UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE ENGENHARIA DE PRODUÇÃO

MARIA PAULA MOREIRA

AVALIAÇÃO DE MÉTODOS HEURÍSTICOS CONSTRUTIVOS EM AMBIENTE *FLOW SHOP* PARA A MINIMIZAÇÃO DO ATRASO NAS ENTREGAS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2020

MARIA PAULA MOREIRA

AVALIAÇÃO DE MÉTODOS HEURÍSTICOS CONSTRUTIVOS EM AMBIENTE *FLOW SHOP* PARA A MINIMIZAÇÃO DO ATRASO NAS ENTREGAS

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

Orientador: Prof. Dr. Fábio José Ceron Branco

PONTA GROSSA

2020

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

TERMO DE APROVAÇÃO DE TCC

AVALIAÇÃO DE MÉTODOS HEURÍSTICOS CONSTRUTIVOS EM AMBIENTE *FLOW SHOP* PARA A MINIMIZAÇÃO DO ATRASO NAS ENTREGAS

por

Maria Paula Moreira

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 3 de agosto de 2020 como requisito parcial para a obtenção do título de Bacharel em Engenharia de Produção. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

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

_____
Prof. Dr. Juan Carlos Garcia
Membro titular

_____
Prof. Dra. Daiane Maria De Genaro Chiroli
Membro titular

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

AGRADECIMENTOS

Agradeço à minha família e amigos por toda paciência, apoio e suporte durante minha trajetória acadêmica.

Ao professor orientador pela oportunidade e auxílio durante a elaboração desse trabalho.

E a todos que de alguma forma contribuíram tanto para elaboração desse trabalho como para minha vida acadêmica no geral.

RESUMO

MOREIRA, Maria Paula. Avaliação de métodos heurísticos construtivos em ambiente *flow shop* para a minimização do atraso nas entregas. 2020. 54 f. Trabalho de Conclusão de Curso Bacharelado em Engenharia de Produção - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2020.

Processos produtivos de qualquer natureza necessitam de um planejamento para se obter uma programação de produção eficaz, visto que com isso as empresas conseguem diminuir custos, melhorar sua produtividade, e aumentar sua confiabilidade de entrega. E uma das técnicas que auxiliam as organizações a definir o sequenciamento da sua produção são os métodos heurísticos. Sendo assim, o presente trabalho tem como objetivo analisar o comportamento desses métodos para a função objetivo atraso de entregas, com intuito de definir quais têm melhor desempenho no problema de *flow shop* no ambiente *no-wait*. Utilizou-se o *software* Gerador de Dados para gerar 5mil matrizes, e assim, avaliar o desempenho dos métodos heurísticos construtivos, através de um código de programação em Pascal, para os problemas propostos, obtendo-se 60mil soluções, onde a heurística SPT e suas variações apresentaram bons resultados com relação a porcentagem de sucesso e desvios relativos.

Palavras-chave*:* *Scheduling*. Heurística. *Flowshop*. *No-wait.*

ABSTRACT

MOREIRA, Maria Paula. Evaluation of constructive heuristic methods in flow shop environment to minimize delivery delays. 2020. 54 p. Trabalho de Conclusão de Curso Bacharelado em Engenharia de Produção - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2020.

Productive processes of any nature need planning to obtain an effective production schedule, since with this the companies are able to reduce costs, improve their productivity, and increase their delivery reliability. And one of the techniques that help organizations to define the sequencing of their production are heuristic methods. Therefore, the present work aims to analyse the behaviour of these methods for the objective function delay delivery, in order to define which, perform better in the flow shop problem in the no-wait environment. The Data Generator software was used to generate 5 thousand matrices, and thus, evaluate the performance of the constructive heuristic methods, through a Pascal programming code, for the proposed problems, obtaining 60 thousand solutions, where the SPT heuristic and its variations showed good results in relation to the percentage of success and relative deviations.

Keywords: Scheduling. Heuristic. Flowshop. No-wait.

LISTA DE GRÁFICOS

SUMÁRIO

1 INTRODUÇÃO

Com o mercado cada vez mais dinâmico e competitivo, com linhas de produção menos manuais, os problemas de programação de produção ganham mais espaço, uma vez que as empresas buscam minimizar seus custos de estoque, aumentar confiabilidade e produtividade das linhas de produção de maneira eficiente, de maneira geral buscam estruturar melhor sua produção.

Todos os produtos ou serviços passam por um processo, e por mais simples que seja, é importante decidir como e quando cada atividade será realizada. Devido ao aumento crescente da demanda e com intuito de ganhar vantagem sobre seus concorrentes, uma das alternativas é a aplicação de heurísticas para aumentar a eficiência da organização. Conforme aumenta essa demanda de produtos e a complexidade das atividades para o produto ou serviço final ser entregue, se torna cada vez mais difícil alocar essas atividades de maneira eficiente, uma vez que o número de possibilidades de solução também aumenta devido ao aumento do número de variáveis envolvidas.

Magatão, Arruda e Júnior (2008) descrevem a diminuição de custos de produção e melhoria da eficiência, tanto em produtos como em serviços, como um objetivo comum a diversas indústrias e setores. De acordo com os mesmos, a tomada de decisão operacional apresenta critérios experimentais que não trabalham de fato com a capacidade produtiva máxima do sistema, gerando a necessidade, no setor industrial, da criação de técnicas de otimização para avaliar de forma criteriosa os recursos críticos com intuito que os mesmos sejam utilizados de forma eficiente.

De acordo com Bilinski et al (2016) um aliado no resultado das tomadas de decisão dentro das organizações é o processo de modelagem, uma vez que ele auxilia os gestores na tomada de decisão para um melhor dimensionamento dos recursos da organização e otimização de seu processo.

Com isso, problemas de programação de produção vêm sendo estudados cada vez mais, na busca de melhores alternativas e métodos de solução que buscam otimizar as funções objetivo desejadas, as quais podem variar de acordo com o sistema produtivo, para melhor utilizar os recursos disponíveis. O sequenciamento da produção define a ordem em que as tarefas serão executadas em cada máquina ou outro recurso pelo qual a mesma precisa passar, e está diretamente ligada a tomada de decisão operacional.

O processo de sequenciamento de tarefas é denominado *scheduling* e o mesmo determina os meios de processamento a serem utilizados e o momento em que cada atividade começa e termina. Para realização do mesmo em sistemas de produção dois métodos são muitos utilizados, sendo eles os métodos exatos e os métodos heurísticos. Os métodos exatos são baseados em modelagem matemática, enquanto os métodos heurísticos consistem em estudos e conhecimentos já realizados sobre o problema proposto, que permitem a construção de modelos computacionais para resolução de problemas. Nesse trabalho são estudados os métodos heurísticos.

1.1 PROBLEMA DE PESQUISA

A finalidade desse trabalho é comparar métodos heurísticos de sequenciamento de tarefas, em problemas de *flow shop,* para responder a seguinte questão: qual método heurístico estudado apresenta melhor desempenho na diminuição no atraso de entregas no sistema de produção *no wait*?

1.2 OBJETIVOS

Buscando responder o problema de pesquisa, o presente estudo se orienta por um objetivo geral e por objetivos específicos que serão listados nessa seção.

1.2.1 Objetivo Geral

O objetivo geral desse trabalho é analisar o comportamento de métodos heurísticos com intuito de definir quais têm melhor desempenho no problema de *flow shop* proposto.

1.2.2 Objetivos Específicos

Os objetivos específicos que se pretender atingir são os seguintes:

- Adaptar os métodos heurísticos encontrados na literatura, para comparar desempenho;

- Realizar a experimentação computacional, em diferentes conjuntos de dados;
- Analisar os resultados por meio de gráficos e tabelas com ferramentas estatísticas apresentadas na literatura;
- Encontrar soluções satisfatórias para as funções objetivo estabelecidas.

A próxima seção trata da justificativa deste trabalho.

1.3 JUSTIFICATIVA

Esse trabalho está relacionado com a pesquisa operacional, que é uma das áreas de atuação da engenharia de produção, segundo a ABEPRO (Associação Brasileira de Engenharia de Produção). A pesquisa operacional busca um melhor desempenho da produção com auxílio de critérios previamente estabelecidos, envolvendo tomada de decisão por meio de modelos matemáticos habitualmente processados computacionalmente. A subárea estudada, também segundo a ABEPRO, é a de modelagem, simulação e otimização. O trabalho é voltado para otimização e tem como uma das suas principais funções encontrar uma ordem para que a sequência de tarefas seja executada nas máquinas disponíveis de maneira a otimizar o processo.

Processos produtivos de qualquer natureza necessitam de um planejamento para se obter uma programação de produção eficaz, visto que com isso as empresas conseguem diminuir custos, melhorar sua produtividade, e aumentar sua confiabilidade; assim como uma programação dos processos inadequada pode gerar alguns problemas, tais como atrasos de entregas e estoques desnecessários.

Com o intuito de evitar esses problemas e para que essa programação seja realizada de maneira mais precisa, buscando sempre o melhor desempenho produtivo das empresas, se faz necessário o conhecimento e o desenvolvimento de alguns métodos de solução, sendo que uma das alternativas para resolução é a aplicação de heurísticas de sequenciamento de tarefas que se adéquem ao sistema produtivo da empresa, aos seus recursos disponíveis e a função objetivo que a mesma deseja otimizar.

O estudo desse tema é fundamental para busca de melhores métodos de solução para esses problemas, uma vez que o mercado está cada vez mais competitivo, possui sistemas de produção cada vez menos manuais e obtém muita informação sobre melhoria contínua, fazendo com que as empresas desejem estruturar melhor sua produção para não dar espaço para erros de programação de produção que façam com que a mesma perca rentabilidade.

1.4 DELIMITAÇÃO DO TRABALHO

Este trabalho possui o seguinte escopo e restrições: é considerado o problema de *flow shop* no ambiente de produção com restrição *no-wait*, com intuito de diminuir os atrasos de entrega, sendo que os dados utilizados no trabalho são gerados de maneira aleatória, mas baseado em parâmetros encontrados na literatura.

2 REFERÊNCIAL TEÓRICO

Para esse estudo, é importante o conhecimento de alguns conceitos sobre *scheduling*, *flow shop*, sistemas de produção, métodos heurísticos e funções objetivo, que são utilizados posteriormente para análise e tratamento dos dados, de forma a demonstrar não apenas os conceitos, mas também o que já se tem na literatura sobre o assunto.

2.1 *SCHEDULING*

O tema de estudo do presente trabalho é *scheduling* ou agendamento que é o termo que encontramos ao traduzir literalmente, associado a programação de produção. De maneira geral o *scheduling,* conforme descreve Buxey (1989), acontece em quatro níveis no processo geral de planejamento, sendo eles: agregado (recurso) agendamento, conclusões de produto final, suporte a agendamentos de componentes e o sequenciamento preciso de tarefas em máquinas; os principais objetivos do *scheduling* devem ser alcançar eficiência, e ter uma melhor utilização de recursos.

Conforme Morais e Moccellin (2010), o exercício da programação é algo extremamente complexo no gerenciamento da produção, visto que os programadores podem lidar com diferentes tipos de recurso simultaneamente, as máquinas podem possuir diferentes capacidades, a equipe apresentará habilidades diferentes, entre outros.

Segundo Napierala (2014), mesmo com os avanços tecnológicos, o *scheduling* ainda é um problema complexo, que tem suporte em um modelo genérico, visto que suas características e particularidades variam de acordo com o caso estudado, pertinente às diferenças das instalações, processos, produtos e ao ambiente do negócio.

