UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BERNARDO HENRIQUE OLBERTZ NETO

ANÁLISE DE MÉTODOS HEURÍSTICOS PARA MINIMIZAÇÃO DO TEMPO TOTAL DA PROGRAMAÇÃO DE OPERAÇÕES NO PROBLEMA *FLOW SHOP* PERMUTACIONAL

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2016

BERNARDO HENRIQUE OLBERTZ NETO

ANÁLISE DE MÉTODOS HEURÍSTICOS PARA MINIMIZAÇÃO DO TEMPO TOTAL DA PROGRAMAÇÃO DE OPERAÇÕES NO PROBLEMA *FLOW SHOP* PERMUTACIONAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Fabio Jose Ceron Branco

PONTA GROSSA

2016

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

TERMO DE APROVAÇÃO

ANÁLISE DE MÉTODOS HEURÍSTICOS PARA MINIMIZAÇÃO DO TEMPO TOTAL DA PROGRAMAÇÃO DE OPERAÇÕES NO PROBLEMA FLOW SHOP PERMUTACIONAL

por

BERNARDO HENRIQUE OLBERTZ NETO

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em sete de novembro de 2016 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Fabio José Ceron Branco
Orientador(a)

_____
Prof. Dr. Shih Yung Chin
Membro titular

_____
Prof(a). Dra. Simone de Almeida
Membro titular

_____
Prof. Dr. Augusto Foronda
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. Dr. Erikson Freitas de Morais
Coordenador do curso

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

DEDICATÓRIA

Dedico este trabalho aos meus pais, pelo apoio incondicional e confiança ao longo da graduação, tanto nos momentos de alegria quanto nos mais difíceis.

AGRADECIMENTOS

Aos meus pais, Nilvia e Marcelo, que batalharam diariamente pela minha educação, que são exemplos que sempre segui e seguirei na minha vida toda. Entendo a dificuldade de ser o filho mais velho, bem como o primeiro a deixar nosso lar, pois a distância foi uma grande dificuldade a ser superada.

A minha namorada, Elismari, que me apoiou e vivenciou as diversas dificuldades diárias e esteve sempre ao meu lado. Ela nunca desistiu de mim, apesar das minhas falhas e espero que viva ao meu lado para nossas futuras vitórias.

Aos diversos professores que me ensinaram não só os conteúdos, mas o modo de pensar e a astúcia de querer aprender mais. Devo a eles também a curiosidade e o gosto pela vida acadêmica.

Ao professor orientador Fábio Branco, pela atenção nos mais diversos horários, compreensão e dedicação ao nosso trabalho. Também pela oportunidade de se disponibilizar a ser orientador de um aluno de outro curso, que tinha a curiosidade e determinação de enfrentar um novo desafio.

RESUMO

OLBERTZ, Bernardo. **Análise de métodos heurísticos para minimização do tempo total da programação de operações no problema *flow shop* permutacional**. 2016. 46 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2016.

Com a crescente competitividade das empresas, o tempo destinado ao processo produtivo torna um elemento precioso e, assim, todo recurso deve ser aproveitado da melhor maneira possível. Neste contexto, este trabalho tem como objetivo a análise de soluções de problemas *flow shop* permutacional de modo a avaliar métodos heurísticos para a diminuição do tempo total da programação, conhecido na literatura por *makespan*, a partir da combinação de de sequências de tarefas. Os métodos usados são avaliados e posteriormente analisados, por meio de experimentações computacionais, como a porcentagem de sucesso, desvio médio relativo e tempo de computação.

Palavras-chave: *Flow Shop* Permutacional. *Makespan*. Programação da Produção. Métodos Heurísticos.

ABSTRACT

OLBERTZ, Bernardo. **Heuristic analysis methods for minimizing the total time of operations scheduling problem in permutation flow shop**. 2016. 46 p. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Federal Technology University Paraná. Ponta Grossa, 2016.

With the increasing competitiveness of enterprises, the time devoted to the production process becomes a precious element and thus every resource should be utilized in the best possible way. In this context, this study aims to analyze permutation flow shop problem solutions in order to evaluate heuristic methods to reduce the total time of the program, known in the literature by makespan from the combination of task sequences. The methods used are evaluated and subsequently analyzed through computational experiments, as the percentage of success and relative mean deviation computation time.

Keywords: Permutation Flow Shop. Makespan. Production Scheduling. Heuristic Methods.

LISTA DE ILUSTRAÇÕES

Figura 1 - Gráfico de Gantt. 18
Figura 2 - Relações entre as classificações dos problemas de produção (Adaptação apresentada por Moccellin e Nagano, 2003, a partir da figura apresentada por MacCarthy e Liu, 1993) 19
Figura 3- Gráfico de Gantt para cálculo do Makespan. 21
Figura 4 - Fluxo de processamento de um *flow shop* generalizado. 24
Figura 5 - Matriz de transferência de um *flow shop* com 5 máquinas. 24
Figura 6 - Representação *flow shop* em Gantt. 26
Figura 7 - *Makespan* parcial 1. 33
Figura 8 - *Makespan* parcial 2. 34
Figura 9 - *Makespan* parcial 3. 34

LISTA DE TABELAS

Tabela 1 - Tempos de processamento de quatro tarefas em quatro máquinas. 29
Tabela 2 - Tempos de processamento para quatro máquinas e quatro tarefas. 30
Tabela 3 - Representação de dez tarefas em cinco máquinas.....31
Tabela 4 - Representação de dez tarefas em cinco máquinas com heurística de Palmer aplicada.....32
Tabela 5 - Composição de cinco máquinas e quatro tarefas. 33
Tabela 6 - Taxa de Sucesso entre heurísticas 38
Tabela 7 - Classe 25 tarefas desvio relativo médio.....38
Tabela 8 -Classe 50 tarefas desvio relativo médio.....39
Tabela 9 - Classe 75 tarefas desvio relativo médio.....39
Tabela 10 - Classe 100 tarefas desvio relativo médio.....39
Tabela 11 - Diferença de *clocks.* 40
Tabela 12 - Comparações heurísticas NEH. 41
Tabela 13 - Porcentagem de sucesso entre heurísticas combinadas. 42

LISTA DE CÓDIGOS

Código 1 - Desempenho em tempo pela função *clock*. 37

LISTA DE EQUAÇÕES

Equação 1 - Coeficiente de Palmer.....32
Equação 2 - Desvio Médio Relativo.....37

LISTA DE ABREVIAÇÕES

DRM- Desvio Relativo Médio
LPT- *Longest Proccessing Time*
LPTNEH - *Longest Proccessing Time* com Nawaz, Enscore, Ham
NEH – Nawaz, Enscore, Ham
NP - *Non-Deterministic Polynomial time*
N&M – Nagano e Moccellin
PS- Porcentagem de Sucesso
SPT- *Shortest Processing Time*
SPTNEH - *Shortest Processing Time* com Nawaz, Enscore, Ham
TMC- Tempo Médio de Computação

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS 13
1.1.2 Objetivos Específicos 14
1.2 JUSTIFICATIVA 14
2 TRABALHOS RELACIONADOS 15
3 PROGRAMAÇÃO DA PRODUÇÃO 17
3.1 CLASSIFICAÇÃO DOS PROBLEMAS DE PRODUÇÃO 18
3.1.1 Função Objetivo *Makespan* 20
3.2 *SCHEDULING* 21
3.3 *FLOW SHOP* 23
3.3.1 *Flow Shop* Permutacional 27
4 HEURÍSTICAS 28
4.1 HEURISTICA *SHORTEST PROCESSING TIME* (SPT) 29
4.2 HEURÍSTICA *LONGEST PROCESSING TIME* (LPT) 30
4.3 HEURÍSTICA DE PALMER 30
4.4 HEURÍSTICA NAWAZ, ENSCORE, HAM (NEH) 32
5 METODOLOGIA 35
6 DESENVOLVIMENTO 36
6.1 ANÁLISE DOS RESULTADOS 38
7 CONCLUSÕES 43
REFERÊNCIAS 44

1. INTRODUÇÃO

O sistema de produção em série, também conhecido como linha de produção, é extremamente utilizado atualmente. Esse sistema de linha de produção necessita de uma programação de tarefas que compõem cada produto a ser produzido e, cada máquina ou etapa realiza parte do todo em um produto final.

Um problema *flow shop* existe quando todas as tarefas partilham da mesma ordem de processamento em todas as máquinas como, por exemplo, em uma gráfica, em que os rolos de papel necessitam seguir uma ordem para que, após passar por todas as máquinas, exista um cartaz pronto.

O problema da alocação dos recursos para o processamento, em que se decide a ordem de processamento associada a alguma medida de desempenho (ou a uma ponderação de medidas), como o tempo total de produção (conhecido como *makespan*), é a base do *flow shop*.

Um *flow shop* é dito permutacional quando além da sequência de máquinas permanece a mesma, a sequência de tarefas também segue o mesmo padrão.

Segundo Baraz e Mosheiov (2008), existem diversas situações para o problema de *flow shop* em várias indústrias, principalmente no caso de a preparação da produção ou custo de funcionamento das máquinas são altos. Exemplos são indústria de borracha, fornos de fundições, metalurgia e grandes máquinas de corte.

1.1. OBJETIVOS

A seguir serão descritos os objetivos geral e específicos a serem atingidos por esse trabalho.

1.1.1. Objetivo Geral

Minimização do *makespan a*nalisando o desempenho de métodos heurísticos para o problema *flow shop* permutacional.

1.1.2. Objetivos Específicos

- Aplicar a heurística *Shortest Processing Time* (SPT) para o problema *flow shop* permutacional com o objetivo de minimizar o *makespan*.
- Aplicar a heurística *Longest Processing Time* (LPT) para o problema *flow shop* permutacional com o objetivo de minimizar o *makespan*.
- Aplicar a heurística de Palmer no problema *flow shop* permutacional com o objetivo de minimizar o *makespan*.
- Adaptar parte da heurística de Nawaz, Enscore, Ham (1983), conhecida pelas iniciais dos autores como NEH, de modo a comparar a qualidade da solução e o custo computacional.
- Combinar partes das heurísticas mencionadas de modo a encontrar um novo método que se mostre eficiente, quanto a qualidade e facilidade de implementação.

