$ e $p'_{2j} = p_{m-1,j} + p_{mj}$, ou seja, aplica-se a Regra de Johnson à soma dos tempos de processamento da primeira com a segunda máquina e da penúltima com a última. No estágio t , os tempos de processamento das duas máquinas “artificiais” serão dados pela Equação 7.

$$p'_{1j} = \sum_{i=1}^t p_{ij} \quad \text{e} \quad p'_{2j} = \sum_{i=1}^t p_{(m-i+1)j} \quad \text{para } j = 1, 2, \dots, n \quad (7)$$

Em cada um dos $(m-1)$ estágios, a sequência de tarefas obtida pela Regra de Johnson é utilizada para calcular a duração total da programação (*makespan*) do problema original. A sequência que fornece a menor duração é escolhida como solução para o problema.

Gupta (1971) sugeriu outro algoritmo similar ao de Palmer (1965), exceto pela forma como define o índice. Reconheceu que o algoritmo de Johnson (1954) para o problema com duas ou três máquinas é, na verdade, um método de ordenação a partir da designação de um índice para cada tarefa, sequenciando-as de acordo com a ordem não decrescente de tais índices. Gupta (1971) generalizou a função de indexação, para o caso de $m \geq 4$ máquinas, definindo, para cada tarefa j o índice z_j , fornecido pelas Equações 8 e 9.

$$z_j = \frac{A}{\min_{1 \leq i \leq m-1} (p_{ij} + p_{i+1j})} \quad \text{para } j = 1, 2, \dots, n \quad (8)$$

Em que:

$$A = \begin{cases} 1 & \text{se } p_{ij} \leq p_{1j} \\ -1 & \text{caso contrário} \end{cases} \quad (9)$$

Dannenbring (1977) sugeriu uma variação para o algoritmo CDS. O método é chamado Procedimento *Rapid Access* (RA), que procura combinar as vantagens do *slope index* de Palmer com as do método CDS, obtendo uma boa solução, de maneira simples e rápida. Ao invés de resolver $(m-1)$ problemas “artificiais” com duas máquinas, o método RA resolve um único problema, no qual os tempos de processamento de cada tarefa em cada máquina são determinados por somas ponderadas fornecidas pela Equação 10.

$$p'_{1j} = \sum_{i=1}^m (m - i + 1)p_{ij} \quad \text{e} \quad p'_{2j} = \sum_{i=1}^m (i)p_{ij} \quad \text{para } j = 1, 2, \dots, n \quad (10)$$

Nawaz, Ensore e Ham (1983) desenvolveram um algoritmo (conhecido por NEH) baseado na hipótese de que às tarefas devem ser designadas prioridades de programação diretamente proporcionais às somas dos seus tempos de processamento nas m máquinas. É interessante, neste ponto, ressaltar que o algoritmo NEH não transforma o problema original de m máquinas em um problema artificial de duas máquinas, à semelhança dos algoritmos CDS e RA. Após a ordenação das tarefa, cada tarefa é testada em todas as posições e inserida na posição que fornece o menor *makespan*.

Na primeira fase, a heurística NEH ordena as tarefas em ordem não crescente da soma de seus tempos de processamento, também conhecida como ordenação LPT (*longest processing time*). A soma dos tempos de processamento é fornecida pela Equação 11.

$$P_j = \sum_{i=1}^m p_{ij}, j \forall n \in N \quad (11)$$

Na segunda fase, uma sequência de tarefas é construída por meio da avaliação de sequências parciais originadas da ordenação inicial fornecida pela primeira fase. Supondo uma sequência já determinada para as $k - 1$ primeiras tarefas, k subsequências são obtidas pela inserção da tarefa k nas k possíveis posições da sequência corrente. Destas k subsequências geradas, aquela que fornecer o menor *makespan* é mantida como a subsequência relativa para as k primeiras tarefas da ordenação da primeira fase. Em seguida, a tarefa na posição $k + 1$ da primeira fase é considerada analogamente, e assim por diante até que as n tarefas tenham sido sequenciadas.

Hundal & Rajgopal (1988) desenvolveram uma extensão do algoritmo de Palmer, a partir do fato de que este algoritmo ignora a máquina $(m+1)/2$ quando m é ímpar, o que pode afetar a qualidade da solução, especialmente quando o número de tarefas é grande. A extensão do algoritmo de Palmer é considerada a partir de dois novos conjuntos de *slope index*, fornecidos pela Equação 12 e 13.

$$\lambda_j = \sum_{i=1}^m (2i - m)p_{ij} \quad \text{para } j = 1, 2, \dots, n \quad (12)$$

$$\omega_j = \sum_{i=1}^m (2i - m - 2)p_{ij} \quad \text{para } j = 1, 2, \dots, n \quad (13)$$

Desta forma, duas outras sequências são obtidas, sendo selecionada a melhor.

Sevast'janov (1995) propôs um algoritmo para o problema de *flow shop* permutacional, reduzindo-o a um problema de adição de vetores. Nesse caso, o algoritmo de Sevast'janov obtém uma sequência com um valor, que satisfaz a Equação 14.

$$C_{max} - C_{max}^* \leq \left(m^2 - 3m + 3 + \frac{1}{m-2} \right) \max_{i,j} (p_{ij}) \quad (14)$$

Onde m é o número de máquinas e C_{max}^* é o valor ótimo do *makespan*.

Koulamas (1998) apresentou um novo método heurístico construtivo denominado HFC para o problema de *flow shop* permutacional e também não-permutacional. A inspiração para o algoritmo HFC foi o algoritmo de Johnson, onde para duas máquinas este último fornece uma solução ótima. A principal ideia proposta é que se a tarefa k precede j na sequência ótima, então para todos os casos M_1-M_2 , M_2-M_3 e M_1-M_3 a tarefa k precede j . Cada tarefa j tem um índice de prioridade I_j inicialmente zero, e após determinada a relação de precedência das tarefas, os índices de prioridade I_k e I_j são quantificados, subtraindo uma unidade para a prioridade I_k da tarefa k e adicionando uma unidade na prioridade I_j da tarefa j . O processo é repetido para todos os pares de tarefas e, em seguida, os índices de prioridade são ordenados de forma não crescente e a sequência das tarefas é obtida. O algoritmo HFC foi comparado com o NEH e para a verificação do seu desempenho foram realizadas duas experimentações: na primeira experimentação, os tempos de processamento de todas as tarefas foram gerados aleatoriamente num intervalo de variação discreta de $[1,100]$ uniformemente distribuídos. Na segunda experimentação, os tempos de processamento das tarefas para cada problema foram gerados nos intervalos de $[1,10]$ e $[10,50]$. Esta forma de geração aleatória dos tempos de processamento das tarefas permite, com maior chance, a formulação de problemas nos quais a solução ótima pode ser não-permutacional e com um subconjunto significativo de soluções não-permutacionais sub-ótimas, ou seja, com qualidade de solução superior às permutacionais.

Recentemente, muitas variações da heurística NEH foram propostas. Analisando a heurística NEH, pode-se notar que existem alguns elementos que podem ser modificados em sua estrutura. Muitas destas extensões alteram os seguintes elementos:

- i. Ordenação inicial: como gerar a ordenação inicial das tarefas na primeira fase;
- ii. Geração da sequência: como a sequência é construída a partir da ordenação inicial;
- iii. Mecanismo de desempate: na geração da sequência, como os empates entre sequências candidatas podem ser tratados.

A seguir será apresentado o detalhamento de cada um destes elementos, bem como as principais extensões da heurística NEH presentes na literatura.

3.2.1 Ordenação inicial

A ordenação inicial de um conjunto de n tarefas determina qual tarefa será selecionada para inserção na subsequência corrente. A proposta original de Nawaz, Enscore e Ham (1983) é a de ordenar as tarefas em ordem não crescente da soma dos tempos de processamento.

Nagano e Moccasin (2002) propuseram uma ordenação inicial baseada na estimação do tempo ocioso entre as tarefas, que, segundo os autores, é superior ao NEH. Tal índice ordena as tarefas em ordem não crescente de I_j . Este índice é calculado pela diferença entre o tempo total de processamento e o maior limitante inferior dos tempos ociosos entre máquinas da tarefa j , fornecido pelas Equações 15, 16 e 17.

Faça com u e v sejam duas tarefas arbitrárias do conjunto de n tarefa. Para qualquer sequência π com tarefas u e v respectivamente nas posições j e $(j+1)$, $j = 0, 1, 2, \dots, n - 1$, tem-se:

$$I_j = \sum_{i=1}^m p_{ij} - \max_{[j; j \neq k]} L_{jk} \quad \text{para } j = 1, 2, \dots, n \quad (15)$$

$$L_{uv}^i = \max[0, (p_{i+1, u} - p_{i, u}) - UB X_{uv}^i] \quad \text{para } i = 1, 2, \dots, m - 1 \quad (16)$$

$$UB X_{uv}^1 = 0 \text{ e } UB X_{uv}^{i+1} = \max\{0, UB X_{uv}^i + (p_{iv} - p_{i+1, u})\} \quad \text{para } i = 1, 2, \dots, m - 1 \quad (17)$$

O valor de $UB X_{uv}^i$ é o limitante superior do tempo ocioso da máquina i entre o final da tarefa u e o início da tarefa v .

Framinan, Leisten e Rajendran (2003) conduziu um extenso estudo com diferentes formas de ordenação inicial e concluiu que a ordenação original proposta por Nawaz, Enscore e Ham (1983) se estabeleceu como a melhor para o problema de minimização do *makespan*. Estes resultados foram também confirmados por Kalczynski e Kamburowski (2007).

Baseando-se na Regra de Johnson, Kalczynski e Kamburowski (2008) divulgaram uma heurística, NEH KK1, que ordena as tarefas conforme o índice c_j . Os autores afirmam que o método proposto é superior à heurística NEH NM de Nagano e Moccasin (2002). O índice c_j é calculado pelas Equações 18, 19 e 20.

$$c_j = \hat{a}_j(c_j = \hat{b}_j) \text{ se } \hat{a}_j \leq \hat{b}_j (\hat{a}_j > \hat{b}_j) \quad \text{para } j = 1, 2, \dots, n \quad (18)$$

Em que:

$$\hat{a}_j = \sum_{i=1}^m |(m-1)(m-2)/2 + m - i| p_{ij} \quad \text{para } j = 1, 2, \dots, n \quad (19)$$

$$\hat{b}_j = \sum_{i=1}^m |(m-1)(m-2)/2 + i - 1| p_{ij} \quad \text{para } j = 1, 2, \dots, n \quad (20)$$

Dong, Huang e Chen (2008) propuseram uma modificação na ordenação inicial da heurística NEH, que aplica uma ordenação inicial baseada na média e no desvio dos tempos de processamento das tarefas, calculados pelas Equações 21 e 22. Este método de ordenação foi baseado nos estudos de Li e Wang (2004), em que foi constatado que quanto maior o desvio dos tempos de processamento de uma tarefa maior a prioridade que esta deve ter na ordenação inicial.

$$AVG_j = \frac{1}{m} \sum_{i=1}^m p_{ij} \quad \text{para } j = 1, 2, \dots, n \quad (21)$$

$$STD_j = \left[\frac{1}{m-1} \sum_{i=1}^m (p_{ij} - AVG_j)^2 \right]^{1/2} \quad \text{para } j = 1, 2, \dots, n \quad (22)$$

Kalczynski e Kamburowski (2009) propuseram uma modificação na ordenação inicial que supera o NEH, chamada NEH KK2, que é baseada na Regra de Johnson. A heurística NEH KK2 ordena as tarefas em ordem não crescente do mínimo entre a_j e b_j , estes índices são calculados pelas Equações 23, 24 e 25.

$$a_j = P_j + U_j \quad \text{para } j = 1, 2, \dots, n \quad (23)$$

$$b_j = P_j - U_j \quad \text{para } j = 1, 2, \dots, n \quad (24)$$

Em que:

$$U_j = \sum_{h=1}^s \left(\frac{h - 3/4}{s - 3/4} \right) (p_{s+1-h,j} - p_{t+h,j}) \quad \text{para } j = 1, 2, \dots, n \quad (25)$$

$$s = \lfloor m/2 \rfloor \quad e \quad t = \lceil m/2 \rceil$$

Mais recentemente Kalczyński e Kamburowski (2011) demonstrou que a heurística NEH KK2 é superior às modificações do NEH propostas por Kalczyński e Kamburowski (2008) e Dong, Huang e Chen (2008) em um conjunto de problemas testes proposto por Kalczyński e Kamburowski (2009), apesar disto não foi provado que a ordenação inicial proposta é superior a original para os problemas testes de Taillard (1993).

3.2.2 Geração da sequência

No método NEH, após a ordenação inicial das tarefas, a cada iteração, uma tarefa é selecionada para ser inserida em todas as posições da sequência parcial corrente, selecionando-se a posição em que o *makespan* é minimizado. Neste sentido, a proposta original é a de inserir a tarefa em k possíveis posições da sequência corrente. No entanto, fica claro que diferentes estratégias poderiam ser adotadas, tanto pela redução do número de candidatos, avaliando apenas uma fração das k possíveis posições, quanto pela exploração de um número maior de sequências candidatas.

Gao e Zhang (2007) propuseram uma heurística chamada INEH em que se aplica a ideia de Deroussi, Gourgand e Norre (2006) de selecionar uma única tarefa a ser reinserida a cada iteração do NEH. Tal tarefa é definida como aquela que, quando removida da sequência parcial, maximiza o ganho relativo do *makespan*. Em outras palavras, seleciona-se a tarefa que apresenta o maior “peso” no *makespan* da sequência. Sendo assim, após realizar a inserção da tarefa pelo método NEH, o método seleciona apenas uma tarefa da sequência parcial, aquela mais “pesada”, para ser reinserida na sequência. O peso das tarefas é calculado

conforme um índice denominado W_j , calculado conforme a Equação 26. A tarefa com o maior valor de W_j é escolhida para ser reinserida.

$$W_j = \frac{Cmax(\pi) - Cmax(\pi - \{j\})}{P_j} \quad \text{para } j = 1, \dots, n \quad (26)$$

Kalczynski e Kamburowski (2011) propôs uma heurística denominada INEH KK2D, que combina os procedimentos de construção INEH, a ordenação NEH KK2 e a forma de desempate da heurística NEH D de Dong, Huang e Chen (2008). Segundo o autor, o método superou as heurísticas NEH, NEH KK2 e NEH D.

Rad, Ruiz e Boroojerdian (2009) propuseram uma heurística de complexidade $O(n^3m)$ chamada de FRB3. Os autores modificaram a segunda fase do método NEH por meio de um mecanismo de reinserção de todas as tarefas da sequência corrente. Ou seja, ao invés de selecionar apenas uma tarefa para reinserção, como Gao e Zhang (2007), o método faz o processo de reinserção para todas as tarefas da subsequência. A heurística FRB3 apresentou resultados muito superiores à heurística NEH, porém a custos de um aumento substancial no tempo computacional. O autor também propôs uma modificação da heurística FRB3, chamada $FRB4_k$, a qual limita o número de tarefas selecionadas para reinserção a um parâmetro k .

Por fim, Ying e Lin (2013) propuseram uma heurística, chamada CL_{wts} , que aplica um mecanismo de permutação de tarefas na sequência parcial fornecida pelo NEH. Este mecanismo coloca cada uma das tarefas da sequência parcial na posição anterior à nova tarefa inserida, mantendo a nova tarefa na segunda posição de todas as sequências geradas. A principal ideia por trás deste método é a de aumentar a diversificação das soluções a fim de escapar de ótimos locais.

3.2.3 Mecanismo de desempate

Como foi visto, na segunda fase do método NEH são geradas sequências pela inserção de uma determinada tarefa em todas as posições da sequência parcial, selecionando aquela sequência que possui o menor *makespan*. No entanto, observou-se que neste processo

ocorriam vários empates entre sequências candidatas, sendo possível então melhorar a solução fornecida pelo NEH pelo tratamento adequado destes empates.

Low, Yeh e Huang (2004) apresentaram uma modificação do NEH (NEH L) na qual os desempates são tratados por meio da seleção da sequência que minimiza o tempo ocioso na máquina com a maior carga. As carga das máquinas são fornecidas pela Equação 27.

$$L_i = \sum_{j=1}^n p_{ij} \quad \text{para } i = 1, 2, \dots, n \quad (27)$$

Kalczynski e Kamburowski (2007) apresentaram uma heurística, chamada NEH KK, que aplica um critério de desempate a partir da avaliação do *makespan* das sequências parciais candidatas. Assumindo que a sequência parcial corrente é $(J_1, J_2, \dots, J_{L-1})$, e que a L -ésima tarefa da ordenação inicial é r . Faça $l^* = 1$ e $\pi^* = (r, J_1, J_2, \dots, J_{L-1})$. Para $l = 2, 3, \dots, L$ faça Se $C_{\max}((J_1, \dots, r, J_l, \dots, J_{L-1})) \leq C_{\max}(\pi^*)$ e $C_{\max}((J_1^*, \dots, J_{l-1}, r)) < C_{\max}(r, J_l, \dots, J_{L-1})$, então faça $l^* = l$ e $\pi^* = (J_1, \dots, r, J_l, \dots, J_{L-1})$.

Na heurística construtiva NEH KK1, Kalczynski e Kamburowski (2008) aplicaram um critério de desempate utilizando os índices $\hat{a}_j$ e $\hat{b}_j$ calculados na primeira fase do método por meio das Equações 18, 19 e 20. Em caso de empate a melhor sequência candidata é referente ao primeiro (último) índice no qual o mínimo é atingido se $\hat{a}_j > \hat{b}_j$ ($\hat{a}_j \leq \hat{b}_j$). Este índice é baseado na Regra de Johnson que fornece a sequência ótima para problemas de duas máquinas.

Na heurística NEH KK2 também se aplica um mecanismo de desempate parecido com o NEH KK1, no qual em caso de empate a melhor sequência candidata é referente ao primeiro (último) índice no qual o mínimo é atingido se $U_j \leq 0$ ($U_j > 0$) e U_j é calculado pela Equação 25.

Na heurística NEH D, Dong, Huang e Chen (2008) aplicou um mecanismo de desempate que seleciona a sequência que melhor balanceia a carga das tarefas nas máquinas. Este índice de balanceamento é denotado por $D_{\pi(x)}$, devendo-se escolher a sequência que minimize o valor desta variável. Este índice é calculado pelas Equações 28 e 29.

$$D_{\pi(x)} = \sum_{i=1}^m \left(\frac{p_{i,\pi(x)}}{S_{i,\pi(x+1)} - C_{i,\pi(x-1)}} - E_{\pi(x)} \right)^2 \quad (28)$$

Em que:

$$E_{\pi(x)} = \frac{1}{m} \sum_{i=1}^m \frac{p_{i,\pi(x)}}{S_{i,\pi(x+1)} - C_{i,\pi(x-1)}} \quad (29)$$

Note que existem algumas exceções. Se $\pi(x)$ é igual a última tarefa da sequência parcial, faça com que $S_{i,\pi(x+1)}$ seja igual ao tempo de conclusão mais tarde da tarefa $\pi(x)$ na máquina i . Se $S_{i,\pi(x+1)} = C_{i,\pi(x-1)}$, faça com que $C_{i,\pi(0)}$ seja igual ao tempo mais cedo de início possível da tarefa $\pi(1)$ na máquina i e também com que $p_{i,\pi(x)}/S_{i,\pi(x+1)} - C_{i,\pi(x-1)}$ seja igual a zero.

Fernandez-Viagas e Framinan (2013) propuseram um método de desempate baseado em uma estimativa do tempo ocioso total. Embora a avaliação do *makespan* de uma sequência candidata possa ser realizada em uma complexidade reduzida por meio da utilização da aceleração de Taillard, o cálculo do tempo ocioso total de uma sequência não. Para se calcular o tempo ocioso total de uma sequência é necessário determinar obrigatoriamente todos os tempos de conclusão das tarefas da sequência candidata, o que não é integralmente possível pela aceleração de Taillard. Isto motivou os autores a desenvolverem um método de desempate que reutiliza o que já foi calculado pelo método de aceleração para estimar o tempo ocioso total das sequências candidatas. Desta forma, esta estimativa reduz a complexidade no tratamento dos empates entre sequências candidatas, podendo então ser integrado à outras heurísticas sem prejudicar suas complexidades. Este critério de desempate forneceu bons resultados quando associado à ordenação inicial proposta por Dong, Huang e Chen (2008). Neste trabalho, a heurística de Fernandez-Viagas e Framinan (2013) com ordenação inicial LPT foi denominada de NEH FF.

Na heurística CL_{wts} de Ying e Lin (2013) os empates são tratados por meio da seleção da sequência que resulte no menor tempo total ocioso entre as máquinas. Este mecanismo quando integrado ao método de permutação de tarefas na geração da sequência, apresenta melhores resultados do que o NEH original; porém, não ficou provada a sua superioridade perante outros métodos mais recentes presentes na literatura.

Em função do exame da literatura, alguns aspectos relevantes podem ser ressaltados, tais como:

- i. A partir de Palmer (1965) até a divulgação de Fernandez-Viagas e Framinan (2013), houve um desenvolvimento de métodos heurísticos construtivos, que pode ser considerado regular ao longo do tempo;
- ii. Na ordenação inicial, a heurística de Kalczynski e Kamburowski (2009) superou a ordenação inicial de Dong, Huang e Chen (2008) em estudos realizados por Kalczynski e Kamburowski (2011), porém não ficou provada sua superioridade para os problemas testes de Taillard (1993);
- iii. Na geração da sequência, as heurísticas FRB3, INEH KK2D superaram o método NEH em termos de qualidade de solução;
- iv. No mecanismo de desempate, Viagas e Framinan (2013) desenvolveram um método que, além de não aumentar a complexidade à heurística, apresentou resultados similares aos de Dong, Huang e Chen (2008).

O Quadro 1 apresenta uma síntese das heurísticas construtivas desenvolvidas até o presente momento.

Na próxima seção, as principais heurísticas revisadas neste trabalho serão melhor detalhadas por meio de pseudocódigos.

Quadro 1 - Variações da heurística NEH e suas composições (Parte 1).