Pacheco e Santoro (2001) analisaram seis empresas brasileiras, com o intuito de detectar deficiências de *scheduling*, buscando entender, através de entrevistas e questionários, a limitação das mesmas para escolha de seus sistemas e concluíram que a avaliação dos métodos de resolução do *scheduling* deve conter

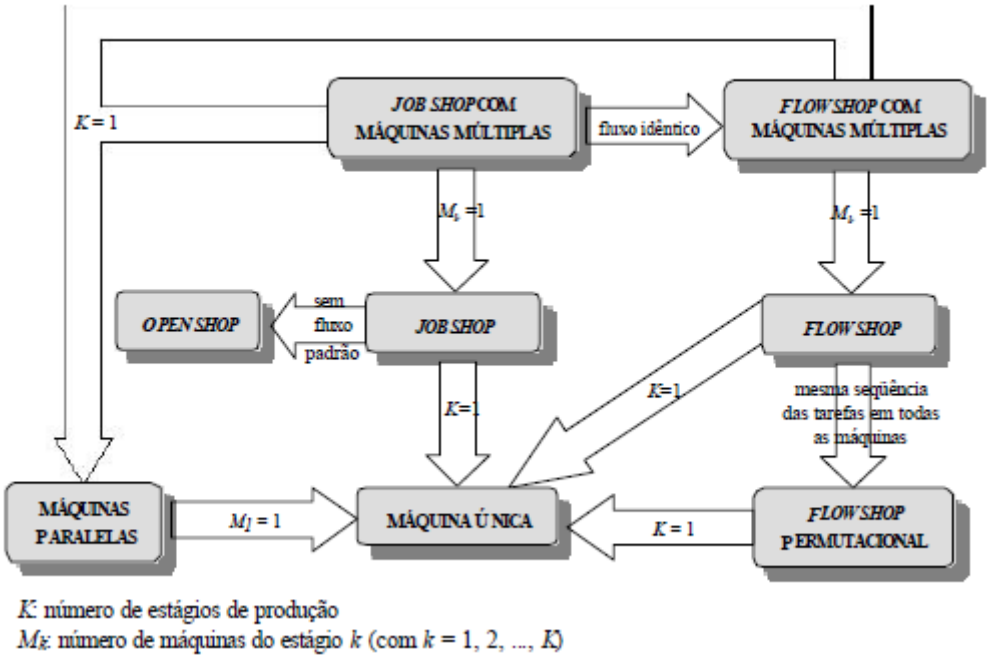
testes com diferentes massas de dados pois isso irá influenciam diretamente no resultado.

De acordo com Moccellin e Nagano (2003) o padrão do fluxo de tarefas nas máquinas é um fator que determina as restrições do *scheduling*, pois as mesmas alteram as restrições de programação das tarefas, sendo as principais classificações para os problemas de programação as seguintes:

- *Job shop*- cada tarefa irá possuir sua própria ordem de processamento nas máquinas;
- *Flow shop*- todas as tarefas possuem o mesmo fluxo de processamento nas máquinas;
- *Flow shop* permutacional- *flow shop* no qual a ordem de processamento de cada tarefa nas máquinas é a mesma;
- *Open shop*- não tem um fluxo definido para ordem de processamento das tarefas.
- Máquina única- apenas uma máquina está disponível para utilização;
- Máquinas Paralelas- mais de uma máquina, normalmente identificas, estão disponíveis para a execução de uma mesma tarefa;
- *Job shop* com múltiplas máquinas- *job shop* onde se têm um conjunto de máquinas paralelas em cada estágio de produção;
- *Flow shop* com múltiplas máquinas- *flow shop* onde se têm um conjunto de máquinas paralelas em cada estágio de produção.

A Figura 1 representa a relação entre estes problemas de programação classificados.

Figura 1- Relação entre as classes de problemas (adaptação a partir da figura apresentada por MacCarthy e Liu, 1993).



Fonte: Moccellin e Nagano, 2003.

Contudo, como padrão do fluxo de tarefas influencia diretamente no problema de *scheduling* foi estabelecido o *flow shop* permutacional como padrão para este estudo, sendo que o mesmo será revisado na próxima seção do referencial.

2.2 *FLOW SHOP*

Em um ambiente *flow shop* temos um problema de programação de produção em que *n* tarefas são processadas por um conjunto *m* de máquinas distintas com o mesmo fluxo de processamento nas máquinas. De acordo com Buzzo e Mocellin (2000), as hipóteses mais empregadas na solução de problemas de programação de operações em ambiente *flow shop* são:

- Cada máquina está disponível sem interrupções, continuamente;
- Cada máquina processa apenas uma tarefa por vez;
- Cada tarefa é processada uma máquina de cada vez;

- Os tempos de processamento das tarefas são fixos e determinados em cada uma das diversas máquinas;
- Todas as tarefas são liberadas na mesma data, para qualquer uma ser programada e executada;
- Nos tempos de processamento são incluídos os tempos de preparação das tarefas nas diversas máquinas e não dependem da sequencia de uma sequencia de operação em cada máquina;
- As tarefas uma vez iniciadas não devem ser interrompidas.

Na teoria que estuda a complexidade dos problemas de natureza combinatorial, o problema em questão é classificado como NP-*hard* (GAREY et al., 1976) sendo resolvido de maneira ótima apenas em casos de pequeno porte, aqueles que possuem até 10 tarefas, entende-se por resolver encontrar uma boa solução ou a solução ótima dentre um número finito de soluções possíveis. Sendo que um problema de programação *flow shop* Permutacional envolvendo apenas 10 tarefas apresenta 3.628.800 soluções possíveis (10!).

Arenales et al. (2007) definem o *flow shop* como um caso particular do ambiente *job shop*, onde as *n* tarefas têm o mesmo roteiro nas *m* máquinas e, quando a sequência de tarefas for a mesma em todas as máquinas têm-se um *flow shop* permutacional.

Como visto anteriormente na Figura 1, o problema em ambiente *flow shop scheduling* apresenta duas classificações, sendo permutacional ou não. A maior diferença entre eles é que no *flow shop* permutacional busca-se encontrar um único sequenciamento de processamento das tarefas para ser aplicado em todas as máquinas. Já o não permutacional busca um sequenciamento para cada máquina. E, ainda, no *flow shop* nem todas as tarefas utilizam todas as máquinas existentes, já no *flow shop* permutacional todas as tarefas demandam uma operação em cada máquina existente.

O presente estudo abordará o problema de *flow shop* permutacional, que consiste em processar sequencialmente *n* tarefas em *m* máquinas, de maneira a encontrar uma sequência que otimize determinada medida de desempenho. Uma das hipóteses que são consideradas nesse estudo é que os tempos de processamento das tarefas nas máquinas são fixos e previamente determinados.

De acordo com Branco e Fuchigami (2017), o problema de *flow shop* permutacional constitui-se em alcançar uma sequência de tarefas que otimize a

medida de desempenho analisada, e nos modelos já pré-estabelecidos de solução as medidas usuais são: *flowtime,* que minimiza o tempo de fluxo e reduz o estoque em processamento; e *makespan,* que minimiza a duração total da programação e é relacionada à utilização eficiente dos recursos produtivos.

Segundo Araújo, Arroyo e Santos (2016), os problemas de *flow shop* têm espaço não apenas na área acadêmica, mas também no mercado de trabalho, uma vez que eles representam o sistema de produção de produção das empresas.

2.3 SISTEMAS DE PRODUÇÃO

Existem diferentes ou sistemas de produção, em que cada empresa adota o que mais se adéqua ao tipo de produto fabricado e o tipo de matéria prima que é utilizada, que influenciam na resolução desse problema, como por exemplo: clássico (sem restrição de produção), com restrição *no wait* ou *no idle*.

De acordo com Nagano e Moccellin (1996), para classificar os problemas de programação deve se levar em conta os fatores internos, uma vez que irão influenciar as restrições dessa classificação, em especial o fluxo de padrão das tarefas, o número e tipos de máquinas disponíveis e a ordem de processamento das tarefas. Ou seja, os fatores internos precisam ser levados em consideração, sendo que os ambientes de produção que a empresa utiliza influenciam diretamente a ordem de processamento das tarefas.

Chen (2010) realizou um estudo analisando um sistema de produção de uma fábrica de *wafers* em que o reconhecimento do sistema de produção foi essencial para realização do trabalho.

Com o intuito de exemplificar o problema de *flow shop* nos diferentes sistemas de produção que são analisados no presente trabalho é utilizada uma matriz referência e gráficos Gantt para cada um dos sistemas, onde *Mi* são as máquina e *Ji* as tarefas a serem executadas. A Tabela 1 representa a matriz referência, e apresenta os tempos de processamentos necessários de um problema de três tarefas e quatro máquinas.

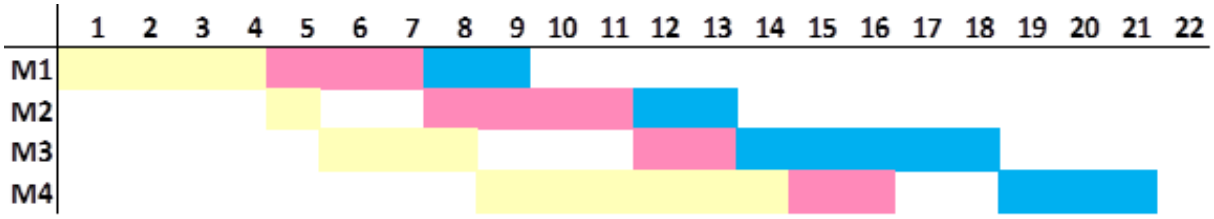
Tabela 1- Matriz Referência

	J1	J2	J3
M1	4	3	2
M2	1	4	2
M3	3	2	5
M4	6	2	3

Fonte: Autoria Própria

O primeiro apresentado é o sistema clássico que consiste nas tarefas sendo alocadas nas máquinas assim que elas estiverem disponíveis, tendo as seguintes restrições: uma máquina realizará apenas uma tarefa por vez, e uma tarefa só pode ser processada por uma máquina. Essas restrições também valem para o outro sistema que será apresentado. No Gráfico 1 o eixo vertical representa as máquinas disponíveis, o eixo horizontal representa o tempo de execução e cada cor uma tarefa a ser executada, na ordem J1/J2/J3.

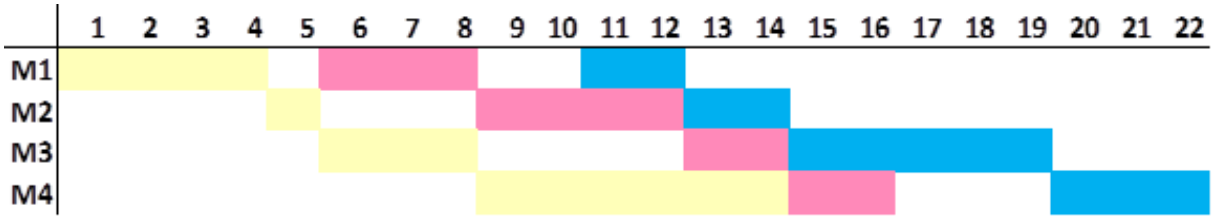
Gráfico 1- Sistema Clássico



Fonte: Autoria Própria

No sistema *no-wait*, mostrado no Gráfico 2, as restrições do modelo clássico continuam valendo, porém as tarefas não podem sofrer interrupções de processamento entre máquinas, ou seja, quando uma tarefa completa o processamento em uma máquina a mesma deve, sem cessar, ser passada para a próxima máquina, em um fluxo contínuo.

Gráfico 2- Sistema *no- wait*



Fonte: Autoria Própria

Contudo, a principal diferença entre os ambientes de produção é que o sistema clássico deve ser programado sem nenhum tipo de restrição de fluxo, o *no-wait* deve ser programado para fluxo contínuo das tarefas, o que fez aumentar o tempo total de programação de 21 para 22.

2.4 MÉTODOS HEURÍSTICOS

Nessa seção são abordados os métodos heurísticos utilizados no presente trabalho, com o intuito de entender as ordens de prioridade de cada um deles e qual o procedimento de cálculos utilizados.

Segundo Morais e Mocellin (2010), os métodos de resolução podem ser classificados em duas categorias, sendo elas: os métodos heurísticos e os métodos de solução ótima. Ambos os métodos utilizam critérios para encontrar a solução, sendo que o método de solução ótima analisa todas as possibilidades para alcança-la, enquanto os métodos heurísticos irão procurar uma boa solução entre as diversas possíveis.

