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

PEDRO VERRI DINIZ
RODOLFO KIYOSHI TANAKA RAMOS

ADAPTAÇÃO E ANÁLISE DE DESEMPENHO DE HEURÍSTICAS PARA MINIMIZAÇÃO DO *GAP* EM PROBLEMAS DE *FLOWSHOP*

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2017

PEDRO VERRI DINIZ
RODOLFO KIYOSHI TANAKA RAMOS

ADAPTAÇÃO E ANÁLISE DE DESEMPENHO DE HEURÍSTICAS PARA MINIMIZAÇÃO DO *GAP* EM PROBLEMAS DE *FLOWSHOP*

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 – UTFPR.

Orientador: Prof. Dr. Fábio José Ceron Branco

PONTA GROSSA
2017

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

TERMO DE APROVAÇÃO DE TCC

ADAPTAÇÃO E ANÁLISE DE DESEMPENHO DE HEURÍSTICAS PARA MINIMIZAÇÃO DO *GAP* EM PROBLEMA DE *FLOWSHOP*

por

Pedro Verri Diniz
Rodolfo Kiyoshi Tanaka Ramos

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 24 de novembro de 2017 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.

_____
Prof. Dr. Fábio José Ceron Branco
Prof. Orientador

_____
Prof. Dr. Juan Carlos Claros Garcia
Membro titular

_____
Prof. Dr. Shih Yung Chin
Membro titular

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

AGRADECIMENTOS

Primeiramente à Deus, que sempre guiou nossos passos durante a trajetória acadêmica, nos concedendo a força necessária para nunca desistir e nos enchendo de perseverança para superar as dificuldades.

Aos pais, que sempre foram motivadores e incentivadores das nossas escolhas. Compreenderam a necessidade da distância, mas mesmo de longe vibraram com as nossas conquistas, se comoveram com as dificuldades e nos deram todo suporte para seguirmos nosso caminho.

Aos professores, que hoje fazem parte do que somos. Se dedicaram das mais diversas formas à nos transformar, buscando à todo momento transmitir o que possuem de melhor, com o objetivo de nos preparar para nossa carreira que está para começar. À eles, devemos uma enorme parcela das nossas conquistas. Aqui também queremos deixar um agradecimento especial ao professor Dr. Fábio Branco, que nos orientou durante o desenvolvimento desse trabalho, com muita disposição e respeito, concedendo todas as ferramentas necessárias.

Aos amigos, àqueles que deixamos no estado de SP, e fizeram valer suas amizades mesmo quando os encontros foram menos frequentes. Também àqueles que vieram junto com a trajetória acadêmica, os quais se tornaram especiais e serão levados para toda vida.

À Universidade Tecnológica Federal do Paraná, que nos concedeu a oportunidade de nos tornar engenheiros, oferecendo todo o aporte necessário para o nosso aprendizado.

RESUMO

DINIZ, P. V.; RAMOS, R. K. T. **Adaptação e Análise de Desempenho de Heurísticas Para Minimização do *gap* em Problema de *Flowshop***. 2017. 71f. Trabalho de Conclusão de Curso (Bacharelado em engenharia de produção). Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2017.

Existem muitas pesquisas voltadas à programação de tarefas, com diversos tipos de otimização (*makespan, flowtime*, atrasos, etc.). No entanto, uma área pouco explorada é a associação da minimização do *gap* entre o início e término de uma nova tarefa n e n+1 em uma máquina m. O *gap* pode gerar uma série de inconveniências à um sistema composto por múltiplas máquinas, como a ociosidade das mesmas, gerando impactos significativos na performance energética e custos de uma empresa. O presente trabalho tem como finalidade associar a minimização do *gap* total, em um sistema *flowshop,* com uma função de minimização de *makespan*, agregando o menor prejuízo possível do mesmo. A partir de uma heurística que combina os métodos LPT e NEH, criou-se uma função biobjetivo que apresentou resultados satisfatórios, com potencial para aumentar a eficiência das máquinas em um ambiente produtivo significativamente. Com a aplicação de um peso nas faixas de 10% e 25% de *gap* na função objetivo da heurística testada, encontrou-se o melhor resultado técnico, com um pequeno acréscimo de *makespan*, mas redução significativa do *gap* total.

Palavras-chaves: *Makespan*. *Gap*. Heurísticas. Sequenciamento de tarefas.

ABSTRACT

DINIZ, P. V.; RAMOS, R. K. T. **Adaptation and Performance Analysis of Heuristics for the Gap Minimization into Flowshop Problems**. 2017. 71p. Work of Conclusion Course (Graduation in engenharia de produção). Federal Technology University - Parana. Ponta Grossa, 2017.

There are many articles, works and researches regarding scheduling, with a wide variety of optimization objectives, such as makespan, flowtime, delays, etc. However, there is a field almost unexplored, and it is the association of the minimization of the gap between the beginning and ending of a job *n* and *n+1* on a machine *m*. This gap can produce a series of inconveniences to a system made of multiples machines, just as idleness of the machines, making significant impacts to the energy performance and costs of an industry. The present research has the objective of associating the minimization of total gap, in a flowshop system, with the objective function of makespan minimization, adding the lower impact possible to the makespan. Starting with a heuristic that combines the methods LPT and NEH, was created a biobjective function that achieved satisfying results, with potential to increase machine efficiency in a productive environment significantly. Within a gap application of 10% and 25% into the objective function, it was found the best techinal result, with a slight increase of makespan, but significant reductions on total gap.

Key-words: *Makespan*. *Gap.* Heuristics. *Scheduling*.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

De modo geral, os problemas de programação de tarefas consistem na alocação de recursos para realização das tarefas desejadas. De acordo com Pinedo (2012), esses recursos podem ser, por exemplo: máquinas, pessoas, unidades de processamento de computadores, pistas de aeroportos, entre outros. Assim como as tarefas podem ser: operações de processos produtivos, atividades diárias do cotidiano, aeronaves que devem decolar e/ou pousar, entre outras.

Em ambientes produtivos, os recursos são as máquinas e as tarefas, os quais trabalham em função das ordens de produção. As ordens de produção têm características como, por exemplo, data de entrega, prioridade da ordem, entre outras possibilidades. Cada ordem pode afetar a máquina de uma maneira, podendo ou não possuir tempo de *setup*, ou seja, um tempo para preparar a máquina para executar determinada tarefa. Os ambientes produtivos podem ter as seguintes disposições: máquina única, máquinas em paralelo, *flowshop* e *job shop.*

A disposição abordada nesse trabalho é a *flowshop*, a qual possui uma configuração em que *m* máquinas estão dispostas em série. Quando todas as tarefas obedecem a mesma sequência dentro das máquinas, ou seja, todas as *n* tarefas são processadas primeiro na máquina 1, em seguida na máquina 2, e seguem desse modo até a última máquina, classifica-se o problema como *flowshop* permutacional, utilizando-se da abreviação PFSP.

Os critérios mais comuns estudados na literatura são: a minimização do tempo total de fluxo (*flowtime*) e a minimização do tempo total da programação (*makespan*). O critério *flowtime* é utilizado para diminuição dos materiais em processamento, ou seja, quanto menor o tempo total de fluxo da programação, menos estoque de materiais semi-processados. Já para o critério *makespan*, busca-se o melhor desempenho na utilização das máquinas, uma vez que quanto menor o tempo total de processamento, mais rapidamente as máquinas estarão livres para realizar a próxima programação. Embora existam diversos outros critérios comumente estudados na literatura, o *gap*,

caracterizado pelo tempo ocioso entre o término de uma tarefa e o início de outra, em uma mesma máquina, ainda se trata de um critério pouco explorado.

O maior problema encontrado para solucionar os problemas de programação de tarefas é composto pelo número de combinações de suas soluções. Por exemplo, para solucionar um problema de *n* tarefas no ambiente produtivo mais simples, o de máquina única, existem *n* fatorial (!) soluções possíveis, o que nos retrata a natureza combinatória dos problemas de programação de tarefas. Dada as proporções que os problemas podem atingir, Garey *et al* (1976) afirmam que a maior parte dos problemas de programação de tarefas em ambiente *flowshop* são NP-*hard*, ou seja, não existem algoritmos conhecidos que sejam capazes de resolver esses problemas em tempo polinomial.

Embora as tentativas sejam inúmeras e, os computadores responsáveis pelo processamento de soluções sejam cada vez mais potentes, a maioria dos problemas (grande porte) de programação de tarefas em um ambiente *flowshop* ainda não possuem soluções ótimas conhecidas. Uma vez que são propostos mais métodos específicos para cada tipo de problema de programação na literatura, o presente trabalho se baseia na seguinte questão: os métodos heurísticos destacados nos trabalhos científicos voltados para minimização de *makespan,* em um ambiente *flowshop,* quando adaptados para considerarem a redução do *gap*, podem gerar resultados satisfatórios?

1.1 OBJETIVOS

Os objetivos da pesquisa são classificados como geral e específicos, de acordo com a seguinte disposição:

1.1.1 Objetivo Geral

Procura-se verificar o sucesso de uma heurística voltada para minimização de *makespan*, em um ambiente *Flowshop*, quanto aos impactos sofridos pela associação de redução do *gap*.

1.1.2 Objetivos Específicos

Neste trabalho espera-se trabalhar com os seguintes objetivos específicos:

i) Geração de um conjunto de instâncias de *n* tarefas por *m* máquinas (n x m), com a devida aleatoriedade de tempo das tarefas, para implementação das heurísticas estudadas.

ii) Definição de uma heurística simples para avaliação do método proposto;

iii) Comparação entre a heurística adaptada com os resultados obtidos pela minimização exclusiva de *makespan*;

iv) Construção de gráficos e tabelas, a partir de estatísticas de comparação de heurísticas.

1.2 JUSTIFICATIVA

Nas indústrias, a programação de tarefas está diretamente relacionada com um dos principais recursos de produção de qualquer negócio: o tempo. Sendo o tempo um fator decisivo no cenário industrial, pesquisadores, de todos os lugares do mundo, depositam esforços na construção de heurísticas, com soluções de alta qualidade para os problemas mais diversos possíveis. Porém, outra dimensão responsável por gerar preocupação no cenário industrial, é a performance de custos. Constantemente as empresas adotam estratégias agressivas para reduzirem os custos de suas operações. Nos estudos do campo de programação de tarefas, a redução de custos é uma

consequência dos resultados, o foco se encontra na produtividade. A proposta do presente trabalho diferencia-se justamente nesse campo pouco explorado, por meio da redução do *gap,* a qual ainda que possa gerar algum prejuízo de produtividade, pode alavancar uma grande redução do custo final de um sistema *flowshop.*

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados os principais autores e conceitos referenciados como base no desenvolvimento da presente pesquisa.

2.1 PROGRAMAÇÃO DA PRODUÇÃO

Para Pinedo (2012) a programação da produção é o processo de tomada de decisão que é responsável por alocar os recursos às tarefas durante um determinado período de tempo, visando otimizar um ou mais objetivos.

Semelhante à definição de Pinedo, Fuchigami (2005) define a programação da produção como sendo a alocação dos recursos necessários para a realização/execução das tarefas baseados no tempo.

Os recursos e tarefas são encontrados de diversas maneiras, os recursos podem ser máquinas em uma indústria; pistas de pouso e decolagem em um aeroporto; computadores em um escritório; caixas em um supermercado; e assim por diante. Enquanto as tarefas podem ser operações de um processo produtivo em uma indústria; decolagens e aterrissagens em um aeroporto; formulários, planilhas ou fichas que devem ser preenchidas em um escritório; e as compras dos clientes em um supermercado; entre outros. Cada tarefa pode ter sua prioridade, o seu tempo de *setup*, data limite para entrega. Os objetivos citados por Pinedo (2012) também podem variar de diversas formas, sendo a minimização do tempo de finalização da última tarefa ou *makespan*, o qual será devidamente aprofundado neste trabalho, e o tempo de fluxo ou *flowtime* os mais recorrentes na literatura.

Segundo Slack *et al.* (1999), a programação da produção é uma das tarefas mais complexas, pois primeiramente é necessário que os programadores lidem com inúmeros tipos de recursos variando simultaneamente, por exemplo, máquinas podem ter diferentes capacidades e pessoas diferentes habilidades. E o que torna ainda mais

difícil a tarefa dos programadores, é que o número de programações possíveis cresce exponencialmente conforme aumenta o número de atividades e processos, ou seja, se considerarmos *n* tarefas para serem programadas, existirá *n***!** (*n* fatorial) programações distintas dos trabalhos em um processo de apenas uma máquina. Se formos ainda além e considerar um ambiente com mais de uma máquina (M > 1), o número de programações possíveis é $(n!)^M$.

Desse modo podemos observar a primeira característica que faz com que os problemas de programação da produção sejam difíceis de se resolver, o fato de esses problemas serem de natureza combinatória, ou seja, o número de soluções possíveis cresce exponencialmente em diversas dimensões, em relação à quantidade de tarefas, operações ou máquinas.

Uma ferramenta muito utilizada na programação é o Gráfico de Gantt, desenvolvido por Henry L. Gantt em 1917, que representa graficamente o tempo com uma barra. A partir do gráfico de Gantt podemos interpretar quando a operação foi iniciada e quando ela foi finalizada, qual tarefa foi executada em qual máquina, entre outras características. Segundo Slack *et al*. (1999, p. 244-245) o gráfico é vantajoso pois proporciona de modo simples uma representação visual do que está ocorrendo, ou do que deve ocorrer em cada operação. Na Figura 1 temos um exemplo do Gráfico de Gantt para uma tarefa e duas máquinas.

Figura 1 - Exemplo de um Gráfico de Gantt

Fonte: Autoria própria

2.1.1 Problemas de Programação de Tarefas em Máquinas

Como descrito por Pinedo (2012) um problema de programação de tarefas em máquinas é descrito por três campos | | . Onde o campo representa o ambiente de máquinas e contém apenas uma entrada; o campo fornece detalhes das restrições e/ou características do processamento, podendo conter nenhuma, uma ou múltiplas entradas, e por último o campo contém a função objetivo do problema, na maior parte das vezes sendo composto por uma única entrada.

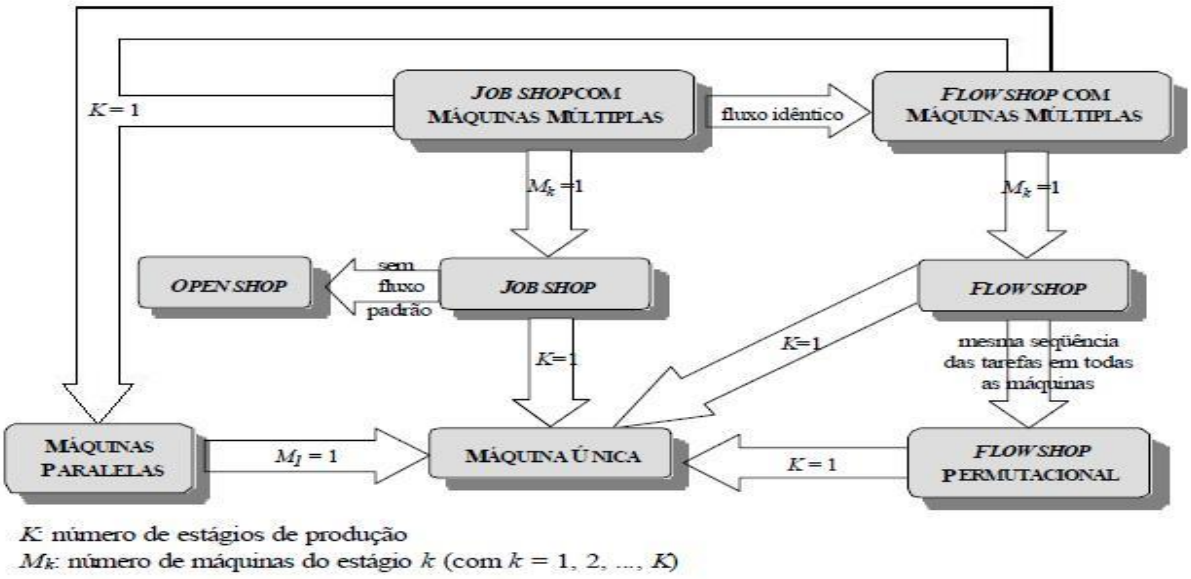
Os ambientes de trabalho possíveis para o campo são:

- **Máquina única**: modelo mais simples de um ambiente de trabalho, composto por apenas uma máquina.
- **Máquinas paralelas idênticas**: ambiente de trabalho composto por *m* máquinas idênticas, que podem executar todas as tarefas.
- ***Flowshop***: existem *m* máquinas em série, e cada tarefa *j* deve ser processada em cada máquina *m*, todas as tarefas devem seguir a mesma ordem de processamento, ou seja, todas as tarefas devem ser processadas primeiro na máquina 1, depois na máquina 2 até chegar na máquina *m*.
- ***Flowshop* flexível**: é a junção dos conceitos *flowshop* e máquinas paralelas, ou seja, existem *c* estágios em série com um certo número de máquinas idênticas em paralelo, e cada tarefa deve ser processada primeiro no estágio 1, depois no estágio 2, até o estágio *c*. Cada estágio funciona como um banco de máquinas em paralelo, onde cada tarefa *j*, pode ser processada em qualquer máquina do estágio *c*.
- ***Job shop***: dentro do *job shop* com *m* máquinas, cada tarefa *j* possui uma rota pré determinada e exclusiva.
- ***Job shop* flexível**: é a combinação do *job shop* e máquinas paralelas, nesse ambiente existem *c* estágios com um certo número de máquinas idênticas em paralelo, cada trabalho tem sua rota pré determinada, e pode ser processado em qualquer máquina do estágio.

