UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO

CURSO DE ENGENHARIA DE PRODUÇÃO

EDUARDO TOMIO KEMURA KUMAGAI

AVALIAÇÃO DE MÉTODOS HEURÍSTICOS PARA PROBLEMAS *FLOWSHOP* PERMUTACIONAL DISTRIBUÍDO COM MINIMIZAÇÃO DO TEMPO TOTAL DE FLUXO E DURAÇÃO TOTAL DA PROGRAMAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2018

EDUARDO TOMIO KEMURA KUMAGAI

AVALIAÇÃO DE MÉTODOS HEURÍSTICOS PARA PROBLEMAS *FLOWSHOP* PERMUTACIONAL DISTRIBUÍDO COM MINIMIZAÇÃO DO TEMPO TOTAL DE FLUXO E DURAÇÃO TOTAL DA PROGRAMAÇÃO

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

Orientador: Prof. Dr. Fábio José Ceron Branco

PONTA GROSSA

2018

	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 para problemas flowshop permutacional distribuído com minimização do tempo total de fluxo e duração total da programação

por

Eduardo Tomio Kemura Kumagai

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 22 de junho de 2018 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. Orientador

_____
Prof. Dr. Shih Yung Chin
Membro titular

_____
Profª. Dra. Yslene Rocha Kachba
Membro titular

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

Dedico este trabalho à minha família, pelo
amor e apoio incondicionais.

AGRADECIMENTOS

Na trajetória de qualquer pessoa, inúmeros percalços virão a incomodá-la, fazê-la tentar desistir, desanimar, ou deixar de sonhar. E nestes momentos de maior dificuldade, em meio a tantos acasos, descasos e adversidades, não existe um único indivíduo no mundo capaz de superá-los sozinho. Do nascimento até a morte, cada passo dado com sucesso é graças a todos aqueles que, de uma forma ou de outra, contribuíram para que a vontade, energia, alegria e determinação os mantivessem em pé. Aos que seguraram minha mão durante essa caminhada, agradeço:

À minha mãe, Lídia, por ser meu apoio incondicional, pelas noites e madrugadas em claro por todos os esforços muitas vezes além do que podia suportar, e por provar que não há nada mais forte que o amor de uma mãe.

Ao meu pai, Edson, por ser o verdadeiro exemplo de caráter e integridade que moldaram o que sou hoje, pelos incontáveis sacrifícios feitos diariamente, pelos conselhos sempre sábios, e por me inspirar a um dia ser um pai tão bom quanto ele.

Ao meu irmão Fernando, por compartilharmos do mesmo sangue, pela parceria nas bagunças de infância, pelas manhãs, tardes e noites de jogatinas, e, por que não, também de programações. Pela paciência, maturidade e por ter a certeza que somos para toda a vida.

Ao meu orientador, Professor Doutor Fábio Branco, pela atenção e disponibilidade excepcionais, pelos e-mails respondidos mais instantaneamente do que em mensagens instantâneas, e por demonstrar um carinho sem igual por seus alunos, honrando com dignidade o papel do professor.

Aos meus amigos que trilharam a árdua estrada da graduação comigo, em especial à Mariana Kato, que através da música e do companheirismo mostrou que é a irmã que a vida me deu; ao Eduardo Canteri e Amanda Radeck, por serem fruto de uma amizade improvável, mas que se fortalece a cada café que milagrosamente conseguimos marcar de tomar. À Nathalia Radeck, por ser um motivo de sorriso durante os duros dias de trabalho e companhia indispensável para os salgados da tarde.

A todos os professores que participaram da minha formação acadêmica, por terem o prazer de compartilhar o conhecimento e formar cidadãos capazes de grandes feitos.

O caminho é longo, e o final desta etapa representa apenas o começo de muitas outras jornadas. Porém, graças a todos estes que cativaram seu lugar neste agradecimento, nenhuma frase traduz melhor meu sentimento como esta, do célebre Isaac Newton:

“Se vi mais longe, foi por estar sobre os ombros de gigantes”.

RESUMO

KUMAGAI, E. T. K. **Avaliação de métodos heurísticos para problemas *flowshop* permutacional distribuído com minimização do tempo total de fluxo e duração total da programação.** 2018. 79 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Produção) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2018.

A crescente demanda por produtos e serviços obriga as empresas a otimizarem cada vez mais seus sistemas produtivos de forma a manterem-se capazes de competir e satisfazer às necessidades do mercado. Uma das ferramentas que auxilia o processo de otimização é o *scheduling*, que pode trazer benefícios para a empresa gerando redução de custos e desperdícios e a maximização da utilização de recursos. Para isso, é válida a implementação de métodos heurísticos para a obtenção de soluções factíveis para problemas de programação de tarefas. Este trabalho tem como intuito embasar teoricamente os conceitos de *scheduling*, aplicar através de um experimento computacional as heurísticas LPT e SPT já conhecidas na literatura para o problema *flowshop* distribuído e propor variações destas para verificar seu desempenho na minimização das funções objetivo da duração total da programação e tempo total de fluxo. Como resultado obtido das experimentações computacionais, obteve-se desempenho semelhante das heurísticas estudadas para a minimização da duração total da programação, e superioridade do método SPT para minimização do tempo total de fluxo.

Palavras-chave: *Scheduling*. Programação de Tarefas. Heurísticas. *Flowshop*. Funções objetivo.

ABSTRACT

KUMAGAI, E. T. K. **Heuristic methods evaluation for distributed permutational flowshop problems with total flow time and makespan minimization.** 2018. 79 f. Trabalho de Conclusão de Curso (Industrial Engineering Bachelor's Degree) - Federal Technology University - Paraná. Ponta Grossa, 2018.

The rising demand for products and services forces companies to optimize continuously their production systems in order to maintain competitivity and satisfy the market necessities. One of the tools that helps this optimization process is scheduling, which is capable of bringing benefits to the company through cost and waste reduction and resource use maximization. It is valid, indeed, the implementation of heuristic methods to obtain feasible solutions for job scheduling problems. This paper aims to consolidate a theoretical base for scheduling concepts, implement the LPT and SPT heuristics already known in literature for the distributed flowshop problem through computational experimentation and propose variations of the heuristics to analyse the overall performance in minimization of the proposed objective functions makespan and flow time. As result of the computational experimentation, a similar performance for the heuristics chosen for this paper in terms of makespan minimization, and SPT heuristic method superiority in terms of flow time minimization.

Keywords: Scheduling. Job Programming. Heuristics. Flowshop. Objective functions.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Para qualquer tipo de produto ou serviço ofertado no mercado, há um processo que define como e quando as atividades serão realizadas. Com a crescente demanda de produtos e serviços, aliada ao aumento da competição, as empresas têm se visto numa posição de necessidade de obter vantagem competitiva em relação aos seus concorrentes. Um dos caminhos para isso é buscar o aumento da eficiência de seus processos, de modo a minimizar cada vez mais os custos e desperdícios e, assim, oferecerem ao consumidor bens ou serviços que satisfaçam suas necessidades a um preço competitivo.

No cenário industrial, os produtos são gerados a partir de sequências de atividades até sua forma final. Segundo Branco (2011), um processo ou conjunto de processos que geram os produtos ou serviços podem ser caracterizados como um sistema de produção. Tais sistemas podem apresentar características distintas de acordo com a natureza e atividades desempenhadas por cada empresa. Johnson e Montgomery (1974) classificam os sistemas de produção em três categorias: o sistema contínuo, comum em produção em massa, com pouca ou nenhuma variação de produtos, porém alto volume de produção; o sistema intermitente, com diferentes tipos de produtos; e o sistema grande projeto, em que os produtos produzidos costumam ser exclusivos, de grande porte ou com características únicas.

O Planejamento e Controle da Produção (PCP) é a área responsável por orientar e direcionar todos os processos produtivos de forma que possam ser executados da forma mais eficiente possível. Assim, o PCP define os tipos e quantidades de produtos que serão produzidos, os processos e suas sequências necessárias para a produção, o local onde a produção ocorrerá, os responsáveis pelas atividades e, por fim, o cronograma a ser seguido (BOIKO, 2008).

Dentro do PCP, a programação da produção tem a responsabilidade de definir quais atividades deverão ser desempenhadas em um determinado período de tempo. Além disso, também deve ser definido o sequenciamento das tarefas de acordo com a disponibilidade de recursos que satisfaça às demais restrições para a programação. Este sequenciamento é denominado pela comunidade científica como *scheduling* que segundo Pinedo (2012), consiste em

determinar os instantes de início e fim das atividades e os meios de processamento a serem utilizados.

Diversos estudos são realizados na área de *scheduling*, localizada na área da Pesquisa Operacional, por conta da complexidade de seus problemas. Esta complexidade pode ser definida pelo caráter combinatorial dos problemas, pois podem tratar de um ou mais tipos distintos de máquinas, que por sua vez realizarão o processamento de um ou mais tipos distintos de produtos. Apesar disso, a aplicação do *scheduling* pode trazer inúmeros benefícios à empresa, pois permite que as atividades sejam realizadas de forma mais eficiente.

Apesar de ser mais frequentemente tratado no âmbito industrial, o sequenciamento de tarefas pode ser aplicado em diversas outras áreas do cotidiano que necessitem uma ordenação de suas atividades, como a sequência de atendimento de clientes numa loja, ou até mesmo a ordem de pousos e decolagens em aeroportos.

Para a realização do *scheduling* em um sistema de produção, é possível utilizar dois tipos de métodos para obtenção de uma solução ótima para o problema proposto, os métodos exatos e os métodos heurísticos. Segundo Silva (2012), os métodos exatos consistem na modelagem matemática através da Programação Inteira Mista, como exemplo, o método *Branch and Bound*. Porém, sua aplicação prática é restrita, sendo pouco utilizado para a resolução de problemas de dimensão mais elevada. Já os métodos heurísticos são baseados em estudos e conhecimentos prévios sobre problema proposto, que possibilitam a construção de modelos computacionalmente viáveis e de boa qualidade. Somado a isso, a modelagem é realizada de acordo com sistemas reais, com a aproximação da sua aplicação a cenários práticos.

1.1 PROBLEMA

Existe um método heurístico cujo desempenho se destaca para satisfazer às funções objetivo de tempo total de fluxo e duração total da programação?

1.2 OBJETIVOS

1.2.1 Geral

- Analisar o desempenho de métodos heurísticos de fácil implementação que forneça soluções de alta qualidade para um problema *flowshop* permutacional distribuído.

1.2.2 Específicos

- Buscar heurísticas de alto desempenho e de fácil implementação na literatura.
- Adaptar métodos heurísticos de outros problemas existentes na literatura para que se adequem ao proposto nesta pesquisa;
- Encontrar soluções que satisfaçam às funções objetivo estabelecidas.