Os métodos heurísticos auxiliam no encontro de soluções para o problema de *scheduling*, uma vez que o número elevado de tarefas a serem realizadas torna difícil uma solução ótima, pois as possibilidades de sequenciamento aumentam significativamente. Os tempos de execução das heurísticas devem ser aceitáveis e suas soluções de alta qualidade em relação a função objetivo analisada.

Napierala (2014) realizou um estudo da aplicação de heurísticas na resolução de um problema de programação da produção em uma empresa do setor automobilístico. E através desse estudo ele alcançou soluções aceitáveis, comprovando a vantagem de se utilizar métodos heurísticos.

Nas últimas décadas muitos métodos heurísticos vêm sendo propostos e os mesmo, podem ser classificados, de maneira geral, em dois grandes grupos: métodos heurísticos construtivos e metaheurísticos.

No presente trabalho os métodos estudados serão os métodos construtivos, que têm como característica gerar uma única solução final para o problema, e uma das formas de chegar a essa solução, que será utilizada, é por meio da ordenação das tarefas segundo índices de prioridade calculados em função dos tempos de processamento das tarefas.

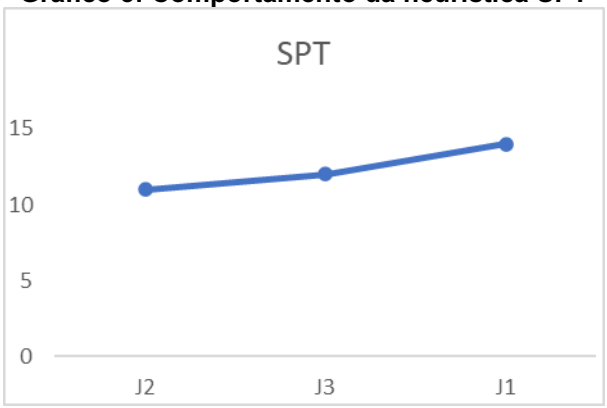
São apresentados na sequência deste trabalho as regras de ordenação *Shortest Processing Time* (SPT), *Longest Processing Time* (LPT), Triangular e Triangular Invertida, selecionados para a experimentação computacional, e apresentados no trabalho de Branco e Santos (2016).

2.4.1 SPT

A SPT é uma heurística em que a regra de prioridade é menor tempo de processamento logo, para realização desse método são somados os tempos de cada tarefa e depois ordenados em ordem não-decrescente. Voltando à Tabela 1, primeiro somaríamos as tarefas obtendo *J1*=14, *J2*=11 e *J3*=12 e a ordem SPT nesse caso seria *J2*/*J3*/*J1*.

No Gráfico 3 pode-se observar o comportamento desse método, onde o eixo vertical representa o tempo de processamento das tarefas e o eixo horizontal as tarefas conforme ordenação vista anteriormente, dos menores para os maiores valores.

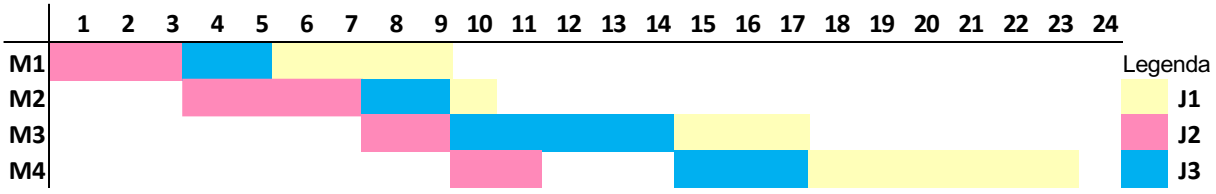
Gráfico 3: Comportamento da heurística SPT



Fonte: Autoria Própria

E a sequência de execução das tarefas pode ser observada no Gráfico 4.

Gráfico 4: Método heurístico SPT



Fonte: Autoria Própria

Esse método heurístico é utilizado quando queremos diminuir o *flowtime*, pois o método geralmente apresenta menores valores para essa função objetivo.

Nagano, Mocellin e Lorena (2005) em um estudo de redução de estoque em processamento obtiveram como resultados experimentais que o método heurístico SPT-*Flowtime* tem desempenho superior a métodos referenciados na literatura tanto com relação a porcentagem de sucesso como no desvio médio relativo.

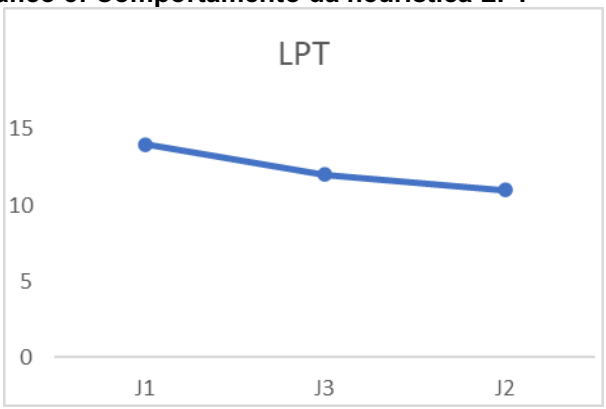
No presente estudo foram propostas de maneira experimental duas variações para o SPT, uma a partir dos prazos de entrega, em que a mesma regra de prioridade é a mesma do SPT tradicional, mas não leva em conta o tempo de processamento das tarefas.

E a segunda proposta considera a diferença o tempo de processamento das tarefas e o prazo de entrega das mesmas, sendo realizada a ordenação a partir desse ponto.

2.4.2 LPT

O procedimento para cálculo do LPT é o mesmo, porém a regra de prioridade é maior tempo de processamento, ou seja, as tarefas são ordenadas de maneira não-crescente, sendo assim tomando como base a Tabela 1 novamente, a ordem LPT é *J1*/*J3*/*J2*. O comportamento desse método, pode ser visualizado no Gráfico 5, onde o eixo vertical representa o tempo de processamento das tarefas e o eixo horizontal as tarefas conforme ordenação, nesse caso, dos maiores valores aos menores.

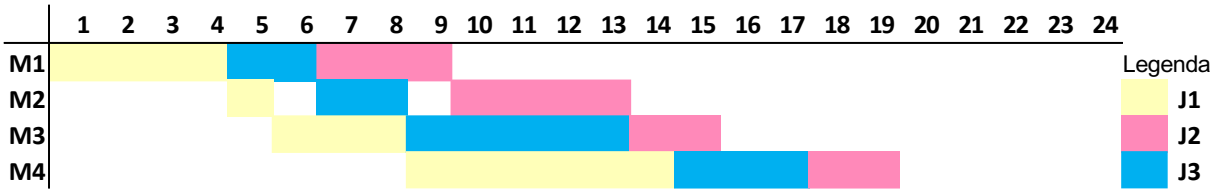
Gráfico 5: Comportamento da heurística LPT



Fonte: Autoria Própria

Já no Gráfico 6 observamos como é realizada a execução das tarefas a partir desse método.

Gráfico 6: Método heurístico LPT



Fonte: Autoria Própria

Essa regra de prioridade normalmente gera menores valores de *makespan*, sendo mais utilizada quando queremos diminuir essa função objetivo.

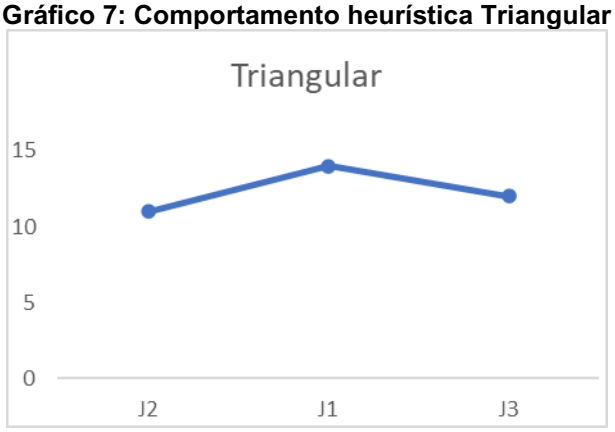
Ji, He e Cheng (2007) analisaram um problema de máquina única com manutenção periódica e função objetivo de minimização *makespan* resultando em LPT como o melhor algoritmo possível.

2.4.3 Triangular

A heurística triangular foi proposta por Branco e Santos (2016) em que a regra de prioridade é baseada no formato de um triângulo, com as extremidades menores e o centro maior, embora a soma dos tempos de processamentos seja diferente a forma ainda assim se assemelhará a um triângulo. Para a matriz referência utilizada nesse trabalho a ordenação triangular será realizada da seguinte forma: *J2*, menor valor da soma dos tempos de processamento das tarefas, sendo alocado na primeira posição, em seguida *J3*, que é o segundo menor valor, sendo alocado na última posição, e por fim, *J1*, o maior valor da soma dos tempos de processamento das tarefas, será alocado no meio, sendo a ordem final *J2*/*J1*/*J3*.

No Gráfico 7 podemos notar o comportamento dessa heurística e seu formato similar a um triângulo, onde o eixo vertical representa o tempo de processamento das tarefas e o eixo horizontal as tarefas conforme ordenação proposta.

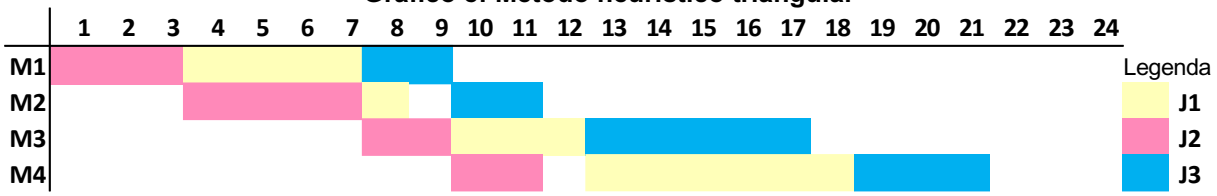
Gráfico 7: Comportamento heurística Triangular



Fonte: Autoria Própria

No Gráfico 8 têm-se a aplicação do método no problema proposto e como são executadas as tarefas a partir desse método.

Gráfico 8: Método heurístico triangular



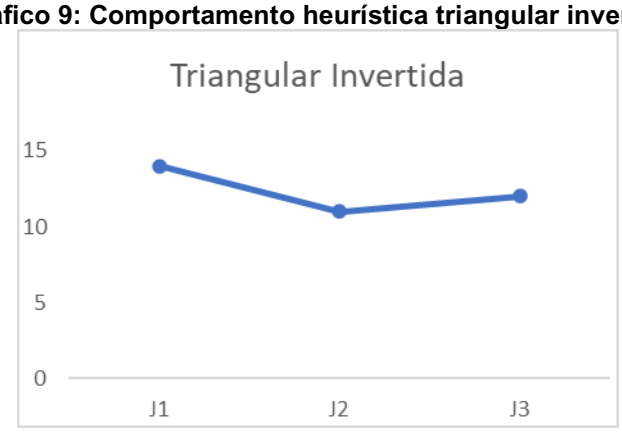
Fonte: Autoria Própria

No trabalho em que foi proposto esse método obteve bons resultados tanto para *flowtime* como para *makespan*.

2.4.4 Triangular Invertida

Essa heurística também foi proposta por Branco e Santos (2016) no mesmo trabalho em que a triangular. Para a triangular invertida a ordem de prioridade conta com os maiores valores nas extremidades e os menores no centro formando um triângulo “invertido”. Usando a matriz referência teremos a seguinte ordenação: *J1*, maior valor da soma dos tempos de processamento das tarefas, sendo alocado na primeira posição, em seguida *J3*, que é o segundo maior valor dos tempos de processamento das tarefas, sendo alocado na última posição, e por fim, *J2*, o menor valor, será alocado no meio, sendo a ordem final *J1*/*J2*/*J3*.

