Universidade de São Paulo
Escola de Engenharia de São Carlos - EESC
Departamento de Engenharia de Produção

GUILHERME BARQUETE ZUCCOLOTTO

ANÁLISE DA INFLUÊNCIA DOS TEMPOS DE PROCESSAMENTO EM HEURÍSTICAS CONSTRUTIVAS EM AMBIENTE DE *FLOW SHOP* PERMUTACIONAL COM CRITÉRIO DE MINIMIZAÇÃO DO *MAKESPAN*

SÃO CARLOS - SP
2017

.

GUILHERME BARQUETE ZUCCOLOTTO

ANÁLISE DA INFLUÊNCIA DOS TEMPOS DE PROCESSAMENTO EM HEURÍSTICAS CONSTRUTIVAS EM AMBIENTE DE *FLOW SHOP* PERMUTACIONAL COM CRITÉRIO DE MINIMIZAÇÃO DO *MAKESPAN*

Trabalho de Conclusão de Curso apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo como parte dos requisitos para obtenção do título de Graduado em Engenharia de Produção Mecânica.

Orientador: Dr. Marcelo Seido Nagano

SÃO CARLOS - SP

2017

.

Z943a

Zuccolotto, Guilherme Barquete
Análise da influência dos tempos de processamento em heuristicas construtivas em ambiente de flow shop permutacional com critério de minimização do makespan / Guilherme Barquete Zuccolotto; orientador Marcelo Seido Nagano; coorientador Fernando Luís Rossi. São Carlos, 2017.

Monografia (Graduação em Engenharia de Produção Mecânica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2017.

1. Programação da Produção. 2. Flow Shop. 3. Flow Shop Permutacional. 4. Métodos Heuristicos. 5. Makespan. 6. Influência. 7. Tempo de Processamento. I. Título.

EESC · USP
Escola de Engenharia de São Carlos
Departamento de Engenharia de Produção

ATET/CoC-EPM/SEP/162/2017

ATESTADO

ATESTO, para os devidos fins, que o **Sr. Guilherme Barquete Zuccolotto** apresentou o Trabalho de Conclusão de Curso intitulado: **"Análise da influência dos tempos de processamento em heurísticas construtivas em ambiente de *Flow Shop permutacional com critério de minimização do MAKESPAN*"**, junto ao Curso de Engenharia de Produção desta Escola, perante a Comissão Julgadora constituída pelos: Prof. Dr. Marcelo Seido Nagano – SEP-EESC-USP(orientador), Prof. Assoc. Antônio Freitas Rentes – SEP – EESC – USP, Doutorando Fernando Luis Rossi – SEP – EESC – USP em sessão pública na data de 17/11/2017, tendo sido aprovado.

Prof. Dr. Marcel Andreotti Musetti

Coordenador da Comissão Coordenadora do Curso de Graduação em Engenharia de Produção Mecânica

Epígrafe

“You may face a mistake like bullshit to be forgotten,
or as a result that points a new direction.”
Steve Jobs

.

Agradecimentos

À Universidade de São Paulo por proporcionar um ensino de qualidade e a oportunidade de escrever sobre um assunto que gosto e me dediquei muito para aprender.

Ao meu orientador professor doutor Marcelo Seido Nagano, ao meu coorientador Fernando Luís Rossi, e também a todos os colegas do laboratório de pesquisa operacional aplicada (LAOR) do departamento de Engenharia de Produção da Universidade de São Paulo – campus de São Carlos que me auxiliaram em toda a parte técnica do meu projeto.

À minha mãe, Christianne, ao meu pai, Wagner, ao meu irmão, Leonardo, e à minha namorada, Caroline, por sempre estarem ao meu lado me incentivando e nunca deixando que eu desista dos meus sonhos, me motivando a continuar seguindo firme rumo aos meus objetivos e a encarar de frente os desafios que a vida coloca em meu caminho, porque com certeza sem eles este trabalho não seria possível.

Aos meus colegas de trabalho, da Hominiss Consulting, por estarem sempre me incentivando a aprender cada vez mais com as experiências que a vida me proporciona.

E por fim, mas não menos importante, à Deus, pois sem a fé em acreditar que um dia atingiria um dos meus maiores objetivos de vida, que é a graduação na melhor faculdade do país, eu não conseguiria concluir este trabalho.

.

Resumo

ZUCCOLOTTO, G, B. **Análise da influência dos tempos de processamento em heurísticas construtivas em ambiente de *flow shop* permutacional com critério de minimização do *makespan***, 2017. Dissertação (conclusão de curso) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2017.

Neste trabalho é abordado o problema da programação da produção em ambiente *flow shop* permutacional com critério de minimização do *makespan* (tempo total de programação da produção). Muitas heurísticas já foram propostas para o problema e todas são comparadas com as da literatura utilizando o conjunto de problemas teste de Taillard, mas até hoje não foram feitas comparações entre as heurísticas considerando diferentes distribuições de tempos de processamento entre as tarefas. Por este motivo, este trabalho tem como objetivo verificar se diferentes distribuições de tempos de processamentos apresentam influência sobre os desempenhos das heurísticas construtivas. Isto foi realizado por meio da geração de problemas testes de programação da produção em ambiente *flow shop* permutacional considerando outras distribuições. Esta análise é importante pois um sequenciamento de atividades em que o tempo total de programação é minimizado, reduz o *Lead Time* da linha de produção e aumenta a eficiência dos recursos produtivos. Como as análises apresentam uma inconsistência nos resultados, com uma alta variação de desvios relativos entre os *makespan*, é possível concluir que este trabalho poderá contribuir para a geração de um melhor conjunto de problemas testes para o problema abordado.

Palavras-chave: programação da produção, *flow shop*, *flow shop* permutacional, métodos heurísticos, *makespan*, influência, tempo de processamento.

.

Abstract

ZUCCOLOTTO, G, B. **Análise da influência dos tempos de processamento em heurísticas construtivas em ambiente de *flow shop* permutacional com critério de minimização do *makespan***, 2017. Dissertação (conclusão de curso) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2017.

This work addresses the scheduling problem in permutation flow shop environment with the criterion of minimize the makespan. Many heuristics have already been proposed for the problem and all are compared with those of the literature using the set of Taillard test problems, but to date no comparisons have been made between heuristics considering different distributions of processing times between tasks. For this reason, this work has as objective to verify if different distributions of processing times have influence on the performance of constructive heuristics. It was accomplished through the generation of test problems of production scheduling in the permutation flow shop environment considering other distributions. This analysis is important because an activity sequencing in which the total scheduling time is minimized reduces the lead time of the production line and increases the efficiency of productive resources. As the analyzes show an inconsistency in the results, with a high variation of the relative deviations between makespan, it is possible to conclude that this work could contribute to the generation of a better set of test problems for the problem addressed.

Keywords: Scheduling, flow shop, permutation flow shop, heuristics methods, makespan, influence, processing time.

.

SUMÁRIO

.

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 os atender 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 observar e comparar as influências geradas pelo tempo de processamento dos melhores métodos heurísticos construtivos. Será abordado, de forma mais específica, quais intervalos são gerados por cada método heurístico, defini-los e otimizar então a forma de avaliação do problema gerado em ambiente *flow shop* permutacional focando a minimização do *makespan*.

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 a seguinte:

- Planejamento da Produção:
    - Planejamento agregado da produção;
    - Planejamento da Capacidade;
    - Planejamento Mestre da Produção (MPS)
    - Previsão de demanda.
- Controle da Produção:
    - Controle da Produção e de Materiais;
- Programar/sequenciar as tarefas nas máquinas

A definição do sistema produtivo a ser empregado na produção dos produtos e/ou serviços é um dos primeiros passos a seguir na programação da produção. Baseando-se 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, são produzidos grandes volumes de poucos tipos de produtos similares. No sistema grande projeto, são feitos produtos muitas vezes únicos, com um elevado grau de complexidade e variados. Já no sistema intermitente, nos estágios produtivos ocorrem mudanças de um produto para outro, como consequência da alta gama de produtos. Dentro desta última categoria se enquadram os ambientes máquina única, máquinas em paralelo, *flow shop*, *job shop*, *open shop*. A Figura 1 mostra a relação hierárquica entre os ambientes de produção proposta por MacCarthy e Liu (1993).

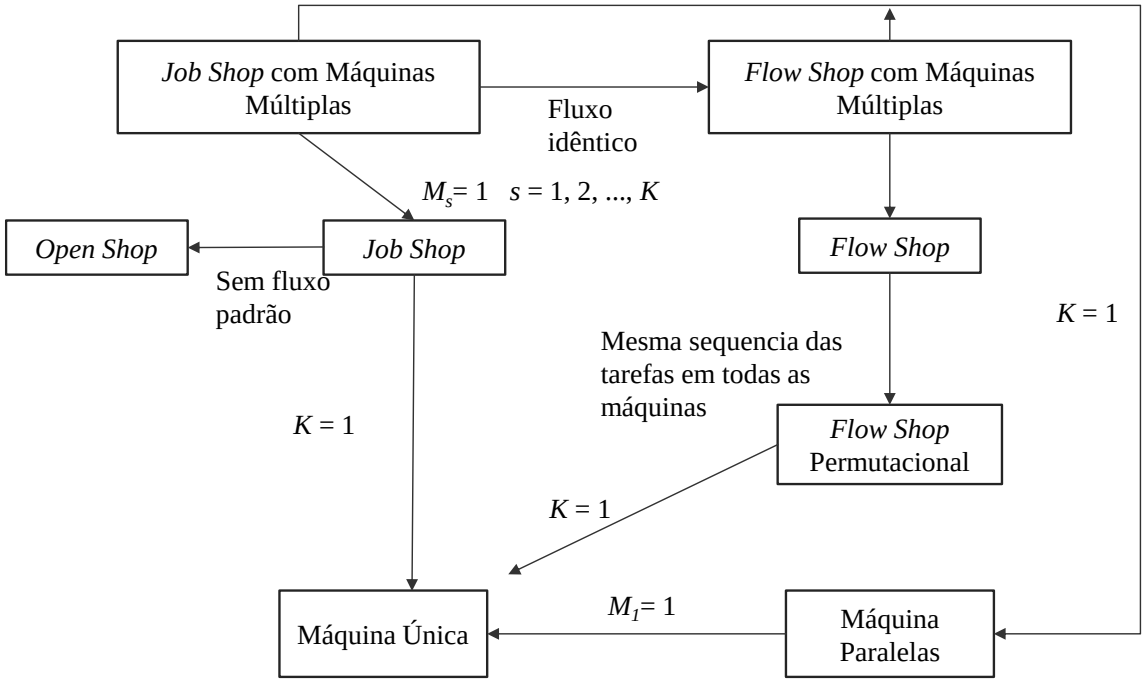

Figura 1 - Relação hierárquica entre os ambientes de produção. Fonte: adaptado de MacCarthy e Liu (1993).

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..., J_n} que têm de serem processadas em *m* máquinas {M_1, M_2, ..., 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 α | β | γ. 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($C_{1, ..., } C_n$) 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, ..., 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 $\alpha \mid \beta \mid \gamma$, será estudado apenas o problema $F_m \mid prmu \mid C_{max}$.

1.2 MOTIVAÇÃO E JUSTIFICATIVA DA PESQUISA

No desenvolvimento de heurísticas para o problema *flow shop* permutacional com critério de minimização do *makespan*, as heurísticas propostas são comparadas com as da literatura utilizando o conjunto de problemas testes de Taillard (1990), que consiste em 120 problemas testes com até 500 tarefas e 20 máquinas em uma distribuição uniforme de tempo de processamento, variando de 1 a 100.