1.2. JUSTIFICATIVA

O problema de programação *flow shop* é recorrente, pois ele se associa ao método de produção mais utilizado atualmente. Existem diversos estudos sobre o problema, entretanto o mesmo continua em aberto por não haver um método de solução ótima para situações de alta complexidade. As soluções encontradas são de alta qualidade para a função-objetivo especificada e amenizam os custos envolvidos.

2. TRABALHOS RELACIONADOS

Este capítulo tem por objetivo apresentar estudos e trabalhos que foram desenvolvidos para o problema de programação *flow shop*, bem como suas particularidades e derivações como o caso do *flow shop* permutacional.

Garey e Johnson (1979) realizaram um estudo em diversos problemas computacionais e evidenciaram que a maioria das extensões de problemas *flow shop* são NP_difícil, o que significa que este problema possui alta complexidade de solução.

Chakraborty (2009) reuniu diversos trabalhos relacionados em seu livro, com técnicas baseadas em inteligência computacional, bem como a importância dos problemas de *flow shop* ao longo de mais de cinquenta anos.

Pan & Ruiz (2013) pesquisaram a avaliação computacional, complexidade e eficiência de diversos algoritmos, tanto de heurísticas simples quanto de heurísticas compostas na solução de problemas *flow shop* permutacional. Este trabalho teve como objetivo uma revisão compreensiva e atualizada da literatura, como também a comparação direta entre algoritmos clássicos e novas versões dos mesmos. O trabalho se estende, juntamente com estes algoritmos, a quatro novas derivações elaboradas pelos autores com resultados significativos.

Nawaz, Enscore e Ham (1983) possuem o trabalho de muita relevância na área de *flow shop* permutacional e, pode-se citar, que são comumente são citados e utilizados no desenvolvimento de outras heurísticas. O algoritmo criado até os dias atuais possui alta eficiência, já que apresenta alta qualidade de solução e rapidez computacional. Muitos trabalhos posteriores surgiram baseados neste algoritmo de modo a otimizá-lo com novas técnicas, principalmente baseadas em novas tecnologias e abordagens.

Diversos livros também abordam os assuntos de *scheduling* e *flow shop*, dentre os quais se destacam Pinedo (2008), que aborda *scheduling* na sua teoria, algoritmos e os sistemas. Emmons e Vairaktarakis (2013) discorrem sobre as convenções de classificação e notação bem como as diversas peculiaridades que envolvem o problema de *flow shop*, como o número de máquinas envolvidas, a multiplicidade de máquinas, *flow shops* híbridos, *flow shops* sem espera entre outros.

Todos estes trabalhos citados foram importantes para diversos estudos que envolvem também comparações e flexibilidade para o complexo problema que é o *flow shop*, a quantidade e qualidade dos trabalhos reflete a importância deste tema. *Flow shop* e suas vertentes são objetos de estudo da área de computação, todavia com a otimização destes é possível melhorar sistemas de produção, objeto de estudo da área de Engenharia de Produção, sendo então o tema multidisciplinar.

3. PROGRAMAÇÃO DA PRODUÇÃO

O conceito de produção no âmbito de economia pode ser definido como um processo, ou grupo de processos, o que resulta em produtos ou serviços. Um sistema de produção pode ser definido como a transformação de insumos (mão-de-obra, matérias-primas, capital, materiais, entre outros) em produtos e serviços, por meio de processos e operações (MONKS, 1987).

O sistema de produção terá sua eficiência essencialmente conectada a diversas decisões, contendo nestas a programação da produção. A programação de produção, segundo Baker (1974), pode ser definida resumidamente como a alocação de recursos por meio do tempo.

Com estes conceitos, temos como essência, como destaca Baker (1974), duas ações primordiais: i) as decisões quanto a alocação dos recursos disponíveis; ii) as decisões no sequenciamento das tarefas.

De modo geral, conforme Taillard (1993), um problema é um conjunto de *n* tarefas {J1, J2, ..., J*n*} que devem ser processadas em um conjunto *m* máquinas {M1, M2, ..., M*m*} resultando em tempos de execução por tarefa em cada máquina.

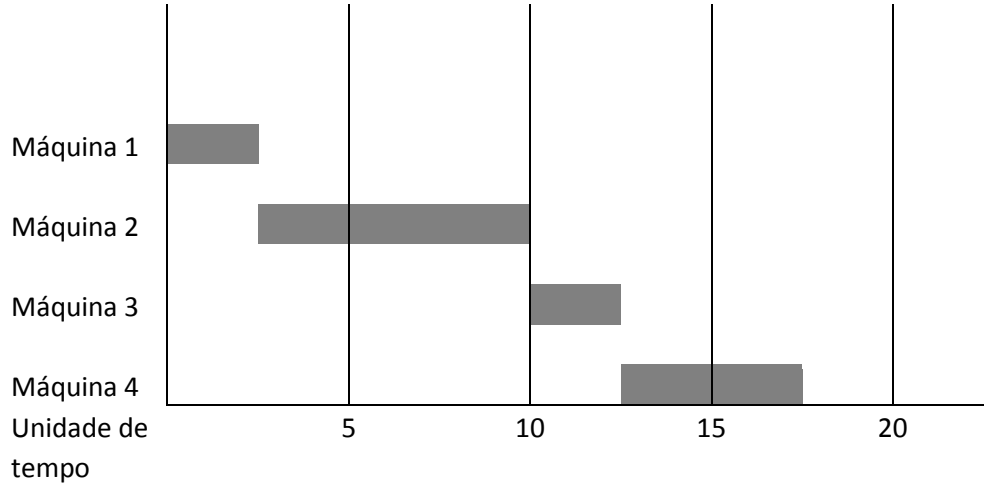
Segundo Slack (1999), a atividade de programação é uma das mais complexas tarefas no gerenciamento de produção. Os programadores necessitam lidar com todos os diferentes tipos de recursos. Conforme o número de tarefas aumenta, crescem também o número de maneiras diferentes destas tarefas serem sequenciadas. Deste modo se existem *n* tarefas, existem *n*! (*n* fatorial) maneiras diferentes de organizá-las. Observando-se a presença de mais de uma máquina (*m>1*), o número de sequências se eleva para $(n!)^m$.

Um modo de exemplificar este modelo geral é pela representação por meio de um gráfico de Gantt, desenvolvido por H. L. Grantt em 1917, em que o tempo é representado por barras. Os momentos de início e fim das tarefas podem ser indicados no gráfico. As vantagens dos gráficos de Gantt é que eles proporcionam uma representação visual simples do que deve ocorrer em cada operação (SLACK *et al.,* 1999).

A Figura 1 representa um exemplo simples de um Gráfico de Gantt para o caso onde existe uma tarefa que é processada por quatro máquinas diferentes, cada

uma com o seu tempo individual de execução, totalizando um tempo total de dezessete unidades e meia de tempo.

Figura 1 - Gráfico de Gantt



Fonte: Autoria própria

Para este trabalho, toma-se que estas tarefas disponíveis são variáveis, entretanto o sequenciamento das tarefas nas máquinas será sempre o mesmo.

3.1. CLASSIFICAÇÃO DOS PROBLEMAS DE PRODUÇÃO

Um problema de programação é especificado, segundo MacCarthy e Liu (1993), em termos das restrições tecnológicas do ambiente de produção em que as tarefas devem ser realizadas e dos objetivos da programação.

De acordo com Pinedo (2008) todos os problemas de programação devem possuir o número de tarefas e máquinas em quantidades finitas. Desta forma, os problemas de programação podem ser classificados conforme MacCarthy e Liu (1993):

I) *job shop*: cada tarefa tem sua própria ordem no processamento nas máquinas;

II) *flow shop*: todas as tarefas possuem o mesmo fluxo de processamento nas máquinas;

III) *open shop*: não existe um fluxo padrão especificado para nenhuma tarefa e cada estágio da produção possui apenas uma máquina;

IV) *flow shop* permutacional: um *flow shop* em que a ordem de processamento das tarefas em todas as máquinas permanece igual;

V) máquina única: existe apenas um estágio de produção com uma única máquina disponível;

VI) máquinas paralelas: existe mais de uma máquina disponível em um único estágio de produção, no qual cada tarefa necessita de apenas uma dessas máquinas.

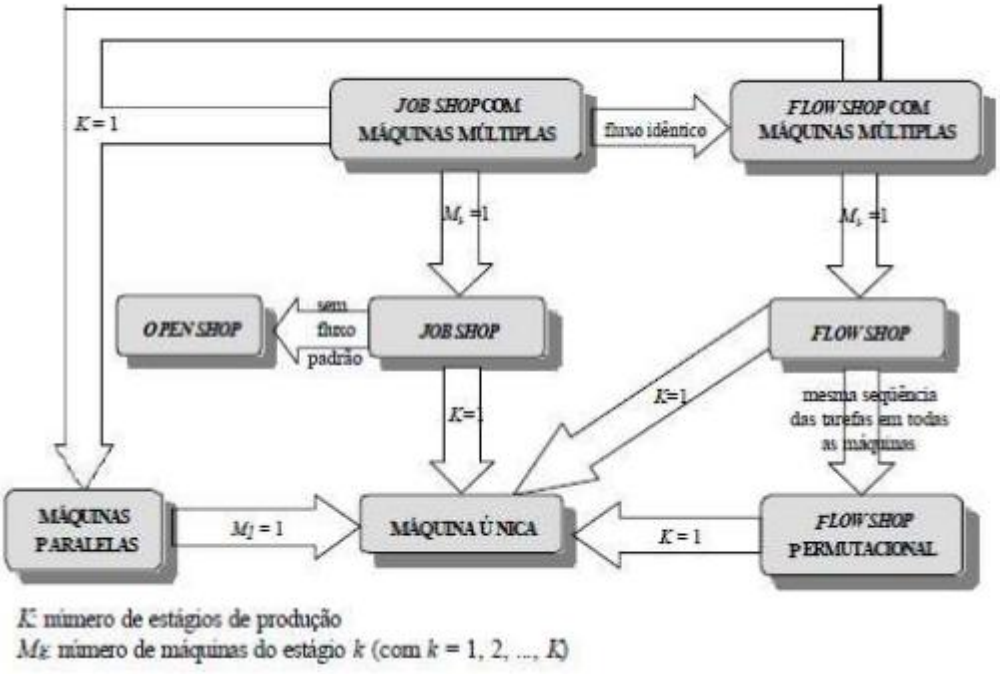
Conforme exemplificam Moccellin e Nagano (2003), existem mais duas classificações:

VII) *job shop* com múltiplas máquinas: *job shop* no qual existe um conjunto de máquinas paralelas em cada estágio;

VIII) *flow shop* com múltiplas máquinas: *flow shop* no qual existe um conjunto de máquinas em cada estágio de produção.