Autor	Ano	Acrônimo	Fase I	Fase II		Complexidade (s/ Taillard's acceleration)
			Ordenação Inicial	Desempate	Construção da Sequência	
Nawaz, Enscore JR e HAM	1983	NEH	LPT	-	NEH	$O(n^3m)$
Nagano e Moccelin	2002	NEH NM	Ordem não crescente de I_j $I_j = \sum_{i=1}^m p_{ij} - \max_{[j; j \neq k]} L_{jk}$	-	NEH	$O(n^3m)$
Low, Yeh e Huang	2004	NEHL	LPT	Escolhe a sequência que minimiza o tempo ocioso na máquina M_{i^*} com a carga máxima. $L_i = \sum_{j=1}^n p_{ij}$	NEH	$O(n^3m)$
Kalczynski e Kamburowski	2007	NEH KK	LPT	Assumindo que a sequência parcial corrente é $(J_1, J_2, \dots, J_{L-1})$, e que a L -ésima tarefa da ordenação inicial é r . Faça $l^* = 1$ e $\pi^* = (r, J_1, J_2, \dots, J_{L-1})$. Para $l = 2, 3, \dots, L$ faça Se $C_{\max}((J_1, \dots, r, J_1, \dots, J_{L-1})) \leq C_{\max}(\pi^*)$ e $C_{\max}((J_1^*, \dots, J_{l-1}, r)) < C_{\max}(r, J_l, \dots, J_{L-1})$, então faça $l^* = l$ e $\pi^* = (J_1, \dots, r, J_1, \dots, J_{L-1})$.	NEH	$O(n^3m)$
Gao e Zhang	2007	INEH	LPT	-	NEH + Reinserção de apenas uma tarefa selecionada criteriosamente	$O(n^3m)$

Quadro 1 - Continuação (parte 2).

Autor	Ano	Acrônimo	Fase I	Fase II		Complexidade (s/ Taillard's acceleration)
			Ordenação Inicial	Desempate	Construção da Sequência	
Kalczynski e Kamburowski	2008	NEH KK1	<p>Calcula:</p> $\hat{a}_j = \sum_{i=1}^m (m-1)(m-2)/2 + m - i p_{ij}$ $\hat{b}_j = \sum_{i=1}^m (m-1)(m-2)/2 + i - i p_{ij}$ <p>Ordena as tarefas em ordem não crescente de c_j, onde $c_j = \hat{a}_j (c_j = \hat{b}_j)$ se $\hat{a}_j \leq \hat{b}_j$ ($\hat{a}_j > \hat{b}_j$).</p>	Em caso de empate a melhor sequência candidata é referente ao primeiro (último) índice no qual o mínimo é atingido se $\hat{a}_j \leq \hat{b}_j$ ($\hat{a}_j > \hat{b}_j$)	NEH	$O(n^3 m)$
Dong e Chen	2008	NEH D	Ordena as tarefas pela soma da média e desvio padrão dos tempos de processamento.	<p>Calcula</p> $E_{\pi(x)} = \frac{1}{m} \sum_{i=1}^m \frac{p_{i,\pi(x)}}{S_{i,\pi(x+1)} - C_{i,\pi(x-1)}}$ $D_{\pi(x)} = \sum_{i=1}^m \left(\frac{p_{i,\pi(x)}}{S_{i,\pi(x+1)} - C_{i,\pi(x-1)}} - E_{\pi(x)} \right)^2$ <p>Onde S e C são os tempos de realização das tarefas na data mais tarde e mais cedo, respectivamente.</p> <p>Escolhe a sequência candidata que minimiza o valor de $D_{\pi(x)}$.</p>	NEH	$O(n^3 m)$
Kalczynski e Kamburowski	2009	NEH KK2	<p>Ordena as tarefas em ordem decrescente de $\min(a_j, b_j)$, onde $a_j = P_j + U_j$, $b_j = P_j - U_j$.</p> $U_j = \sum_{h=1}^s \left(\frac{h-3/4}{s-3/4} \right) (p_{s+1-h,j} - p_{t+h,j})$ <p>$s = \lfloor m/2 \rfloor$ e $t = \lceil m/2 \rceil$</p>	Em caso de empate a melhor sequência candidata é referente ao primeiro (último) índice no qual o mínimo é atingido se $U_j \leq 0$ ($U_j > 0$).	NEH	$O(n^3 m)$

Quadro 1 - Continuação (parte 3).

Autor	Ano	Acrônimo	Fase I	Fase II		Complexidade (s/ Taillard's acceleration)
			Ordenação Inicial	Desempate	Construção da Sequência	
Rad, Ruiz e Borojerdian	2009	FRB3	LPT	-	NEH + Reinserção de todas as tarefas da sequência.	$O(n^4 m)$
Rad, Ruiz e Borojerdian	2009	FRB4 _L	LPT	-	NEH + Reinserção das L primeiras tarefas da sequência corrente.	$O(Ln^3 m)$
Kalczynski e Kamburowski	2011	INEH KK2D	NEH KK2	NEH D	INEH	$O(n^3 m)$
Ying e Lin	2013	CL _{wtS}	LPT	Seleciona a sequência com o menor tempo total ocioso entre as máquinas.	NEH + Permutação de tarefas em um número limitado de posições.	$O(n^3 m)$
Viagas e Framinan	2013	NEH FF	LPT	Escolhe a sequência que minimiza a estimativa do tempo ocioso total $it'' = \sum_{i=1}^m f_{il} - e_{il} + p_{il} - p_{ir} + \max(f'_{i-1,l} - f_{il}, 0)$	NEH	$O(n^3 m)$

3.3 HEURÍSTICAS CONSTRUTIVAS PARA O PROBLEMA F | $prmu$ / C_{MAX}

Nesta seção, será detalhado o funcionamento das principais heurísticas construtivas apresentadas na revisão da literatura para o problema de *flow shop* permutacional para minimização do *makespan*. Foram selecionadas para detalhamento as seguintes heurísticas construtivas: NEH, NEH NM, NEH KK, NEH KK1, NEH D, NEH KK2, FRB3, INEH KK2D e NEH FF.

i. Heurística NEH

A heurística NEH foi proposta por Nawaz, Enscore e Ham (1983). Este procedimento é demonstrado pelo pseudocódigo na Figura 6.

```
procedimento NEH  
  Calcule  $P_j = \sum_{i=1}^m p_{ij}, \forall n \in N$ ;  
  Ordene  $P$  em ordem decrescente;  
   $\pi := \emptyset$ ;  
  para passo := 1 até  $n$  faça  
     $j := tarefa(P_j[passo])$ ;  
    Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ ;  
    Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor  
     $C_{max}$ ;  
  fim  
fim
```

Figura 6 – Heurística NEH.

Sem a aceleração de Taillard a complexidade da heurística NEH é $O(n^3m)$, visto que o cálculo do *makespan* pode ser feito em $O(nm)$ pela Equações 1. Taillard (1990) propôs um mecanismo de aceleração que reduz a complexidade da heurística para $O(n^2m)$.

ii. Heurística NEH NM

Nagano e Moccellini (2002) realizaram uma modificação na primeira fase do método, substituindo a ordenação LPT por outro índice calculado pela Equação 15 denominado de I_j . A Figura 7 apresenta o pseudocódigo do método NEH NM.

```

procedimento NEH NM
    Calcule  $UBX_{uv}^i$  (Equação 17);
    Calcule  $L_{uv}^i$  (Equação 16);
    Calcule  $I_j$  (Equação 15);
    Ordene  $I$  em ordem decrescente;
     $\pi := \emptyset$ ;
    para  $passo := 1$  até  $n$  faça
         $j := tarefa(I[passo])$ ;
        Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ ;
        Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor
         $C_{max}$ ;
    fim
fim

```

Figura 7 – Heurística NEH NM.

iii. Heurística NEH KK

A heurística NEH KK foi proposta por Kalczynski e Kamburowski (2007). O método implementa um mecanismo de desempate na segunda fase do método NEH. A Figura 8 ilustra o funcionamento do método NEH KK.

```

procedimento NEH KK
    Calcule  $P_j = \sum_{i=1}^m p_{ij}, \forall n \in N$ ;
    Ordene  $P$  em ordem decrescente;
     $\pi := \emptyset$ ;
    para  $passo := 1$  até  $n$  faça
         $j := tarefa(P[passo])$ ;
        Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ ;
        Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
        Em caso de empate, aplique o desempate do NEH KK;
    fim
fim

```

Figura 8 – Heurística NEH KK.

iv. Heurística NEH KK1

A heurística NEH KK1 foi proposta por Kalczynski e Kamburowski (2008). Na primeira fase da heurística NEH KK1, ao invés de se utilizar a ordenação LPT, ordena-se as tarefas em ordem não crescente de c_j , calculado por meio da Equação 18. No procedimento de construção, aplica-se um critério de desempate em que a melhor sequência candidata é referente ao primeiro (último) índice no qual o mínimo é atingido se $\hat{a}_j \leq \hat{b}_j$ ($\hat{a}_j > \hat{b}_j$), calculados por meio das Equações 19 e 20. A Figura 9 apresenta o procedimento da heurística NEH KK1.

```

procedimento NEH KK1

    Calcule  $\hat{a}_j$  e  $\hat{b}_j \forall n \in N$  (Equações 19 e 20);
    Calcule  $c_j \forall n \in N$  (Equação 18);
    Ordene  $c_j$  as tarefas em ordem decrescente;
     $\pi := \emptyset$ ;
    para  $passo := 1$  até  $n$  faça
         $j := tarefa(c[passo])$ ;
        Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ ;
        Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
        Em caso de empate, a melhor sequência candidata é referente ao
        primeiro (último) índice no qual o mínimo é atingido se  $\hat{a}_j \leq \hat{b}_j$  ( $\hat{a}_j >$ 
         $\hat{b}_j$ );
    fim
fim

```

Figura 9 – Heurística NEH KK1.

v. Heurística NEH D

A heurística NEH D de Dong, Huang e Chen (2008) fornece a ordenação inicial pela soma da média com desvio dos tempos de processamento das tarefas, conforme as Equações 21 e 22. O desempate é realizado escolhendo-se a posição que fornece o menor valor de $D_{\pi(x)}$, o qual é calculado pelas Equações 28 e 29. A Figura 10 apresenta o pseudocódigo da heurística NEH D.

```

procedimento NEH D
    Calcule  $AVG_j$  e  $STD_j \forall n \in N$  (Equações 21 e 22);
    Ordene  $AVG_j + STD_j$  em ordem decrescente;
     $\pi := \emptyset$ ;
    para  $passo := 1$  até  $n$  faça
         $j := tarefa(AVG[passo] + STD[passo])$ ;
        Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ ;
        Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
        Em caso de empate, escolher a sequência de tarefas
        que possui o menor valor de  $D_{\pi(x)}$ . (Equações 28 e 29);
    fim
fim

```

Figura 10 – Heurística NEH D.

vi. Heurística NEH KK2

O método NEH KK2, proposto por Kalczynski e Kamburowski (2009), substitui a ordenação inicial LPT por uma indexação das tarefas em ordem não crescente do mínimo entre a_j e b_j , fornecidos pelas Equações 23 e 24. Na segunda fase do método, um método de desempate é aplicado em que a melhor sequência candidata é referente ao primeiro (último) índice no qual o mínimo é atingido se $U_j \leq 0$ ($U_j > 0$), sendo U_j calculado pela Equação 25. A Figura 11 apresenta o pseudocódigo da heurística NEH KK2.

```

procedimento NEH KK2
    Calcule  $U_j \forall n \in N$  (Equação 25);
    Calcule os valores de  $a_j$  e  $b_j$  (Equações 23 e 24)
    Ordene as tarefas em ordem decrescente do mínimo entre  $a_j$  e  $b_j$ ;
     $\pi := \emptyset$ ;
    para  $passo := 1$  até  $n$  faça
         $j := tarefa(\min(a[passo], b[passo]))$ ;
        Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ ;
        Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
        Em caso de empate, a melhor sequência candidata é
        referente ao primeiro (último) índice no qual o mínimo é
        atingido se  $U_j \leq 0$  ( $U_j > 0$ );
    fim
fim

```

Figura 11 – Heurística NEH KK2.

vii. Heurística FRB3

A heurística FRB3, proposta por Rad, Ruiz e Boroojerdian (2009), realiza um procedimento de reinserção de todas as tarefas a cada iteração do NEH. A Figura 12 mostra o funcionamento do algoritmo.

```
procedimento FRB3
    Calcule  $P_j \ \forall n \in N$ ;
    Ordene  $P$  em ordem decrescente;
     $\pi := \emptyset$ ;
    para  $passo := 1$  até  $n$  faça
         $j := tarefa(P[passo])$ ;
        Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ .
        Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor  $C_{max}$ .
        para  $passo2 := 1$  até  $n$  faça
            Extraia a tarefa  $h$  da posição  $passo2$  de  $\pi$  .
            Teste a tarefa  $h$  em todas as posições possíveis de  $\pi$ .
            Insira a tarefa  $h$  na posição de  $\pi$  que resulte no menor  $C_{max}$ .
        fim
    fim
fim
```

Figura 12 – Heurística FRB3.

viii. Heurística INEH KK2D

A heurística INEH KK2D, proposta por Kalczynski e Kamburowski (2011), utiliza a ordenação inicial da heurística NEH KK2, o critério de desempate da heurística NEH D e realiza apenas uma reinserção a cada iteração do NEH, selecionando a que fornece o menor peso W_j , calculado pela Equação 26. A Figura 13 mostra o pseudocódigo da heurística INEH KK2 D.

```

procedimento INEH KK2D

  Calcule  $P_j \ \forall n \in N$ ;
  Ordene  $P$  em ordem decrescente;
   $\pi := \emptyset$ ;
  para  $passo := 1$  até  $n$  faça
     $j := tarefa(P[passo])$ ;
    Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ ;
    Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
    para  $passo2 := 1$  até  $n$  faça
       $j := tarefa(passo2)$ ;
      Retire a tarefa  $j$  da sequência  $\pi$ ;
      Calcule  $W_j = \frac{C_{max}(\pi) - C_{max}(\pi - \{j\})}{P_j}$ ;

    fim
    Ordene  $W$  em ordem decrescente;
     $h := tarefa(W[1])$ ;
    Retire a tarefa  $h$  de  $\pi$  ;
    Teste a tarefa  $h$  em todas as posições possíveis de  $\pi$ ;
    Insira a tarefa  $h$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
    Em caso de empate, escolha a sequência de tarefas;
    que possui o menor valor de  $D_{\pi(x)}$ ;

  fim
fim

```

Figura 13 – Heurística INEH KK2D.

ix. Heurística NEH FF

O método NEH FF, proposto por Viagas e Framinan (2013), executa um procedimento de desempate das sequências candidatas de acordo com um índice denominado it'' . Seu pseudocódigo é apresentado na Figura 14. Na próxima seção serão apresentadas as heurísticas construtivas propostas para o problema $F_m | pmu / C_{max}$.

procedimento NEH FF

Calcule $P_j = \sum_{i=1}^m p_{ij}, \forall n \in N$;

Ordene P em ordem decrescente;

$\pi := \emptyset$;

para $k := 1$ **até** n **faça**

$j := tarefa(P[k])$;

Determine os valores de e_{ij}, q_{ij}, f_{ij} (Equações 2, 3 e 4);

Teste a tarefa j em todas as posições possíveis de π ;

$bp :=$ primeira posição onde o *makespan* é mínimo;

$tb :=$ número de posições com *makespan* mínimo (número de empates);

$ptb :=$ vetor (de tamanho tb) com as posições onde o *makespan* é mínimo;

$it_{bp} :=$ valor muito grande;

se $tb > 1$ **e** $k < n$ **então**

para $l = 1$ **até** tb **faça**

$it'' := 0$;

se $ptb[l] = k$ **então**

para $i = 2$ **até** m **faça**

$it'' := it'' + f_{ik} - e_{ik-1} - p_{ij}$;

fim

se não

$f'_{1,ptb[l]} := f_{i,ptb[l]} + p_{1,ptb[l]}$;

para $i = 2$ **até** m **faça**

$it'' := it'' + f_{i,ptb[l]} + e_{i,ptb[l]} + p_{i,ptb[l]} - p_{ij} +$

$\max\{0, f'_{i-1,ptb[l]} - f_{i,ptb[l]}\}$;

$f'_{i,ptb[l]} := \max\{f'_{i-1,ptb[l]}, f_{i,ptb[l]}\} + p_{i,ptb[l]}$;

fim

fim

Se $it_{bp} > it''$ **então**

$bp := ptb[l]$

$it_{bp} := it''$

fim

fim

fim

fim

fim

Figura 14 - Procedimento NEH FF.

3.4 NOVOS MÉTODOS HEURÍSTICOS PARA O PROBLEMA $F | prmu / C_{MAX}$

A primeira heurística apresentada é denominada Gx ($x = 1, 2, \dots, 15$). A heurística proposta é semelhante ao método FRB3, que modifica a fase de construção da solução da heurística NEH, e também se baseou no mecanismo de busca local utilizado pelo método IG_RS de Ruiz e Stützle (2007), que possui uma fase iterativa de destruição e reconstrução da solução.

A heurística IG_RS possui duas fases, uma fase de destruição da solução, na qual se retira uma tarefa aleatoriamente da sequência, e uma fase de construção, onde se coloca a tarefa retirada na posição que fornece o menor C_{max} . Este procedimento é realizado iterativamente até que não se melhore mais a solução. A Figura 15 apresenta o pseudocódigo do método IG_RS.

```
procedimento IG_RS
    Gere uma solução inicial  $\pi$  pelo método NEH;
    melhorou := verdade;
    enquanto (melhoria := verdade) faça
        melhorou := falso;
        para  $k := 1$  até  $n$  faça
            remova a tarefa  $j$  de uma posição aleatória de  $\pi$  (sem repetição);
            teste a tarefa  $j$  em todas as posições de  $j$  de  $\pi$ ;
             $\pi' :=$  sequência com a tarefa  $j$  na posição que resultou no menor  $C_{max}$ ;
            se  $C_{max}(\pi') < C_{max}(\pi)$  então
                 $\pi := \pi'$ ;
                melhorou := verdadeiro;
            fim
        fim
    fim
fim
```

Figura 15 – Pseudocódigo da heurística IG_RS. Fonte: adaptado de Ruiz e Stützle, 2007.

Para um melhor entendimento do funcionamento do método IG_RS, a Figura 16 ilustra de forma geral o procedimento executado pelo método.

Figura 16 - Ilustração do procedimento IG_RS.

Semelhante à heurística FRB3, após cada iteração do NEH, o método Gx realiza uma busca na vizinhança na solução baseada no procedimento IG_RS. Este novo procedimento executa uma fase de destruição da solução, por meio da retirada de pares de tarefas da sequência, e uma segunda fase de reconstrução da sequência, reinserindo o par de tarefas novamente na solução. Na fase de destruição, um par de tarefas é retirado da sequência, testando a primeira tarefa do par em todas as posições e escolhendo a posição que fornece o menor *makespan* e em seguida realizando o mesmo procedimento com a outra tarefa do par.

No desenvolvimento da heurística foram testadas várias formas de seleção de pares da sequência corrente para reinserção, gerando um total de 15 procedimentos diferentes de seleção. As formas testadas estão apresentadas no Quadro 2, onde se tem uma sequência hipotética $\{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8\}$, e são exemplificadas as várias lógicas de seleção dos pares de cada um dos métodos criados.

Quadro 2 - Diferentes formas de seleção dos pares de tarefas.

Heurística	Sequência = $\{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8\}$
	Pares de tarefas selecionados para destruição/reconstrução
G1	$\{J_1, J_2\} \{J_2, J_3\} \{J_3, J_4\} \{J_4, J_5\} \{J_5, J_6\} \{J_6, J_7\} \{J_7, J_8\}$
G2	$\{J_8, J_7\} \{J_7, J_6\} \{J_6, J_5\} \{J_5, J_4\} \{J_4, J_3\} \{J_3, J_2\} \{J_2, J_1\}$
G3	$\{J_1, J_8\} \{J_2, J_7\} \{J_3, J_6\} \{J_4, J_5\}$
G4	$\{J_1, J_3\} \{J_2, J_4\} \{J_3, J_5\} \{J_4, J_6\} \{J_5, J_7\} \{J_6, J_8\}$
G5	$\{J_1, J_4\} \{J_2, J_5\} \{J_3, J_6\} \{J_3, J_7\} \{J_4, J_8\}$
G6	$\{J_1, J_2\} \{J_3, J_4\} \{J_5, J_6\} \{J_7, J_8\} \{J_2, J_3\} \{J_4, J_5\} \{J_6, J_7\}$
G7	$\{J_1, J_3\} \{J_3, J_5\} \{J_5, J_7\} \{J_2, J_4\} \{J_4, J_6\} \{J_6, J_8\}$
G8	$\{J_1, J_2\} \{J_3, J_4\} \{J_5, J_6\} \{J_7, J_8\} \{J_2, J_3\} \{J_4, J_5\} \{J_6, J_7\}$
G9	$\{J_1, J_8\} \{J_2, J_7\} \{J_3, J_6\} \{J_4, J_5\} \{J_1, J_2\} \{J_3, J_4\} \{J_5, J_6\} \{J_7, J_8\}$
G10	$\{J_1, J_2\} \{J_1, J_3\} \{J_3, J_4\} \{J_3, J_5\} \{J_5, J_6\} \{J_5, J_7\} \{J_6, J_7\} \{J_6, J_8\}$
G11	$\{J_1, J_4\} \{J_3, J_6\} \{J_5, J_7\} \{J_2, J_5\} \{J_4, J_7\}$
G12	Seleciona n pares que possuem maior soma do tempo total de processamento
G13	Seleciona n pares que possuem menor soma do tempo total de processamento
G14	Seleciona n pares de tarefas j e k , priorizando os que possuem menor $UBX[m][j][k]$
G15	Seleciona n pares de tarefas j e k , priorizando os que possuem menor $UBX[m][j][k]$

Experimentalmente, a heurística G8 mostrou desempenho superior às outras formas de seleção dos pares nos problemas testes do Taillard, como será mostrado na seção 3.5. Quando aplicada a aceleração de Taillard, o método G8 possui uma complexidade de $O(n^3m)$ (a mesma do método FRB3). A Figura 17 exemplifica parte do funcionamento do mecanismo de destruição e reconstrução da heurística G8. O pseudocódigo da heurística G8 é apresentado na Figura 18.

Figura 17 - Mecanismo de destruição e reconstrução da heurística G8.

```

procedimento G8
    Calcule  $P_j \forall n \in N$ ;
    Ordene  $P$  em ordem decrescente;
     $\pi := \emptyset$ ;
    para  $passo := 1$  até  $n$  faça
         $j := tarefa(P[passo])$ ;
        Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ ;
        Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
        para  $passo2 := 1$  até  $n - 1$  faça
            Extraia as tarefas  $h$  e  $r$  das posições  $passo2$  e  $passo2 + 1$  de  $\pi$ ,
respectivamente;
            Teste a tarefa  $h$  em todas as posições possíveis de  $\pi$ ;
            Insira a tarefa  $h$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
            Teste a tarefa  $r$  em todas as posições possíveis de  $\pi$ ;
            Insira a tarefa  $r$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
        fim
        para  $passo3 := 2$  até  $n - 1$  faça
            Extraia as tarefas  $h$  e  $r$  das posições  $passo3$  e  $passo3 + 1$  de  $\pi$ ,
respectivamente;
            Teste a tarefa  $h$  em todas as posições possíveis de  $\pi$ ;
            Insira a tarefa  $h$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
            Teste a tarefa  $r$  em todas as posições possíveis de  $\pi$ ;
            Insira a tarefa  $r$  na posição de  $\pi$  que resulte no menor  $C_{max}$ ;
        fim
    fim
fim

```

Figura 18 – Pseudocódigo da heurística G8.