- ***Open shop***: as tarefas não possuem uma ordem de processamento, ou seja, elas podem ser executadas em qualquer máquina *m* em qualquer ordem que o programador determinar como a mais adequada para o objetivo do problema.

A Figura 2 demonstra a relação entre os ambientes de trabalho que compõem este objeto de estudo.

Figura 2 - Relação entre as classes de problemas de programação.



Fonte: Fuchigami, 2005.

Alguns exemplos de restrições e/ou característica de processamento que podem compor o campo são:

- Tempos de *setup*;
- Tempos de *setup* dependentes da sequência;
- Permutação, Pinedo (2012) explica permutação como sendo a limitação que implica que a ordem de cada tarefa tem na primeira máquina, se mantém por todo o processo;

- Preemptividade, que segundo Pinedo (2012) implica que uma tarefa não precisa necessariamente ser finalizada uma vez que iniciada em uma máquina, ou seja, a tarefa pode ser interrompida durante o seu processamento e ser colocada outra tarefa em seu lugar, como demonstrado no gráfico de Gantt da Figura 3.

Figura 3 - Gráfico de Gantt ilustrando uma programação com preemptividade

Fonte: Autoria própria

- *No-wait*, restrição que garante que uma vez que uma tarefa inicia o seu processo na primeira máquina de um *flowshop*, esta deve seguir seu processo até a última máquina sem esperar entre o tempo final de uma máquina e o tempo de início da máquina subsequente, isso resulta em um atraso no tempo de início da tarefa na máquina 1 para permitir que a tarefa percorra todo seu caminho sem esperar entre as máquinas. A Figura 4 representa um gráfico de Gantt que ilustra uma programação respeitando à restrição *no-wait*, com três tarefas e quatro máquinas.

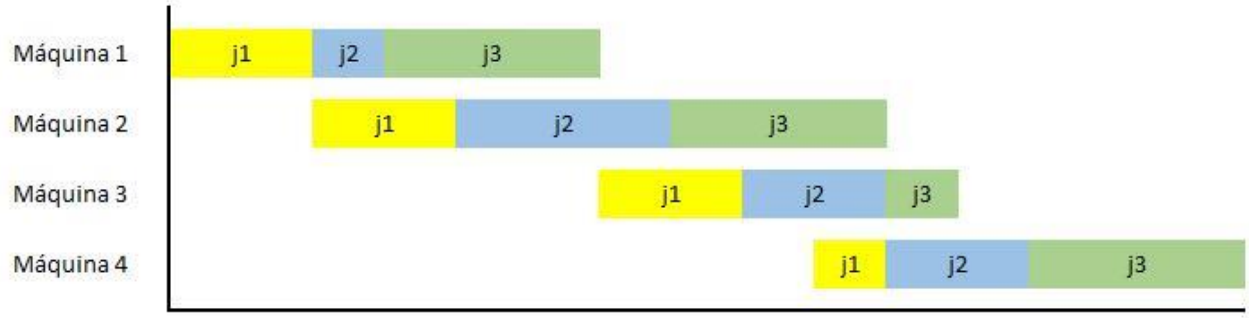
Figura 4 - Gráfico de Gantt de uma programação com restrição No-wait

Fonte: Autoria própria

- *No-idle*, essa restrição solicita que uma vez iniciada uma tarefa qualquer na máquina *m*, todas as tarefas que necessitam do processamento nesta máquina sejam executadas sem pausar o processamento da máquina, ou seja, sem que a máquina espere entre o término do processamento da tarefa *j* na posição *i*, e o início da tarefa

j na posição *i*+1. A Figura 5 ilustra um gráfico de Gantt de uma programação que respeita a condição *no-idle*, com três tarefas e quatro máquinas.

Figura 5 - Gráfico de Gantt de uma programação com restrição No-idle



Fonte: Autoria própria

Por fim, algumas das funções objetivos que podem fazer parte do campo são:

- ***Makespan***: é o tempo total gasto para finalizar a última tarefa do sistema;
- ***Flowtime***: tempo total de fluxo das tarefas, ou seja, é o somatório dos tempos em que cada tarefa esteve dentro do processo até a sua finalização na última máquina;
- **Atraso máximo**: visa minimizar o maior atraso das tarefas executadas;
- **Atraso total**: visa minimizar o somatório dos atrasos de cada tarefa.

Esses quatro tipos de função objetivo são as que possuem maior aplicabilidade para os sistemas produtivos atuais. Embora a restrição *no-idle* se assemelhe com a proposta do presente trabalho, não há correlação entre ambas, uma vez que buscamos a redução do gap mantendo como objetivo minimizar *makespan*. Ou seja, não se pretende restringir totalmente a heurística estudada, mas sim amortecer os impactos gerados pelo *gap* incluindo uma restrição de penalidade para cada unidade de *gap.* Desta forma o problema de programação de máquinas estudado neste trabalho pode ser descrito como *Flowshop* | %*Gap* | *Makespan* (| |).

2.2 PROBLEMAS DE SEQUENCIAMENTO DE ATIVIDADES EM UM SISTEMA *FLOWSHOP*

Dentre os tópicos listados no campo , este trabalho possui como enfoque o sequenciamento de tarefas dentro de um sistema *flowshop*.

Nawaz *et al*. (1983) afirmam que o problema de sequenciamento de atividades se caracteriza pela ordenação de um grupo de tarefas que demandam processamento em uma ou mais máquinas, ou seja, cada tarefa é alocada em uma posição *i* relativa às demais tarefas. Os problemas de sequenciamento podem ser aplicados em diversas ocasiões, uma vez que surgem sempre que há possibilidade de escolha na ordem das atividades de um processo, tais como tarefas de manufatura em uma indústria.

Conforme apontado anteriormente, o ambiente *flowshop* é caracterizado por um tipo comum de *layout* em ambientes produtivos, nos quais m máquinas são dispostas em série. De acordo com a descrição de Pan e Ruiz (2013), o problema de sequenciamento em um ambiente *flowshop* é definido por tarefas que passam por todas as máquinas de acordo com uma ordem em comum, ou seja, as tarefas são submetidas de acordo com uma ordenação pré definida das máquinas, resultando em um conjunto de possibilidades de solução de *n!*, denomina-se esse tipo de problema como problema de *flowshop* permutacional ou pela siga *PFSP* (*permutational flowshop problem*).

É verdade que uma ou mais tarefas do processo podem não ter a necessidade de passarem por uma determinada máquina da disposição em série, no entanto, ainda é necessário que a tarefa passe por todas as estações, sem exceção; isso significa que o tempo de processamento atribuído da tarefa nessa máquina é zero (Gupta e Stafford, 2006, p. 701).

Um ambiente *flowshop*, como descrito por Pinedo (2012, p. 151) é um ambiente produtivo em que as tarefas devem seguir uma série de operações, na maior parte das vezes essas operações devem ser realizadas para todas as tarefas, com a mesma rota dentro do ambiente produtivo. Sendo assim assume-se que as máquinas se encontram em série e esse ambiente é o que caracteriza um ambiente *flowshop*.

Gupta e Stafford (2006) caracterizam o *flowshop* por um fluxo mais ou menos contínuo e ininterrupto de tarefas através de diversas máquinas em série. Considera-se esse fluxo como unidirecional, uma vez que todas as tarefas seguem a mesma rota através das máquinas.

Baker (1974) estabelece uma série de presunções para o *PFSP*:

- Cada tarefa *i* pode ser processada no máximo em uma máquina *j* ao mesmo tempo;
- Cada máquina M pode processar apenas uma tarefa ao mesmo tempo;
- Não é possível preempção, isto é, o processamento de uma tarefa *i* em uma máquina *j* não pode ser interrompido;
- Todas as tarefas são independentes, portanto estão disponíveis para processamento desde o instante 0.
- Os tempos de *setup* podem ser desprezíveis ou adicionados aos tempos de processamento e, portanto, podem ser ignorados;
- As máquinas estão disponíveis continuamente;
- Uma possível fila de tarefas é permitida, ou seja, se uma máquina necessária para processamento estiver ocupada, a tarefa pode esperar até o instante de sua desocupação.

2.2.1 Restrições e Funções Objetivo Para o Ambiente *Flowshop*

Segundo Pinedo (2012), o ambiente *flowshop* pode possuir algumas restrições e/ou características de processamento. Essas características de processamento ou restrições são propriedades genéricas encontradas nos diversos ambientes de trabalho, não somente no *flowshop*. Essas restrições encontram-se listadas e explicadas no campo da seção 2.1.1. Cada uma dessas restrições afeta o ambiente produtivo de determinada forma, e podem aparecer combinadas umas às outras.

As funções objetivo são referentes ao campo e também são genéricas para aplicação nos diversos tipos de ambiente de produção. Para Pinedo (2012), uma função objetivo é caracterizada pela construção de uma equação que busca um resultado ótimo dentro de um universo de possibilidades. Portanto, as funções objetivo podem ser empregadas em uma infinidade de situações, com a finalidade de alcançar resultados melhores ou mesmo ótimos para os problemas investigados.

As funções objetivo são denominadas dessa forma, justamente por representarem a busca por um resultado específico, um objetivo. Os objetivos podem variar de diversas maneiras, dependendo exclusivamente do que o pesquisador busca como resultado ótimo. Nas funções construídas para o sequenciamento de tarefas, geralmente buscam-se minimizações, as quais podem ser referentes a tempo de término, tempo de fluxo das tarefas, atrasos, adiantamentos, custos, entre outras possibilidades. Conforme mencionado anteriormente, as funções mais recorrentes na literatura são direcionadas para a minimização do *makespan* (tempo total de término), *flowtime* (tempo de fluxo), o somatório dos atrasos de cada atividade e o atraso máximo dentre as tarefas estudadas (Baker e Trietsch, 2009).

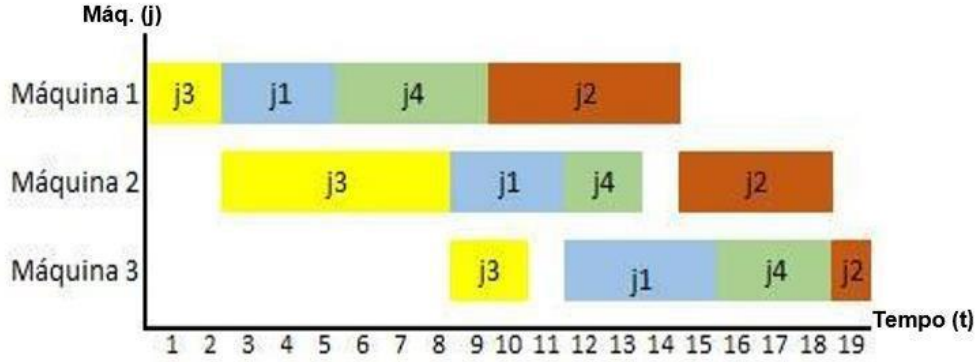
Nas seções subsequentes, essas funções objetivo são aprofundadas de forma a conceder ao leitor um entendimento melhor sobre as particularidades de cada uma das principais funções.

2.2.1.1 Makespan

O tempo total de término é representado pelo instante no qual a última atividade é finalizada na última máquina. Segundo Ruiz e Stützle (2007), a minimização do tempo total de término é o critério mais utilizado nos problemas referentes a *PFSP.* A minimização do makespan é importante por representar a redução do tempo de término do conjunto das atividades de um problema de programação de tarefas. Portanto, sempre que o *makespan* é reduzido, automaticamente aumenta-se a capacidade de

produção. A Figura 6 represente um gráfico de Gantt, no qual é possível extrair o valor do *makespan*.

Figura 6 - Gráfico de Gantt para sequenciamento de tarefas em ambiente flowshop



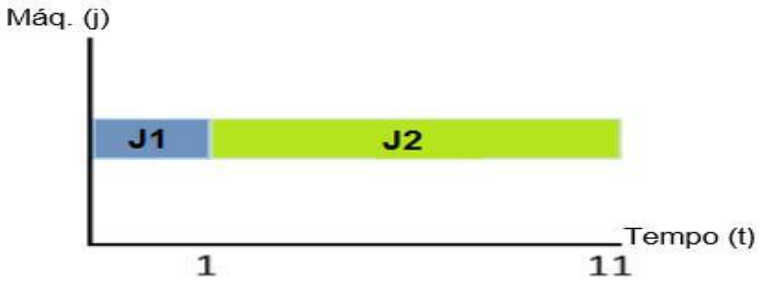
Fonte: Autoria própria

Na Figura 6, o gráfico de Gantt representa o sequenciamento de quatro tarefas em três máquinas, em um ambiente *flowshop*. Uma vez que o makespan é representado pelo instante de término da última tarefa na última máquina, o mesmo é correspondente ao valor de 19 unidades de tempo conforme exibido na escala horizontal do gráfico, no momento em que a tarefa j2 é encerrada na máquina 3.

2.2.1.2 *Flowtime*

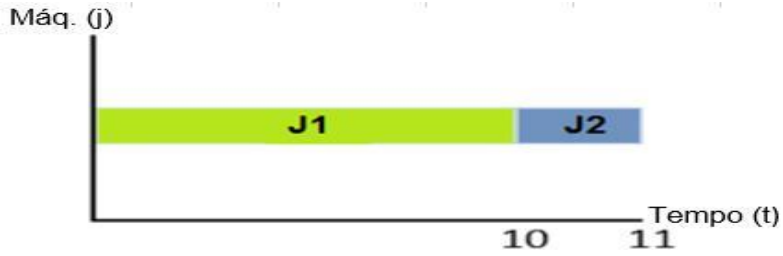
De acordo com Baker e Trietsch (2009), *flowtime* ou tempo de fluxo consiste no somatório dos tempos de término de cada atividade em uma dada sequência. No caso de duas tarefas, J_1 e J_2, com tempos de processamento, t_1 e t_2, de 1 e 10 respectivamente, se a sequência for primeiro J_1 e depois J_2, conforme ilustrado pela Figura 7, o tempo de fluxo obtido será de 12 (1 + 11). No entanto, se a sequência for alterada para primeiro J_2 e depois J_1, como mostrado na Figura 8, o tempo de fluxo será: 10 (tempo de término de J_2) + 11 (tempo de término de J1) = 21.

Figura 7 - Gráfico de Gantt para sequenciamento de duas tarefas em máquina única, com flowtime igual a 12



Fonte: Autoria própria

Figura 8 - Gráfico de Gantt para sequenciamento de duas tarefas em máquina única, com flowtime igual a 21



Fonte: Autoria própria

Para Liu e Reeves (2001), a minimização do tempo total de fluxo passou a despertar mais a atenção de pesquisadores, essa mudança de comportamento dos pesquisadores pode ser atribuída ao fato dos ambientes de produção se tornarem mais dinâmicos, além da relação direta que o tempo de fluxo apresenta com operações logísticas como o nível do estoque de material em processamento. Além desses aspectos, Tasgetiren *et al.* (2007) também consideram como impactos da redução do tempo de fluxo, o uso de recursos de forma mais estável e uma rápida rotação das tarefas.

2.2.1.3 Atraso total e atraso máximo

As funções de atraso podem se desdobrar em diversas possibilidades. As mais recorrentes na literatura são representadas pelo atraso máximo de uma tarefa e pelo

somatório dos atrasos de cada tarefa. Pode-se desejar reduzir o atraso do sequenciamento, quando os dados do problema possuem tarefas com um determinado tempo de entrega.

O atraso total do sequenciamento de tarefas se dá pelo somatório de todos os atrasos do sistema. A minimização do tempo de atraso total pode resultar em diversos benefícios para a indústria, como a redução de multas devido aos atrasos ou mesmo pela melhoria da reputação da companhia. O atraso máximo é representado pelo maior valor de atraso dentre todas as tarefas. É possível que pesquisadores busquem a redução do atraso máximo, devido a um tempo limite de atraso, o qual pode ser definido, por exemplo, por uma política de boa relação com os clientes (Baker e Trietsch, 2009).

2.2.2 Modelos Matemáticos Aplicados em *Scheduling*

Segundo Baker e Trietsch (2009), os modelos matemáticos são amplamente utilizados nos contextos de planejamento e logísticos. É pela modelagem matemática que é possível obter resultados ótimos pelas funções objetivos. De acordo com a equação da função objetivo trabalhada, são incluídas uma série de outras equações, geralmente chamadas de restrições.

A equação da função objetivo por si só não possui capacidade de gerar resultados plausíveis, afinal uma função de minimização dentro da programação de máquinas tende a zerar quaisquer resultados, uma vez que é utilizado a medida de tempo como resultado final e não é possível obter um valor menor que zero para tempo. Portanto, Baker e Trietsch (2009) salientam a importância do uso de condições, para que o problema possa ser resolvido de forma lógica, nos modelos matemáticos, os quais podem ser inseridos em diversos *softwares* para a obtenção de um resultado ótimo. No entanto, um dos problemas da modelagem matemática, é o tempo que a complexidade de certos problemas pode tomar, como no caso de *PFSP* estudado.

3 MÉTODOS HEURÍSTICOS PARA SEQUENCIAMENTO DE TAREFAS

Os esforços em busca de uma solução ótima para os problemas de sequenciamento de tarefas são incessantes. No entanto, a complexidade dos problemas de *PFSP* abriu espaço para outros métodos de busca por soluções, substituindo os algoritmos exatos pelas chamadas heurísticas, as quais se caracterizam por técnicas de soluções aproximadas. As técnicas de solução exata a princípio resolvem o problema, mas esbarram em barreiras como a quantidade de tempo e memória que determinados problemas de sequenciamento requerem. Portanto, as heurísticas, ainda que não garantam resultados ótimos, apresentam soluções aproximadas em pouco tempo, agregando em economia e eficiência na busca por melhores resultados (Nawaz *et al*., 1982, p.91).