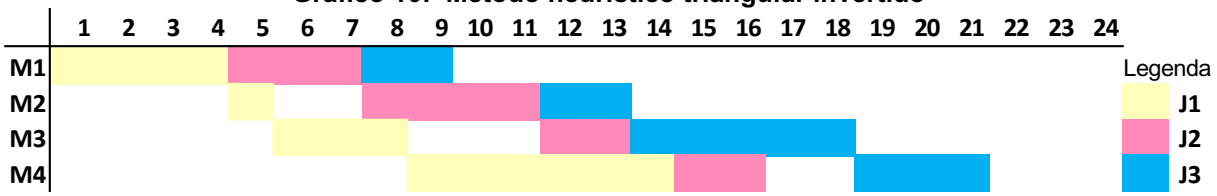
Gráfico 9: Comportamento heurística triangular invertida



Fonte: Autoria Própria

No Gráfico 9 podemos observar o comportamento da heurística e o formato do “triangulo invertido” que esse método gera, onde o eixo vertical representa o tempo de processamento das tarefas e o eixo horizontal as tarefas conforme ordenação proposta.

Gráfico 10: Método heurístico triangular invertido



Fonte: Autoria Própria

O Gráfico 10 apresenta a execução das tarefas a partir da ordenação triangular invertida. Podemos notar ainda que mesmo com poucas tarefas (3) e poucas máquinas (4) nenhuma ordenação proposta pelos métodos foi igual a outra.

2.5 FUNÇÕES OBJETIVO

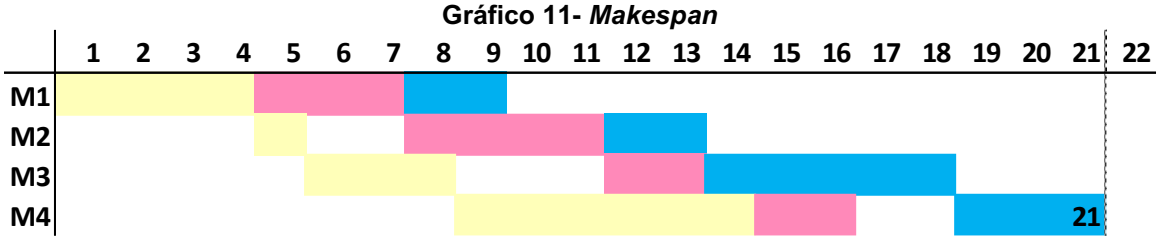
Os problemas de otimização sempre buscam maximizar ou minimizar uma função ou critério específico, denominadas funções objetivo. Essa seção aborda algumas das funções objetivo mais utilizados em problemas de *scheduling* e a função objetivo utilizada no presente trabalho, com a finalidade de entender melhor cada uma dessas funções.

Nas próximas seções são apresentadas as seguintes funções objetivo: *makespan*, *flowtime* e soma de atrasos.

2.5.1 *Makespan*

O *makespan* calcula o tempo total gasto até a última tarefa do sistema ser realizada, ou seja, é a diferença entre o começo e o final de uma sequência de tarefas. No Gráfico 11 temos um *makespan* igual a 21, que é o tempo total de programação.

Gráfico 11- *Makespan*



Fonte: Autoria Própria

O *makespan* representa a quantidade de recursos alocados a um conjunto de tarefas (GUPTA e HO, 2000). Essa função está associada com a racionalidade na utilização temporal da linha e com maior confiabilidade nos prazos.

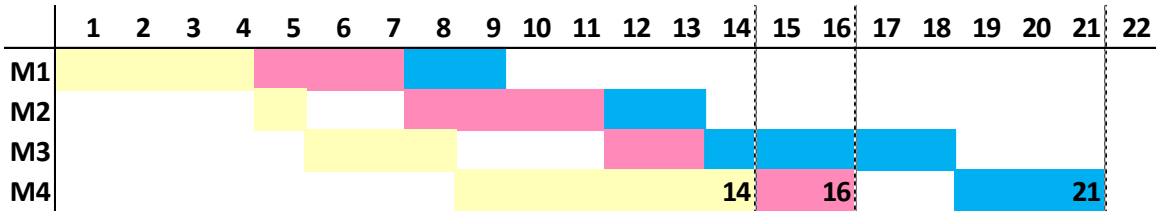
Silva, Morabito e Yamashita (2013) utilizaram essa função objetivo para programar a produção em uma indústria aeronáutica com o intuito de minimizá-la e obteve resultados positivos, e ainda percebeu que com os modelos de programação propostos era possível reduzir em 30% a mão de obra utilizada.

Teixeira e Mendes (1997) propuseram um modelo de programação horária de uma célula flexível com o uso de múltiplas ferramentas, considerando um modelo *job shop* e o critério de avaliação foi o *makespan.* O mais interessante no modelo proposto foi o fato de que quanto maior o número de ferramentas no sistema o *makespan* tendia a melhorar, mas apenas até certo limite.

2.5.2 *Flowtime*

O *flowtime* ou tempo médio de fluxo é obtido por meio da soma dos tempos de conclusão de todas as tarefas, no Gráfico 12 temos o *flowtime* igual a 51, ou seja, 14+16+21.

Gráfico 12- *Flowtime*



Fonte: Autoria Própria

Essa função objetivo está relacionada com a redução de estoque em processamento e minimização de custos.

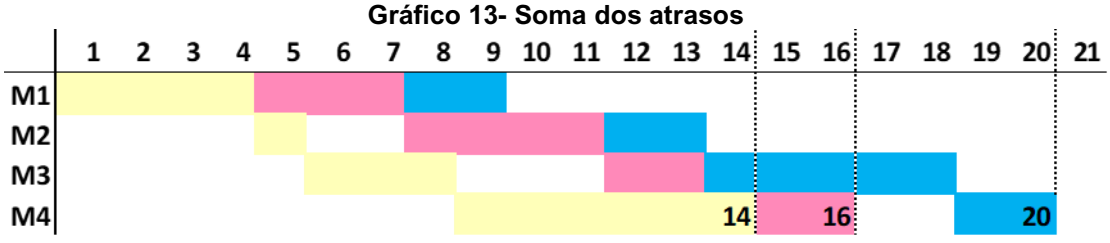
Branco e Nagano (2008) utilizaram essa função objetivo como critério de avaliação em seu estudo para o problema de *no-wait flow shop*.

Morais e Moccellin (2010) propuseram métodos heurísticos construtivos para redução de estoque através da minimização do *flowtime* e, para isso, analisaram quatro métodos distintos com o intuito de avaliar qual o melhor.

2.5.3 Soma de Atrasos

A soma de atraso está relacionada com a diminuição dos atrasos das entregas das tarefas, e é calculada por meio da diferença entre o tempo de entrega real e o prazo de entrega, sendo que essa diferença pode ser calculada tanto com relação a número de tarefas em atrasos quanto com relação a tempo de atraso.

Gráfico 13- Soma dos atrasos



Fonte: Autoria Própria

Usando o Gráfico 13 para exemplificar e adotando como prazo de entrega o tempo 15 para todas as tarefas, com relação ao tempo de atraso teríamos que a primeira tarefa entregue, em 14 não teria nenhum atraso, portanto não somaríamos nada, já a segunda como foi entregue no tempo 16 teria 1 tempo de atraso (16-15) e a terceira tarefa teria 5 tempos de atraso (20-15), logo, o atraso total nesse caso é 6 (0+1+5). Já com relação ao número de tarefas em atraso, também de acordo com o

Gráfico 13, teríamos o atraso igual a 2 (0+1+1), visto que duas tarefas foram entregues fora do prazo.

Segundo Cury (1999), as datas de entrega têm forte peso em problemas de *scheduling* para determinação de soluções viáveis, uma vez que se as datas mudam as relações de precedência podem ser alteradas.

A minimização do número de tarefas atrasadas foi abordada por Ishibuchi et al (1994) no problema em questão, após a data de entrega ser ultrapassada o nível de satisfação decrescia gradualmente e o autor encontrou uma função não linear para representar esse problema.

Kiyuzato et al (2010) também apresentaram um estudo de caso com essa função objetivo, em uma empresa do setor automotivo em que conseguiu validar a utilização heurísticas em problemas reais de programação de produção, além de propor interfaces de entrada e saída para tratamento de dados.

No presente estudo, os atrasos são abordados com relação ao tempo de atraso total e não com relação ao número de tarefas em atraso. Quando se estuda o número de tarefas em atrasos, que é uma das análises que se pode fazer, refere-se exemplificar em termos de indústrias, como a relação de confiabilidade através do número pedidos ou clientes que estão sendo entregues com atrasos e como isso afeta a empresa. Já quanto a variável tempo de atraso, pode-se analisar em relação a produtividade, quanto tempo a mais está sendo gasto para realizar as tarefas e como isso afeta a empresa.

3 METODOLOGIA

A metodologia tem como objetivo apresentar uma sequência de passos a serem realizados para se atingir o objetivo proposto no presente estudo, e de acordo com Cauchick et al (2012), isso permite a concentração de recursos e esforços em um foco específico. Não é apenas apresentar uma ordem a ser seguida, essa ordem precisa fazer sentido e estar atrelada a conceitos pertinentes ao estudo.

Esse capítulo contém os procedimentos metodológicos utilizados e suas etapas para elaboração do trabalho.

3.1 CLASSIFICAÇÃO

Este trabalho é considerado uma pesquisa aplicada de caráter quantitativo visto que o mesmo busca avaliar o desempenho de métodos heurísticos para diminuição dos atrasos de entrega de tarefas, e que pode ser utilizado para problemas específicos e reais.

Bryman (1989) descreve como principais preocupações da abordagem quantitativa a mensurabilidade; causalidade; generalização e replicação, sendo que o método de modelagem/simulação é um dos mais apropriados para conduzir esse tipo de abordagem na área de engenharia de produção.

Sobre os procedimentos técnicos o presente trabalho pode ser apontado como bibliográfico dado que o mesmo se baseia em teorias e estudos materializados e registrados na literatura. Outra possível classificação para o trabalho com relação à procedimentos técnicos é a experimental, já que de acordo com Nascimento (2016) a pesquisa experimental tem como objetivo investigar o impacto de uma ou mais variáveis independentes mediante uma variável dependente.

3.2 PROCEDIMENTOS DA PESQUISA

Para a realização do trabalho seguiu-se as seguintes etapas: revisão bibliográfica, introdução e definição dos objetivos, geração de dados, implementação dos métodos heurísticos, análise dos dados gerados, análise dos resultados obtidos, e finalmente, as conclusões.

A primeira etapa consiste na pesquisa em artigos científicos, base de dados, teses e livros sobre o tema escolhido, em que os materiais foram escolhidos com base em resumos, palavras-chave e títulos que enquadram no tema, com intuito de se desenvolver um trabalho estruturado. Adotou-se como parâmetros: o número de citações para escolha das referências; e um corte temporal com trabalhos realizados entre 2005 e 2019, porém ao decorrer da pesquisa trabalhos considerados importantes citados nos artigos analisados anteriormente foram incorporados ao referencial.

A etapa de introdução e definição dos objetivos envolve uma contextualização do tema estudado, sua importância, definição dos objetivos a serem alcançados com esse trabalho e uma delimitação do escopo do problema estudado.

A próxima etapa consiste na coleta de dados que é realizada através da geração aleatória de um banco de dados utilizando um *software* gerador para geração tanto dos dados das matrizes máquina x tarefa quanto para os prazos de entrega.

Taillard (1993) para experimentação computacional propôs a geração aleatória de 260 problemas, uma vez que os tamanhos dos problemas correspondem às dimensões reais dos problemas industriais.