1.3 JUSTIFICATIVA

A competitividade do mercado atual gera disputas cada vez mais intensas pela liderança no mais diversos setores da economia. Por isso, cada vez mais torna-se necessário que as empresas busquem formas de aumentar a eficiência de suas atividades, de movo a obter vantagem competitiva em relação a seus concorrentes.

Mesmo com as mais diversas ferramentas disponíveis, ainda é um fato que as empresas têm dificuldades para otimizar a utilização de seus recursos materiais e também do tempo, o que gera desperdícios que acabam refletindo no custo do produto e, certamente, no desempenho da empresa no mercado.

Tendo em vista esta realidade, estudar *scheduling* apresenta-se como uma possibilidade de que as tarefas em um ambiente produtivo sejam programadas de forma a encontrar um ponto de equilíbrio entre o baixo custo e a minimização do tempo de processamento nas máquinas.

A aplicação de heurísticas para este problema mostra-se interessante visto que pode trazer um impacto positivo no desempenho da empresa, sendo de fácil implementação e com soluções de alta qualidade.

Neste trabalho, o estudo será focado no *scheduling* com a utilização de métodos heurísticos para a solução de um problema *flowshop* permutacional, com as funções objetivo de minimização do tempo total de fluxo e duração total da programação. A razão da escolha deste tipo de problema é o fato de que os demais tipos podem ser considerados generalizações do problema *flowshop*, enquanto as funções objetivo podem ser consideradas clássicas para problemas de *scheduling*.

1.4 ESTRUTURA DO TRABALHO

Este trabalho é estruturado em cinco capítulos. Neste primeiro, é dada uma introdução sobre o tema do estudo, com uma análise geral do cenário industrial atual, a necessidade das empresas de planejar suas atividades em busca de uma maior eficiência das operações e a apresentação do *scheduling* como uma ferramenta para o sequenciamento de tarefas em um sistema produtivo. Em seguida, são apresentados os objetivos geral e específicos do estudo, assim como a justificativa para a realização deste.

No segundo capítulo, é apresentado o referencial teórico referente ao contexto geral do Planejamento, Programação e Controle da Produção, sistemas produtivos, ambientes de programação, os tipos de restrições e funções objetivo para problemas de *scheduling*.

O terceiro capítulo é dedicado à descrição da metodologia proposta para o estudo a ser realizado.

O quarto capítulo contém o detalhamento do desenvolvimento do estudo.

O quinto e último capítulo é referente às conclusões do estudo após realização das análises.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo é dedicado à apresentação da revisão bibliográfica realizada para este estudo. Para um melhor entendimento do problema, está dividido em nove seções.

2.1 SISTEMAS DE PRODUÇÃO

Um sistema de produção pode ser definido, segundo Harding (1981), como um ambiente formado por partes inter-relacionadas, as quais recebem as entradas, ou *inputs*, e as transformam, resultando nas saídas, ou *outputs*. Estes processos de transformação, por sua vez, devem ser realizados de forma padronizada e controlada.

Os sistemas de produção podem ser classificados das seguintes formas, de acordo com Johnson e Montgomery (1974):

- Sistema contínuo: é um sistema em que são produzidos em larga escala itens com pouca diversificação, padronizados;
- Sistema intermitente: um sistema em que são produzidos produtos com diferentes características, e se pode identificar duas subdivisões:
    - *Flowshop*: os itens fabricados possuem a mesma rota de processamento em todas as máquinas do sistema; e
    - *Job Shop*: cada item fabricado possui sua própria rota de processamento nas máquinas do sistema.
- Sistema grande projeto: sistema em que são fabricados itens com características únicas, com alto grau de sofisticação ou de grande porte.

É um fato que a competitividade atual exige capacidade das empresas de satisfazer às demandas do mercado, e que a exigência por produtos cada vez mais diferenciados segue elevada. Neste contexto, pode-se concluir que há a necessidade de um sistema flexível, capaz de atender às variações da demanda. Assim, neste trabalho, o foco do estudo será no sistema intermitente.

2.2 PLANEJAMENTO, PROGRAMAÇÃO E CONTROLE DA PRODUÇÃO

A função do Planejamento e Controle da Produção é, segundo Slack *et. al* (1999), garantir que a produção possa ser realizada conforme o planejado e produzir bens e serviços dentro dos padrões de qualidade estabelecidos. Para isso, é imprescindível que a empresa tenha à disposição os recursos para a produção na quantidade, no momento e no nível de qualidade necessários. Chiavenato (1991) afirma que o PCP busca aumentar a eficiência e eficácia da empresa através do planejamento antecipado e do controle da produção.

Em outras palavras, o PCP busca organizar todo o processo de manufatura ou de serviços ao definir objetivos e necessidades para cada etapa do processo, de modo a evitar a falta ou desperdício de recursos ou de tempo, reduzindo ou eliminando não-conformidades através do controle das operações para, finalmente, entregar o bem ou serviço ao cliente dentro do prazo estipulado.

A Programação da Produção, por sua vez, é responsável por estipular especificamente as quantidades e prazos a serem atendidos pela produção para atender à demanda. É dentro da Programação da Produção em que se encontra a etapa de sequenciamento de operações, ou *scheduling*, que consiste na definição de prioridades das operações a serem realizadas e a ordenação das mesmas num sistema, de forma que seja possível atingir aos objetivos determinados (CORRÊA; CORRÊA, 2013).

2.3 *SCHEDULING*

O *scheduling*, segundo a definição de Pinedo (2012), é um processo de tomada de decisão que permite otimizar um ou mais objetivos através da alocação de recursos e tarefas em um determinado período de tempo, e pode ser aplicado tanto nos ambientes de manufatura quanto de serviços.

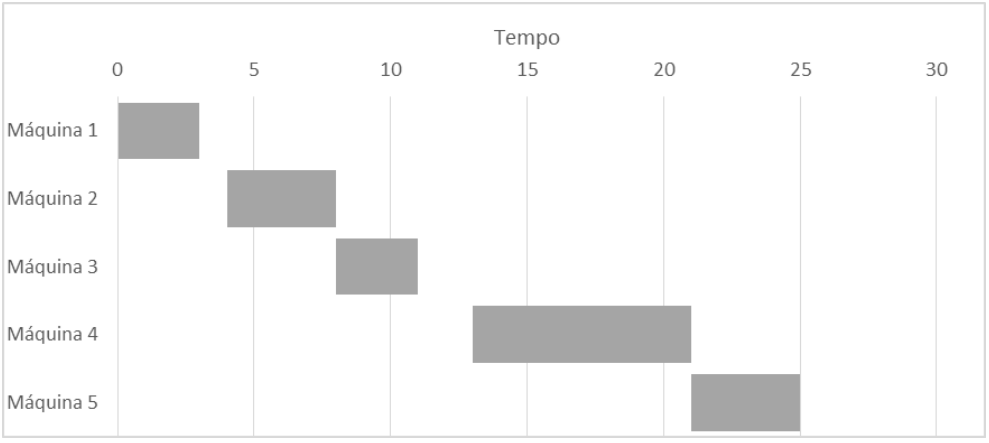
Tradicionalmente, um problema de *scheduling* tem como objetivo alocar n tarefas em m máquinas, sendo que a operação de processamento da primeira tarefa na primeira máquina pode ser chamado de o_{ij}. Esta alocação visa satisfazer uma ou mais funções objetivos julgadas interessantes para o sistema produtivo em estudo. Segundo Fuchigami (2010), a complexidade do problema

de programação aumenta de acordo com as restrições a que está sujeito. Como normalmente são considerados finitos e determinados a quantidade de tarefas e recursos, pode haver limitações em relação à utilização de recursos em tarefas, sequência de processamento de duas atividades distintas, simultaneidade de utilização de recursos, tempo de processamento, entre outros. A dificuldade para encontrar soluções factíveis para o problema cresce à medida que a complexidade também aumenta. Desta forma, é comum que a busca seja feita através de soluções para problemas mais simples, que podem ser adaptadas e aproximadas para problemas mais complexos. Para isso, um dos métodos mais comuns é a utilização de heurísticas.

Uma ferramenta muito comum para auxílio em problemas de programação de tarefas é o gráfico de Gantt, proposto por Henry Laurence Gantt no encontro anual da *American Society of Mechanical Engineers*, no ano de 1918. Trata-se de uma representação visual, muito simplificada em relação à teoria original de Gantt, que permite uma rápida compreensão do início e fim de cada atividade, bem como seu andamento (PINEDO, 2012).

A Figura 1 apresenta um exemplo do gráfico de Gantt com cinco máquinas.

Figura 1: Gráfico de Gantt



Fonte: Autor, 2016

A próxima seção apresenta o problema de programação e suas subdivisões, conforme as características produtivas.

2.4 PROBLEMAS DE PROGRAMAÇÃO DE TAREFAS

Segundo Pinedo (2005), uma variedade de fatores pode caracterizar um sistema de manufatura, dentre eles o número de recursos/máquinas, a configuração do sistema, o nível de automação, o tipo de material a ser utilizado, entre outras. Estes fatores dão ao sistema suas caraterísticas que abrem a possibilidade de inúmeros modelos para o sequenciamento das operações. Com isso, pode-se obter nove diferentes classificações para os problemas de programação.

2.4.1 Máquina Única

O problema de máquina única consiste em uma única máquina que processa todas as tarefas em uma única etapa.

2.4.2 Máquinas Paralelas

Este problema consiste em um conjunto de máquinas idênticas que processam todas as tarefas numa mesma etapa, sendo que cada tarefa só pode ser processada por uma única máquina. É comum que problemas de máquinas paralelas sejam divididos em múltiplos problemas de máquina única, de modo a simplificar o problema e facilitar a modelagem.

2.4.3 *Job Shop*

No *job shop*, cada tarefa possui sua própria sequência de processamento. Há também apenas uma máquina por etapa, sendo que cada tarefa passa apenas uma vez por cada máquina.

2.4.4 *Job Shop* com Máquinas Múltiplas

Proposto por Moccellin e Nagano (2003), e assim como no *job shop*, cada tarefa possui sua própria sequência de processamento, porém há mais de uma máquina por etapa. As tarefas, por sua vez, são processadas por apenas uma das máquinas em cada etapa.

2.4.5 *Flowshop*

No *flowshop*, todas as tarefas possuem a mesma ordem de processamento em todas as máquinas. O número de máquinas, por sua vez, é apenas um a cada etapa de processamento. Pinedo (2005) afirma que um *flowshop* pode ser classificado como um *job shop* em que todas as tarefas possuem a mesma sequência de processamento.