Mesmo que, recentemente, diversas heurísticas tenham sido propostas para o problema, até o momento não foram realizadas comparações entre as heurísticas considerando outras distribuições de tempos de processamento, por exemplo: 1 a 25, 25 a 50 ou 1 a 75. Portanto, uma heurística que tenha desempenho superior no conjunto de problemas teste de Taillard pode vir a apresentar soluções inferiores, ou vice-versa, quando problemas testes, significamente diferentes, são utilizados para fins de comparação.

A partir desta constatação, é de fundamental importância investigar se há a necessidade ou não da consideração de outras distribuições de tempos de processamento na geração de problemas testes.

Sendo assim, este trabalho se propõe, através de experimentos computacionais e estatísticos, avaliar se as diferentes distribuições de tempos de processamento têm influência sobre os resultados obtidos nas heurísticas.

1.3 OBJETIVO DA PESQUISA

Este trabalho tem como objetivo verificar se diferentes distribuições de tempos de processamento alteram ou não os resultados obtidos das heurísticas construtivas. Isto será realizado por meio da geração de problemas testes de programação da produção em ambiente *flow shop* permutacional considerando outras distribuições.

1.4 METODOLOGIA DE REVISÃO DA LITERATURA

O desenvolvimento da pesquisa teve início na delimitação da pesquisa, seguido por um processo de busca e por fim, uma filtragem das publicações.

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

Inicialmente durante o processo de busca, realizou-se uma pesquisa por publicações no idioma inglês 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;

A medida em que as publicações foram sendo encontradas, o processo de busca se deu, então, 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 março de 2016 até junho de 2016.

As *strings* utilizadas para buscas foram "*permutation flow shop flowshop flow-shop makespan*", que poderiam estar contidos no título do artigo.

Na fase final, a seleção das publicações passou por um processo 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 o problema $F_m \mid prmu \mid C_{max}$.

2 PROBLEMA DE PROGRAMAÇÃO FLOW SHOP

Os problemas de programação de operações *flow shop* encontrados nos ambientes industriais são, frequentemente, complexos e apresentam características singulares a cada empresa.

Quando tratados no âmbito da Pesquisa Operacional, esses problemas são representados, de forma majoritária, por modelos matemáticos contendo as variáveis e os parâmetros que explicam significativamente o comportamento dos problemas tratados.

Segundo a literatura, o problema *flow shop* apresentas as seguintes principais hipóteses simplificadoras:

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.

O problema de programação de operações *flow shop* permutacional, como já explicado, 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*). O objetivo dessa solução é importante nas indústrias, pois lida com decisões relevantes que afetam o desempenho dos sistemas produtivos. A Figura 2 a seguir exemplifica um Diagrama de Gantt de uma possível solução para um problema de programação da produção *flow shop* permutacional com duas máquinas e três tarefas. Lembrando que é muito utilizado o Diagrama de Gantt na representação de uma solução de um problema de programação da produção.

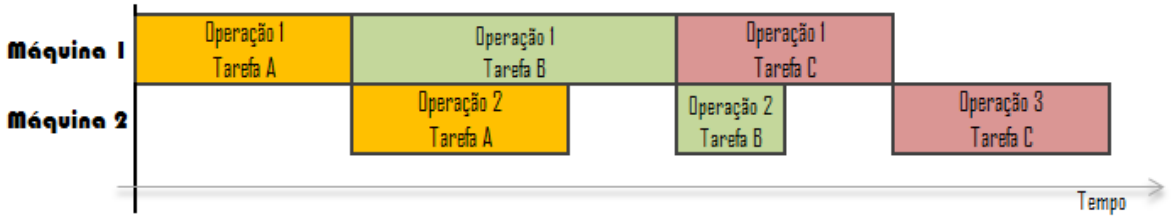

Figura 2 - Exemplo de um Diagrama de Gantt. Fonte: Autor, 20l7.

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).

Por outro lado, 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 descrevê-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.

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

O problema a ser estudado nesta dissertação é o problema *flow shop* permutacional com minimização do *makespan*. Nesta seção serão estudadas e avaliadas as influências dos tempos de processamento das operações das principais heurísticas construtivas para o problema *flow shop* permutacional.

3.1 DEFINIÇÃO DO PROBLEMA

O problema em consideração pode ser definido da seguinte maneira. Seja $\pi = \{J_1, J_2, ..., J_j, ..., J_n\}$ 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} ($j = 1, 2, ..., n$; $i = 1, 2, ..., m$). 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.

$$C_{ij} = \max\left(C_{i-1j}, C_{ij-1}\right) + p_{ij} \quad (1)$$
$$(i = 1, \dots, m) \quad (j = 1, \dots, n)$$
$$C_{0j} = C_{i0} = 0$$
$$C_{max} = C_{mn}$$

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.

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

(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.

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

(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.

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

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

$$\begin{gathered} C_{max} = \max(f_{ik}, q_{ik}) \\ (k = 1, \dots, j) \quad (i = 1, \dots, m) \end{gathered} \tag{5}$$

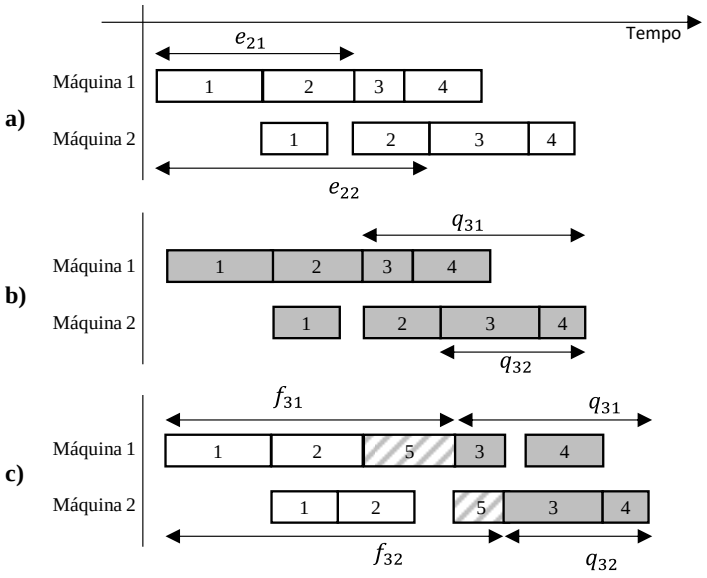

Figura 3 - Ilustração do método: inserção da tarefa 5 na 3ª posição. Fonte: Rossi, 2014.

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.

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

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, ..., 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 máquina, 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.

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

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.

$$z_j = \frac{A}{\min\limits_{1 \leq i \leq m-1} (p_{ij} + p_{i+1j})} \qquad \text{para } j = 1, 2, \ldots, n \tag{8}$$

Em que:

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

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.

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

Nawaz, Enscore 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 tarefas, 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.

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

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.

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

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

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.

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

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 Moccellin (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, ..., *n* – 1, tem-se:

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

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

$$UBX_{uv}^{1} = 0 \; e \; UBX_{uv}^{i+1} = \max\{0, UBX_{uv}^{i} + \left(p_{iv} - p_{i+1,u}\right) \quad \text{para } i = 1, 2, \dots\ m-1 \tag{17}$$

O valor de UBX_{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 Moccellin (2002). O índice c_j é calculado pelas Equações 18, 19 e 20.

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

Em que:

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

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

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.

$$AVG_j = \frac{1}{m} \sum_{i=1}^{m} p_{ij} \qquad \text{para } j = 1, 2, \ldots, n \tag{21}$$

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

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.

$$a_j = P_j + U_j \qquad \text{para } j = 1, 2, \ldots, n \tag{23}$$

$$b_j = P_j - U_j \qquad \text{para } j = 1, 2, \ldots, n \tag{24}$$

Em que:

$$U_j = \sum_{h=1}^{s} \left(\frac{h - 3/4}{s - 3/4} \right) \left(p_{s+1-h,j} - p_{t+h,j} \right) \qquad \text{para } j = 1, 2, \ldots, n$$
$$s = \lfloor m/2 \rfloor \quad e \quad t = \lceil m/2 \rceil \tag{25}$$

Mais recentemente Kalczynski e Kamburowski (2011) demonstrou que a heurística NEH KK2 é superior às modificações do NEH propostas por Kalczynski e Kamburowski (2008) e Dong, Huang e Chen (2008) em um conjunto de problemas testes proposto por Kalczynski 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.

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

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 custas 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*.

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.

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 cargas das máquinas são fornecidas pela Equação 27.

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

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, ..., J_{L-1}), e que a L-ésima tarefa da ordenação inicial é r. Faça $l^* = 1$ e $\pi^* = (r, J_1, J_2, ..., J_{L-1})$. Para $l = 2, 3, ..., L$ faça Se $C_{max}((J_1, ..., r, J_l, ..., J_{L-1})) \leq Cmax(\pi^*)$ e $Cmax((J_{l^*}, ..., J_{l-1}, r)) < Cmax(r, Jl, ..., J_{l-1})$, então faça $l^* = l$ e $\pi^* = (J_1, ..., r, J_l, ..., J_{L-1})$.

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.

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

Em que:

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

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.

3.2.4 CONSIDERAÇÕES DA REVISÃO DA LITERATURA

Após análise da literatura, destacam-se alguns aspectos relevantes, segundo Rossi (2014):

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;

i. 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);
ii. Na geração da sequência, as heurísticas FRB3, INEH KK2D superam o método NEH em termos de qualidade de solução;
iii. 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).

A Tabela 1 apresenta uma síntese das heurísticas construtivas desenvolvidas até o presente momento.

Na próxima seção, as principais heurísticas revisadas neste trabalho serão comparadas e analisadas a fim de identificar se suas performances são influenciadas pela distribuição do tempo de processamento e variação da quantidade de máquinas e tarefas.

Tabela 1 - Variações da heurística NEH e suas composições (parte 1). Fonte: Rossi, 2014

Autor	**Ano**	**Acrônimo**	**Fase I**	**Fase II**	
			Ordenação Inicial	**Desempate**	**Construção da Sequência**
Nawaz, Enscore JR e HAM	1983	NEH	LPT	-	NEH
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
Low, Yeh e Huang	2004	NEHL	LPT	Escolhe a sequencia que minimiza o tempo ocioso na máquina M_{i*} com a carga máxima. $L_i = \sum_{j=1}^{n} p_{ij}$	NEH
Kalczynski e Kamburowski	2007	NEH KK	LPT	Assumindo que a sequência parcial corrente é (J_1, J_2, ..., J_{L-1}), e que a L- ésima tarefa da ordenação inicial é r. Faça $l^* = 1$ e $\pi^* = (r, J_1, J_2, ..., J_{L-1})$. Para $l = 2, 3, ..., L$ faça Se $C_{max}((J_1, ..., r, J_l, ..., J_{L-1})) \leq Cmax(\pi^*)$ e $Cmax((J_{l*}, ..., J_{l-1}, r)) < Cmax(r, Jl, ..., J_{l-1})$, então faça $l^* = l$ e $\pi^* = (J_1, ..., r, J_l, ..., J_{L-1})$.	NEH
Gao e Zhang	2007	INEH	LPT	-	NEH + Reinserção de apenas uma tarefa selecionada criteriosamente

Tabela 1 - Continuação (parte 2).