A fim de melhorar o desempenho do método proposto, foram integrados à heurística G8 mecanismos de desempate e diferentes ordenações iniciais das principais heurísticas da literatura. Estas heurísticas geradas por combinações foram denominadas de Fx ($x = 1, 2, \dots, 52$). Na ordenação inicial, foram implementados os métodos de indexação das heurísticas NEH NM, NEH KK1, NEH KK2, NEH D. No desempate, foram utilizados os procedimentos utilizados nas heurísticas NEH KK1, NEH KK2, NEH D e NEH FF.

Na heurística G8, o mecanismo de desempate pode ser aplicado em duas partes do procedimento, na fase de inserção do NEH e no procedimento de reconstrução da sequência.

Por esse motivo, um mecanismo de desempate pode ser utilizado na inserção do NEH e outro procedimento distinto pode ser integrado na fase de reconstrução da sequência. Também é possível aplicar um método de desempate apenas na fase do NEH, ou também ao contrário, aplicar apenas na fase de reconstrução. O método de desempate utilizado na heurística NEH D não foi aplicado na fase de reconstrução, por não ser possível integrar completamente a aceleração de Taillard ao procedimento de desempate, o que alteraria a complexidade da heurística G8. Neste sentido, o mecanismo de desempate do NEH D foi aplicado apenas na fase de inserção do NEH, o que não impacta significativamente o custo computacional da heurística. No total foram realizadas 52 combinações, que estão apresentadas no Quadro 3.

Quadro 3 – Combinações entre mecanismos de desempate e ordenações iniciais.

Acrônimo	Ordenação Inicial	Desempate		Acrônimo	Ordenação Inicial	Desempate	
		Inserção	Reconstrução			Inserção	Reconstrução
F1	KK1			F27	D	D	
F2	KK2			F28	KK1	D	
F3	D			F29	KK2	D	
F4	LPT	KK1	KK1	F30	LPT	D	KK1
F5	LPT	KK2	KK2	F31	D	D	KK1
F6	KK1	KK1	KK1	F32	KK1	D	KK1
F7	KK2	KK2	KK2	F33	LPT	D	KK2
F8	D	KK1	KK1	F34	D	D	KK2
F9	D	KK2	KK2	F35	KK2	D	KK2
F10	LPT	KK1		F36	LPT	FF	FF
F11	LPT	KK2		F37	KK1	FF	FF
F12	KK1	KK1		F38	KK2	FF	FF
F13	KK2	KK2		F39	D	FF	FF
F14	D	KK1		F40	LPT		FF
F15	D	KK2		F41	KK1		FF
F16	LPT		KK1	F42	KK2		FF
F17	LPT		KK2	F43	D		FF
F18	KK1		KK1	F44	LPT	FF	KK1
F19	KK2		KK2	F45	KK1	FF	KK1
F20	D		KK1	F46	KK2	FF	KK1
F21	D		KK2	F47	D	FF	KK1
F22	LPT	FF		F48	LPT	FF	KK2
F23	KK1	FF		F49	KK1	FF	KK2
F24	KK2	FF		F50	KK2	FF	KK2
F25	D	FF		F51	D	FF	KK2
F26	LPT	D		F52	NM	KK2	

Na próxima seção, serão apresentadas as análises dos experimentos computacionais e análises estatísticas realizadas. As heurística serão comparadas em relação à qualidade de solução e eficiência computacional.

3.5 EXPERIMENTAÇÃO COMPUTACIONAL E ANÁLISE DOS RESULTADOS DO PROBLEMA $F | prmu / C_{MAX}$

Nesta seção foi conduzida uma extensa experimentação computacional e avaliação estatística das melhores heurísticas existentes na literatura bem como dos métodos propostos para o problema *flow shop* permutacional com minimização do *makespan*. Foram analisadas um total de 77 heurísticas construtivas, entre as quais 67 são métodos propostos e 10 são heurísticas provenientes da literatura. Para analisar os resultados dos métodos recorreu-se aos problemas testes de Taillard (1993) e também ao conjunto de problemas VFR proposto por Vallada, Ruiz e Framinan (2014). Todos os métodos foram programados em C++ e foram executados em um processador i7 4770 3.9 GHz com 8 GB de memória RAM e um SSD de 120GB. O método de aceleração proposto por Taillard (1990) foi implementado onde possível nos métodos comparados.

Os problemas testes do Taillard é um conjunto conhecido de problemas para *flow shop* permutacional com critério de minimização do *makespan*, que consiste em um total de 120 problemas testes de variados tamanhos, possuindo 20, 50, 100, 200 e 500 tarefas e 5, 10 ou 20 máquinas. Estes problemas testes são divididos em 12 subgrupos, cada um contendo 10 problemas. Os subconjuntos são denotados conforme seus tamanhos: 20 x 5, 20 x 10, 20 x 20, 50 x 5, 50 x 10, 50 x 20, 100 x 5, 100 x 10, 100 x 20, 200 x 10, 200 x 20, 500 x 20.

A medida de performance utilizada foi o desvio relativo (DR) calculado conforme a Equação 30:

$$DR(C_{max}(\pi^h)) = \frac{(C_{max}(\pi^h) - C_{max}^*)}{C_{max}^*} \times 100 \quad (30)$$

O valor de $C_{max}(\pi^h)$ é o *makespan* fornecido pela sequência π^h gerada pela heurística h . Para os problemas testes do Taillard, C_{max}^* é a solução ótima ou o menor limitante superior conhecido para os problemas testes do Taillard até Dezembro de 2013. Estas soluções estão presentes no endereço <<http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html>>. Para os problemas testes do VFR, C_{max}^* é a melhor solução encontrada entre todas as heurísticas implementadas

neste trabalho. O Desvio Relativo Médio (DRM) é a média dos desvios relativos obtidos pelo método para um determinado conjunto de problemas, podendo estar relacionado a um determinado agrupamento de problemas testes ou a todos os problemas testes de um determinado conjunto.

Para analisar o tempo computacional dos métodos, foi utilizada a medida de performance proposta por Fernandez-Viagas e Framinan (2015) denominada Tempo Relativo Porcentual Médio (TRPM). Segundo os autores, este indicador suaviza as distorções nas análises de tempo computacional entre heurísticas de complexidades diferentes. O valor do TRPM da heurística h é definido por meio da Equação 31.

$$\text{TRPM}_h = \sum_{i=1}^I \frac{\text{TRP}_{wh}}{Y} \quad (31)$$

O valor do Tempo Relativo Porcentual (TRP) da heurística h para o problema teste w é calculado pela Equação 32, sendo Y o número total de problemas testes e T_{ih} o tempo computacional utilizado pela heurística h para executar o problema teste w .

$$\text{TRP}_{wh} = \frac{T_{wh} - \text{TCM}_w}{\text{TCM}_w} \quad (32)$$

O Tempo Computacional Médio (TCM) do problema teste w é definido pela Equação 33, sendo H o número total de heurísticas.

$$\text{TCM}_w = \sum_{h=1}^H T_{wh} / H \quad (33)$$

Primeiramente, foram analisadas somente as principais heurísticas reportadas na literatura para determinar quais métodos possuem melhor performance. Os métodos que apresentaram melhor performance foram então comparados com os algoritmos propostos neste trabalho. Os resultados dos métodos estão dispostos na Tabela 1. Todos os tempos de computação fornecidos neste trabalho são dados em segundos.

Tabela 1 – DRM para cada classe de problemas e média geral obtidas pelos métodos heurísticos nos problemas testes do Taillard.

Problema	NEH	NEH NM	NEH KK	NEH KK1	CL _{WTS}	NEH KK2	NEH FF	NEH D	INEH KK2D	FRB3
20 x 5	3,30	3,21	2,48	2,81	2,84	2,48	2,29	2,81	2,05	1,13
20 x 10	4,60	4,21	4,45	4,43	4,00	3,69	4,15	3,75	3,49	2,50
20 x 20	3,73	3,41	3,58	3,34	3,72	3,56	3,30	3,64	2,77	1,97
50 x 5	0,73	0,61	0,68	0,67	0,44	0,51	0,92	0,73	0,58	0,24
50 x 10	5,07	5,46	5,01	5,46	5,17	5,13	5,15	4,66	3,85	2,25
50 x 20	8,33	7,87	8,03	7,92	7,92	8,03	7,89	7,49	6,67	5,41
100 x 5	0,53	0,61	0,43	0,49	0,40	0,30	0,42	0,36	0,34	0,17
100 x 10	2,21	2,20	1,97	1,78	1,93	1,96	2,17	1,70	1,48	1,05
100 x 20	6,17	6,41	5,99	6,06	6,16	6,25	5,84	5,80	5,03	3,68
200 x 10	1,26	1,37	1,21	1,19	1,17	1,09	0,97	0,96	0,72	0,47
200 x 20	4,46	4,37	4,35	4,16	4,06	4,21	4,12	3,80	3,36	2,06
500 x 20	2,08	2,23	2,09	2,00	1,97	1,86	1,78	1,68	1,28	0,76
Média	3,54	3,50	3,36	3,36	3,32	3,26	3,25	3,12	2,63	1,81

A partir dos resultados, o método FRB3 claramente é o que fornece melhores soluções, superando os outros métodos em todos os subgrupos de problemas e fornecendo uma média de 1,81% para o conjunto de problemas do Taillard, contra 2,63% do segundo melhor método, o INEH KK2D. As Figuras 19 e 20 apresentam o DRM agrupado, respectivamente, pelo número de tarefas e máquinas das heurísticas que obtiveram melhor performance.

Figura 19 - DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes do Taillard.

A Figura 20 mostra que existe uma tendência de melhoria na qualidade de solução com o aumento do número de tarefas, com exceção de problemas com 50 tarefas, onde as heurísticas comparadas obtiveram uma performance inferior aos outros conjuntos de problemas. Os métodos apresentam melhor performance para problemas com 500 tarefas, se aproximando consideravelmente das melhores soluções para os problemas testes do Taillard.

Figura 20 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes do Taillard.

Por meio da Figura 20, pode ser verificada uma tendência de piora da qualidade de solução para todos os métodos conforme o número de máquinas aumenta. O método FRB3 apresenta uma performance superior às heurísticas comparadas em todos os agrupamentos.

Os intervalos das médias dos métodos implementados foram dispostas em um gráfico apresentado pela Figura 21. Foi utilizado um nível de confiança de 95%. Se os intervalos de duas médias se sobrepuserem, pode ser concluído que não existem diferenças significativas entre as médias no intervalo de confiança estabelecido.

Figura 21 – Gráfico com os intervalos das médias das heurísticas com nível de confiança de 95%.

Pode ser observado que o método FRB3 é estaticamente diferente dos outros métodos, visto que o intervalo de sua média não se sobrepõe ao de outros métodos comparados.

Outra forma de analisar os resultados, utilizada por Rad, Ruiz e Boroojerdian (2009), é calcular o desvio relativo porcentual em relação ao NEH obtido pela heurística h (DRP_h) conforme a Equação 34.

$$DRP_h = \frac{DR_{NEH}}{DR_h} \times 100 \quad (34)$$

Se o valor de DRP_h for igual a 100, a heurística comparada tem desvio igual à solução referencial, enquanto que valores de $DRP_h > 100$ ou $DRP_h < 100$ indicam melhor e pior performance relativa para uma determinada heurística em relação ao NEH, respectivamente. Quando o valor de DRP_h for igual ou muito próximo de zero, a Equação 34 resulta em uma divisão por zero ou em um valor muito grande, neste caso o valor de DRP_h é desconsiderado. Isto significa que os resultados obtidos são menores do que os valores reais. A Tabela 2 apresenta os resultados desta análise.

Tabela 2 –DRP médio obtido pelas heurísticas nos problemas testes de Taillard.

Problema	NEH	NEH NM	NEH KK	NEH KK1	CL _{WTS}	NEH KK2	NEH FF	NEH D	INEH KK2D	FRB3
20 x 5	100,00	95,91	136,77	115,42	119,21	124,03	157,32	132,24	210,04	306,07
20 x 10	100,00	118,35	109,93	112,71	121,08	133,61	115,50	125,10	166,35	223,99
20 x 20	100,00	110,60	115,28	119,46	99,16	114,18	115,00	112,54	137,13	214,46
50 x 5	100,00	140,99	146,21	139,90	236,86	225,79	129,27	110,80	201,30	447,64
50 x 10	100,00	93,14	102,42	94,02	102,37	98,60	107,97	111,25	138,24	254,02
50 x 20	100,00	107,10	104,55	107,10	106,27	105,16	107,20	113,94	127,31	158,40
100 x 5	100,00	94,64	143,86	147,17	161,08	177,94	260,21	133,83	198,72	231,25
100 x 10	100,00	100,42	116,23	127,45	120,74	116,80	112,25	138,53	158,74	262,06
100 x 20	100,00	95,98	104,86	102,63	100,91	100,69	106,72	108,10	124,40	170,59
200 x 10	100,00	93,16	115,14	131,89	111,90	139,50	319,20	141,44	251,16	677,61
200 x 20	100,00	101,76	102,86	108,17	109,60	105,80	108,57	118,74	135,65	239,10
500 x 20	100,00	94,12	100,92	104,99	107,01	112,81	118,19	125,85	167,17	281,43
Média	100,00	103,85	116,59	117,58	124,68	129,58	146,45	122,70	168,02	288,89

Como pode ser visualizado, há ganhos significativos em relação ao NEH, com o melhor desempenho obtido pelo método FRB3, que alcança um ganho relativo de performance médio de 188,89%. Porém, apesar do FRB3 soluções de alta qualidade, o seu custo computacional é significativamente maior em relação aos outros métodos. A Tabela 3 apresenta os resultados da análise do esforço computacional de cada método.

Tabela 3 - Eficiência computacional das heurísticas nos problemas testes do Taillard.

Problema	NEH	NEH NM	NEH KK	NEH KK1	CL _{WTS}	NEH KK2	NEH FF	NEH D	INEH KK2D	FRB3
20 x 5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
20 x 10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
20 x 20	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
50 x 5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
50 x 10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,01
50 x 20	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00	0,00	0,02
100 x 5	0,00	0,00	0,01	0,00	0,01	0,00	0,01	0,01	0,01	0,02
100 x 10	0,00	0,00	0,01	0,00	0,02	0,00	0,01	0,02	0,02	0,05
100 x 20	0,00	0,00	0,01	0,00	0,04	0,00	0,02	0,02	0,02	0,12
200 x 10	0,00	0,00	0,07	0,01	0,14	0,01	0,05	0,02	0,03	0,34
200 x 20	0,01	0,01	0,09	0,02	0,28	0,02	0,06	0,02	0,03	0,80
500 x 20	0,04	0,04	0,45	0,12	4,65	0,13	0,36	0,22	0,44	10,27
Média	0,00	0,01	0,05	0,01	0,43	0,01	0,04	0,03	0,05	0,97
TRPM	-0,98	-0,92	-0,39	-0,81	0,52	-0,61	0,09	-0,34	0,29	3,13

O método FRB3 possui um custo computacional maior do que outras heurísticas. Enquanto que o método NEH demora 30 milissegundos para executar um problema de 500 tarefas e 20 máquinas, a heurística FRB3 consome aproximadamente 11 segundos. Por meio

da análise do indicador TRPM, o método FRB3 é o menos eficiente, com um TRPM de 3,13, seguido pelo método INEH KK2D com um TRPM de 0,29.

Como pode ser observado na Tabela 3, os métodos NEH (tempo médio de 0,00s), NEH NM (0,01s), NEH KK (0,05s), NEH KK1 (0,01s), NEH D (0,03s) NEH KK2(0,01s), NEH FF (0,04s), INEH KK2D (0,05s) apresentam um tempo médio consideravelmente menor do que o obtido pelo método CL_{WTS} (0,43s). A heurística CL_{WTS} realiza uma busca na vizinhança por permutação, na qual não é possível utilizar o método de aceleração, exigindo o cálculo do *makespan* por meio da Equações 1. A Figura 22 apresenta um gráfico com o tempo computacional médio e TRPM das heurísticas comparadas.

Figura 22 - Tempo computacional médio e TRPM das heurística nos problemas testes do Taillard.

Vale ressaltar que, mesmo possuindo um tempo significativamente maior, o tempo computacional médio do FRB3 é razoável em termos práticos, demandando em média 11 segundos para executar cada um dos problemas de 500 tarefas e 20 máquinas, não inviabilizando sua utilização.

Em vista das análises realizadas, pode ser afirmado que o método construtivo FRB3 é superior as heurísticas comparadas. O método fornece um ganho substancial em qualidade de solução a um custo computacional adicional razoável. Sendo assim, os novos métodos propostos para o problema com critério de minimização do *makespan* foram comparados ao método FRB3.

Devido ao grande número de heurísticas propostas, foram selecionadas sete heurísticas para uma análise mais aprofundada. Dentre as várias formas de seleção de pares (G1, G2, ..., G15), o método G8 foi que obteve melhores resultados. Este método foi então combinado com mecanismos de desempate e ordenações iniciais. Dentre estes métodos gerados por meio de combinação (F1, F2, ..., F52), os seis melhores métodos foram selecionados para avaliação. O critério de escolha foi o menor DRM obtido pelos métodos em relação a todos os problemas testes do Taillard. De forma resumida, a Tabela 4 apresenta uma lista em ordem crescente de DRM das heurísticas implementadas no conjunto de problemas testes do Taillard.

Tabela 4 – Heurísticas construtivas em ordem crescente de DRM nos problemas testes do Taillard.

*	Heurística	DRM	Tempo Médio	*	Heurística	DRM	Tempo Médio	*	Heurística	DRM	Tempo Médio
1	F13	1,41	1,92	27	F12	1,53	1,94	53	F52	1,59	1,99
2	F34	1,43	2,25	28	F35	1,53	2,25	54	F37	1,60	2,08
3	F6	1,44	2,00	29	F19	1,53	1,99	55	F10	1,60	1,94
4	F31	1,44	2,34	30	F50	1,53	2,07	56	F11	1,60	1,93
5	F7	1,45	2,04	31	F4	1,53	2,04	57	F39	1,61	2,07
6	F9	1,45	2,03	32	F36	1,53	2,09	58	F41	1,62	2,06
7	F2	1,45	1,92	33	G10	1,54	2,10	59	F40	1,62	2,03
8	F24	1,45	1,94	34	F5	1,54	2,03	60	G11	1,65	1,96
9	F29	1,47	2,22	35	G9	1,54	2,03	61	G5	1,68	2,03
10	F8	1,48	2,03	36	F38	1,54	2,06	62	G6	1,69	1,06
11	F1	1,48	1,92	37	F28	1,55	2,23	63	G12	1,74	2,34
12	F23	1,48	1,95	38	G1	1,56	2,08	64	G13	1,74	2,32
13	F42	1,49	2,04	39	F26	1,56	2,24	65	G14	1,76	2,49
14	F15	1,49	1,91	40	F32	1,57	2,31	66	G3	1,76	1,06
15	F46	1,50	2,10	41	F30	1,58	2,34	67	G15	1,79	0,00
16	G8	1,50	1,90	42	F33	1,58	2,28	68	FRB3	1,81	0,97
17	F22	1,50	1,97	43	F3	1,58	1,92	69	INEH KK2D	2,63	0,05
18	F18	1,50	2,00	44	F25	1,58	1,94	70	NEH D	3,12	0,03
19	F45	1,50	2,08	45	G2	1,58	2,07	71	NEH FF	3,25	0,04
20	F49	1,50	2,08	46	F43	1,59	2,03	72	NEH KK2	3,26	0,01
21	F21	1,51	1,99	47	F17	1,59	2,01	73	CL _{WTS}	3,32	0,43
22	F51	1,51	2,06	48	F48	1,59	2,11	74	NEH KK	3,36	0,05
23	F20	1,51	2,00	49	G7	1,59	2,00	75	NEH KK1	3,36	0,01
24	F47	1,51	2,06	50	G4	1,59	2,08	76	NEH NM	3,45	0,01
25	F14	1,51	1,93	51	F16	1,59	1,99	77	NEH NM	3,50	0,01
26	F27	1,51	2,22	52	F44	1,59	2,06	78	NEH	3,54	0,00

* Colocação da Heurística

Com base na ordenação apresentada na Tabela 4, foram selecionadas as seis primeiras heurísticas (F13, F34, F6, F31, F7 e F9) e o método G8 para comparação com o método FRB3.

A heurística F6 utiliza a ordenação inicial e a forma desempate da heurística NEH KK1. A heurística F9 calcula ordenação inicial pela soma da média e desvio padrão dos tempos de processamento das tarefas, como faz o NEH D, e utiliza a forma de desempate empregada no método NEH KK2. Na heurística F13, foi implementada a ordenação inicial e o mecanismo de desempate da heurística NEH KK2, sendo o mecanismo de desempate apenas aplicado na inserção do NEH e não na fase de reconstrução. A heurística F13 aplica a ordenação inicial do NEH D e os desempates NEH D e NEH KK1, respectivamente nas fases de inserção e reconstrução. A heurística F34 apenas substitui na heurística F31 o desempate NEH KK1 pelo desempate utilizado no NEH KK2. Vale ressaltar que todas as heurísticas utilizam o mecanismo de construção da solução proposto pelo método G8. O Quadro 4 apresenta a composição das fases das heurísticas selecionadas. A Tabela 5 apresenta os resultados das heurísticas para os problemas testes do Taillard.

Quadro 4 - Composição das fases das heurísticas selecionadas para avaliação.

Método	Ordenação Inicial	Construção da Solução	Desempate	
			Inserção	Reconstrução
G8	LPT	G8	-	-
F6	KK1	G8	KK1	KK1
F7	KK2	G8	KK2	KK2
F9	D	G8	KK2	KK2
F13	KK2	G8	KK2	-
F31	D	G8	D	KK1
F34	D	G8	D	KK2

Tabela 5 - DRM para cada classe de problemas e média geral obtidas pelos métodos heurísticos nos problemas testes do Taillard.

Problema	FRB3	G8	F7	F9	F6	F31	F34	F13
20 x 5	1,13	0,87	0,96	0,87	0,57	0,94	0,94	0,84
20 x 10	2,50	1,75	1,46	1,68	1,80	1,23	1,23	1,28
20 x 20	1,97	1,52	1,58	1,30	1,27	1,11	1,11	1,09
50 x 5	0,24	0,14	0,22	0,16	0,26	0,33	0,36	0,21
50 x 10	2,25	1,96	1,73	2,05	2,09	2,19	2,11	2,23
50 x 20	5,41	4,80	4,76	4,75	4,73	4,54	4,54	4,54
100 x 5	0,17	0,12	0,06	0,11	0,10	0,21	0,15	0,21
100 x 10	1,05	0,79	0,65	0,66	0,74	0,76	0,78	0,70
100 x 20	3,68	3,14	3,35	3,17	3,10	3,21	3,08	3,22
200 x 10	0,47	0,45	0,24	0,27	0,24	0,49	0,45	0,27
200 x 20	2,06	1,71	1,73	1,79	1,74	1,77	1,82	1,73
500 x 20	0,76	0,70	0,63	0,60	0,65	0,55	0,57	0,65
Média	1,81	1,50	1,45	1,45	1,44	1,44	1,43	1,41

Dos resultados apresentados, o método G8 com uma média de 1,5% superou a heurística FRB3 com 1,81%. Dentre as heurísticas geradas por combinação, o método F13 apresentou os melhores resultados com uma média de 1,41%. Vale notar que a heurística FRB3 não fornece o menor DRM em nenhuma das classes de problemas. As Figuras 23 e 24 apresentam um gráfico com o DRM das heurísticas FRB3, G8 e F13 nos conjuntos de problemas testes agrupados, respectivamente, pelo número de tarefas e máquinas.