2.4.6 *Flowshop* com Máquinas Múltiplas

Também proposto por Moccellin e Nagano (2003), é similar ao *flowshop*, em que a ordem de processamento de todas as tarefas também é a mesma, mas há mais de uma máquina em cada etapa do processamento. Porém, cada tarefa é processada apenas por uma máquina em cada etapa.

2.4.7 *Flowshop* Permutacional

Um *flowshop* em que a ordem de processamento de todas as tarefas deve ser o mesmo em todas as máquinas do sistema.

2.4.8 *Flowshop* Permutacional Distribuído

O conceito do *flowshop* permutacional distribuído é semelhante ao *flowshop* permutacional, porém com a particularidade da existência de uma quantidade determinada de células, ou fábricas, de processamento, cada qual com uma quantidade m de máquinas. Assim, é possível distribuir as tarefas a serem processadas entre as células, de modo a minimizar o tempo de

processamento (NADERI; RUIZ, 2010). Este problema é o tratado neste trabalho.

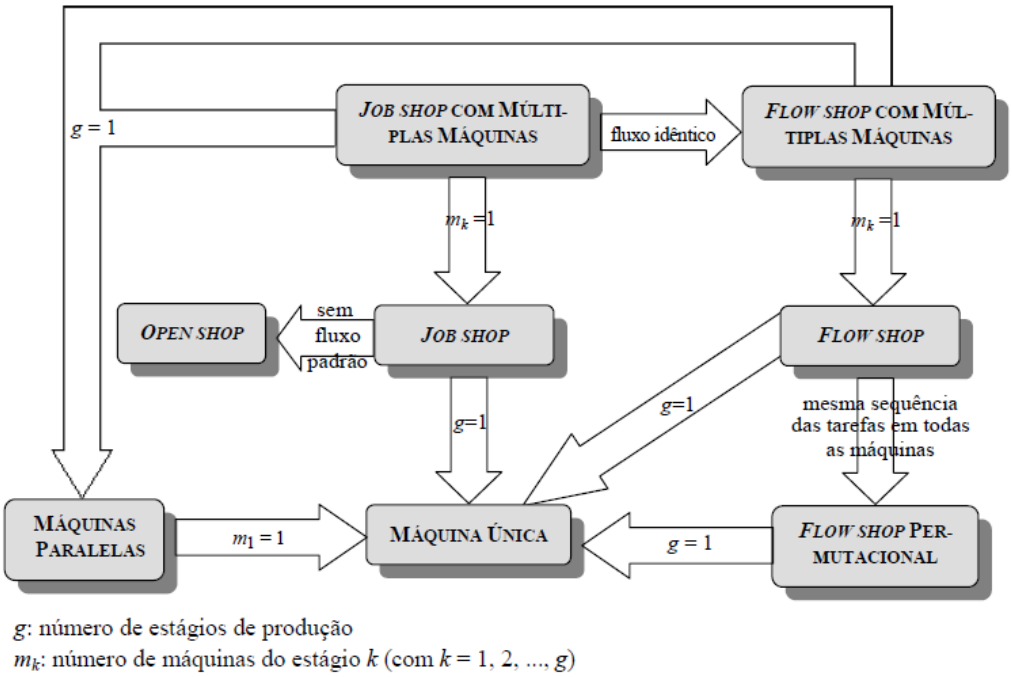
2.4.9 *Open-shop*

No *open-shop*, as tarefas não possuem ordem de sequenciamento pré-determinadas, podendo ser processadas apenas uma vez em qualquer máquina. Neste caso, há apenas uma máquina por etapa de processamento.

2.4.10 Relações entre Problemas de Programação

É possível observar a relação entre os diferentes problemas de programação através da Figura 2.

Figura 2: Relação entre problemas de programação



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

2.5 COMPLEXIDADE COMPUTACIONAL

Problemas computacionais possuem níveis de complexidade, o que influencia na viabilidade de solução através de esforços computacionais. Isso

significa que a complexidade determina se um problema pode ser solucionado de acordo com restrições tecnológicas computacionais, como memória e tempo de processamento.

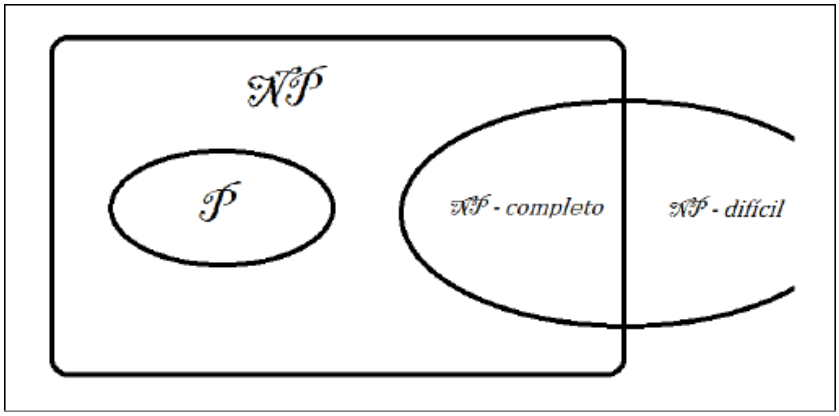
Num problema simples de programação de n tarefas em máquina única, tem-se um conjunto de $n!$ soluções possíveis. Caso os parâmetros do problema sejam alterados, como por exemplo se a rota de processamento das n tarefas nas m máquinas não seja o mesmo, tem se um conjunto de $n!^m$ de soluções possíveis (SLACK *et al.*, 1999).

De acordo com Silva (2012), há quatro principais classes de complexidade:

- P: um problema de classe de complexidade P possui um algoritmo cujos custos envolvidos podem ser representados por n^k, sendo n o tamanho do problema e k uma constante;
- NP: classe de problema cuja solução pode ser obtida em tempo polinomial;
- NP-*complete*: é a classe em que todos os problemas NP nela contidos podem ser reduzidos a um problema P;
- NP-*hard*: é a classe em que todos os problemas NP e NP-*complete* podem ser reduzidos a um problema P. Segundo Pinedo (2012), problemas NP-*hard* não possuem algoritmos para solução em tempo polinomial.

Problemas da classe NP-*hard* ainda não foram totalmente identificados, o que direciona esforços a esta classe de complexidade (SILVA, 2012). A Figura 3 apresenta a relação entre as classes de complexidade.

Figura 3: Relação entre classes de complexidade



Fonte: Silva, 2012

Os métodos heurísticos têm sido adotados como principal ferramenta para a resolução de problemas complexos de *scheduling*. Embora haja casos em que uma solução ótima não pode ser obtida através de heurísticas, estas permitem a obtenção de soluções de boa qualidade e em tempo computacional aceitável.

2.6 AMBIENTES DE PROGRAMAÇÃO

Existem dois ambientes em que as operações de um sistema produtivo podem ser realizadas, consequentemente gerando dois ambientes de programação distintos (BOIKO, 2008). De acordo com Baker e Trietsch (2009), a definição dos ambientes de programação é dividida em estático e dinâmico, e comentados nas seções que seguem.

2.6.1 Estático

No sistema estático, o número de tarefas no sistema se mantém constante durante toda a programação. Silva (2012) complementa que um sistema estático é aquele que, inicialmente ocioso, recebe um número n de tarefas de forma simultânea e que pode iniciar o processamento de qualquer uma delas de forma imediata.

2.6.2 Dinâmico

No sistema dinâmico, existe a possibilidade de novas tarefas serem incluídas no processamento.

2.7 RESTRIÇÕES PARA O PROBLEMA *FLOWSHOP*

As restrições são condições às quais os problemas de programação estão sujeitos, e são determinados de acordo com o tipo de atividade a ser realizado pela empresa, ou a natureza de seus produtos e serviços. As restrições mais comuns em problemas de *scheduling* são a *no-wait*, em que não se permite tempo de espera entre máquinas, e a *no-idle*, em que não se permite tempo de espera entre tarefas. A seguir, as duas restrições serão detalhadas.

2.7.1 *No-Wait*

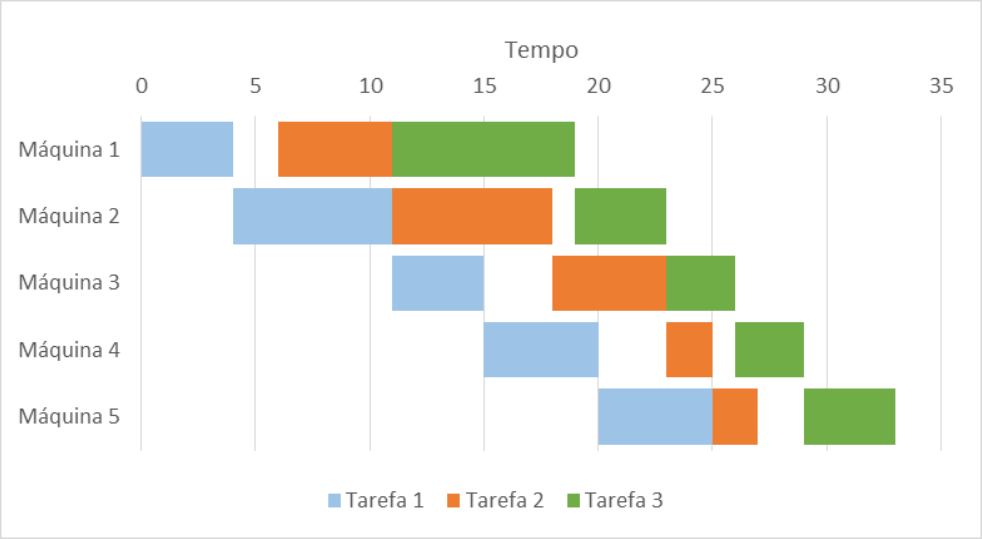
A restrição *no-wait* determina que, num sistema de n tarefas e m máquinas, uma tarefa que se inicia na primeira máquina deve ser processada em todas as máquinas subsequentes sem que haja interrupção ou atrasos entre elas, do mesmo modo até o último processamento na última máquina, ou seja, sem tempo de espera entre máquinas. Portanto, tem-se que o processamento da tarefa na máquina segunda máquina deve começar imediatamente após o término do processamento na primeira máquina, e assim sucessivamente.

É possível identificar ocorrências de *no-wait flowshops* no ambiente industrial principalmente em casos em que o tipo de produto requer um processamento contínuo de modo a evitar sua degradação ou defeitos. Sapkal e Laha (2013) citam como exemplos de sistemas *no-wait flowshops* as indústrias siderúrgicas, plásticas, alimentícias, farmacêuticas, químicas, entre outras, em que não se pode haver esperas entre as etapas de processamento para que se mantenha o produto em condições apropriadas. Para garantir que a restrição *no-wait* seja respeitada, Zhu, Li e Wang (2011) afirmam que o início do processamento de uma dada tarefa numa dada máquina deve ser atrasado quando necessário para que se possa garantir que os processamentos nas máquinas subsequentes sejam contínuos, sem espera. Ou seja, o único

momento em que é aceitável que haja espera é antes que a primeira tarefa inicie seu processamento na primeira máquina (BRANCO, 2006).

