UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO
ENGENHARIA DE PRODUÇÃO

DIEGO FLORIDO PORTO
SAYANE SCHWAB BIEDERMANN

OTIMIZAÇÃO DO MAKESPAN EM UM PROBLEMA FLOWSHOP PERMUTACIONAL DISTRIBUÍDO EM CÉLULAS UTILIZANDO MÉTODOS HEURÍSTICOS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2019

DIEGO FLORIDO PORTO

SAYANE SCHWAB BIEDERMANN

OTIMIZAÇÃO DO MAKESPAN EM UM PROBLEMA FLOWSHOP PERMUTACIONAL DISTRIBUÍDO EM CÉLULAS UTILIZANDO MÉTODOS HEURÍSTICOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Produção, do Departamento de Engenharia de Produção, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Fábio José Ceron Branco.

PONTA GROSSA

2019

	Ministério da Educação **UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ** **CÂMPUS PONTA GROSSA** Departamento Acadêmico de Engenharia de Produção	

TERMO DE APROVAÇÃO DE TCC

OTIMIZAÇÃO DO MAKESPAN EM UM PROBLEMA FLOWSHOP PERMUTACIONAL DISTRIBUÍDO EM CÉLULAS UTILIZANDO MÉTODOS HEURÍSTICOS

por

DIEGO FLORIDO PORTO E SAYANE SCHWAB BIEDERMANN

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 27 de novembro de 2019 como requisito parcial para a obtenção do título de Bacharel em Engenharia de Produção. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

A Folha de Aprovação assinada encontra-se na Coordenação do Curso de Engenharia de Produção.

_____
Prof. Dr. Fábio José Ceron Branco
Prof. Presidente da banca

_____
Prof. Dr. João Carlos Colmenero
Membro titular

_____
Profª. Drª. Yslene Rocha Kachba
Membro titular

AGRADECIMENTOS

Quando imaginamos como será nosso percurso ao longo da vida, nunca paramos para olhar que quase tudo que está em nossa volta muitas vezes sequer foi sonhado em existir.

A vida é como um barco navegando de porto a porto pela imensidão do mundo, enfrentando tempestades, dias de calmaria, dias quentes e noites na escuridão, mas sempre continuando em uma direção.

Esses anos de universidade nos mostraram caminhos a seguir, pessoas a compartilhar cada momento, e, por tudo isso, chegamos a esse dia. O dia que marca o fim de uma fase e também o começo de outra nova.

A todos estes que acompanharam e compartilharam nossa trajetória, agradecemos:

A nossos pais, que nos foram base para continuarmos nosso caminho mesmo quando achávamos que não iríamos a lugar algum, nos motivando, muitas vezes nos consolando e acreditando em nós em momentos que não achávamos que fôssemos capazes.

Aos nossos familiares, que em muitos momentos não entendiam a nossa situação, mas estavam dispostos a nos ajudar.

Ao nosso orientador Prof. Dr. Fábio Branco, que nos acompanhou não somente neste trabalho, mas em todo o período de graduação, sempre disponível a nos ajudar, responder nossas dúvidas e nos conduzir para o nosso desenvolvimento acadêmico.

Aos nossos amigos, alguns hoje distantes, que nos marcaram a cada dia que dividimos ao longo desses anos, que nos deram o companheirismo e a amizade em momentos de alegria, desespero, tristeza, ansiedade, choros e gargalhadas e que estarão conosco por toda a nossa vida.

A nossa universidade, UTFPR, por tantos momentos únicos nesses anos, por todas as histórias contadas e vividas entre seus blocos e campos.

Por fim, pela amizade que nasceu nesses corredores e que hoje escreve esse trabalho.

RESUMO

PORTO, Diego F, BIEDERMANN, Sayane S. **Otimização do makespan em um problema flowshop permutacional distribuído em células utilizando métodos heurísticos**. 2019. 54 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Produção) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2019.

A crescente descentralização das indústrias pelo mundo disseminou o estudo do problema de sequenciamento permutacional distribuído (DPFSP), no qual há complexidade maior em relação a um problema *flowshop* permutacional (PFSP), e isto se deve ao fato de existir uma variável a mais, a unidade fabril a ser utilizada, com a mesma quantidade de máquinas. Esta nova distribuição de fábricas tem sido utilizada por se apresentar mais vantajosa ao minimizar os custos de processos, melhorar a administração e assegurar uma qualidade superior. Para o presente estudo, foi desenvolvida, através de experimentos computacionais, a análise entre quatro diferentes heurísticas (A, B, C e D), derivadas de outras já existentes na literatura, principalmente a NEH e uma quinta (C2), resultado de uma alternativa à heurística com o melhor desempenho comparada às outras, nesse caso a C. Como critérios de desempenho foram utilizados a minimização da duração total da programação (*makespan*) e o tempo de processamento das quatro heurísticas propostas. O estudo revelou a superioridade da heurística C no campo experimental proposto, com um valor de *makespan* mais baixo, alto índice de sucesso e baixo desvio, garantindo assim sua eficiência, tudo isso sem deixar de lado um tempo viável de processamento da máquina.

Palavras Chaves: *Flowshop. Flowshop* distribuído. *Makespan*. Otimização. Heurística.

ABSTRACT

PORTO, Diego F, BIEDERMANN, Sayane S. **Makespan optimization in a distributed permutation flowshop scheduling problem using heuristic methods**. 2019. 54 P. Work of Conclusion Course (Graduation in Industrial Engineering) - Federal Technology University - Paraná. Ponta Grossa, 2019.

The increasing decentralization of industries around the world has spread the study of distributed permutational sequencing (DPFSP),where there is greater complexity in relation to a permutational flowshop problem (PFSP), and this is due to the fact that there is one more variable, the plant to be used, with the same number of machines. This new plant distribution has been used for being more advantageous in minimizing process costs, improving management and ensuring superior quality. For the present study, it was developed, through computational experiments, the analysis between four different heuristics (A, B, C and D), derived from others already existing in the literature, mainly NEH and a fifth (C2), the result of a alternative to the heuristic with the best performance compared to the others, in this case the C. Performance criteria were minimization of the total programming duration (makespan) and the processing time of the four proposed heuristics. The study revealed the superiority of the C heuristic in the proposed experimental field, with a lower makespan value, high success rate and low deviation, thus ensuring its efficiency, all without neglecting a viable machine processing time.

Keywords: Flowshop. Distributed Flowshop. Makespan. Optimization. Heuristic.

Sumário

1 INTRODUÇÃO

A evolução da linha de produção foi um fator determinante no desenvolvimento da industrialização, nesse quesito um fator chave em todo esse avanço é a administração de todos os fatores e recursos envolvidos num processo produtivo, desde insumos até alocação de espaço. Neste âmbito, aparece o problema de agendamento de tarefas (*scheduling*), utilizado no meio industrial a partir dos primeiros estudos de Johnson (1954), buscando assim uma otimização da produção através do sequenciamento de tarefas necessárias para a confecção de um produto final no maquinário disponível. Dessa maneira, o *scheduling* tem como finalidade alocar os recursos de fabricação, como máquinas e ferramentas, aos processos e operações, sendo que estes estão sujeitos a restrições tais como tempo de liberação, datas de entrega e capacidade da produção.

As empresas buscam cada vez mais se desenvolver em seus setores, isso impulsionou a descentralização industrial e promoveu a alocação física de diversas plantas das empresas ao redor do mundo. Segundo Naderi e Ruiz (2010), nos dias atuais, essa tendência de unicidade de planta de fábrica vem se reduzindo, pois segundo Moon, Kim e Hur (2002), as manufaturas distribuídas tem permitido que as empresas alcancem maior qualidade dos produtos, menores riscos de gerenciamento e custos da produção cada vez menores. Entretanto, a maioria dos estudos existentes é pautada na área econômica da produção distribuída, sendo raro o estudo acerca da capacidade finita que as fábricas possuem.

Ao passo que um problema de *flowshop* permutacional tradicional envolve apenas o sequenciamento de uma ou mais linha de apenas uma única unidade, o sequenciamento distribuído se torna mais complexo, pois existem duas variáveis de decisão importantes a serem consideradas: a atribuição de trabalho às fábricas e o sequenciamento das tarefas em cada uma delas. Essas decisões estão intimamente interligadas e não podem ser resolvidas individualmente caso o alto desempenho seja desejado.

O gerenciamento dessa produção é uma parte muito importante em toda a cadeia, envolvendo, através de um bom plano de gerenciamento, objetivos como reduzir os custos, reduzir as perdas e maximizar a alocação de recursos e é dever

do gerente desenvolver meios de desempenhar essas funções da maneira mais otimizada possível.

Pensando nisto, as heurísticas são algoritmos capazes de facilitar a busca por soluções boas e rápidas em problemas complexos. Existem, na literatura, diversos exemplos de heurísticas criadas por pesquisadores ao decorrer dos anos, e, dentre as características que garantem a eficiência delas, podemos destacar o desempenho satisfatório, quando a heurística consegue desenvolver uma solução perto da ótima, a rapidez, que diz respeito à velocidade do processamento, a robustez, quando pode ser aplicada para diversas complexidades de problemas e a fácil implementação, já que é de extrema necessidade que a heurística ajude a resolver um problema de maneira acessível.

Sendo assim, o objetivo do presente trabalho é utilizar as heurísticas já existentes, adaptadas ao problema de *flowshop* distribuído, que trata da produção em duas plantas ou células.

1.1 PROBLEMA

Este estudo procura avaliar, de forma experimental em laboratório, modelos de resolução para uma eficaz programação de tarefas em duas unidades fabris com a mesma capacidade de produção instalada, buscando uma otimização da linha.

1.2 JUSTIFICATIVA

Embora seja frequentemente encontrada multiplicidade de unidades fabris em empresas ao redor do planeta e suas constantes transformações, o estudo do sequenciamento de produção do tipo *flowshop* permutacional distribuído ainda é recente, tendo como pioneiros os trabalhos de Naderi e Ruiz (2010) no início do século XXI.

Partindo da necessidade crescente de otimização da linha de produção por conta de eventuais reduções de custos, alocação correta de recursos e maior qualidade do produto final, é justificável a abordagem desse trabalho em propor

uma solução de alta qualidade, a priori teórica e computacional, para os problemas de sequenciamento.

Neste âmbito, o desenvolvimento de heurísticas baseadas nas já existentes na literatura torna essencial características de otimização para o gerenciamento da sequência produtiva através de funções objetivo, neste caso o tempo total de programação (*makespan*).

1.3 OBJETIVO GERAL

O objetivo geral deste trabalho é analisar diferentes heurísticas existentes e modificadas, de fácil implementação em um ambiente de produção *flowshop* permutacional distribuído em duas unidades fabris idênticas, de mesma capacidade, e as soluções fornecidas por esses experimentos, em busca de boas soluções para a minimização do *makespan.*

1.4 OBJETIVOS ESPECÍFICOS

- Implementar heurísticas já existentes na literatura;
- Adaptar essas heurísticas para o problema a ser resolvido;
- Comparar as diferentes soluções da função objetivo desejada, o *makespan.*

1.5 DELIMITAÇÃO DO TEMA

Esse estudo do sequenciamento (*scheduling)* das tarefas é pautado na busca por uma heurística suficientemente eficaz, com uma solução de alta qualidade, rápida e de fácil implementação, para poder ser aplicada em linhas reais de montagem em futuros trabalhos. Resultados estes que tem como parâmetro de validação a função objetivo *makespan*. Em tal ponto entra a área produtiva representada pelas células fabris, nesse caso, linhas de produção com capacidades idênticas.

O tema proposto dentro na programação vai além dos temas tradicionais de manufatura, onde o sequenciamento é essencial a otimização de apenas uma

única linha, neste estudo é realizado tal sequenciamento com 2 duas linhas ao mesmo tempo, o que gera um maior nível de complexidade de resolução.

2 REFERENCIAL TEÓRICO

Este capítulo tem como objetivo resumir os principais conceitos sobre o assunto a ser tratado neste trabalho, proporcionando ao leitor maior entendimento.