Figura 23 – DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes do Taillard.

Pode ser constatado que existe uma tendência da qualidade da solução melhorar conforme o número de tarefas aumenta, com exceção do agrupamento com 50 tarefas, onde há uma piora significativa. As heurísticas F13 e G8 superaram o método FRB3 em todos os agrupamentos de problemas.

Figura 24 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes do Taillard.

Na Figura 34, verifica-se que quanto maior o número de máquinas piores são os resultados. Os métodos G8 e F13 superam a heurística FRB3 para todos os grupos de problemas. Para problemas com cinco máquinas a diferença de DRM entre o método FRB3 e as heurísticas propostas é pequena; porém, para problemas com 10 e 20 máquinas a diferença aumenta consideravelmente. A Tabela 6 apresenta o desvio relativo médio porcentual em relação ao FRB3 obtido pelas heurísticas propostas.

Tabela 6 - DRP médio obtido pelas novas heurísticas propostas nos problemas testes de Taillard.

Problema	FRB3	G8	F7	F9	F6	F31	F34	F13
20 x 5	100,00	229,91	92,80	89,57	204,12	154,18	154,18	70,75
20 x 10	100,00	200,76	335,71	418,95	175,09	393,69	393,69	406,45
20 x 20	100,00	141,30	152,60	268,69	171,52	344,60	344,60	325,08
50 x 5	100,00	81,52	57,62	74,37	57,70	62,35	66,24	67,22
50 x 10	100,00	118,56	141,73	115,95	108,28	110,06	113,00	106,43
50 x 20	100,00	117,20	116,04	115,24	115,97	120,86	120,86	119,92
100 x 5	100,00	136,00	214,50	117,71	218,57	70,22	92,86	78,20
100 x 10	100,00	155,79	185,10	229,25	179,66	192,12	405,84	171,12
100 x 20	100,00	119,48	112,41	116,70	122,34	116,82	121,10	116,12
200 x 10	100,00	144,85	242,48	171,57	104,23	122,24	133,62	163,82
200 x 20	100,00	132,94	135,74	119,39	126,20	117,78	116,00	130,25
500 x 20	100,00	112,14	126,83	133,96	121,90	152,46	137,14	118,40
Média	100,00	140,87	159,46	164,28	142,13	163,11	183,26	156,15

As heurísticas propostas obtiveram um ganho considerável em qualidade de solução em relação ao FRB3, sendo o maior desvio relativo médio porcentual obtido pela heurística F34 de 83,26%. O método G8 obteve um ganho de aproximadamente 41%. Pode ser observado também que os métodos propostos, de forma geral, possuem um desempenho melhor para problemas com 20 tarefas, fornecendo soluções em média 142% superiores ao FRB3. Os intervalos das médias das heurísticas comparadas podem ser visualizadas na Figura 25.

Figura 25 – Gráfico com o intervalo das médias das novas heurísticas propostas com um nível de confiança de 95%.

Em termos estatísticos, os métodos propostos não são significativamente diferentes da heurística FRB3, apesar de apresentarem um desvio relativo médio consideravelmente menor. A Tabela 7 apresenta a análise do tempo computacional de cada heurística para os problemas testes do Taillard.

Tabela 7 -Tempo computacional dos métodos propostos nos problemas testes do Taillard.

Problema	FRB3	G8	F7	F9	F6	F31	F34	F13
20 x 5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
20 x 10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
20 x 20	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
50 x 5	0,00	0,00	0,00	0,01	0,01	0,01	0,01	0,01
50 x 10	0,01	0,01	0,01	0,01	0,01	0,02	0,02	0,01
50 x 20	0,02	0,03	0,03	0,03	0,03	0,04	0,03	0,03
100 x 5	0,02	0,04	0,04	0,05	0,05	0,06	0,06	0,04
100 x 10	0,05	0,10	0,10	0,10	0,10	0,12	0,12	0,09
100 x 20	0,12	0,22	0,22	0,22	0,21	0,25	0,24	0,22
200 x 10	0,34	0,66	0,73	0,72	0,71	0,95	0,91	0,65
200 x 20	0,80	1,54	1,60	1,57	1,55	1,74	1,71	1,54
500 x 20	10,27	20,19	21,75	21,68	21,31	24,84	23,93	20,50
Média	0,99	2,01	2,04	2,03	2,04	2,34	2,22	1,91
TRPM	-0,42	-0,18	0,03	0,02	0,15	0,24	0,23	-0,07

O tempo computacional médio dos métodos propostos é em média 110% maior do que o FRB3. Para problemas de 500 tarefas e 20 máquinas, enquanto que os métodos propostos consomem em média 22 segundos, o algoritmo FRB3 demanda um tempo médio aproximado de 11 segundos. O valor do TRPM também é menor para o método FRB3 com um valor de -0,42. A Figura 26 apresenta um gráfico com o tempo computacional médio e o TRPM dos métodos comparados.

Figura 26 - Tempo computacional médio e TRPM das heurísticas propostas nos problemas testes do Taillard.

Para uma análise global dos resultados, a Figura 27 apresenta um gráfico do intervalo das médias de todas as heurística selecionadas para comparação nos problemas testes do Taillard.

Figura 27 – Gráfico do intervalo das médias das heurísticas comparadas nos problemas testes do

Taillard com um grau de confiança de 95%.

Por meio das análises realizadas no conjunto de problemas testes do Taillard, é possível afirmar que os métodos que obtiveram melhor performance foram os algoritmos G8 e F13. Porém, em relação à heurística FRB3, em termos estatísticos, as heurísticas propostas não apresentaram diferença significativa. Por esses motivos, para extrair conclusões mais consolidadas dos resultados, foram selecionados os métodos NEH, FRB3, G8, F6, F7, F9, F13, F31 e F34 para serem avaliados no conjunto problemas testes VFR, recentemente proposto por Vallada, Ruiz e Framinan (2014).

O conjunto de problemas testes VFR consiste de 240 problemas testes de grande porte e 240 de pequeno porte. Os 240 problemas de pequeno porte são divididos nas seguintes combinações de número de tarefas (n) e número de máquinas (m): $n = \{10, 20, 30, 40, 50, 60\}$, $m = \{5, 10, 15, 20\}$. Em relação aos problemas de grande porte, também há um grupo de 240 problemas onde $n = \{100, 200, 300, 400, 500, 600, 700, 800\}$ e $m = \{20, 40, 60\}$. Cada combinação contém 10 problemas testes. A Tabela 8 sumariza os resultados obtidos pelas heurísticas nos problemas de pequeno porte do conjunto de problemas testes VFR.

Tabela 8 - DRM para cada classe de problemas e média geral obtidas pelos novos métodos propostos nos problemas testes do VFR.

Problema	NEH	FRB3	G8	F6	F13	F7	F31	F9	F34
10 x 5	2,18	0,59	0,34	0,49	0,18	0,05	0,37	0,50	0,25
10 x 10	1,83	0,30	0,68	0,84	0,68	0,77	0,73	0,74	0,73
10 x 15	1,41	0,89	0,78	0,61	0,70	0,70	0,10	0,08	0,10
10 x 20	1,77	0,07	0,19	0,19	0,19	0,19	0,30	0,30	0,30
20 x 5	1,45	0,86	0,21	0,50	0,71	0,64	0,47	0,72	0,37
20 x 10	3,65	1,35	1,14	1,27	0,95	1,18	0,50	1,11	0,50
20 x 15	3,48	0,94	0,89	0,62	1,02	1,05	0,99	0,69	0,75
20 x 20	3,41	1,33	0,50	0,84	0,36	0,40	0,65	0,90	0,68
30 x 5	1,33	0,38	0,30	0,16	0,25	0,23	0,23	0,21	0,20
30 x 10	3,46	1,10	1,18	0,56	0,89	0,69	0,88	0,74	0,92
30 x 15	4,10	1,05	1,13	0,86	0,94	0,72	0,66	0,79	0,76
30 x 20	3,56	1,48	0,89	0,85	0,93	0,44	0,58	0,54	0,58
40 x 5	1,07	0,15	0,07	0,05	0,06	0,07	0,05	0,11	0,07
40 x 10	3,31	0,96	0,84	0,70	0,51	0,53	0,78	0,31	0,67
40 x 15	3,89	1,21	0,77	0,48	0,95	0,86	0,63	0,48	0,53
40 x 20	3,20	1,35	0,94	0,89	0,75	0,86	0,53	0,54	0,53
50 x 5	0,54	0,22	0,09	0,01	0,00	0,00	0,07	0,01	0,12
50 x 10	3,57	1,21	0,65	0,47	0,56	0,58	0,51	0,26	0,73
50 x 15	4,17	1,64	0,91	0,77	0,88	0,64	0,64	0,78	0,39
50 x 20	3,67	1,03	0,81	0,79	0,69	0,76	0,47	0,57	0,50
60 x 5	0,89	0,21	0,00	0,07	0,01	0,02	0,10	0,08	0,03
60 x 10	3,10	0,97	0,58	0,35	0,44	0,42	0,39	0,52	0,41
60 x 15	3,70	1,30	0,85	0,56	0,67	0,67	0,59	0,48	0,52
60 x 20	3,98	1,08	0,55	0,60	0,80	0,60	0,50	0,36	0,51
Média	2,78	0,90	0,64	0,56	0,59	0,54	0,49	0,49	0,46

O método G8 obteve uma média de 0,64% contra 0,90% do método FRB3. O método G8 supera a heurística FRB3 em 21 das 24 subconjuntos de problemas. Para problemas de pequeno porte, a heurística F34 possui a menor média de 0,46%, superando o FRB3 em 22 dos 24 subconjuntos de problemas. As Figuras 28 e 29 apresentam um gráfico com o DRM dos problemas agrupados, respectivamente, pelo número de tarefas e máquinas.

Figura 28 – DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes de pequeno porte do VFR.

Pode-se notar que, com exceção do agrupamento com $n = 10$, a heurística G8 supera a heurística FRB3 em todos os grupos de problemas. O método F34 obteve a melhor performance, superando os métodos FRB3 e G8 em todos os conjuntos de problemas.

Figura 29 – DRM para cada subconjunto de problemas agrupados por

número de máquinas nos problemas testes de pequeno porte do VFR.

Na Figura 29, pode ser notado que as heurísticas G8 e F34 superam o método FRB3 para todos os agrupamentos de problemas por número de máquinas. O método F34 obteve os melhores resultados, principalmente para problemas com 15 e 20 máquinas. Observa-se também que quanto maior o número de máquinas maior a diferença entre o método FRB3 e as heurística G8 e F34. A Figura 30 apresenta um gráfico com o intervalo das médias das heurísticas no conjunto de problemas testes VFR de pequeno porte.

Figura 30 - Gráfico com os intervalos das médias das heurísticas nos problemas testes de pequeno porte do VFR com nível de confiança de 95%.

As heurísticas propostas são significativamente diferentes do método FRB3, visto que não há sobreposição entre os intervalos das média. Porém, entre os métodos propostos não há diferença significativa. Portanto, para os problemas de pequeno porte do VFR, pode ser concluído que os métodos propostos são claramente superiores ao método FRB3.

Os métodos propostos consumiram um tempo computacional maior do que a heurística FRB3 para o conjunto de pequeno porte de problemas do VFR, como pode ser visualizado na Tabela 9. A Figura 31 apresenta um gráfico com o tempo computacional médio e o TRPM dos métodos comparados no conjunto de problemas testes de pequeno porte do VFR.

Tabela 9 – Eficiência computacional das heurísticas para os problemas testes de pequeno porte do VFR.

Problema	NEH	FRB3	G8	F6	F13	F7	F31	F9	F34
10 x 5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
10 x 10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
10 x 15	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
10 x 20	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
20 x 5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
20 x 10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
20 x 15	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
20 x 20	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
30 x 5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
30 x 10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,01	0,00
30 x 15	0,00	0,00	0,00	0,00	0,01	0,00	0,01	0,01	0,01
30 x 20	0,00	0,00	0,01	0,00	0,01	0,01	0,01	0,01	0,01
40 x 5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,01	0,00
40 x 10	0,00	0,00	0,01	0,01	0,01	0,01	0,01	0,01	0,01
40 x 15	0,00	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
40 x 20	0,00	0,01	0,02	0,01	0,02	0,02	0,02	0,01	0,02
50 x 5	0,00	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
50 x 10	0,00	0,01	0,01	0,01	0,01	0,01	0,02	0,01	0,02
50 x 15	0,00	0,01	0,02	0,02	0,02	0,02	0,03	0,02	0,03
50 x 20	0,00	0,02	0,03120	0,03	0,03000	0,03	0,03	0,03	0,03
60 x 5	0,00	0,00	0,01	0,01	0,01	0,01	0,01	0,01	0,01
60 x 10	0,00	0,01	0,02	0,02	0,02	0,02	0,03	0,03	0,03
60 x 15	0,00	0,02	0,03	0,04	0,04	0,04	0,04	0,04	0,04
60 x 20	0,00	0,03	0,05	0,05	0,05	0,05	0,05	0,05	0,06
Médias	0,00	0,00	0,01	0,01	0,01	0,01	0,01	0,01	0,01
TRPM	-0,81	-0,54	-0,03	0,15	0,06	0,13	0,45	0,08	0,51

Figura 31 - Tempo computacional médio e TRPM das heurísticas no conjunto de problemas testes VFR de pequeno porte.

Como pode ser notado, as heurísticas propostas possuem um custo computacional maior do que o método FRB3. A Tabela 10 apresenta os resultados das heurísticas para problemas de grande porte do VFR.

Tabela 10 – DRM para cada classe de problemas e média geral obtidas pelos novos métodos propostos nos problemas testes de grande porte do VFR.

Problema	NEH	FRB3	G8	F13	F6	F7	F9	F31	F34
100 x 20	3,81	1,23	0,60	0,52	0,49	0,46	0,50	0,42	0,53
100 x 40	3,55	0,94	0,53	0,53	0,67	0,53	0,48	0,45	0,45
100 x 60	3,12	0,97	0,33	0,44	0,41	0,26	0,25	0,22	0,17
200 x 20	3,18	0,72	0,48	0,40	0,33	0,32	0,36	0,25	0,27
200 x 40	3,38	0,58	0,40	0,33	0,21	0,46	0,28	0,24	0,19
200 x 60	3,29	0,57	0,28	0,25	0,27	0,36	0,31	0,21	0,17
300 x 20	2,47	0,53	0,29	0,27	0,14	0,14	0,21	0,14	0,13
300 x 40	3,20	0,60	0,37	0,30	0,25	0,38	0,12	0,15	0,15
300 x 60	3,09	0,47	0,33	0,26	0,24	0,26	0,15	0,26	0,24
400 x 20	2,22	0,35	0,31	0,13	0,18	0,11	0,16	0,13	0,07
400 x 40	3,10	0,52	0,27	0,22	0,32	0,25	0,24	0,04	0,10
400 x 60	3,07	0,52	0,26	0,28	0,20	0,23	0,15	0,10	0,12
500 x 20	2,00	0,28	0,20	0,14	0,16	0,10	0,08	0,15	0,08
500 x 40	2,85	0,47	0,32	0,22	0,27	0,20	0,26	0,05	0,09
500 x 60	2,79	0,48	0,24	0,29	0,27	0,28	0,14	0,19	0,18
600 x 20	1,44	0,25	0,15	0,12	0,11	0,07	0,13	0,10	0,11
600 x 40	2,82	0,45	0,25	0,17	0,19	0,08	0,17	0,15	0,14
600 x 60	2,74	0,44	0,24	0,24	0,17	0,32	0,18	0,11	0,08
700 x 20	1,27	0,20	0,16	0,09	0,11	0,08	0,07	0,10	0,08
700 x 40	2,63	0,33	0,23	0,11	0,17	0,14	0,12	0,07	0,09
700 x 60	2,69	0,39	0,23	0,11	0,20	0,20	0,10	0,12	0,11
800 x 20	1,15	0,19	0,15	0,07	0,06	0,08	0,08	0,10	0,06
800 x 40	2,38	0,30	0,26	0,11	0,18	0,10	0,12	0,13	0,15
800 x 60	2,56	0,28	0,23	0,20	0,20	0,19	0,11	0,15	0,06
Média	2,70	0,50	0,30	0,24	0,24	0,23	0,20	0,17	0,16

Para problemas de grande porte do VFR, a heurística G8 obteve uma média de 0,30% contra 0,50% do método FRB3, superando-o em todos os subconjuntos de problemas. A heurística F34 obteve a menor média de 0,16%, superando as heurísticas G8 e FRB3. As Figuras 33 e 34 apresentam um gráfico com o DRM das heurísticas FRB3, G8 e F34 agrupado, respectivamente, pelo número de tarefas e máquinas dos problemas.

Figura 32 - DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes de grande porte do VFR.

Por meio da Figura 32, observa-se que as heurística G8 e F34 superam o método FRB3 em todos os agrupamentos de problemas. Existe também uma tendência da diferença de DRM entre o método FRB3 e as heurísticas G8 e F34 diminuir conforme o número de tarefas aumenta.

Figura 33 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes de grande porte do VFR.

Pode ser observado na Figura 33 que, novamente, o método FRB3 é superado pelas heurísticas G8 e F34 em todas os agrupamentos. Também nota-se que as heurísticas apresentam resultados mais estáveis nos problemas testes do VFR. A Figura 34 apresenta os intervalos de média de cada um dos métodos.

Figura 34 – Gráfico com os intervalos de média das heurísticas para os problemas de grande porte do VFR com nível de confiança de 95%.

Como pode ser notado, os métodos propostos obtiveram significância estatística em relação ao FRB3, evidenciando a superioridade dos métodos propostos nos problemas de grande porte do VFR.

O tempo computacional dos métodos propostos é aproximadamente 109% maior do que o método FRB3. O método FRB3 consome em média 31 segundos para executar cada problema, enquanto que as heurísticas propostas demandam em média 65 segundos, como pode ser visto na Tabela 11. A Figura 35 apresenta um gráfico com o tempo computacional médio e o TRPM dos métodos comparados no conjunto de problemas testes de grande porte do VFR.

Tabela 11 - Eficiência computacional médio para problemas de grande porte do VFR.

Problema	NEH	FRB3	G8	F13	F6	F7	F9	F31	F34
100 x 20	0,01	0,12	0,21	0,21	0,22	0,23	0,22	0,24	0,24
100 x 40	0,01	0,27	0,50	0,50	0,50	0,52	0,51	0,53	0,56
100 x 60	0,02	0,43	0,79	0,78	0,80	0,81	0,81	0,84	0,88
200 x 20	0,02	0,80	1,51	1,50	1,56	1,58	1,58	1,66	1,72
200 x 40	0,05	1,93	3,65	3,63	3,75	3,75	3,78	3,89	4,09
200 x 60	0,09	3,18	6,00	5,95	6,12	5,91	6,20	6,27	6,65
300 x 20	0,04	2,48	4,72	4,70	4,92	4,78	4,97	5,31	5,55
300 x 40	0,10	5,98	11,62	11,66	11,91	11,44	12,07	12,31	13,04
300 x 60	0,19	9,96	19,18	19,37	19,71	18,85	19,55	20,15	21,21
400 x 20	0,07	5,49	10,74	10,98	11,52	11,04	11,53	12,18	12,84
400 x 40	0,19	13,29	26,27	26,31	27,26	25,87	27,61	27,71	27,85
400 x 60	0,34	22,32	43,33	43,87	44,38	42,72	45,07	45,03	44,30
500 x 20	0,11	10,29	20,08	20,84	21,53	20,89	21,97	23,26	22,97
500 x 40	0,29	24,93	49,14	50,54	49,41	48,62	51,13	50,68	50,85
500 x 60	0,53	41,80	81,84	83,77	81,86	80,65	84,62	83,29	84,16
600 x 20	0,15	17,00	33,95	34,89	36,38	35,70	38,03	40,31	40,14
600 x 40	0,42	40,82	82,48	84,78	86,02	81,67	83,94	89,42	88,41
600 x 60	0,75	68,75	137,16	144,40	142,53	135,19	142,21	144,79	142,71
700 x 20	0,20	26,23	52,60	55,42	59,05	56,30	58,61	63,98	62,59
700 x 40	0,56	62,85	127,45	133,31	131,10	126,43	132,47	137,15	140,29
700 x 60	1,02	105,99	212,34	218,13	224,35	209,95	216,88	229,38	234,58
800 x 20	0,27	38,93	77,91	78,32	91,46	84,80	87,89	102,02	100,16
800 x 40	0,73	94,17	186,42	188,03	193,78	185,15	191,57	209,52	202,16
800 x 60	1,32	155,94	309,17	316,84	309,13	305,70	316,67	337,86	326,42
Médias	0,31	31,41	62,46	64,11	64,97	62,44	65,00	68,66	68,10
TRPM	-0,99	-0,41	0,15	0,16	0,19	0,16	0,20	0,26	0,28

Figura 35 - Tempo computacional médio e TRPM das heurísticas nos problemas testes do VFR de grande porte.

Com base nas análises computacionais e estatísticas realizadas é possível afirmar: (1) para os problemas testes do Taillard, o método proposto G8 superou o método FRB3 com um DRM de 1,50% contra 1,83% do FRB3, porém, não apresentou significância estatística para um nível de confiança de 95%; (2) neste conjunto de problemas, a heurística F13 obteve o melhor performance com um DRM de 1,41%; (3) nos problemas testes de pequeno porte do VFR, o método G8 superou a heurística FRB3 com um DRM 0,64% contra 0,90% e foi demonstrado que as médias são significativamente diferentes; (4) a heurística F34 forneceu as melhores soluções com um DRM de 0,46%; (5) para problemas testes de grande porte do VFR, o método G8 também superou a heurística FRB3 com um DRM de 0,30% contra 0,50%; (6) a melhor performance foi obtida pelo algoritmo F34 com um DRM de 0,16%; (7) para o conjunto de problemas testes do VFR, todos os métodos apresentaram diferença estatística significativa em relação ao FRB3; (8) apesar das heurísticas propostas possuírem um esforço computacional maior, há um ganho substancial em qualidade de solução, o que justifica o tempo adicional utilizado. Os resultados desta seção foram sumarizados na Tabela 12.

Tabela 12 – Resultados consolidados.