A Figura 2 apresenta a relação entre as classificações dos problemas de produção:

Figura 2 - Relações entre as classificações dos problemas de produção (Adaptação apresentada por Moccellin e Nagano, 2003, a partir da figura apresentada por MacCarthy e Liu, 1993)



Fonte: Moccellin e Nagano (2003)

Além dos diferentes tipos de problemas, existem diferentes tipos de funções objetivo que podem ser abordadas nos estudos, cada uma com sua peculiaridade, entretanto com a característica de comumente possuírem o tempo de execução das

tarefas como medida de otimização. São exemplos de função objetivo segundo Pinedo (2008):

- *Makespan*: tempo total gasto até a conclusão da última tarefa finalizada, quando minimizado a função demonstra boa utilização de máquinas;
- *Flowtime*: tempo total de fluxo das tarefas, quando minimizado demonstra redução das matérias-primas em processamento;
- *Tardiness*: atraso das tarefas, representa o tempo excedido em relação a entrega;
- *Earliness*: adiantamento do término das tarefas em relação à entrega.

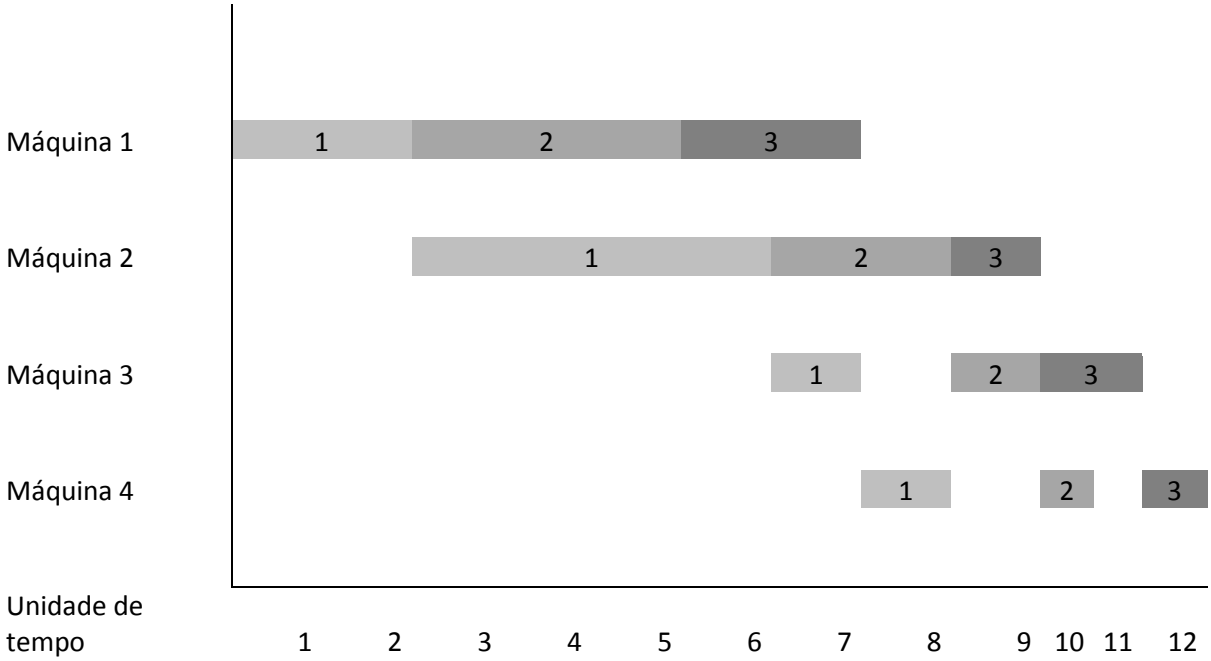
Considerando estes exemplos, o presente trabalho adotou pela utilização do *makespan*. A escolha se deve por esta ser comumente usada na literatura e o conceito possuir maior interação com a utilização direta em programações *flow shop*.

3.1.1. Função Objetivo *Makespan*

Sendo a função objetivo o *makespan*, o objetivo é a minimização da duração completa da programação, diminuindo o tempo total temos o melhor uso dos recursos disponíveis e melhor resposta quanto a entrega das tarefas.

A Figura 3 representa um ambiente de sequenciamento das tarefas n = 3 juntamente com o grupo m = 4 quanto as máquinas. Este ambiente de sequenciamento é o *flow shop*. A tarefa n = 1 é iniciada no instante de 0 unidades de tempo na máquina 1 e acaba de ser processada após 2 unidades de tempo; sendo processada em sequência pela máquina 2 em 4 unidades de tempo continuando por 1 unidade de tempo na máquina 3 e 1 unidade de tempo na máquina 4. A tarefa n = 2 então é iniciada na máquina 1 no instante de 2 unidades de tempo, processada em 3 unidades de tempo prosseguindo para a máquina 2 no instante de 6 unidades de tempo sendo processada em 2 unidades e iniciando em 8 unidades de tempo na máquina 3 aonde fica por 1 unidade de tempo finalizando na máquina 4 por mais 1 unidade de tempo. A tarefa *n* = 3 inicia em 5 unidades de tempo na máquina 1, é processada por 2 unidades nesta, inicia em 8 unidades na máquina 2 onde é processada em 1 unidade de tempo, inicia em 9 unidades de tempo na máquina 3, é processada em 2 unidades de tempo e, por fim, inicia em 11 unidades de tempo e termina toda a sequência do *flow shop* em 12 unidades de tempo.

Figura 3- Gráfico de Gantt para cálculo do Makespan



Fonte: Autoria própria

O tempo total de processamento da sequência apresentada é de doze unidades de tempo, este tempo total é o *makespan*. Com a redução do *makespan* obtém-se um menor tempo de máquinas em execução, o que possibilita um melhor aproveitamento do ciclo.

3.2. *SCHEDULING*

Scheduling é um processo de tomada de decisão que é usado nas mais diversas áreas da indústria, tanto nas partes de manufatura de produtos quanto em serviços prestados.

Segundo Santoro (1982), há uma nomenclatura de termos e conceitos geralmente usada em diversos trabalhos, se mostrando atual por convenção e clareza até os dias atuais, havendo apenas a exceção de sequência de prioridade.

- Tarefa: que também pode ser designada por ordem, produto, trabalho (*job*), corrida, entre outros. Pode ser um produto ou um lote de produtos idênticos, que devem ser processados pelos recursos produtivos.
- Máquina: que também pode ser posto, processador, equipamento entre outros, é o recurso produtivo destinado a executar uma operação. Outro recurso produtivo é o recurso humano que pode ser designado por homem.

- Operação ou processamento: é o trabalho realizado pelo recurso produtivo sobre a tarefa.
- Tempo de processamento: é o tempo necessário para se concluir uma operação sobre uma tarefa.
- Roteiro: é a sequência ordenada de recursos produtivos pelos quais devem passar as tarefas.
- Sequência de prioridade: é a disposição das tarefas por ordem de atendimento, ou seja, cada tarefa terá uma prioridade de atendimento que será ditada pela posição da tarefa na sequência. Deste modo, nas filas formadas nos recursos produtivos, serão atendidas primeiramente as tarefas com melhor prioridade de atendimento, ou seja, as tarefas com posições à frente das outras tarefas que estão posicionadas na fila do recurso produtivo.
- Programa: é uma sequência viável das tarefas nos recursos produtivos.
- Tempo ocioso: é o intervalo de tempo em que uma máquina pode ficar parada esperando, estando disponível e tendo uma tarefa para atender, ela espera para atender outra tarefa.

No presente trabalho os conceitos de maior uso se resumem a tarefa, máquina, processamento e tempo de processamento, este último analisado para a minimização do *makespan* como objetivo.

Segundo Baker (1974), *scheduling* é um plano tangível, tal como um plano de horários de ônibus ou de aulas. Usualmente uma programação informa quando as coisas devem acontecer e mostra o plano de certas atividades respondendo a questão: “quando alguma coisa terá lugar?”. A resposta para essa questão usualmente nos informa o horário (data), como por exemplo: uma viagem de ônibus começa às 8:00 horas e termina as 12:00 horas. Entretanto, uma outra resposta válida poderia ser em termos de sequência e não horário (data).

Pinedo (2008) conceitua que o objetivo geral do *scheduling* é organizar os recursos da forma mais eficiente possível para que se possa otimizar um processo dado, seja ele qualquer.

Os recursos a serem organizados tendem a ser variados, dependendo sempre das características do ambiente. Os recursos podem ser parafusos em uma construção, unidades de processamento em um processador de computador,

maquinário na manufatura de um motor ou em uma indústria de tintas dentre outros variados.

Seguindo o exemplo da manufatura de um motor, procurando-se minimizar os custos da produção de motores, elabora-se uma sequência de trabalho para as máquinas, em que não exista paradas entre um processo e outro. O trabalho contínuo faz com que o tempo de produção seja constante ou próximo de constante. Não havendo perdas de tempo com a organização das tarefas. Tem-se a minimização do tempo como objetivo e a organização dos recursos (a sucessão correta das máquinas) é um *scheduling*.

A programação envolvida mostra-se de alta complexidade aonde o *scheduling* procura a otimização de um objetivo. Sabe-se que a otimização visa o melhor resultado, a solução definitiva, entretanto o problema pertence à classe dos *NP*-completos. Não existe solução polinomial em uma máquina determinística para esse tipo de problema.