2.1 SISTEMAS DE PRODUÇÃO

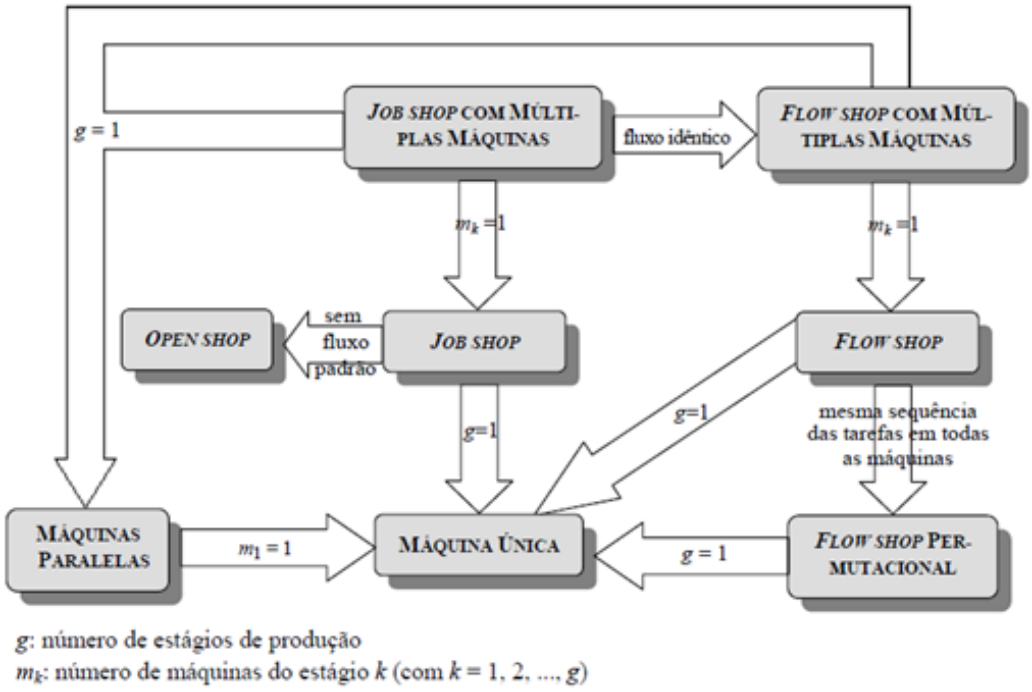
Um sistema de produção pode ser definido, segundo Harding (1981), como um ambiente formado por partes inter-relacionadas que recebem as entradas (*inputs*) e as transformam para resultar em uma saída (*output*).

De acordo com Boiko, Tsujiguchi e Varolo (2009), uma das categorias em que os sistemas de produção podem ser classificados é conforme o Subsistema de Conversão/Transformação e dentro dela temos cinco classificações, sendo:

a) Quanto à ação principal no subsistema de conversão/transformação
b) Quanto ao ambiente de produção
c) Quanto ao posicionamento do processo de produção
d) Quanto ao fluxo no subsistema de conversão/transformação
e) Quanto ao grau de contato com o consumidor

Para o estudo em questão é importante focar apenas nos sistemas classificados quanto ao posicionamento do processo de produção, que está representado de maneira simplificada na Figura 1.

Figura 1 - Relação entre os problemas de *scheduling*



Fonte: Adaptação de Moccellin e Nagano, 2003 (inicialmente apresentada por McCarthy e Liu 1993).

Assim, dentro dessa classificação, temos cinco sistemas principais (Tipos de Processos de Produção) que são explanados nos tópicos seguintes.

2.1.1 Sistemas Máquinas Únicas

O problema de máquina única consiste em uma única máquina que processa todas as tarefas em uma etapa. A sequência de processamento pode ser definida aleatoriamente ou de acordo com as necessidades da empresa, uma vez que, independente da ordem de alocação, o tempo total de processamento de todas as tarefas será o mesmo.

2.1.2 Sistemas Máquinas Paralelas

Pode-se comparar grosseiramente um sistema de máquinas paralelas com vários sistemas de máquinas únicas reunidos. O problema consiste em um

conjunto de máquinas idênticas, que processam as tarefas da mesma maneira, numa mesma etapa, sendo que cada tarefa só pode ser processada por uma única máquina. O desafio é distribuir as tarefas a serem executadas entre todas as máquinas para que o tempo total de processamento seja menor ou ainda para que se possa satisfazer alguma restrição da empresa ou do produto.

A existência de máquinas paralelas em um ou vários estágios geralmente resulta em um problema de programação mais complexo, já que além de existir um sequenciamento de tarefas, adicionalmente algumas delas precisam ser atribuídas a uma das máquinas paralelas. Pode ser vantajoso atribuir partes de um mesmo produto a várias máquinas, cada uma ainda mantendo o pressuposto da tecnologia do grupo. Nesse caso, várias configurações se tornam necessárias (NEUFELD *et al*., 2016).

2.1.3 Parâmetros para um Sistema *Flowshop*

Para melhor entendimento do processo de um sistema de *flowshop* tem-se por definições na literatura parâmetros a serem utilizados, há uma diversa gama desses parâmetros que são expressados ao longo de trabalhos com diferentes nomes, sem unanimidade, afirma Faber e Coves (2005).

Apesar disso, alguns são principais para o entendimento dos problemas de *scheduling,* dos quais os mais relevantes serão os utilizados neste trabalho, logo os que serão explicados.

Ainda na explicação desses parâmetros, pode-se encontrar uma classificação deles em $\alpha, \beta \, e \, \gamma$, onde α refere-se aos ambiente da estação de trabalho, à máquina, β refere-se às características do processo e γ ao objetivo a ser minimizado naquele problema.

Os principais parâmetros que podem ser destacados são:

- Tarefa (*Job) j:* É uma peça, uma parte de montagem ou a montagem completa, a ser processada por uma máquina, é a ação de mudança de um insumo; o conjunto de tarefas é representado por *n*;
- Máquina (*Station*) *m:* É a estação responsável fisicamente pela transformação das tarefas *j*, ou seja, dos insumos que passam pelo processo;

- Tempo de operação (*Processing Time*): É o tempo necessário para uma tarefa ser processada em uma máquina, na natureza do *flowshop* caso uma tarefa não necessite passar por uma máquina, essa é considerada com o tempo igual a 0 (zero);
- Tempo de *setup (Setup Time)*: É o tempo de limpeza e ajuste de uma máquina, de uma tarefa para a seguinte, nesse tempo a máquina não realiza nenhuma operação do processo, esse tempo normalmente é adicionado ao próprio tempo de processamento da tarefa em um problema de *flowshop,* caso que ocorre nesse estudo;
- Tempo de conclusão de tarefas (*Completion Time)*: É o tempo em que a última tarefa é processada na última máquina, finalizando o processo de produção.

2.1.4 Sistemas *Flowshop* (*FSP*)

Um problema de *flowshop* segundo Naderi e Ruiz (2010) consiste em um conjunto de *n* tarefas não relacionadas que necessitam passar por um conjunto de *m* máquinas. Essas máquinas são dispostas em uma ordem definida previamente, através de alguma definição de ordenação, e cada tarefa é processada em cada máquina nessa mesma ordem. Assim, toda tarefa é composta de uma operação em *m* máquinas. Os tempos de processamento necessários geralmente são conhecidos antecipadamente, não negativos e determinísticos.

Os *FSP* necessitam de algumas prerrogativas, como: Todas as tarefas são independentes; As tarefas estão disponíveis para serem processadas no tempo 0 (zero); As máquinas estão disponíveis todo o tempo (sem quebra); Cada máquina pode processar apenas uma tarefa por vez e em contrapartida uma tarefa também só pode ser processada por uma máquina por vez; Uma tarefa não pode ser interrompida a partir do momento em que começa a ser processada; Os tempos de *setup* são desconsiderados ou incluídos nos tempos de processamento; É possível considerar o armazenamento como infinito.

O objetivo de resolução de um problema de *flowshop* é otimizar a produção através do sequenciamento. Existem *n!* combinações entre tarefas e em cada

máquina, assim a expressão que pode representar todas as combinações de soluções possíveis é dada por $(n!)^m$.

2.1.4.1 *Flowshop* Permutacional (*PFSP*)

O *flowshop* permutacional aparece na literatura ao assumir que a mesma permutação de tarefas seja mantida em todas as máquinas, isso na prática proíbe a passagem de tarefas entre as máquinas e, consequentemente, reduz o espaço de soluções para *n!,* afirmam os autores. Isso consiste num regime em que há restrições de precedência, ou seja, uma tarefa só pode seguir para o próximo processamento ao terminar o primeiro, enfatizam Ferreira e Neto (2017). A revisão sobre o PFSP na literatura descreve que há uma suposição comum entre os estudiosos que esse modelo de problema funciona para apenas um centro de produção, ou seja, as tarefas são todas processadas na mesma fábrica.

2.1.4.2 *Flowshop* Permutacional Distribuído (*DPFSP*)

O estudo de Naderi e Ruiz (2010) apresenta outro problema de *flowshop* nas empresas: o distribuído, pois é cada vez mais comum a operação de mais de uma fábrica por uma mesma empresa por conta da maior qualidade, menores custos de produção e menores riscos de gerenciamento. Em comparação com a produção tradicional, o *scheduling* do *DPFSP* tem maior complexidade. Enquanto no *PFSP* a maior preocupação é quanto à organização otimizada de tarefas em máquinas, no *DPFSP* existem também decisões quanto ao gerenciamento e otimização entre as fábricas a serem alocadas as tarefas.

Define-se *DPFSP* como um conjunto de *n* postos de trabalho que são processados em um conjunto de *f* fábricas, as quais contêm a quantidade idêntica de máquinas *m*, onde todas as fábricas possam operar todas as tarefas e quando uma tarefa é atribuída a uma fábrica *f* essa mesma não pode ser transferida para nenhuma outra fábrica.

A expressão matemática do número de soluções de um *DPFSP* pode ser dada por $f1(n,F) = (\, n + F - 1\; F - 1\,)$, onde *n* é o número de tarefas e *F* o

número de fábricas. Essa solução permite ainda que uma das alocações de fábrica esteja vazia, entretanto, utilizando o critério de otimização $C_{máx}$, não faz sentido deixar uma fábrica vazia.

Após esse estudo pioneiro, os conseguintes envolvendo o *DPFSP* ganharam foco, aumentando, assim, o número de publicações com a introdução de um algoritmo simplificado de NEH. Logo, os pesquisadores iniciaram a construção desse modelo, que foi o pioneiro de diversos estudos envolvendo modelos matemáticos e outras heurísticas (RUIZ; PAN; NADERI; 2018).

O *DPFSP* é, em princípio, o ambiente escolhido para ser estudado para o desenvolvimento da pesquisa do trabalho de conclusão do curso.

2.1.4.3 Problema de Fluxograma de Permutação de Montagem Distribuída (*DAPFSP*)

A combinação entre o modelo *DPFSP*, anteriormente explicado, com o conceito de DMS *(Distributed Manufacturing Systems)* gerou uma um novo problema na literatura de *scheduling*, o *DAPFSP*. O conceito instituído por Gonzalez-Neira *et al.* (2017) é definido em duas etapas: (1) Num primeiro momento é determinado às *f* fábricas idênticas um modelo de *PFSP*; (2) No segundo momento, que pode ser chamado de montagem, cada produto possui *Nh* partes mescladas para ocorrer a montagem.

Dessa maneira, foi adotado que cada parte do produto pode ser realizada em cada máquina, o que significa que as partes de um mesmo produto podem ser geradas em diferentes fábricas.

2.1.5 Sistemas *JobShop*

O que difere o *jobshop* do *flowshop* é quanto ao sequenciamento de máquinas que uma tarefa deve realizar, enquanto no *flowshop* a ordem deve ser a mesma para todas as tarefas, a partir da primeira, no *jobshop* essa sequência é realizada de modo aleatório. Rajendran e Holthaus (1999) classificam o *jobshop* em *closeshop*, no qual há um número fixo de rotas a serem seguidas, e *openshop*,

no qual não há limitação dessas rotas e cada tarefa pode seguir qualquer encaminhamento.

2.1.6 Sistemas *OpenShop*

Num problema de *openshop* também existe um conjunto de *n* tarefas para serem realizadas em *m* máquinas, mas a ordem das operações de trabalho é irrelevante, afirmam Adiri e Amit (1984). Os pressupostos existentes no *flowshop* também são válidos para o *openshop*.

2.1.7 Manufatura Celular

Nos sistemas de manufatura celular, diferente dos outros sistemas apresentados até aqui, os recursos são divididos em unidades organizacionais chamadas células ou linhas de manufatura e cada uma delas produz certos conjuntos de produtos.

A principal vantagem do arranjo em células, além da minimização do tempo e custos com setup, é a simplificação do fluxo dos materiais. O controle da produção pode ser tratado dentro de cada célula de forma autônoma, já que existe a responsabilidade de uma equipe por cada conjunto limitado de peças. Como resultado tem-se uma produção mais confiável e com melhor qualidade, menores custos de retrabalho, pode alcançar tempos de produção mais curtos, menores estoques e redução do manuseio de materiais e custos de produção (NEUFELD *et al*., 2016).

A manufatura celular é vantajosa principalmente para sistemas com fluxos de materiais complexos e um alto nível de automação.