Resumidamente, pode-se afirmar que as heurísticas são mecanismos criados para a redução do universo total de soluções possíveis. Em problemas complexos como o de *PFSP,* o universo de soluções pode ser imenso, de tal forma que os avanços tecnológicos atuais ainda não são suficientes para realizar a verificação de todas as soluções possíveis. Portanto, uma heurística bem construída tem como característica a redução do universo de soluções possíveis, fazendo com que as soluções sejam construídas de acordo com determinados critérios. Apesar de os resultados nem sempre serem resultados ótimos, as heurísticas trabalham com um número muito grande de possibilidades em frações de segundos, sendo uma ferramenta indispensável para problemas de *scheduling* (Baker e Trietsch, 2009).

Em 1983, Nawaz *et al.* propuseram uma heurística de minimização de *makespan* para o problema de *PFSP*, chamando-a de *NEH*. Ruiz e Stützle (2007) afirmam em seu texto a consideração pela literatura ao método *NEH*, dizendo que não há evidência de que foram desenvolvidos métodos heurísticos substancialmente melhores do que o proposto por Nawaz *et al*. Motivados pelo método *NEH*, diversos autores desenvolveram heurísticas mais eficazes, no entanto, ainda seguindo as premissas estabelecidas pelo mesmo; como a heurística *Raj* desenvolvida por

Rajendran que utiliza a regra de ordenação e alocação da heurística *NEH*, ou a heurística *WY* elaborada por Woo e Yim, que apesar de não apresentar similaridade no método de ordenação, faz uso do mesmo método de alocação de Nawaz *et al.*, além de centenas de outros autores que continuam desenvolvendo heurísticas baseadas no método concebido pela heurística *NEH* (Ruiz e Maroto, 2005, p. 119).

Com a evolução das heurísticas ao longo do tempo, também surgiram novos métodos para a construção das mesmas. Framinan *et al*. (2005) descrevem uma classificação das heurísticas existentes. Primeiro, as heurísticas são divididas em três etapas: desenvolvimento de índice, construção da solução e aperfeiçoamento da solução, sendo que uma heurística pode incluir uma ou mais etapas. No desenvolvimento do índice, as tarefas são ordenadas de acordo com algum critério estabelecido pelas características do problema, por exemplo o tempo de processamento utilizado no método *NEH*. A etapa de construção é caracterizada pelo desenvolvimento de uma solução a partir da ordenação realizada na fase anterior. Por fim, a etapa de aperfeiçoamento de soluções consiste na empregabilidade de um método de busca local com o objetivo de melhorar a solução já encontrada. Uma heurística é considerada composta quando envolve ambas, a etapa de construção e a de aperfeiçoamento (Framinam *et al.*, p.1241).

3.1 HEURÍSTICAS CONSTRUTIVAS

O algoritmo de Johnson (1954) é reconhecido por ser a primeira heurística construtiva explorando o problema de PFSP. A heurística proposta por Johnson é de caráter simples, tendo por objetivo a geração de uma solução ótima para um problema de duas máquinas. Posteriormente, diversos autores exploraram a regra de duas máquinas, desenvolvida por Johnson, no desenvolvimento de heurísticas mais robustas, adaptando o número de *m* máquinas para duas máquinas “virtuais” (Ruiz e Maroto, 2005, p.481).

Para Campello e Maculan (1994), uma heurística construtiva se caracteriza pela construção de uma rota a partir de interações singulares, isto é, cada interação é responsável pela atribuição do próximo valor dentro de uma sequência, de acordo com os critérios estabelecidos pela heurística. A sequência é construída a partir de uma solução vazia e uma vez que os critérios são atendidos, não é possível a alteração do valor atribuído.

3.1.1 Aperfeiçoamento de Heurísticas

O aperfeiçoamento de heurísticas é composto pela construção de uma heurística a partir de uma solução factível já encontrada. Portanto, de forma diferente das heurísticas construtivas, as heurísticas de aperfeiçoamento são passíveis de alterações das sequências definidas previamente. Os métodos de aperfeiçoamento se caracterizam por múltiplas repetições, onde as soluções são comparadas uma a uma durante as iterações, mantendo sempre o melhor resultado entre as comparações. O resultado final das heurísticas de aperfeiçoamento se dá pela instituição de um ótimo local, isto é, uma sequência que não é mais passível de melhoramento de acordo com os critérios adotados na heurística (Cordenonsi, 2008, p.32).

3.1.2 Metaheurísticas

As metaheurísticas geralmente são compostas por uma combinação de procedimentos heurísticos, que também podem ser empregados nos problemas de PFSP. Normalmente, uma metaheurística se inicia com uma sequência construída por heurísticas simples, as quais são combinadas com um mecanismo de repetição de acordo com um critério de parada, retornando a melhor sequência encontrada (Ruiz e Maroto, 2005, p.483).

Segundo Ruiz e Stützle (2007), as metaheurísticas são capazes de oferecer resultados excelentes em termos de comparação com as heurísticas convencionais.

Porém, eles afirmam que a complexidade de alguns métodos propostos constitui um problema quanto aos métodos propostos; alguns métodos são tão complicados que uma codificação independente tende a não resultar na mesma efetividade.

Um outro ponto, defendido por Pan e Ruiz (2013), se dá pelo maior consumo de tempo para o processamento das metaheurísticas, as quais tendem a serem processadas durante alguns minutos, enquanto heurísticas tomam segundos para retornar uma solução, o que pode ser um grande empecilho para problemas de larga escala. Também é verdade que heurísticas eficientes ainda se fazem necessárias para gerar a sequência inicial dos métodos metaheurísticos, essenciais para os problemas de sequenciamento de tarefas.

3.2 SISTEMA *NO-IDLE*

Existem diversas restrições que podem ser impostas em um problema dentro da programação de tarefas em um ambiente *flowshop*. O sistema *no-idle* é aquele onde a sequência de trabalho de uma máquina é realizado sem interrupções, ou seja, para um problema onde é necessário sequenciar 3 tarefas em 3 máquinas, utilizando o sistema *no-idle*, a sequência escolhida deve ser processada em todas as máquinas sem que haja paradas nos processamentos das máquinas, podem adiar o início do processamento da tarefa em uma das máquinas para que o processo siga fluido e sem parada de máquinas. Esse sistema é muito utilizado para processos produtivos em casos onde: o custo de *setup* é muito alto ou o custo de manter o equipamento ligado é grande. Em vista disto é ideal que após o início de processamento da máquina, esta realize todas as tarefas da sequência sem interrupções. Um exemplo de indústria que utiliza esse sistema é a de fibra de vidro, visto que os custos de *setup* são muito altos e o desligamento e religamento dos equipamentos não seria efetivo em termos de custos.

3.3 HEURÍSTICAS

Nesse tópico são apresentados os mecanismos heurísticos "*Longest Process Time"* (LPT) e o clássico método desenvolvido por Nawaz *et. al* (NEH) que foram utilizados como base para a aplicação da heurística proposta.

3.3.1 Heurística Longest Processing Time – LPT

Uma das primeiras heurísticas desenvolvidas para a programação de tarefas em máquinas paralelas, Santos e Moccellin (2001) explicam que a heurística LPT prioriza a tarefa que possua a maior soma de tempos de processamento, ou seja, a heurística LPT tem como regra a ordenação das tarefas em ordem decrescente do tempo total de processamento. O LPT é utilizado para problemas que buscam minimizar o *makespan*.

3.3.2 Heurística Nawaz, Enscore e Ham – NEH

A heurística NEH criada em 1983 por Nawaz, Enscore e Ham, e que recebeu esse nome a partir das iniciais de seus criadores, é considerado segundo Ruiz e Maroto (2005) o melhor método heurístico para o critério de minimização de *makespan* da literatura. A heurística proposta por Nawaz *et al*. toma por base o método LPT, que se dá pela presunção de que as tarefas com o tempo de processamento total mais longo, são as que devem ter prioridade de processamento.

Após a análise LPT dos tempos de processamento e a criação de uma sequência base, o método NEH realiza testes a partir dessa sequência, isto é em um caso de 3 máquinas e 3 tarefas com objetivo de minimização de *makespan*, onde a sequência dada pelo método LPT é 3, 1 e 2, a heurística NEH testaria primeiramente o par 3 e 1 comparado ao par 1 e 3, para verificar qual possui menor *makespan*, digamos que o menor *makespan* foi dado pelo par 1, 3 em seguida é necessário testar e verificar

o tempo de processamento (*makespan)* para os trios 1, 3 e 2; 1, 2 e 3; e 2, 1 e 3, note que a tarefa que não havia sido testada anteriormente é colocada em cada uma das posições possíveis da sequência sem alterar a ordem determinada como melhor pelo teste anterior, após o teste é encontrada a sequência com menor *makespan* e esta é aceita como a sequência que as tarefas devem ser alocadas nas máquinas.

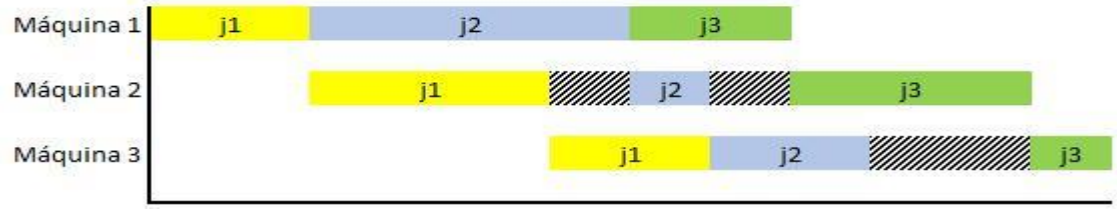
O mesmo se aplica para casos com mais tarefas, onde a próxima tarefa a entrar na sequência deve sempre ser testada em todas as posições possíveis, até atingir a última tarefa da sequência. A efetividade da heurística NEH motiva até hoje uma vasta quantidade de pesquisas para a redução do tempo total de realização das atividades.

3.3.3 Heurística Proposta

O presente trabalho busca propor uma heurística que apresente um equilíbrio entre o problema de ociosidade das máquinas (sistema *no-idle*), em que a função objetivo das heurísticas buscam sequências onde o GAP, tempo em que uma máquina aguarda a próxima tarefa da sequência finalizar o seu processo na máquina anterior, seja zero; e a minimização do *makespan*, que é o objetivo da maior parte dos processos produtivos pois reduz o tempo total de processamento. Essa proposta se mostra importante para a diminuição da ociosidade nos processos, sem o aumento do tempo total de processamento que é imposto pelo sistema *no-idle*, que por buscar o GAP zero muitas vezes sacrifica o desempenho quanto ao tempo total de processamento.

Desta forma a proposta consiste na utilização do método NEH com a modificação da função objetivo original de minimização do *makespan*, para uma função bi-objetiva, na qual a função objetivo mescle duas variáveis com peso no resultado final, permitindo a minimização de duas variáveis em conjunto. Para isso utiliza-se o conceito de GAP, que é o tempo que uma máquina deve esperar para iniciar o processo de uma tarefa que ainda não finalizou o seu processamento na máquina anterior, como mostrado no gráfico de Gantt da Figura 9.

Figura 9 - Gráfico de Gantt exemplificando uma situação de sequenciamento com GAP, representado pelas partes hachuradas



Fonte: Autoria própria

O conceito de *Gap* é importante dentro das indústrias, uma vez que em um sequenciamento de tarefas com um valor alto de *Gap* total, implica em um alto índice de ociosidade dos equipamentos, o que pode resultar em desperdícios.

4 METODOLOGIA

Seguindo as orientações de Gil (1999), a presente pesquisa é classificada quanto à natureza, objetivo e procedimento da seguinte forma:

- Natureza aplicada - Considera-se essa pesquisa como de natureza aplicada devido a aplicação do conhecimento na área de programação e sequenciamento de tarefas. A pesquisa é direcionada para geração de melhoria na área de *scheduling*, havendo a possibilidade de aplicabilidade prática em empresas interessadas, uma vez que as adaptações das heurísticas podem apresentar potencial quanto à redução de custos para os problemas de sequenciamento em um ambiente *flowshop*.

- Procedimento Bibliográfico - A busca por artigos, teses e dissertações em diversas bases de pesquisa, para a realização das adaptações propostas, concede caráter bibliográfico a pesquisa. Para minimizar os efeitos negativos, como a replicação de erros ou equivocações em artigos publicados, a pesquisa bibliométrica seguiu altos padrões, realizando-se a busca exclusivamente em bases de dados confiáveis e passando-os por uma filtragem em relação ao fator de impacto, onde materiais que apresentassem pontuação inferior a um ponto, foram descartados.

4.1 PROCEDIMENTOS ADOTADOS

Primeiramente, foi realizada uma revisão da literatura na qual, por meio da análise de teses, dissertações, artigos de revistas nacionais e internacionais e livros, relacionados com os temas: *Scheduling*, Programação da produção, Heurísticas para solução de problemas de *scheduling*, foi feito um estudo dos diversos conceitos das pesquisas que estão sendo realizadas nessa área, para entender e compreender a

profundidade do tema e ter conhecimento para desenvolver um trabalho estruturado e embasado nos conhecimentos mais atuais da área.

A partir da revisão de literatura, foi feito um apanhado das principais heurísticas que são utilizadas para solução de problemas de programação de tarefas, tendo como função objetivo a minimização do *makespan*. Com o objetivo de analisar e propor as adaptações, foi escolhida uma heurística comum no meio acadêmico para testar a performance da proposta, sendo composta pela combinação dos métodos LPT (*Longest Process Time)* e NEH.

Para a análise qualitativa das adaptações, se fez necessário a criação de um conjunto de instâncias, que possibilitam a análise de diversos cenários e suas variações na função objetivo. Os métodos de implementação devem ser realizados por linguagem computacional Pascal, utilizando-se para tanto, o software Dev-Pascal.

Com isso tem-se a possibilidade de realizar experimentos computacionais, que a partir da variação das instâncias, as quais simulam cenários com diferentes números de tarefas e máquinas, permitem testar as adaptações e obter resultados quantitativos da eficácia das heurísticas em alcançar os melhores resultados possíveis para o problema escolhido.

Com os resultados dos experimentos computacionais, é feita a comparação com os resultados obtidos de modo tradicional, e a partir dessa comparação podemos identificar se as adaptações obtiveram resultados satisfatórios. O procedimento para comparação das heurísticas é estatístico, sendo utilizado uma série de dados estatísticos, como médias, desvios, entre outros.

Por fim será realizado um compilado com as informações coletadas, e analisada a viabilidade e confiabilidade das adaptações propostas.

5 DESENVOLVIMENTO

Para avaliar o desempenho do método de redução do *GAP* proposto nesse trabalho, primeiramente foi aplicado um método tradicional de minimização de *makespan,* de modo a utilizar os resultados obtidos como parâmetro de comparação. A principal alteração entre o método tradicional e o método proposto está associada por uma mudança na função objetivo, mantendo os métodos LPT e NEH sem alterações. A implementação foi realizada por meio de linguagem Pascal, com o auxílio do *software* Dev-Pascal e implementação do código, o qual é exibido integralmente no apêndice A. Basicamente a função objetivo do primeiro método é composta de forma ilustrativa por:

$$\text{f.o.} = \min \{\text{makespan}\} \quad (1)$$

Após a alteração, a mesma pode ser representada da seguinte forma:

$$\text{f.o.} = \min \{\text{makespan} + \%GAP\} \quad (2)$$

Para encontrar o valor do *makespan*, que representa o tempo total de processamento, é preciso apenas encontrar o tempo de término da última tarefa na última máquina, já o valor do *GAP*, que representa o valor de *gap* total do problema, é necessário verificar a existência de um *gap* e então calcular o seu valor. Para isso é preciso comparar o tempo de término $x_{n,m}$ da tarefa *n* na máquina *m*, com o tempo de término $x_{n-1,m+1}$ da tarefa *n-1* na máquina *m+1*, caso $x_{n,m}$ for maior que $x_{n-1,m+1}$, isso caracteriza um *gap* para o início da tarefa *n* na máquina *m+1*, pois quando a máquina *m+1* terminar de processar a tarefa *n-1*, a tarefa *n* ainda não terá terminado seu processo na máquina *m*, fazendo com que a máquina *m+1* fique com um *gap*, um momento de ociosidade; para o caso de $x_{n-1,m+1}$ ser maior que $x_{n,m}$ o valor de *gap* sempre será considerado como zero, uma vez que não é possível existir um *gap* negativo, pois a tarefa não pode ser iniciada enquanto a outra não finalizar o seu processo na máquina anterior. O Quadro 1 ilustra e mostra como identificar os *gap's* em um problema de programação de tarefas.

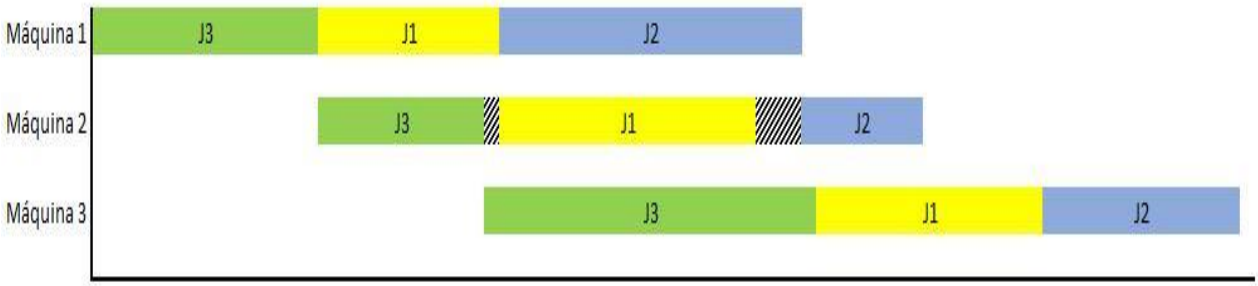
Quadro 1 - Tempos de processamento de um problema de 3 máquinas e 3 tarefas

Tempo de Processamento	J1	J2	J3
Máquina 1	12	20	15
Máquina 2	17	8	11
Máquina 3	15	13	22

Fonte: Autoria própria

Com o método de ordenação LPT, a ordem para esse problema se dá pela sequência J3, J1 e J2 (soma dos tempos de processamento são, respectivamente, 48, 44 e 41), e desse modo, a Figura 10 mostra um gráfico de Gantt para ilustrar o problema, enquanto o Quadro 2 mostra como ficariam os tempos de término de cada tarefa nesta ordem, nas máquinas, e como é feito o cálculo para a verificação do *gap*.