Na Figura 4, pode-se observar um sistema *no-wait flowshop* pelo gráfico de Gantt para três tarefas e cinco máquinas.

Figura 4: Gráfico de Gantt – Sistema *no-wait flowshop*



Fonte: Autor, 2016

Na próxima seção é comentada outra restrição comum para problemas de *flowshop*.

2.7.2 *No-Idle*

O problema *no-idle* não permite tempos de espera, mas em contraste com o problema *no-wait*, este tempo de espera refere-se às máquinas do sistema produtivo. Ou seja, num sistema *no-idle flowshop*, todas as máquinas devem processar todas as operações sem que haja tempo ocioso entre uma tarefa e outra, até o fim da operação. Assim, o processamento de uma dada tarefa numa dada máquina deve ser iniciado imediatamente após o término do processamento da tarefa anterior na mesma máquina, de forma que o processamento de todas as tarefas seja feito de forma contínua e ininterrupta.

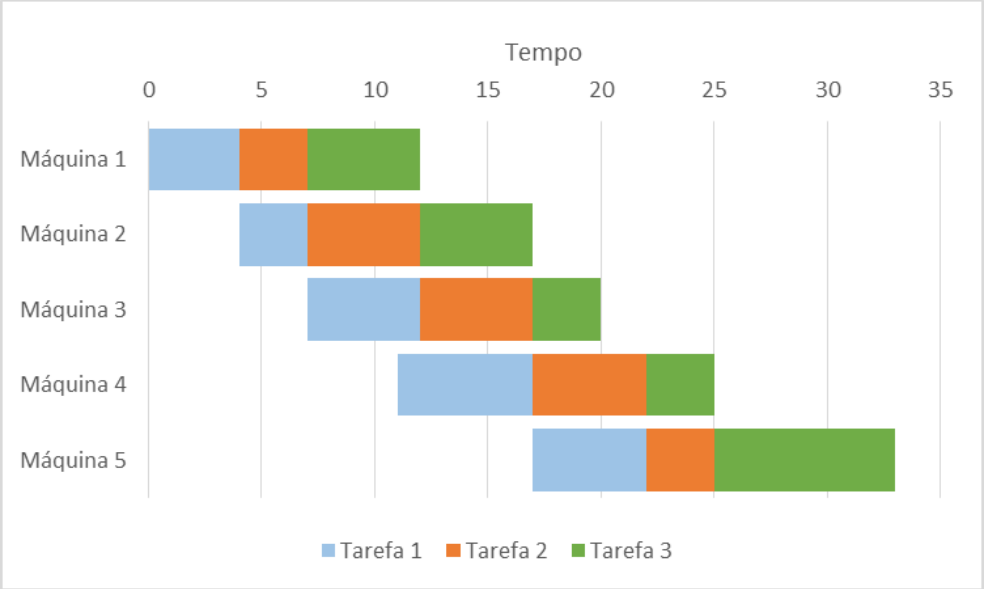
A restrição *no-idle* é identificada em sistemas produtivos em que o custo de preparação e utilização das máquinas é elevado, o que gera dispêndios desnecessários em caso de paradas para *setup* ou ociosidade. No caso dos

tempos de *setup*, são comumente incluídos no tempo de processamento ou desconsiderados caso sua duração não seja relevante em relação ao tempo de processamento (BRANCO, 2011).

Para satisfazer a uma restrição *no-idle*, permite-se que o início do processamento de uma tarefa em máquinas subsequentes seja atrasado, o que gera tempo de espera entre tarefas, porém garante o processamento contínuo de tarefas subsequentes numa mesma máquina.

Na Figura 5, observa-se pelo gráfico de Gantt um sistema *no-idle flowshop* para três tarefas e cinco máquinas.

Figura 5: Gráfico de Gantt – Sistema *no-idle flowshop*



Fonte: Autor, 2018

Este trabalho comenta na próxima seção as principais funções objetivo dos problemas de programação.

2.8 FUNÇÕES OBJETIVO

As funções objetivo, também conhecidas como medidas de desempenho, são o que determinam o parâmetro que se busca atingir ao final da programação. A seguir, serão apresentadas algumas das principais funções objetivo em problemas de programação de tarefas.

2.8.1 Tempo de Fluxo

Conhecido também como *flow time*, é o tempo desde a disponibilidade da tarefa para ser processada até o final do seu processamento, ou seja, o tempo total que a tarefa permanece numa etapa de processamento (BOIKO, 2008).

Para o exemplo da Figura 4 de um sistema *no-wait flowshop*, apresentado no item 2.7.1, tem-se os seguintes tempos de processamento, apresentados na Tabela 1.

Tabela 1: Tempos de processamento de um sistema *no-wait flowshop*

	Tarefa 1	Tarefa 2	Tarefa 3
Máquina 1	4	5	8
Máquina 2	7	7	4
Máquina 3	4	5	3
Máquina 4	5	2	3
Máquina 5	5	2	4

Fonte: Autor, 2016

O *total flow time* corresponde à soma do tempo de término de cada tarefa na Máquina 5. De acordo com a Figura 4, o tempo final de processamento da Tarefa 1 na Máquina 5 é de 25 unidades de tempo. A Tarefa 2, por sua vez, só inicia seu processamento na Máquina 1 após o processamento da Tarefa 1 na mesma máquina mais um tempo ocioso, totalizando num tempo final de processamento de 27 unidades de tempo na Máquina 5. Já a Tarefa 3 só inicia seu processamento após o processamento da Tarefa 1, da Tarefa 2 e mais o tempo ocioso entre estes, totalizando um tempo final de processamento de 33 unidades de tempo na Máquina 5. Isso resulta num valor de 85 unidades de tempo.

Já no exemplo da Figura 5 de um sistema *no-idle flowshop*, apresentado no item 2.7.2, tem-se os seguintes tempos de processamento, apresentados na Tabela 2.

Tabela 2: Tempos de processamento de um sistema *no-idle flowshop*

	Tarefa 1	Tarefa 2	Tarefa 3
Máquina 1	4	3	5
Máquina 2	3	3	5
Máquina 3	4	4	3
Máquina 4	6	5	3
Máquina 5	4	3	5

Fonte: Autor, 2016

A Tarefa 1 inicia seu processamento na Máquina 1 no instante zero. Porém, para respeitar a restrição *no-idle*, seu processamento tem um atraso de 2 unidades de tempo na Máquina 2, e atraso de 1 unidade de tempo nas máquinas 4 e 5 respectivamente, totalizando um tempo final de processamento de 25 unidades de tempo.

A Tarefa 2 inicia seu processamento na Máquina 1 imediatamente após o término do processamento da Tarefa 1 na mesma máquina. Porém, a Tarefa sofre atrasos de, respectivamente, 2, 1 e 3 unidades de tempo nas máquinas 2, 3 e 4, resultando num tempo final de processamento de 24 unidades de tempo.

Por fim, a Tarefa 3 inicia seu processamento na Máquina 1 imediatamente ao final do processamento da Tarefa 2 na mesma máquina, sofrendo apenas um atraso de 5 unidades de tempo na Máquina 4. Com isso, o tempo final de processamento da Tarefa 3 é de 26 unidades de tempo.

Assim, tem-se que o *flow time* resulta num valor de 75 unidades de tempo.

2.8.2 Duração Total da Programação

A Duração Total da Programação, também conhecido como *makespan* ou *Maximum Completion Time*, constitui no tempo total utilizado desde o início do processamento da primeira tarefa na primeira máquina, até o final do processamento da última tarefa na última máquina.

Em problemas de programação, busca-se minimizar o *makespan*, o que consequentemente maximiza a utilização de recursos (PINEDO, 2012).

Para o exemplo da Figura 4 de um sistema *no-wait flowshop*, apresentado no item 2.7.1 e segundo os valores da Tabela 1, o cálculo do *makespan*, correspondente ao intervalo de tempo compreendido entre o início do processamento da Tarefa 1 na Máquina 1 até o fim do processamento da Tarefa 3 na Máquina 5, resulta num valor de 33 unidades de tempo.

Já no exemplo da Figura 5 de um sistema *no-idle flowshop*, apresentado no item 2.7.2 e segundo os valores da Tabela 2, o cálculo do *makespan* resulta num valor de 34 unidades de tempo.

2.8.3 *Lateness*

É o a diferença entre o tempo de término de uma tarefa e a data limite, ou *due date*. Como uma tarefa pode terminar antes ou depois de sua *due date*, o valor do *lateness* assume um valor negativo para o primeiro caso, e positivo para o último (BOIKO, 2008).

2.8.4 *Tardiness*

É quanto o tempo de término de uma tarefa atrasa em relação à *due date*; corresponde a um *lateness* positivo. Pinedo (2005) afirma que o término de uma tarefa após a data limite é permitido, mas que podem ser incorridas penalidades pelo atraso.

2.9 HEURÍSTICAS

As heurísticas têm se apresentado como uma eficiente ferramenta para resolução de problemas de *scheduling*. Segundo Carneiro (2010, p.24), são “aqueles que levam a soluções viáveis através de avaliações baseadas nas características do problema ou ainda com base em critérios computacionais, mas sem percorrer todas as soluções possíveis”. Ainda de acordo com Carneiro (2010), tem-se que as soluções obtidas não são necessariamente as ótimas para o problema proposto, porém são soluções de boa qualidade e obtidas dentro de um tempo computacional razoável.

A utilização de heurísticas se dá através de algoritmos computacionais e racionais e são classificados de acordo com Fuchigami (2010) em:

- Métodos construtivos: procedimentos iterativos realizados a partir de soluções parciais fornecem a solução para o problema, e;
- Métodos melhorativos: procedimentos iterativos são realizados a partir de uma solução inicial, ao passo que se busca obter soluções cada vez melhores em relação à função objetivo proposta.

As heurísticas mostram-se bastante convenientes principalmente para a resolução de problemas de alta complexidade. Várias heurísticas foram desenvolvidas, adaptadas e melhoradas ao longo dos anos, sendo que algumas das principais serão abordadas neste trabalho.

2.9.1 *Longest Processing Time* (LPT)

A heurística LPT consiste na ordenação das tarefas em ordem decrescente em relação à soma dos tempos de processamento (CRUZ; BRANCO, 2009). Ou seja, a próxima tarefa a ser alocada é sempre aquela cujo tempo de processamento é imediatamente menor do que a anterior, e assim sucessivamente até que todas as tarefas sejam alocadas.