Autor	**Ano**	**Acrônimo**	**Fase I**	**Fase II**	
			Ordenação Inicial	**Desempate**	**Construção da Sequência**
Kalczynski e Kamburowski	2008	NEH KK1	Calcula: $\hat{a}_j = \sum_{i=1}^{m} \lvert(m-1)(m-2)/2 + m - i\rvert p_{ij}$ $\hat{b}_j = \sum_{i=1}^{m} \lvert(m-1)(m-2)/2 + i - i\rvert p_{ij}$ Ordena as tarefas em ordem não crescente de c_j , onde $c_j = \hat{a}_j (c_j = \hat{b}_j)$ se $\hat{a}_j \leq \hat{b}_j$ $(\hat{a}_j > \hat{b}_j)$.	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
Dong e Chen	2008	NEH D	Ordena as tarefas pela soma da média e desvio padrão dos tempos de processamento.	Calcula $E_{\pi(x)} = \frac{1}{m}\sum_{i=1}^{m} \frac{p_{i,\pi(x)}}{S_{i,\pi(x+1)} - C_{i,\pi(x-1)}}$ $D_{\pi(x)} = \sum_{i=1}^{m} \left(\frac{p_{i,\pi(x)}}{S_{i,\pi(x+1)} - C_{i,\pi(x-1)}} - E_{\pi(x)} \right)^2$ Onde S e C são os tempos de realização das tarefas na data mais tarde e mais cedo, respectivamente. Escolhe a sequência candidata que minimiza o valor de $D\pi(x)$.	NEH
Kalczynski e Kamburowski	2009	NEH KK2	Ordena as tarefas em ordem decrescente de min (a_j, b_j), onde $a_j = P_j + U_j$, $b_j = P_j - U_j$. $U_j = \sum_{h=1}^{s} \left(\frac{h-3/4}{s-3/4}\right) \left(p_{s+1-h,j} - p_{t+h,j}\right)$ $s = \lfloor m/2 \rfloor$ e $t = \lceil m/2 \rceil$	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

Tabela 1 - Continuação (parte 3).

Autor	**Ano**	**Acrônimo**	**Fase I**	**Fase II**	
			Ordenação Inicial	**Desempate**	**Construção da Sequência**
Rad, Ruiz e Boroojerdian	2009	FRB3	LPT	-	NEH + Reinserção de todas as tarefas da sequência.
Rad, Ruiz e Boroojerdian	2009	$FRB4_L$	LPT	-	NEH + Reinserção das *L* primeiras tarefas da sequência corrente.
Kalczynski e Kamburowski	2011	INEH KK2D	NEH KK2	NEH D	INEH
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.
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

3.3 HEURÍSTICAS CONSTRUTIVAS PARA O PROBLEMA F | PRMU | CMAX

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 4.

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 π;
Insira a tarefa *j* na posição de π que resulte no menor C_{max};
fim
fim

Figura 4 – Heurística NEH. Fonte: Rossi, 2014

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 Moccellin (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 5 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 π;

Insira a tarefa j na posição de π que resulte no menor C_{max};

fim

fim

Figura 5 – Heurística NEH NM. Fonte: Rossi, 2014

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 6 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 π;
Insira a tarefa *j* na posição de π que resulte no menor C_{max};
Em caso de empate, aplique o desempate do NEH KK;
fim
fim

Figura 6 – Heurística NEH KK. Fonte: Rossi, 2014

i. 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 7 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 π;
Insira a tarefa *j* na posição de π 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 7 – Heurística NEH KK1. Fonte: Rossi, 2014

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 8 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 π;

Insira a tarefa j na posição de π 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 8 – Heurística NEH D. Fonte: Rossi, 2014

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 9 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 π;

Insira a tarefa *j* na posição de π 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 9 – Heurística NEH KK2. Fonte: Rossi, 2014

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 10 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 π.
Insira a tarefa *j* na posição de π que resulte no menor C_{max}.
para passo2 := 1 **até** n **faça**
Extraia a tarefa *h* da posição *passo2* de π .
Teste a tarefa *h* em todas as posições possíveis de π.
Insira a tarefa *h* na posição de π que resulte no menor C_{max}.
fim
fim
fim

Figura 10 – Heurística FRB3. Fonte: Rossi, 2014

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 11 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 π;
Insira a tarefa *j* na posição de π que resulte no menor C_{max};
para $passo2 := 1$ **até** n **faça**
$j := tarefa(passo2)$;

Retire a tarefa *j* da sequência π;

Calcule $W_j = \frac{Cmax(\pi) - Cmax(\pi - \{j\})}{P_j}$;

fim

Ordene W em ordem decrescente;

$h := tarefa(W[1])$;

Retire a tarefa *h* de π ;

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

Insira a tarefa h na posição de π 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 11 – Heurística INEH KK2D. Fonte: Rossi, 2014

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 12. Na próxima seção serão apresentadas as heurísticas construtivas propostas para o problema $F_m \mid prmu \mid C_{max}$.

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 12 - Procedimento NEH FF. Fonte: Rossi, 2014

Agora, para um melhor entendimento das heurísticas utilizados no processamento dos dados deste trabalho, segue uma breve explicação sobre elas:

- **NEH:** heurística de Nawaz, Enscore Jr e Ham (1983), ordenação implementada de acordo com o *Longest Processing Time* (LPT) e critério de desempate de acordo com a primeira posição de empate;
- **NEH D:** heurística proposta por Dong e Chen (2008);
- **NEH KK2:** heurística proposta por Kalczynski e Kamburowski (2009);
- **ORD NEH D:** heurística com ordenação implementada de acordo com a heurística NEH D e critério de desempate de acordo com a primeira posição de empate;
- **ORD NEH KK2:** heurística com ordenação implementada de acordo com a heurística NEH KK2 e critério de desempate de acordo com a primeira posição de empate;
- **ORD NEH NM:** heurística com ordenação implementada de acordo com a heurística NEH NM e critério de desempate de acordo com a primeira posição de empate;
- **ORD NEH SPT:** heurística com ordenação implementada de acordo com *Shortest Processing Time* (SPT) e critério de desempate de acordo com a primeira posição de empate;
- **TIE NEH D:** heurística com ordenação de acordo com a ordenação LPT e critério de desempate NEH D;
- **TIE NEH FF:** heurística com ordenação de acordo com a ordenação LPT e critério de desempate NEH FF
- **TIE NEH KK2:** heurística com ordenação de acordo com a ordenação LPT e critério de desempate NEH KK2;
- **TIE NEH LT:** heurística com ordenação de acordo com a ordenação LPT e critério de desempate de acordo com a última posição de empate.

Por meio de características em comum, as heurísticas foram divididas em 3 classes:

- **Classe 1:** heurísticas que utilizam tanto as extensões de desempate quanto as de ordenação diferentes. São elas: NEH, NEH D e NEH KK2;
- **Classe 2:** heurísticas que apresentação alteração na ordenação, a fim de avaliar os impactos que a ordenação causa, enquanto o critério de desempate é de acordo com a primeira posição de empate. São elas: ORD NEH D, ORD NEH KK2, ORD NEH NM e ORD NEH SPT;
- **Classe 3:** heurísticas que mantiveram as ordenações (utilizando a maior soma dos tempos de processamento das tarefas, também conhecido como LPT) e variaram o mecanismo de desempate a fim de avaliar os impactos nos resultados. São elas: TIE NEH D, TIE NEH FF, TIE NEH KK2 E TIE NEH LT.

3.4 IMPLEMENTAÇÃO DOS MÉTODOS

Para realizar os experimentos computacionais, foi gerado um conjunto de problemas testes considerando diferentes distribuições de tempo de processamento. As heurísticas selecionadas para avaliação (Seção 3.3) foram avaliadas utilizando o novo conjunto de problemas. Os resultados obtidos foram analisados estatisticamente. Desta forma, foi possível verificar se diferentes distribuições de tempos de processamento alteram ou não os resultados obtidos das heurísticas construtivas.

O conjunto de problemas teste foi organizado da seguinte maneira:

- Existem 97 bases;
- Cada base possui 150000 problemas testes;
- Cada problema teste possui uma quantidade *n* de tarefas e uma quantidade *m* de máquinas;
- Cada tarefa executada em cada máquina possui um tempo de processamento p_{mn}, que é diferente em cada problema teste.

Em cada uma das bases tem-se a seguinte numeração de tarefas (*n*) e máquinas (*m*):

- **Problemas testes de 1 a 10000:** 20 tarefas e 5 máquinas;
- **Problemas testes de 10001 a 20000:** 20 tarefas e 10 máquinas;
- **Problemas testes de 20001 a 30000:** 20 tarefas e 20 máquinas;
- **Problemas testes de 30001 a 40000:** 50 tarefas e 5 máquinas;
- **Problemas testes de 40001 a 50000:** 50 tarefas e 10 máquinas;
- **Problemas testes de 50001 a 60000:** 50 tarefas e 20 máquinas;
- **Problemas testes de 60001 a 70000:** 100 tarefas e 5 máquinas;
- **Problemas testes de 70001 a 80000:** 100 tarefas e 10 máquinas;
- **Problemas testes de 80001 a 90000:** 100 tarefas e 20 máquinas;
- **Problemas testes de 90001 a 100000:** 200 tarefas e 5 máquinas;
- **Problemas testes de 100001 a 110000:** 200 tarefas e 10 máquinas;
- **Problemas testes de 110001 a 120000:** 200 tarefas e 20 máquinas;
- **Problemas testes de 120001 a 130000:** 500 tarefas e 5 máquinas;
- **Problemas testes de 130001 a 140000:** 500 tarefas e 10 máquinas;
- **Problemas testes de 140001 a 150000:** 500 tarefas e 20 máquinas.

Cada base possui uma distribuição uniforme de tempos de processamento para cada um dos conjuntos das máquinas. A distribuição uniforme dos tempos de processamento foi gerada por números aleatórios através do algoritmo *Mersenne Twister*. Abaixo está indicado como foi feita a distribuição dos tempos de processamento em cada base, que são delimitados por um valor mínimo e um máximo através dessa distribuição:

- **Base 1:** 100% das tarefas têm distribuição [1, 100];
- **Base 2:** 100% das tarefas têm distribuição [1, 75];
- **Base 3:** 25% das tarefas têm distribuição [76, 100] e 75% têm distribuição [1, 75];
- **Base 4:** 50% das tarefas têm distribuição [76, 100] e 50% têm distribuição [1, 75];
- **Base 5:** 75% das tarefas têm distribuição [76, 100] e 25% têm distribuição [1, 75];
- **Base 6:** 100% das tarefas têm distribuição [1, 50];
- **Base 7:** 25% das tarefas têm distribuição [51, 100] e 75% têm distribuição [1, 50];
- **Base 8:** 50% das tarefas têm distribuição [51, 100] e 50% têm distribuição [1, 50];
- **Base 9:** 75% das tarefas têm distribuição [51, 100] e 25% têm distribuição [1, 50];