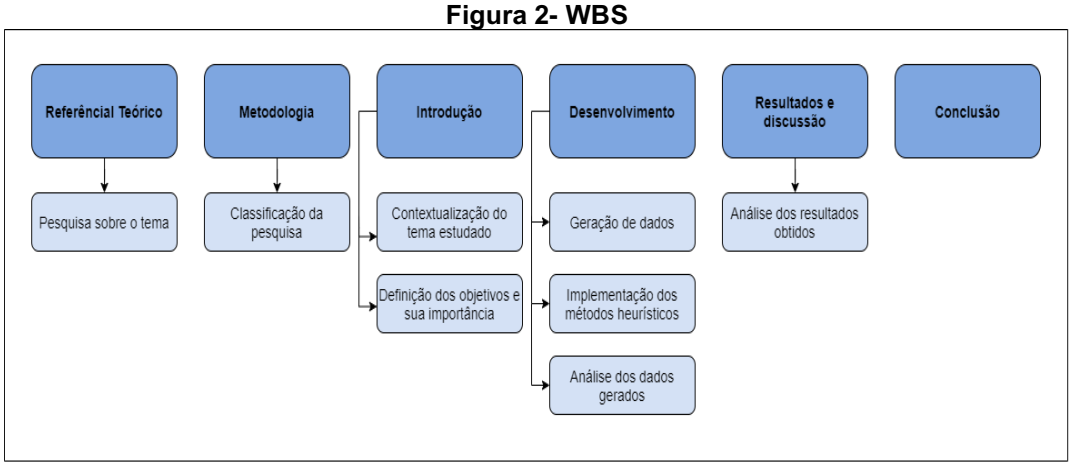
Posteriormente, na etapa de implementação dos métodos heurísticos, executou-se os códigos computacionais na linguagem de computação Pascal utilizando o *software* Dev-Pascal, com a intenção de obter soluções para a função objetivo nos diferentes métodos analisados.

Para a análise dos dados gerados é averiguada se a solução obtida é aceita ou se tem necessidade de alguma reformulação no método proposto por meio da análise do problema estudado.

Por fim, são analisados os resultados obtidos e comparados com o que já são apresentados na literatura, para assim poder concluir sobre a qualidade e viabilidade das heurísticas no problema estudado, para essa etapa utilizou-se o *software* Excel, tanto para cálculos como para ferramentas gráficas.

De acordo com os procedimentos da pesquisa descritos anteriormente realizou-se um WBS (*Work breakdown structure*), conforme a Figura 2, para identificar todas as partes que compõe o este trabalho.

Figura 2- WBS



Fonte: Autoria Própria

O WBS é uma ferramenta de projetos utilizada para subdividir o mesmo em componentes menores, para essa análise os capítulos desse trabalho foram classificados como macro atividades e as principais atividades para realização de cada capítulo as atividades menores.

Pode-se notar que as atividades que compõe a macro atividade desenvolvimento foram descritas anteriormente como etapas do procedimento da pesquisa, isso porque as mesmas demandam um maior detalhamento.

4 DESENVOLVIMENTO

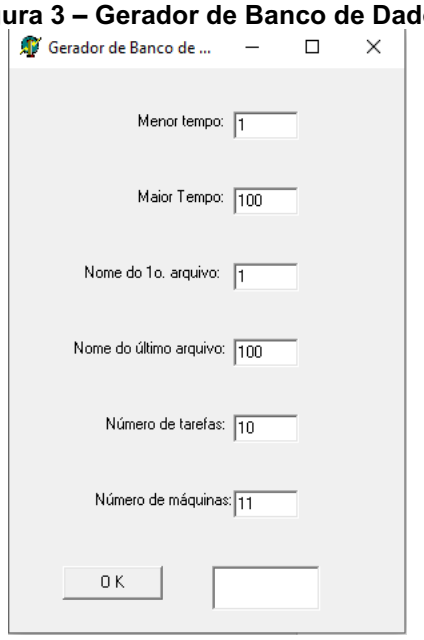
Nesse capítulo é detalhado todo o procedimento utilizado no desenvolvimento do trabalho, conforme as etapas descritas no capítulo anterior.

4.1 PROCEDIMENTO DE COLETA DE DADOS

Primeiramente criou-se um banco de dados através do *software* Gerador de Dados com 5mil matrizes e seus respectivos prazos de entrega. As matrizes foram geradas de maneira aleatória de tamanhos diferentes, onde tarefas (colunas) da matriz variaram de 10 a 100 e as máquinas (linhas) da matriz variaram de 10 a 50, ambas mudando de escala de 10 em 10. De maneira a gerar um conjunto de problemas com cinquenta classes diferentes, e cada classe contando com 100 problemas por classe.

O primeiro conjunto de matrizes gerado foi 10x10, onde a primeira matriz do conjunto foi salva no arquivo 1 e a última no 100, como visto na Figura 3 o número de máquinas para gerar a matriz foi 11 e não 10, devido ao prazo de entrega que também foi gerado com a matriz e corresponde a última linha da mesma.

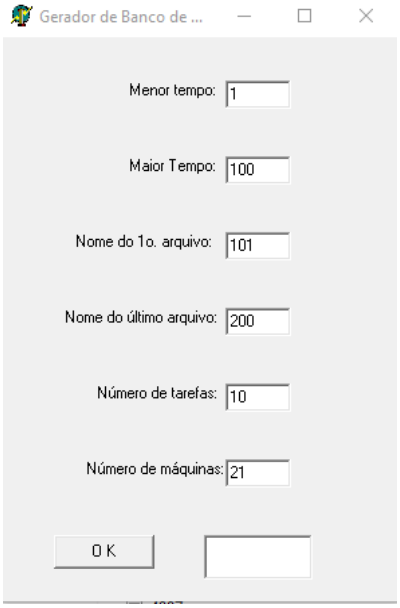
Figura 3 – Gerador de Banco de Dados



Fonte: Autoria própria

Optou-se por uma variação de tamanho de 10 em 10 e começando a variação pelas máquinas, sendo assim no próximo conjunto de dados as matrizes são 10x20, como visto na Figura 4 onde a primeira matriz do conjunto foi salva no arquivo 101 e a última no 200.

Figura 4- Gerador de Banco de Dados 2



Fonte: Autoria própria

Esse procedimento foi realizado até o número de máquinas ser igual a 50, e então as tarefas começou-se a variar as tarefas, aumentando também em 10, e as maquinas retornam ao valor 10 inicial, sendo então, o próximo conjunto de matrizes 20x10, se repetindo o processo de aumentar as máquinas. Essa sequência foi seguida até se obter matrizes 100x50, conforme observa-se na Tabela 2.

Tabela 2- Combinação de matrizes para construção do banco de dados

Combinações de Matrizes	Arquivos
10X11	1-100
10X21	101- 200
10X31	201- 300
10X41	301-400
10X51	401-500
20X11	501-600
20X21	601-700
20X31	701-800

20X41	801-900
20X51	901-1000
30X11	1001-1100
30X21	1101-1200
30X31	1201-1300
30X41	1301-1400
30X51	1401-1500
40X11	1501-1600
40X21	1601-1700
40X31	1701-1800
40X41	1801-1900
40X51	1901-2000
50X11	2001-2100
50X21	2101-2200
50X31	2201-2300
50X41	2301-2400
50X51	2401-2500
60X11	2501-2600
60X21	2601-2700
60X31	2701-2800
60X41	2801-2900
60X51	2901-3000
70X11	3001-3100
70X21	3101-3200
70X31	3201-3300
70X41	3301-3400
70X51	3401-3500
80X11	3501-3600
80X21	3601-3700
80X31	3701-3800
80X41	3801-3900
80X51	3901-4000
90X11	4001-4100
90X21	4101-4200
90X31	4201-4300
90X41	4301-4400
90X51	4401-4500
100X11	4501-4600
100X21	4601-4700
100X31	4701-4800

100X41	4801-4900
100X51	4901-5000

Fonte: Autoria Própria

Todas as matrizes geradas foram salvas em arquivos enumerados de 1 a 5000 de acordo com a Tabela 2 e posteriormente foram utilizados para a execução das heurísticas.

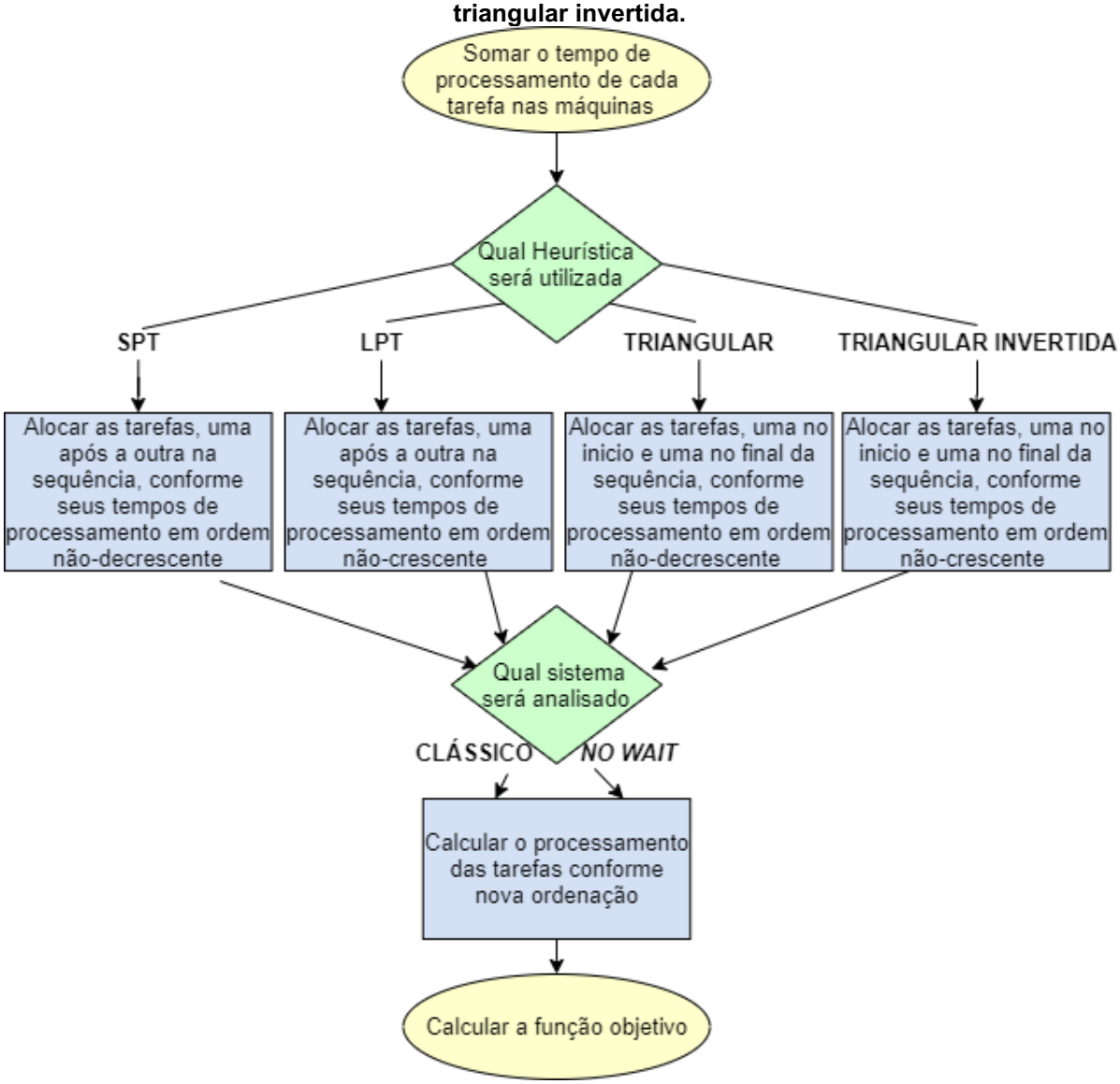
4.2 IMPLEMENTAÇÃO DOS MÉTODOS HEURÍSTICOS

Para a analise dos dados foi utilizado o *software* Dev-Pascal com códigos de programação de ordenação em Pascal, o código utilizado encontra-se como ANEXO A no presente trabalho.

Primeiro, na parte de leitura das matrizes, o código realizava a soma das linhas de cada coluna com seus para que a última linha, que se refere ao prazo de entrega, contasse com valores possíveis, e esse novo valor era associado ao prazo de entrega e a última linha desconsiderada para o processo de ordenação nos diferentes métodos (SPT, LPT, triangular, triangular invertida e variações do SPT) e sistemas (clássico e *no wait*).

O Fluxograma 1 representa como foi realizado o início do processo de ordenação até o cálculo da função objetivo para os métodos heurísticos SPT, LPT, triangular, triangular invertida.