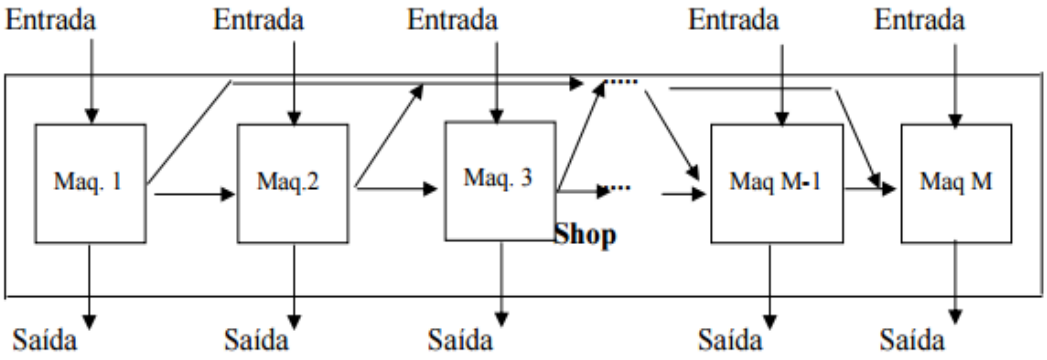
Um dos problemas da classe de *scheduling* em que a locação de recursos é a parte central é conhecido pelo nome de *FLOW SHOP*, o nome vem de *flow* (fluxo) e *shop* (logística, negócio). Assim o modo de “fluxo logístico” é de alta complexidade, como todo escalonamento, que aumenta conforme o número de elementos envolvidos. O próximo item apresenta outros detalhes *problema flow shop.*

3.3. *FLOW SHOP*

Dentro do âmbito de *scheduling*, o objeto de estudo é o *flow shop*. Em um *flow shop*, segundo Baker (1974), cada tarefa tem sua própria sequência de processamento com fluxo linear unidirecional, ou seja, sem retorno no fluxo. Então no ambiente *flow shop*, *n* tarefas devem ser programadas para processamento em um conjunto de *m* máquinas distintas, tendo o mesmo fluxo de processamento, o fluxo unidirecional. Quando, em todas as máquinas, a ordem de processamento das tarefas é a mesma, tem-se o *flow shop permutacional*.

A Figura 4 mostra o esquema genérico de um *flow shop*, conforme Baker (1974), em que as tarefas podem entrar por qualquer máquina, pular máquinas, mas sempre seguindo um fluxo unidirecional.

Figura 4 - Fluxo de processamento de um *flow shop* generalizado



Fonte: BAKER (1974)

Outro modo de exemplificar um *flow shop* é por meio de uma matriz de transferência, como visto em Hax e Candea (1984), aonde só pode existir entrada na linha *i* e coluna *j*, se *j*≥*i*. Estas entradas demonstram a fração de processamento transferida da máquina *i* para máquina *j* como é mostrado na Figura 5. A primeira linha mostra que a tarefa pode entrar na estação de processamento pelas máquinas 1, 2 e 4. A última coluna indica que as tarefas podem sair da estação de processamento pelas máquinas 2, 3, 4 e 5.

Figura 5 - Matriz de transferência de um *flow shop* com 5 máquinas

		PARA					
		Maq1	Maq2	Maq3	Maq4	Maq5	Saída
DE	Entrada	0.7	0.2		0.1		
	Maq1		0.5	0.3	0.1	0.1	
	Maq2			0.7	0.2		0.1
	Maq3				0.6	0.2	0.2
	Maq4					0.6	0.4
	Maq5						1.0

Fonte: Hax e Candea (1984)

O problema de programação *flow shop scheduling* foi provado por Garey e Johnson em 1979.

> *INSTANCE: Number $m \in Z^+$ of processors, set J of Jobs, each job $j \in J$ consisting of n tasks $t_1[j], t_2[j], ..., t_m[j]$, a length $l(t) \in Z^+$ for each such task t, and overall deadline $D \in Z^+$.*
> *QUESTION: Is there a flow-shop schedule for J that meets the overall deadline, where such a schedule is identical to an open-shop schedule with the additional constraint that, for each $j \in J$ and $1 \leq i < m$, $\delta_{i+1}(j) \geq \delta_i(j) + l(t_i[j])$?*
> *Reference: [Garey, Johnson, and Sethi, 1976]. Transformation from 3-PARTITION.*
> *Comment: NP-complete in the strong sense for m=3. Solvable in polynomial time for m=2 [Johnson, 1954]. The same results hold if "preemptive" schedules are allowed [Gonzalez and Sahni, 1978a], although if release times are added in this case the problem is NP-complete in the strong sense, even for m=2 [Cho and Sahni, 1978]. If the goal is to meet a bound K on the sum, over all $j \in J$, of $\delta_m(j) + l(t_m[j])$, the the non-preemptive problem is NP-complete in the strong sense even if m=2 [Garey, Johnson, and Sethi, 1976].*

Esta é a definição do problema, aonde existe um número inteiro positivo de processadores e um conjunto de sequências *J* que é composto por *n* tarefas, cada tarefa possui um tempo de processamento *l(t)* que é inteiro positivo, existindo um total inteiro positivo D.

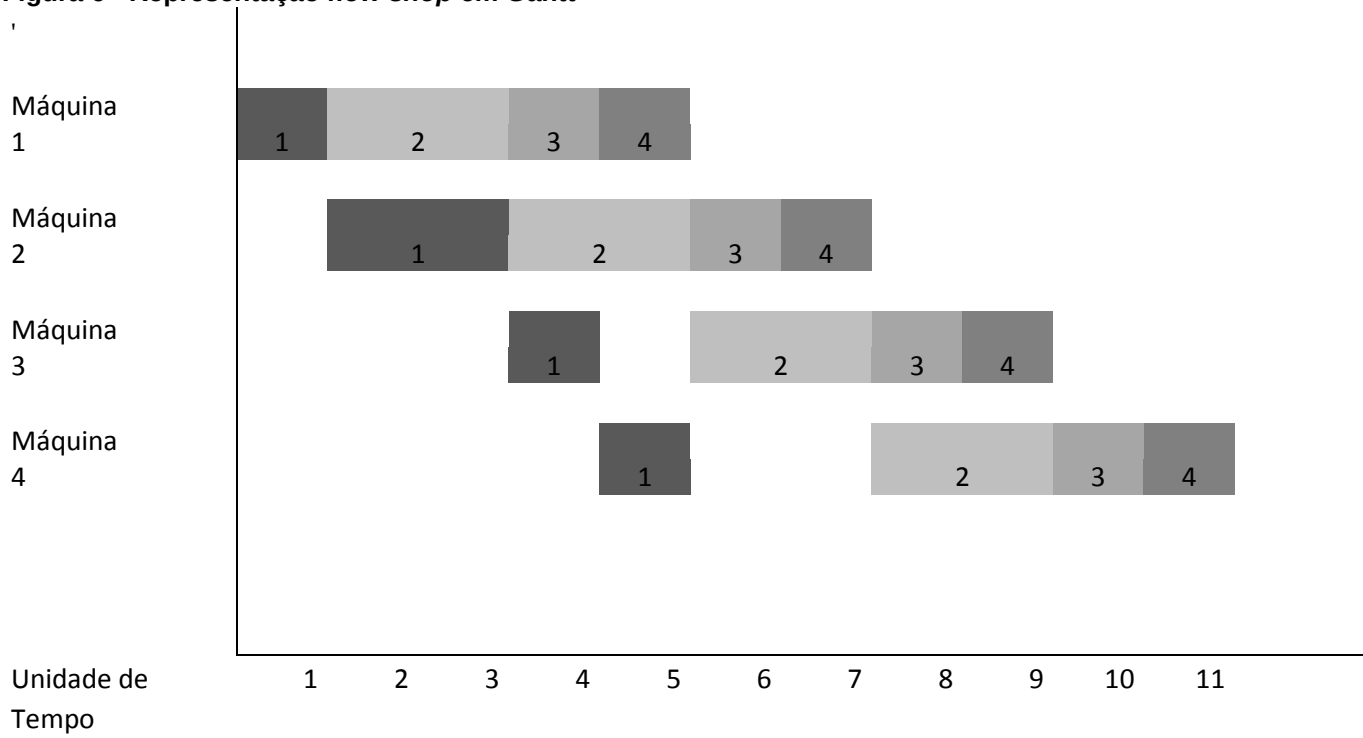
Quando existem três ou mais máquinas, o problema é *NP*-completo, entretanto se o problema apresentar apenas 2 máquinas ele pode ser resolvido em tempo polinomial como provado por Johnson em 1954.

Informalmente, existem m máquinas e um conjunto de tarefas que necessita ser processado por estas máquinas, este conjunto de tarefas será sempre processado na mesma sequência nestas máquinas. O tempo total das tarefas no "circuito" de máquinas é denominado *Makespan*.

Com a redução do *makespan*, como objetivo principal, isto é, a solução do problema, tem-se em contrapartida a maximização da produção do produto.

Geralmente são demonstradas as interações de *flow shop* por meio de um gráfico de Gantt. Caso o *flow shop* consista em um conjunto de 4 jobs e 4 máquinas, a sua representação por gráfico de Gantt, conforme a Figura 6 apresenta.

Figura 6 - Representação *flow shop* em Gantt



Fonte: Autoria própria

Nota-se que para a próxima tarefa iniciar, ou seja, a tarefa 2 ser iniciada somente após o encerramento da tarefa 1 na primeira máquina. Esta sequência será seguida por todas as tarefas. Não há mudança em nenhum dos aspectos referentes à sequência, em nenhum momento.

Uma das principais medidas de um *flow shop* é o *makespan* como citado anteriormente. Como exemplo da Figura 6, o *makespan* é o tempo total da *job* após a *job* ser processada por todas as máquinas (salientando que a *job* é o conjunto de tarefas, segundo a Figura 6 a *job* é constituída por 4 tarefas), o total do tempo, ou do *makespan*, é de 11 unidades de tempo.

Segundo Al-Harkan (2005) considerando que existe uma sequência arbitrária de *jobs* em cada máquina, a complexidade envolvida será de $(n!)^m$ sendo “n” o número de tarefas a serem processadas (tamanho da *job*) e “m” a quantidade de máquinas.