Se considerarmos um sistema com *5* tarefas a serem executadas em *3* máquinas em uma única célula ou unidade fabril, tomando como exemplo os tempos da Tabela 1, é possível perceber que a simples distribuição da produção em duas linhas, sem qualquer regra de sequenciamento ou ordenação já reduz consideravelmente o *makespan da* sequência.

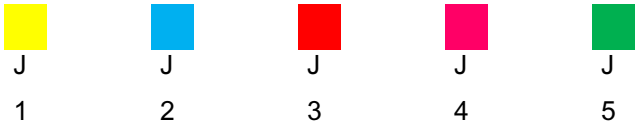
Tabela 1 – Tempos de processamento das tarefas

Máquina / Tarefa	J1	J2	J3	J4	J5
M1	5	8	1	5	2
M2	3	4	2	2	6
M3	2	5	6	1	3
Soma Tarefas	10	17	9	8	11

Fonte: Autoria própria, 2019.

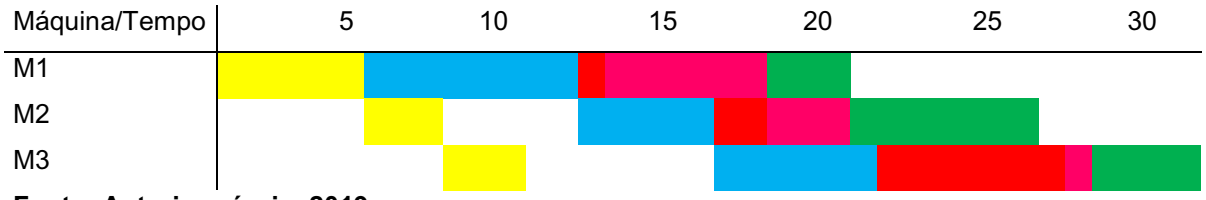
Considerando a legenda apresentada na Figura 2 para os três próximos gráficos, os resultados para *makespan* da sequência completa e para duas linhas seguem no Gráfico 1.

Figura 2 – Legenda dos Gráficos



Fonte: Autoria própria, 2019.

Gráfico 1 – Gráfico de Gantt da sequência completa

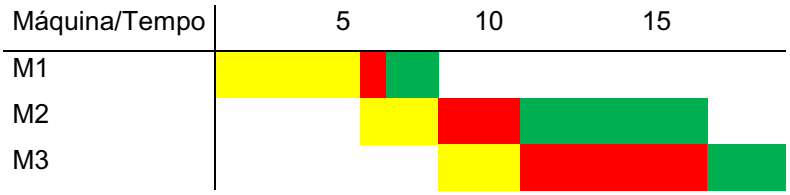


Fonte: Autoria própria, 2019.

Neste caso, utilizando um modelo centralizado em apenas uma unidade para toda a produção, o *makespan* da sequência totaliza 31u.t.

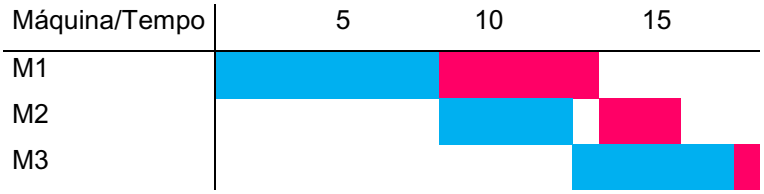
Fazendo a divisão da produção, de maneira aleatória, em duas linhas idênticas, o *makespan* reduziria para 19u.t. na linha 1 e 18u.t. na linha dois, como mostrado nos Gráficos 2 e 3.

Gráfico 2 – Gráfico de Gantt da linha 1



Fonte: Autoria própria, 2019.

Gráfico 3 – Gráfico de Gantt da linha 2



Fonte: Autoria própria, 2019.

Como *makespan* final da sequência temos o maior valor encontrado entre as linhas, 19u.t., o que representa uma redução de 12u.t. no tempo total da produção, somente fazendo a divisão da produção em duas unidades idênticas, cabendo ainda a aplicação de regras de sequenciamento para a melhora dos resultados.

2.2 PROGRAMAÇÃO DE OPERAÇÕES

A programação das operações, segundo Pinedo (2008), está relacionada com a ordenação das tarefas a serem executadas, em uma ou mais máquinas, considerando seus tempos de início e fim. Essa ordenação pode ficar complexa em função das exigências do processo a qual pertencem as tarefas e pelo grande número de combinações possíveis.

A programação das operações de modo a minimizar funções como *makespan* e *flowtime* é fundamental no planejamento de uma empresa, uma vez que muitas outras medidas de desempenho são derivadas deles, como a melhor utilização das linhas de produção, o cumprimento de prazos, a redução de atrasos, entre outras (DANG *et al*., 2018).

2.2.1 *Scheduling*

O *scheduling (*ou agendamento na tradução livre), segundo Pinedo (2008), é um processo de tomada de decisão no qual se quer otimizar a alocação de recursos e tarefas em um período de tempo determinado, otimizando os objetivos corporativos, geralmente tendendo a minimizar os tempos totais de produção. Há diferenças entre recursos e tarefas de uma empresa para outra, entretanto o *scheduling* é essencial para a otimização da produção e, consequentemente, é muito utilizado pelas empresas. A diferença do *scheduling* para um sequenciamento se deve ao fato das configurações de alocação de tarefas em máquinas serem mais complexas.

2.2.2 Gráfico de Gantt

Também conhecido como Diagrama de Gantt, o gráfico de Gantt é uma ferramenta para controlar o cronograma de programação da produção, ajudando a avaliar os prazos de término e entrega e os recursos críticos. Pode ser utilizado para acompanhar as operações programadas em cada máquina na fábrica e visualizar de maneira simples os gargalos e as máquinas ociosas da fábrica. Trata-se de uma representação visual, muito simplificada em relação à teoria original de Gantt, que permite uma rápida compreensão do início e fim de cada atividade, bem como seu andamento e suas precedentes e sucessoras (PINEDO, 2008).

A primeira versão do gráfico era conhecida como *Harmonogram* e foi desenvolvida pelo engenheiro polonês Karol Adamiecki no início do século XIX. Um século depois, o norte americano Henry Laurence Gantt (1861 – 1919) se inspirou no modelo e criou o Gráfico de Gantt conhecido atualmente. Seu objetivo era evitar atrasos na produção das fábricas americanas, auxiliando os supervisores industriais.

Atualmente o gráfico é usado amplamente no gerenciamento de projetos e na gestão da programação da produção por todo tipo de empresa e indústria.

2.3 FUNÇÕES OBJETIVO

A otimização de um processo é baseada em um objetivo a ser atingido, seja de minimização ou maximização, nesse tópico serão esclarecidas algumas das principais funções objetivo a serem utilizadas nesse trabalho.

2.3.1 *Makespan*

O *makespan* (*Maximum Completion Time*, ou Duração Total da Programação), segundo Pinedo (2008), é definido como $C_{máx}$, que equivale ao tempo de término da última tarefa na última máquina, que encerra o processo. Quanto menor esse tempo, melhor será o desempenho da máquina utilizada. Li, Dai e Zhang (2015) afirmam ainda que o *makespan* está relacionado com a melhoria da utilização das linhas de fluxo. Esta é a função objetivo escolhida para ser minimizada neste trabalho.

2.3.2 *Work-In-Progress* (WIP)

WIP é um termo que se refere ao nível de estoque de uma determinada linha, esse nível está relacionado, segundo Li, Dai e Zhang (2015), com os tempos médios de conclusão de duas máquinas adjacentes, apesar de muitas vezes ser utilizado a média de conclusão apenas da última máquina *m*. O estudo dessa função tem como objetivo minimizar o tempo médio de processamento.

2.3.3 *Flowtime*

Dang et al. (2018) definem *flowtime* (tempo de fluxo) como o “tempo total de conclusão de todos os trabalhos", que afeta os níveis de estoque WIP. Para Boiko (2008), é o tempo total desde a disponibilidade da tarefa para ser processada até o final do seu processamento, ou seja, o tempo total que a tarefa permanece numa etapa de processamento. Pode ser calculado somando-se o tempo de término de cada tarefa na última máquina do sistema.

2.4 RESTRIÇÕES DO SISTEMA PRODUTIVO

Existem diversas condições que podem restringir um modelo de programação de produção, geralmente devido à atividade desenvolvida pela empresa e pela natureza de seus produtos e serviços. Entre as restrições mais comuns em problemas de *scheduling* podemos encontrar a *no-wait*, em que uma tarefa deve sair de uma máquina e ir imediatamente para outra, sem esperas e a *no-idle*, em que não existe tempo ocioso nas máquinas e uma tarefa começa imediatamente após o término da outra.

Num sistema sem restrições, as tarefas são processadas nas máquinas, uma em seguida da outra, logo que haja disponibilidade da máquina, geralmente de acordo com um sequenciamento pré-determinado, de modo a favorecer a função objetivo escolhida pela empresa. Neste trabalho considera-se o sistema sem restrição, porém, ainda assim, os autores consideram importante apresentar as restrições *no-wait* e *no-idle* nos próximos itens, pois são encontrados facilmente em linhas de produção reais.

2.4.1 *No-wait*

Um modelo de trabalho *no-wait flowshop* significa que cada tarefa, uma vez iniciada, deve ser executada sem interrupção até que as suas operações em todas as máquinas sejam concluídas. Assim, em um sistema de n tarefas e m máquinas, uma tarefa se inicia na primeira máquina (1) e deve ser processada em todas as máquinas, sem que haja interrupção ou atrasos entre elas, até o processamento na última máquina (m), sem tempo de espera. Portanto, o processamento de uma tarefa na segunda máquina deve começar imediatamente após o término do processamento na primeira máquina, seguindo assim para todas as tarefas (de 1 a n).

Na prática, esse requisito pode surgir de certas características do produto ou da indisponibilidade de armazenamento intermediário do produto entre as máquinas.

Sapkal e Laha (2013) citam como exemplos de sistemas *no-wait flowshop* as indústrias siderúrgicas, plásticas, alimentícias, farmacêuticas, químicas, entre outras, em que não se pode haver esperas entre as etapas de processamento para que se mantenha o produto em condições apropriadas.

Na indústria metalúrgica, por exemplo, um metal precisa ser processado a uma temperatura alta do início ao fim de sua modelagem e esperas podem ocasionar o resfriamento e perda de qualidade do material. Já na indústria química, um composto volátil precisa ser envasado logo após sua produção para evitar perdas, assim como em uma indústria de refrigerantes, que precisam ser envasados rapidamente a fim de preservar a quantidade de gás carbônico em sua composição.

O sistema *no-wait* pode ser utilizado ainda por uma empresa de serviços, uma vez que longas esperas entre as etapas do serviço prestado são indesejadas pelo cliente.

Segundo Zhu, Li e Wang (2011), para garantir que a restrição *no-wait* seja respeitada, o início do processamento de uma dada tarefa numa dada máquina deve ser atrasado, quando necessário, para garantir que os processamentos nas máquinas subsequentes sejam contínuos, sem espera. Isso significa dizer que o único momento em que é aceitável existir espera é antes do início do processamento da primeira tarefa na primeira máquina (BRANCO, 2006).

2.4.2 *No-idle*

O problema de produção conhecido como *no-idle*, diferentemente do *no-wait*, não permite tempos de espera dentro das máquinas, ou seja, as máquinas devem processar todas as operações sem que haja tempo ocioso entre uma tarefa e outra, até o fim da operação.

Ele ocorre quando o tempo de preparação ou o custo do uso da máquina são altos e o processo para ligá-la e prepará-la mais vezes que o necessário se torna um desperdício. Isso significa que cada máquina, uma vez iniciada sua função, tem que processar todas as operações atribuídas a ela sem qualquer interrupção. Tal restrição é muito natural em situações da vida real em que as máquinas geralmente são equipamentos caros que devem ser alugados apenas durante o período de operações, ou seja, desde o início da primeira operação na primeira máquina até o final da última operação na última máquina.

Temos como exemplo fornos de cerâmica que consomem grande quantidade de gás natural enquanto operam e não são viáveis paradas ou

reinícios de seu funcionamento. Nesse caso, a melhor alternativa é determinar um sequenciamento que considere a restrição da não ociosidade nas máquinas.

É comum que os tempos de *setup* das máquinas sejam incluídos no tempo de processamento ou desconsiderados caso sua duração não seja relevante em relação ao tempo de processamento (BRANCO, 2011).

Para satisfazer uma restrição do tipo *no-idle*, permite-se que o início do processamento de uma tarefa na máquina subsequente não seja imediato após o término do seu processamento na máquina anterior e que este seja atrasado para que se possa garantir o processamento contínuo de todas as tarefas em uma máquina.