Figura 10 – Gráfico de Gantt do problema de 3 máquinas e 3 tarefas, onde as partes hachuradas representam o *gap*



Fonte: Autoria própria

Quadro 2 - Exemplo da solução e do cálculo do gap para a solução do problema 3x3

	J3	*gap* J3-J1	J1	*gap* J1-J2	J2
Máquina 1	15	-	27	-	47
gap M1-M2	-	*gap* = 1	-	*gap* = 3	-
Máquina 2	26	-	44	-	55
gap M2-M3	-	*gap* = 0	-	*gap* = 0	-
Máquina 3	48	-	63	-	76

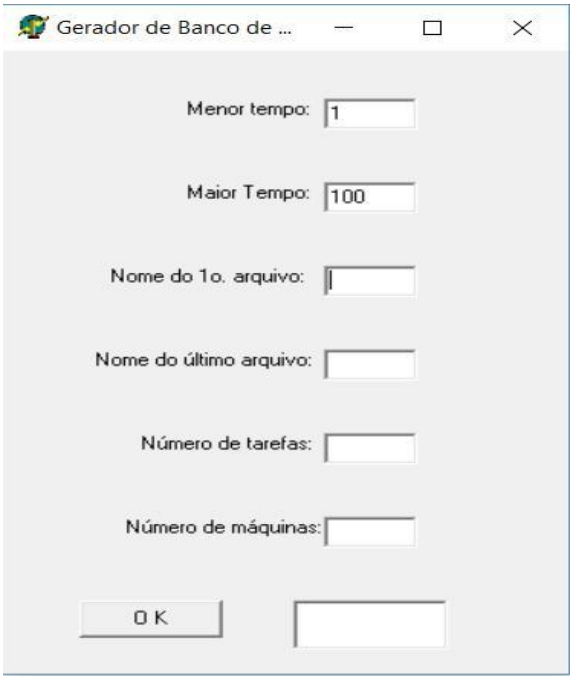
Fonte: Autoria própria

Com base na tabela acima podemos concluir que o *GAP* total desse cenário é 4, que corresponde à soma dos *gaps* intermediários, e esse valor será utilizado para a aplicação de porcentagem do *GAP* (%*GAP*) e juntamente com o *makespan* compor a função objetivo modificada proposta neste trabalho.

Para o método proposto, foi adicionada uma porcentagem do valor final do *gap*, ou seja, o valor a ser minimizado deixa de estar exclusivamente associado ao valor de *makespan*, mas também à uma parcela do valor do *gap*. Para tanto, a heurística foi testada com as seguintes porcentagens: 1%*GAP;* 10%*GAP;* 25%*GAP;* 50%*GAP* e 100%*GAP.*

Foi gerada uma base de dados com 5000 condições diferentes, onde a quantidade de máquinas *m* varia entre 5, 10, 15, 20, 25 e as tarefas *n* varia entre 10, 20, 30, 40, 50, 60, 70, 80, 90 e 100. Com o auxílio de um software gerador de dados, exibido na Figura 10, foi criado um banco de dados com cem variações de cada uma das seguintes combinações: 10x5; 10x10; 10x15; 10x20; 10x25; 20x5; 20x10; 20x15; 20x20; 20x25; 30x5; 30x10; 30x15; 30x20; 30x25; 40x5; 40x10; 40x15; 40x20; 40x25; 50x5; 50x10; 50x15; 50x20; 50x25; 60x5; 60x10; 60x15; 60x20; 60x25; 70x5; 70x10; 70x15; 70x20; 70x25; 80x5; 80x10; 80x15; 80x20; 80x25; 90x5; 90x10; 90x15; 90x20; 90x25; 100x5; 100x10; 100x15; 100x20; 100x25.

Figura 11 - Tela do gerador de Banco de Dados



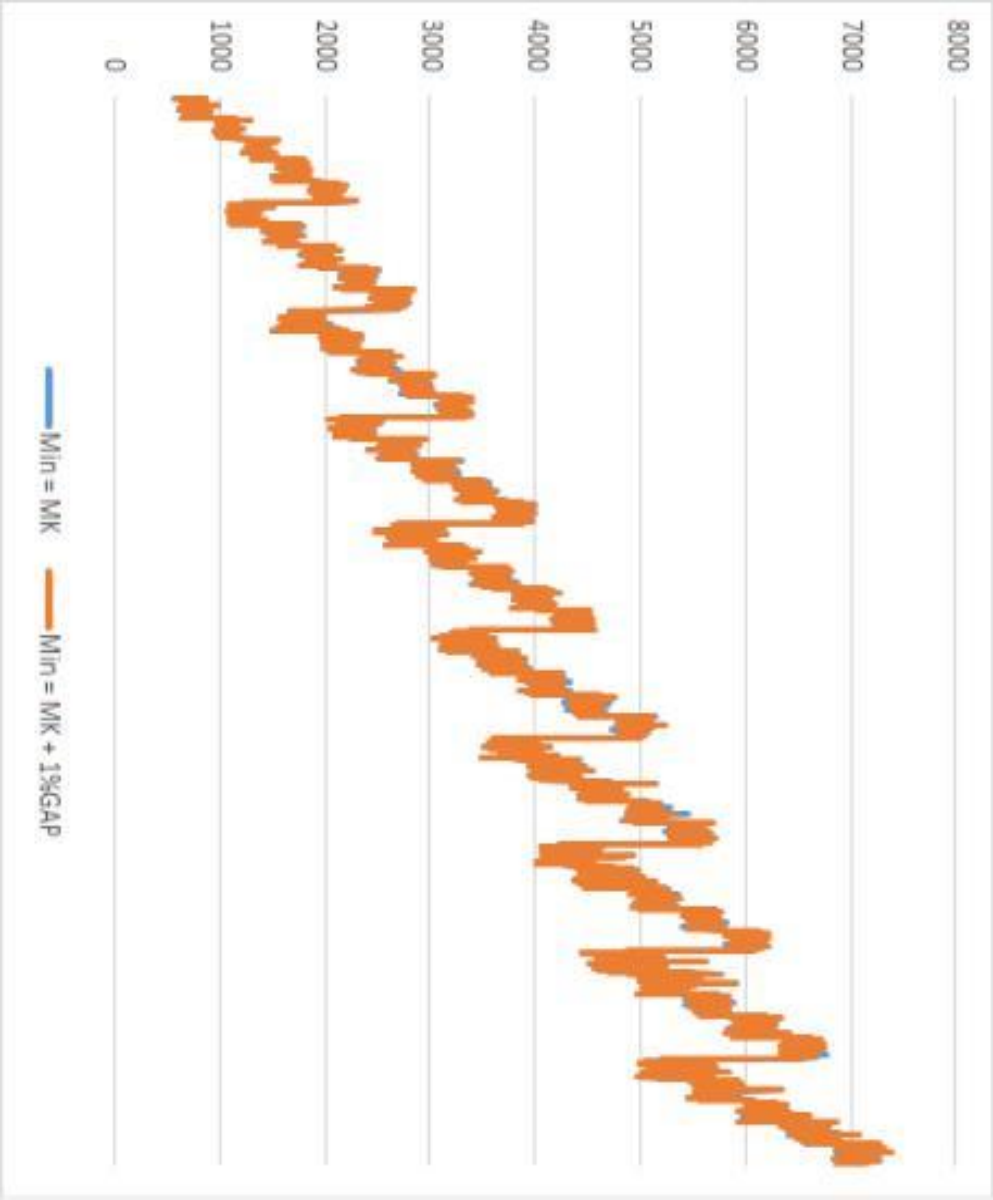
Fonte: Autoria própria

5.1 ANÁLISE DOS RESULTADOS

Após a experimentação da base de dados com a heurística proposta utilizando as diferentes porcentagens de penalidade do *gap*, os primeiros dados observados foram as comparações de *makespan* e *gap*, com o objetivo de verificar a dimensão dos impactos provocados pelas alterações na função objetivo do programa. Desse modo, utilizando-se dos valores brutos de *makespan,* obtidos em cada uma das 5000 situações diferentes, os seguintes gráficos foram construídos:

- Heurística sem alterações x Heurística com 1% de impacto de *gap;*
  MK (Média) ≅ 3831 MK’ (Média) ≅ 3828

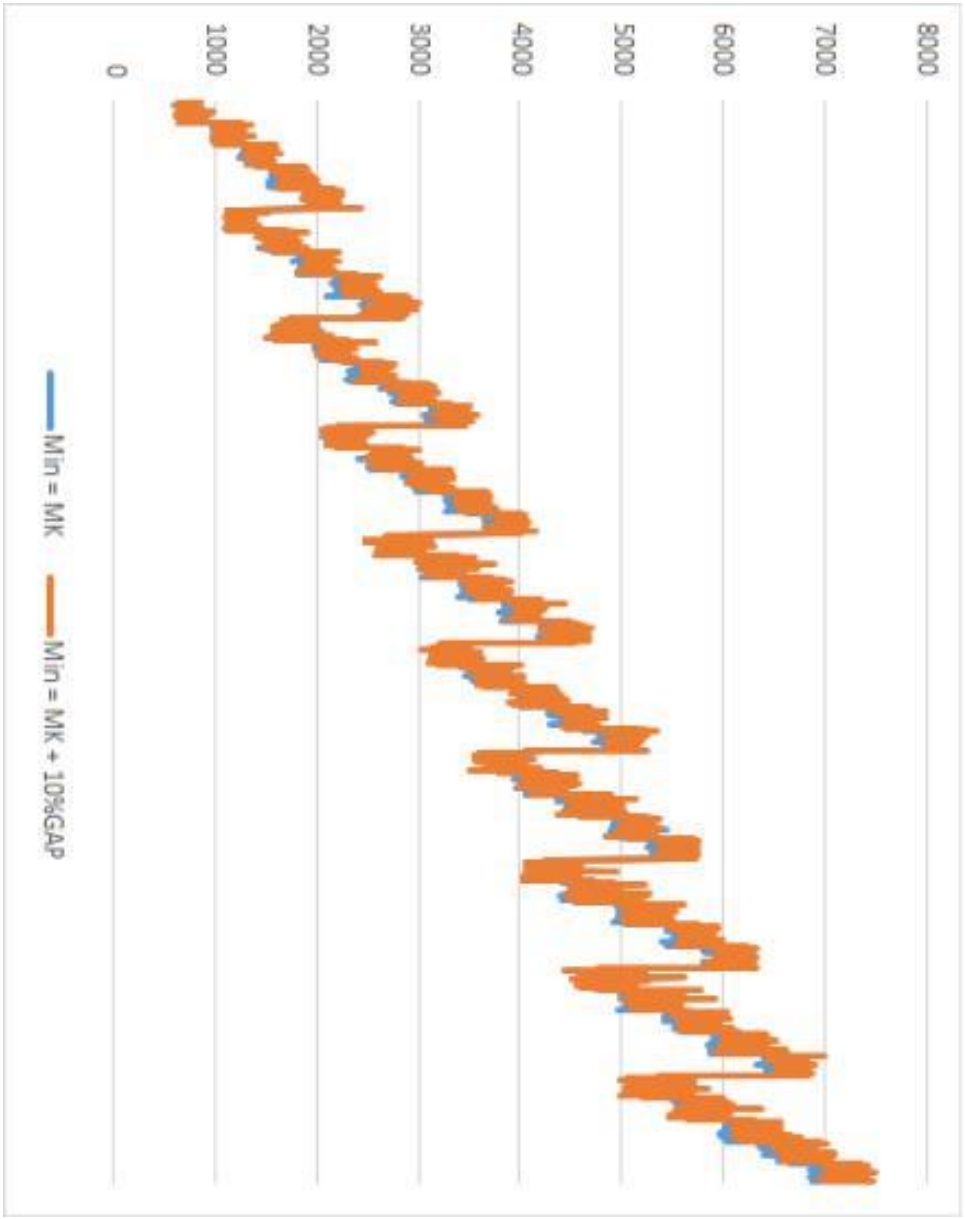
Figura 12 - Gráfico comparando os resultados de *makespan* obtidos pela heurística modificada com 1% e a Heurística original



Fonte: Autoria própria

- Heurística sem alterações x Heurística com 10% de impacto de *gap;*
  MK (Média) ≅ 3831 MK’ (Média) ≅ 3908

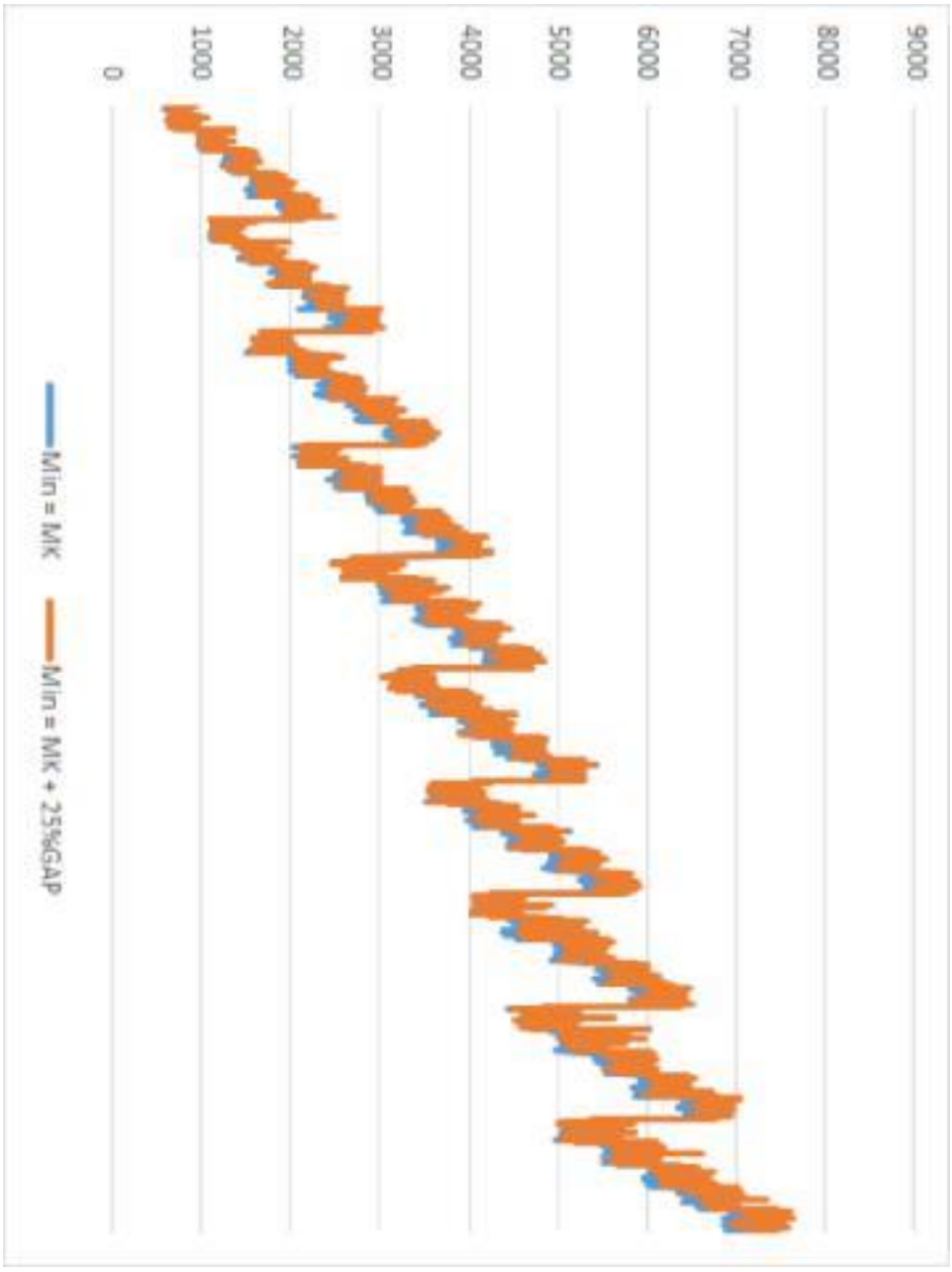
Figura 13 - Gráfico comparando os resultados de *makespan* obtidos pela Heurística modificada com 10% e a Heurística original



Fonte: Autoria própria

- Heurística sem alterações x Heurística com 25% de impacto de *gap;*
  MK (Média) ≅ 3831 MK’ (Média) ≅ 3962