O caso especial de um *flow shop* existe quando adicionada a propriedade de permutação no caso geral.

3.3.1. *Flow Shop* Permutacional

Um *flow shop* é dito permutacional se apresentar algumas propriedades particulares como todas as tarefas devem ser processadas ao menos uma vez em cada máquina do sistema e se a tarefa X foi processada na *n*-ésima posição na primeira máquina do sistema, ela deve ser processada na *n*-ésima posição na última máquina do sistema.

O *flow shop* permutacional segue a mesma sequência de máquinas, por ser um *flow shop* e segue a mesma sequência de tarefas, por ser permutacional.

A complexidade deste caso específico conforme Al-Harkan (2005) é (n!) onde “n” refere-se ao número de tarefas que serão processadas por todas as máquinas.

A importância deste assunto surge desde que o trabalho de Johnson (1954) que aborda *flow shop* permutacional com duas máquinas foi publicado. Após este trabalho foram surgindo diversas pesquisas principalmente devido ao fato da complexidade do problema ser *NP*-completo. Também por causa de sua complexidade o foco principal dos trabalhos que surgiram foram em heurísticas. As mais conhecidas são a de Campbell, Dudek e Smith (1970), denominada método CDS e a de Nawaz, Enscore, Ham (1983), o método NEH.

Pode-se citar com destaque as metaheurísticas de *Simulated Annealing* de Osman e Potts (1989), Busca Tabu de Widmer e Hertz (1989) e Algoritmo Genético de Reeves (1995).

Ressaltam-se também os trabalhos que procuraram otimizar o método NEH ou superá-lo como o caso do N&M introduzido por Nagano e Moccellin (2002). Fuchigami (2005) explica como o método explorou que a prioridade do NEH de acordo com um limite inferior para o tempo de espera de uma tarefa entre o fim de sua operação em uma máquina qualquer e o início da operação da próxima. O método N&M se mostra superior para problemas com até 10 máquinas e 100 tarefas sendo que o esforço computacional não se mostra diferente do método NEH.

4. HEURÍSTICAS

De acordo com Fuchigami (2005), um método heurístico é um processo de solução de problema apoiado em critérios racionais ou computacionais para escolher um caminho entre vários possíveis, sem a preocupação de percorrer todas as possibilidades ou atingir a melhor opção. Esta busca por um determinado objetivo visa encontrar uma solução viável, pelo menos próxima da ótima, cujo tempo de computação seja aceitável. Na maioria dos casos, é mais viável se procurar por uma solução heurística do que uma solução ótima devido ao esforço computacional envolvido em uma solução ótima.

Métodos heurísticos podem ser definidos de diversas maneiras, Souza e Moccellin (2000) separam em dois grupos:

- Métodos construtivos: a sequência adotada como solução do problema é obtida:
  - diretamente a partir da ordenação das tarefas segundo índices de prioridade calculados em função dos tempos de processamento das tarefas, como por exemplo em Palmer (1965) e Gupta (1971); ou
  - escolhendo-se a melhor sequência das tarefas a partir de um conjunto de sequências também obtidas utilizando-se índices de prioridade associados às tarefas como em Campbell, Dudek e Smith (1970) e Hundal e Rajgopal (1988); ou ainda
  - a partir da geração sucessiva de sequências parciais (subsequências) das tarefas até a ordenação de uma sequência completa por meio de algum critério de inserção de tarefas, como por exemplo em NEH e N&M.
- Métodos melhorativos: obtém-se uma solução inicial e posteriormente por meio de algum procedimento iterativo (geralmente envolvendo trocas de posições das tarefas na sequência de processamento das máquinas) busca-se encontrar uma programação das tarefas melhor que a atual quanto à medida de desempenho adotada.

No presente trabalho serão abordadas quatro heurísticas: *Shortest Processing Time* (SPT), *Longest Proccessing Time* (LPT), heurística de Palmer e a heurística de Nawaz, Enscore, Ham (NEH).

4.1. HEURISTICA *SHORTEST PROCESSING TIME* (SPT)

É uma das heurísticas mais utilizadas por ser de rápido entendimento e custo computacional aceitável. Segundo Vizzoto e Branco (2009), a heurística SPT realiza a ordenação das atividades nas máquinas na ordem crescente da soma dos tempos de processamento.

Informalmente, cada tarefa passa um tempo pelo ciclo de máquinas, esse tempo define a nova sequência das tarefas pela ordem do menor tempo até o maior tempo.

A modo de exemplificar o sequenciamento das tarefas nesta heurística, foi utilizado a Tabela 1 para a ordenação utilizando a heurística SPT.

Tabela 1 - Tempos de processamento de quatro tarefas em quatro máquinas

	T1	T2	T3	T4
M1	3	2	12	3
M2	4	4	1	1
M3	7	7	4	2
M4	9	8	5	3

Fonte: Autoria própria

Na tabela, estão dispostas quatro tarefas (T1, T2, T3 e T4) que são processadas por quatro máquinas (M1, M2, M3 e M4). O tempo total de processamento da tarefa 1 (T1) é de 23 unidades de tempo, da tarefa 2 (T2) é de 21 unidades de tempo, da tarefa 3 (T3) de 22 unidades de tempo e da tarefa 4 é de 9 unidades de tempo. Aplicando a heurística SPT a sequência mudaria ao invés de ser T1, T2, T3 e T4 como no exemplo anterior, a nova sequência seria T4, T2, T3 e T1 por causa dos seus tempos de processamento.

4.2. HEURÍSTICA *LONGEST PROCESSING TIME* (LPT)

Bem como o SPT, é muito utilizada na literatura em geral pelo seu entendimento e custo computacional aceitável. Segundo Vizzoto e Branco (2009), esta heurística obedece uma ordem decrescente dos tempos de processamento.

Informalmente, cada tarefa passa um tempo pelo ciclo de máquinas, esse tempo define a nova sequência das tarefas pela ordem do maior tempo até o menor tempo.

A modo de exemplificar o sequenciamento das tarefas nesta heurística, foi utilizado a Tabela 2 para a ordenação utilizando a heurística LPT.

Tabela 2 - Tempos de processamento para quatro máquinas e quatro tarefas

	T1	T2	T3	T4
M1	5	2	12	10
M2	4	4	1	1
M3	7	7	4	2
M4	9	8	7	3

Fonte: Autoria própria

Na tabela, estão dispostas quatro tarefas (T1, T2, T3 e T4) que são processadas por quatro máquinas (M1, M2, M3 e M4). O tempo total de processamento da tarefa 1 (T1) é de 25 unidades de tempo, da tarefa 2 (T2) é de 21 unidades de tempo, da tarefa 3 (T3) de 24 unidades de tempo e da tarefa 4 é de 16 unidades de tempo. Aplicando a heurística LPT a sequência mudaria ao invés de ser T1, T2, T3 e T4 como no exemplo anterior, a nova sequência seria T4, T2, T3 e T1 por causa dos seus tempos de processamento.

4.3. HEURÍSTICA DE PALMER

Segundo Emmons e Vairaktarakis (2013), foi a heurística mais cedo publicada para o problema de *flow shop* permutacional datando 1965. Ainda conforme Emmons e Vairaktarakis (2013), ela tende a primeiro dar prioridade as tarefas cujo tempo de execução são mais altos (como no LPT) depois de feito o cálculo utilizando a seguinte fórmula, equação 1:

$$Si = \sum_{j=1}^{m} [m - (2j - 1)] Pi, j \quad (1)$$

Este coeficiente é aplicado em cada tarefa nos seus tempos individuais por máquina, em que é aplicada a máquina (de 1 até o número total de máquinas) e subtraído dela 2 valor de *j* menos 1, aonde j refere-se a máquina da tarefa em questão.

Para melhor entendimento segue a Tabela 3, seguida pela Tabela 4 que mostra a aplicação da heurística de Palmer.

Tabela 3 - Representação de dez tarefas em cinco máquinas

	Máquina				
Tarea	1	2	3	4	5
1	3	7	3	3	2
2	10	4	9	9	8
3	7	6	3	1	10
4	2	3	1	7	1
5	3	2	4	2	4
6	10	8	7	10	8
7	9	1	10	4	4
8	10	5	8	1	5
9	8	2	9	4	1
10	6	1	7	4	4

Fonte: RESTREPO (2010)

Tabela 4 - Representação de dez tarefas em cinco máquinas com heurística de Palmer aplicada

	Máquina				
Tarea	1	2	3	4	5
4	2	3	1	7	1
5	3	2	4	2	4
2	10	4	9	9	8
3	7	6	3	1	10
10	6	1	7	4	4
6	10	8	7	10	8
1	3	7	3	3	2
7	9	1	10	4	4
9	8	2	9	4	1
8	10	5	8	1	5

Fonte: RESTREPO (2010)

Observa-se que a ordem que era T1, T2, T3, T4, T5, T6, T7, T8, T9 e T10 alterou para T4, T5, T2, T3, T10, T6, T1, T7, T9, T8 isto se deve pois, como exemplo, a Tarefa 4 com o coeficiente aplicado alteram-se os tempos de execução na máquina 1 de 2 para -8, na máquina 2 de 3 para -6, na máquina 3 de 1 para 0, na máquina 4 de 7 para 14 e na máquina 5 de 1 para 4, a soma dos novos coeficientes de tempo para cada máquina resulta em 4, em contrapartida a Tarefa 8 possui um coeficiente de -28 sendo a última.

4.4. HEURÍSTICA NAWAZ, ENSCORE, HAM (NEH)

O trabalho de Nawaz, Enscore e Ham é, até os dias atuais, o mais conceituado e citado em diversas pesquisas. Segundo Nawaz (2008), a heurística possui dois estágios: o primeiro é a geração de uma ordem inicial de tarefas respeitando um valor indicado e o segundo é a inserção iterativa de tarefas em uma sequência parcial de acordo com a ordem inicial do primeiro estágio.