- **Base 10:** 75% das tarefas têm distribuição [1, 50] e 25% têm distribuição [75, 100];
- **Base 11:** 50% das tarefas têm distribuição [75, 100] e 50% têm distribuição [1, 50];
- **Base 12:** 75% das tarefas têm distribuição [75, 100] e 25% têm distribuição [1, 50];
- **Base 13:** 100% das tarefas têm distribuição [1, 25];
- **Base 14:** 25% das tarefas têm distribuição [26, 100] e 75% têm distribuição [1, 25];
- **Base 15:** 50% das tarefas têm distribuição [26, 100] e 50% têm distribuição [1, 25];
- **Base 16:** 75% das tarefas têm distribuição [26, 100] e 25% têm distribuição [1, 25];
- **Base 17:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [26, 75];
- **Base 18:** 50% das tarefas têm distribuição [26, 75] e 50% têm distribuição [1, 25];
- **Base 19:** 75% das tarefas têm distribuição [26, 75] e 25% têm distribuição [1, 25];
- **Base 20:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [26, 50];
- **Base 21:** 50% das tarefas têm distribuição [26, 50] e 50% têm distribuição [1, 25];
- **Base 22:** 75% das tarefas têm distribuição [26, 50] e 25% têm distribuição [1, 25];
- **Base 23:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [50, 75];
- **Base 24:** 50% das tarefas têm distribuição [50, 75] e 50% têm distribuição [1, 25];
- **Base 25:** 75% das tarefas têm distribuição [50, 75] e 25% têm distribuição [1, 25];
- **Base 26:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [50, 100];
- **Base 27:** 50% das tarefas têm distribuição [50, 100] e 50% têm distribuição [1, 25];
- **Base 28:** 75% das tarefas têm distribuição [50, 100] e 25% têm distribuição [1, 25];
- **Base 29:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [75, 100];
- **Base 30:** 50% das tarefas têm distribuição [75, 100] e 50% têm distribuição [1, 25];
- **Base 31:** 75% das tarefas têm distribuição [75, 100] e 25% têm distribuição [1, 25];
- **Base 32:** 100% das tarefas têm distribuição [26, 100];
- **Base 33:** 75% das tarefas têm distribuição [26, 100] e 25% têm distribuição [1, 25];
- **Base 34:** 50% das tarefas têm distribuição [26, 100] e 50% têm distribuição [1, 25];
- **Base 35:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [26, 100];
- **Base 36:** 100% das tarefas têm distribuição [26, 75];
- **Base 37:** 75% das tarefas têm distribuição [26, 75] e 25% têm distribuição [1, 25];
- **Base 38:** 50% das tarefas têm distribuição [26, 75] e 50% têm distribuição [1, 25];
- **Base 39:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [26, 75];
- **Base 40:** 75% das tarefas têm distribuição [26, 75] e 25% têm distribuição [76, 100];

- **Base 41:** 50% das tarefas têm distribuição [26, 75] e 50% têm distribuição [76, 100];
- **Base 42:** 75% das tarefas têm distribuição [76, 100] e 25% têm distribuição [26, 75];
- **Base 43:** 100% das tarefas têm distribuição [26, 50];
- **Base 44:** 75% das tarefas têm distribuição [26, 50] e 25% têm distribuição [1, 25];
- **Base 45:** 50% das tarefas têm distribuição [1, 25] e 50% têm distribuição [26, 50];
- **Base 46:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [26, 50];
- **Base 47:** 75% das tarefas têm distribuição [26, 50] e 25% têm distribuição [51, 75];
- **Base 48:** 50% das tarefas têm distribuição [51, 75] e 50% têm distribuição [26, 50];
- **Base 49:** 75% das tarefas têm distribuição [51, 75] e 25% têm distribuição [26, 50];
- **Base 50:** 75% das tarefas têm distribuição [26, 50] e 25% têm distribuição [51, 100];
- **Base 51:** 50% das tarefas têm distribuição [51, 100] e 50% têm distribuição [26, 50];
- **Base 52:** 75% das tarefas têm distribuição [51, 100] e 25% têm distribuição [26, 50];
- **Base 53:** 75% das tarefas têm distribuição [26, 50] e 25% têm distribuição [75, 100];
- **Base 54:** 50% das tarefas têm distribuição [75, 100] e 50% têm distribuição [26, 50];
- **Base 55:** 75% das tarefas têm distribuição [75, 100] e 25% têm distribuição [26, 50];
- **Base 56:** 100% das tarefas têm distribuição [50, 100];
- **Base 57:** 75% das tarefas têm distribuição [50, 100] e 25% têm distribuição [1, 25];
- **Base 58:** 50% das tarefas têm distribuição [50, 100] e 50% têm distribuição [1, 25];
- **Base 59:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [50, 100];
- **Base 60:** 75% das tarefas têm distribuição [50, 100] e 25% têm distribuição [1, 49];
- **Base 61:** 50% das tarefas têm distribuição [50, 100] e 50% têm distribuição [1, 49];
- **Base 62:** 75% das tarefas têm distribuição [1, 49] e 25% têm distribuição [50, 100];
- **Base 63:** 75% das tarefas têm distribuição [50, 100] e 25% têm distribuição [25, 49];
- **Base 64:** 50% das tarefas têm distribuição [50, 100] e 50% têm distribuição [25, 49];
- **Base 65:** 75% das tarefas têm distribuição [25, 49] e 25% têm distribuição [50, 100];
- **Base 66:** 100% das tarefas têm distribuição [50, 75];
- **Base 67:** 75% das tarefas têm distribuição [50, 75] e 25% têm distribuição [1, 25];
- **Base 68:** 50% das tarefas têm distribuição [1, 25] e 50% têm distribuição [50, 75];
- **Base 69:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [50, 75];
- **Base 70:** 75% das tarefas têm distribuição [50, 75] e 25% têm distribuição [1, 49];
- **Base 71:** 50% das tarefas têm distribuição [1, 49] e 50% têm distribuição [50, 75];

- **Base 72:** 75% das tarefas têm distribuição [1, 49] e 25% têm distribuição [50, 75];
- **Base 73:** 75% das tarefas têm distribuição [50, 75] e 25% têm distribuição [25, 49];
- **Base 74:** 50% das tarefas têm distribuição [25, 49] e 50% têm distribuição [50, 75];
- **Base 75:** 75% das tarefas têm distribuição [25, 49] e 25% têm distribuição [50, 75];
- **Base 76**: 75% das tarefas têm distribuição [50, 75] e 25% têm distribuição [76, 100];
- **Base 77:** 50% das tarefas têm distribuição [50, 75] e 50% têm distribuição [76, 100];
- **Base 78:** 75% das tarefas têm distribuição [76, 100] e 25% têm distribuição [50, 75];
- **Base 79:** 100% das tarefas têm distribuição [75, 100];
- **Base 80:** 75% das tarefas têm distribuição [75, 100] e 25% têm distribuição [1, 25];
- **Base 81:** 50% das tarefas têm distribuição [1, 25] e 50% têm distribuição [75, 100];
- **Base 82:** 75% das tarefas têm distribuição [1, 25] e 25% têm distribuição [75, 100];
- **Base 83:** 75% das tarefas têm distribuição [75, 100] e 25% têm distribuição [1, 50];
- **Base 84:** 50% das tarefas têm distribuição [1, 50] e 50% têm distribuição [75, 100];
- **Base 85:** 75% das tarefas têm distribuição [1, 50] e 25% têm distribuição [75, 100];
- **Base 86:** 75% das tarefas têm distribuição [75, 100] e 25% têm distribuição [1, 74];
- **Base 87:** 50% das tarefas têm distribuição [1, 74] e 50% têm distribuição [75, 100];
- **Base 88:** 75% das tarefas têm distribuição [1, 74] e 25% têm distribuição [75, 100];
- **Base 89:** 75% das tarefas têm distribuição [75, 100] e 25% têm distribuição [25, 50];
- **Base 90:** 50% das tarefas têm distribuição [25, 50] e 50% têm distribuição [75, 100];
- **Base 91:** 75% das tarefas têm distribuição [25, 50] e 25% têm distribuição [75, 100];
- **Base 92:** 75% das tarefas têm distribuição [75, 100] e 25% têm distribuição [25, 74];
- **Base 93:** 50% das tarefas têm distribuição [25, 74] e 50% têm distribuição [75, 100];
- **Base 94:** 75% das tarefas têm distribuição [25, 74] e 25% têm distribuição [75, 100];
- **Base 95:** 75% das tarefas têm distribuição [75, 100] e 25% têm distribuição [50, 74];
- **Base 96:** 50% das tarefas têm distribuição [50, 74] e 50% têm distribuição [75, 100];
- **Base 97:** 75% das tarefas têm distribuição [50, 74] e 25% têm distribuição [75, 100].

Pode-se notar que cada base tem uma distribuição uniforme diferente dos tempos de processamento, os quais variam entre um máximo e um mínimo. Dessa forma, tem-se uma grande diversidade de problemas testes distintos, que permitirão a obtenção de resultados

consistentes com relação ao desempenho das diversas heurísticas ao variar a distribuição do tempo de processamento das tarefas.

O tempo despendido para rodar essa grande quantidade de bases com 150000 problemas testes cada uma é muito elevado e, por isso, acaba levando meses para concluir o teste computacional de apenas uma heurística. Desta forma, depois de mais de um ano de estudo e análise, conseguiu-se realizar por completo todos os testes computacionais de todas as heurísticas estudadas (NEH, NEH D, NEH KK2, ORD NEH D, ORD NEH KK2, ORD NEH NM, ORD NEH SPT, TIE NEH D, TIE NEH KK2, TIE NEH FF e TIE NEH LT). A partir desses resultados obtidos, foi realizado um estudo estatístico comparando o desempenho dessas heurísticas (comparando tanto de forma conjunta como separadas em classes – heurísticas puras, com alteração na ordenação e com alteração no método de desempate) e também como a variação do tempo de processamento interfere no desempenho das heurísticas com extensões de ordenação e mecanismo de desempate.

3.5 EXPERIMENTAÇÃO COMPUTACIONAL E ANÁLISE DOS RESULTADOS DO PROBLEMA F | PRMU | CMAX

A partir da implementação dos métodos, obtivemos os resultados (*makespan*) de cada problema teste para cada heurística avaliada: NEH, NEH D, NEH KK2, ORD NEH D, ORD NEH KK2, ORD NEH NM, ORD NEH SPT, TIE NEH D, TIE NEH FF, TIE NEH KK2 e TIE NEH LT, as quais foram analisadas estatisticamente. Essa análise consiste na comparação de desempenho (qualidade da solução) entre as heurísticas, considerando a variação da distribuição do tempo de processamento e do número de máquinas e tarefas.

A ferramenta estatística utilizada para efetuar a comparação, em termos de qualidade da solução, entre as heurísticas foi o cálculo do desvio relativo da heurística *i* para o problema teste *k* (DR_{ik}), que pode ser escrito pela expressão 30:

$$DR_{ik} = \frac{(C_{max} - C_{max(min)})}{C_{max(min)}} \qquad (30)$$

onde C_{max} - *makespan* da heurística i para o problema teste *k*; $C_{max(min)}$- menor *makespan* entre todas as soluções encontradas para o problema teste *k.*

Desta forma, é possível chegar ao desvio relativo referente ao desempenho das heurísticas. Quanto mais próximo de zero for o desvio relativo de uma heurística, mais próxima ela estará da melhor solução encontrada para o problema teste e, portanto, essa heurística terá uma melhor qualidade de solução. Caso contrário, ela estará mais distante da melhor solução encontrada e, consequentemente, a sua qualidade de solução não será muito satisfatória.

Os Desvios Relativos Médios (DRM) calculados foram apresentados em um gráfico (Figura 12) e verificou-se que apenas uma heurística, ORD NEH SPT se destacou negativamente por apresentar resultados consideravelmente inferiores às demais. O que mostra que a ordenação SPT não é indicada para o problema de minimização do *makespan.* Já na Figura 13, temos os resultados de todas as heurísticas com exceção da ORD NEH SPT, onde pode-se observar que a performance das demais heurísticas são muito semelhantes, apresentando apenas uma divergência considerável nos resultados entres as bases 13 e 31, 33 a 40 e 56 a 62.