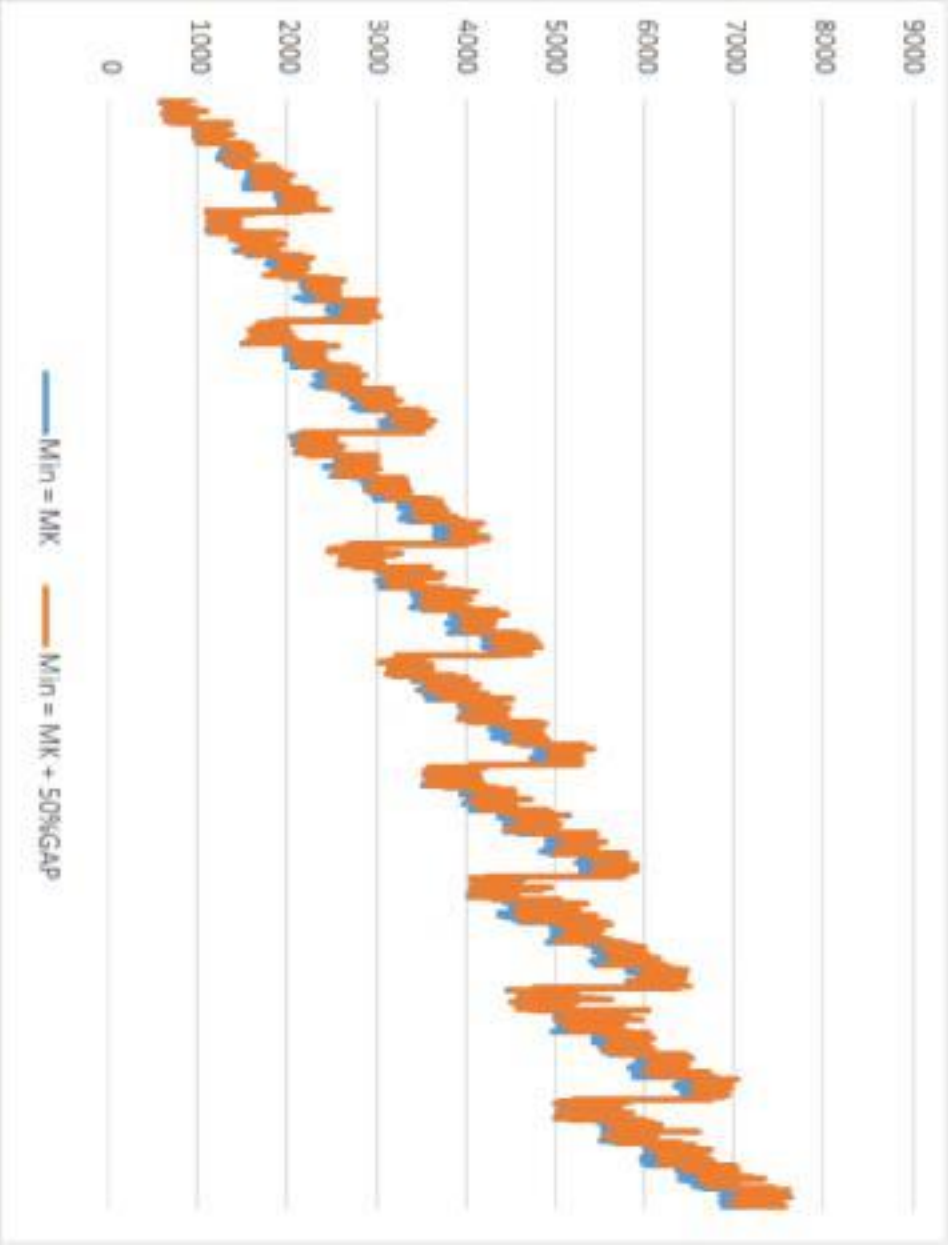
Figura 14 - Gráfico comparando os resultados de *makespan* obtidos pela heurística modificada com 25% e a heurística original



Fonte: Autoria própria

- Heurística sem alterações x Heurística com 25% de impacto de *gap;*
  MK (Média) ≅ 3831 MK’ (Média) ≅ 4005

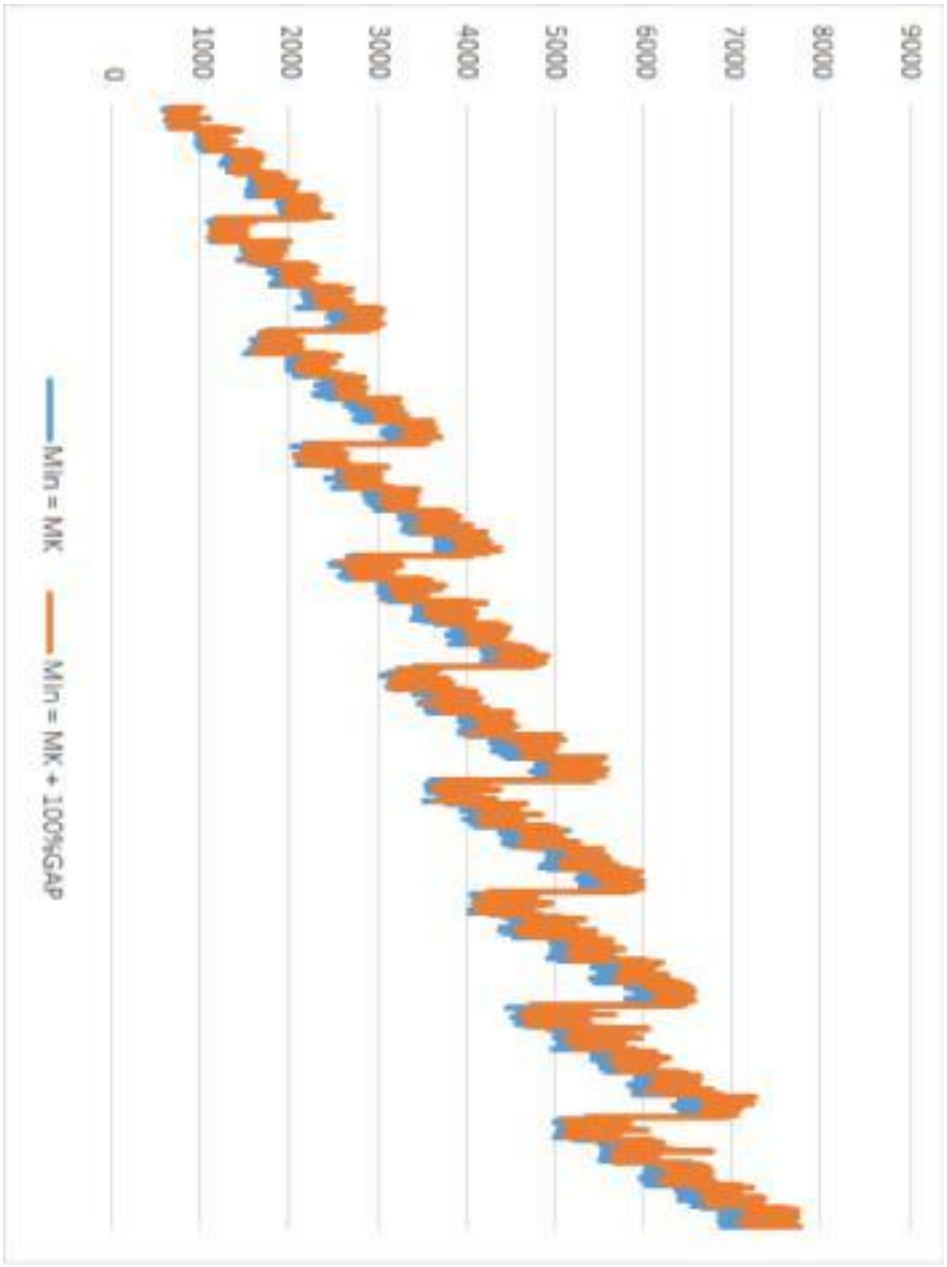
Figura 15 - Gráfico comparando os resultados de *makespan* obtidos pela Heurística modificada com 50% e a Heurística original



Fonte: Autoria própria

- Heurística sem alterações x Heurística com 100% de impacto de *gap;*
  MK (Média) ≅ 3831 MK’ (Média) ≅ 4039

Figura 16 - Gráfico comparando os resultados de *makespan* obtidos pela Heurística modificada com 100% e a Heurística original



Fonte: Autoria própria

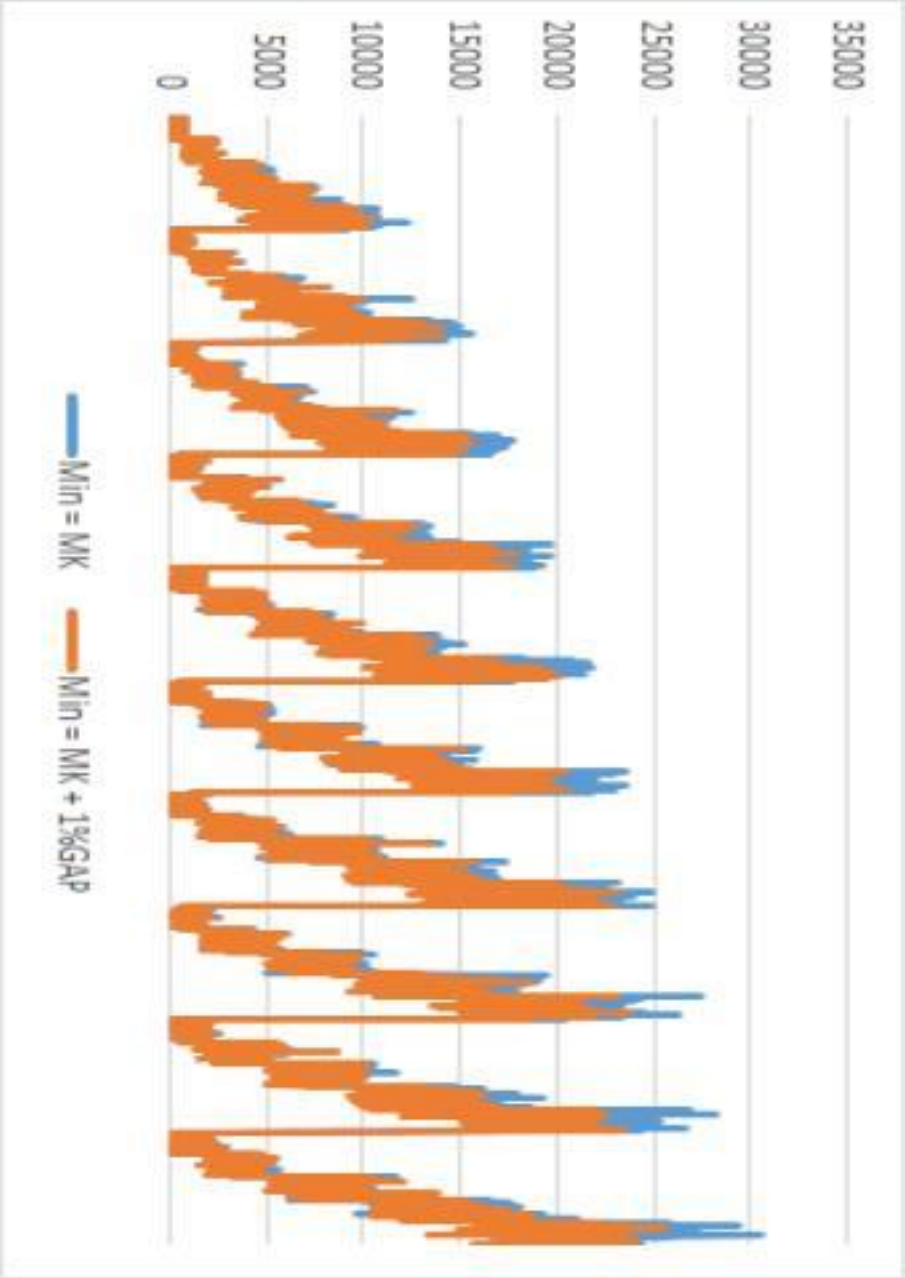
Nos gráficos acima, a linha azul representa os resultados da base de dados quando foi minimizado exclusivamente o *makespan* por meio do método determinado (LPT + NEH). A linha laranja representa os valores de *makespan* da mesma base de dados, após a adição de uma porcentagem de *gap* na função objetivo. As médias MK e MK', representam os valores de *makespan* antes e após as alterações na função objetivo, consecutivamente.

No gráfico da Figura 12 as linhas se sobrepõem uma à outra, e indica que os valores de MK' sofreram um baixo impacto quando submetidos à adição do impacto de 1% de *gap* na função objetivo. As médias de MK e MK', inclusive sugerem um impacto positivo, ou seja, devido ao método de "tentativa e erro" condicionado por diversos métodos heurísticos, em diversas situações dentro da base de dados foram encontrados valores menores de MK' em relação ao MK, de modo que a média para os 5000 valores foi ainda menor para MK'. Nas Figuras subsequentes (13, 14, 15 e 16), em que as parcelas atribuídas de *gap* foram maiores, é possível observar que os impactos são negativos, e eleva os valores pontuais e consequentemente a média do MK' comparando-se com os resultados de MK.

A partir do mesmo mecanismo foram construídos mais cinco gráficos, representados nas Figuras 17 à 21, de modo a analisar o comportamento do *gap* dentro da base de dados proposta. Portanto, o objetivo dos gráficos ilustrados a seguir é de indicar se, houve ou não, redução significativa em relação aos valores de *gap*.

- Heurística sem alterações x Heurística com 1% de impacto de *gap;*
  GAP (Média) ≅ 7646 GAP’ (Média) ≅ 6844

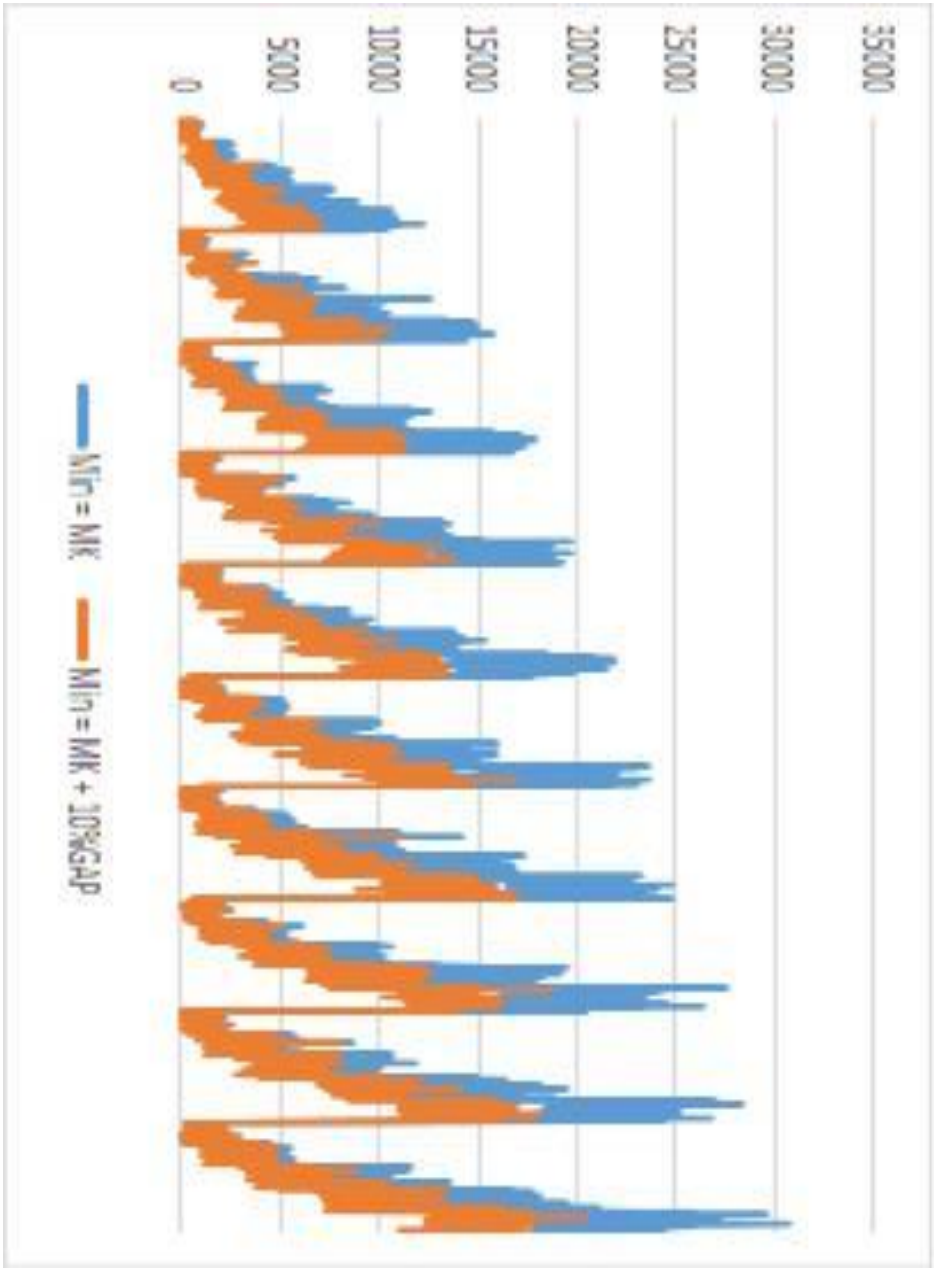
Figura 17 - Gráfico comparando os resultados de *gap* obtidos pela Heurística modificada com 1% e a Heurística original



Fonte: Autoria própria

- Heurística sem alterações x Heurística com 10% de impacto de *gap;*
  GAP (Média) ≅ 7646 GAP' (Média) ≅ 4955