De acordo ainda com Nawaz (2008), são quatro passos para a minimização do *makespan* usando o algoritmo NEH:

i) Ordenar as tarefas pela soma não crescente dos tempos de processo das máquinas;

ii) Selecionar as duas primeiras tarefas e sequenciá-las em ordem de minimizar o *makespan* parcial como se apenas houvessem estas duas tarefas;

iii) Para k = 3 até n fazer o passo quatro. Isto é, para o número de 3 tarefas até o número total de tarefas, realizar o passo quatro;

iv) Inserir a *k*-ésima tarefa na sequência sempre minimizando o *makespan* parcial entre todos os *k* possíveis.

Exemplificando os passos temos a Tabela 5, temos por objetivo com ela minimizar o *makespan* utilizando a heurística NEH.

Tabela 5 - Composição de cinco máquinas e quatro tarefas

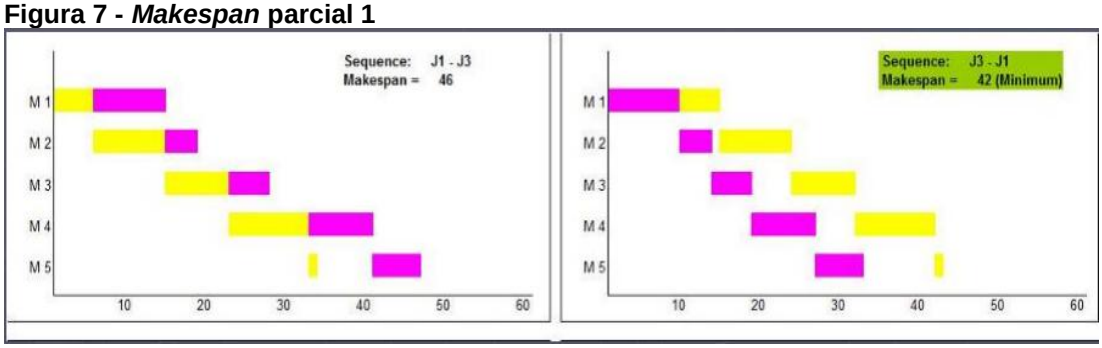
	M1	M2	M3	M4	M5
T1	5	9	8	10	1
T2	9	3	10	1	8
T3	9	4	5	8	6
T4	4	8	8	7	2

Fonte: Nawaz (2008)

O primeiro passo é calcular os tempos de execução de cada tarefa. T1 possui 33 unidades de tempo de execução, T2 possui 31 unidades de tempo de execução, T3 possui 32 unidades de tempo de execução e, por fim, T4 possui 29 unidades de tempo de execução. A sequência então deve ser T1, T3, T2 e T4.

O segundo passo consiste em selecionar as duas primeiras tarefas e minimizar o *makespan* parcial entre estas. A Figura 7 representa à esquerda o *makespan* da sequência T1-T3 e à direita o *makespan* da sequência T3-T1.

Figura 7 - *Makespan* parcial 1



Fonte: Nawaz (2008)

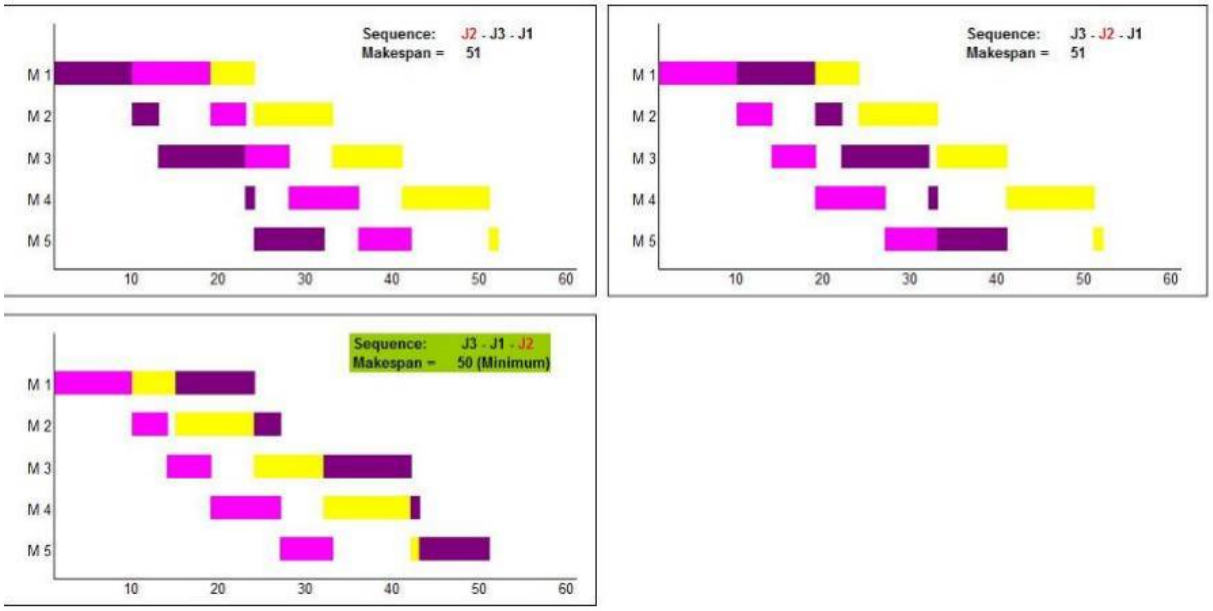
Deste modo, temos que o *makespan* a direita (da sequência parcial T3-T1) é o menor entre os dois, deste modo a sequência deve sempre permanecer iniciando com T3-T1.

O terceiro passo remete ao quarto, aonde deve-se inserir as tarefas em todas as posições possíveis e calcular o *makespan* parcial das sequências de $k = 3$ até a *n*-ésima tarefa.

O quarto e último passo, devemos manter a sequência T3-T1 e selecionar T2 para a inserção em todas as *k*-ésimas posições possíveis. No caso temos três possíveis posições para T2 (T2-T3-T1, T3-T2-T1, T3-T1-T2). Feitas as inserções de

T2, é calculado novamente um *makespan* parcial para as possíveis sequências, como é demonstrado na Figura 8.

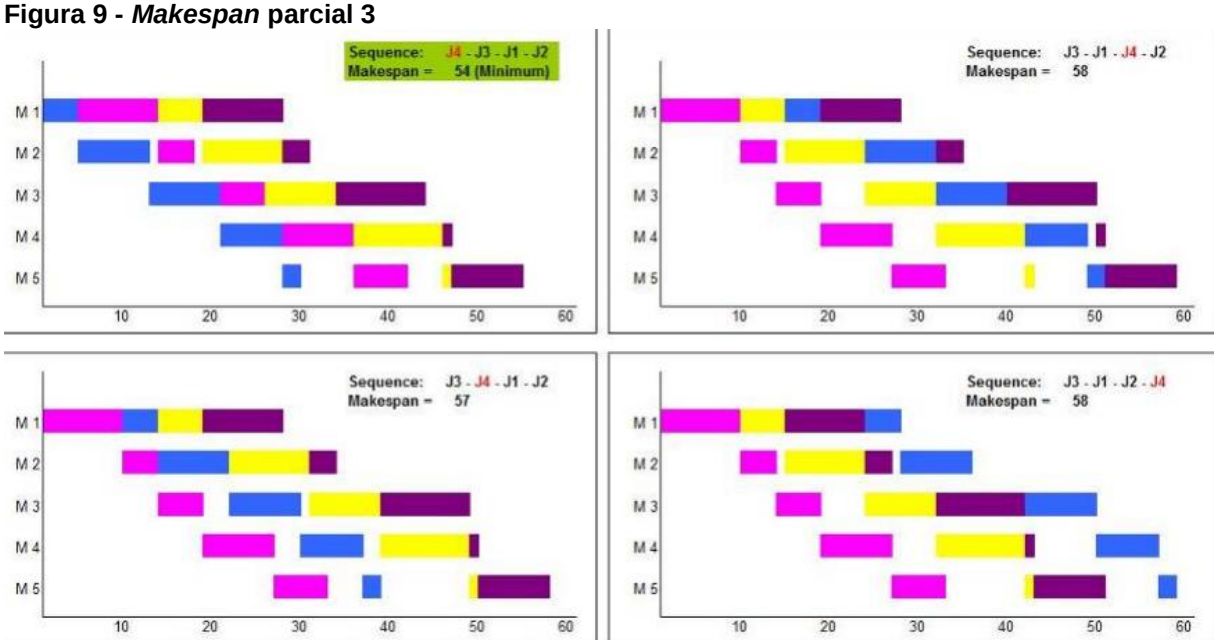
Figura 8 - *Makespan* parcial 2



Fonte: Nawaz (2008)

Repete-se o quarto passo, desta vez inserindo a T4 em todas as possíveis posições (T4-T3-T1-T2, T3-T1-T4-T2, T3-T4-T1-T2 e T3-T1-T2-T4), calculando em cada sequência o seu *makespan* parcial. A Figura 9 ilustra as inserções bem como mostra que a sequência para este caso deve ser T4-T3-T1-T2.

Figura 9 - *Makespan* parcial 3



Fonte: Nawaz (2008)

5. METODOLOGIA

De acordo com Gil (1999), a pesquisa é “um processo formal e sistemático de desenvolvimento do método científico. O objetivo fundamental da pesquisa é descobrir respostas para problemas mediante o emprego de procedimentos científicos”.

A pesquisa é classificada como um procedimento formal cujo é utilizado método de pensamento reflexivo que necessita de abordagem científica e visa e desenvolver diretrizes para a resposta das perguntas, mediante a utilização de metodologia científica (MARCONI; LAKATOS, 2011).

Para o presente estudo, foram realizados os seguintes estágios de aprendizado e execução:

- Revisão de Literatura: foram analisadas teses, dissertações, artigos científicos, livros, pesquisas (entre vários materiais) sobre o tema do trabalho e suas derivações. Este estágio se deve a necessidade de adquirir conhecimento sobre o problema, bem como as soluções usadas por pesquisadores e a evolução destas soluções.
- Conjunto de instâncias: geração de conjuntos de exemplos que foram analisados com o intuito de concretizar o objetivo do trabalho, minimizar o *makespan*.
- Experimentos computacionais: após gerar o banco de dados com os conjuntos de instâncias que tinham como base a variação da quantidade de máquinas e tarefas, as heurísticas são implementadas para avaliação do objetivo.
- Análise dos experimentos computacionais: após os testes de cada heurística em determinados conjuntos de instâncias, existe a necessidade de analisar os resultados para qualificação dos mesmos.
- Interpretação dos resultados e dedução: com as informações compiladas, são comparados os resultados de cada heurística para o objetivo proposto, diminuição do *makespan*. Deste modo pode-se classificar e qualificar as soluções.