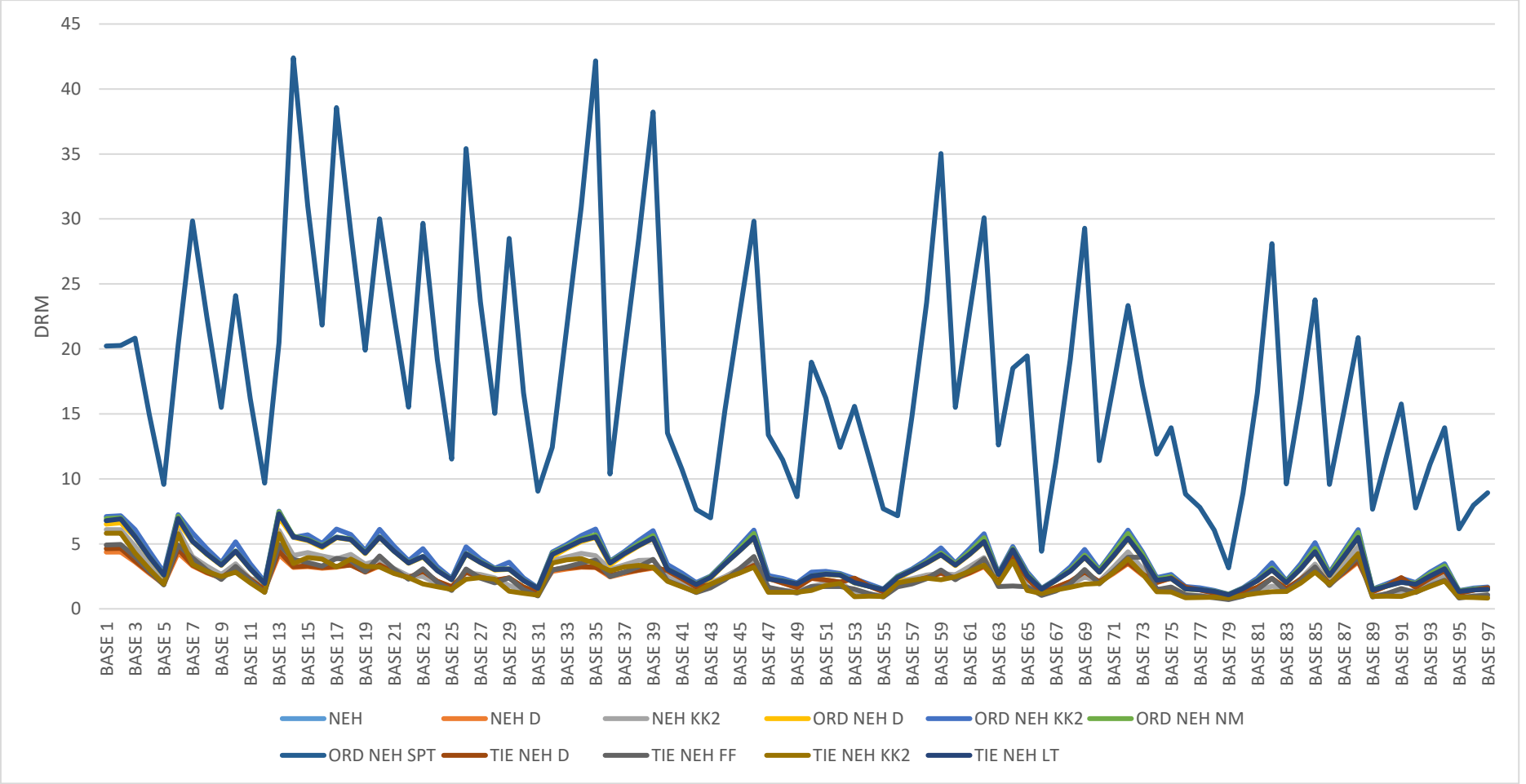

Figura 12 – Desvios Relativos Médios por base de problemas testes. Fonte: Autor, 2017

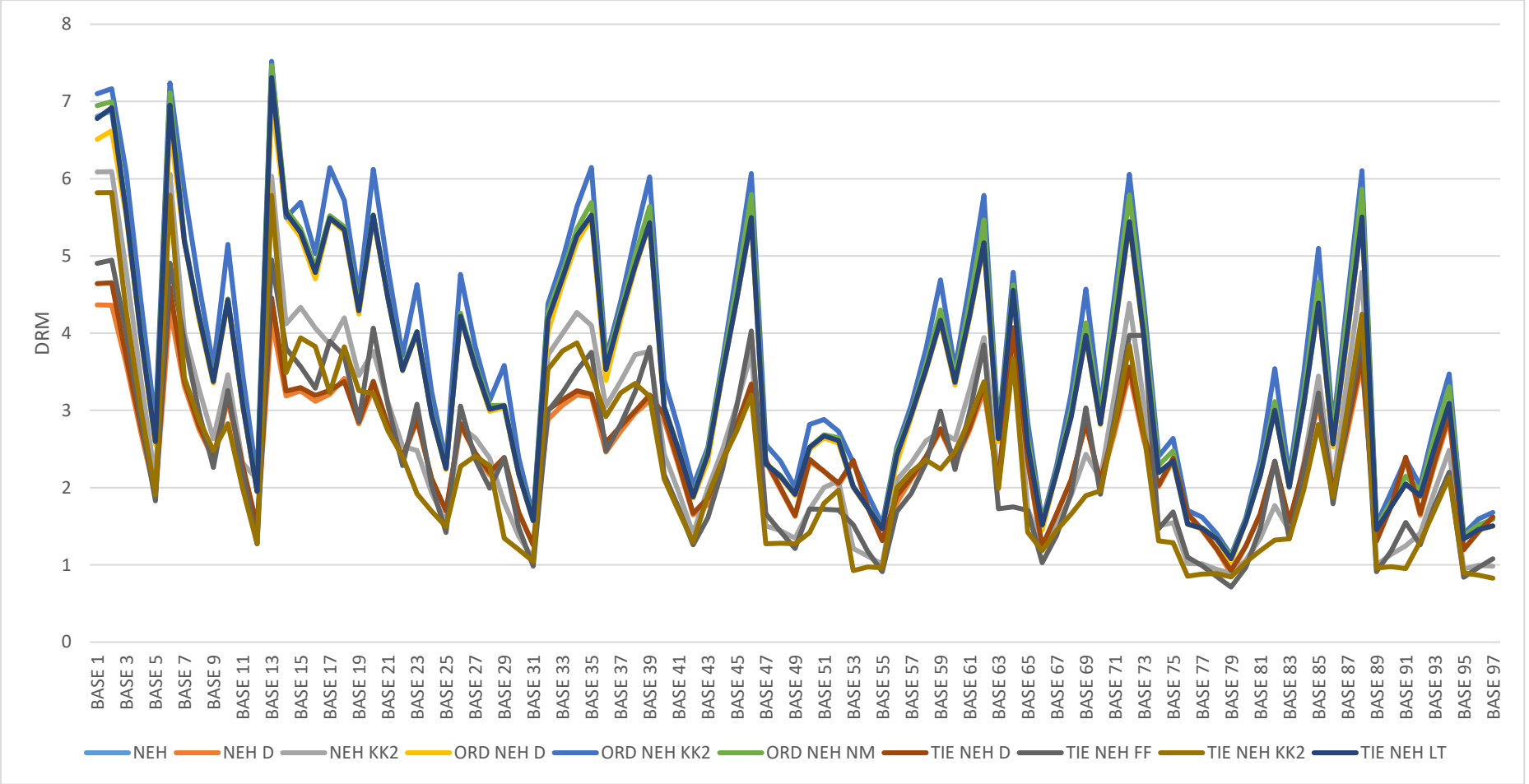

Figura 13 - Desvios Relativos Médios por base de problemas testes com exceção da ORD NEH SPT. Fonte: Autor, 2017

Mesmo que tenha uma inconsistência no desempenho das heurísticas, observando as curvas do gráfico, é possível verificar que a TIE NEH KK2 se destaca positivamente por apresentar resultados mais satisfatórios em grande parte da análise, enquanto a ORD NEH KK2 tem uma performance ligeiramente inferior às demais. Isso demonstra que extensões de ordenação e de mecanismos de desempate quando aplicadas, separadamente, em uma mesma heurística original, gera resultados divergentes.

Vale ressaltar também, que quanto menor o range de tempo de processamento distribuído entre as tarefas, encontrado em bases de números mais altos (93,94, 96 e 97), menor é o DRM resultante, ou seja, há uma dispersão menor dos valores.

Foi encontrada a média dos desvios relativos de cada heurística para cada tipo de problema, ou seja, para cada configuração de *n* tarefas e *m* máquinas estabelecidas para os problemas testes. Desta forma, também será possível observar como as heurísticas se comportaram a partir da variação destas quantidades (Tabela 2, os melhores DRMs estão destacados em negrito).

Tabela 2 – Melhores DRMs das heurísticas por distribuição de tarefas e máquinas.

n x m	NEH D	NEH KK2	ORD NEH D	ORD NEH KK2	ORD NEH LPT	ORD NEH NM	ORD NEH SPT	TIE NEH D	TIE NEH FF	TIE NEH KK2	TIE NEH LT
20x5	1,07	0,84	1,00	1,17	1,03	1,08	4,23	1,10	0,92	**0,68**	1,03
20x10	0,94	0,95	1,04	1,15	1,06	1,09	5,22	0,99	0,96	**0,86**	1,05
20x20	**0,68**	0,80	0,79	0,88	0,82	0,82	5,42	0,71	0,74	0,74	0,81
50x5	0,50	**0,30**	0,55	0,44	0,46	0,49	2,96	0,42	0,31	0,33	0,46
50x10	1,56	1,55	1,77	2,00	1,90	1,94	4,76	1,65	**1,27**	1,43	1,88
50x20	1,64	1,79	1,87	2,18	2,05	2,05	4,92	1,85	**1,44**	1,65	2,02
100x5	0,19	**0,08**	0,22	0,13	0,17	0,18	1,89	0,15	0,10	0,12	0,17
100x10	0,89	0,71	1,11	1,06	1,13	1,16	3,73	0,88	**0,61**	0,79	1,14
100x20	1,13	1,28	1,46	1,75	1,64	1,64	3,90	1,27	**0,85**	1,18	1,63
200x5	0,07	**0,02**	0,08	0,04	0,06	0,06	1,08	0,05	0,03	0,04	0,06
200x10	0,42	**0,27**	0,57	0,44	0,56	0,58	2,48	0,41	**0,27**	0,39	0,56
200x20	0,71	0,76	1,13	1,22	1,23	1,24	3,00	0,80	**0,43**	0,69	1,24
500x5	0,02	**0,00**	0,02	0,01	0,01	0,02	0,49	0,01	0,01	0,01	0,01
500x10	0,13	**0,07**	0,19	0,11	0,17	0,17	1,23	0,12	0,08	0,12	0,17
500x20	0,34	0,29	0,66	0,58	0,72	0,73	2,03	0,39	**0,00**	0,35	0,72
Média	0,69	0,65	0,83	0,88	0,87	0,88	3,16	0,72	**0,53**	0,62	0,86

Pela análise da Tabela 2, verifica-se que há uma inconsistência nos resultados, porém é possível identificar que algumas heurísticas se comportam melhor diante uma baixa quantidade de máquinas, como é o exemplo da NEH KK2 e outras apresentam uma melhor performance com um número maior de máquinas, NEH FF. Em relação à variação da quantidade de tarefas, não houve uma superioridade de resultados por parte de alguma das heurísticas. Desta forma, verificou-se que a quantidade de máquinas influencia diretamente na performance das heurísticas, enquanto a variação da quantidade de tarefas não apresentou grande divergência nos resultados.

Para uma melhor comparação entre as heurísticas, foi feita uma análise através de um ranking gerado (Tabela 3), com classificação definida a partir da qualidade das respostas obtidas, onde é possível identificar a posição de cada heurística por base estudada. A classificação por posição tem um range de 11, estando na primeira posição a heurística que apresenta melhor qualidade de resposta e na décima primeira, a heurística com a pior qualidade. O intuito é identificar se há ou não uma superioridade de alguma das heurísticas.