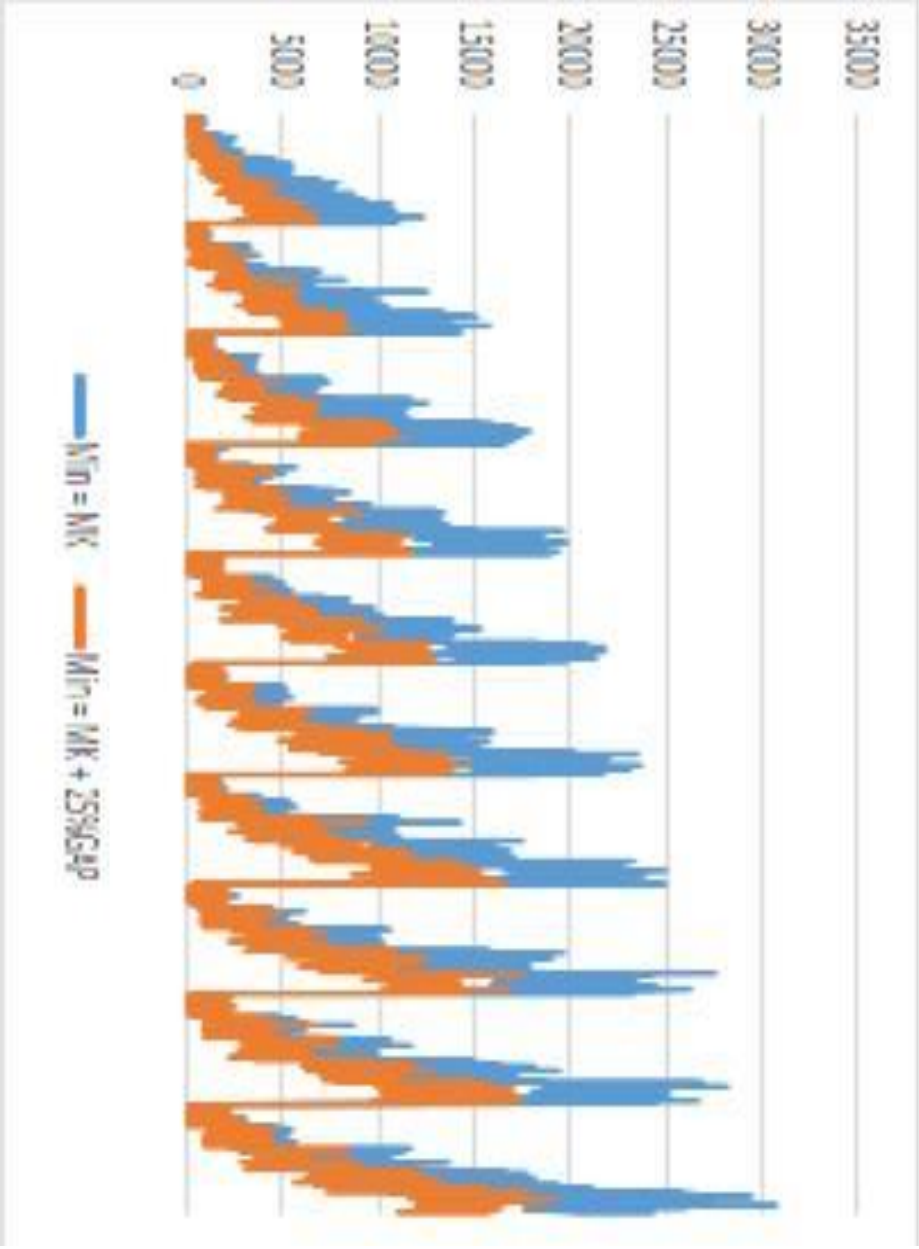
Figura 18 - Gráfico comparando os resultados de *gap* obtidos pela Heurística modificada com 10% e a Heurística original



Fonte: Autoria própria

- Heurística sem alterações x Heurística com 25% de impacto de *gap;*
  GAP (Média) ≅ 7646 GAP' (Média) ≅ 4607

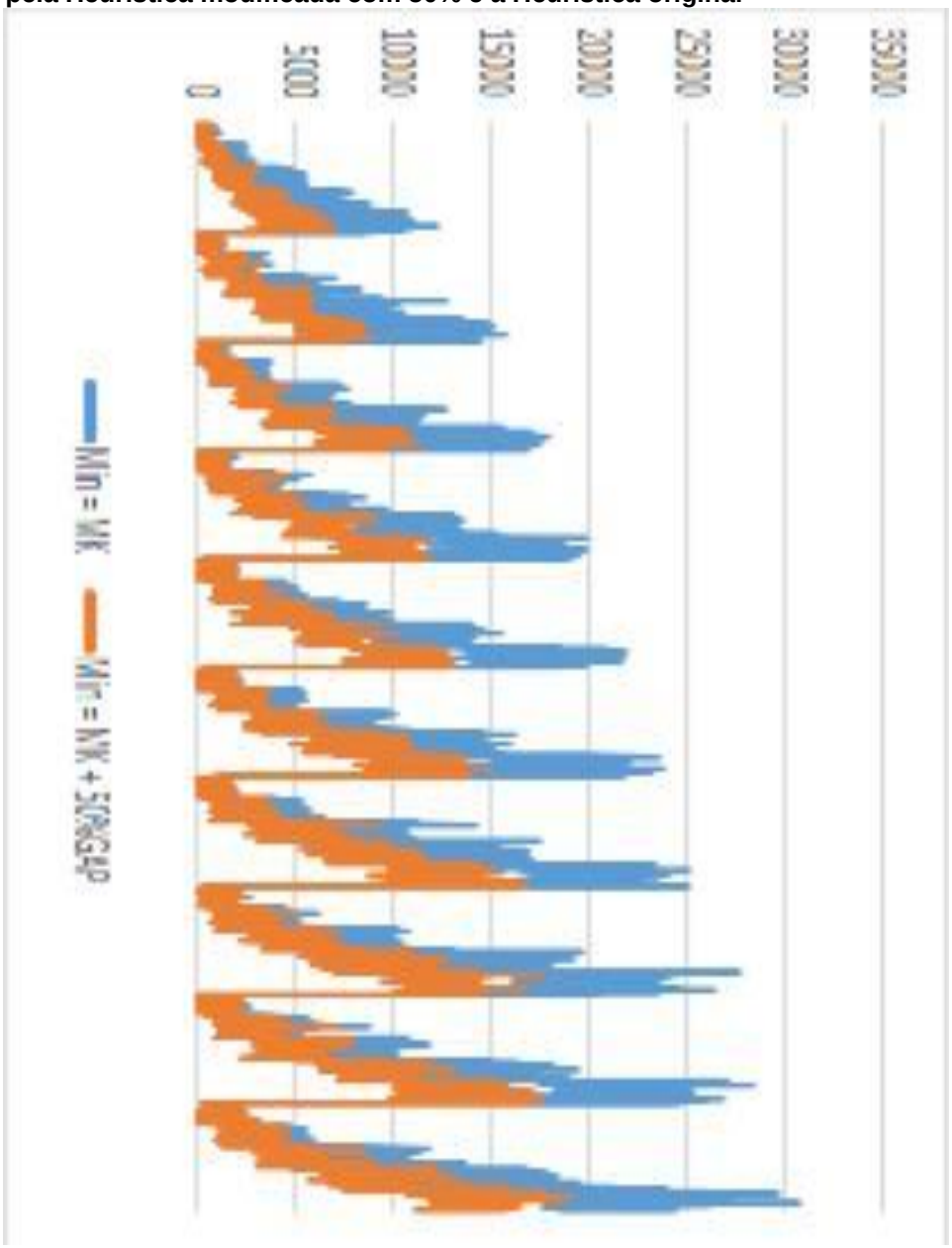
Figura 19 - Gráfico comparando os resultados de *gap* obtidos pela Heurística modificada com 25% e a Heurística original



Fonte: Autoria própria

- Heurística sem alterações x Heurística com 50% de impacto de *gap;*
  GAP (Média) ≅ 7646 GAP' (Média) ≅ 4485

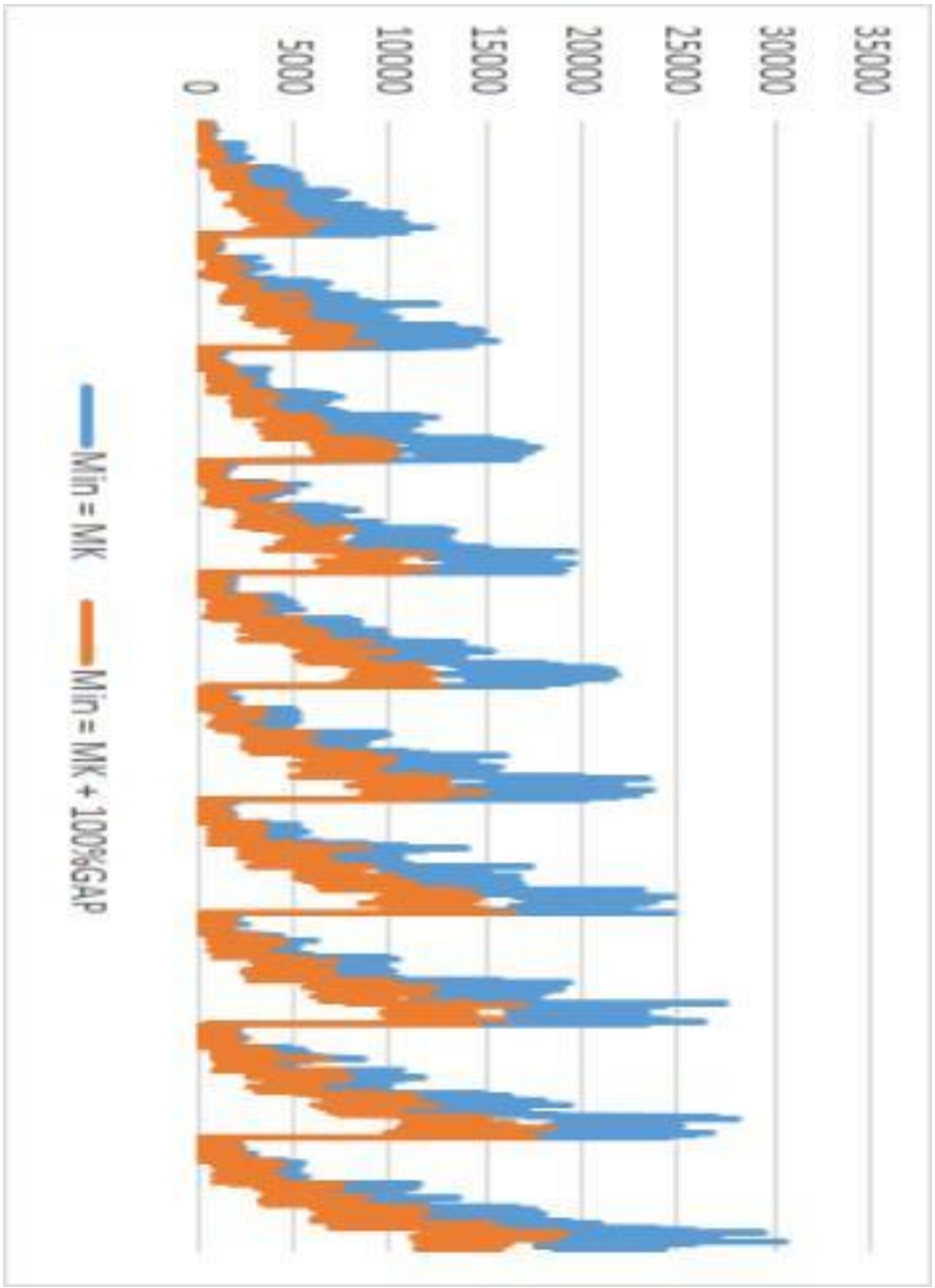
Figura 20 - Gráfico comparando os resultados de *gap* obtidos pela Heurística modificada com 50% e a Heurística original



Fonte: Autoria própria

- Heurística sem alterações x Heurística com 100% de impacto de *gap;*
  GAP (Média) ≅ 7646 GAP’ (Média) ≅ 4447

Figura 21 - Gráfico comparando os resultados de *gap* obtidos pela Heurística modificada com 100% e a Heurística original

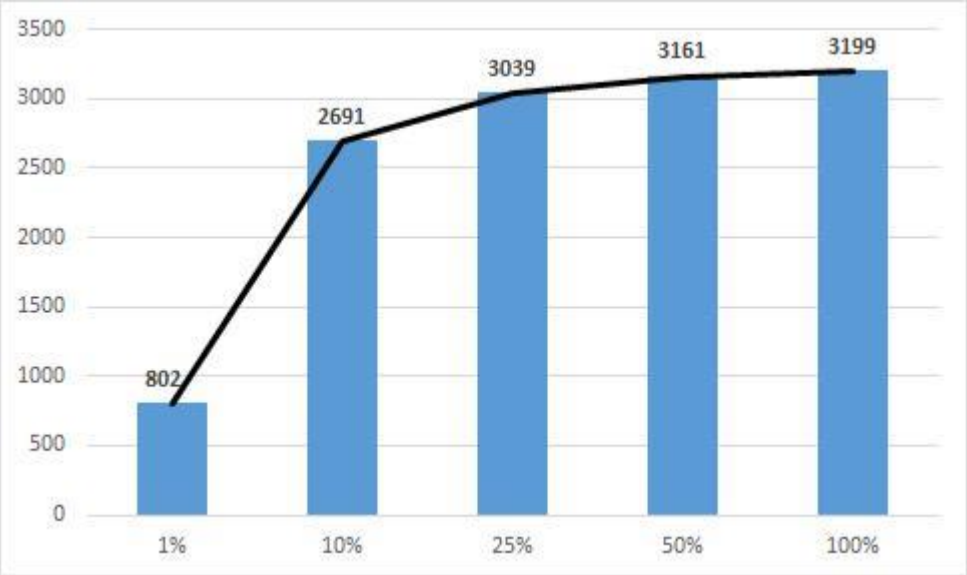


Fonte: Autoria própria

De maneira similar aos gráficos das Figuras 12 à 16, os gráficos das Figuras 17 à 21 apresentam uma linha azul e uma laranja, onde a primeira é composta pelos valores individuais de *gap* anterior à sua utilização na função objetivo e, a segunda é composta pelos valores individuais de *gap* posterior à adição de uma porcentagem específica do *gap* na função objetivo da heurística.

Já no gráfico da Figura 17, é possível observar uma redução significativa nos valores individuais de *gap*, mesmo utilizando-se apenas 1% de *gap* em relação ao *makespan.* A média final dada por GAP' representa uma redução de aproximadamente 800 unidades de tempo em relação ao GAP, para cada condição dentro da base de dados. Conforme aumentou-se a porcentagem do *gap* adicionada à função objetivo, as reduções foram ainda mais significativas, e chega à aproximadamente 3200 unidades de tempo. Embora haja redução, conforme há aumento da parcela do *gap,* é possível perceber que as reduções se tornam menos significantes, de acordo com os resultados exibidos nos gráficos das Figuras 18, 19, 20 e 21.

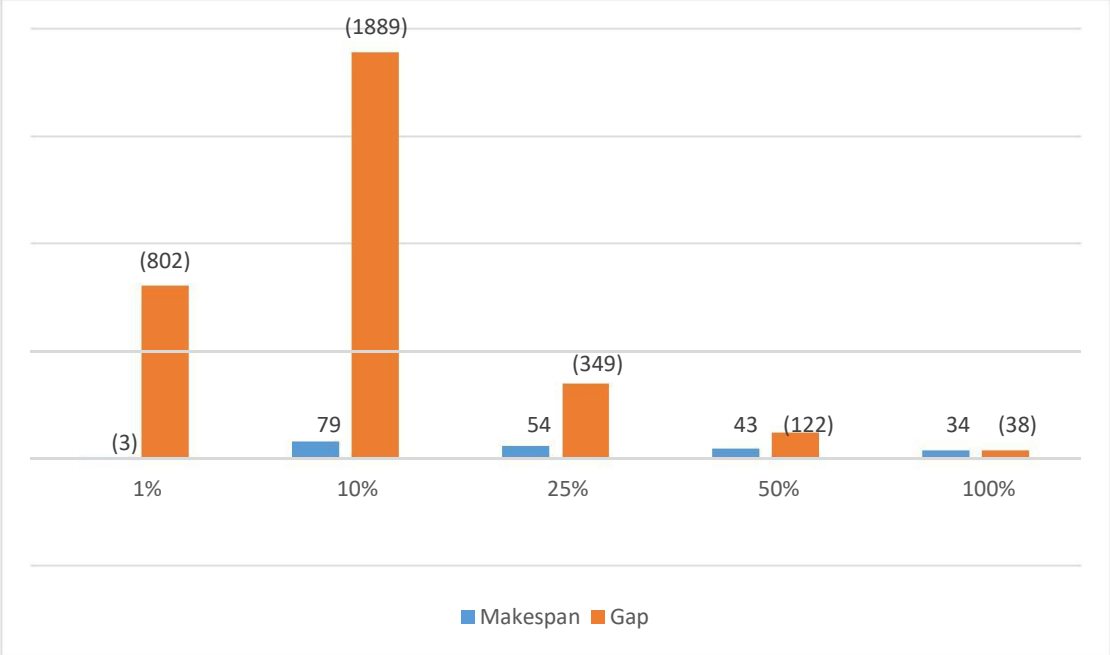
Figura 22 - Gráfico que mostra a diferença média de cada Heurística modificada em relação à Heurística original



Fonte: Autoria própria

Com os dados do gráfico da Figura 22, é possível perceber claramente que o impacto gerado na redução do *gap* perde eficiência a partir da adição de 25% do *gap* na função objetivo. De modo grosseiro, pode-se afirmar que a aplicação do *gap* é vantajosa até os 10%, quando os resultados ainda apresentam elevado potencial de eficiência para um cenário de produção onde o *gap* nas máquinas é relevante. Porém, os resultados posteriores ainda apresentam reduções que podem contribuir em resultados significativos, o que depende da relevância que o *gap* possua em um cenário real. Na Figura 23 é possível visualizar o aumento da média MK' e redução de GAP' entre uma experimentação e outra. Embora o impacto em MK' apresente o mesmo comportamento decrescente do impacto em GAP', o mesmo apresenta uma queda muito menos expressiva entre os experimentos, sendo possível estimar que em algum ponto próximo na qual a porcentagem do *gap* aplicada na função objetivo ultrapasse 100%, haverá um aumento de MK' numericamente maior que a redução de GAP'.