6. DESENVOLVIMENTO

Para o estágio de experimentação computacional foram utilizadas classes de problemas que continham 500 problemas. Foram geradas dezesseis classes combinatórias de *n* tarefas por *m* máquinas. As classes geradas são as combinatórias descritas: 25x25, 25x50, 25x75, 25x100, 50x25, 50x50, 50x75, 50x100, 75x25, 75x50, 75x75, 75x100, 100x25, 100x50, 100x75 e 100x100.

Por meio de um *software* gerador de dados das combinatórias descritas no parágrafo anterior, foi formada uma base de dados de 8000 arquivos distintos, sendo 500 arquivos por classe. Os arquivos de saída do gerador de dados serviram como base para as comparações das heurísticas SPT, LPT, Palmer, SPTNEH, LPTNEH e PalmerNEH (aonde as três últimas são a combinação das heurísticas com a heurística NEH).

Para cada uma das heurísticas, foi executada a comparação entre os valores apresentados pelos seus pares, isto é, foi feita a comparação direta dos valores minimizados de *makespan* pela heurística SPT e SPTNEH, LPT e LPTNEH, Palmer e PalmerNEH.

Como existem dois grupos de heurísticas, foram realizados 16000 testes e analisado o desempenho dos mesmos.

Para que o desempenho dos algoritmos foi utilizado: porcentagem de sucesso (PS), desvio relativo médio (DRM) e tempo médio de computação (TMC). A porcentagem de sucesso é definida pela comparação direta do *makespan* minimizado nos casos. O desvio relativo médio é a relação da comparação do melhor caso com o resultado geral, ele será aplicado tanto no melhor caso por classe, como na comparação direta dos pares. O cálculo do desvio relativo médio é realizado pela equação 2:

$$DRM = \frac{Dh - Dm}{Dm} \; x \; 100 \qquad (2)$$

Dh é o valor da função-objetivo (*makespan*) da heurística que está sendo analisada enquanto o Dm é o valor da função-objetivo da heurística que apresentou melhor solução (minimização do *makespan*). O tempo de execução das soluções foi guiado pelo Código 1.

Código 1 - Desempenho em tempo pela função *clock*

```
clock_t tic = clock();
{...}
clock_t toc = clock();
x = (double)(toc - tic) / CLOCKS_PER_SEC
```

Fonte: Autoria própria

A função *clock* é inserida no começo da execução do sistema que aplica as heurísticas e depois ao fim da mesma, sendo convertida então para o tempo em segundos.

Por haver dependência do *clock* e, este ser um elemento variável, foram realizados cinco vezes cada teste, chegando-se a soma de 80.000 testes. Quanto ao *clock* sempre foi selecionado o *clock* médio, isto é, para o grupo de cinco testes, o terceiro maior *clock* foi selecionado, sendo que os resultados da função-objetivo permaneciam inalterados.

A experimentação computacional foi realizada em um microcomputador com processador Intel Core 5 4210U 1,7GHz, com memória RAM de 8Gb, sistema operacional Windows 10.

6.1. ANÁLISE DOS RESULTADOS

As heurísticas SPT, LPT e Palmer, bem como as suas combinações com o NEH, foram analisadas com base nas suas porcentagens de sucesso, desvio relativo médio e tempo médio de computação, para cada uma das classes de *n* tarefas (25, 50, 75 e 100) e *m* máquinas (25, 50, 75 e 100). Levando em consideração a variação do *clock*, as 8000 instâncias foram testadas em dois conjuntos de heurísticas e cinco vezes por esta variação, totalizando 80.000 testes.

As tabelas e gráficos apresentados na sequência demonstram os dados obtidos por meio de cálculos descritos anteriormente para que possam ser comparados os métodos e analisar a eficiência da heurística NEH em comparação aos demais.

Tabela 6 - Taxa de Sucesso entre heurísticas

Classes	SPTxSPTNEH	LPTxLPTNEH	PALMERxPALMERNEH
25x25	100%	100%	100%
25x50	100%	100%	100%
25x75	100%	100%	100%
25x100	100%	100%	100%
50x25	100%	100%	100%
50x50	100%	100%	100%
50x75	100%	100%	100%
50x100	100%	100%	100%
75x25	100%	100%	100%
75x50	100%	100%	100%
75x75	100%	100%	100%
75x100	100%	100%	100%
100x25	100%	100%	100%
100x50	100%	100%	100%
100x75	100%	100%	100%
100x100	100%	100%	100%

Fonte: Autoria própria

Na tabela 6 observa-se a comparação de sucesso em todas as classes de problemas, nota-se que as classes com a heurística NEH apresentaram 100% de sucesso em comparação as outras, isto é, as heurísticas combinadas com a NEH possuem melhor minimização do *makespan* em comparação das heurísticas sem o NEH. Este resultado era esperado pela natureza e complexidade das heurísticas.

Tabela 7 - Classe 25 tarefas desvio relativo médio

25x25	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	3127	1876	3125	1781	2927	1814
Desvio Relativo Médio	66,6%		75,4%		61,3%	
25x50	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	4824	3039	4786	2824	4553	2914
Desvio Relativo Médio	58,7%		69,4%		56,25%	
25x75	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	6354	4448	6458	4327	6183	4266
Desvio Relativo Médio	42,8%		49,2%		44,9%	
25x100	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	7829	5793	8011	5528	7588	5463
Desvio Relativo Médio	35,1%		44,9%		38,8%	

Fonte: Autoria própria

Tabela 8 - Classe 50 tarefas desvio relativo médio

50x25	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	4849	3055	4839	2903	4390	2678
Desvio Relativo Médio	58,73%		66,66%		63,93%	
50x50	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	6755	4391	6715	4163	6422	4367
Desvio Relativo Médio	53,84%		61,29%		47,05%	
50x75	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	8569	5741	8538	5635	8114	5680
Desvio Relativo Médio	49,25%		51,51%		42,85%	
50x100	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	10258	7283	10187	6927	9763	7029
Desvio Relativo Médio	40,84%		47,05%		38,88%	

Fonte: Autoria própria

Tabela 9 - Classe 75 tarefas desvio relativo médio

75x25	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	6424	4176	6444	3995	5959	3814
Desvio Relativo Médio	53,84%		61,29%		56,25%	
75x50	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	8458	5667	8448	5322	8062	5482
Desvio Relativo Médio	49,25%		58,73%		47,05%	
75x75	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	10391	7170	10364	6633	9987	6891
Desvio Relativo Médio	44,92%		56,25%		44,92%	
75x100	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	12171	8520	12146	8138	11782	8365
Desvio Relativo Médio	42,85%		49,25%		40,84%	

Fonte: Autoria própria

Tabela 10 - Classe 100 tarefas desvio relativo médio

100x25	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	7944	5322	7887	4969	7293	4813
Desvio Relativo Médio	49,25%		58,73%		51,51%	
100x50	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	10206	7144	10122	6681	9630	6548
Desvio Relativo Médio	42,85%		51,51%		47,05%	
100x75	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	12200	8906	12193	8657	11735	8449
Desvio Relativo Médio	36,98%		40,84%		38,88%	
100x100	SPT	SPTNEH	LPT	LPTNEH	PALMER	PALMERNEH
Média	14047	10535	14010	10227	13605	10340
Desvio Relativo Médio	33,33%		36,98%		31,57%	

Fonte: Autoria própria

As Tabelas 7, 8, 9 e 10 mostram os desvios relativos entre os melhores casos das heurísticas em comparação com as suas combinatórias com o NEH. Destacam-se dois comportamentos baseados nestes dados:

I. Conforme o aumento do número de máquinas, a combinação com a NEH perde o potencial de diminuir o *makespan*.

II. Das três heurísticas, a que pior se desenvolveu foi a do SPT, isto pode indicar que as diferenças entre SPT e NEH contribuíram para isto.

Estas tabelas também ajudam a compreender a Tabela 6, aonde é mostrado que as heurísticas que contém a NEH são melhores que as sem a NEH, isto levando em consideração a minimização do *makespan*.

Quanto ao *clock*, foram cálculos os ciclos completos das operações, em segundos, conforme mostrado anteriormente por meio do Código 1. Assim foram gerados dezesseis resultados para as heurísticas e dezesseis resultados para as heurísticas com a NEH, conforme Tabela 11:

Tabela 11 - Diferença de *clocks*

Classe	SPT/LPT/ PALMER	SPTNEH/LPTNEH/PAL MERNEH	Diferença em Segundos	Diferença percentual
25x25	75,24	82,28	7,04	8,56%
25x50	123,74	158,87	35,13	22,11%
25x75	155,92	228,76	72,84	31,84%
25x100	209,78	310,18	100,40	32,37%
50x25	130,77	142,08	11,31	7,96%
50x50	211,08	257,56	46,48	18,05%
50x75	300,14	389,78	89,64	23,00%
50x100	409,00	617,41	208,41	33,76%
75x25	190,11	213,88	23,77	11,11%
75x50	351,25	413,56	62,31	15,07%
75x75	483,77	589,42	105,65	17,92%
75x100	609,67	876,42	266,75	30,44%
100x25	250,54	285,29	34,75	12,18%
100x50	468,55	551,28	82,73	15,01%
100x75	613,11	749,87	136,76	18,24%
100x100	772,48	1287,48	515,00	40,00%

Fonte: Autoria própria

Deste modo, evidencia-se que conforme o maior número de operações, a heurística NEH começa a consumir maior tempo, ou seja, a heurística NEH começa

a perder desempenho quanto maior o número de tarefas e, principalmente, de máquinas.