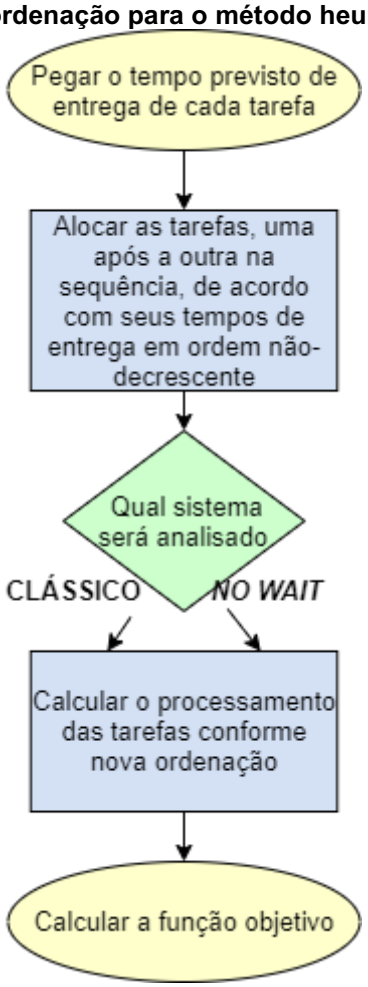
Fluxograma 1 – Processo de ordenação para os métodos heurísticos SPT, LPT, triangular, triangular invertida.



Fonte: Autoria Própria

O Fluxograma 2, também, mostra como foi realizado o início do processo de ordenação até o cálculo da função objetivo, porém para o método heurístico variação SPT-Atraso.

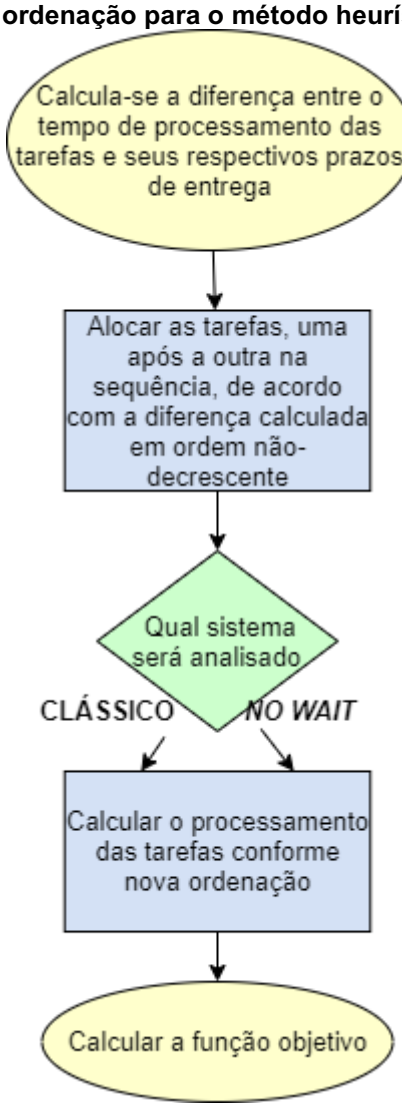
Fluxograma 2 – Processo de ordenação para o método heurísticos variação SPT-Atraso.



Fonte: Autoria Própria

O Fluxograma 3, assim como os anteriores, mostra como foi realizado o início do processo de ordenação até o cálculo da função objetivo, porém para o método heurístico variação SPT-Diferença.

Fluxograma 3 – Processo de ordenação para o método heurísticos variação SPT-Diferença.



Fonte: Autoria Própria

Com relação aos cálculos da função objetivo utilizou-se as equações a seguir:

$$Atraso = T_{final} - T_e \quad (1)$$

$$T_e = T_p + A \quad (2)$$

$$T_e = A \quad (3)$$

$$T_e = Tp - A \quad (4)$$

$$Tp = \sum_{i=1}^{i=máx} Jij \quad (5)$$

$$T_{final} = Tp + G \quad (6)$$

Onde, temos as seguintes variáveis:

T_{final} - Tempo total em que a tarefa foi processada;

T_e - Tempo de entrega esperado, que corresponde ao prazo de entrega, que é determinado;

A- Número aleatório gerado através do *software* gerador de dados;

T_p - Tempo mínimo necessário para o processamento das tarefas;

G- *gaps* de processamento;

J_{ij}- Tempo de processamento da tarefa (j), gerado de maneira aleatória e variando de 1 a 100;

i- Número de máquinas.

A função objetivo analisada que buscou-se minimizar, tempo de atraso total, foi a representada na Equação 1.

Para os métodos heurísticos SPT, LPT, triangular e triangular invertida, T_e foi calculado através da Equação 2. Já para a variação do SPT que não leva em conta o tempo de processamento das tarefas temos a Equação 3. E para a segunda proposta, que considera a diferença o tempo de processamento das tarefas e o prazo de entrega das mesmas, usou-se a Equação 4. Uma vez que, cada método possui sua própria forma de ordenação, feitas a partir dos tempos de entrega esperados (T_e) sendo assim, apresentam diferentes códigos e soluções, pois influencia diretamente na função objetivo.

E ainda, para o sistema *no wait* foi necessário desenvolver uma matriz *delay* para os cálculos.

4.3 PROCEDIMENTOS DE ANÁLISE DE DADOS

Para compilar as informações utilizou-se o aplicativo editor de planilhas *Microsoft Office* Excel e por meio dele realizou-se gráficos para indicadores de sucesso por máquinas e tarefas e desvio relativo. Sendo os mesmos calculados através das formulas 7 e 8, respectivamente.

$$PS = \frac{NPb}{NPtotal} \tag{7}$$

Onde:

PS- Porcentagem de sucesso;

NPb- Número de problemas para os quais o método b obteve melhor tempo de atraso total;

NPtotal- Número total de problemas resolvidos.

$$Drb = \frac{Db - D*}{D*} \tag{8}$$

Onde:

Drb- Desvio relativo do método b;

Db- Tempo de atraso total obtido pelo método b;

D*- Melhor tempo de atraso obtido, para um determinado problema.

Para isso, todos os dados de saída do *software* Dev-Pascal foram colocados no Excel, posteriormente os cálculos para obtenção dos indicadores foram realizados e a partir disso os gráficos foram gerados possibilitando uma melhor analise dos resultados.

5 RESULTADOS E DISCUSSÃO

Nesse capítulo os resultados gerados são analisados a partir de gráficos. Os dois primeiros conjuntos de dados correspondem as análises dos métodos heurísticos Triangular, Triangular Invertida, SPT e LPT, e os dois últimos ao método SPT e suas variações.

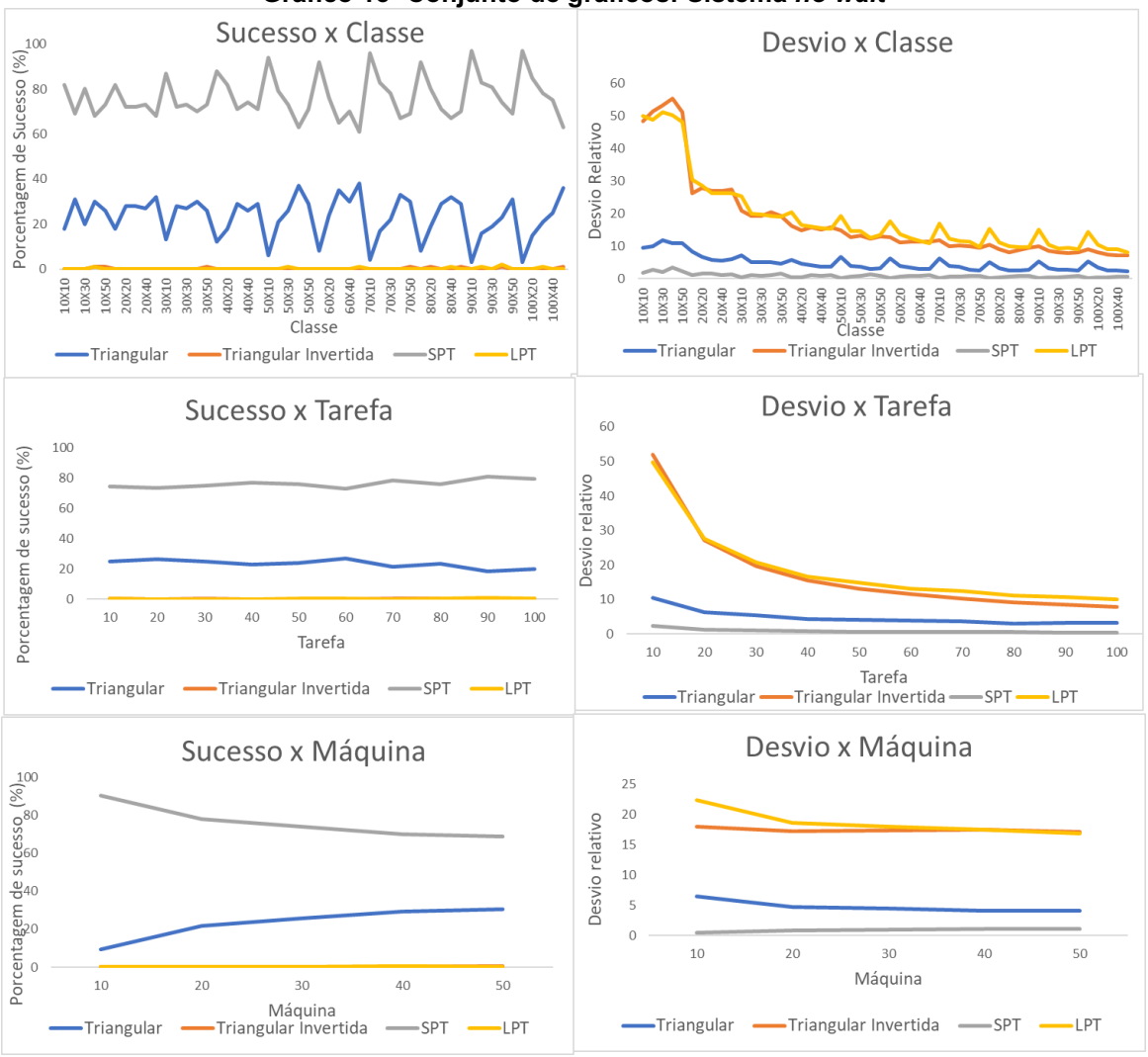
No Gráfico 14 pode-se observar um conjunto de quadros nos quais observa-se que o método SPT apresenta os melhores sucessos com algumas oscilações em classes com o número de tarefas menor. Já com relação aos desvios, pode-se notar que as outras heurísticas apresentam maiores desvios principalmente quando as tarefas são menores.

Gráfico 14- Conjunto de gráficos: Sistema clássico

Fonte: Autoria Própria

No Gráfico 15 pode-se observar um padrão parecido, mesmo com a mudança para o sistema *no wait,* onde a heurística SPT apresenta os melhores resultados, porém aqui as oscilações são muito maiores que nos gráficos anteriores e que, ainda, a heurística Triangular tem um sucesso melhor conforme aumenta-se o número de máquinas. Já em relação aos desvios, assim como no sistema clássico, temos que menores números de tarefas geram maiores desvios.

Gráfico 15- Conjunto de gráficos: Sistema *no wait*



Fonte: Autoria Própria

Devido aos resultados positivos da heurística SPT, optou-se por calcular variações dessa heurística nos diferentes sistemas para análise dos resultados. Uma a partir dos prazos de entrega, denominada na legenda por SPT- atraso, em que a mesma regra de prioridade é a mesma do SPT tradicional, mas não leva em conta o tempo de processamento das tarefas. E a outra proposta, denominada SPT-

diferença, considera a diferença o tempo de processamento das tarefas e o prazo de entrega das mesmas, sendo realizada a ordenação a partir desse ponto.

A partir do conjunto de gráficos representados no Gráfico 16 pode-se perceber que apesar do SPT original continuar sendo melhor que as variações proposta têm-se uma importante oscilação com relação ao sucesso entre ela e a variação que não leva em conta o tempo de processamento das tarefas, em que algumas classes a mesma obtêm maior sucesso.