Figura 23 - Gráfico mostrando comparação entra as diferenças médias obtidas pela Heurística modificada em relação à Heurística original na primeira coluna, e em seguida com a porcentagem seguinte do teste



Fonte: Autoria própria

5.1.1 Variação de Acordo com a Quantidade de Tarefas

Uma vez que a heurística proposta atendeu a expectativa de redução considerável de *gap* gera pouco prejuízo ao *makespan*, é preciso entender se a variação da quantidade de tarefas (10 à 100) influencia na qualidade dos resultados. Ou seja, nesse tópico é abordada a eficiência da heurística proposta em função da quantidade de tarefas.

Para o desenvolvimento das análises necessárias, executou-se o cálculo da diferença média em porcentagem, em que os resultados das heurísticas propostas foram comparados aos resultados da heurística comum, por meio das seguintes fórmulas:

$$\%MK' \text{ Média} = \frac{\sum 100(\ '-\ )}{} \quad (3)$$

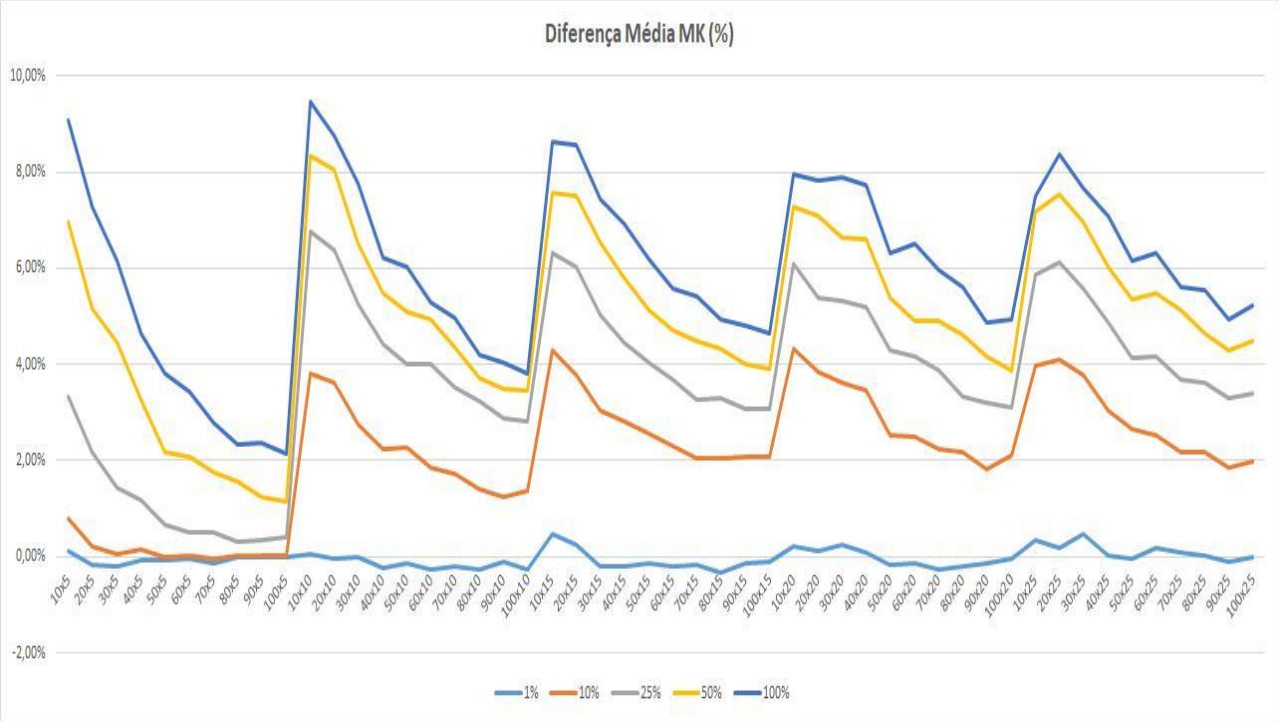
%MK' Média - diferença média em porcentagem do *makespan* para cada grupo de tarefa x máquina {10x5; 10x10; …; 10x25; 20x5; ….. ;100x25};
MK' - *Makespan* da função objetivo modificada;
MK - *Makespan* da função objetivo original.

$$\%GAP' \text{ Média} = \frac{\sum 100}{}$$

%GAP' Média - diferença média em porcentagem do *gap* para cada grupo de tarefa x máquina {10x5; 10x10; …; 10x25; 20x5; ….. ;100x25};
GAP' - *Gap* da função objetivo modificada;
GAP - *Gap* da função objetivo original.

Os resultados dos cálculos obtidos por meio das fórmulas acima, podem ser observados individualmente no Apêndice B. Estes valores foram plotados simultaneamente em um único gráfico para permitir a visualização de tendências dos resultados para o *makespan,* de acordo com a variação da quantidade de tarefas.

Figura 24 - Gráfico da Diferença Média do *makespan* em porcentagem com variação da quantidade de tarefas

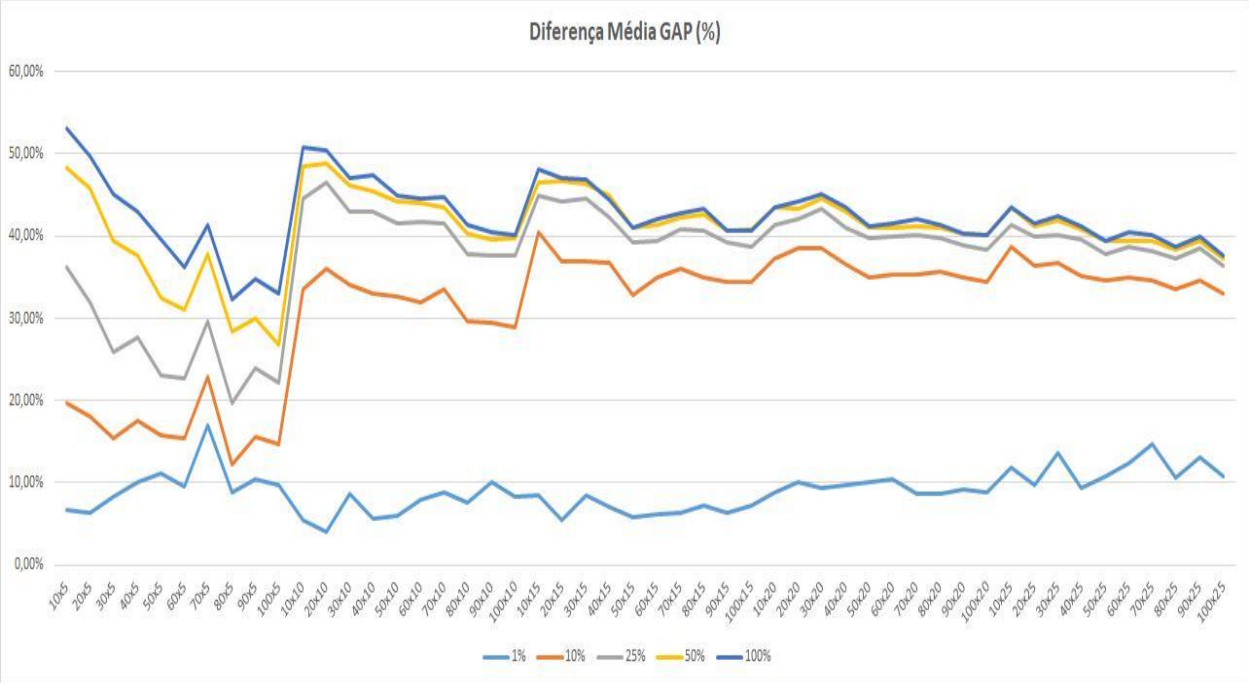


Fonte: Autoria própria

No gráfico da Figura 24 é possível observar que a modificação da função objetivo apresentou uma redução da diferença média de *makespan* conforme houve aumento da quantidade de tarefas, com exceção da primeira aplicação de apenas 1% do *gap,* no qual o comportamento foi variável. O resultado para as demais aplicações exibe uma diluição maior do impacto no *makespan* quando o número de tarefas é maior. Isso pode ser explicado por uma eventual facilidade ao se encontrar valores de MK' mais próximos de MK, uma vez que a heurística trabalha com um conjunto de combinações mais extenso.

Do mesmo modo, os valores da diferença média em porcentagem de *gap* foram plotados e a os resultados apresentados na Figura 25.

Figura 25 - Gráfico da Diferença Média do *gap* em porcentagem com variação da quantidade de tarefas



Fonte: Autoria própria

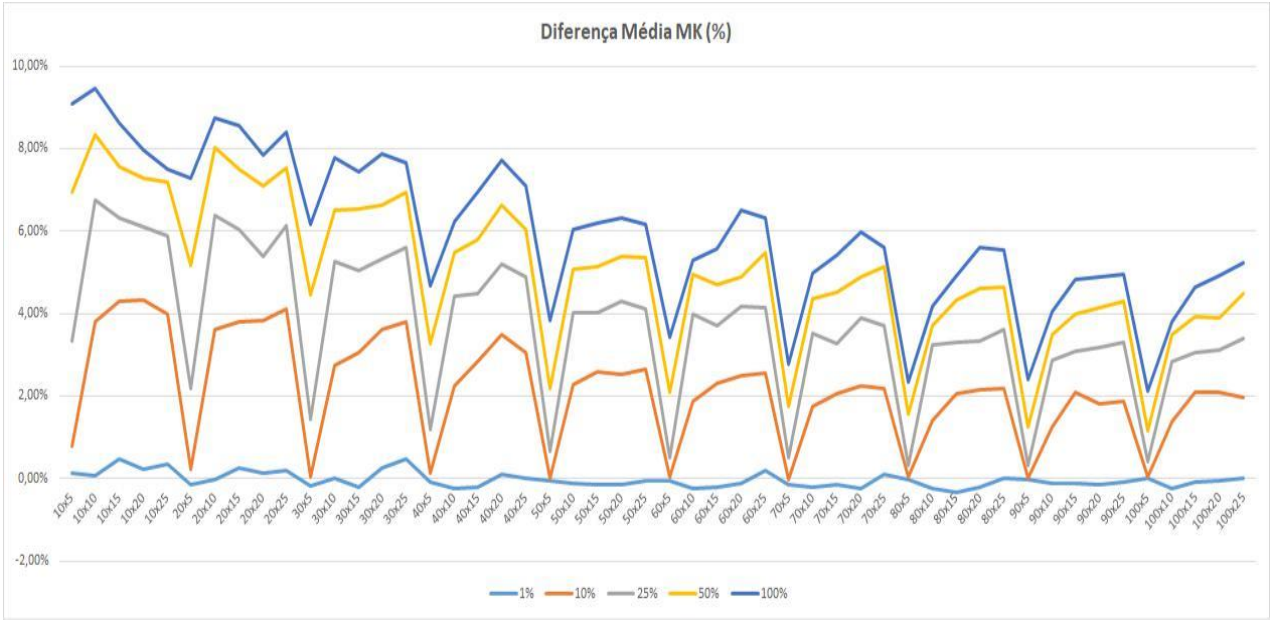
Novamente, a aplicação de 1% do *gap* apresentou um comportamento distinto das demais aplicações. Para essas aplicações, identifica-se que a diferença média em porcentagem de *gap* é mais sensível ao aumento do número de tarefas, quando o número de máquinas é menor. Conforme aumenta-se a quantidade de máquinas, as médias máximas (10 tarefas) e mínimas (100 tarefas) se aproximam, e mostra uma curva menos acentuada no final do gráfico da Figura 25.

5.1.2 Variação das Máquinas

Em seguida foi analisado como a variação da quantidade de máquinas (5 à 25) influencia e altera o comportamento dos resultados da heurística proposta. Utilizando os mesmos cálculos citados no tópico 4.1.1 foram construídos gráficos que ilustram o comportamento da variação da quantidade de máquinas nos resultados obtidos.

Para os resultados da diferença média em porcentagem do *makespan*, utilizando todos os valores individuais em um único gráfico organizado de forma a enxergar as possíveis tendências causadas pela variação da quantidade de máquinas, foi gerado o gráfico da Figura 26.

Figura 26 - Gráfico da Diferença Média do makespan em porcentagem com variação da quantidade de máquinas



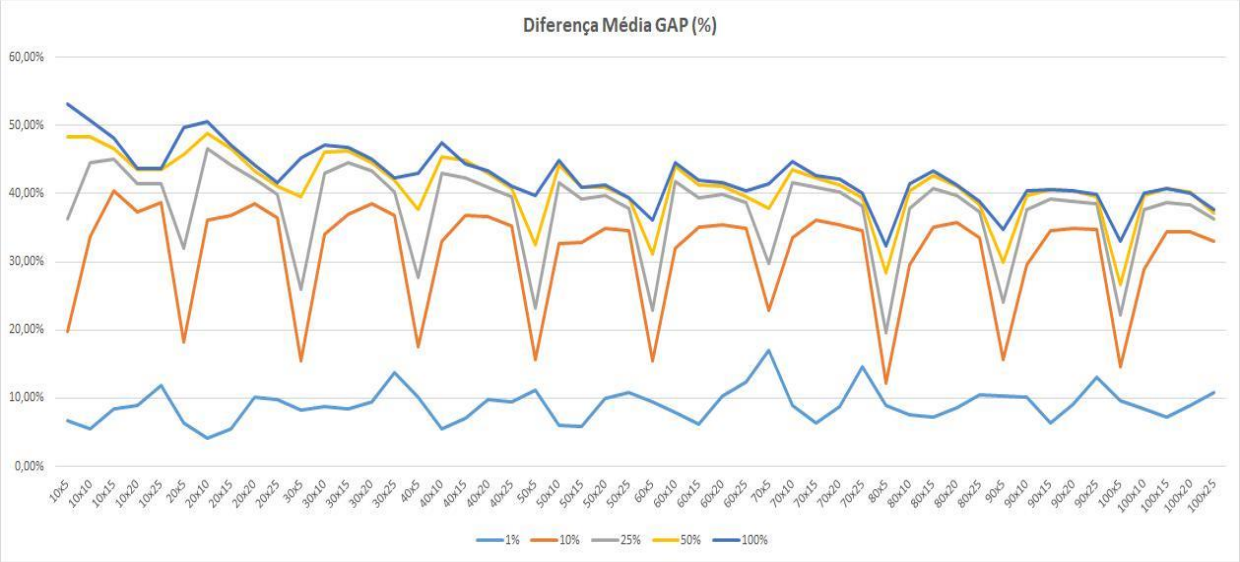
Fonte: Autoria própria

A partir do gráfico da Figura 26 nota-se que conforme a quantidade de máquinas aumenta, a porcentagem de diferença média do *makespan* também aumenta, de forma padrão para todas as aplicações de porcentagem do *gap*, exceto para aplicação de 1%, que assim como no tópico anterior exibe um comportamento variável.

Mesmo seguindo um padrão dentro das diferentes aplicações do *gap*, existem alguns casos em que a porcentagem de diferença média de *makespan* diminui mesmo ao aumentar o número de máquinas como por exemplo no ambiente de 10 tarefas que os valores sempre diminuem conforme a quantidade de máquinas aumenta, ou no caso que testa o ambiente de 20 tarefas e 15 máquinas, que tem uma porcentagem de diferença média de *makespan* menor que o caso de 20 tarefas e 10 máquinas e maior que o de 20 tarefas e 20 máquinas, e adquire um comportamento contrário à tendência do gráfico.

Um outro gráfico foi plotado para analisar o efeito de variação nas quantidades de máquinas para a porcentagem de diferença média do *gap*, conforme mostra a Figura 27.

Figura 27 - Gráfico da Diferença Média do *gap* em porcentagem com variação da quantidade de máquinas



Fonte: Autoria própria

Mais uma vez a Figura 27 mostra que a aplicação de 1% de *gap* resulta em uma variabilidade elevada nos seus resultados, e não apresenta um comportamento padrão. Para a aplicação de 10% de *gap* é possível observar um padrão nos

resultados, porém este ainda difere do padrão encontrado nas demais aplicações, que apresentam resultados com um padrão muito similar.

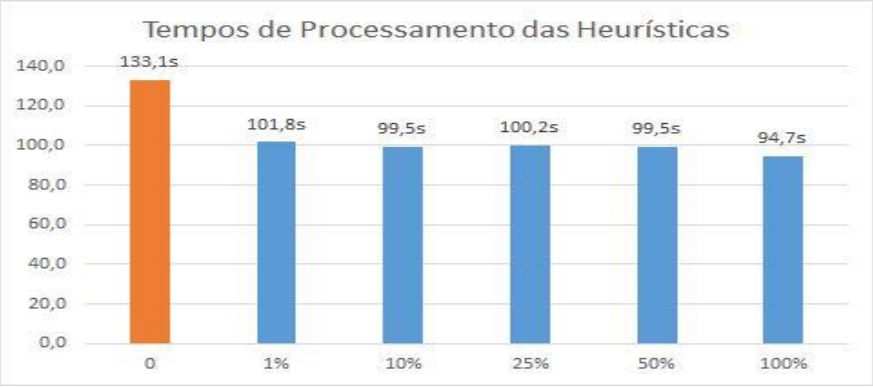
No caso da aplicação de 10% de *gap* pode-se observar uma tendência de aumento da porcentagem de diferença média de *gap* conforme o número de máquinas aumenta. Nas demais aplicações, exceto 1% de *gap*, pode-se notar que existe uma tendência de diminuição na porcentagem de diferença média de *gap* à medida que aumenta-se o número de máquinas.

5.2 TEMPO DE PROCESSAMENTO COMPUTACIONAL

Em relação ao tempo de processamento computacional requerido para cada cenário da base de dados, identifica-se um acréscimo natural de acordo com o aumento do número de tarefas x máquinas. No Anexo A encontram-se as especificações da máquina utilizada para execução do algoritmo.