2.5 HEURÍSTICAS

Heurísticas são como “Regras de sequenciamento”, regras de decisão lógica que selecionam uma ordem de produção a ser executada de modo a aproximar o resultado encontrado da solução ideal, praticamente impossível (em tempo hábil) de ser encontrada.

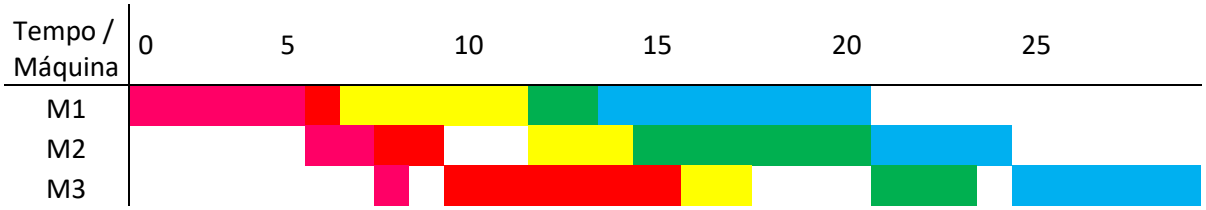
Métodos heurísticos são algoritmos exploratórios que buscam resolver problemas. Esses métodos se assemelham a uma “busca cega”, pois nem sempre tem como alvo a solução ótima, uma vez que, tendo como ponto de partida uma solução viável, baseiam-se em sucessivas aproximações direcionadas a um ponto ótimo. Logo, estes métodos costumam encontrar as melhores soluções possíveis para problemas, e não soluções exatas, perfeitas, definitivas.

Algumas heurísticas de ordenação são muito conhecidas e largamente utilizadas, principalmente no planejamento de produção, como as que seguem nos próximos tópicos. Para melhor sintetizar os processos descritos a seguir (*SPT* e *LPT*) considerou-se os tempos da Tabela 1 (página 19) para um melhor entendimento.

2.5.1 *Shortest Processing Time (SPT)*

Uma das heurísticas mais simples e conhecidas de ordenação, o *SPT* se baseia no sequenciamento das tarefas de acordo com o tempo de processamento de cada uma, sendo alocada a tarefa mais rápida primeiramente e do mesmo modo as subsequentes, em uma ordem não decrescente de tempos, afirma Pinedo (2008). Utilizando o *SPT* na alocação das tarefas da Tabela 1, obtêm-se a sequência J4/J3/J1/J5/J2, mostrada no Gráfico 4.

Gráfico 4 – Sequência SPT



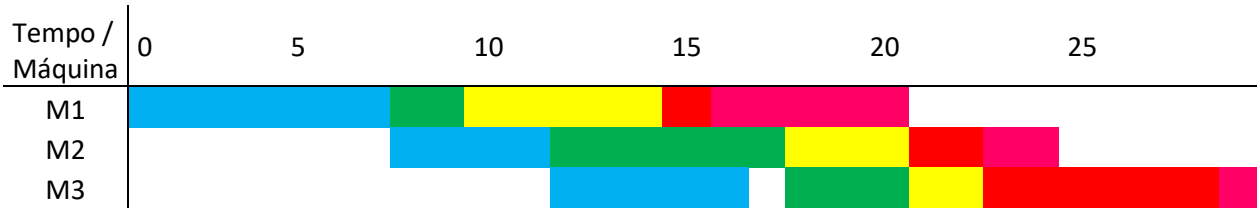
Fonte: Autoria própria, 2019.

Utilizando uma sequência que processa primeiramente as tarefas com menores tempos, o *makespan* da linha teria um total de 29u.t., um tempo menor se comparado ao sequenciamento aleatório (Gráfico 1).

2.5.2 *Longest Processing Time (LPT)*

De modo similar ao *SPT*, o *LPT* é uma heurística conhecida e simples de ordenação, mas funciona de modo contrário. Ainda segundo Pinedo (2008), o *LPT* se baseia em ordenar as tarefas a partir de tempos de processamento maiores; as tarefas mais longas são as primeiras a serem alocadas e na sequência uma ordem não crescente de tempos. Essa heurística tenta buscar um equilíbrio no processamento ao atribuir as tarefas mais curtas por último e as mais demoradas por primeiro. Utilizando o *LPT* na alocação das tarefas da Tabela 1, obtêm-se a ordem J2/J5/J1/J3/J4, como apresentado no Gráfico 5.

Gráfico 5 – Sequência LPT



Fonte: Autoria própria, 2019.

Pelo simples fato de alterar a ordem de processamento das tarefas, a sequência que aloca as tarefas mais demoradas por primeiro, neste exemplo, gera um resultado de *makespan* de 2u.t. a menos do que a sequência inicial apresentada.

2.5.3 A Regra de Johnson

Pioneiro na área de ordenação e problemas de *scheduling*, Johnson (1954) revolucionou ao publicar sobre problemas de *schedule* de uma e duas máquinas. Ele considera para a resolução de problemas uma maneira simplificada de duas etapas. Antes, o autor descreve algumas premissas para que seja viável a análise, são elas: Tempos de *setup* estão incluídos nos tempos de processamento; As etapas são sucessivas; Pode ser aplicada a qualquer processo, desde que sejam conhecidos os tempos de processamentos previamente.

Para a regra de duas etapas um conjunto de tarefas *n* deve ser executado em duas estações de trabalho no estudo de Johnson (1954), esses processos devem ser subsequentes. A lógica pode ser analisada numa tabela, na qual é encontrado o menor valor entre os tempos de processamento e em seguida é feita a análise da posição desse tempo nas máquinas; caso esse processamento esteja alocado na máquina 1, linha de cima da tabela, essa tarefa é a primeira a ser sequenciada, caso esteja na segunda linha da tabela, máquina 2, essa tarefa será a última a ser sequenciada.

Após ser alocada, uma tarefa não pode mais ser realizada novamente. Esse processo é repetido até que todas as tarefas sejam alocadas na sequência.

Já para um problema de três máquinas, Johnson (1954) propõe a concepção de uma tabela, onde é necessário satisfazer duas condições:

$$Min(Tan) \geq Max(Tbn) \; e \; Min(Tcn) \geq Max(Tbn)$$

Onde:

Tan = Tempo de processamento na estação A;

Tbn = Tempo de processamento na estação B;

Tcn = Tempo de processamento na estação C.

Ao ser aceita uma das condições acima, pode-se modelar o problema num problema de duas estações de trabalho, criando duas estações virtuais, X e Y, sendo os tempos de processamentos dessas estações as somas respectivas, $Taj + Tbj$ e $Tbj + Tcj$.

A partir da soma desses tempos pode-se resolver com a mesma metodologia de duas estações.

2.5.4 NEH

Um problema de sequenciamento tem como ponto chave a resolução quanto à ordenação das tarefas, seja a partir de prioridades ou ranqueamento. Nisso surge a necessidade de implementação de uma heurística capaz de permitir maior agilidade com maior eficiência de toda a sequência de tarefas. A partir disso Nawaz, Enscore e Ham (1983) desenvolveram um algoritmo pioneiro e determinante para a resolução de problemas de *scheduling* nomeado com as iniciais de seus nomes, o NEH.

O algoritmo procura minimizar o tempo de programação das *n* tarefas de um problema que, como já visto, tem como quantidade de solução *n!*, o que transforma uma simples ordenação em complexa e inviável para um número *n* grande. Basicamente o algoritmo proposto por esses pesquisadores busca minimizar esse tempo de processamento ao assumir que a tarefa que demanda maior tempo de processamento deve ter a maior prioridade dentro do sistema. Nessa lógica, o processo do algoritmo pode ser realizado em 3 (três) etapas:

(1) Somam-se os tempos de processamento de cada tarefa em cada máquina;

(2) Destaca-se os dois maiores tempos entre todas as tarefas, os seleciona e analisa a interação entre eles através de uma “busca exaustiva” calculando o *makespan* dessas duas diferentes sequências.

(3) A próxima tarefa com o maior tempo de processamento é escolhida e novamente é analisada a interação dela com as outras duas já selecionadas, as quais estão fixadas e não podem ser alteradas entre si, somente a nova tarefa deve ser alocada em todas as posições possíveis do sequenciamento, buscando o menor tempo total.

Esse algoritmo é muito simples e eficaz, tanto que, a partir desse estudo pioneiro, diversos outros o utilizaram como base para a implementação e o aperfeiçoamento de seus estudos, até mesmo em estudos mais recentes em pleno ano de 2019.

3 METODOLOGIA

Esse capítulo está dividido em duas seções a fim de descrever as etapas necessárias para o desenvolvimento do estudo em questão e apresentar a metodologia utilizada.

3.1 PROCEDIMENTOS

Em relação aos procedimentos, classifica-se este trabalho como pesquisa bibliográfica, pois é elaborado a partir de dados obtidos em fontes bibliográficas e estudos previamente realizados e registrados na literatura e, ainda, como experimental, pois a análise dos resultados é realizada após um processo de experimentação computacional.

Os principais conceitos apresentados neste estudo foram obtidos em artigos de periódicos encontrados em plataformas como *Science Direct*, *Scielo, Scopus*, *Web of Science*, no Portal de Periódicos da CAPES e na Revista Pesquisa Operacional da Sociedade Brasileira de Pesquisa Operacional (SOBRAPO), assim como em teses encontradas no Banco de Teses e, ainda, em livros (para os conceitos mais clássicos).

3.2 ETAPAS DO DESENVOLVIMENTO

Para realizar o estudo de forma a atingir o objetivo final, o desenvolvimento deste trabalho seguiu as cinco principais etapas apresentadas a seguir:

3.2.1 Revisão Bibliográfica

Primeiramente, foi feita a análise de livros, artigos científicos de revistas, periódicos e congressos nacionais e internacionais, teses e dissertações, relacionados ao tema proposto, com a finalidade de embasar aquilo que se apresentou no trabalho através de exemplos e estudos realizados anteriormente, encontrados na literatura.

Na literatura mais recente, é nítida a utilização da heurística NEH por apresentar maior eficiência e velocidade no processamento, resultando em boas

soluções. É um método que pode ser aplicado juntamente com outras heurísticas, como é o caso do trabalho de Ying e Lin (2018), e também no desenvolvimento de novas heurísticas modificadas da própria NEH, exemplificado nos estudos de Allahverdi, Aydilek, Aydilek em 2018 e 2020, e também nas pesquisas de Pan *et al* (2019).

O fato de que o NEH é uma heurística antiga que não cai em desuso foi um dos critérios adotado para a aplicação desta heurística no desenvolvimento do trabalho.

3.2.2 Levantamento de Dados

Para o estudo foram gerados valores aleatórios, com o software Gerador de Dados, disponibilizado pelo orientador, o qual forneceu valores de tempo aleatórios de processamento no formato de um documento de texto, a partir do valor de máquinas e tarefas que inserimos no programa, utilizando assim combinações de *n* tarefas com *m* máquinas, 100 dados para cada combinação. Os primeiros valores (1 a 100) foram gerados para três máquinas e 10 tarefas e, para os próximos dados acrescentava-se três máquinas ao processo, até que totalizassem 12 máquinas, sem mudar o número de tarefas. Então a quantidade de máquinas retornava ao valor inicial de três e o número de tarefas aumentava 10 (de 10 para 20) e repetia-se o processo. Assim seguiu-se até que os valores chegassem em 12 máquinas e 100 tarefas, como pode ser observado na Tabela 2.

Tabela 2 – Sequência de arquivos gerados.

Maquina/Tarefa	Arquivos
3x10	1 - 100
6x10	101 - 200
9x10	201 - 300
12x10	301 - 400

3x20	401 - 500
6x20	501 - 600
9x20	601 - 700
...	...
12x100	3901 - 4000

Fonte: Autoria própria, 2019.

Desta maneira, foram gerados ao todo 4000 dados aleatórios, armazenados em arquivos em formato de texto, os quais foram lidos e utilizados durante a experimentação computacional das heurísticas.

3.2.3 Implementação do Código Computacional

Após a geração dos 4000 dados, foi utilizado o *software* Dev-Pascal (versão 1.9.2) para elaboração de um código na linguagem de programação Pascal, por meio de um computador (*Windows 10 Home Single Language*, Sistema operacional 64 Bits, Processador Intel(R) Core™ i7-4510U CPU @2.00GHz 2.6GHz e memória RAM de 8,00 GB) que realizava a separação dos dados em duas linhas ou unidades fabris e a ordenação utilizando o método em estudo NEH, no sistema Clássico de produção, a fim de otimizar a função objetivo *makespan*.