A Figura 14 apresenta a variação da classificação das heurísticas entre as bases, a fim de facilitar o acompanhamento da performance das heurísticas no ranking elaborado.

Tabela 3 – Posição de cada heurística por base estudada (parte 1)

Heurísticas	NEH	NEH D	NEH KK2	ORD NEH D	ORD NEH KK2	ORD NEH NM	ORD NEH SPT	TIE NEH D	TIE NEH FF	TIE NEH KK2	TIE NEH LT
BASE 1	8	1	5	6	10	9	11	2	3	4	7
BASE 2	7	1	5	6	10	9	11	2	3	4	8
BASE 3	7	1	5	6	10	9	11	2	3	4	8
BASE 4	8	1	5	6	10	9	11	2	3	4	7
BASE 5	7	3	5	6	10	9	11	4	1	2	8
BASE 6	8	1	5	6	10	9	11	2	3	4	7
BASE 7	9	1	5	6	10	8	11	2	4	3	7
BASE 8	7	1	5	6	10	9	11	2	4	3	8
BASE 9	8	2	5	6	10	9	11	3	1	4	7
BASE 10	8	2	5	6	10	9	11	3	4	1	7
BASE 11	8	3	5	6	10	9	11	4	2	1	7
BASE 12	7	3	9	5	9	8	11	4	1	2	6
BASE 13	8	1	5	6	10	9	11	2	3	4	7
BASE 14	8	1	5	6	6	10	11	2	4	3	9
BASE 15	8	1	5	6	10	9	11	2	3	4	7
BASE 16	8	1	5	6	10	9	11	2	3	4	7
BASE 17	8	1	4	6	10	9	11	3	5	2	7
BASE 18	8	2	5	6	10	9	11	1	3	4	7
BASE 19	7	1	5	6	10	9	11	2	3	4	8
BASE 20	7	2	4	6	10	9	11	3	5	1	8
BASE 21	7	2	5	6	10	9	11	3	4	1	8
BASE 22	8	2	5	6	10	9	11	3	1	4	7
BASE 23	7	3	2	6	10	9	11	4	5	1	8
BASE 24	8	5	2	7	10	9	11	4	3	1	6
BASE 25	7	4	3	6	10	9	11	5	1	2	8
BASE 26	8	3	2	6	10	9	11	4	5	1	7
BASE 27	8	3	5	6	10	9	11	4	1	2	7
BASE 28	8	2	5	6	10	9	11	3	1	4	7
BASE 29	7	3	2	6	10	9	11	5	4	1	8
BASE 30	6	5	2	7	10	9	11	4	3	1	8
BASE 31	6	5	3	7	10	9	11	4	1	2	8
BASE 32	8	1	5	6	10	9	11	3	2	4	7
BASE 33	8	1	5	6	10	9	11	2	3	4	7
BASE 34	7	1	5	6	10	9	11	2	3	4	8
BASE 35	7	1	5	6	10	9	11	2	4	3	8
BASE 36	7	1	5	6	10	9	11	3	2	4	8
BASE 37	7	1	5	6	10	9	11	2	3	4	8
BASE 38	8	1	5	6	10	9	11	2	3	4	7
BASE 39	8	1	4	6	10	9	11	3	5	2	7
BASE 40	8	4	3	6	10	9	11	5	2	1	7
BASE 41	8	4	3	6	10	9	11	5	2	1	7
BASE 42	8	4	3	6	10	9	11	5	1	2	7
BASE 43	8	2	5	6	10	9	11	3	1	4	7
BASE 44	7	2	5	6	10	9	11	3	1	4	8
BASE 45	7	2	5	6	10	9	11	3	4	1	8
BASE 46	7	2	4	6	10	9	11	3	5	1	8
BASE 47	4	8	2	5	10	7	11	9	3	1	6
BASE 48	8	4	3	6	10	9	11	5	2	1	7
BASE 49	7	4	3	6	10	9	11	5	1	2	8
BASE 50	7	4	2	6	10	8	11	5	3	1	9
BASE 51	7	5	3	6	10	9	11	4	1	2	8
BASE 52	8	3	5	6	10	9	11	4	1	2	7
BASE 53	4	9	2	5	8	6	11	10	3	1	7

Tabela 3 – Continuação (parte 2)

Heurísticas	NEH	NEH D	NEH KK2	ORD NEH D	ORD NEH KK2	ORD NEH NM	ORD NEH SPT	TIE NEH D	TIE NEH FF	TIE NEH KK2	TIE NEH LT
BASE 54	4	7	2	5	10	9	11	8	3	1	6
BASE 55	6	5	3	8	10	9	11	4	1	2	7
BASE 56	7	2	5	6	10	9	11	3	1	4	8
BASE 57	7	2	5	6	10	9	11	3	1	4	8
BASE 58	7	3	5	6	10	9	11	4	1	2	8
BASE 59	8	3	2	6	10	9	11	4	5	1	7
BASE 60	7	2	5	6	10	9	11	3	1	4	8
BASE 61	7	1	5	6	10	9	11	2	4	3	8
BASE 62	8	1	5	6	10	9	11	2	4	3	7
BASE 63	8	3	5	6	10	9	11	4	1	2	7
BASE 64	7	5	3	6	10	9	11	4	1	2	8
BASE 65	7	4	3	6	10	9	11	5	2	1	8
BASE 66	7	3	4	6	10	9	11	5	1	2	8
BASE 67	6	4	3	8	10	9	11	5	1	2	7
BASE 68	7	5	2	8	10	9	11	4	3	1	6
BASE 69	7	3	2	6	10	9	11	4	5	1	8
BASE 70	7	3	5	6	10	9	11	4	1	2	8
BASE 71	8	1	5	6	10	9	11	2	4	3	7
BASE 72	8	1	5	6	10	9	11	2	4	3	7
BASE 73	7	1	4	5	10	9	11	2	8	3	6
BASE 74	8	4	3	7	10	9	11	5	2	1	6
BASE 75	5	7	2	4	10	9	11	8	3	1	6
BASE 76	5	8	2	4	10	7	11	9	3	1	6
BASE 77	7	4	3	6	10	9	11	5	2	1	8
BASE 78	6	4	3	7	10	9	11	5	1	2	8
BASE 79	8	3	4	6	10	9	11	5	1	2	7
BASE 80	6	4	3	7	10	9	11	5	1	2	8
BASE 81	6	4	2	8	10	9	11	5	3	1	7
BASE 82	8	4	2	7	10	9	11	3	5	1	6
BASE 83	6	4	3	7	10	9	11	5	2	1	8
BASE 84	6	3	5	7	10	9	11	4	2	1	8
BASE 85	8	2	5	6	10	9	11	3	4	1	7
BASE 86	8	3	5	6	10	9	11	4	1	2	7
BASE 87	7	1	5	6	10	9	11	2	3	4	8
BASE 88	8	1	5	6	10	9	11	2	3	4	7
BASE 89	8	4	3	7	10	9	11	5	1	2	6
BASE 90	5	7	2	6	10	9	11	8	3	1	4
BASE 91	6	9	2	5	8	7	11	10	3	1	4
BASE 92	8	4	3	6	10	9	11	5	1	2	7
BASE 93	8	4	3	6	10	9	11	5	2	1	7
BASE 94	8	4	3	6	10	9	11	5	2	1	7
BASE 95	8	5	3	7	10	9	11	4	1	2	6
BASE 96	6	4	3	7	10	9	11	5	2	1	8
BASE 97	5	8	2	4	10	7	11	9	3	1	6

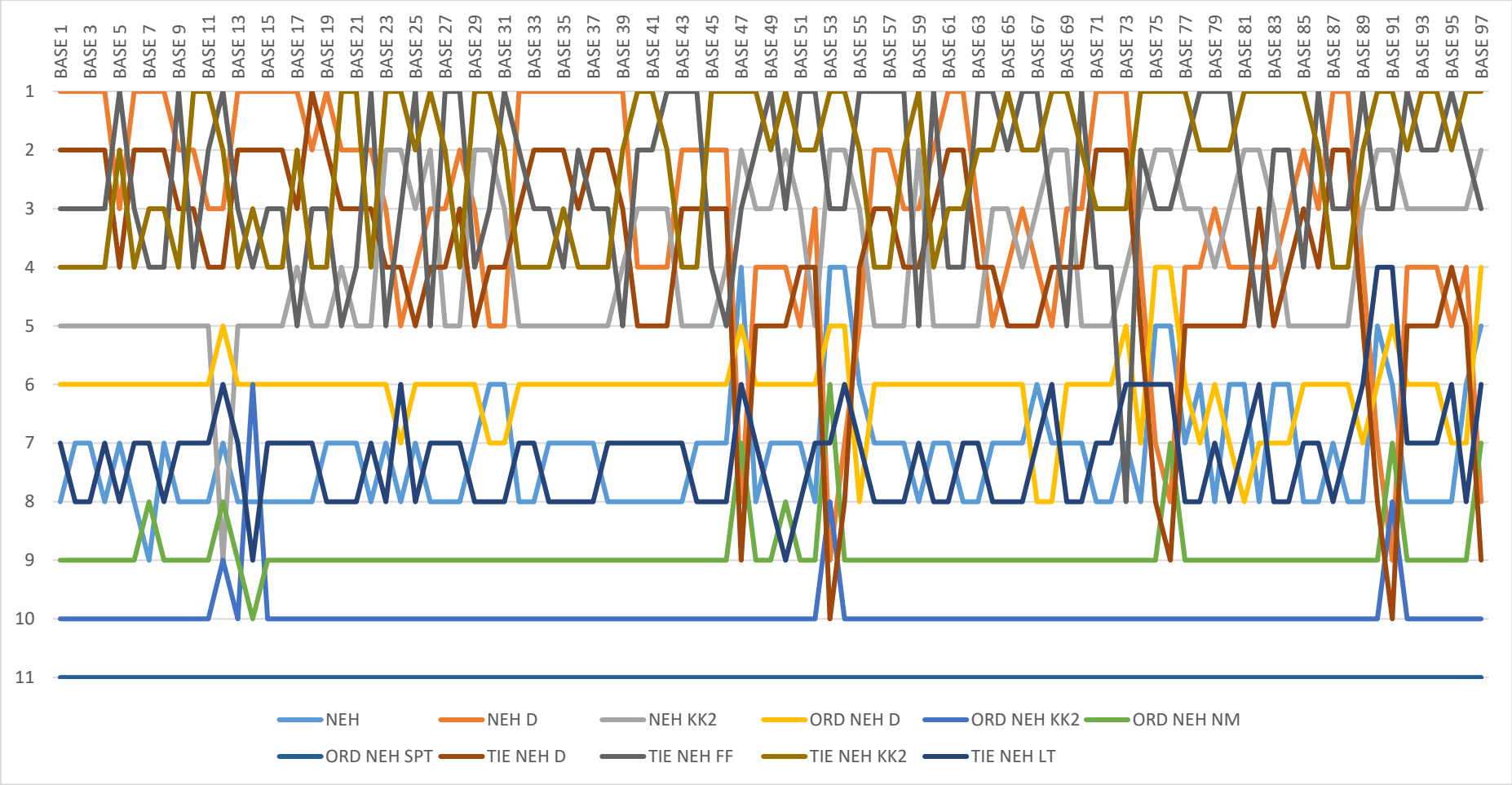

Figura 14 – Variação da classificação das heurísticas entre as bases. Fonte: Autor, 2017

Pela Figura 14, verifica-se que o ranking é alterado a todo momento no decorrer dos testes, apresentando uma inconsistência de resultados que pode ser explicada pela variação na distribuição dos tempos de processamento entre os problemas testes e pela variação no número de tarefas e máquinas.