É possível encontrar referências ao método como *Longest Processing Time First*, o que reforça a prioridade de alocação às tarefas que possuem os maiores valores para os tempos de processamento (HUANG; ZHANG; YU, 2013).

2.9.2 *Shortest Processing Time* (SPT)

Oposta à heurística LPT, o método SPT realiza a ordenação das tarefas em ordem crescente em relação à soma dos tempos de processamento (CRUZ; BRANCO, 2009). Desta forma, a próxima tarefa alocada é sempre aquela cujo tempo de processamento é imediatamente maior do que a anterior, até que todas as tarefas sejam alocadas.

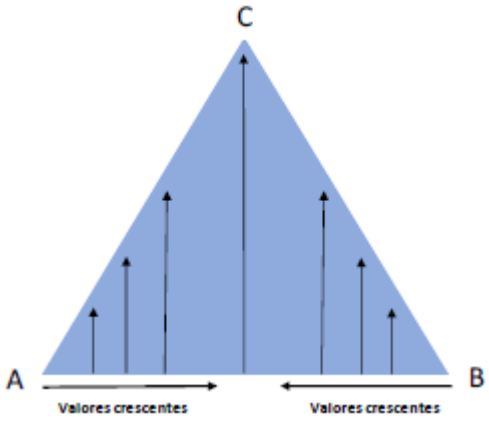
Assim como no caso do LPT, também há referências ao SPT como *Shortest Processing Time First*, cujas prioridades são das tarefas com menores tempos de processamento (BOBELIN; MARTINEAU; HE, 2016).

2.9.3 Triangular

Proposta por Branco e Santos (2016), a heurística Triangular é um método de ordenação inicial das tarefas, e como o próprio nome define, busca a construção de um triângulo aproximado formado pelos tempos de processamento de cada tarefa. Desta forma, ficam dispostas as tarefas com menor tempo de processamento nas extremidades do triângulo, enquanto a tarefa com maior tempo de processamento fica na posição central do triângulo.

A Figura 6 apresenta de forma visual a alocação das tarefas na heurística Triangular.

Figura 6: Representação gráfica da heurística Triangular



Fonte: Branco e Santos, 2016

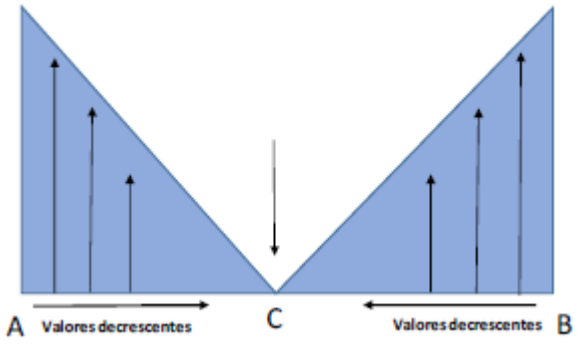
A distribuição das tarefas é alternada entre os dois lados do triângulo, sendo que se inicia alocando a tarefa de menor tempo de processamento no lado esquerdo do triângulo, em seguida alocando a tarefa com segundo menor tempo de processamento no lado direito do triângulo, e assim sucessivamente até que todas as tarefas estejam alocadas (BRANCO; SANTOS, 2016).

2.9.4 Triangular Invertida

A heurística Triangular Invertida, também proposta por Branco e Santos (2016), tem o mesmo conceito da heurística Triangular, porém com a particularidade de que as tarefas são ordenadas de modo a formar um triângulo invertido, com as tarefas com maior tempo de processamento nas extremidades do triângulo, e a tarefa com menor tempo de processamento na posição central.

A Figura 7 apresenta de forma visual a alocação das tarefas na heurística Triangular Invertida.

Figura 7: Representação gráfica da heurística Triangular Invertida



Fonte: Branco e Santos, 2016

A distribuição das tarefas na heurística Triangular Invertida é, tal como a Triangular, alternada entre os dois lados do triângulo. Porém, inicia-se alocando a tarefa de maior tempo de processamento no lado esquerdo do triângulo, em seguida alocando a tarefa com segundo maior tempo de processamento no lado direito do triângulo, e assim sucessivamente até que todas as tarefas estejam alocadas (BRANCO; SANTOS, 2016).

2.9.5 Nawaz, Enscore e Ham (NEH)

Uma heurística de duas fases proposta inicialmente por Nawaz, Enscore e Ham, que dão nome ao método, tem sido considerada por vezes a melhor heurística para minimização da duração total da programação em sistemas *flowshop* permutacionais (KALCZYNSKI; KAMBUROWSKI, 2007).

De acordo com Branco (2011), a heurística NEH é constituída por duas etapas:

- Indexação das tarefas: é realizada uma ordenação das tarefas de acordo com os valores das somas dos tempos de processamento, sendo estes não-crescentes;
- Construção da sequência solução: as posições das tarefas da solução inicial são verificadas em todas as demais posições possíveis nas sequências parciais, sendo que a tarefa é alocada na posição de menor duração total da programação.

Diversas modificações e adaptações da heurística NEH foram propostas, como a heurística NEH-D, proposta por Dong, Huang e Chen (2008), que apresentou melhoras significativas em relação ao método NEH original.

3 METODOLOGIA

Uma pesquisa, como é definida por Gil (2002, p.17), é “o procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos”. Em outras palavras, define-se como pesquisa o processo de busca e sintetização de informações disponíveis com o intuito de se obter um entendimento sobre determinado problema.

Ainda segundo Gil (2002), inúmeras fases podem compreender o processo de pesquisa, sendo estas iniciadas com a formulação precisa do problema a ser estudado até a obtenção e apresentação dos resultados.

3.1 CLASSIFICAÇÃO DE PESQUISA

Esta pesquisa pode ser classificada de acordo com:

3.1.1 Natureza

A pesquisa pode ser classificada como sendo de natureza aplicada, que segundo Gil (2008), é aquela que possui aplicação prática dos estudos realizados. Trata-se de um estudo com relevância no cenário real industrial, inclusive sendo capaz de proporcionar benefícios com sua utilização. O *scheduling* traz, através do estudo de heurísticas, a análise de métodos utilizados em situações reais e registradas na literatura, sendo possível adaptar tais métodos e obter soluções para problemas também reais.

3.1.2 Objetivos

Segundo os objetivos, a pesquisa pode ser classificada como exploratória, tal como descritiva. Gil (2008) define a pesquisa exploratória como aquela com o intuito de gerar um melhor entendimento sobre um determinado fato; quando não há grande exploração do tema ou se a pesquisa pode vir a requerer etapas posteriores de investigação. Já a pesquisa descritiva tem como finalidade descrever características de uma população, um fenômeno ou de relações entre diferentes variáveis.

O estudo de *scheduling* possui suas particularidades para cada problema estudado, sendo necessário a adaptação de métodos heurísticos para a aplicação em cada caso. Isso implica que cada estudo pode trazer novas vertentes futuramente aplicáveis a novos problemas.

Em adição a isso, o caráter descritivo do estudo se dá pela análise de heurísticas em relação a uma função objetivo e parâmetros do problema.

3.1.3 Procedimentos

Os procedimentos deste trabalho são de caráter bibliográfico, pois baseia-se em estudos previamente realizados e registrados na literatura; e também experimental, pois a análise dos resultados só pode ser realizada após um processo de experimentação computacional.

3.2 ETAPAS

Este estudo consiste nas seguintes etapas:

- Revisão bibliográfica: Análise de livros, artigos científicos de revistas, periódicos e congressos nacionais e internacionais, assim como teses e dissertações, relacionados ao tema proposto, de modo a embasar de forma concisa o trabalho através de exemplos encontrados na literatura.
- Geração de dados para análise: Através de um *software* de geração de dados, um banco de instâncias foi gerado aleatoriamente segundo um conjunto de fatores que caracterizarão o problema proposto.
- Implementação do código computacional: A implementação e adaptações das heurísticas foram realizadas utilizando a linguagem de programação Pascal.
- Experimentação computacional: A análise dos dados gerados será feita a partir de experimentos computacionais das heurísticas existentes na literatura, realizando adaptações necessárias para o problema.
- Interpretação dos resultados: Os resultados obtidos através da experimentação computacional serão interpretados e verificados em relação às

funções objetivo propostas, o que permite um posicionamento e conclusão do estudo.

4 DESENVOLVIMENTO

Para este estudo, foi definido um conjunto de problemas-teste de número de tarefas n e número de máquinas m. Estes conjuntos foram divididos em classes referentes à combinação de tarefas e máquinas, n x m. Definiu-se para o número de tarefas os valores de 40, 80, 120,160 e 200, enquanto para o número de máquinas os valores foram de 5, 10, 15 e 20. Assim, obteve-se as 20 classes apresentadas na Tabela 3.

Tabela 3: Classes de problemas-teste

Tarefas	Máquinas	Classe
40	5	40x5
40	10	40x10
40	15	40x15
40	20	40x20
80	5	80x5
80	10	80x10
80	15	80x15
80	20	80x20
120	5	120x5
120	10	120x10
120	15	120x15
120	20	120x20
160	5	160x5
160	10	160x10
160	15	160x15
160	20	160x20
200	5	200x5
200	10	200x10
200	15	200x15
200	20	200x20

Fonte: Autor, 2018

Para obtenção do banco de dados, foi utilizado um *software* gerador de dados para gerar 2000 arquivos, sendo 100 para cada classe apresentada na Tabela 3. Tais arquivos foram utilizados na experimentação computacional das heurísticas LPT, SPT, Triangular e Triangular Invertida no sistema de produção Clássico, implementadas com o código desenvolvido em linguagem Pascal apresentado no Apêndice A. As funções-objetivo utilizadas foram a duração total da programação e tempo total de fluxo para um problema *flowshop* permutacional distribuído, com as tarefas distribuídas em 2, 4 e 8 células de processamento.

Para cada heurística trabalhada, a distribuição das tarefas nas células obedeceu a duas variações propostas neste trabalho, a simples e a alternada, gerando as variações LPT-A, SPT-A, Triangular-A e Triangular Invertida-A. Na distribuição simples, as tarefas são alocadas de acordo com as instruções apresentadas a seguir:

- A tarefa 1 é alocada na primeira posição da célula 1; a tarefa 2 é alocada na primeira posição da célula 2, sucessivamente até que a primeira posição de cada célula tenha uma tarefa alocada;
- A próxima tarefa será alocada na segunda posição da célula 1, sucessivamente até que a segunda posição de cada célula tenha uma tarefa alocada;
- Repete-se o processo até que todas as tarefas estejam alocadas nas células.