Partindo do pressuposto de que o objeto de estudo envolve a otimização de duas linhas idênticas, o *makespan* foi utilizado como parâmetro para determinar a efetividade da heurística aplicada.

Assim, foram inicialmente criadas quatro heurísticas derivadas do NEH, para experimentação e comparação dos resultados. No geral, as heurísticas seguem os mesmos passos (iniciando no 1), o que muda é a ordem deles, como detalhado na Tabela 3.

Tabela 3 – Algoritmo das heurísticas

PASSO / HEURÍSTICA	A	B	C	D
Ordenação do vetor inicial por LPT	1	1	1	1

Reordenação do vetor por NEH		2		
A primeira tarefa do vetor ordenado é designada para a linha 1 e a segunda para a linha 2	2	3	2	2
A terceira tarefa é adicionada nas duas linhas	3	4	3	3
Cálculo do makespan nas duas linhas	4	5		4
Cálculo do makespan nas duas linhas utilizando o NEH			4	7
A tarefa é alocada na linha na qual, contando com ela, o makespan ficar menor	5	6	5	5
O vetor e o makespan da linha escolhida são atualizados e a tarefa é removida da outra linha	6	7	6	6
Repetem-se os passos executados com a terceira tarefa até que todas estejam alocadas em uma das duas linhas	7	8	7	8

Fonte: Autoria própria, 2019.

Como pode ser observado na Tabela 3, todas as heurísticas passam por uma ordenação inicial e a divisão de um vetor em duas linhas, cada uma a seu modo. Após a análise dos resultados dessas quatro heurísticas, foi criada uma quinta heurística (C2), semelhante à heurística C, acrescentando nela a reordenação por NEH como segundo passo da sequência.

3.2.4 Interpretação dos Resultados

Após a experimentação dos 4000 dados em cada um dos quatro cenários, os valores finais obtidos para a função objetivo *Makespan* e os seus respectivos tempos de processamento foram colocados no software *Microsoft Excel(TM)*, para

manipulação, a fim de descobrir a eficiência de cada heurística, como será apresentado em gráficos no capítulo 4. Nesta análise, foram consideradas as ferramentas estatísticas comumente utilizadas na literatura, como porcentagem de sucesso, desvio relativo médio, tempo de processamento de CPU, necessárias para obtenção de conclusões confiáveis.

4. RESULTADOS E DISCUSSÃO

4.1 INTERPRETAÇÃO DOS DADOS DAS HEURÍSTICAS A, B, C, D

Ao analisar a execução do programa computacional foi possível identificar, comparando as quatro heurísticas iniciais propostas nesse trabalho, através de gráficos, as variações de tempo de processamento, sucesso de tarefas, máquinas e classes, assim como seus desvios.

O estudo, como já dito anteriormente, tem como princípio a experimentação computacional para a comparação de heurísticas para minimização da função objetivo *makespan.*

Foi dividido a plotagem dos gráficos a partir da classe, que é o conjunto de interações entre tarefas e máquinas, exemplo 10 tarefas em 20 m quantidade de tarefas e quantidade de máquinas. Para ser considerado relevante os resultados encontrados ao longo da pesquisa devem-se observar três fatores de comparação, o sucesso, o desvio e o tempo, nesse ponto, o sucesso de uma heurística é o percentual de vezes que aquela heurística encontrou o menor valor da função objetivo naquele ponto, o desvio é o valor em unidades da diferença entre o valor da heurística naquele ponto em relação ao menor ponto encontrado, e o tempo é o valor do processamento.

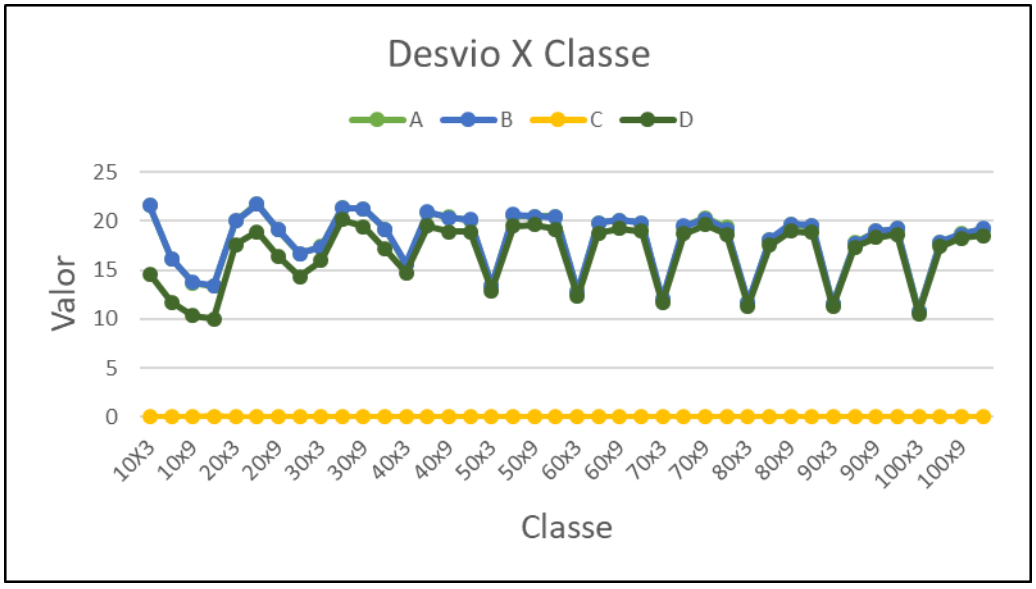
Ao observar os gráficos comparativos, é perceptível a eficiência da heurística C, na qual é utilizado o método NEH durante a escolha da alocação da tarefa em dois vetores. Isso pode ser comprovado pelos altos valores de sucesso resultantes da heurística nas classes trabalhadas, como mostra o Gráfico 6.

Gráfico 6 – Sucesso das heurísticas em cada classe

Fonte: Autoria própria, 2019.

Além do sucesso da heurística C ser nítido perante as demais, pode-se perceber no Gráfico 7 como ela é eficiente quando se aumenta a complexidade dos arquivos lidos, mantendo-se com baixos desvios mesmo com o aumento da complexidade do problema. Isso garante à heurística C um potencial de aplicabilidade em diversos problemas existentes.

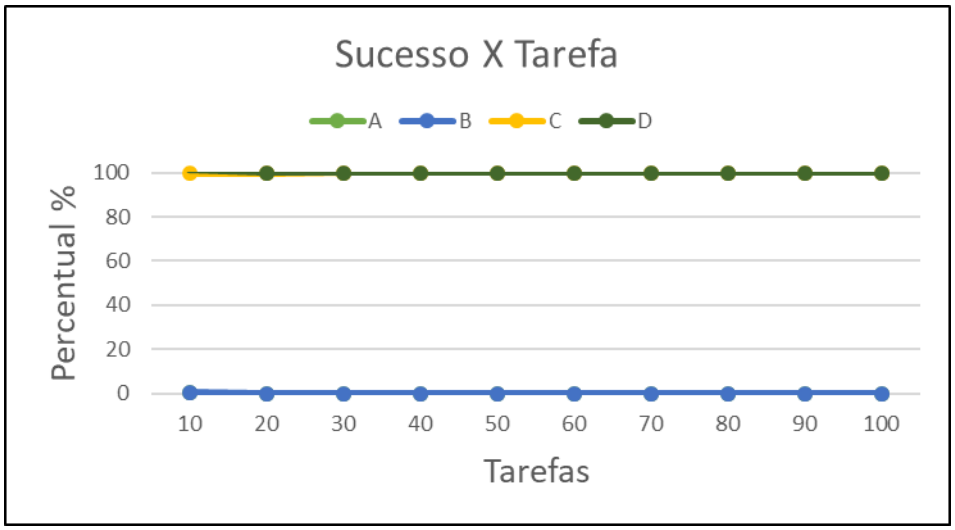
Gráfico 7 – Desvio das heurísticas em cada classe



Fonte: Autoria própria, 2019.

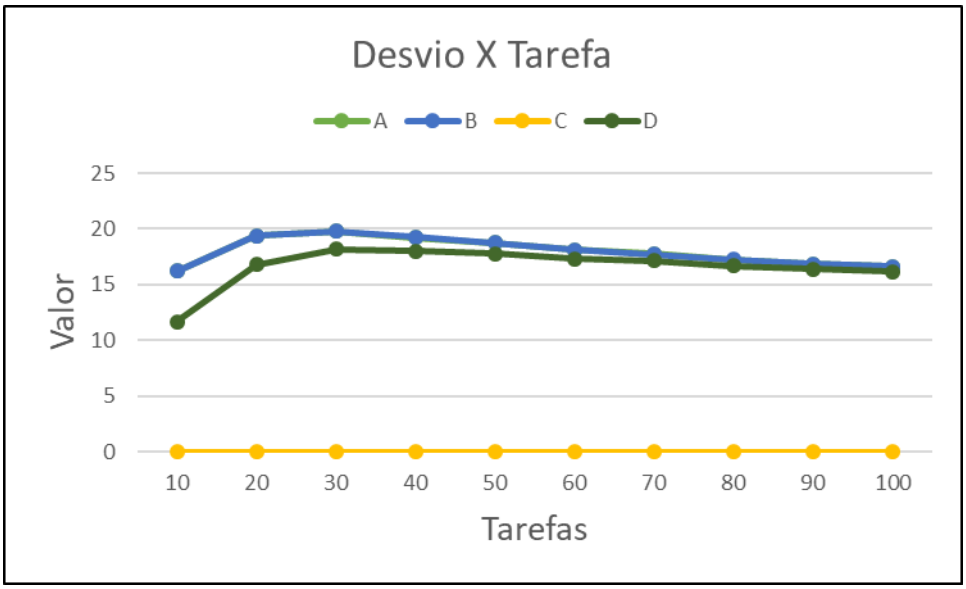
Ao observar os Gráficos 8 e 9, pode-se perceber que as heurísticas C e D apresentam sucessos semelhantes quando comparadas perante ao número de tarefas, enquanto as demais heurísticas apresentam um sucesso tendendo a zero. Apesar de esse gráfico apresentar certa competitividade entre as heurísticas C e D, o Gráfico 9 exibe um diferencial: enquanto a heurística C continua com seu desvio próximo a zero, garantindo uma efetividade com o aumento da complexidade, a heurística D não apresenta esse mesmo comportamento, mantendo uma leve curva e tendendo à estabilidade, mas não próxima de zero. Assim, essa curva não garante a eficiência da heurística com o aumento da complexidade do problema.

Gráfico 8 - Sucesso das heurísticas por número de tarefas



Fonte: Autoria própria, 2019.

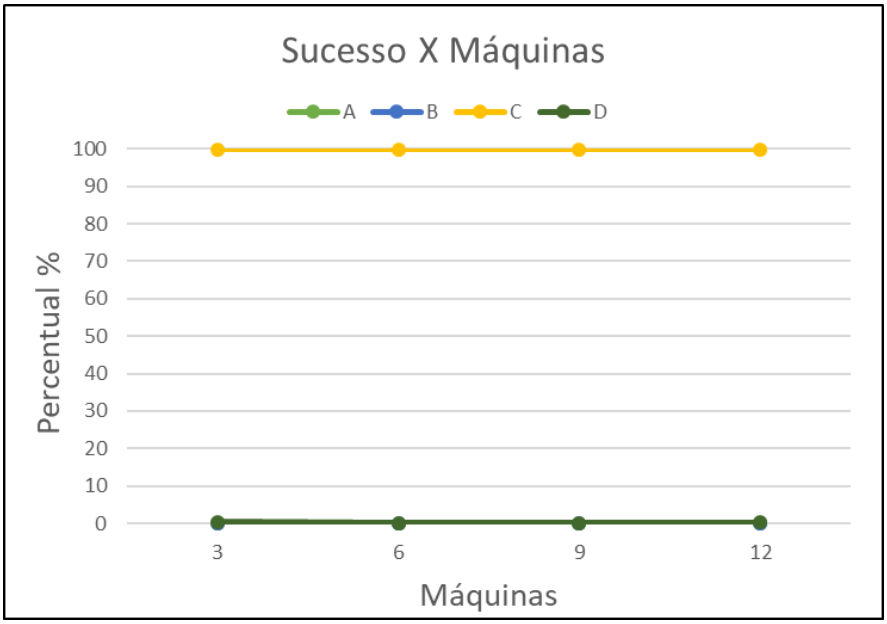
Gráfico 9 - Desvio das heurísticas por número de tarefas



Fonte: Autoria própria, 2019.