Um dos destaques sobre esta figura é que 3 heurísticas ocupam a primeira posição de forma alternada durante toda a análise, são elas: NEH D, TIE NEH FF E TIE NEH KK2. A TIE NEH KK2 supera as concorrentes em 37 vezes, enquanto a TIE NEH FF fica na frente em 31 oportunidades e a NEH D obteve a melhor posição em 28 bases. A única vez que nenhuma destas 3 ficaram em primeiro lugar, foi na base 18, onde quem apresentou a melhor performance foi a TIE NEH D. Um segundo ponto é que enquanto há uma disputa pela primeira colocação, a última posição é preenchida pela mesma heurística em todas as disputas. A heurística ORD NEH SPT apresentou resultados inferiores em todas as bases estudadas. Além disso, pode-se inferir que Extensões de Ordenação não geram resultados tão bons quanto as heurísticas em seu modo original ou com Extensões de Mecanismos de Desempate, uma vez que as heurísticas da Classe 2 apresentaram resultados significamente inferiores às demais.

Foi feita uma análise, então, por classe de heurística e chegou-se às Figuras 15, 16 e 17: A Figura 15 apresenta a classificação das heurísticas da Classe 1 (Extensões de Ordenação e Mecanismos de Desempate), enquanto a Figura 16 apresenta a classificação das heurísticas da Classe 2 (Extensões de Ordenação) e a Figura 17 apresenta a classificação das heurísticas da Classe 3 (Extensões de Mecanismos de Desempate).

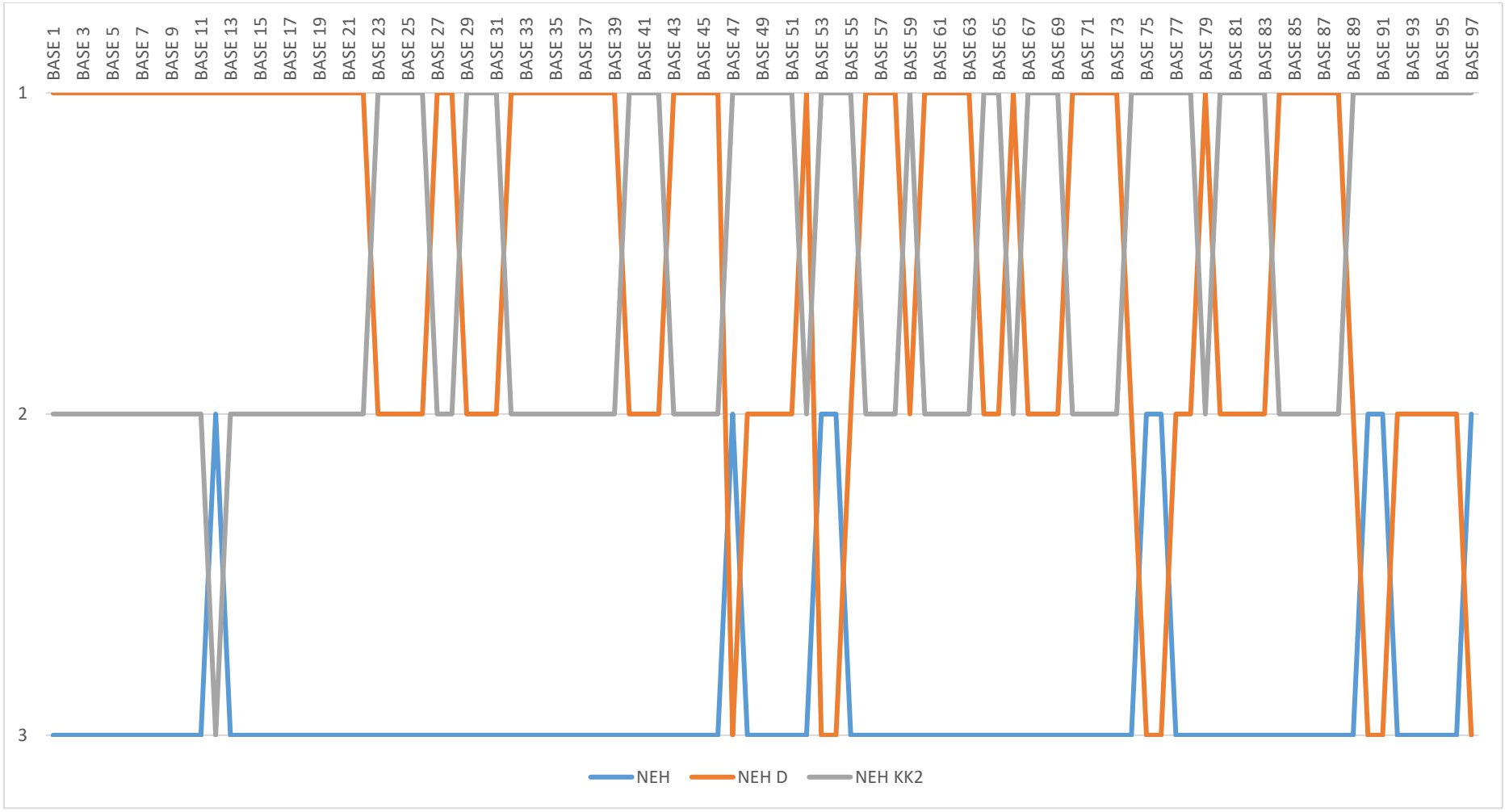

Figura 15 – Classificação das heurísticas da Classe 1 (Extensões de Ordenação e Mecanismos de Desempate). Fonte: Autor, 2017

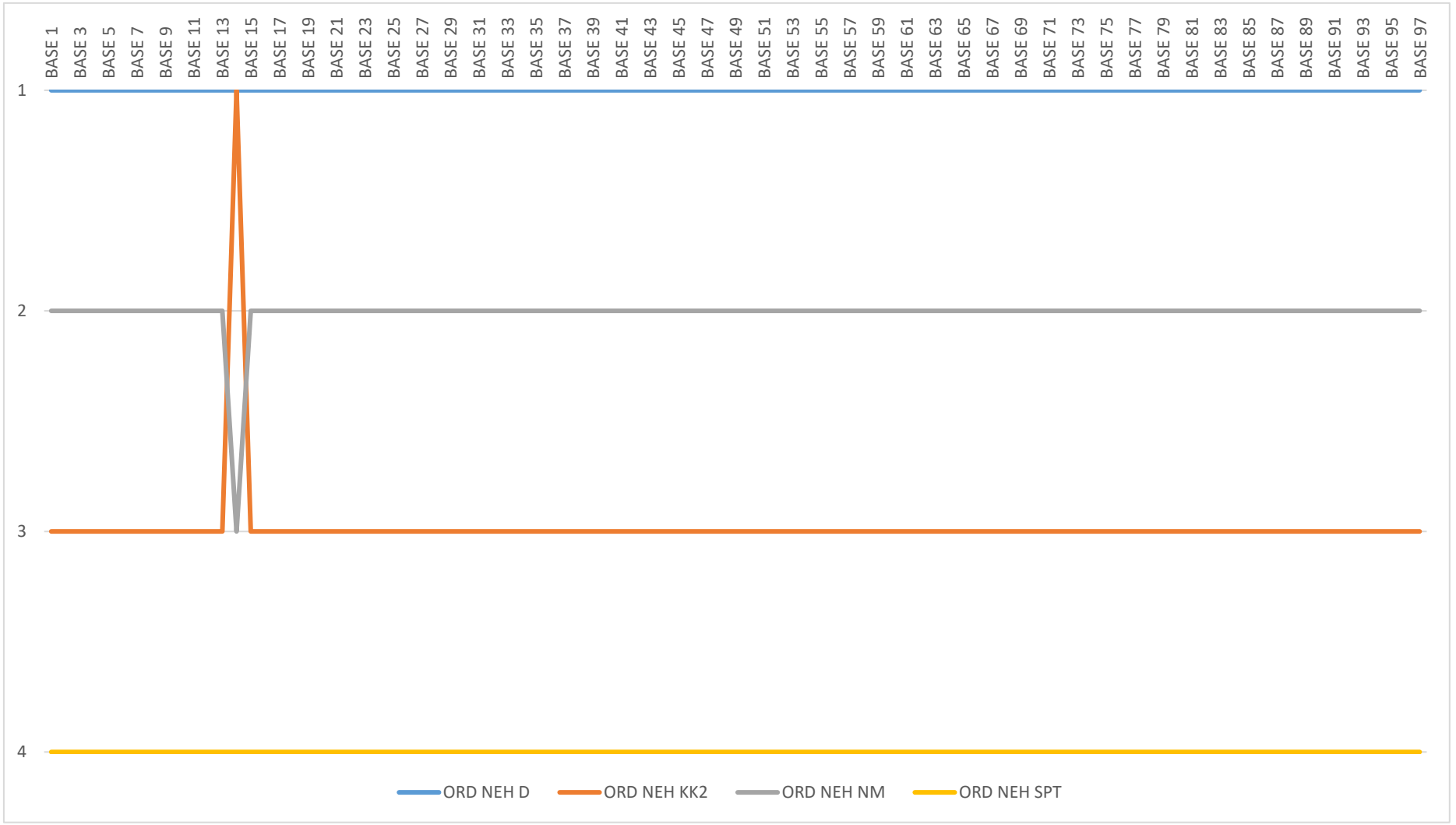

Figura 16 - Classificação das heurísticas da Classe 2 (Extensões de Ordenação). Fonte: Autor, 2017