Banco de Problemas	NEH	FRB3	G8	F7	F9	F6	F31	F34	F13
Taillard	3,54*	1,81	1,50	1,45	1,45	1,44	1,44	1,43	1,41
	(0,00)**	(0,99)	(2,00)	(2,04)	(2,03)	(2,03)	(2,33)	(2,22)	(1,91)
VFR pequeno porte	2,78	0,90	0,64	0,54	0,49	0,56	0,49	0,46	0,59
	(0,00)	(0,00)	(0,01)	(0,01)	(0,01)	(0,01)	(0,01)	(0,01)	(0,01)
VFR grande porte	2,70	0,50	0,30	0,23	0,20	0,24	0,17	0,16	0,24
	(0,31)	(31,4)	(62,4)	(62,4)	(64,9)	(64,9)	(68,6)	(68,0)	(64,1)

*DRM **Tempo médio em segundos

3.6 CONSIDERAÇÕES FINAIS DO PROBLEMA $F | pmu | C_{MAX}$

Nesta seção, foi realizada uma revisão das heurísticas para o problema de programação *flow shop* com critério de minimização do *makespan*. A revisão avaliou as extensões existentes da heurística NEH, que exploram novas formas de ordenação inicial, mecanismos de desempate das sequências candidatas e outras formas de construção da solução. Com os conhecimentos obtidos, um novo método foi proposto e foi combinado com procedimentos de ordenação inicial e desempate procedentes da literatura. Os problemas testes do Taillard e do VFR foram utilizados. Todas as heurísticas foram programadas na mesma linguagem e foram testadas na mesma plataforma computacional, portanto, os resultados são inteiramente comparáveis. Além disso, foram utilizados testes de significância estatísticas por meio da interpretação de gráficos de intervalos de média.

A partir das comparações iniciais realizadas foi constatado que o método FRB3 é o que oferece melhor performance entre os principais métodos da literatura. Por meio de análises, foi demonstrado que o método proposto G8 superou a heurística FRB3 e que as heurísticas F13 e F34, geradas por combinação com outros procedimentos, melhoram consideravelmente a qualidade das soluções. No entanto, os algoritmos propostos requerem um esforço computacional superior ao FRB3. Porém, a melhoria na qualidade de solução justifica o tempo adicional utilizado. Concluindo, o método apresentado pode ser considerado importante para o estado da arte para os métodos construtivos para o problema *flow shop* permutacional com critério de minimização do *makespan*.

Na próxima seção, será abordado o segundo problema objeto de estudo desta dissertação: o problema de programação da produção em ambiente *flow shop* permutacional com critério de minimização do *total flowtime* ($F_m | pmu | \sum C_j$).

4 PROBLEMA DE PROGRAMAÇÃO *FLOW SHOP* COM CRITÉRIO DE MINIMIZAÇÃO DO *TOTAL FLOWTIME*

O segundo problema a ser estudado nesta dissertação é o problema *flow shop* permutacional com critério de minimização do *total flowtime*, que leva a minimização do estoque em processo (RAJENDRAN, ZIEGLER, 1997). Nesse sentido, no ambiente dinâmico atual, este objetivo é considerado relevante e significativo para tornar as empresas mais competitivas. (LIU, REEVES, 2001).

4.1 DEFINIÇÃO DO PROBLEMA

O problema em consideração pode ser definido da seguinte maneira. Seja $\pi = \{ J_1, J_2, \dots, J_j, \dots, J_n \}$ uma sequência com um conjunto de n tarefas que devem ser processadas, na mesma ordem, por um conjunto de m máquinas distintas. O tempo de processamento da tarefa j na máquina i é p_{ij} ($j = 1, 2, \dots, n; i = 1, 2, \dots, m$). Se uma determinada tarefa não tiver operação em uma certa máquina, seu correspondente tempo de processamento é assumido como igual a zero. Seja C_{ij} o tempo de conclusão da tarefa j na máquina i , problema em questão consiste em minimizar o valor do *total flowtime* (TFT). O valor de C_{ij} pode ser calculado de forma recursiva por meio da Equação 34 e o valor do *total flowtime* é fornecido pela Equação 35. Na teoria que estuda a complexidade dos problemas de natureza combinatória, o problema em questão é classificado como NP-hard (GONZALES; SAHNI, 1978), de forma que podem ser resolvido eficientemente de maneira ótima somente em casos de pequeno porte.

$$\begin{aligned} C_{ij} &= \max(C_{i-1j}, C_{ij-1}) + p_{ij} \\ (i &= 1, \dots, n) \quad (j = 1, \dots, m) \\ C_{0j} &= C_{i0} = 0 \end{aligned} \tag{34}$$

$$TFT = \sum_{j=1}^n C_{mj} \quad (35)$$

Na próxima seção serão revisadas as principais heurísticas presentes na literatura para o problema *flow shop* permutacional com critério de minimização do *total flowtime*.

4.2 REVISÃO DA LITERATURA

No problema de minimização do *total flowtime*, as heurísticas são classificadas entre simples e compostas. Segundo Liu e Reeves (2001), uma heurística composta integra uma heurística construtiva a algum procedimento de busca local. No entanto, o termo “heurística composta” não ficou totalmente claro (FRAMINAN; LEISTEN; RAJENDRAN, 2005). Sendo assim, com base na composição das fases de um método construtivo proposta por Framinan, Gupta e Leisten. (2004), as quais são, respectivamente, desenvolvimento de um índice, construção da solução e melhoria da solução, Framinan, Leisten e Rajendran (2005) estenderam a classificação de Liu e Reeves (2001). De acordo com Framinan, Leisten e Rajendran (2005), o método é também classificado como composto se este usa uma heurística simples em uma ou mais das fases citadas, como, por exemplo, um método composto que utiliza uma heurística simples para desenvolver o índice inicial e outra para construir a solução. Em síntese, um método composto pode ser definido como aquele que possui uma heurística construtiva integrada a um procedimento de busca, bem como um método que emprega uma heurística simples em uma das três fases que o compõem. A revisão da literatura, portanto, se dividirá entre heurísticas simples e compostas. Na Figura 36, é possível visualizar a diferença entre as heurísticas simples e compostas.

Acrônimo	Classificação	Fase I	Fase II	Fase III
Heurística A	Simples	LPT*	Construtivo A	
Heurística B	Simples	SPT**	Construtivo B	
Heurística C	Composta	Heurística A	Construtivo B	
Heurística D	Composta	LPT*	Construtivo A	Busca Local

*SPT: *Shortest Processing Time*

****LPT: Longest Processing Time**

Figura 36 – Classificação das heurísticas.

Por meio da Figura 36, observa-se que a Heurística A e B são métodos simples, porque utilizam a ordenação LPT e SPT, respectivamente, na primeira fase e aplica um mecanismo de construção da solução na segunda fase. Já a Heurística C é composta, porque utiliza a Heurística A para gerar a ordenação inicial na primeira fase. A Heurística D, apesar de não utilizar nenhum método simples em sua composição, é um método composto, porque executa um procedimento de busca local na terceira fase do método.

4.2.1 Heurísticas simples

Rajendran e Chaudhuri (1991) desenvolveram três heurísticas que obtiveram melhor desempenho em termos da qualidade de solução e eficiência computacional que os métodos heurísticos propostos por Gupta (1972) e Miyazaki, Nishiyama e Hashimoto (1978), que é baseado em trocas adjacentes de pares de tarefas.

Na continuidade de suas pesquisas, Rajendran (1993) propôs um novo método que apresentou melhor desempenho do que as heurísticas acima mencionadas em termos de qualidade da solução, porém com alto esforço computacional. A heurística, denotada de Raj, realiza procedimentos de inserção muito similar a heurística NEH. O método ordena as tarefas conforme uma ponderação do tempo total de processamento e insere as tarefas em um número limitado de posições na sequência parcial.

Ho (1995) propôs uma heurística de diferentes iterações no processo de melhoria de uma solução inicial, a partir da obtenção de um ótimo local por permutação de pares de tarefas adjacentes, melhorando posteriormente a solução por movimentos de inserção de tarefas. Tal método heurístico apresentou desempenho melhor que os anteriores reportados na literatura, embora com desvantagem quanto ao tempo de computação, pelo fato de não ser uma heurística construtiva, apresentando semelhanças com metaheurísticas como *Simulated Annealing* e Busca Tabu.

Rajendran e Ziegler (1997) propuseram a heurística RZ, que consiste de duas fases: na primeira, uma sequência inicial é gerada utilizando uma regra de prioridade similar à

shortest weighted total processing time; na segunda fase, a sequência inicial obtida é melhorada por meio de inserções das tarefas em sequências parciais, sucessivamente obtidas. A complexidade da heurística é de $O(n^3m)$.

Wang, Chu e Proth.(1997) desenvolveram duas heurísticas denominadas LIT (*less idle time*) e SPD (*smallest process distance*). A primeira heurística foca em reduzir o tempo ocioso entre as máquinas, enquanto que a segunda trabalha tanto na redução do tempo ocioso entre máquinas quanto na minimização do tempo de espera das tarefas. Ambas as heurísticas propostas por Wang, Chu e Proth (1997) possuem complexidade de $O(n^2m)$. Suas heurísticas não são comparadas com as existentes, mas sim com um limitante inferior do tempo médio de fluxo, proposto por Ahmadi e Bagchi (1990).

Woo e Yim (1998) desenvolveram uma heurística denominada WY, comparando seu desempenho com a heurística de Rajendran (1993), e também com adaptações do NEH (NAWAZ; ENSCORE; HAM, 1983) e CDS (CAMPBELL; DUDEK; SMITH, 1970) para o critério de minimização do tempo médio de fluxo. A heurística WY, ao contrário dos métodos anteriores, não necessita de uma ordenação inicial. No entanto, o método possui uma fase de inserção em que as tarefas são inseridas em todas as posições possíveis da sequência parcial. A heurística é baseada no método NEH, mas sua complexidade é de $O(n^4m)$. Eles verificaram que seu método apresentava desempenho superior aos outros, principalmente quando comparado ao de Rajendran (1993).

Liu e Reeves (2001) desenvolveram uma heurística construtiva, chamada de LR, que inicialmente ordena as tarefas de acordo com um índice chamado ξ_{jk} que considera o tempo ocioso nas máquinas e os efeitos da inserção da tarefa nos tempos de conclusão das tarefas posteriores. A heurística LR não fixa o número de sequencias a serem geradas e, portanto, é flexível em relação ao seu tempo computacional, podendo este ser ajustado de acordo com o problema. Para cada tarefa $j \in U$, ξ_{jk} é calculado por meio da Equação 36.

$$\xi'_{jk} = (n - k - 2) \cdot IT'_{jk} + AT'_{jk} \quad (36)$$

Os valores de AT'_{jk} e IT'_{jk} são obtidas pela Equações 37 e 38, respectivamente.

$$IT'_{jk} = \sum_{i=2}^m \frac{m \cdot \max\{C_{i-1,j} - C_{i,[k]}, 0\}}{i + k \cdot (m - i)/(n - 2)} \quad (37)$$

$$AT'_{jk} = C_{m,j} + C_{m,p} \quad (38)$$

Framinan, Leisten e Ruiz-Usano (2002) propôs modificar a heurística NEH a fim de adaptá-la ao critério de minimização do *total flowtime*, foi provado que a heurística NEH com ordenação inicial de acordo com a ordem não decrescente (ao invés de não crescente) das somas dos tempos de processamento das tarefas nas máquinas supera a adaptação da heurística NEH original ao problema. Em termos de qualidade das soluções, o procedimento proposto por Framinan, Leisten e Ruiz-Usano (2002) ficou muito próximo à heurística WY, mas com um menor custo computacional. Mais tarde, Framinan, Leisten e Rajendran (2003) estudaram extensamente a ordenação inicial do NEH e propôs um método chamado B5FT, o qual superou as heurísticas RZ e WY.

Outra heurística foi proposta por Framinan e Leisten. (2003) chamada FL, com a mesma complexidade do método WY. O método prioriza as tarefas em ordem não decrescente das somas dos tempos de processamento das tarefas. Assim, após cada iteração de inserção do método NEH, a sequência parcial obtida é melhorada por meio de um procedimento de troca de pares de tarefas. Se resultados melhores são obtidos, a nova sequência parcial é mantida como a sequência parcial corrente. Os resultados reportaram que a heurística supera os métodos RZ e WY.

Li e Wu (2005) propuseram uma melhoria na heurística RZ, denotada de RZ-LW, em que os autores geram uma sequência inicial por meio da ordenação das tarefas em ordem não decrescente das somas dos tempos de processamento, e então realizam a busca local RZ na solução até que nenhuma melhoria seja realizada. Os resultados são comparáveis aos obtidos por Framinan e Leisten (2003), mas com tempo computacional bem menor.

Laha e Sarin (2009) apresentaram uma modificação da heurística FL, chamada de FL-LS. O método implementa uma iteração de inserção na fase da heurística NEH por meio da reinserção de tarefas ao invés de troca de pares (presente na heurística FL). Os autores provaram, por meio de experimentos computacionais, que a modificação melhora significativamente a performance da heurística FL enquanto que sua complexidade não é alterada.

Pan e Ruiz (2013) estudaram e avaliaram extensivamente várias heurísticas simples e compostas para o problema de minimização do *total flowtime*. Os autores propuseram uma heurística simples para o problema, chamada de LR-NEH(x). A heurística primeiramente gera uma sequência parcial de d tarefas usando o LR(x), e então o restante das $n-d$ tarefas são inseridas na sequência parcial por meio da heurística NEH. A posição relativa das tarefas fornecida pelo método LR(x) não é alterada conforme a sequência é gerada. A heurística LR-NEH(x) tem um único parâmetro d , que é basicamente o número de tarefas sequenciadas pelo LR(x) antes de que o NEH seja iniciado. Pan e Ruiz (2013) afirmam que dentre as heurísticas simples o melhor método em termos de qualidade de solução é o FL-LS de Laha e Sarin (2009). No entanto, os autores notam que o tempo de computação requerido pelo FL-LS é relativamente alto. Pan e Ruiz (2013) também concluem que o método proposto LR-NEH(x) apresenta um bom *trade-off* entre tempo de computação e qualidade da solução, superando a maior parte das outras heurísticas.

Fernandez-Viagas e Framinan (2015) apresentaram uma heurística simples denominada FF-(x). A heurística é semelhante à LR(x) de Liu e Reeves (2001), adicionando as tarefas, uma a uma, no final da sequência de acordo com um índice chamado ξ'_{jk} que é baseado no tempo ocioso e no *makespan* da tarefa inserida. A complexidade do método proposto é de $O(x \cdot n^2)$, menor do que o método LR. Fernandez-Viagas e Framinan (2015) também integram o procedimento à várias outras heurísticas, substituindo o procedimento LR(x) pela nova forma de avaliação das tarefas presente na heurística FF(x). Para cada tarefa $j \in U$, ξ'_{jk} é calculado por meio da Equação 39. Os autores determinaram experimentalmente que os valores $a = 4$ e $b = 1$ resultam em melhores soluções.

$$\xi'_{jk} = \frac{(n - k - 2)}{a} IT'_{jk} + AT'_{jk} \quad (39)$$

Os valores de AT_{jk} e IT_{jk} são calculados pelas Equações 40 e 41, respectivamente.

$$IT'_{jk} = \sum_{i=2}^m \frac{m \cdot \max\{C_{i-1,j} - C_{j,[k]}, 0\}}{i - b + k \cdot (m - i + b) / (n - 2)} \quad (40)$$

$$AT'_{jk} = C_{m,j} \quad (41)$$

Para as heurísticas simples, de acordo com a revisão da literatura efetuada na pesquisa relatada neste trabalho, os métodos simples LR-NEH(x), FF-NEH(x) e FL-LS são os melhores em termos de qualidade da solução para a programação em ambiente *flow shop* permutacional para a minimização do tempo médio de fluxo.

4.2.2 Heurísticas compostas

Liu e Reeves (2001) propuseram um mecanismo de melhoria baseado na permutação de pares de tarefas. Este procedimento é conhecido como *forward pairwise exchange* (FPE). A versão invertida que verifica as permutações da direita para a esquerda na sequência é chamada de *backward pairwise exchange* (BPE). Os autores estudaram a efetividade de diferentes combinações de sua heurística simples com essas buscas locais, chamadas de LR(x)-FPE e LR(x)-BPE, respectivamente. O resultado foi que os métodos compostos são superiores aos simples a um custo computacional maior.

Allahverdi e Aldowaisan (2002) apresentaram sete heurísticas denominadas IH x ($x = 1, 2, \dots, 7$), que combinam os métodos NEH, WY e RZ com os procedimentos de busca local da heurística RZ. O melhor desempenho foi alcançado pela heurística IH7, que é constituída de três fases. Na primeira fase, uma solução inicial é obtida aplicando-se a heurística WY (WOO e YIM, 1998). Esta primeira solução é utilizada na segunda fase como sequência inicial para a heurística RZ (RAJENDRAN e ZIEGLER, 1997). Finalmente, a segunda solução é melhorada por meio de uma busca local, com procedimentos de permutação nas posições das tarefas.

Framinan e Leisten (2003) propuseram o método denominado FL-IH7, que consiste em uma extensão do IH7 proposta por Allahverdi e Aldowaisan (2002). A diferença relevante situa-se na obtenção da solução inicial do processo de três fases do IH7. No FL-IH7 tal solução inicial é obtida por um método construtivo que utiliza, inicialmente, o mesmo procedimento de obtenção de sequências parciais do NEH (Nawaz, Ensore e Ham, 1983), porém ordenando as tarefas de acordo com a soma dos tempos de processamento em ordem não decrescente. A seguir, as sequências parciais são melhoradas por um procedimento de

busca, chamado FPE-R, nas respectivas vizinhanças de permutação. Por meio de uma experimentação computacional constatou-se que o FL-IH7 apresenta um desempenho melhor quando comparado com IH7 original.

Framinan, Leisten e Ruiz-Usano (2005) apresentou uma extensa comparação das recentes heurísticas para o problema. Foram propostas duas heurísticas compostas, C1 e C2. O método C1 integra a heurística LR com o mecanismo construtivo do método FL. Já a heurística composta C2 integra ao método C1 à busca na vizinhança FIE-R (*foward insertion exchange-restart*). Uma destas novas heurísticas, chamada de C2-FL, apresentou melhores soluções do que os métodos propostos por Framinan, Leisten e Rajendran (2003).

Nagano e Moccasin (2008), apresentaram um método composto, chamado NM *total flowtime*, que forneceu melhores soluções do que a heurística C2 em um tempo computacional menor. A heurística NM-*total flowtime*, integra os procedimentos do NEH com uma busca local híbrida de permutação e inserção.

Mais recentemente Li, Wang e Wu (2009) apresentaram três heurísticas compostas, chamadas IC1, IC2 e IC3, respectivamente, integrando as buscas locais FPE, FPE-R e RZ junto a um método iterativo eficiente que se repete até que não se melhore a solução ou até que se satisfaça um dado critério de parada. Os resultados computacionais mostram que os três algoritmos propostos superam as heurísticas C1-FL e C2-FL de Framinan, Leisten e Ruiz-Usano (2005) e a FL-IH7 de Framinan e Leisten (2003). Dentre essas heurísticas, IC3 desempenhou melhor em termos de qualidade de solução mas necessita de muito mais tempo computacional do que ambas IC1 e IC2. Em um trabalho relacionado, Li e Wang (2006) divulgaram duas heurísticas compostas chamadas ECH1 e ECH2, que são similares as heurísticas IC1, IC2 e IC3.

Por fim, Pan e Ruiz (2013) propuseram as heurísticas compostas PR1(x), PR2(x), PR3(x) e PR4(x), em que x delimita o número de sequências construídas na primeira fase do método.

A heurística PR1(x) integra os métodos LR-NEH(x) e a busca local proposta por Rajendran e Ziegler (1997), chamada de iRZ. Este algoritmo de busca local é baseado em um procedimento de inserção na vizinhança. No procedimento PR1(x) cada uma das soluções geradas pelo método LR-NEH(x) é melhorada pelo método de busca local iRZ.

O procedimento PR2(x) simplesmente troca a busca local iRZ pelo algoritmo de busca variável na vizinha (VNS), que é uma metaheurística eficiente proposta por Mladenovic

e Hansen (1997). Tasgetiren *et al.* (2007) propuseram uma busca local baseada na inserção e permutação do método VNS e integraram-na a um método de otimização de enxame de partículas para resolver os problemas de *flow shop* permutacional com os critérios de minimização do *makespan* e do *total flowtime*. No procedimento VNS, a permutação de pares de tarefas seleciona aleatoriamente duas tarefas da sequência corrente e troca suas posições. O movimento de inserção remove aleatoriamente uma tarefa de sua posição original e insere-a em outra posição aleatória da sequência.

A heurística PR3(x) primeiramente gera uma solução por meio do método LR-NEH(x) e então melhora esta solução por meio de um procedimento de melhoria proposto pelos autores, que integra as buscas locais iRZ e NEH, o qual refere-se neste trabalho por iRZ-NEH.

Por fim, na heurística PR4(x) ao invés de utilizar-se o método iRZ no procedimento de busca proposto pelos autores, usa-se o método VNS, este procedimento é referenciado neste trabalho por VNS-NEH. Os autores afirmam que as heurísticas compostas propostas são superiores as existentes até o momento.

Fernandez-Viagas e Framinan (2015) trocaram a heurística LR(x) presente no métodos composto PR1(x) pelo método proposto FF(x). Esta alteração melhorou ligeiramente os resultados, enquanto que o tempo computacional foi reduzido. Este método recebeu no nome de FF-PR1(x).

Para as heurísticas compostas, de acordo com a revisão da literatura efetuada na pesquisa relatada neste trabalho, os métodos compostos PR1(x)-PR4(x) são os melhores em termos de qualidade da solução para a programação em ambiente *flow shop* permutacional para a minimização do *total flowtime*.

O Quadro 5 apresenta um resumo das principais heurísticas simples e compostas para o problema *flow shop* com critério de minimização do *total flowtime*. Alguns dos termos referenciados no Quadro 5 serão explicados abaixo.

Em relação à Fase I, têm-se as seguintes formas de ordenação inicial:

- SPT: Shortest Total Processing Time. Consiste em ordenar as tarefas em ordem não decrescente da soma dos tempos de processamento;
- SWPT: Shortest Weighted Total Processing Time. Consiste em ordenar as tarefas em ordem não decrescente da soma ponderada dos tempos de processamento;

- Alguns métodos compostos utilizam heurísticas para gerar a ordenação inicial, como, por exemplo, o método ICH1, que utiliza a heurística LR para gerar a ordenação inicial, que posteriormente é melhorada por um método iterativo de melhoria.