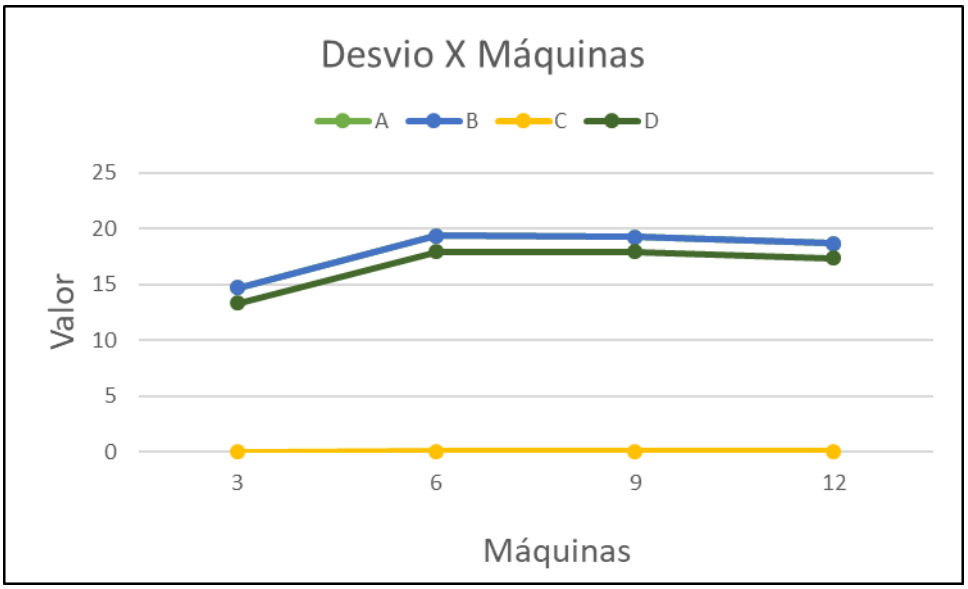
Nos Gráficos 10 e 11, comparam-se os resultados das curvas das heurísticas em relação ao número de máquinas; nesse quesito, novamente, a heurística C mostrou-se mais eficiente, com sucesso perto de 100, enquanto as demais mantiveram-se perto de 0, além do desvio da heurística C próximo a 0. Ou seja, essa heurística se comporta da mesma maneira, independente da quantidade de máquinas envolvidas no problema.

Gráfico 10 - Sucesso das heurísticas por número de máquinas



Fonte: Autoria própria, 2019.

Gráfico 11 - Sucesso das heurísticas por número de máquinas



Fonte: Autoria própria, 2019.

Neste ponto já é correto afirmar que a heurística C foi a mais eficiente, se comparada às outras três empregadas neste trabalho, quando correlacionada com o *makespan*, uma vez que o algoritmo dessa heurística apresentou resultados muito bons para resolução dos problemas criados em nível de pesquisa computacional. A aplicação do NEH durante o processo de divisão da sequência original em duas linhas otimizou a cada nova tarefa introduzida o valor final do *makespan,* o que acabou sendo um critério de escolha decisivo para minimização real da função objetivo.

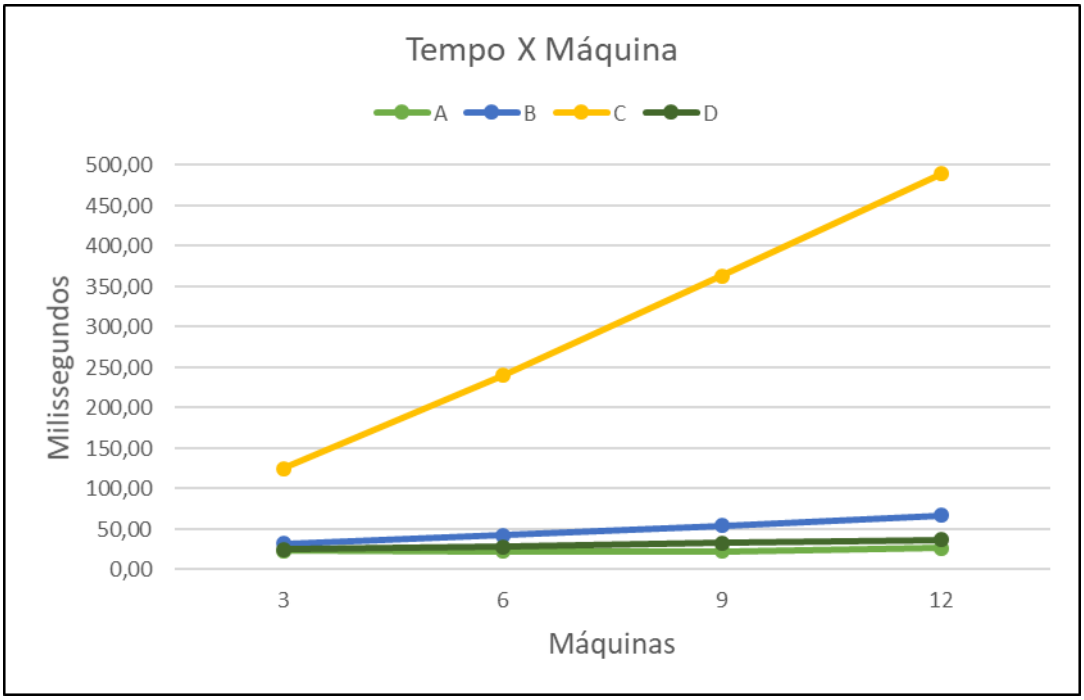
Outro ponto de extrema importância para a real eficiência da heurística proposta é o tempo de processamento do programa computacional, que não pode ser prejudicial à viabilidade da execução do algoritmo.

Após a experimentação computacional foram construídos os Gráficos 12 e 13 com a média de tempos do processamento por máquina e tarefa para cada uma das heurísticas.

O gráfico evidenciou um tempo de processamento superior da heurística C, consideravelmente superior em ambos os gráficos de tempo, crescendo tanto linearmente quando comparado em máquinas, quanto exponencialmente quando comparado em tarefas. Esses valores mostram que para a minimização do *makespan* foi necessário um maior tempo de processamento da CPU.

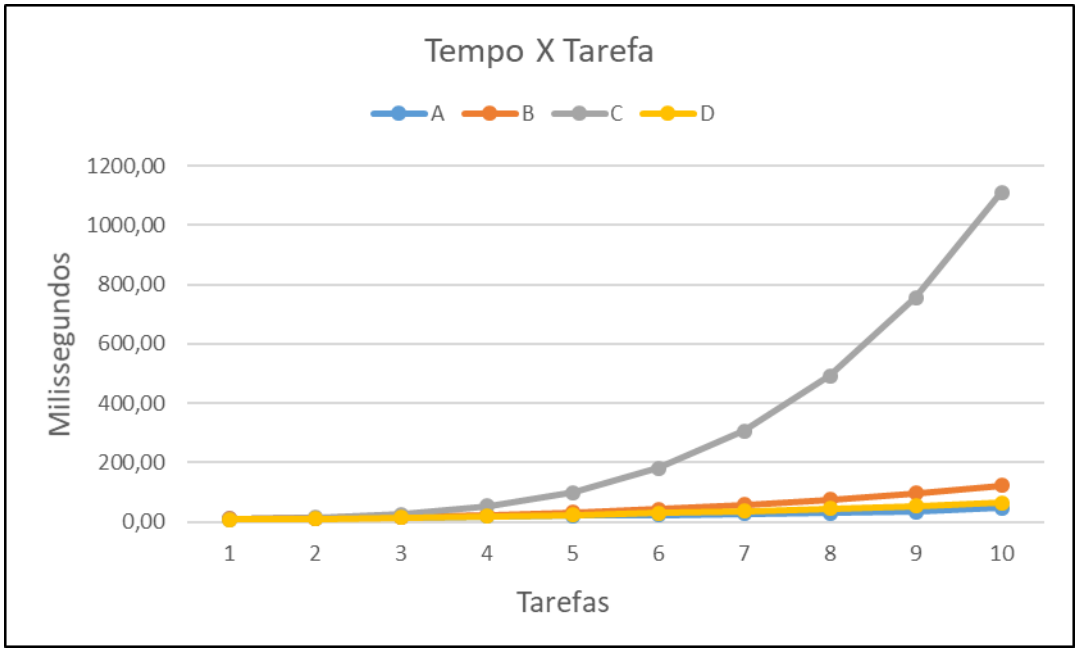
Por ser uma pesquisa acadêmica controlada em ambiente de laboratório, neste trabalho a diferença de tempo entre a heurística C e as demais não pode ser impeditiva para que se possa considerar os resultados superiores da heurística C.

Gráfico 12 – Tempo de processamento por número de máquinas



Fonte: Autoria própria, 2019.

Gráfico 13 – Tempo de processamento por número de tarefas



Fonte: Autoria própria, 2019.

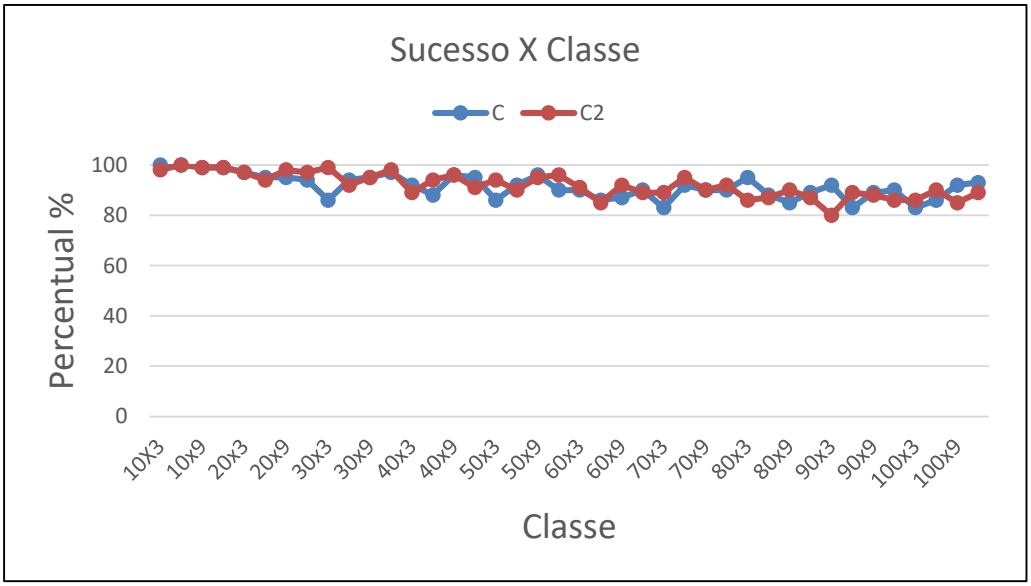
Como os tempos da heurística C, apesar de maiores que os tempos das outras três, ainda são consideravelmente pequenos e razoáveis, é possível afirmar que a heurística é muito eficiente.

Sendo assim, foi criada uma quinta heurística, C2, derivada da C, alterando somente a ordenação inicial do vetor, aplicando o NEH após a ordenação por LPT e antes de fazer a divisão em duas linhas.

4.2 INTERPRETAÇÃO DOS RESULTADOS C E C2

Tomando como base os resultados da heurística C, os gráficos apresentados a seguir foram plotados a partir da comparação desta heurística mais efetiva com um passo a mais implementado, como já comentado. Os desempenhos das duas heurísticas, por se tratarem de algoritmos muito parecidos, apresentaram certa similaridade de resultados quando comparados. O gráfico de sucesso x classe, apresentado no Gráfico 14, mostra uma oscilação entre as duas heurísticas em relação ao sucesso do programa nos problemas a partir de 20 tarefas, alternando entre picos e vales, de maneira a repetir esse ciclo e, em alguns momentos, formar duas curvas muito parecidas.