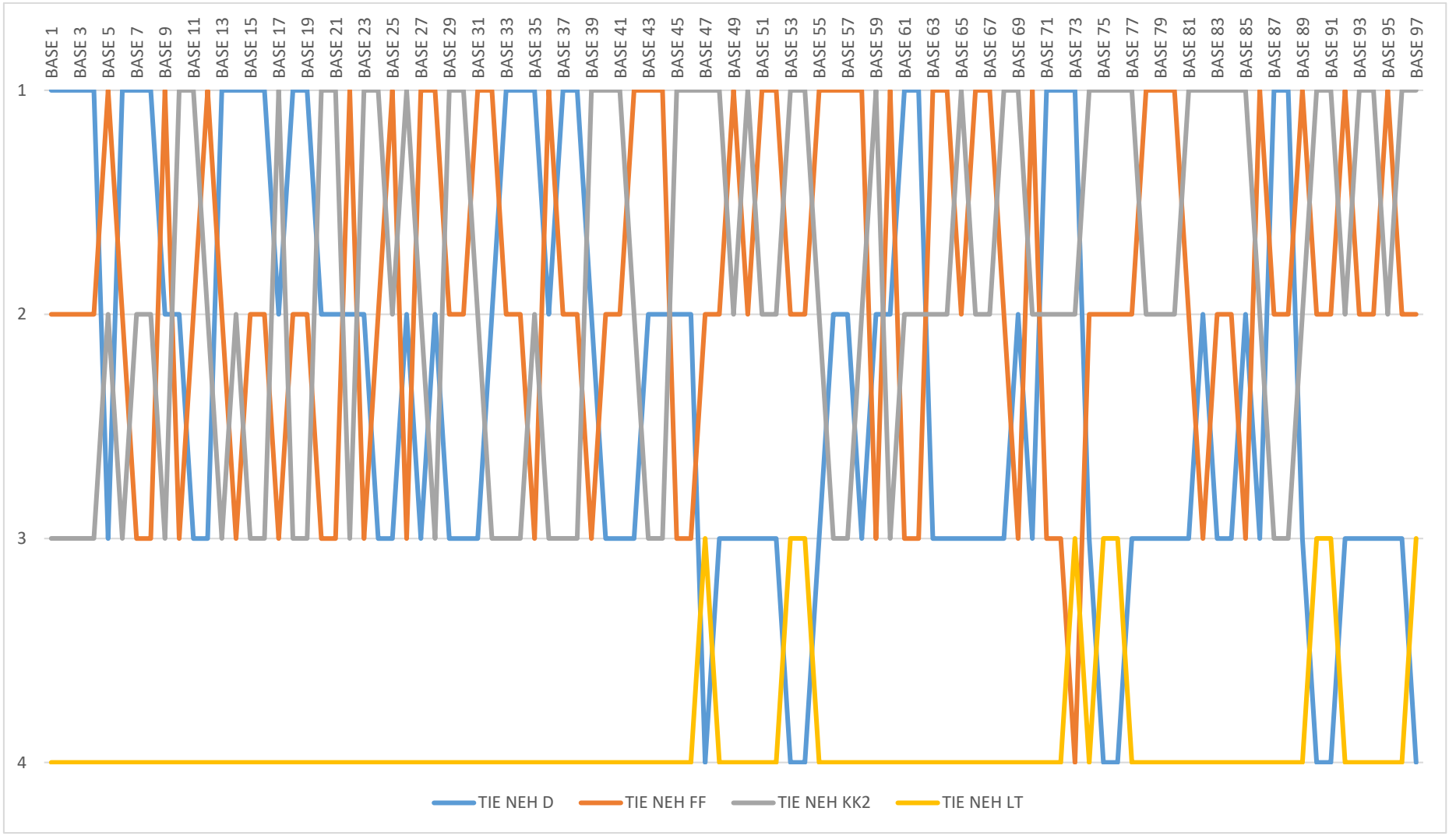

Figura 17 - Classificação das heurísticas da Classe 3 (Extensões de Mecanismos de Desempate). Fonte: Autor, 2017

Através da análise das Figuras 15, 16 e 17, foi possível encontrar as heurísticas mais bem classificadas por classe:

- **Classe 1:** A heurística NEH D apresentou performance superior às demais na maior parte dos casos. Da base 1 à base 22 sua superioridade ficou muito clara, mas a partir da base 23 apresentou certa inconsistência e alterou muito de posição com a NEH KK2. Mesmo sendo a melhor da classe, a NEH D chegou a estar em último em 8 ocasiões. Já a NEH não foi a melhor em nenhum dos casos, conseguindo apenas ficar em segundo lugar em 9 bases estudadas.
- **Classe 2:** As heurísticas apresentaram um comportamento estático, alterando de posição em apenas uma base (base 14). A heurística ORD NEH D se manteve à frente em todas as disputas, apresentando uma superioridade significativa. A ORD NEH NM só não esteve em segundo lugar na já citada base 14, onde trocou de lugar com a terceira colocada, ORD NEH KK2. A ORD NEH SPT, como já visto em outras análises, apresentou os piores resultados em todas as bases.
- **Classe 3:** Esta classe foi a que apresentou maior inconsistência de resultados, exceto pela TIE NEH LT que ficou em último lugar em 91 das 97 bases. As heurísticas TIE NEH KK2, TIE NEH FF e TIE NEH D alternaram muito de posição. Mesmo que a TIE NEH D alcançou o topo do ranking em algumas oportunidades, a disputa maior foi entre TIE NEH FF e TIE NEH KK2, em que quem acabou levando a melhor em quase 40% das vezes foi a TIE NEH KK2.

Foi feita uma comparação entre as melhores heurísticas de cada classe a fim de definir a melhor heurística para o problema de programação estudado. A comparação é apresentada na Figura 18.

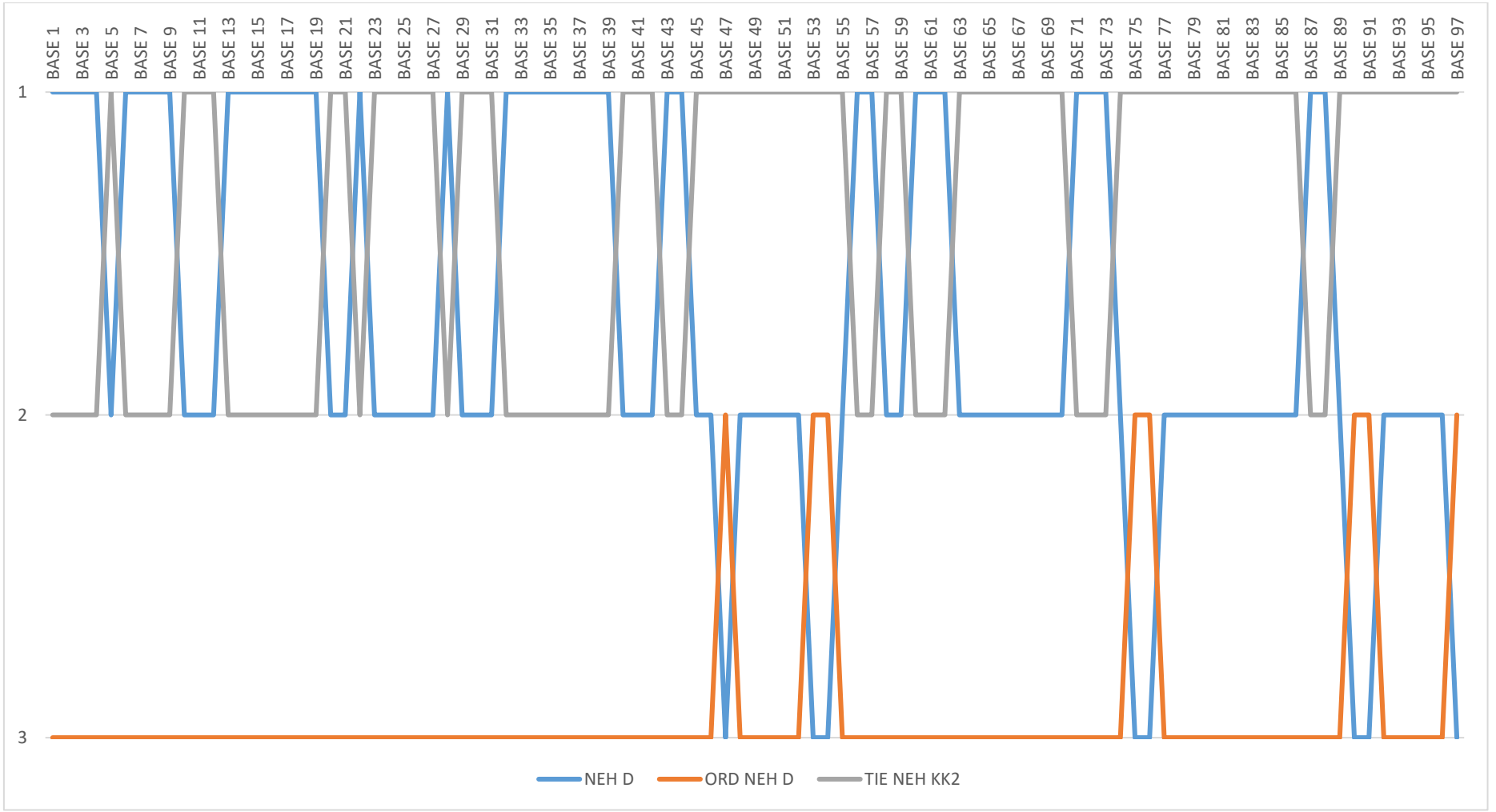

Figura 18 – Comparação entre as melhores heurísticas de cada classe. Fonte: Autor, 2017

Na Figura 18, o comportamento das heurísticas continua instável. A ORD NEH D é a que apresenta o pior desempenho, enquanto a TIE NEH KK2 detém os melhores resultados obtidos. Nesta análise entre classes, pode-se inferir que as heurísticas com Extensão de Mecanismos de Desempate apresentam uma melhor performance, seguida das heurísticas em sua forma original e por fim as com Extensão de Ordenação. Vale ressaltar que a NEH D apresentou movimento de queda nas mesmas bases em todas as análises realizadas (bases 47, 53, 54, 75, 76, 90, 91 e 97).

Conclui-se, então, que a variação da distribuição dos tempos de processamento afeta significamente os resultados obtidos pelas heurísticas, pois essa variação gera muita instabilidade na qualidade das respostas. Pelas análises realizadas, é possível identificar também, que a heurística com maior qualidade de resposta é a TIE NEH KK2, estando em primeiro lugar em 37 das 97 bases e a pior é a ORD NEH SPT, apresentando resultados significamente inferiores em 100% das bases.

4 CONSIDERAÇÕES FINAIS

A busca por soluções para problemas de sistemas de produção *flow shop* com critério de minimização do *makespan* é uma prática que pode contribuir com a redução de desperdícios e custos nas empresas (Slack, 2009).

As análises apresentadas neste trabalho contribuem para a seleção das melhores heurísticas para o problema de programação da produção em ambiente *flow shop* com critério de minimização do *makespan*. Escolhendo a heurística com a melhor performance, a organização pode atingir uma maior qualidade na solução do problema em um tempo satisfatório.

Os resultados obtidos com o estudo e análise de desempenho das heurísticas NEH, NEH D, NEH KK2, ORD NEH D, ORD NEH KK2, ORD NEH NM, ORD NEH SPT, TIE NEH D, TIE NEH FF, TIE NEH KK2 e TIE NEH LT mostraram que a TIE NEH KK2 possui uma melhor performance na qualidade das soluções, superando a TIE NEH FF e a NEH D, heurísticas que apresentaram ótimos resultados.

A constante variação do ranking de desempenho, que tem como fator comparativo a qualidade da resolução, mostra que a variação da distribuição e dos intervalos de valores do tempo de processamento das tarefas influenciam no desempenho das heurísticas.

Foi apresentado também, que a quantidade de tarefas não gerou impacto considerável sobre a qualidade da solução, porém a quantidade de máquinas tem forte influência sobre ela. As heurísticas corresponderam de formas diferentes à variação da quantidade de máquinas, mas não seguiram nenhum padrão de comportamento.

Por fim, concluiu-se que a alteração na ordenação e no método de desempate também influencia na performance das heurísticas. Esta constatação é apresentada nos gráficos comparativos de classes, uma vez que na sua forma original, a heurística NEH D apresentou melhores resultados que a NEH KK2, porém quando alterado o método de desempate, a heurística TIE NEH KK2 foi muito superior à TIE NEH D.

Os resultados obtidos neste trabalho possibilitam a criação de um novo conjunto de problemas testes. Isso pode ser realizado por meio da seleção dos problemas de mais difícil solução. Desta forma, poderia se gerar um conjunto de problemas mais robusto que pode contribuir na comparação de futuros métodos propostos na literatura, possibilitando

comparações mais extensivas do que as possibilitadas pelo conjunto de Taillard. Para a seleção dos problemas mais difíceis, seria necessário a implementação de um método mais intensivo computacionalmente (buscas locais, metaheurísticas, algoritmos gulosos iterativos) e que fornecesse soluções significativamente melhores que as atuais. Essas soluções encontradas seriam comparadas às melhores soluções obtidas até o momento, e os problemas que gerassem uma maior variação entre as soluções seriam selecionados.

5 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=sd01ountu02dq0r3f7thqbkd61&label=CNRS&lan ue=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 shop*s. 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 shop*s. **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, Enscore 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 shop*s. **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 shop*s. 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 shop*s. 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 shop*s 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 || $\sum C_j$ 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 shop*s. **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.

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.

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 shop*s 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 Resarch**, 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 shop*s. **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.