Quanto à Fase II, têm-se os seguintes procedimentos de construção:

- NEH-C: este método de geração da sequência foi proposto por Nawaz, Enscore JR. & Ham (1983). A cada iteração do procedimento realiza-se a inserção da k -ésima tarefa da ordenação inicial nas $k+1$ possíveis posições da sequência parcial, gerando $k+1$ sequências candidatas que são avaliadas, selecionando a melhor segundo o critério utilizado. O processo continua até que se tenha uma sequência completa de n tarefas;
- RZ-C: este método de melhoria da sequência está presente na segunda fase da heurística RZ de Rajendran e Ziegler (1997). O procedimento consiste na inserção sequenciada de cada uma das tarefas da sequência original em cada uma das posições possíveis da sequência;
- WY: este é o procedimento de construção da sequência desenvolvido por Woo e Yim (1998), que, baseado no método NEH, insere cada uma das tarefas que ainda não foram sequenciadas em todas as posições da sequência parcial, escolhendo a sequência candidata que fornece o menor *total flowtime*;
- FL-C: este mecanismo de construção está presente na heurística FL de Framinan e Leistein (2003). O método consiste em realizar um procedimento de permutação de tarefas na sequência corrente após cada iteração do método NEH-C;
- iRZ-C: este procedimento iterativo de melhoria é derivado da segunda fase do método RZ de Rajendran e Ziegler (1997). O método é análogo ao mecanismo de melhoria RZ-C, com a diferença de ser reiniciado quando uma solução melhor é encontrada. O procedimento é iterado até que um determinado número de tentativas sem melhoria sejam feitas;
- FL-LS-C: este método de construção da sequência está presente na heurística FL-LS de Laha e Sarin (2009). O procedimento é derivado do método FL de Framinan e

Leistein (2003), possuindo a diferença de se realizar um procedimento de inserção de tarefas ao invés de permutação;

- LR(x)-NEH: esta heurística integra os procedimentos de construção da sequência das heurísticas LR e NEH e foi proposta por Pan e Ruiz (2013). Neste procedimento parte da sequência é construída pela heurística LR e parte é gerada pelo NEH.

Na Fase III, os seguintes procedimentos de melhoria são aplicados:

- FPE: *foward pairwise exchange*. Este é um mecanismo de melhoria de uma sequência inicial que tenta melhora-la pela troca de cada uma das tarefas com um certo número de tarefas posteriores a estas na sequência. Para cada tentativa, se uma sequência melhor é obtida, a troca é efetivamente realizada. Em caso contrário, as duas tarefas permutadas permanecem nas suas posições originais. Depois que todas as permutações são avaliadas, o processo recomeça do início da sequência, e o procedimento continua até que nenhum tipo de melhoria seja feita por um número determinado de tentativas;
- BPE: *backward pairwise exchange*. Esta é a versão invertida do método FPE que verifica as permutações da direita para a esquerda na sequência;
- FPE-R: *foward pairwise exchange - restart*. Este procedimento é idêntico ao método FPE, mas quando uma troca é efetivamente realizada pelo nova sequência ser melhor que a corrente, o processo de permutação recomeça da primeira tarefa da sequência;
- FIE-R: *foward insertion exchange - restart*. Este procedimento é análogo ao método FPE, utilizando a inserção de tarefas ao invés da permutação, reiniciando o método a cada melhoria realizada;
- NM: este é um procedimento híbrido de melhoria que utiliza inserção juntamente com permutação para melhorar a sequência, e está presente na heurística composta de Nagano e Moccellini (2008);
- VNS: *Variable Neighborhood Search*. Esta é uma busca na vizinha baseada na inserção e permutação de tarefas da metaheurística VNS;

- iRZ-NEH: este método de melhoria proposto por Pan e Ruiz (2013) trabalha com os procedimentos de buscas locais iRZ e NEH (versão de melhoria iterativa do método);
- VNS-NEH: este é um procedimento de melhoria proposto por Pan e Ruiz (2013) que integra os procedimentos de buscas locais VNS e NEH, e é similar ao método iRZ-NEH.

Quadro 5 – Resumo da composição das heurísticas simples e compostas

Ano	Autores	Acrônimo	Classificação	Fase I	Fase II	Fase III
1997	Rajendran e Ziegler	RZ	Simples	SWPT	RZ-C	
1998	Woo e Yim	WY	Simples		WY	
2001	Liu e Reeves	LR(x)	Simples		LR(x)	
		LR(x)-FBE	Composta		LR(x)	FPE
		LR(x)-BPE	Composta		LR(x)	BPE
2002	Framinan et al.	NEH-flowtime	Simples	SPT	NEH-C	
2002	Framinan e Leisten	IH1	Composta	NEH-flowtime		FPE-R
		IH2	Composta	NEH-flowtime(melhor de 5 tentativas)		
		IH3	Composta	NEH-flowtime(melhor de 5 tentativas)		
		IH4	Composta	WY		FPE-R
		IH5	Composta	RZ		FPE-R
		IH6	Composta	WY	RZ-C	
		IH7	Composta	WY	RZ-C	FPE-R
2003	Framinan e Leisten	FL	Simples	SPT	FL-C	
		IH7-FL	Composta	FL	RZ-C	FPE-R
2005	Framinan et al.	C1-FL	Composta	LR(x)	FL-C	
		C2-FL	Composta	LR(x)	FL-C	FIE-R
2005	Li e Wu	RZ-LW	Simples	SWPT	iRZ	
2007	Nagano e Moccellini	NM-flowtime	Composta	SPT	NEH-C	NM
2009	Li et al.	ICH1	Composta	LR(n/m)	iRZ	
		ICH2	Composta	LR(n/m)	iRZ	FPE
		ICH3	Composta	LR(n/m)	iRZ	FPE-R
2009	Laha e Sarin	FL-LS	Simples	SPT	FL-LS-C	
2013	Pan e Ruiz	LR-NEH(x)	Simples	LR-NEH(x)		
		PR1(x)	Composta	LR-NEH(x)		iRZ
		PR2(x)	Composta	LR-NEH(x)		VNS
		PR1(x)	Composta	LR-NEH(x)		iRZ-NEH
		PR2(x)	Composta	LR-NEH(x)		VNS-NEH
2015	Fernandez-Viagas e Framinan	FF (x)	Simples		FF (x)	
		FF-NEH(x)	Simples	FF-NEH(x)		
		FF-PR1(x)	Composta	FF-NEH(x)		iRZ

Dentro do problema com critério de minimização do *total flowtime*, o presente trabalho delimitou o seu objeto de estudo às heurísticas construtivas simples. Portanto, apenas método construtivos simples serão propostos e comparados. Na próxima seção serão detalhadas as melhores heurísticas simples presentes na literatura.

4.3 HEURÍSTICAS SIMPLES PARA O PROBLEMA $F | pmu / \sum C_i$

Para problema *flow shop* permutacional com minimização do *total flowtime*, Pan e Ruiz (2013) realizaram uma comparação com um número significativo de heurísticas simples e compostas nos problemas testes de Taillard. Para fins de comparação com o método desenvolvido neste trabalho, foram selecionadas as heurísticas simples que obtiveram melhor performance nas análises experimentais e estatísticas realizadas por Pan e Ruiz (2013). Nesta seção será apresentado o detalhamento das heurísticas implementadas para o problema $F | pmu / \sum C_i$. Tabela 13 mostra os resultados obtidos por Pan e Ruiz (2013) por meio das comparações realizadas.

Tabela 13 – Resultados obtidos nas experimentações computacionais de Pan e Ruiz (2013).

Problema	Raj		LIT		SPD1		SPD2		RZ		WY		LR(1)		LR(n/m)	
	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)
20 x 5	5,80	0,00	9,40	0,00	18,85	0,00	19,35	0,00	2,90	0,00	3,03	0,00	2,74	0,00	2,55	0,00
20 x 10	4,61	0,00	11,23	0,00	15,56	0,00	14,37	0,00	1,90	0,00	2,80	0,00	3,77	0,00	3,42	0,00
20 x 20	4,55	0,00	6,95	0,00	10,23	0,00	10,07	0,00	1,97	0,00	2,96	0,00	3,21	0,00	3,21	0,00
50 x 5	4,51	0,00	7,69	0,00	21,01	0,00	21,56	0,00	2,70	0,00	3,70	0,01	2,20	0,00	1,56	0,01
50 x 10	6,28	0,00	9,45	0,01	15,96	0,01	14,24	0,00	3,22	0,01	3,39	0,02	5,26	0,00	2,97	0,01
50 x 20	5,88	0,00	10,19	0,02	11,57	0,01	10,74	0,00	2,94	0,01	3,22	0,04	3,85	0,01	3,40	0,01
100 x 5	4,09	0,00	5,42	0,03	23,05	0,03	23,31	0,00	2,57	0,02	2,32	0,17	1,25	0,01	0,63	0,17
100 x 10	4,73	0,00	8,04	0,06	20,98	0,05	19,55	0,00	3,26	0,05	2,78	0,35	2,89	0,01	2,03	0,14
100 x 20	6,03	0,01	8,19	0,14	12,41	0,11	10,22	0,00	2,72	0,09	3,03	0,68	4,28	0,03	3,36	0,14
200 x 10	4,64	0,03	6,39	0,54	22,25	0,46	21,50	0,10	2,77	0,33	2,56	6,03	2,47	0,10	1,39	1,99
200 x 20	4,98	0,06	9,14	1,01	17,80	0,90	14,95	5,82	2,51	0,70	2,35	11,72	3,81	0,20	2,00	2,00
500 x 20	4,21	0,81	7,03	15,28	18,76	12,61	18,87	11,29	2,34	10,01	1,79	483,79	1,79	3,10	0,94	77,23
Média	5,02	0,08	8,26	1,42	17,37	1,18	16,56	1,43	2,65	0,94	2,83	41,90	3,13	0,29	2,29	6,81

Problema	LR(n)		NEH		FL		RZ-LW		FL-LS		LR-NEH(5)		LR-NEH(10)		LR-NEH(15)	
	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)	DRM	CPU (s)
20 x 5	2,44	0,00	5,05	0,00	2,49	0,00	1,54	0,00	1,74	0,00	2,26	0,00	2,26	0	2,26	0
20 x 10	3,35	0,00	4,14	0,00	2,40	0,00	1,43	0,00	1,25	0,00	2,51	0,00	2,51	0	2,51	0
20 x 20	2,42	0,01	3,88	0,00	2,44	0,00	1,07	0,00	1,04	0,01	2,01	0,00	1,89	0	1,89	0,01
50 x 5	1,56	0,05	4,25	0,00	2,02	0,03	1,63	0,01	1,37	0,04	1,33	0,01	1,33	0,01	1,33	0,02
50 x 10	2,81	0,10	5,08	0,00	2,54	0,05	1,44	0,04	1,61	0,08	2,48	0,01	2,43	0,02	2,43	0,04
50 x 20	3,07	0,23	4,39	0,01	2,01	0,10	1,63	0,05	1,28	0,17	2,67	0,02	2,43	0,05	2,43	0,07
100 x 5	0,63	0,83	3,18	0,01	1,20	0,31	1,69	0,10	1,12	0,52	1,35	0,04	1,25	0,07	1,25	0,11
100 x 10	2,00	1,37	4,54	0,02	2,41	0,73	1,09	0,28	1,12	1,24	1,67	0,08	1,54	0,16	1,54	0,24
100 x 20	2,67	2,76	4,48	0,04	2,04	1,47	0,87	0,57	0,91	2,59	2,44	0,17	2,16	0,34	1,98	0,52
200 x 10	1,39	19,85	3,16	0,14	1,39	10,16	1,40	2,42	1,11	18,39	1,03	0,59	0,96	1,16	0,95	1,74
200 x 20	1,94	39,98	3,88	0,29	1,62	20,91	0,80	6,26	1,32	36,61	1,57	1,27	1,52	2,57	1,48	3,74
500 x 20	0,82	1544,32	2,32	4,00	1,29	701,12	0,88	114,84	0,73	1381,23	0,80	17,66	0,68	35,24	0,65	52,85
Média	2,09	134,12	4,03	0,37	1,99	61,24	1,29	10,38	1,22	120,24	1,84	1,65	1,75	3,3	1,72	4,94

Fonte: adaptado de Pan e Ruiz, 2013.

Como pode ser observado, as heurísticas simples que obtiveram melhores resultados nas análises estatísticas realizadas por Pan e Ruiz (2013) foram os métodos FL-LS e LR-

NEH(x). Como a heurística FL-LS de Laha e Sarin (2009) é derivada do algoritmo FL de Framinan e Leisten. (2003), o método FL também será comparado para avaliar a evolução do algoritmo. O método FF-NEH(x) proposto recentemente por Fernandez-Viagas e Framinan (2015) também será analisado. Em suma, serão comparados as seguintes heurísticas simples: FL, FL-LS, LR-NEH(x) e FF-NEH(x). O pseudocódigo destas heurística serão apresentados nos próximos tópicos.

i. Heurística FL

Framinan e Leisten. (2003) desenvolveram uma heurística simples denominada FL. A heurística realiza um procedimento de permutação de tarefas a cada iteração do NEH. A Figura 37 apresenta o pseudocódigo do método FL.

```

procedimento FL
    Calcule  $P_j \forall n \in N$ ;
    Ordene  $P$  em ordem crescente;
     $\pi := \emptyset$ ;
    para  $passo := 1$  até  $n$  faça
         $j := tarefa(P[passo])$ ;
        Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ ;
        Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor  $\sum C_i$ ;
        para  $passo2 := 1$  até  $n$  faça
            para  $passo3 := passo2 + 1$  até  $n$  faça
                 $\pi^p \leftarrow \pi$ ;
                Troque as tarefas nas posições  $passo2$  e  $passo3$  em  $\pi^p$ ;
                Avalie o valor de  $\sum C_i$  de  $\pi^p$ ;
            fim
        fim
        Escolha a sequência  $\pi^p$  resultante da permutação que possua o menor valor de
         $\sum C_i$ ;
    fim
fim

```

Figura 37 – Heurística FL.

ii. Heurística FL-LS

Laha e Sarin (2009) apresentaram uma heurística chamada FL-LS que realiza um procedimento de reinserção de todas as tarefas na sequência após cada iteração do NEH. A Figura 38 apresenta o pseudocódigo deste método.

```

procedimento FL-LS
    Calcule  $P_j \forall n \in N$ ;
    Ordene  $P$  em ordem crescente;
     $\pi := \emptyset$ ;
    para  $passo := 1$  até  $n$  faça
         $j := tarefa(P[passo])$ ;
        Teste a tarefa  $j$  em todas as posições possíveis de  $\pi$ ;
        Insira a tarefa  $j$  na posição de  $\pi$  que resulte no menor  $\sum C_i$ ;
        para  $passo2 := 1$  até  $n$  faça
             $\pi^a \leftarrow \pi$ 
            Extraia a tarefa  $h$  da posição  $passo2$  de  $\pi^a$ ;
            Teste a tarefa  $h$  em todas as posições possíveis de  $\pi^a$ ;
            Insira a tarefa  $h$  na posição de  $\pi^a$  que resulte no menor  $\sum C_i$ ;
        fim
        Escolha a sequência  $\pi^a$  que apresentou o menor valor de  $\sum C_i$ ;
    fim
fim

```

Figura 38 – Heurística FL-LS.

iii. Heurística LR-NEH(x)

Pan e Ruiz (2013) desenvolveram um método chamado LR-NEH(x) que se baseia nas heurísticas NEH e LR(x) de Liu e Reeves (2001) para construir uma sequência. A Figura 39 ilustra o pseudocódigo desta heurística.

Procedimento LR-NEH(x)

Calcule os valores de ξ_{ik} , AT_{jk} e IT_{jk} por meio das Equações 36, 37 e 38, respectivamente;

Gere uma lista de tarefas $L = \{L_1, L_2, \dots, L_n\}$ em ordem crescente de ξ_{j0} (faça o desempate pelo menor valor de IT_{j0});

$d = 3 \cdot n/4$;

para $l := 1$ **até** x **faça**

$\pi^l := \{L_l\}$, $U := J - \{L_l\}$;

para $k := 2$ **até** d **faça**

Pegue a tarefa j com o menor valor de ξ_{jk} (faça o desempate pelo menor valor de IT_{ik-1}) de U e coloque-a no final da sequência π^l ;

Remova a tarefa j de U ;

fim

Calcule $P_j = \sum_{i=1}^m p_{ij} \forall j \in U$;

Ordene P_j em ordem crescente;

para $k := 1$ **até** $n - d$ **faça**

$j := \text{tarefa}(P[k])$;

Teste a tarefa j em todas as posições de π^l ;

Insira a tarefa j em π^l na posição que resulte no menor valor de $\sum C_i$;

fim

fim

retorne a sequência $\pi \in \{\pi^1, \pi^2, \dots, \pi^x\}$ com o menor valor de $\sum C_i$.

Figura 39 – Heurística LR-NEH(x). Fonte: adaptado de Pan e Ruiz, 2013.

iv. Heurística FF-NEH(x)

Fernandez-Viagas e Framinan (2015) modificaram o procedimento de seleção de tarefas LR(x) desenvolvido por Liu e Reeves (2001) por uma outra forma alternativa do cálculo do índice ξ_{jk} ; este novo procedimento foi denominado de FF(x). Fernandez-Viagas e Framinan (2015) substituíram o procedimento LR(x) da heurística LR-NEH(x) de Pan e Ruiz (2013) pelo método proposto FF(x) apresentando uma heurística denominada de FF-NEH(x). A Figura 40 apresenta o pseudocódigo desta heurística.

Procedimento FF-NEH(x)

Calcule os valores de ξ'_{ik} , AT'_{jk} e IT'_{jk} por meio das Equações 39, 40 e 41, respectivamente;

Gere uma lista de tarefas $L = \{L_1, L_2, \dots, L_n\}$ em ordem crescente de ξ'_{j_0} (faça o desempate pelo menor valor de IT'_{j_0});

$d = 3 \cdot n/4$;

para $l := 1$ **até** x **faça**

$\pi^l := \{L_l\}$, $U := J - \{L_l\}$;

para $k := 2$ **até** d **faça**

Pegue a tarefa j com o menor valor de ξ'_{jk} (faça o desempate pelo menor valor de IT'_{ik-1}) de U e coloque-a no final da sequência π^l ;

Remova a tarefa j de U ;

fim

Calcule $P_j = \sum_{i=1}^m p_{ij} \forall j \in U$;

Ordene P_j em ordem crescente;

para $k := 1$ **até** $n - d$ **faça**

$j := \text{tarefa}(P[k])$;

Teste a tarefa j em todas as posições de π^l ;

Insira a tarefa j em π^l na posição que resulte no menor valor de $\sum C_i$;

fim

fim

retorne a sequência $\pi \in \{\pi^1, \pi^2, \dots, \pi^x\}$ com o menor valor de $\sum C_i$.

Figura 40 – Heurística FF-NEH(x). Fonte: adaptado de Fernandez-Viagas e Framinan (2015).

Na próxima seção serão apresentadas as heurísticas simples propostas para o problema $F | pmu / \sum C_i$.

4.4 NOVOS MÉTODOS HEURÍSTICOS PARA O PROBLEMA $F | pmu / \sum C_j$

A heurística proposta para o problema *flow shop* permutacional com critério de minimização do *total flowtime* foi denominada de FF-RN(x, y). O método foi baseado na heurística LR-NEH(x) de Pan e Ruiz (2013), que gera uma solução parcial com d tarefas usando o LR(x), e então o restante das $n - d$ são inseridas na sequência usando a heurística NEH. A ideia inicial era trocar o método NEH pela heurística FL-LS no método LR-NEH(x), porém experimentalmente esta substituição se mostrou inviável computacionalmente. O tempo computacional consumido pelo método foi aproximadamente 17 vezes maior do que o tempo demandado pela heurística FL-LS, que já é intensiva computacionalmente.

A saída encontrada foi limitar o número de tarefas selecionadas para reinserção no procedimento FL-LS com um parâmetro denominado de y . No entanto, surgiu a questão de como selecionar estas y tarefas para reinserção. Para isso, recorreu-se a ideia proposta por Gao e Zhang (2007) de selecionar as tarefas conforme uma variável que atribui pesos às tarefas. Para isso foi desenvolvido uma função de peso ligeiramente diferente da proposta por Gao e Zhang (2007), que ordena as tarefas em ordem crescente do índice W_j' atribuído a estas, calculado pela Equação 42, onde j é a tarefa retirada de uma sequência π com k tarefas.

$$W_j' = Total\ Flow\ Time(\pi - \{j\}) \quad para\ j = 1, \dots, k \quad 42)$$

Como pode ser notado, a tarefa a ser priorizada é aquela que quando retirada da sequência resulta no menor *total flowtime*; ou seja, atribui-se um peso à tarefa j na sequência π . Quanto maior o peso W_j' , maior a probabilidade da reinserção da tarefa j na sequência π melhorar o *total flowtime* da solução corrente.

A heurística LR(x), utilizada no método LR-NEH(x), também foi substituída pelo método FF(x) de Fernandez-Viagas e Framinan (2015), que obteve um desempenho semelhante ao LR(x), mas com um esforço computacional menor. Portanto, a heurística FF-RN(x, y) gera uma solução parcial com d tarefas utilizando a heurística FF(x), e então o restante das $n - d$ são inseridas utilizando um método baseado no FL-LS, denominado de RN, que faz a reinserção de apenas y tarefas da sequência parcial a cada iteração do NEH. A

Figura 41 apresenta o pseudocódigo da heurística FF-RN(x, y), e a Figura 42 ilustra um exemplo do funcionamento da heurística. Na próxima seção, o método FF-RN(x, y) será comparado com outras heurísticas por meio de experimentos computacionais e análise estatísticas. A heurística será comparada em termos da qualidade de solução e esforço computacional.

Procedimento FF-NEH(x)

Calcule os valores de ξ'_{ik} , AT'_{jk} e IT'_{jk} por meio das Equações 39, 40 e 41, respectivamente;

Gere uma lista de tarefas $L = \{L_1, L_2, \dots, L_n\}$ em ordem crescente de ξ'_{j0} (faça o desempate pelo menor valor de IT'_{j0}); $d = 3 \cdot n/4$;

para $l := 1$ **até** x **faça**

$\pi^l := \{L_l\}$, $U := J - \{L_l\}$;

para $k := 2$ **até** d **faça**

Pegue a tarefa j com o menor valor de ξ'_{jk} (faça o desempate pelo menor valor de IT'_{ik-1}) de U e coloque-a no final da sequência π^l ;

Remova a tarefa j de U ;

fim

Calcule $P_j = \sum_{i=1}^m p_{ij} \forall j \in U$;

Ordene P_j em ordem crescente;

para $k := 1$ **até** $n - d$ **faça**

$j := tarefa(P[k])$;

Teste a tarefa j em todas as posições de π^l ;

Insira a tarefa j em π^l na posição que resulte no menor valor de $\sum C_i$.

para $q := 1$ **até** $n - d + 1$ **faça**

$j := tarefa(q)$;

Retire a tarefa j de π^l ;

$W'_j = TFT(\pi^l)$;

fim

Ordene W'_j em ordem crescente;

para $q := 1$ **até** y **faça**

$j := tarefa(W'[q])$;

Retire a tarefa j de π^l ;

Teste a tarefa j em todas as posições de π^l ;

Insira a tarefa j em π^l na posição que resulte no menor *total flowtime*;

fim

fim

fim

retorne a sequência $\pi \in \{\pi^1, \pi^2, \dots, \pi^x\}$ com o menor valor de $\sum C_i$.