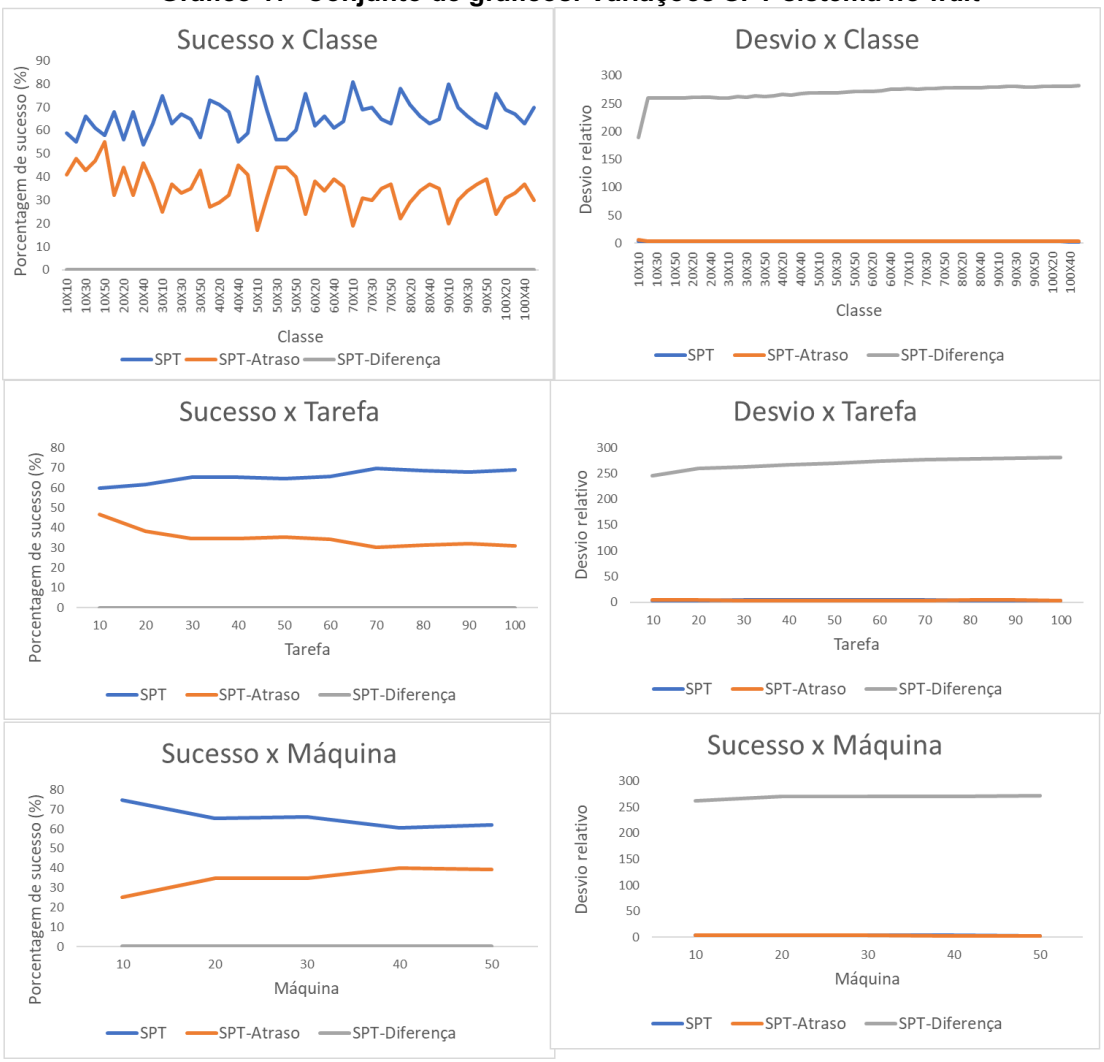
Gráfico 16- Conjunto de gráficos: Variações do SPT no sistema clássico

Fonte: Autoria Própria

O Conjunto de quadros representados no Gráfico 17 está relacionado as variações do SPT, porém no sistema *no wait*, nesse conjunto pode-se visualizar que a variação SPT-atraso melhora, com relação ao sucesso, conforme aumenta o

número de máquina e diminui o número de tarefas. Já em relação aos desvios não se tem muita diferença para com o conjunto anterior.

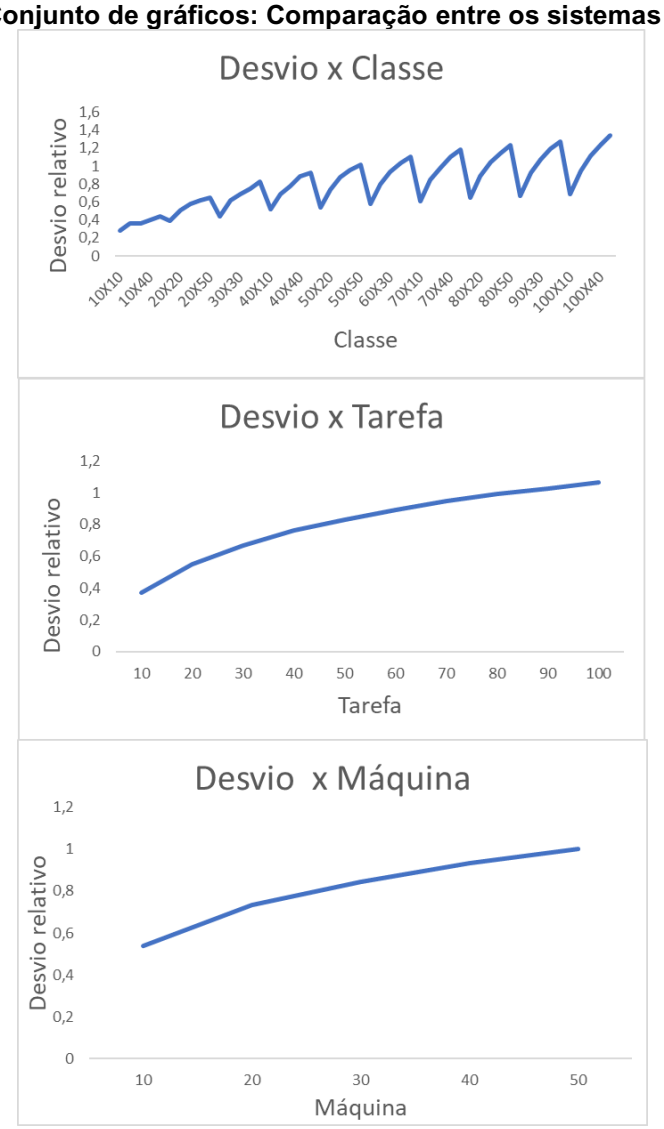
Gráfico 17- Conjunto de gráficos: Variações SPT sistema *no wait*



Fonte: Autoria Própria

Com o intuito de comparar os diferentes sistemas de produção, analisou-se os desvios relativo de um sistema para o outro, também com relação a classe, tarefa e máquina, para realização dessa analise apenas os dados de SPT foram utilizados. Essa comparação pode ser observada no conjunto de gráficos representados no Gráfico 18.

Gráfico 18- Conjunto de gráficos: Comparação entre os sistemas de produção



Fonte: Autoria própria

Conforme observa-se no Gráfico 18, pode-se notar que os desvios do sistema *no wait* em relação ao sistema clássico aumentam conforme aumenta o número de máquinas e/ou tarefas, e, com relação a classe percebe-se que os desvios diminuem conforme diminui o número de máquinas.

6 CONSIDERAÇÕES FINAIS

O sequenciamento de tarefas é uma ferramenta muito útil para as indústrias e o problema de minimização de tempo total de atraso se faz muito importante principalmente nos tempos atuais em que se têm uma concorrência muito grande em praticamente todas as áreas, atrasos podem ocasionar multas contratuais e pode fazer com que a empresa perca parte de seus clientes e do seu lucro.

Para esse trabalho foram avaliadas inicialmente quatro heurísticas, sendo elas SPT, LPT, Triangular e Triangular Invertida, e posteriormente mais duas heurísticas que são variações do SPT. Todas as seis heurísticas abordadas no trabalho foram analisadas tanto no sistema clássico de produção quanto no sistema *no wait*, para cada um dos 5.000 problemas gerados, e assim, obteve-se 60.000 soluções referentes aos cálculos da função objetivo.

O objetivo geral do presente trabalho foi analisar o comportamento de métodos heurísticos com intuito de definir quais têm melhor desempenho em ambiente *flow shop* para minimização dos atrasos de entrega, sendo assim conclui-se através dos resultados obtidos que o SPT obteve um melhor resultado geral tanto com relação à sucesso quanto em relação a desvios em todas as análises. Mas o SPT-atraso, a sua variação que não leva em conta o tempo de processamento das máquinas, também obteve resultados muito bons e que podem ser analisados em trabalhos futuros, principalmente no sistema *no wait* em casos com maior número de máquinas ou menor número de tarefas para problemas de atraso que buscam a minimização do tempo de atraso total.

Com relação aos sistemas de produção o sistema *no wait* apresenta uma restrição de não parada do fluxo da tarefa quando comparado ao sistema clássico, era esperado que se obtivesse resultados piores, o que pode-se observar nos gráficos de desvios que com maiores números máquinas e/ou tarefas o mesmo também aumentava, porém quando analisou-se os desvios com relação as classes notou-se que com números baixos de máquinas e altos de tarefas os desvios diminuíam.

O objetivo específico adaptar os métodos heurísticos encontrados na literatura, para comparar desempenho, foi alcançado, uma vez que além da adaptação para calculo da função objetivo proposta, também adaptou-se a heurística SPT apresentando outras duas variações para a mesma.

Para o objetivo específico realizar a experimentação computacional, em diferentes conjuntos de dados foram gerados conjuntos de dados de diferentes quantidades de máquinas e tarefas.

Com relação objetivo específico analisar os resultados por meio de gráficos e tabelas com ferramentas estatísticas apresentadas na literatura, foi usado os cálculos de porcentagem de sucesso e desvio médio relativo para realizados dos gráficos apresentados nos resultados que facilitaram as análises.

O objetivo específico encontrar soluções satisfatórias para as funções objetivo estabelecidas foi atingido uma vez que as soluções encontradas através das heurísticas foram condizentes com o que era esperado.

As principais dificuldades encontradas durante a execução do trabalho foram a falta de familiaridade com a programação que fizeram com que a mesma demorasse mais do que o previsto para ser finalizada, e a geração do extenso banco de dados devido a dificuldade de conseguir dados reais para realização do trabalho.

Uma sugestão para trabalhos futuros é a realização do mesmo em um caso real e específico, com intuito de constatar se para a nova análise os resultados serão parecidos ou alguma outra heurística dentre as propostas terá um resultado superior.

REFERÊNCIAS

ARAUJO, Matheus de F.; ARROYO, José E. C.; SANTOS, André G. dos. **Método *Proximity Search* para a resolução do problema de *flow shop scheduling* não permutacional com trabalhadores heterogêneos**. Anais do XLVIII SBPO Simpósio Brasileiro de Pesquisa Operacional Vitória, ES, 27 a 30 de setembro de 2016.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. H. **Pesquisa operacional para cursos de engenharia**. [S.l.]: Elsevier, 2007.

BILINSKI, Patricia Aparecida; et al. **Aplicação da pesquisa operacional na otimização da lucratividade de uma empresa do segmento de marcenaria.** XXXVI Encontro nacional de engenharia de produção, João Pessoa, PB, 2016.

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.

BRANCO, Fábio J. C.; FUCHIGAMI, Helio Yochihiro. **Métodos de alto rendimento e baixa complexidade em *flowshop*.** Revista Gestão da Produção Operações e Sistemas, v. 12, n. 4, p. 32, 2017.

BRANCO, Fábio J. C.; NAGANO, Marcelo S.; MOCCELLIN, João V. **Minimização da duração total da programação em sistemas de produção *flow shop* sem interrupção na execução das tarefas.** GEPROS - Gestão da Produção, Operações e Sistemas, v. 2, n. 3, p.39-47, 2008.