Para exemplificar o método, toma-se como base um sistema produtivo de 6 tarefas e 3 máquinas com os tempos apresentados na Tabela 4.

Tabela 4: Exemplo de sistema produtivo para distribuição simples

	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5	Tarefa 6
Máquina 1	50	52	26	44	85	76
Máquina 2	54	97	91	4	4	92
Máquina 3	59	79	77	43	42	88

Fonte: Autor, 2018

Para um caso em que se tem 2 células de processamento com 3 máquinas em cada, a distribuição simples aloca as tarefas em cada célula conforme as Tabelas 5 e 6.

Tabela 5: Distribuição simples para célula de processamento 1

	Tarefa 1	Tarefa 3	Tarefa 5
Máquina 1	50	26	85
Máquina 2	54	91	4
Máquina 3	59	77	42

Fonte: Autor, 2018

Tabela 6: Distribuição simples para célula de processamento 2

	Tarefa 2	Tarefa 4	Tarefa 6
Máquina 1	52	44	76
Máquina 2	97	4	92
Máquina 3	79	43	88

Fonte: Autor, 2018

Já na distribuição alternada, as tarefas são alocadas de acordo com as instruções apresentadas a seguir:

- A tarefa 1 é alocada na primeira posição da célula 1; a tarefa 2 é alocada na primeira posição da célula 2, sucessivamente até que a primeira posição de cada célula tenha uma tarefa alocada;
- A próxima tarefa será alocada na segunda posição da última célula, com as tarefas subsequentes sendo alocadas em ordem inversa até que a segunda posição da primeira célula tenha uma tarefa alocada;
- A próxima tarefa será alocada na terceira posição da última célula, com as tarefas subsequentes sendo alocadas em sequência até que a terceira posição da última célula tenha uma tarefa alocada;
- Repete-se o processo até que todas as tarefas estejam alocadas nas células.

Exemplificando numericamente utilizando o mesmo sistema apresentado na Tabela 4, tem-se a distribuição alternada conforme as Tabelas 7 e 8.

Tabela 7: Distribuição alternada para célula de processamento 1

	Tarefa 1	Tarefa 4	Tarefa 5
Máquina 1	50	44	85
Máquina 2	54	4	4
Máquina 3	59	43	42

Fonte: Autor, 2018

Tabela 8: Distribuição alternada para célula de processamento 2

	Tarefa 2	Tarefa 3	Tarefa 6
Máquina 1	52	26	76
Máquina 2	97	91	92
Máquina 3	79	77	88

Fonte: Autor, 2018

Os exemplos apresentados foram simplificados para melhor entendimento. Porém, a experimentação computacional trabalha com números de tarefas e máquina superiores, gerando uma maior variedade de instâncias.

O cálculo das funções-objetivo, realizado no *software* DevPascal em sua versão 1.9.2, gerou arquivos de saída contendo o resultado para cada um dos problemas-teste, que foram então transferidos para uma planilha eletrônica no *software* Microsoft Excel 2013 para análise do desempenho de cada heurística. A comparação se deu através do cálculo de sucesso e desvio relativo médio, onde o sucesso é determinado pela quantidade de vezes em que uma heurística apresentou melhor desempenho, sendo que quanto maior o sucesso, melhor será a heurística; e o desvio relativo médio a relação entre uma heurística e a heurística que apresentou o melhor desempenho, sendo que quanto menor o desvio relativo médio, melhor será a heurística.

A Equação 1 apresenta a fórmula utilizada para o cálculo do desvio relativo médio:

$$Desvio\ relativo\ médio = \frac{FO_h - FO_m}{FO_m}\ x\ 100, \quad (1)$$

onde FO_h é o valor da função-objetivo da heurística em análise e FO_m é o valor da função-objetivo da heurística de melhor desempenho.

Após a comparação das heurísticas, foram plotados gráficos de Sucesso x Tarefa, Sucesso x Máquina, Desvio x Tarefa e Desvio x Máquina, permitindo uma análise visual mais simples para o desempenho de cada heurística.

A última comparação foi em relação às médias dos valores de duração total da programação e tempo total de fluxo para cada uma das heurísticas implementadas, para verificar se há correlação entre o número de células do sistema e as funções objetivo.

O equipamento utilizado para realização da experimentação computacional foi um computador com processador Intel Core i5-3230M 2,60GHz, com memória RAM de 4GB e sistema operacional Microsoft Windows 10, com arquitetura 64 bits.

4.1 ANÁLISE DOS RESULTADOS

A experimentação computacional foi realizada para duas funções objetivo: a duração total da programação, ou *makespan*; e o tempo total de fluxo, ou *flow time*. Com a aplicação das heurísticas, buscou-se a minimização de ambas as funções objetivo, obtendo soluções de boa qualidade e em tempo computacional aceitável. Para este estudo, o tempo computacional não foi considerado, visto que o tempo de execução das heurísticas foi semelhante.

4.1.1 Minimização da Duração Total da Programação

A comparação das heurísticas LPT, SPT, Triangular e Triangular Invertida e suas variações, a LPT-A, SPT-A, Triangular-A e Triangular Invertida-A para minimização da duração total da programação gerou as porcentagens de

sucesso de cada heurística por classe, apresentadas nas Tabelas 9, 10 e 11, em pontos percentuais, com o método vencedor em destaque.

Tabela 9: Porcentagem de sucesso por classe para 2 células

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	3	10	**24**	6	10	15	24	8
40x10	10	13	**21**	6	16	7	17	10
40x15	6	11	**20**	11	13	16	18	5
40x20	14	11	10	7	13	**20**	19	6
80x5	**16**	10	13	10	11	13	16	11
80x10	12	10	14	9	18	9	**24**	4
80x15	8	16	18	8	13	12	**21**	4
80x20	12	15	**18**	8	16	9	17	5
120x5	13	15	13	11	**21**	11	13	3
120x10	14	9	**27**	8	7	13	14	8
120x15	**17**	14	16	8	17	8	11	9
120x20	7	16	14	6	14	14	**17**	12
160x5	**21**	16	12	9	17	8	10	7
160x10	14	17	**18**	10	10	9	17	5
160x15	**16**	11	16	8	14	14	16	5
160x20	5	11	13	11	**20**	13	20	7
200x5	14	8	**17**	11	16	13	13	8
200x10	**20**	16	11	9	15	7	11	11
200x15	15	9	**19**	5	10	12	19	11
200x20	11	17	**21**	6	12	13	13	7

Fonte: Autor, 2018

Tabela 10: Porcentagem de sucesso por classe para 4 células

(continua)

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	12	12	19	7	12	17	**20**	1
40x10	14	10	13	8	15	12	**22**	6
40x15	13	13	20	4	12	12	**21**	5
40x20	13	8	**21**	10	8	15	16	9
80x5	12	11	12	11	14	10	**19**	11
80x10	**19**	3	19	8	17	13	14	7
80x15	10	6	**24**	6	10	16	23	5
80x20	11	15	**16**	8	10	19	16	5
120x5	13	11	17	0	**19**	14	18	8
120x10	15	13	**19**	8	10	12	15	8
120x15	14	12	**20**	5	19	15	10	5

Tabela 10: Porcentagem de sucesso por classe para 4 células

(conclusão)

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
120x20	15	16	13	11	9	7	**21**	8
160x5	12	12	15	8	**17**	12	12	12
160x10	14	12	18	6	11	11	**19**	9
160x15	18	8	**20**	10	13	9	17	5
160x20	13	14	18	4	17	5	**19**	10
200x5	12	14	16	9	14	**18**	12	5
200x10	16	11	**18**	9	11	15	15	5
200x15	11	5	**20**	4	19	17	15	9
200x20	8	16	13	11	8	15	**21**	8

Fonte: Autor, 2018

Tabela 11: Porcentagem de sucesso por classe para 8 células

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	8	7	16	7	10	**19**	16	17
40x10	11	10	**19**	6	16	12	16	10
40x15	10	13	17	6	11	12	**19**	12
40x20	11	9	17	5	**20**	14	12	12
80x5	10	15	**23**	8	14	16	10	4
80x10	**17**	11	17	4	16	14	11	10
80x15	7	**22**	15	11	13	14	13	5
80x20	11	13	**19**	7	12	14	16	8
120x5	8	15	**21**	10	18	8	17	3
120x10	17	11	**19**	5	12	11	17	8
120x15	12	9	**21**	8	13	17	12	8
120x20	9	13	17	16	8	12	**20**	5
160x5	12	13	16	9	11	11	**18**	10
160x10	10	15	**20**	9	12	13	16	5
160x15	15	11	13	7	14	8	**25**	7
160x20	16	7	15	8	**18**	12	16	8
200x5	8	11	**18**	13	15	11	12	12
200x10	9	13	**21**	8	13	9	18	9
200x15	10	10	**22**	6	12	12	14	14
200x20	14	15	**22**	8	10	13	13	5

Fonte: Autor, 2018

Os resultados para a soma dos desvios relativos médios são apresentados nas Tabelas 12, 13 e 14, em porcentagem.

Tabela 12: Soma dos desvios relativos médios por classe para 2 células

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	6,87	6,24	**4,57**	7,52	6,01	6,26	4,63	7,00
40x10	4,60	4,99	**3,68**	5,90	4,65	5,03	3,68	5,89
40x15	4,80	3,97	3,81	5,43	4,06	4,23	**3,50**	5,24
40x20	3,90	3,82	**3,08**	4,21	3,29	3,45	3,26	4,19
80x5	3,99	4,57	**3,22**	5,01	4,05	4,71	3,98	4,93
80x10	3,35	3,65	3,29	4,28	3,46	4,15	**2,91**	4,66
80x15	3,78	3,15	2,62	3,84	3,48	3,26	**2,13**	4,03
80x20	2,82	2,45	**2,07**	3,41	2,40	2,76	2,13	3,37
120x5	3,17	3,37	3,18	4,27	2,97	4,11	**2,76**	4,52
120x10	3,58	3,16	**2,28**	3,89	3,38	2,97	2,71	3,67
120x15	**2,35**	3,13	2,41	3,22	2,40	2,97	2,74	3,43
120x20	2,76	2,57	**2,39**	2,76	2,96	2,47	2,44	3,18
160x5	**3,15**	3,23	3,20	3,81	3,16	3,72	3,15	3,48
160x10	2,97	2,84	**2,32**	3,42	2,77	3,08	2,56	3,46
160x15	2,59	2,54	**2,11**	3,18	2,49	2,67	2,20	2,88
160x20	2,33	2,23	**1,80**	2,72	1,91	2,36	2,00	2,72
200x5	2,80	3,06	**2,78**	3,17	2,85	2,87	2,81	3,13
200x10	**2,12**	2,34	2,43	2,47	2,43	2,77	2,50	2,85
200x15	2,07	2,46	**1,86**	2,70	2,21	2,47	2,05	2,57
200x20	2,54	1,93	**1,86**	2,91	2,74	2,34	2,11	2,70