Figura 41 – Heurística FF-RN(x, y) .

Passos	Exemplo
1) Uma lista L de tarefas em ordem crescente de ξ_{j0}^1 é gerada (desempate pelo menor valor de IT_{j0}^1).	$\{J_1, J_2, J_3, J_4, J_5, J_6\}$
2) A primeira posição da sequência é atribuída a primeira tarefa da lista L .	J_1
3) d tarefas são inseridas de acordo com o menor valor de ξ_{jk}^1 (desempate pelo menor valor de IT_{j0}^1).	$J_1 \quad J_2 \quad d = 4$ $J_1 \quad J_2 \quad J_4$ $J_1 \quad J_2 \quad J_4 \quad J_3$
3) As $n - d$ tarefas restantes são ordenadas pela regra SPT.	$\{J_5, J_6\}$
4) A primeira tarefa da ordenação SPT é testada em todas as posições. (NEH)	$J_5 \quad J_1 \quad J_2 \quad J_4 \quad J_3$ $J_1 \quad J_5 \quad J_2 \quad J_4 \quad J_3$... $J_1 \quad J_2 \quad J_4 \quad J_3 \quad J_5$ Melhor Solução $J_1 \quad J_2 \quad J_5 \quad J_4 \quad J_3$
5) É gerada uma lista W' das tarefas da sequência em ordem crescente de W_j^j , e y tarefas são testadas em todas as posições (neste exemplo $y = 2$)	$\{J_2, J_3, J_1, J_5, J_4\}$
6) A primeira tarefa da lista W' é testada em todas as posições	$J_2 \quad J_1 \quad J_5 \quad J_4 \quad J_3$ $J_1 \quad J_2 \quad J_5 \quad J_4 \quad J_3$... $J_1 \quad J_5 \quad J_4 \quad J_3 \quad J_2$ Melhor Solução $J_1 \quad J_5 \quad J_4 \quad J_2 \quad J_3$
7) A segunda tarefa da lista W' é testada em todas as posições	$\{J_2, J_3, J_1, J_5, J_4\}$ $J_3 \quad J_1 \quad J_5 \quad J_4 \quad J_2$ $J_1 \quad J_3 \quad J_5 \quad J_4 \quad J_2$... $J_1 \quad J_5 \quad J_4 \quad J_2 \quad J_3$ Melhor Solução $J_1 \quad J_3 \quad J_5 \quad J_4 \quad J_2$
8) A segunda tarefa da ordenação SPT é testada em todas as posições (NEH).	$J_6 \quad J_1 \quad J_3 \quad J_5 \quad J_4 \quad J_2$ $J_1 \quad J_6 \quad J_3 \quad J_5 \quad J_4 \quad J_2$... Melhor Solução $J_1 \quad J_3 \quad J_6 \quad J_5 \quad J_4 \quad J_2$
9) É gerada uma lista das tarefas da sequência em ordem crescente de W_j^j , e y tarefas são testadas em todas as posições (neste exemplo $y = 2$)	$\{J_2, J_6, J_1, J_3, J_4, J_5\}$
10) A primeira tarefa da lista W' é testada em todas as posições, e a posição que resultou no menor TFT é assumida como solução corrente.	$\{J_2, J_6, J_1, J_3, J_4, J_5\}$...
11) A segunda tarefa da lista W' é testada em todas as posições, e a posição que resultou no menor TFT é assumida como solução corrente.	$\{J_2, J_6, J_1, J_3, J_4, J_5\}$...
12) A sequência fornecida pelo passo 11) que resultou no menor TFT é assumida como melhor solução.	$J_1 \quad J_2 \quad J_3 \quad J_6 \quad J_5 \quad J_4$
13) Volta ao passo 3). O procedimento é novamente executado com a segunda tarefa da lista L na primeira posição.	J_2 $J_2 \quad J_1$...

Figura 42 – Exemplo ilustrativo do funcionamento da heurística FF-RN(x, y).

4.5 EXPERIMENTAÇÃO COMPUTACIONAL E ANÁLISE DOS RESULTADOS PARA O PROBLEMA $F | pmu | \sum C_i$

Nesta seção foi conduzida uma avaliação estatística e computacional das melhores heurísticas existentes bem como dos métodos propostos para o problema *flow shop* permutacional com critério de minimização do *total flowtime*. Foram testadas um total de 12 heurísticas simples nos problemas testes do Taillard e aquelas que tiveram melhor performance foram selecionadas para serem avaliadas nos problemas testes do VFR de Vallada, Ruiz e Framinan (2014).

As heurísticas simples que foram superadas na revisão e análise de Pan e Ruiz (2013) não foram comparadas neste trabalho. Foram avaliadas as heurísticas FL, FL-LS, LR-NEH(x) e FF-NEH(x) bem como as heurísticas propostas. Também não foram empregados métodos de aceleração computacional nas heurísticas implementadas. Todos os métodos foram programados em C++ e rodaram em um processador i7 4770 3.9 GHz com 8 GB de memória RAM e um SSD de 120 GB.

Segundo Pan e Ruiz (2013), recentemente, um número crescente de pesquisadores tem utilizado os problemas testes do Taillard para avaliar seus métodos lidando com o critério de minimização do *total flowtime*. (LIU; REEVES, 2001; LI; LIU; WU, 2006; LI; WANG, 2006; TASGETIREN *et al.*, 2007; JARBOUI; EDDALY; SIARRY, 2009, DONG; HUANG; CHEN, 2009; LI; WANG; WU, 2009) e Zhang, Li e Wang (2009). Portanto, as heurística simples serão avaliadas neste conjunto de problemas testes. Também será utilizado o conjunto de problemas recentemente proposto por Vallada, Ruiz e Framinan (2014) chamado de VFR.

As heurísticas LR-NEH(x) e FF-NEH(x) foram testadas com os valores de x iguais a 5, 10 e 15. A heurística FF-RN(x, y) foi testada com valores de x igual a 15 e y igual a 10, 15 e 20. Como foi explicado, a heurística FF-RN(x, y) gera uma lista de priorização de tarefas antes de realizar o procedimento de reinserção. Para avaliar o impacto desta ordenação no desempenho da heurística FF-RN(x, y), também será avaliado o método proposto sem a priorização de tarefas, selecionando y tarefas da sequência de frente para trás. Esta versão modificada foi denotada de FF-RN(x, y)*.

Os resultados obtidos pelas heurísticas simples avaliadas foram agrupados por subconjuntos de problemas e são apresentados nas Tabelas 14 e 15. Nesta seção, todos os tempos de computação são dados em segundos.

Tabela 14 – DRM para cada classe de problemas e média geral obtidas pelos novos métodos heurísticos nos problemas testes do Taillard.

Problema	FL	FL-LS	LR-NEH(10)	LR-NEH(15)	FF-NEH(10)	FF-NEH(15)
20 x 5	2,40	1,67	1,93	1,88	1,70	1,70
20 x 10	2,19	1,00	2,31	2,31	2,26	2,26
20 x 20	2,27	1,20	1,85	1,85	1,85	1,85
50 x 5	1,73	0,78	0,75	0,75	0,81	0,81
50 x 10	1,98	1,36	2,04	2,04	1,81	1,81
50 x 20	2,05	0,99	2,02	2,02	1,86	1,86
100 x 5	0,88	0,87	0,36	0,36	0,42	0,42
100 x 10	1,74	0,84	0,68	0,68	0,75	0,75
100 x 20	1,59	0,84	1,57	1,29	1,36	1,22
200 x 10	1,18	1,03	0,58	0,55	0,42	0,42
200 x 20	1,47	0,73	1,06	1,06	0,84	0,75
500 x 20	0,99	0,55	0,43	0,33	0,37	0,34
Média	1,71	0,99	1,30	1,26	1,21	1,18

Tabela 15 - Continuação da Tabela 14.

Problema	FF-RN(15;10)*	FF-RN(15; 10)	FF-RN(15; 15)*	FF-RN(15; 15)	FF-RN(15; 20)*	FF-RN(15; 20)
20 x 5	0,77	0,22	0,32	0,09	0,18	0,04
20 x 10	0,58	0,40	0,29	0,10	0,14	0,10
20 x 20	0,68	0,23	0,19	0,17	0,12	0,11
50 x 5	0,75	0,32	0,62	0,23	0,49	0,23
50 x 10	1,27	0,41	1,01	0,23	0,86	0,18
50 x 20	1,35	0,59	1,18	0,27	0,95	0,11
100 x 5	0,32	0,07	0,29	0,05	0,28	0,03
100 x 10	0,75	0,28	0,60	0,18	0,59	0,08
100 x 20	1,11	0,33	1,02	0,25	0,70	0,00
200 x 10	0,46	0,18	0,45	0,08	0,44	0,10
200 x 20	0,76	0,20	0,71	0,20	0,70	0,15
500 x 20	0,38	0,09	0,40	0,07	0,38	0,03
Média	0,76	0,28	0,59	0,16	0,49	0,10

Desconsiderando os métodos propostos, o método FL-LS é o que apresenta a melhor performance com um DRM de 0,99%, superando as heurísticas LR-NEH(x) e FF-NEH(x) com 1,26% e 1,18%, respectivamente.

Em relação aos métodos propostos, considerando a totalidade dos problemas testes, a heurística FF-RN(15, 20) supera o método FL-LS em qualidade de solução com 0,10% de

DRM, apresentando melhores resultados para quase a totalidade dos subconjuntos de problemas. Pode ser observado que, geralmente, quanto maior o valor de y melhores são os resultados. Tal resultado já era esperado, visto que y dimensiona o tamanho da busca na vizinhança por inserção realizada pelo algoritmo. As Figuras 43 e 44 apresentam o DRM de cada heurística por grupos de problemas agregados por número de tarefas e máquinas, respectivamente.

Figura 43 – DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes do Taillard.

Como pode ser observado, quanto maior o número de tarefas, melhores são os resultados obtidos pelas heurísticas FL, FL-LS, LR-NEH(x) e FF-NEH(x). A heurística FF-RN(x , y) possui um comportamento mais estável nos agrupamentos de problemas, diminuindo a superioridade em relação às outras heurísticas conforme o número de tarefas aumenta. É interessante notar que os métodos LR-NEH(x) e FF-NEH(x) apresentam melhor performance que a heurística FL-LS para problemas com mais de 100 tarefas, mesmo sendo esta última mais intensiva computacionalmente.

Figura 44 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes do Taillard.

Para agrupamentos de problemas por número de máquinas os métodos apresentam um comportamento variado. As heurísticas LR-NEH(x) e FF-NEH(x) pioram consideravelmente os resultados para problemas com 10 e 20 máquinas, sendo superadas pelo método FL-LS. O método FF-RN(x, y) supera as heurísticas comparadas em todos os conjuntos de problemas e apresenta um comportamento estável. A Figura 45 apresenta o gráfico com os intervalos das médias para as 12 heurísticas simples avaliadas, com um intervalo de confiança de 95%.

Figura 45 – Gráfico com o intervalo das médias das heurísticas comparadas nos problemas testes do Taillard com um nível de confiança de 95%.

Em termos estatísticos, a heurística FF-RN(x ; y) apresenta uma diferença significativa em relação aos outros métodos para todas as combinações dos parâmetros x e y . O método FF-RN(x , y)*, que não gera a lista de tarefas antes de reinseri-las, apresentou um desempenho inferior aos métodos que possuem o mecanismo de priorização de tarefas. As Tabelas 16 e 17 apresentam o tempo computacional médio de cada heurística. A Figura 46 apresenta um gráfico com o esforço computacional de cada método comparado.

Tabela 16 - Tempo computacional das heurísticas nos problemas testes do Taillard.

Problema	FL	FL-LS	LR-NEH(10)	LR-NEH(15)	FF-NEH(10)	FF-NEH(15)
20 x 5	0,00	0,00	0,00	0,00	0,00	0,00
20 x 10	0,00	0,00	0,00	0,00	0,00	0,00
20 x 20	0,00	0,00	0,00	0,01	0,00	0,01
50 x 5	0,01	0,02	0,01	0,01	0,01	0,01
50 x 10	0,02	0,04	0,02	0,03	0,01	0,02
50 x 20	0,04	0,08	0,04	0,06	0,03	0,04
100 x 5	0,15	0,31	0,06	0,08	0,04	0,05
100 x 10	0,31	0,63	0,11	0,17	0,08	0,12
100 x 20	0,63	1,27	0,26	0,35	0,19	0,26
200 x 10	4,98	10,15	0,61	0,94	0,45	0,62
200 x 20	10,29	20,76	1,37	2,07	0,96	1,39
500 x 20	402,20	855,76	19,42	28,56	12,56	18,74
Média	34,88	74,08	1,83	2,69	1,19	1,77
TRPM	-0,48	0,08	-0,87	-0,82	-0,91	-0,87

Tabela 17 - Continuação da Tabela 16.

Problema	FF-RN(15;10)*	FF-RN(15; 10)	FF-RN(15; 15)*	FF-RN(15; 15)	FF-RN(15; 20)*	FF-RN(15; 20)
20 x 5	0,01	0,01	0,01	0,01	0,01	0,01
20 x 10	0,02	0,02	0,02	0,02	0,03	0,03
20 x 20	0,03	0,03	0,04	0,05	0,05	0,05
50 x 5	0,08	0,09	0,11	0,12	0,15	0,17
50 x 10	0,18	0,20	0,24	0,26	0,30	0,34
50 x 20	0,37	0,41	0,52	0,51	0,63	0,64
100 x 5	0,48	0,50	0,60	0,65	0,74	0,81
100 x 10	0,88	1,00	1,24	1,32	1,56	1,80
100 x 20	1,96	2,28	2,88	3,14	3,68	4,18
200 x 10	5,75	6,77	8,29	9,50	10,80	12,63
200 x 20	13,32	15,89	19,32	22,54	25,30	29,16
500 x 20	199,03	230,11	280,76	333,23	369,92	421,12
Média	18,51	21,44	26,17	30,95	34,43	39,25
TRPM	0,12	0,27	0,57	0,71	0,99	1,23

Figura 46 - Tempo computacional média e TRPM de cada método nos problemas testes do Taillard.

Com as informações obtidas dos resultados acima, pode ser concluído que as heurísticas LR-NEH(x) e FF-NEH(x) possuem um menor tempo computacional médio para todos os subconjuntos de problemas. O método FF-RN(15, 20) apresenta um tempo médio 47% menor do que o FL-LS. No entanto, esta proporção de tempo não se mantém para todos

os problemas. Em relação ao FL-LS, para problemas com $n \leq 100$ o método proposto consome um tempo maior, para problemas com $n = 200$ o método apresenta uma eficiência próxima e para $n = 500$ tarefas um tempo significativamente menor. Pelo fato de os problemas de 500 tarefas representarem 89% do tempo médio, o método FF-RN(15, 20) acaba por ter um tempo médio menor do que o FL-LS, mesmo consumindo mais tempo para outros subconjuntos de problemas.

Por causa destas distorções foi adotado o indicador TRPM como critério de avaliação da eficiência nos problemas testes do Taillard. Neste sentido, o método FF-RN(15, 20) apresenta a menor eficiência computacional com um índice de 1,23 contra 0,08 do FL-LS. Os métodos propostos que mais se aproximam ao FL-LS em eficiência são os métodos FF-RN(15, 10)* e FF-RN(15, 10), que possuem TRPM de 0,12 e 0,27 e DRM iguais a 0,76% 0,28%, respectivamente.

O método FF-RN(x, y)*, que não prioriza as tarefas, consome em média 14% menos tempo computacional e apresenta resultados significativamente inferiores às heurísticas que geram a lista W'_j antes da reinserção, como pode ser visto na Tabela 18. Sendo assim, entende-se que o ganho fornecido pela priorização das tarefas justifica o maior tempo computacional utilizado.

Tabela 18 – Impacto da geração da lista W' no desempenho da heurística FF-RN(x, y).

Indicadores	FF-RN(15;10)*	FF-RN(15;10)	FF-RN(15;15)*	FF-RN(15;15)	FF-RN(15;20)*	FF-RN(15;20)
DRM	0,78	0,28	0,59	0,15	0,49	0,10
Tempo (s)	18,51	21,44	26,17	30,95	34,43	39,25
TRPM	0,12	0,27	0,51	0,71	0,99	1,23

*sem priorização de tarefas

Com as análises realizadas, pode ser afirmado que: (1) a heurística FF-RN(x, y) supera todas as heurísticas em qualidade de solução; (2) os métodos FF-NEH(x) e LR-NEH(x) apresentam o melhor trade-off entre qualidade de solução e tempo computacional; (3) o mecanismo de priorização da tarefas melhora substancialmente os resultados do método FF-RN(x, y).

Por meio das análises nos problemas testes do Taillard foram selecionados os melhores métodos para serem avaliados nos problemas testes do VFR de Vallada, Ruiz e Framinan (2014), que é mais extenso. Este *benchmark* possui 480 problemas testes divididos

igualmente em dois grupos de problemas (grande porte e pequeno porte) e possui problemas com até 800 tarefas e 60 máquinas. A Tabela 19 apresenta os valores de DRM para os problemas testes de pequeno porte do VFR.

Tabela 19 – DRM para cada classe de problemas e média geral obtidas pelos novos métodos heurísticos nos problemas testes de pequeno porte do VFR.

Problema	LR-NEH(15)	FF-NEH(15)	FL-LS	FF-RN(15; 20)
10 x 5	0,68	0,68	0,53	0,10
10 x 10	1,29	1,29	0,34	0,00
10 x 15	1,48	1,48	0,28	0,06
10 x 20	1,20	1,20	0,11	0,00
20 x 5	1,50	1,55	1,14	0,00
20 x 10	2,72	2,41	1,44	0,00
20 x 15	2,63	2,68	1,04	0,01
20 x 20	2,66	2,64	0,77	0,08
30 x 5	1,04	1,02	0,92	0,00
30 x 10	2,18	2,08	1,45	0,00
30 x 15	2,67	2,44	0,81	0,00
30 x 20	2,64	2,75	1,18	0,04
40 x 5	0,83	1,03	0,79	0,04
40 x 10	1,81	1,63	1,50	0,00
40 x 15	2,09	1,92	1,24	0,00
40 x 20	2,78	2,66	0,98	0,13
50 x 5	0,77	0,66	0,54	0,09
50 x 10	1,93	1,77	0,86	0,19
50 x 15	1,42	1,87	0,84	0,08
50 x 20	2,06	1,91	0,65	0,03
60 x 5	0,69	0,59	0,77	0,09
60 x 10	1,04	1,34	1,19	0,00
60 x 15	1,51	1,31	0,81	0,02
60 x 20	1,88	1,79	0,95	0,02
Média	1,73	1,70	0,88	0,04

Constata-se que a heurística FF-RN(15; 20) supera as outras heurísticas em qualidade de solução em todos os subgrupos de problemas de pequeno porte do VFR. As Figuras 47 e 48 apresentam o DRM das heurísticas por conjuntos de problemas agrupados por número de tarefas e máquinas, respectivamente.

Figura 47 – DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes de pequeno porte do VFR.

Na Figura 47, observa-se que as heurísticas FF-NEH(x) e LR-NEH(x) obtêm melhores soluções conforme o número de tarefas aumenta, com exceção de problemas com 10 tarefas. O método FF-RN(15; 20) apresenta um comportamento estável, superando as outras heurísticas para todos os grupos de problemas.

Figura 48 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes de pequeno porte do VFR.

Por meio da Figura 48, é possível notar que quanto maior o número de máquinas, piores são os resultados para as heurísticas FF-NEH(x) e LR-NEH(x). O método FF-RN(15;20) obtém melhores resultados em todos os agrupamentos de problemas. A Figura 49 apresenta um gráfico dos intervalos das médias das heurísticas comparadas, com um grau de confiança de 95%.

Figura 49 – Gráfico com os intervalos das médias das heurísticas comparadas nos problemas testes de pequeno porte do VFR.

Como pode ser observado, a heurística FF-RN(15; 20) é estatisticamente diferente dos outros métodos. A Tabela 20 apresenta a eficiência computacional dos métodos comparados. A Figura 50 dispõe um gráfico com o tempo computacional médio e o TRPM dos métodos comparados.

Tabela 20 – Tempo médio computacional e TRPM das heurísticas nos problemas testes de pequeno porte do VFR.

Problema	LR-NEH(15)	FF-NEH(15)	FL-LS	FF-RN(15; 20)
10 x 5	0,00	0,00	0,00	0,00
10 x 10	0,00	0,00	0,00	0,00
10 x 15	0,00	0,00	0,00	0,00
10 x 20	0,00	0,00	0,00	0,00
20 x 5	0,00	0,00	0,00	0,01
20 x 10	0,00	0,00	0,00	0,03
20 x 15	0,00	0,00	0,00	0,04
20 x 20	0,00	0,00	0,00	0,06
30 x 5	0,00	0,00	0,00	0,04
30 x 10	0,01	0,01	0,01	0,09
30 x 15	0,01	0,01	0,01	0,15
30 x 20	0,02	0,01	0,01	0,20
40 x 5	0,01	0,00	0,01	0,09
40 x 10	0,01	0,01	0,02	0,20
40 x 15	0,02	0,02	0,02	0,31
40 x 20	0,04	0,03	0,03	0,39
50 x 5	0,01	0,01	0,02	0,17
50 x 10	0,03	0,02	0,04	0,33
50 x 15	0,04	0,03	0,06	0,50
50 x 20	0,06	0,04	0,07	0,65
60 x 5	0,02	0,01	0,04	0,25
60 x 10	0,05	0,03	0,08	0,49
60 x 15	0,07	0,05	0,12	0,74
60 x 20	0,10	0,08	0,16	1,05
Médias	0,02	0,02	0,03	0,24
TRPM	-0,72	-0,80	-0,73	2,24

Figura 50 - Tempo médio computacional ou TRPM dos métodos comparados nos problemas testes de pequeno porte do VFR

Para problemas de pequeno porte do VFR, pode ser constatado que a heurística FF-RN(15; 20) fornece melhores soluções à custas de um esforço computacional maior, apresentando o maior valor de TRPM e também o maior valor de tempo computacional médio dentre todas as heurísticas. Os resultados obtidos pelos métodos para os problemas de grande porte do VFR estão dispostos na Tabela 21.

Tabela 21 – DRM para cada classe de problemas e média geral obtidas pelos novos métodos heurísticos nos problemas testes de grande porte do VFR.