Quanto ao tempo utilizado pelas heurísticas modificadas para realizar o processamento total da base de dados, houve uma redução considerável ao aplicar as porcentagens do *gap* na heurística original, o que pode ser visualizado no gráfico da Figura 28.

Figura 28 - Gráfico de Tempos de Processamento das Heurísticas



Fonte: Autoria própria

Observa-se na Figura 28, uma redução de aproximadamente 30 segundos, quando adicionada uma parcela de 1% do *gap* na função objetivo, a qual se mantém nas experimentações subsequentes.

6 CONCLUSÕES

Os resultados exibidos no item 5.1 credenciam a relevância da proposta desenvolvida nesse trabalho. Embora os resultados sejam numéricos, não é possível realizar um julgamento sobre qual aplicação do *gap* na função objetivo é mais eficaz, uma vez que houve melhoria quanto à redução de *gap* entre uma experimentação e outra e, por outro lado houve um impacto maior quanto ao prejuízo no *makespan*. Esse julgamento seria possível se houvesse valores definidos, como por exemplo, o lucro obtido por unidade produzida e o custo por unidade de tempo ao manter uma máquina ligada.

Quando adiciona-se 1% do *gap* na minimização, os resultados apresentam vantagens além das esperadas, como a redução na média geral do *makespan*, além de uma redução no tempo de implementação da base de dados. Portanto, esses ganhos podem ser considerados, tanto para contribuir com uma eficiência melhor de um processo, quanto para redução de tempo computacional utilizado pelas heurísticas que fazem uso do método NEH. Embora as aplicações subsequentes não apresentem os mesmos ganhos adicionais, apresentam vantagem em relação à uma contribuição maior na redução da média final do *gap*.

No item 5.1.1, observa-se que o aumento do número de tarefas tende a gerar menos impacto no acréscimo do *makespan* final entre a heurística proposta e a original, e também existe uma perda de eficácia quanto à redução do *gap*. Ainda assim, não se descarta a utilização dos métodos propostos para um universo com até 100 tarefas, uma vez que a redução de *gap* nas máquinas, ainda é bastante superior ao aumento de *makespan*.

Sobre a variação da quantidade de máquinas, exibida no item 5.1.2, os resultados apresentam comportamento variável, porém os valores da diferença média de *gap* e *makespan* indicam variações menos bruscas comparados ao incremento do número de tarefas.

O tempo de implementação das heurísticas trabalhadas são razoáveis para aplicação prática, variando entre 1min35seg à 2min10seg com uma base de dados de 5000 cenários diferentes.

As análises e resultados do método proposto, indicam que este trabalho pode ser utilizado em diversas situações práticas, como projetos de redução de custo, redução de consumo de recursos (água, energia, etc.), aumento da vida útil de equipamentos, entre diversas outras possibilidades em que um equipamento deve se manter em funcionamento entre uma tarefa e outra.

O estudo abordado no presente trabalho, pode ser continuado por meio de combinações com outros métodos heurísticos de minimização e/ou funções objetivo, com o intuito de verificar se resultados semelhantes podem ser extraídos. Outra possibilidade de continuidade é a aplicação em um estudo de caso, que será possível dimensionar um “peso” ideal para o *gap* aplicado na função objetivo.

REFERÊNCIAS

BAKER, K. R. **Introduction to sequencing and scheduling**. 1. ed. New York: John Wiley & Sons, 1974.

BAKER, K. R.; TRIETSCH, D. **Principles of Sequencing and Scheduling**. 1. ed. New York: John Wiley & Sons, 2009.

CAMPELLO, C. R., MACULAN, N. **Algoritmos e Heurísticas**. 1. ed. Niterói: Editora da Universidade Federal Fluminense, 1994.

CORDENONSI, A. Z. **Ambientes, Objetos e Dialogicidade. Uma Estratégia de Ensino Superior em Heurísticas e Metaheurísticas.** Tese (Doutorado). Programa de Pós-Graduação em informática na educação. Universidade Federal do Rio Grande do Sul. Porto Alegre, 2008.

FRAMINAN, J. M.; LEISTEN, R.; RUIZ-USANO, R**.** Comparison of heuristics for flowtime minimization in permutation flowshops. **Computers and Operations Research,** v. 32, n. 5, p. 1237-1254, mar. 2005.

FUCHIGAMI, H. Y. **Métodos heurísticos construtivos para o problema de programação da produção em sistemas flowshop híbridos com tempos de preparação das máquinas assimétricos e dependentes da sequência.** Dissertação (Mestrado em Engenharia de Produção) - Escola de Engenharia de São Carlos, Universidade São Paulo, São Carlos, 2005.

GAREY, M.R.; JOHNSON, D.S. & SETHI, R. The Complexity of Flowshop and Jobshop Scheduling. **Mathematics of Operations Research,** v.1, n. 2, p. 117-129, mar. 1976.

GIL, A. C. **Métodos e técnicas de pesquisa social**. 6. ed. São Paulo: Atlas, p. 42. 1999.

GUPTA, J N D; STAFFORD, E F. Flowshop scheduling research after five decades. **European Journal of Operational Research***,* v. 169, n. 3, p. 699-711, mar. 2006.

JOHNSON, S. M. Optimal Two- and Three-Stage Production Schedules with Setup Times Included. **Naval research logistics quarterly**, v.1, n. 1, p. 61-68, mar. 1954.

LIU, J.; REEVES, C. R. Constructive and composite heuristic solutions to the P//∑ Ci scheduling problem. **European Journal of Operational Research**, v. 132, n. 2, p. 439-452, jun. 2001.

NAWAZ, M; ENSCORE, E E; HAM, I. A Heuristic Algorithm For The M-Machine, N-Job Flowshop Sequencing Problem. **Omega-International Journal of Management Science**, v. 1, n. 11, p. 91-95, jun. 1983.

PAN, Q.; RUIZ, R. A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. **Computers & Operations Research,** v. 40, n. 1, p. 117-128, jan. 2013.

PINEDO, M. L. **Scheduling: Theory, Algorithms and Systems**. 4. ed. New York: Springer, 2012.

RUIZ, R.; STUTZLE, T. A Simple And Effective Iterated Greedy Algorithm For The Permutation Flowshop Scheduling Problem. **European Journal of Operational Research**, v. 177, n. 3, p. 2033-2049, set. 2007.

RUIZ, R; MAROTO, C. A comprehensive review and evaluation of permutation flowshop heuristics. **European Journal of Operational Research**, v. 165, n. 2, p. 479-494, jan. 2005.

SLACK, N. et al. **Administração da Produção**. 1 ed. 2 tir. São Paulo: Atlas. Editora Compacta, 1999.

TASGETIREN, M. F. *et al*. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. **European Journal of Operational Research**, v. 177, n. 3, p. 1930-1947, mar. 2007.

APÊNDICE A – Código da Heurística Proposta

```
        program Heurística Proposta;
uses Windows, SysUtils;

type
arrn = array[0..200] of integer;
var i,ii, iii, j,jj,jjj,k,m,n,a,c,aa,bb,cc,dd,ee,xx,ax,ft,tempoinicial,tempofinal,mk,fof,tempo,gap,g,
ob2 : integer;
   ma,mb,d : array [0..200,0..200] of integer;
   s,t,w,w1,sp : array [0..200] of integer;
   arquivo, arquivo1, arq : text;

procedure fo(ax:arrn;b:integer);          //Modificação da F.O.
begin
  gap:= 0;
  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
       g := 0;
       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]];
       if mb[ii,jj-1] < mb[ii-1,jj] then g := mb[ii-1,jj] - mb[ii,jj-1];
       gap := gap + g;
     end;
  mk := mb[m,b];
  ob2 := gap;
  ft := 0;
  for ii := 1 to b do ft := ft + mb[m,ii];
  fof := 10*mk + 1*ob2; // Método utilizado para aplicação de 10% de gap na
f.o. end;
```

```
begin
assign(arquivo1,'adress');
rewrite(arquivo1);
for xx := 1 to 5000 do
begin
tempoinicial := GetTickCount;
assign(arquivo,'adress'+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;

writeln('Tarefas: ',n);
writeln('Maquinas: ',m);
{---------------------------------------------------------------}

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 for j := 1 to n do if t[i] < t[j] then s[i] := s[i] + 1;        //LPT < ou SPT >
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;
```

```
for i := 1 to n do w1[i] := w[i];

{2a. fase--NEH-----------------------------------------------------------------}
fo(w,2);
ax := fof;
w1[1] := w[2];
w1[2] := w[1];
fo(w1,2);
if fof >= ax then for i := 1 to 2 do w1[i] := w[i];
for i := 3 to n do w1[i] := w[i];
for i := 3 to n do
  begin
    ax := 0;
        fo(w1,i);
        ax := fof;
        for k := 1 to i do sp[k] := w1[k];
        for k := i downto 2 do
          begin
            aa := sp[k];
            sp[k] := sp[k-1];
            sp[k-1] := aa;
            fo(sp,i);
            if fof < ax then
              begin
                ax := fof;
                for aa := 1 to i do w1[aa] := sp[aa];
              end;
          end;
  end;

 {-------------------------------------------------------------------------}
writeln('Melhor sequencia encontrada:'); for i := 1 to n do
write(w1[i],' ');
```

```
fo(w1,n);
writeln;
writeln('Funcao – biobjetivo da sequencia: ',fof);
writeln('Makespan: ',mk);
writeln('GAP: ',gap);
{readln;} // Retira-se as chaves para observar cada
resultado close(arquivo);
tempofinal := GetTickCount;
tempo := tempofinal - tempoinicial;
writeln(arquivo1,fof,'-',tempo,'-',gap);
end;
close(arquivo1);
readln;
end.
```

APÊNDICE B – Tabela com os resultados médios por agrupamento

Diferença Média MK (%)						Diferença Média GAP (%)					
TxM	**1%**	**10%**	**25%**	**50%**	**100%**	**TxM**	**1%**	**10%**	**25%**	**50%**	**100%**
10x5	0,13%	0,79%	3,34%	6,95%	9,07%	**10x5**	6,63%	19,76%	36,28%	48,28%	53,15%
10x10	0,07%	3,81%	6,76%	8,34%	9,45%	**10x10**	5,44%	33,64%	44,48%	48,40%	50,72%
10x15	0,46%	4,29%	6,33%	7,57%	8,62%	**10x15**	8,42%	40,43%	44,97%	46,53%	48,06%
10x20	0,22%	4,32%	6,10%	7,29%	7,97%	**10x20**	8,89%	37,37%	41,38%	43,42%	43,59%
10x25	0,35%	3,97%	5,88%	7,17%	7,50%	**10x25**	11,82%	38,71%	41,42%	43,47%	43,59%
20x5	-0,16%	0,23%	2,18%	5,15%	7,28%	**20x5**	6,40%	18,14%	31,97%	45,81%	49,76%
20x10	-0,04%	3,61%	6,38%	8,04%	8,76%	**20x10**	4,08%	36,05%	46,52%	48,81%	50,48%
20x15	0,27%	3,79%	6,04%	7,50%	8,57%	**20x15**	5,43%	36,84%	44,24%	46,68%	47,10%
20x20	0,13%	3,83%	5,39%	7,09%	7,83%	**20x20**	10,16%	38,45%	42,17%	43,35%	44,14%
20x25	0,20%	4,11%	6,13%	7,54%	8,39%	**20x25**	9,72%	36,38%	39,96%	41,13%	41,57%
30x5	-0,18%	0,04%	1,43%	4,45%	6,15%	**30x5**	8,27%	15,46%	25,96%	39,50%	45,18%
30x10	0,00%	2,75%	5,27%	6,51%	7,76%	**30x10**	8,71%	34,03%	42,91%	46,15%	47,10%
30x15	-0,21%	3,05%	5,04%	6,53%	7,44%	**30x15**	8,50%	36,95%	44,58%	46,31%	46,84%
30x20	0,26%	3,62%	5,32%	6,62%	7,88%	**30x20**	9,40%	38,48%	43,28%	44,55%	45,03%
30x25	0,47%	3,80%	5,59%	6,94%	7,65%	**30x25**	13,71%	36,84%	40,20%	41,90%	42,36%
40x5	-0,08%	0,15%	1,19%	3,27%	4,65%	**40x5**	10,13%	17,48%	27,65%	37,60%	42,92%
40x10	-0,24%	2,25%	4,42%	5,48%	6,23%	**40x10**	5,57%	32,96%	43,05%	45,45%	47,41%
40x15	-0,20%	2,83%	4,47%	5,79%	6,93%	**40x15**	7,05%	36,82%	42,26%	44,94%	44,33%
40x20	0,09%	3,47%	5,20%	6,62%	7,72%	**40x20**	9,79%	36,57%	41,01%	42,95%	43,42%
40x25	0,01%	3,04%	4,88%	6,04%	7,10%	**40x25**	9,40%	35,21%	39,57%	40,75%	41,15%
50x5	-0,06%	0,00%	0,67%	2,18%	3,83%	**50x5**	11,21%	15,72%	23,15%	32,42%	39,65%
50x10	-0,13%	2,26%	4,01%	5,08%	6,04%	**50x10**	6,04%	32,74%	41,60%	44,16%	44,95%
50x15	-0,15%	2,58%	4,03%	5,13%	6,19%	**50x15**	5,79%	32,85%	39,23%	40,98%	41,00%
50x20	-0,16%	2,52%	4,28%	5,37%	6,31%	**50x20**	10,02%	34,93%	39,78%	40,98%	41,24%
50x25	-0,04%	2,66%	4,12%	5,36%	6,17%	**50x25**	10,88%	34,58%	37,79%	39,37%	39,46%
60x5	-0,05%	0,04%	0,52%	2,08%	3,42%	**60x5**	9,49%	15,42%	22,79%	31,10%	36,16%
60x10	-0,25%	1,86%	4,00%	4,94%	5,30%	**60x10**	7,90%	31,97%	41,79%	44,02%	44,56%
60x15	-0,21%	2,30%	3,70%	4,71%	5,58%	**60x15**	6,19%	35,05%	39,42%	41,34%	42,03%
60x20	-0,12%	2,50%	4,17%	4,89%	6,50%	**60x20**	10,37%	35,41%	39,97%	41,02%	41,57%
60x25	0,20%	2,54%	4,15%	5,48%	6,30%	**60x25**	12,32%	34,95%	38,67%	39,49%	40,43%
70x5	-0,14%	-0,03%	0,52%	1,75%	2,77%	**70x5**	17,07%	22,91%	29,71%	37,89%	41,38%
70x10	-0,20%	1,74%	3,52%	4,36%	4,98%	**70x10**	8,87%	33,51%	41,54%	43,50%	44,66%
70x15	-0,16%	2,06%	3,27%	4,50%	5,42%	**70x15**	6,36%	36,05%	40,86%	42,23%	42,72%
70x20	-0,25%	2,25%	3,88%	4,90%	5,97%	**70x20**	8,75%	35,40%	40,18%	41,24%	42,06%
70x25	0,10%	2,18%	3,70%	5,14%	5,62%	**70x25**	14,63%	34,55%	38,18%	39,45%	40,05%
80x5	-0,01%	0,03%	0,32%	1,56%	2,34%	**80x5**	8,92%	12,27%	19,68%	28,44%	32,32%
80x10	-0,25%	1,40%	3,25%	3,71%	4,19%	**80x10**	7,56%	29,66%	37,83%	40,34%	41,45%
80x15	-0,34%	2,05%	3,29%	4,32%	4,93%	**80x15**	7,22%	35,01%	40,75%	42,67%	43,35%
80x20	-0,21%	2,17%	3,34%	4,61%	5,61%	**80x20**	8,59%	35,75%	39,80%	41,05%	41,35%
80x25	0,02%	2,19%	3,62%	4,65%	5,53%	**80x25**	10,57%	33,55%	37,31%	38,43%	38,79%
90x5	-0,01%	0,02%	0,33%	1,26%	2,38%	**90x5**	10,37%	15,57%	24,01%	29,92%	34,79%
90x10	-0,11%	1,24%	2,88%	3,50%	4,04%	**90x10**	10,11%	29,54%	37,65%	39,66%	40,44%
90x15	-0,12%	2,08%	3,08%	3,99%	4,81%	**90x15**	6,42%	34,49%	39,22%	40,60%	40,61%
90x20	-0,14%	1,82%	3,19%	4,15%	4,88%	**90x20**	9,15%	34,98%	38,85%	40,33%	40,33%
90x25	-0,09%	1,86%	3,30%	4,29%	4,94%	**90x25**	13,12%	34,70%	38,45%	39,50%	39,88%
100x5	-0,01%	0,02%	0,42%	1,16%	2,13%	**100x5**	9,68%	14,65%	22,17%	26,74%	32,94%
100x10	-0,25%	1,37%	2,83%	3,47%	3,80%	**100x10**	8,34%	28,88%	37,59%	39,72%	40,06%
100x15	-0,10%	2,10%	3,06%	3,93%	4,65%	**100x15**	7,23%	34,42%	38,65%	40,78%	40,72%
100x20	-0,04%	2,10%	3,12%	3,89%	4,93%	**100x20**	8,87%	34,40%	38,30%	40,16%	40,10%
100x25	0,00%	1,97%	3,39%	4,47%	5,23%	**100x25**	10,77%	33,06%	36,30%	37,22%	37,62%

ANEXO A – Especificações do hardware utilizado para execução das heurísticas testadas

Edição do Windows

Windows 10 Pro

Sistema

Processador:	Intel(R) Core(TM) i7-4700HQ CPU @ 2.40GHz 2.40 GHz
Memória instalada (RAM):	8,00 GB
Tipo de sistema:	Sistema Operacional de 64 bits, processador com base em x64
Caneta e Toque:	Nenhuma Entrada à Caneta ou por Toque está disponível para este vídeo