Tabela 12 - Comparações heurísticas NEH

Classes	SPTNEH	LPTNEH	PALMERNEH
25x25	1876	1781	1814
Desvio Relativo Médio	5,33%	0	1,85%
25x50	3039	2824	2914
Desvio Relativo Médio	7,61%	0	3,18%
25x75	4448	4327	4266
Desvio Relativo Médio	4,26%	1,42%	0
25x100	5793	5528	5463
Desvio Relativo Médio	6,04%	1,18%	0
50x25	3055	2903	2678
Desvio Relativo Médio	14,07%	8,4%	0
50x50	4391	4163	4367
Desvio Relativo Médio	5,47%	0	4,9%
50x75	5741	5635	5680
Desvio Relativo Médio	1,8%	0	0,79%
50x100	7283	6927	7029
Desvio Relativo Médio	5,13%	0	1,47%
75x25	4176	3995	3814
Desvio Relativo Médio	9,49%	4,74%	0
75x50	5667	5322	5482
Desvio Relativo Médio	6,48%	0	3%
75x75	7170	6633	6891
Desvio Relativo Médio	8,09%	0	3,88%
75x100	8520	8138	8365
Desvio Relativo Médio	4,69%	0	2,78%
100x25	5322	4969	4813
Desvio Relativo Médio	10,57%	3,24%	0
100x50	7144	6681	6548
Desvio Relativo Médio	9,1%	2,03%	0
100x75	8906	8657	8449
Desvio Relativo Médio	5,4%	2,46%	0
100x100	10535	10227	10340
Desvio Relativo Médio	3,01%	0	1,1%

Fonte: Autoria própria

A Tabela 12 representa o desvio relativo médio entre as heurísticas NEH, destacando nesse caso que a heurística SPT, neste quesito, sempre foi alvo de comparação, significando que não conseguiu atingir o melhor resultado em nenhuma das classes testadas.

A Tabela 13 apresenta a porcentagem de sucesso entre as heurísticas, mostrando a variação dos valores para cada classe. Ou seja, quantas vezes das 500 instâncias por classe a heurística obteve o melhor resultado.

Tabela 13 - Porcentagem de sucesso entre heurísticas combinadas

Classes	SPTNEH	LPTNEH	PALMERNEH
25x25	13,6%	54,2%	32,2%
25x50	8,2%	67,2%	24,6%
25x75	11,4%	33%	55,6%
25x100	3,8%	39,2%	57%
50x25	3,2%	61%	35,8%
50x50	12,6%	74%	13,4%
50x75	12,8%	33,4%	53,8%
50x100	5%	33,4%	61,6%
75x25	2,6%	78,8%	18,6%
75x50	7,4%	70,4%	22,2%
75x75	0,2%	85,8%	14%
75x100	4,4%	71,6%	24%
100x25	0,6%	68,4%	31%
100x50	0%	71,8%	28,2%
100x75	1,4%	79,4%	19,2%
100x100	6,8%	59,2%	34%

Fonte: Autoria própria

Por meio da Tabela 13 observa-se que a combinação das heurísticas SPT com NEH não obteve boa porcentagem de sucesso, provavelmente pelo modo que se desenvolvem as heurísticas como explicado anteriormente.

Também se ressalta que a partir de um número de tarefas a combinação das heurísticas LPT e NEH, desenvolve-se com melhores resultados.

7. CONCLUSÕES

As heurísticas avaliadas no presente trabalho, SPT, LPT, Palmer e NEH, bem como as suas combinações, foram executadas com o objetivo de minimizar o *makespan* em ambientes de *flow shop* permutacional. Com diferentes heurísticas é possível a comparação dos resultados das saídas para diferenciar tanto o seu desempenho quanto a função objetivo, quanto no seu desempenho computacional, evidenciou-se que uma melhora na qualidade de solução era compensada por um aumento no gasto do tempo de computação.

O presente trabalho teve por objetivo atuar em um problema de programação de tarefas em máquinas em um ambiente *flow shop* com a restrição permutacional. Seguindo o objetivo, neste ambiente, de minimizar o *makespan* e, principalmente, a comparação dos efeitos nos resultados quando é usado a heurística NEH.

Foram realizadas 16.000 resoluções de problemas divididos nas 16 classes de problemas, sendo que devido a variação do *clock* os problemas foram resolvidos cinco vezes cada um, totalizando em 80.000 resoluções pelas 6 heurísticas.

Dentre todas as heurísticas, fica evidente que sempre que havia a combinação com a heurística NEH, a função-objetivo foi melhorada além das heurísticas. Em contrapartida sempre que aplicada a heurística NEH, o tempo de processamento aumentava.

Os problemas de programação de tarefas em máquinas podem ser testados e implementados no plano tangível, isto é, podem ser testados e implementados no mundo real, em empresas e em outros casos aplicáveis. Releva-se que a função-objetivo deste trabalho foi o *makespan*, e os mesmos casos poderiam ser testados para outras funções-objetivos.

Por fim, com diferentes funções-objetivos e heurísticas, este trabalho pode ser estendido, aperfeiçoado e usado como base para outros estudos. Destaca-se a importância da heurística NEH, que mesmo depois de anos, está entre os melhores resultados e é motivo de aperfeiçoamento e pesquisa.

REFERÊNCIAS

AL-HARKAN, I. M. **Algorithms For Sequencing And Scheduling.** Saudi Arabia, Riyadh, College of Engineering, King Saud University, 2005. Disponível em: <http://faculty.ksu.edu.sa/ialharkan/IE428/Algorithms_for_Sequencing_and_Scheduling1.pdf>. Acesso em: 15 nov. 2015.

BAKER, K. R. **Introduction To Sequencing and Scheduling.** New York: John Wiley & Sons, Inc. 1997.

BARAZ, D.; MOSHEIOV, G. **A Note on a Greedy Heuristic for the Flow Shop Makespan Minimization With No Machine Idle-Time.** European Journal of Operational Research ed. 184, p. 810, 2008.

CAMPBELL, H. G.; DUDEK, R. A.; SMITH, M. L. **A Heuristic Algorithm For The *n* Job *m* Machine Sequencing Problem.** Management Science, Rhode Island. V.16. 1970

CHAKRABORTY, U. K., **Computational Intelligence in Flow Shop and Job Scheduling.** Springer Science & Business Media, 2009.

EMMONS, H.; VAIRAKTARAKIS, G.; **Flow Shop Scheduling: Theoretical Results, Algorithms and Applications.** 1. ed. International Series in Operations Research & Management Science 182. 2013.

FUCHIGAMI, Hélio Y. **Métodos Heurísticos Construtivos Para o Problema de Programação da Produção em Sistemas Flowshop Híbridos com Tempos de Preparação das Máquinas Assimétricos e Dependentes da Sequência.** 2005. 135f. Dissertação (Mestreado em Engenharia de Produção) – Escola de Engenharia de São Carlos, Universidade São Paulo, São Carlos, 2005.

GAREY, M. R.; JOHNSON, D. S.; **Computers and Intractability:** A guide to the theory of NP-Completeness. 1. ed. Murray Hill, New Jersey: Bell Laboratories, 1979.

GIL, Antonio Carlos. (1999). **Métodos e técnicas de Pesquisa Social.** São Paulo: Atlas.

GUPTA, J. N. D.; **Functional Heuristic Algorithm For The Flowshop Scheduling Problem.** Operational Research Quarterly, Oxford, v:22, n1, p.39-47. 1971

LAKATOS, E. M.; MARCONI, M. A.; **Fundamentos de Metodologia Científica**. 3. ed. São Paulo: Atlas, 1991.

HAX, A. C.; CANDEA, D.; **Production and Inventory Management**. New Jersey: Prentice Hall, 1984.

HUNDAL, T. S.; RAJGOPAL, J.; **An Extension of Palmer's Heuristic for the Flow-Shop Scheduling Problem**," International Journal of Production Research**,** Vol. 26, No. 6, pp. 1119-1124, 1988.

MARCONI, M.A.; LAKATOS, E.M. **Metodologia do Trabalho Científico**. São Paulo: Atlas, 2011.

MacCARTHY, B. L.; LIU, J. Y.; **Adressing a 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.

MOCCELLIN, J. V.; NAGANO, M. S.; **Flow Shop Híbrido Com Estágios Gargalos.** Simpósio Brasileiro de Pesquisa Operacional, XXXV. Natal, 2003.

MONKS, J. G., **Administração da Produção e Operações.** 2 ed. São Paulo: McGraw Hill, 1987.

NAWAZ, M. **NEH Algorithm.** Apresentação. Instituto Real de Tecnologia de Melbourne. 2008

NAWAZ, M.; ENSCORE, E. E.; HAM, A. **A Heuristic Algorithm For The *m*-machine *n*-job Flow-Shop Sequencing Problem.** International Journal of Management Science 11. Omega. P.91-95. 1983.

PALMER, D. S.; **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, Qk.; RUIZ, R.; **A Comprehensive Review and Evaluation of Permutation Flowshop Heuristics to Minimize Flowtime.**Computers and Operations Research. v. 40. p. 117-128. 2013.

PINEDO, M.; **Scheduling: Theory, Algorithms and Systems.** 3 ed. Spinger. 2008.

RESTREPO, J. H.; **Aplicación de la Heuristica de Palmer en la Secuenciácion de n Tareas em m Máquinas: Um Caso de Estudio.** Sciencia Et Techica. v. 3. Ed. 46. p. 175-177. 2010.

SANTORO, M. C.; **Modelo de Programação para Produção Intermitente com Composição de Produtos Variável no Tempo.** Tese de Doutorado, São Paulo, Escola Politécnica Da Universidade de São Paulo, 1982.

SLACK, N.; **Administração da Produção.** 1.ed. São Paulo: Atlas. 1999.

SOUZA, A. B. D.; MOCCELLIN, J. V.; **Metaheurística Hibrída Algoritmo Genético Busca-Tabu Para a Programação de Operações *Flow Shop*.** Simpósio Brasileiro de Pesquisa Operacional, XXII, Viçosa. 2000.

TAILLARD, E.; **Benchmarks for Basic Scheduling Problems.** European Journal of Operational Research. v. 64. p. 278-285. Amsterdam. 1993