Problema	LR-NEH(15)	FF-NEH(15)	FL-LS	FF-RN(15; 20)
100 x 20	1,28	1,07	0,38	0,10
100 x 40	1,34	1,30	0,33	0,06
100 x 60	0,83	0,64	0,69	0,01
200 x 20	1,23	0,90	0,37	0,11
200 x 40	1,04	0,92	0,14	0,08
200 x 60	0,38	0,50	0,50	0,03
300 x 20	0,98	0,73	0,53	0,04
300 x 40	1,28	1,12	0,08	0,27
300 x 60	0,46	0,35	0,68	0,01
400 x 20	0,70	0,50	0,13	0,06
400 x 40	1,07	0,88	0,10	0,20
400 x 60	0,27	0,33	0,43	0,03
500 x 20	0,63	0,60	0,28	0,09
500 x 40	0,99	0,84	0,07	0,08
500 x 60	0,32	0,30	0,66	0,01
600 x 20	0,39	0,42	0,43	0,01
600 x 40	0,77	0,58	0,12	0,05
600 x 60	0,26	0,22	0,59	0,00
700 x 20	0,52	0,36	0,41	0,02
700 x 40	0,75	0,71	0,15	0,14
700 x 60	0,13	0,18	0,60	0,03
800 x 20	0,46	0,34	0,34	0,00
800 x 40	0,61	0,53	0,11	0,09
800 x 60	1,24	1,18	0,62	0,05
Média	0,75	0,65	0,36	0,06

Nos problemas testes de grande porte do VFR, a heurística FF-RN(15; 20), novamente, possui a menor média geral de 0,06%, contra 0,36% do método FL-LS. A heurística proposta também supera os outros métodos em 21 dos 24 subgrupos de problemas.

As Figuras 51 e 52 apresentam o DRM das heurísticas por conjuntos de problemas agrupados por número de tarefas e máquinas, respectivamente.

Figura 51 – DRM para cada subconjunto de problemas agrupados por número de tarefas nos problemas testes de grande porte do VFR

Nota-se que as heurísticas FF-NEH(x) e LR-NEH(x) tendem a melhorar os resultados conforme o número de tarefas aumenta, diminuindo a diferença de performance existente em relação aos métodos FL-LS e FF-RN(15; 20). O método proposto superou as heurísticas em todos os agrupamentos de problemas.

Figura 52 – DRM para cada subconjunto de problemas agrupados por número de máquinas nos problemas testes de grande porte do

VFR

Como pode ser observado, para problemas com 40 máquinas, os métodos FF-RN(15; 20), LR-NEH(x) e FF-NEH(x) apresentam uma piora nos resultados e nos problemas com 20 e 60 máquinas melhoram a performance. O método FL-LS possui um comportamento contrário, obtendo melhores resultados para problemas com 40 máquinas e piores para 20 e 60 máquinas. Contudo, o método FF-RN(15; 20) apresenta menor DRM para todos os agrupamentos de problemas. Os intervalos das médias obtidos pelos métodos comparados foram plotados em um gráfico, que está disposto na Figura 53.

Figura 53 – Gráfico com os intervalos das médias das heurísticas com grau de confiança de 95% nos problemas testes de grande porte do VFR.

Como pode ser visto na Figura 53, o método FF-RN(15; 20) também apresenta diferença estatística para problemas de grande porte do VFR. Na Tabela 22 são apresentadas as análises do tempo computacional dos métodos comparados para este conjunto de problemas. A Figura 54 apresenta o tempo médio computacional e o TRPM dos métodos comparados.

Tabela 22 - Eficiência das heurísticas para problemas de grande porte do VFR

Problema	LR-NEH(15)	FF-NEH(15)	FL-LS	FF-RN(15; 20)
100 x 20	0,36	0,27	1,26	3,95
100 x 40	0,72	0,57	2,56	10,00
100 x 60	1,09	0,91	3,87	17,28
200 x 20	1,94	1,45	20,45	27,06
200 x 40	4,87	3,93	41,31	78,80
200 x 60	8,36	7,11	61,47	144,27
300 x 20	6,10	4,36	102,80	88,15
300 x 40	15,92	13,04	205,29	260,98
300 x 60	27,89	23,33	310,38	480,59
400 x 20	14,20	10,30	323,29	208,76
400 x 40	37,47	30,40	647,51	618,22
400 x 60	65,17	53,92	985,61	1107,13
500 x 20	27,15	19,30	788,52	408,90
500 x 40	72,62	54,83	1580,11	1238,23
500 x 60	124,30	98,75	2413,83	2139,64
600 x 20	46,50	33,79	1633,54	677,03
600 x 40	120,29	96,45	3277,64	2052,88
600 x 60	207,16	158,88	5014,79	3563,49
700 x 20	73,89	49,20	3024,54	1106,46
700 x 40	190,86	138,90	6073,28	3201,82
700 x 60	324,19	244,27	9280,43	5454,73
800 x 20	111,26	72,59	5155,25	1667,91
800 x 40	277,77	202,96	10460,76	4604,74
800 x 60	469,65	351,87	15810,76	8040,92
Médias	92,90	69,64	2800,80	1550,08
TRPM	-0,88	-0,91	0,98	0,81

A heurística FF-RN(15; 20) mostra-se mais eficiente do que o método FL-LS, consumindo um tempo computacional médio de 1550s contra 2801s do FL-LS e um TRPM de 0,81, contra 0,98 do FL-LS. Portanto, para problemas de grande porte, além do método FF-RN(15; 20) superar os outros algoritmos em qualidade de solução, este é mais eficiente do que a heurística FL-LS.

Figura 54 - Tempo computacional médio e TRPM das heurísticas para os problemas de grande porte do VFR

Como pode ser visto, a heurística FL-LS apresenta tempo computacional médio e TRPM maiores do que as outras heurísticas. Os métodos LR-NEH(15) e FF-NEH(15) demandam um tempo consideravelmente menor em comparação com os outros métodos.

Com base nas análises e comparações realizadas, pode ser afirmado que todas as heurísticas comparadas são superadas pelo método FF-RN(x; y), em relação à qualidade de solução. Para problemas de grande porte do VFR, o algoritmo FL-LS, melhor método até então (PAN; RUIZ, 2013), foi superado tanto em termos de qualidade de solução, quanto em relação à eficiência computacional. Também pode ser afirmado que os métodos LR-NEH(x) e FF-NEH(x), apesar de fornecerem soluções piores, são muito eficientes, visto que possuem um custo computacional substancialmente menor em relação aos outros métodos comparados, fornecendo um bom *trade-off* entre custo computacional e qualidade da solução.

4.6 CONSIDERAÇÕES FINAIS DO PROBLEMA $F | pmu / \sum C_i$

Neste capítulo foi realizada uma revisão das heurísticas simples e compostas para o problema de programação *flow shop* com critério de minimização do *total flowtime*. Com os conhecimentos obtidos, um novo método heurístico simples foi proposto. A partir da extensa comparação realizada por Pan e Ruiz (2013), foram selecionadas as melhores heurísticas reportadas na literatura, FL-LS, LR-NEH(x) e FF-NEH(x), para comparação com o método proposto. Os problemas testes do Taillard e do VFR foram utilizados. Todas as heurísticas foram programadas na mesma linguagem e foram testadas na mesma plataforma computacional, o que torna os resultados inteiramente comparáveis. Além disso, foram utilizados testes de significância estatísticas por meio da interpretação de gráficos de intervalos de média com um nível de confiança de 95%.

Pan e Ruiz (2013) afirmaram que o método FL-LS foi o que mostrou a melhor performance dentre as heurísticas simples comparadas. A partir das análises realizadas neste trabalho, o método proposto FF-RN($x; y$) superou a heurística FL-LS. Em relação ao custo computacional, o método apresentou uma eficiência similar ao método FL-LS para problemas de grande porte do VFR. No entanto, o algoritmo proposto é custoso computacionalmente para problemas de pequeno porte. As heurísticas LR-NEH(15) e FF-NEH(15) forneceram um bom trade-off entre custo computacional e qualidade de solução. Concluindo, o método proposto pode ser considerado uma contribuição para o estado da arte em métodos heurísticos simples para o problema *flow shop* permutacional com critério de minimização do *total flowtime*.

5 CONSIDERAÇÕES FINAIS

Neste trabalho, foram definidos os seguintes objetivos: (i) estudar os problemas de programação da produção em ambiente *flow shop* permutacional com critério de minimização do *makespan* e do *total flowtime*; (ii) desenvolver heurísticas que forneçam soluções de alta qualidade e que sejam eficientes computacionalmente.

Sendo assim, foram abordados dois problemas de programação da produção em ambiente *flow shop* permutacional, o primeiro com critério de minimização do *makespan*, e o segundo com objetivo de minimizar o *total flowtime*. Para cada um dos problemas foram realizadas revisões da literatura e implementação dos métodos para fins de comparação. Com o conhecimento oriundos da revisão, foram propostos novos métodos para cada um dos problemas abordados.

Para o problema com critério de minimização do *makespan*, foi demonstrado que a heurística FRB3 fornece a melhor performance dentre os principais métodos revisados na literatura. Para superar esta heurística foi proposto o método G8. Este método construtivo foi posteriormente combinado com diferentes procedimentos de ordenação inicial e mecanismos de desempate existentes na literatura. Foi demonstrado que os métodos G8, F13 e F34 superam a heurística FRB3 com uma eficiência computacional razoável, mas com soluções de alta qualidade.

No problema de minimização do *total flowtime*, foi proposto o método simples FF-RN(x, y), que superou o método FL-LS, que, segundo Pan e Ruiz (2013), é a heurística simples com melhor performance na literatura. Para os problemas testes de pequeno porte do VFR, o método apresentou uma eficiência computacional menor do que o FL-LS. No entanto, para problemas de grande porte, além de fornecer melhor qualidade de solução, o método FF-RN(x, y) apresentou uma eficiência melhor do que o FL-LS.

Como conclusão, para ambos os problemas em consideração, as heurísticas propostas superaram os principais métodos heurísticos construtivos. Com isso, os objetivos definidos (i) e (ii) foram satisfeitos.

Pela delimitação do escopo deste trabalho, não foram abordadas heurísticas compostas. Sendo assim, os métodos compostos PR1-PR4(x) de Pan e Ruiz (2013) e a modificação FF-PR1(x) proposta por Fernandez-Viagas e Framinan (2015) continuam sendo os

melhores métodos compostos para o problema com critério de minimização do *total flowtime*. Para trabalhos futuros, uma proposta seria criar um método compostos por meio da utilização de procedimentos de melhorias na terceira fase do método FF-RN(x, y) e com isso superar as heurísticas compostas FF-PR1-PR4(x).

Para o problema com critério de minimização do *makespan*, a aceleração de Taillard permanece como uma ferramenta fundamental na implementação das heurísticas, visto que diminui substancialmente o tempo computacional demandado. Porém, só é possível usar este procedimento na inserção de tarefas em uma sequência, não sendo possível utiliza-la na avaliação do *makespan* em uma permutação de tarefas, por exemplo. Sendo assim, uma sugestão de futura pesquisa seria propor um mecanismos de aceleração para o cálculo do *makespan* para a permutação de tarefas. Isto proporcionaria oportunidades para a criação de formas de construção e melhoria que divergissem do esquema NEH de inserção, que predomina nas heurísticas construtivas atualmente. Outra sugestão seria propor metaheurísticas para o problema que fornecessem soluções de alta qualidade.

REFERÊNCIAS

AHMADI, R. H.; BAGCHI, U. Improved lower bounds for minimizing the sum of completion times of n jobs over m machines in a *flow shop*. **European Journal of Operational Research**, v. 44, n. 3, p. 331-6, 1990.

ALLAHVERDI, A.; ALDOWAISAN, T. New heuristics to minimize total completion time in m -machine flowshops. **International Journal of Production Economics**, v. 77, n. 1, p. 71-83, 2002.

BAKER, K. R. **Introduction to Sequencing and Scheduling**. New York. John Wiley, 1974.

CAMPBELL, H. G.; DUDEK, R. A.; SMITH, M. L. Heuristic algorithm for n job, m machine sequencing problem. **Management Science Series B-Application**, v. 16, n. 1, p. 630-637, 1970.

DANNENBRING, D. An evaluation of *flow shop* sequencing heuristics. **Management Science**, v. 23, p. 1174-1182, 1977.

DEROUSSI, L.; GOURGAND, M.; NORRE, S. New effective neighborhoods for the permutation *flow shop* problem. Disponível em: <http://hal.archives-ouvertes.fr/view_by_stamp.php?&halsid=sd01ountu02dq0r3f7thqbk61&label=CNRS&language=en&action_todo=view&id=hal-00678053&version=1&view=extended_view>. Acesso: 29/05/2014, 2006.

DONG, X.; HUANG, H.; CHEN, P. An improved NEH-based heuristic for the permutation *flow shop* problem. **Computers & Operations Research**, v. 35, n. 12, p. 3962-3968, 2008.

FERNADEZ-VIAGAS, V. F.; FRAMINAN, J. M. On insertion tie-breaking rules in heuristics for the permutation *flow shop* scheduling problem. **Computers & Operations Research**. 2013.

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 the Operational Research Society**, v. 55, n. 12, p. 1243-55, 2004.

FRAMINAN, J. M.; LEISTEN, R. An efficient constructive heuristic for *total flowtime* minimization in permutation *flow shops*. OMEGA, **International Journal of Management Science**, v. 31, n. 4, p. 311-7, 2003.

FRAMINAN, J. M.; LEISTEN R; RUIZ-USANO, R. Comparison of heuristics for *total flowtime* minimization in permutation *flow shops*. **Computers & Operations Research**, v. 32, n. 5, p. 1237-54, 2005.

FRAMINAN, J. M.; LEISTEN, R; RUIZ-USANO, R. Efficient heuristics for *flow shop* sequencing with the objectives of *makespan* and *total flowtime* minimization. **European Journal of Operational Research**., v.141, n. 3, p. 559-69, 2002.

FRAMINAN, J. M; LEISTEIN, R; RAJENDRAN, C. Different initial sequences for the heuristic of Nawaz, Ensore and Ham to minimize *makespan*, *idletime* or *total flowtime* in the static permutation *flow shop* sequencing problem. **International Journal of Production Research**, v. 31, n. 1, p. 121-148, 2003.

GAREY M. R.; JOHNSON D. S.; SETHI R. The complexity of flowshop and jobshop scheduling. *Math Opns Res*, v. 1, p. 117-129.

GAO, S. W.; ZHANG, W. D. The modification on NEH heuristic in solving permutation *flow shop* scheduling problems. 2007.

GONAZALES, T.; SAHNI, S. *Flow shop* and *job shop* schedules: complexity and approximation. **Operations Research**, v. 26, n. 1, p.36-52, 1978.

GRAHAM, R. L.; LAWLER, E. L.; LENSTRA, J. K.; RINNOOY KAN, A. H. G. Optimization and approximation in deterministic sequencing and scheduling: a survey. **Ann. Discrete Math**, v. 4, p. 287-326, 1979.

GROMICHOA, A. S.; HOORNA, J.; SALDANHA, F. G. Solving the job-shop scheduling problem optimally by dynamic programming. **Computers & Operations Research**. 2012.

GUPTA J. A functional heuristic algorithm for the flow-shop scheduling problem. **Operational Research Quarterly**, v. 22, p.39-47, 1978.

HO, J. C. Flowshop sequencing with mean flowtime objective. **European Journal of Operational Research**, v. 81, n. 3, p. 571-8, 1995.

HUDANL, T.; RAJGOPAL, J. An extension of Palmer's heuristic for the flow-shop scheduling problem. **International Journal of Production Research**, v. 26, p. 1119-1124, 1988.

JARBOUI, B.; EDDALY, M.; SIARRY, P. An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. **Computers & Operations Research**, v. 36, n. 9, p. 2638-2646, 2009.

JOHNSON, L. A.; MONTGOMERY, D. C. **Operations Research in Production Planning, Scheduling and Inventory Control**. New York. Wiley, 1974.

JOHNSON, S. M. Optimal two- and three-stage production schedules with setup times included. **Naval Research Logistics Quarterly**, v. 1, p. 61-68, 1954.

KALCZYNSKI, P. J; KAMBUROWSKI, J. An empirical analysis of the optimality rate of *flow shop* heuristics. **European Journal of Operational Research**, v. 198, n. 1, p. 93 – 101, 2009.

KALCZYNSKI, P. J; KAMBUROWSKI, J. An improved NEH heuristic to minimize *makespan* in permutation *flow shops*. **Computers & Operations Research**, v. 35. n. 9, p. 3001-3008, 2008.

KALCZYNSKI, P. J; KAMBUROWSKI, J. On recent modifications and extensions of the NEH heuristic for *flow shop* sequencing. **Foundations of Computing and Decision Sciences**, v. 36, n. 1, p. 17-34, 2011.

KALCZYNSKI, P. J; KAMBUROWSKI, J. On the NEH heuristic for minimizing the *makespan* in permutation *flow shops*. OMEGA, **The international Journal of Management Science**, v. 35, n. 1, p. 53-60, 2007.

KOULAMAS, C. A new constructive heuristic for the flowshop scheduling problem. **European Journal of Operational Research**, v. 105, p. 66-71, 1996.

LAHA, D; SARIN S. C. A heuristic to minimize *total flowtime* in permutation *flow shop*. OMEGA, **International Journal of Management Science**, v. 37, n. 3, p. 734-9, 2009.

LI, X. P; WANG, Q; WU, C. Efficient composite heuristic for *total flowtime* minimization in permutation *flow shops*. OMEGA, **International Journal of Management Science**, v. 37, n. 1, p. 155-64, 2009.

LI, X.; LIU, L.; WU, C. A fast method for heuristics in large-scale flow shop scheduling. **Tsinghua Science & Technology**, v. 11, n. 1, p. 12-18.

LI, X.; WANG, Q. Iterative heuristics for permutation *flow shops* with *total flowtime* minimization. In: Shen W, editor. Information technology for balanced manufacturing systems. IFIP TC5. **Proceedings** of the WG 5.5 seventh international conference on information technology for balanced automation systems in manufacturing and services. New York: Springer, p. 349-56, 2006.

LI, X. P.; WANG, Y. X. Heuristic algorithms for large flowshop scheduling problems. In **proceedings** of the 5th world congress on intelligent control and automation. Hangzhou, China. p. 2999-3003, 2004.

LIU, J.; REEVES, C. R. Constructive and composite heuristic solutions to the $P \parallel \sum C_j$ scheduling problem. **European Journal of Operational Research**, v. 132, p. 439-52, 2001.

LOW, C.; YEH, J. Y.; HUANG, K. I. A robust simulated annealing heuristic for *flow shop* scheduling problems. **International Journal of Advanced Manufacturing Technology**, v. 23, p. 762-767, 2004.

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

MIYAZAKI S.; NISHIYAMA N.; HASHIMOTO F. Na adjacente pairwise approach to the mean flow-time scheduling problem. **Journal of the Operations Research Society of Japan**, v. 21, n. 2, p. 287-299, 1978.

MLADENOVIC, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, v. 24, n. 11, p. 1097, 1997.

NAGANO, M.; MOCCELLIN, J. A high quality solution constructive heuristic for the *flow shop* sequencing. **Journal of the Operational Research Society**, v. 53, n. 12, p. 1374-1379, 2002.

NAGANO, M; MOCCELLIN, J. Reducing mean *flowtime* in permutation *flow shop*. **Journal of the Operational Research Society**, v. 59, p. 939-945, 2008.

NAWAZ, M.; ENSCORE, J. R. E. E.; HAM, I. A Heuristic Algorithm for the m-Machine, n-Job *Flow shop* Sequencing Problem. **OMEGA, The International Journal of Management Science**, v. 11, n. 1, p. 91-95, 1983.

PALMER, D. Sequencing jobs through a multi-stage process in the minimum total time – A quick method of obtaining a near optimum. **Operational Research Quarterly**, v. 16 ,p .101-107, 1965.

PAN, Q. K.; TASGETIREN, M. F.; LIANG, Y. C. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. **Computers & Industrial Engineering**, v. 55, n. 4, p. 795-816, 2008.

PAN, Q, RUIZ, R. A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. **Computers & Operations Research**, v. 40, p. 117-128, 2013.

PINEDO, M. 2008. **Theory, Algorithms, and Systems**. 3 ed. New Jersey, Prentice-Hall, Inc.

RAD, S. F.; RUIZ, R.; BOROOJERDIAN, N. New high performing heuristics for minimizing *makespan* in permutation *flow shops*. **OMEGA, The international Journal of Management Science**, v. 37, n. 2, p. 331-345, 2009.

RAJENDRAN, C.; CHAUDHURI, D. A flowshop scheduling algorithm to minimize total flowtime. **Journal of the Operations Research Society of Japan**, v. 34, n. 1, p. 28-46, 1991.

RAJENDRAN, C. Heuristic algorithm for scheduling algorithm to minimize total flowtime. **International Journal of Production Economics**, v. 29, n, 1, p. 65-73, 1993.

RANJENDRAN, C.; ZIEGLER, H. An efficient heuristic for scheduling in a *flow shop* to minimize total weighted *total flowtime* of jobs. **European Journal of Operational Research**, v. 103, p. 129-138, 1997.

RUIZ, R.; STÜTZLE, T. A simple and effective iterated greedy algorithm for the permutation flow-shop scheduling problem. **Eur J Oper Res**, v. 177, n. 466, p. 2033–49, 2007.

SEVAST'JANOV, S. Vector summation in Banach space and polynomial algorithms for *flow shops* and open shops, **Mathematics of Operations Research**, v. 20, p. 90-103, 1995.

SLACK, N. **Administração da produção**. 3 ed. São Paulo. Atlas, 2009.

TAILLARD, E. Benchmarks for basic scheduling problems. **European Journal of Operational Research**, v. 64, n. 2, p. 278-85, 1993.

TAILLARD, E. Some efficient methods for the *flow shop* sequencing problem. **European Journal of Operational Research**, v. 47, n. 1, p. 65-74, 1990.

TASGETIREN, M. F.; LIANG, Y. C.; SEVKLI, M.; GENCYILMAZ, G. A particle swarm optimization algorithm for *makespan* and *total flowtime* minimization in the permutation *flow shop* sequencing problem. **European Journal of Operational Research**, v. 177, n. 3, p. 1930-47, 2007.

VALLADA, E.; RUIZ, R.; FRAMINAN, J. New hard benchmark for flowshop scheduling problems minimising makespan. **European Journal of Operational Research**, 2014.

WANG, C. G.; CHU, C. B.; PROTH, J. M. Heuristic approaches for $n|m|F|\sum C_i$, scheduling problems. **European Journal of Operational Research**, v. 96, n. 3, p. 636-44, 1997.

WOO, H. S.; YIM, D. S. A heuristic algorithm for mean flowtime objective in flowshop scheduling. **Computers & Operations Research**, v. 25, n. 3, p. 175-82, 1998.

YING, K. C.; LIN, S. W. A high-performing constructive heuristic for minimizing *makespan* in permutation *flow shops*. **Journal of Industrial and Production Engineering**. 2013.

ZHANG, Y.; LI, X. P.; WANG, Q. Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. **European Journal of Operational Research**, v. 196, n. 3, p. 869-876, 2009.