Gráfico 14 - Sucesso das heurísticas em cada classe

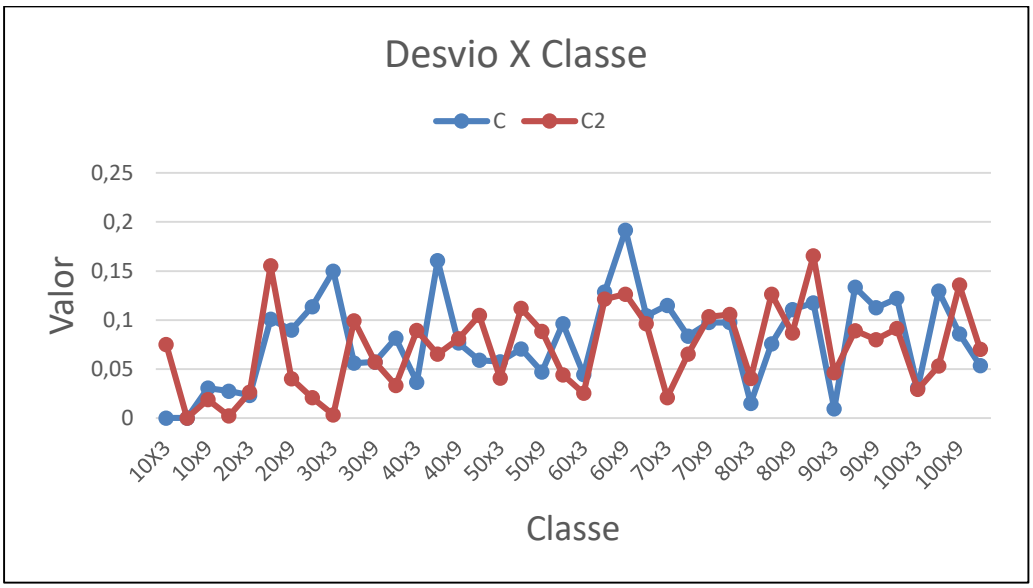


Fonte: Autoria própria, 2019.

Ao analisarmos o Gráfico 15, com os desvios das heurísticas em relação ao melhor resultado, pode-se notar novamente um padrão de picos e vales, sempre de modo alternado entre elas, demonstrando um certo grau de estabilidade em ambas as heurísticas com o aumento da complexidade computacional.

Diferentemente da experimentação anterior, é possível relacionar de uma forma errônea o desvio da mesma heurística (a C) em ambos os casos, pois os gráficos anteriores mostram um desvio dessa heurística com uma tendência a zero, fato que, ao primeiro olhar nesse gráfico, é possível dizer que não ocorre, justamente pelos picos e vales. Nesse mesmo exemplo, pode-se perceber que os valores são muito próximos a zero, das duas heurísticas, caracterizando um desvio muito baixo e valores parecidos.

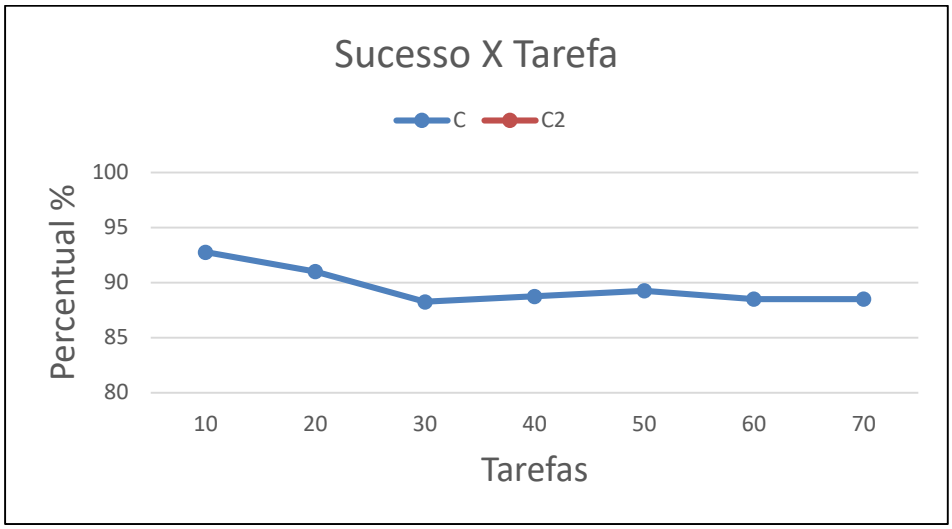
Gráfico 15 - Desvio das heurísticas em cada classe



Fonte: Autoria própria, 2019.

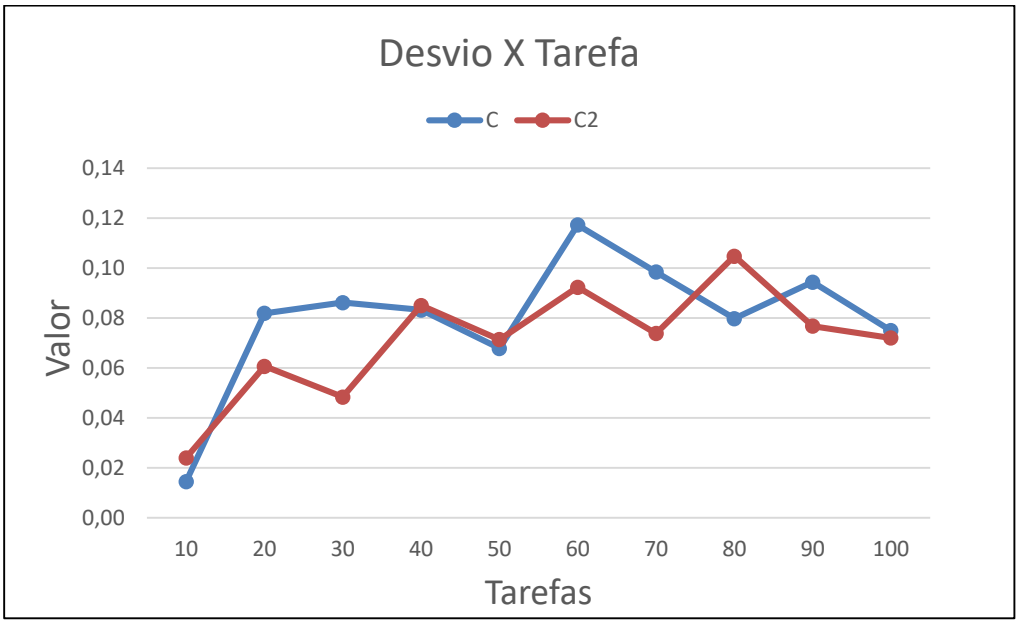
Já ao analisarmos os gráficos de sucesso e desvio por tarefa, Gráficos 16 e 17 respectivamente, é interessante a percepção que as duas curvas se sobrepõem, de modo a gerar a impressão dos resultados serem idênticos. Apesar disso, o sucesso de ambas se mostrou alto, enquanto o desvio percentual tende a zero, o que garante, assim, que o desempenho dos métodos é semelhante em termos de qualidade de solução.

Gráfico 16 - Sucesso das heurísticas por número de tarefas



Fonte: Autoria própria, 2019.

Gráfico 17 - Desvio das heurísticas por número de tarefas

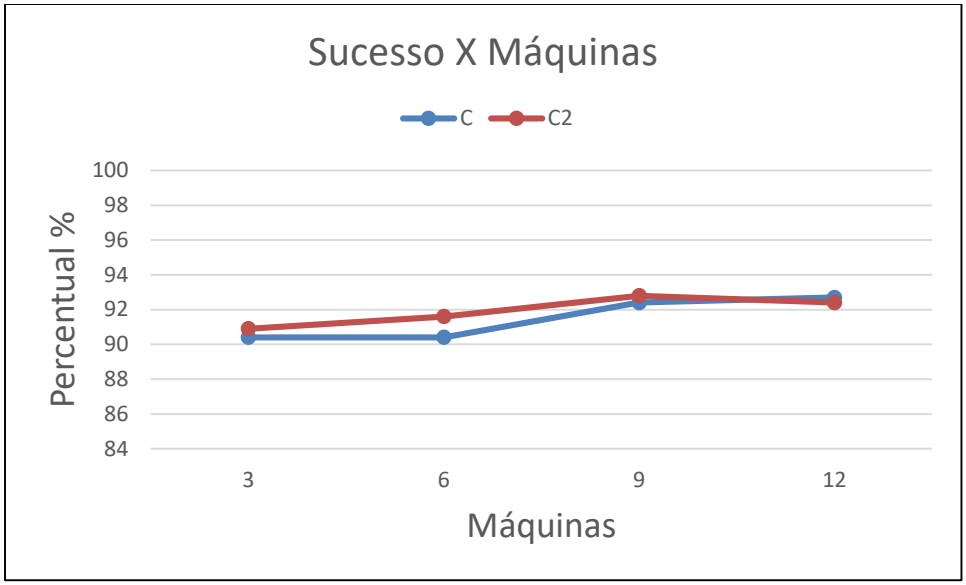


Fonte: Autoria própria, 2019.

Quando comparados os gráficos apresentados anteriormente aos Gráficos 18 e 19, é perceptível a diferença; ambas as heurísticas não apresentam uma curva quase idêntica, nota-se que a C apresenta uma leve desvantagem no sucesso em relação a C2 em um primeiro momento, fato que ao longo do aumento da complexidade tende a diminuir e as curvas tendem a se assemelhar, garantindo efetividades parecidas de ambas as heurísticas.

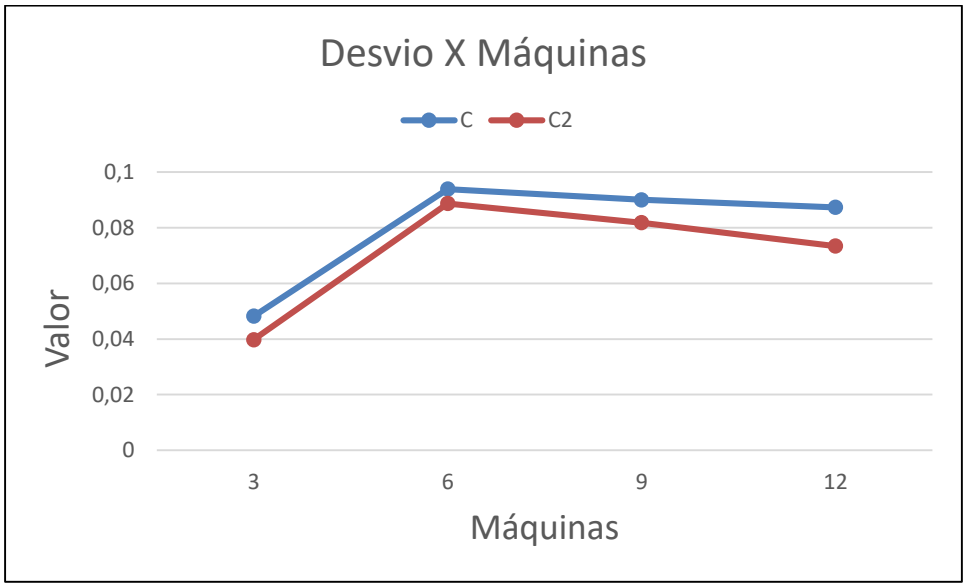
De modo análogo, no gráfico de desvio, a heurística C apresenta um desvio um pouco mais acentuado, entretanto ao observar a escala em que se encontram, o valor passa a ser quase irrisório, principalmente pelo formato da curva manter-se semelhante entre as heurísticas.

Gráfico 18 - Sucesso das heurísticas por número de máquinas



Fonte: Autoria própria, 2019.

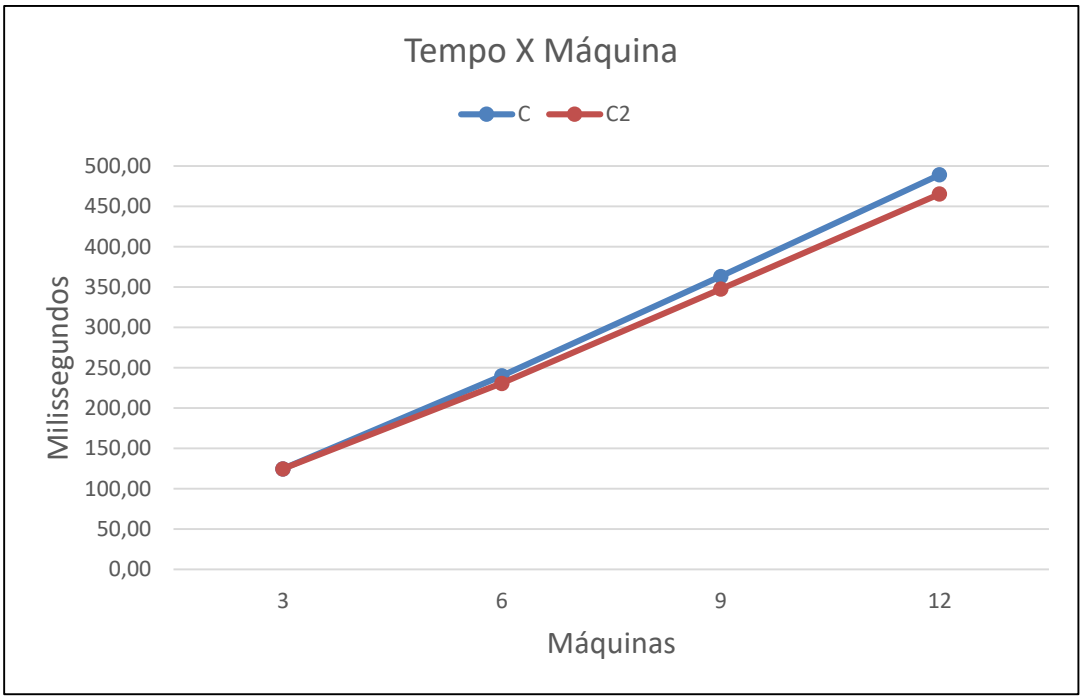
Gráfico 19 - Desvio das heurísticas por número de máquinas



Fonte: Autoria própria, 2019.