BRANCO, Fábio J. C; SANTOS, Alessandra L. dos. **Avaliação de ordenações iniciais para o problema *flowshop* com restrição *no-wait* e minimização de função-objetivo ponderada entre *makespan* e *flowtime*.** Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional, 2016, Vitória.

BRYMAN, A. ***Research methods and organization studies***. Londres: Unwin Hyman, 1989.

BUXEY, Geoff. ***Production Scheduling: Practice and Theory***. *European Journal of Operational Research*, vol. 39, p. 17-31, 1989.

BUZZO, W. R., MOCELLIN, J. V. **Programação da produção em sistemas *flow shop* utilizando um método heurístico híbrido algoritmo genético-*simulated annealing*.** Revista Gestão e Produção, v.7, n.3,p.364-377, 2000.

CARVALHO, Larissa de; et al. **Modelos de Programação Linear orientados a programação da produção: Uma análise dos trabalhos orientados aos sistemas *flow shop* tradicional.** Anais do IX EEPA Encontro de Engenharia de Produção Agroindustrial, Campo Mourão, PR, 19 a 20 de novembro de 2015.

CAUCHICK, Paulo A.; et al. **Metodologia da Pesquisa em engenharia de produção e gestão de operações.** 2. ed. Rio de Janeiro: Elsevier: ABEPRO, 2012.

CHEN, Toly. ***Intelligent scheduling approaches for a wafer fabrication factory*.** Disponível em: <https://doi.org/10.1007/s10845-010-0445-9>. Acesso em 07 mai.2019.

CURY, Ricardo M. **Uma abordagem difusa para o problema de *Flow-shop scheduling*.** Tese (Doutorado) - Programa de Pós-Graduação em Engenharia de Produção Universidade Federal de Santa Catarina, Florianópolis, 1999.

FERREIRA, Deisemara. **Abordagens para o problema integrado de dimensionamento e sequenciamento de lotes da produção de bebidas.** Tese (Doutorado em Engenharia de Produção) - Universidade Federal de São Carlos, São Carlos, 2006.

FUCHIGAMI, Helio Y. **Flexibilidade *flow line* com tempos de *setup*: métodos heurísticos.** 2010. 342 f. Tese (Doutorado) - Curso de Engenharia de Produção, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

GAREY, Michael R.; JOHNSON, David S.; SETHI, Ravi. ***The complexity of flowshop and jobshop scheduling.*** *Mathematics of operations research*, 1976, 1.2: 117-129.

GUPTA, J. N. D; HO, J. C. ***Minimizing flowtime subject to optimal makespan on two identical parallel machines*.** Pesquisa Operacional, v.20, n.1, p.05-17, 2000.

ISHIBUCHI, Hisao; et al. ***Genetic Algorithms and Neighborhood Search Algorithms for Fuzzy Flowshop Scheduling Problems***. *Fuzzy Sets and Systems*, vol.67, pp.81-100, 1994.

JI, Min; HE, Yong; CHENG, TC Edwin. ***Single-machine scheduling with periodic maintenance to minimize makespan. Computers & operations research***, v. 34, n. 6, p. 1764-1770, 2007.

KIYUZATO et al. **O problema de programação de tarefas para minimizar a soma dos atrasos: um estudo de caso.** Anais do XXX ENEGEP Encontro Nacional de Engenharia de Produção, 2010.

LUCHE, José R. D.; MORABITO, Reinaldo. **Otimização na programação da produção de Grãos eletro fundidos: um estudo de caso**. Gestão & Produção. São Paulo, SP, v.12, 2005.

MAGATÃO, Leandro; ARRUDA, Lúcia V. R. de; JUNIOR, Flávio N. **Um modelo híbrido (CLP-MILP) para *scheduling* de operações em polidutos.** Pesqui. Oper., Rio de Janeiro, RJ, v.28, n.3, p. 511-543, 2008.

MOCCELLIN, João V.; NAGANO, Marcelo Seido. ***Flow shop* com máquinas paralelas genéricas**. Simpósio Brasileiro de Pesquisa Operacional, n35, Natal-RN, 2003.

MORAIS, Márcia de F.; MOCCELLIN, João V. **Métodos heurísticos construtivos para redução do estoque em processo em ambientes de produção *flow shop* híbridos com tempos de *setup* dependentes da sequência.** Gest. Prod., São Carlos, SP, v. 17, n. 2, p. 367-375, 2010.

NAGANO, Marcelo; MOCCELLIN, João Vitor; LORENA, Luiz Antonio. **Redução do estoque em processamento em sistemas de produção *flow shop* permutacional.** Revista Produção Online, v. 5, n. 3, 2005.

NAGANO, Marcelo Seido; MOCCELLIN, João V. **Análise de procedimentos de busca tabu para solução do problema de programação de operações *flow shop* permutacional utilizando busca aleatória na vizinhança**. Anais do Rio de Janeiro: [s.n.], 1996.

NAPIERALA, Hieronim. **O algoritmo genético para solucionar a programação de produção do sistema de fluxo permutacional.** Ciências sociais aplicadas em revistas, v. 13, n24, 2014.

NASCIMENTO, Paulo F. **Metodologia da Pesquisa Científica: teoria e prática – como elaborar TCC**. Brasília: Thesaurus, 2016.

PACHECO, Ricardo F.; SANTORO, Miguel C. **A adoção de modelos de *scheduling* no Brasil: deficiências do processo de escolha.** Gest. Prod. [online]. 2001, vol.8, n.2, pp.128-138. ISSN 0104-530X.

SILVA, Bruno J. V.; MORABITO, Reinaldo; YAMASHITA, Denise Sato. **Otimização na programação de montagens na indústria aeronáutica**. Gest. Prod., São Carlos, v. 21, n.1, pp.33-44. Epub Nov 05, 2013. ISSN 0104-530X.

SOUZA, Eduardo C. **Programação de tarefas em um *flow shop*.** Tese (Doutorado em Engenharia Naval e Oceânica) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2009.

TAILLARD, Eric. ***Benchmarks for basic scheduling problems. european journal of operational research,*** v. 64, n. 2, p. 278-285, 1993.

TEIXEIRA, Evelin M. A.; MENDES, Alexandre de S. **Modelo de programação horária de uma célula flexível de manufatura com uso de múltiplas ferramentas.** Anais do XII ENEGEP Encontro Nacional de Engenharia de Produção, 1997.

VAZ, Cristiano M.; MORAIS, Vinicius F. S.; FUCHIGAMI, Hélio Y. **Modelo de programação linear inteira mista para minimização dos adiantamentos e atrasos em *flow shop* com *setup* dependende da sequência.** 2013. 6f. Trabalho de Conclusão de Curso (Graduação em Matemática Industrial) - Universidade Federal de Goiás, 2013.

ANEXO A - Programa em linguagem Pascal utilizado para implementação dos métodos heurísticos

```
program tmo;


          uses Windows, SysUtils;
          type
          arrn = array[0..200] of integer;
          var
mk,ft,i,ii,j,jj,jjj,k,m,n,a,aa,bb,cc,dd,ee,xx,ax,fof,tempoinicial,tempofinal,tempo,atraso,x,
y : integer;
             ma,mb, d, atrasos : array [0..200,0..200] of integer;
             s,t,w,w1,sp : array [0..200] of integer;
             arquivo, arquivo1 : text;
          {
          procedure fo(a:arrn;ee:integer);      //NW
            var lll : integer;
            begin
              mk:=0;
              ft:=0;
              for lll := 1 to ee-1 do mk := mk + d[a[lll],a[lll+1]];
              mk := mk+t[a[ee]];
              for lll := 2 to ee do ft := ft + (ee+1-lll) * d[a[lll-1],a[lll]];
              for lll := 1 to ee do ft := ft + t[a[lll]];


              atraso := 0;
              x := 0;
              mb[m,1] := t[a[1]];
              mb[m-1,1] := 0;
              for lll := 2 to ee do mb[m-1,lll] := mb[m-1,lll-1] + d[a[lll-1],a[lll]];
              for lll := 2 to ee do mb[m,lll] := mb[m-1,lll] + t[a[lll]];


               for lll := 1 to ee do  //FUNÇÃO OBJETIVO
                 begin
                   x := ma[m+1,a[lll]] - mb[m,lll];
                   if x < 0 then atraso := atraso - x;
                 end;

              fof := atraso;

            end;
          }


          procedure fo(ax:arrn;b:integer);         //CLÁSSICO
          begin
            for ii := 1 to m do for jj := 1 to n do mb[ii,jj] := 0;
            mb[1,1] := ma[1,ax[1]];
            for ii := 2 to b do mb[1,ii] := mb[1,ii-1] + ma[1,ax[ii]];
            for ii := 2 to m do mb[ii,1] := mb[ii-1,1] + ma[ii,ax[1]];
```

```
  for ii := 2 to m do
    for jj := 2 to b do
      begin
        a := mb[ii,jj-1];
        if a < mb[ii-1,jj] then a := mb[ii-1,jj];
        mb[ii,jj] := a + ma[ii,ax[jj]];
      end;
  atraso := 0;
  x := 0;
  for ii := 1 to b do  //FUNÇÃO OBJETIVO
    begin
      x := ma[m+1,ax[ii]] - mb[m,ii];
      if x < 0 then atraso := atraso - x;
    end;

  fof := atraso;

end;



begin             //LEITURA DOS DADOS
assign(arquivo1,'C:\Users\Asus\Desktop\TCC 2\DADOS\saida.txt');
rewrite(arquivo1);
for xx := 1 to 5000 do
begin
tempoinicial := GetTickCount;
assign(arquivo,'C:\Users\Asus\Desktop\TCC 2\DADOS\'+IntToStr(xx)+'.txt');
reset(arquivo);
readln(arquivo,m,n);
a := 0;
for i := 1 to m do
  begin
    for j := 1 to (n-1) do read(arquivo,ma[i,j]);
    readln (arquivo,ma[i,n]);
  end;


m := m-1;

writeln('colunas: ',n);
writeln('linhas: ',m);


{aa := 0; // MATRIZ DELAY NO WAIT
bb := 0;
cc := 0;
dd := 0;
for jj := 1 to n do
  begin
```

```
            for jjj := 1 to n do
              begin
                for i := 1 to m do
                  begin
                    ma[0,jjj] := 0;
                    aa := aa + ma[i,jj];
                    bb := bb + ma[i-1,jjj];
                    cc := aa - bb;
                    if cc > dd then
                      begin
                        d[jj,jjj] := cc;
                        dd := d[jj,jjj];
                      end
                      else
                        d[jj,jjj] := dd;
                        if i = m then
                          begin
                            aa:=0;
                            bb:=0;
                            dd:=0;
                          end;
                  end;
                if jj = jjj then d[jj,jjj] := 0;
              end;
          end;
        }

        for i := 1 to n do s[i] := 1;
        for i := 1 to n do t[i] := 0;
        for i := 1 to n do for j := 1 to m do t[i] := t[i] + ma[j,i];

        for i := 1 to n do ma[m+1,i] := ma[m+1,i] + t[i];

        for i := 1 to n do for j := 1 to n do if t[i] < t[j] then s[i] := s[i] + 1;          //< LPT
ou > SPT            //> Triangular  ou < Triangular invertida
        for i := 1 to n do for j := 1 to n do if i <> j then if s[i] = s[j] then s[j] := s[j] + 1;
          for i := 1 to n do
            begin
              ee := s[i];
              w[ee] := i;
            end;


        aa := n div 2;  // TRIANGULAR E TRIANGULAR INVERTIDA
        for i := 1 to aa do w1[i] := w[2*i-1];
        for i := aa+1 to n do w1[i] := w[2*n-2*i+2];



        //for i := 1 to n do w1[i] := w[i];  // LPT E SPT
```

```
for i := 1 to n do write(w1[i],' ');
fo(w1,n);

for i := 1 to n do write(mb[m,i],' ');
writeln;
writeln('funcao-objetivo da sequencia w1: ',fof);

close(arquivo);
tempofinal := GetTickCount;
tempo := tempofinal - tempoinicial;
writeln(arquivo1,fof,'-',tempo);
end;
close(arquivo1);
readln(n);
end.
```