Fonte: Autor, 2018

Tabela 13: Soma dos desvios relativos médios por classe para 4 células

(continua)

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	7,76	6,83	**4,86**	7,43	6,47	6,00	5,54	8,30
40x10	4,80	5,29	4,50	5,66	4,13	5,37	**4,03**	6,20
40x15	4,87	4,71	3,75	5,27	4,04	4,14	**3,61**	4,97
40x20	**3,24**	3,99	3,26	4,16	3,96	3,61	3,85	4,45
80x5	5,21	5,22	4,05	5,59	4,86	5,09	**3,92**	5,78
80x10	3,53	4,83	**3,17**	5,22	3,82	4,38	3,40	4,95
80x15	3,65	3,79	2,73	4,57	3,78	3,53	**2,54**	4,50
80x20	3,10	3,06	2,51	3,39	3,15	2,84	**2,46**	3,64
120x5	3,68	4,04	3,39	5,03	3,75	4,65	**3,15**	5,01
120x10	3,57	3,70	3,17	4,55	3,57	3,59	**2,90**	3,95

Tabela 13: Soma dos desvios relativos médios por classe para 4 células

(conclusão)

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
120x15	**2,50**	3,29	2,62	3,79	2,81	3,21	2,93	3,62
120x20	2,82	2,62	2,54	2,90	3,22	2,92	**2,22**	3,25
160x5	3,77	3,91	3,34	3,91	**3,18**	3,75	3,61	3,89
160x10	3,51	3,28	**2,97**	4,38	3,34	3,59	3,02	3,92
160x15	2,92	2,86	**2,27**	3,45	2,81	2,99	2,36	3,37
160x20	2,61	2,32	2,08	3,34	2,51	2,66	**2,00**	3,25
200x5	3,64	3,56	**3,05**	3,40	3,46	3,08	3,31	4,02
200x10	2,66	2,93	2,61	3,52	**2,49**	3,20	2,69	3,53
200x15	2,59	2,78	**1,99**	3,09	2,65	2,42	2,20	3,24
200x20	2,64	2,39	**2,02**	2,86	2,53	2,05	2,05	2,63

Fonte: Autor, 2018

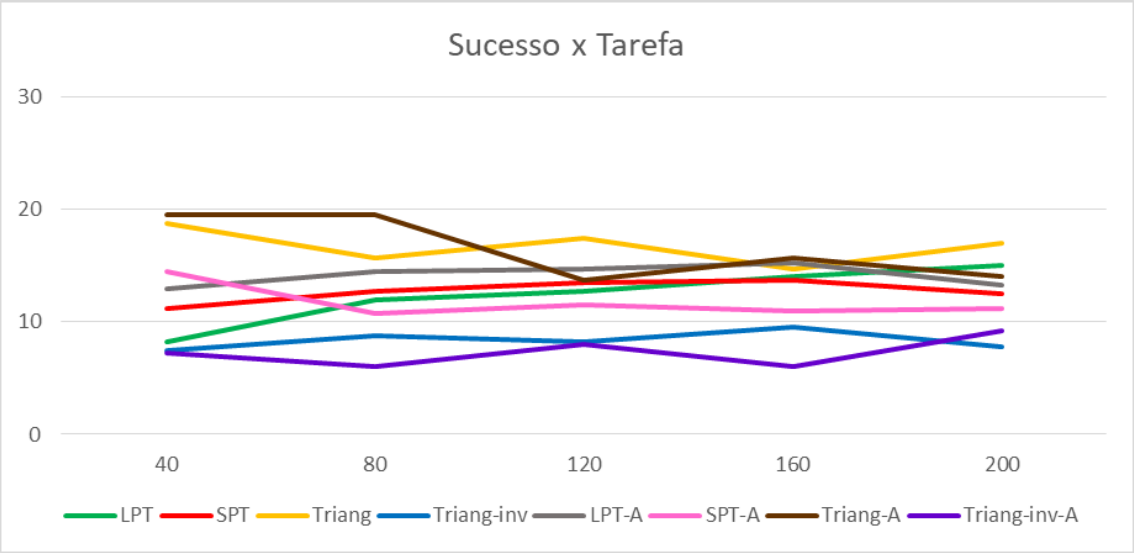
Tabela 14: Soma dos desvios relativos médios por classe para 8 células

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	7,77	7,13	5,42	7,30	6,89	6,61	**5,27**	7,19
40x10	5,24	6,17	**4,10**	4,95	4,17	5,17	4,29	5,00
40x15	5,21	4,92	4,23	5,03	4,32	4,09	**3,67**	4,63
40x20	4,05	4,23	3,48	4,09	**3,28**	3,91	3,78	4,12
80x5	6,35	5,98	**4,44**	5,92	5,24	5,25	4,78	6,98
80x10	4,22	4,89	**3,24**	4,68	3,78	4,25	3,88	5,35
80x15	3,91	3,42	3,16	4,30	3,69	3,00	**2,96**	4,82
80x20	3,00	3,29	**2,54**	3,68	2,86	2,85	3,24	3,89
120x5	5,18	5,03	**3,44**	5,65	4,16	5,06	3,83	6,21
120x10	3,78	4,78	**2,95**	4,38	3,75	3,62	3,05	4,23
120x15	3,25	3,45	2,92	3,74	**2,88**	2,97	2,95	4,53
120x20	3,04	2,86	2,52	2,90	2,83	2,88	**2,18**	3,49
160x5	4,38	4,46	3,98	4,86	4,13	4,01	**3,86**	4,81
160x10	3,92	3,33	2,79	3,94	3,32	3,38	**2,61**	4,18
160x15	2,86	2,98	2,72	3,29	2,48	3,18	**2,35**	3,52
160x20	2,29	2,92	**2,10**	3,19	2,49	2,58	2,51	3,37
200x5	3,76	3,68	**3,44**	3,80	3,51	3,61	3,59	3,97
200x10	3,05	3,27	**2,82**	3,77	2,86	3,24	3,06	3,87
200x15	3,29	3,05	**2,38**	3,43	2,88	2,88	2,77	3,32
200x20	2,76	2,35	**1,93**	3,09	2,74	2,33	2,29	3,36

Fonte: Autor, 2018

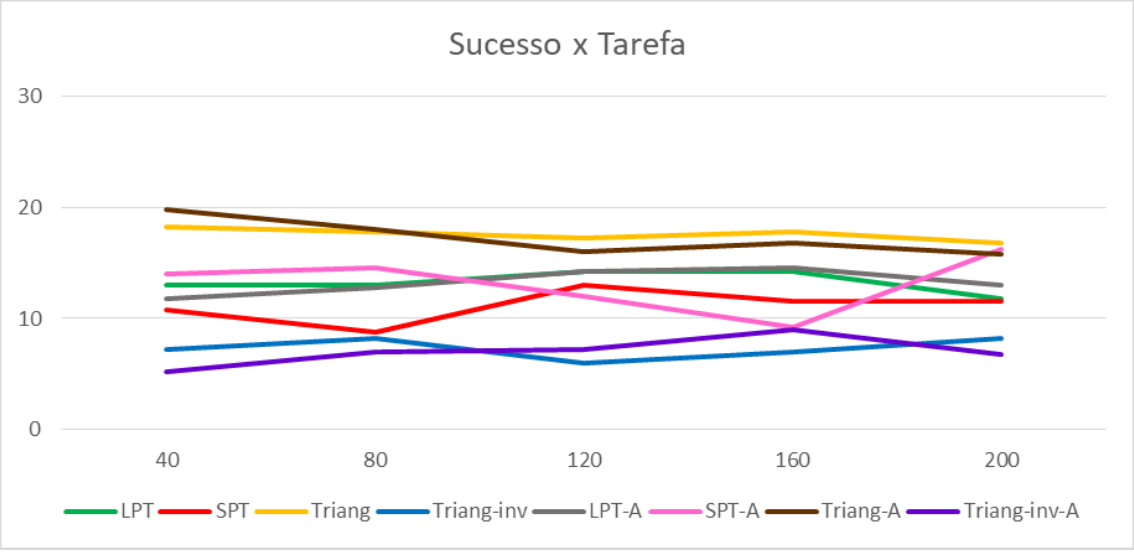
Para uma melhor análise, foram plotados gráficos de Sucesso x Tarefa, Sucesso x Máquina, Desvio x Tarefa e Desvio x Máquina, conforme as Figuras 8, 9, 10 e 11 a seguir.