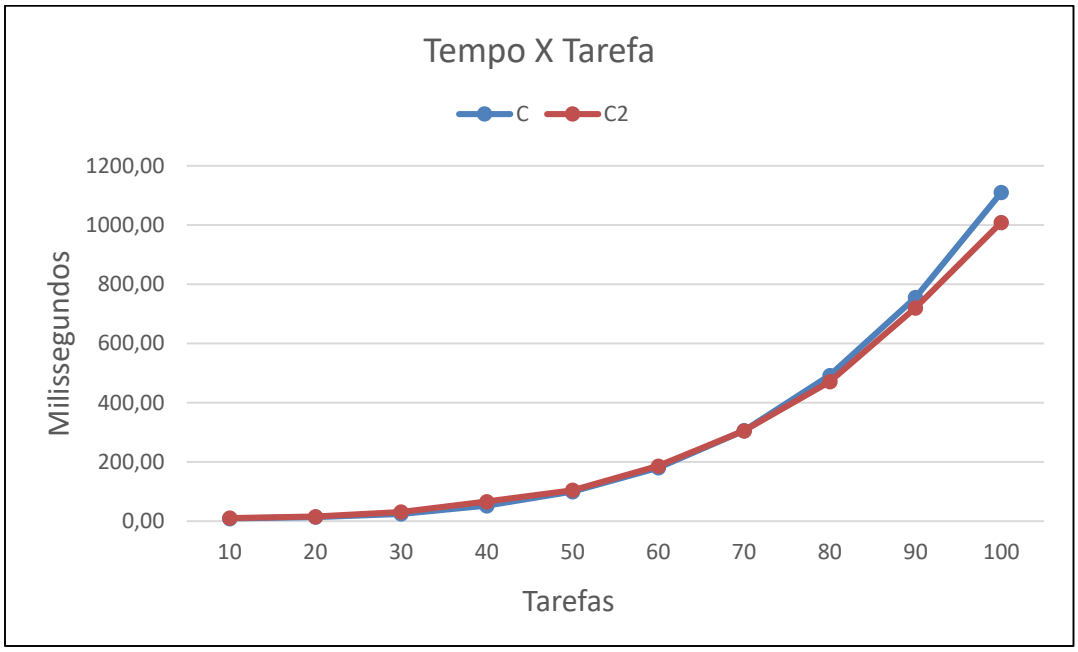
Outro ponto de importância tanto quanto na primeira experimentação é a questão do tempo de processamento de CPU dessas heurísticas. Ao longo da discussão é notável que a incrementação de mais um passo na heurística C, gerando a C2, poderia ser um fator de aumento de complexidade computacional em algum nível, por envolver mais etapas de processamento. Fato esse que, ao analisar os Gráficos 20 e 21, não se concretizou, pelo contrário, o tempo médio de execução diminuiu, o que pode ter ocorrido por uma combinação de diversos fatores que culminaram em um tempo de processamento menor no caso de C2, porém, muito próximo ao de C.

Gráfico 20 - Tempo de processamento por número de máquinas



Fonte: Autoria própria, 2019.

Gráfico 21 - Tempo de processamento por número de tarefas



Fonte: Autoria própria, 2019.

Atrelado ao desempenho quanto ao quesito tempo de CPU tem-se os valores muito semelhantes de sucesso e desvio nos dois casos, o que acarreta na viabilidade e eficiência de ambas as heurísticas.

5. CONCLUSÃO

Problemas de programação de tarefas em máquinas estudados através de heurísticas, pelo seu caráter experimental, não trazem soluções finais ótimas, porém são capazes de trazer soluções de boa qualidade com resultados satisfatórios. A busca por melhoria na implementação das heurísticas e, consequentemente, das soluções, permite uma grande variedade de estudos dentro do *scheduling*.

A implementação das heurísticas no estudo, principalmente adaptadas do NEH, para um problema *flowshop* permutacional distribuído foi motivada pela comparação e verificação de seus desempenhos em relação à função objetivo definida.

O objetivo deste trabalho foi de verificar, dentre heurísticas conhecidas na literatura juntamente com adaptações destas, qual possui o melhor desempenho na minimização do *makespan* (duração total da programação) num sistema de produção clássico.

Após a realização da experimentação computacional, para este estudo, pode-se verificar que as heurísticas aplicadas apresentaram diferentes comportamentos mesmo sendo derivadas de uma mesma heurística principal.

Ao tratar de um problema de divisão de uma sequência original em duas linhas diferentes, um cenário ainda pouco explorado nos estudos, todas as heurísticas apresentaram uma variação e melhoria significativa na minimização do *makespan* comparado a um ambiente de *flowshop* permutacional, o que já justificaria o uso de qualquer uma delas em um problema real.

A heurística C, apesar de ter maior complexidade de implementação e, portanto, maior tempo de processamento de CPU do que as outras, mostrou superioridade de eficiência quando se trata de minimização do valor do *makespan*. Assim, além de se provar eficiente em seus resultados, o impacto relacionado ao tempo de processamento, de um cenário controlado de laboratório, pode ser considerado como pequeno se comparado aos ganhos proporcionados pela otimização da função objetivo.

Os resultados obtidos neste trabalho abrem lacunas importantes para estudos futuros, com a implementação de novas heurísticas e também adaptações

de outras, adequando-as ao problema em questão e verificando seu desempenho para a otimização destas e outras funções objetivo, além do fato de poderem ser aplicadas em cenários reais com diversos parâmetros variáveis passíveis de manipulação.

REFERÊNCIAS

ADIRI, I.; AMIT, N. **Openshop and flowshop scheduling to minimize sum of completion times**. In: Computers & Operations Research. p 275-284. 1984.

ALLAHVERDI, Ali.; AYDILEK, Harun; AYDILEK, Asiye. **No-wait flowshop scheduling problem with two criteria; total tardiness and makespan**. In: European Journal of Operational Research, v. 269, p 590-601. 2018.

ALLAHVERDI, Ali.; AYDILEK, Harun; AYDILEK, Asiye. **No-wait flowshop scheduling problem with separate setup times to minimize total tardiness subject to makespan.** In: Applied Mathematics and Computation, v. 365. 2020.

BOIKO, Thays J. P.; TSUJIGUCHI, Lucas T. A.; VAROLO, Fernando W. R. **Classificação de sistemas de produção: uma abordagem de Engenharia de Produção.** Encontro de Produção Científica e Tecnológica, Campo Mourão, 2009.

BOIKO, Thays J. P. **Métodos heurísticos para a programação em Flow Shop Permutacional com tempos de setup separados dos tempos de processamento e independentes da sequência de tarefas.** 2008. 207 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2008.

BRANCO, Fábio J. C. **Avaliação de métodos heurísticos para o problema no-wait flowshop com o critério de minimização da duração total da programação.** 2006. 456 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2006.

BRANCO, Fábio J. C. **Um novo método heurístico construtivo de alto desempenho para o problema no-idle flow shop.** 2011. 112 f. Tese (Doutorado) - Curso de Engenharia de Produção, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2011.

CAMPBELL, H. G.; DUDEK, R. A.; SMITH, M.L. **A heuristic algorithm for the *n* job, *m* machine sequencing problem.** In: Management Science. 1970.

DANG, Feidi; LI, Wei; YE, Honghan. **An efficient constructive heuristic to balance trade-offs between makespan and flowtime in permutation flow shop scheduling.** In: 46th SME North American Manufacturing Research Conference, NAMRC 46, Texas, USA, 2018.

EREN, T. **A bicriteria m-machine flowshop scheduling with sequenc-e-dependent setup times.** In: Applied Mathematical Modelling. Elsevier, p 284-293. 2010.

FARBER, G, COVES, A. **Overview on sequencing in mixed model flowshop production line with static and dynamic context**. Universitat Politècnica de Catalunya. 2005.

FERREIRA, G. S.; NETO, F. R. T. **Duas abordagens para a minimização do makespan em um sistema integrado flowshop - VRP.** XLIX Simpósio Brasileiro de Pesquisa Operacional. Blumenau. 2017.

GIL, Antonio Carlos. **Métodos e Técnicas de Pesquisa Social.** 6. ed. São Paulo: Atlas, 2008. p.199.

GONZALEZ-NEIRA, E. M. **A biased-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastics processing times**. In: Simulation Modelling Practice and Theory. Elsevier, p 23-36. 2017.

HARDING, H. A. **Administração da produção**. São Paulo: Atlas, 1981. 207 p

JOHNSON, S. M. **Optimal two- and three-stage production schedules with setup times included**. 1954

JOHNSON, L.A.; MONTGOMERY, D.C. **Operations research in production planning: scheduling and inventory control**. Nova Iorque: Wiley, 1974.

LI, W.; DAI, H.; ZHANG, D. **The relationship between maximum completion time and total completion time in flowshop production**. In: Procedia Manufacturing. Elsevier, p 146-156. 2015.

MARCONI, Marina de Andrade; LAKATOS, EVA Maria. **Fundamentos da metodologia científica**. 5. ed. São Paulo: Atlas 2003.

MOON, C.; KIM, J.; HUR, S. **Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain.** In: Computers & Industrial Engineering, 2002.

NADERI, B.; RUIZ, Rubén. **The distributed permutation flowshop scheduling problem**. In: Computers & Operations Research. Elsevier, p 754-768. 2010.

NAGANO, Marcelo S.; JANUÁRIO, João Carlos S. S. **Evolutionary heuristic for makespan minimization in no-idle flow shop production systems**. In: Acta Scientiarum Technology, 2013.

NAWAZ, M.; ENSCORE, E. JR; HAM, I. **A heuristic Algorithm for the m-Machine n-Job Flow-Shop Sequencing Problem**. In: Omega. Elsevier, p 91-95. 1983.

NEUFELD, Janis S.; GUPTA, Jatinder N.D.; BUSCHER, Udo. **A comprehensive review of flowshop group scheduling literature.** In: Computers & Operations Research 70, p. 56–74, 2016.

PAN, Quan-Ke *et al*. **Effective Heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem.** In: Expert Systems with Applications, p. 309–324, 2019.

PINEDO, M. L. **Scheduling: Theory, Algorithms and Systems**. Nova Iorque: Prentice Hall. 2008.

RAJENDRAN, C.; HOLTHAUS, O. **A comparative study of dispatching rules in dynamic flowshops and jobshops**. In: European Journal of Operational Research. Elsevier, p 156-170. 1999.

RUIZ, R.; PAN, Q.; NADERI, B. **Iterated greedy methods for distributed permutation flowshop scheduling problem**. In: Omega. Elsevier, p 1-10. 2018.

RUIZ, R.; STUTZLE, T. **A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem**. In: European Journal of Operational Research. Elsevier, 2007.

SAPKAL, Sagar U.; LAHA, Dipak. **A heuristic for no-wait flow shop scheduling**. International Journal Of Advanced Manufacturing Technology, Londres: Springer-Verlag, v. 38, p.1327-1338, 2013.

TAILLARD, E. (1990). **Some efficient heuristic methods for the flow shop sequencing problem.** European Journal of Operational Research, 47:67–74
THORNTON, H. W.; HUNSUCKER, J. L. **A new heuristic for minimal makespan in flow shops with multiple processors and no intermediate storage**. In: European Journal of Operational Research. Elsevier, 96-114. 2004.

VERGARA, Sylvia C. **Projetos e Relatórios de Pesquisa em Administração**. 2ª ed. São Paulo: Atlas, 1998.

WANG, Ji-Bo; XIA, Zun-Quan. **No-wait or no-idle permutation flowshop scheduling with dominating machines.** In: J. Appl. Math. & Computing Vol. 17(2005), No. 1 - 2, pp. 419 – 432.

YING, Kuo-Ching; LIN, Shih-Wei. **Minimizing makespan for no-wait flowshop scheduling problems with setup times.** In: Computers & Industrial Engineering Vol. 121(2018), pp. 73 – 81.

ZHU, Xia; LI, Xiaoping; WANG, Qian. **An evolutionary algorithm for no-wait flowshop problems with flowtime minimization**. In: 15th International Conference on Computer Supported Cooperative Work in Design, 2011, Lausanne. 2011. p. 285 - 290.