Figura 8: Comparação do percentual de sucesso x tarefa para 2 células



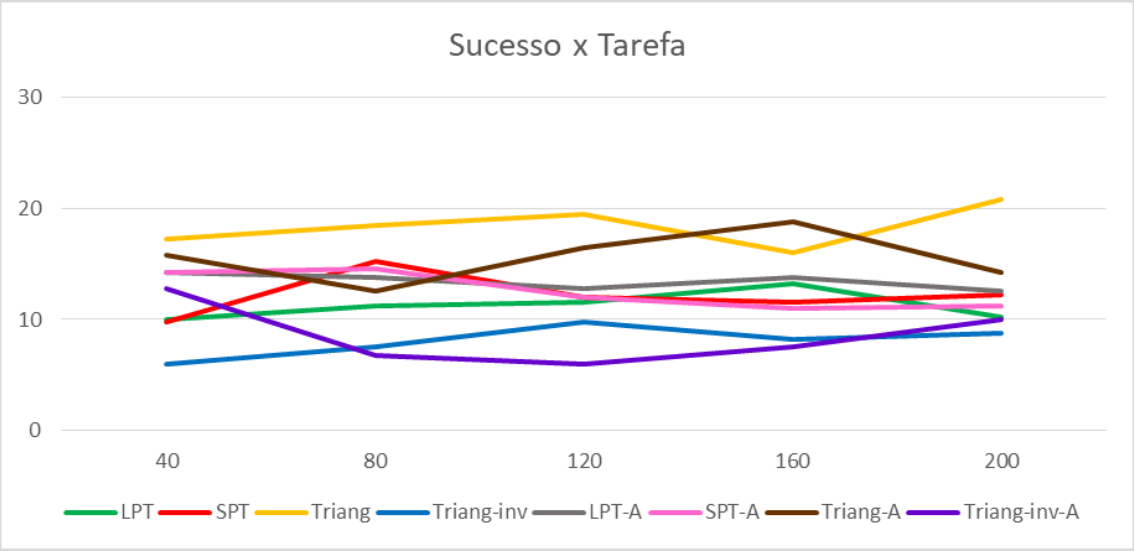
Fonte: Autor, 2018

Figura 9: Comparação do percentual de sucesso x tarefa para 4 células



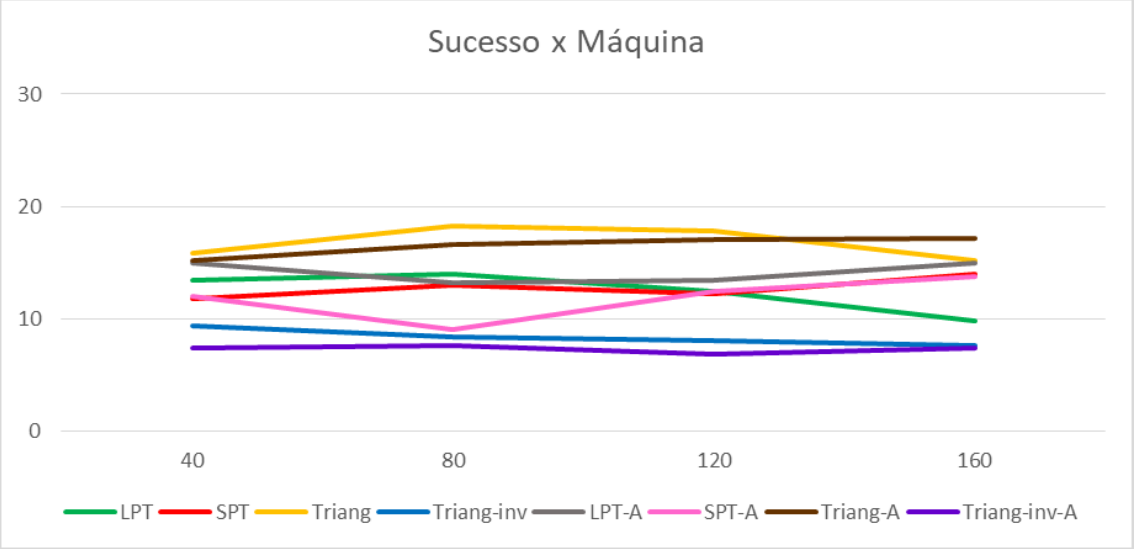
Fonte: Autor, 2018

Figura 10: Comparação do percentual de sucesso x tarefa para 8 células



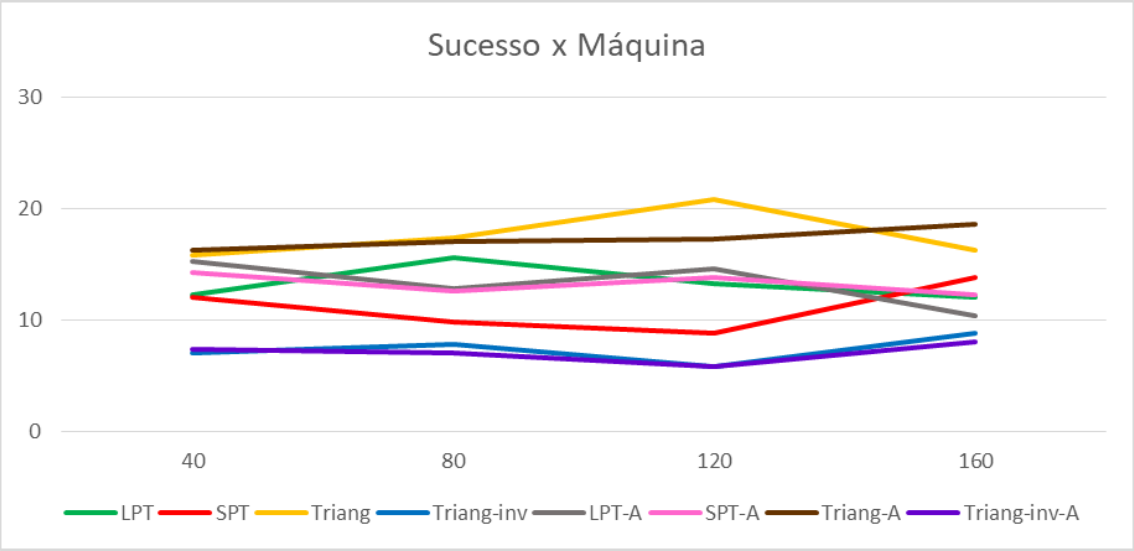
Fonte: Autor, 2018

Figura 11: Comparação do percentual de sucesso x máquina para 2 células



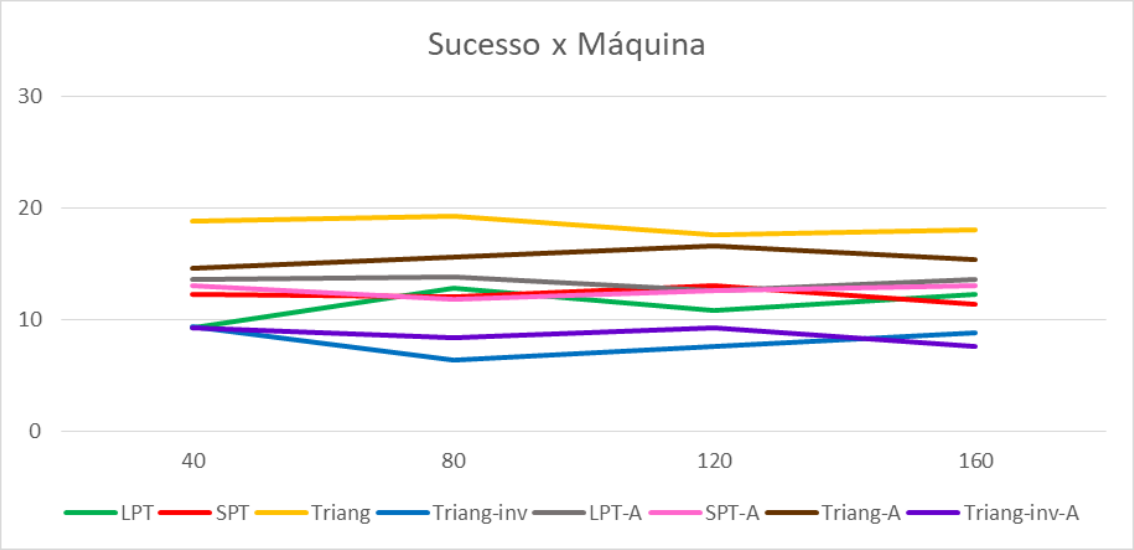
Fonte: Autor, 2018

Figura 12: Comparação do percentual de sucesso x máquina para 4 células



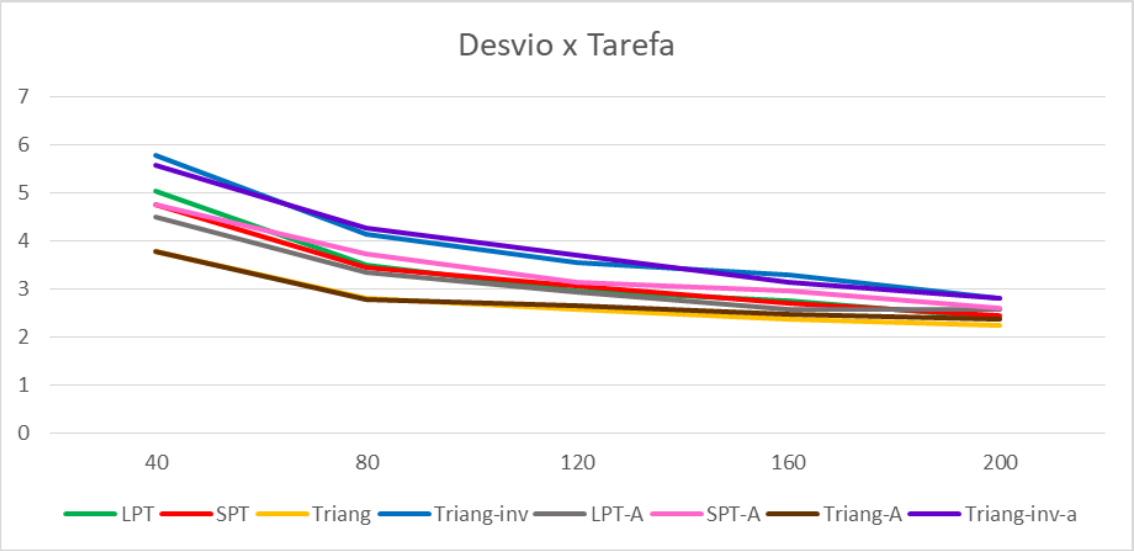
Fonte: Autor, 2018

Figura 13: Comparação do percentual de sucesso x máquina para 8 células



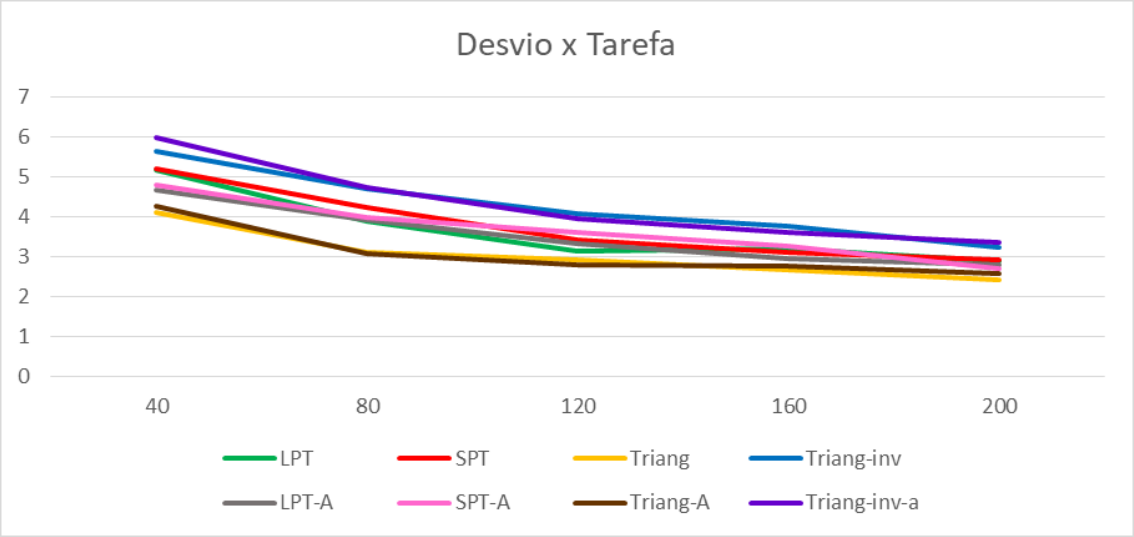
Fonte: Autor, 2018

Figura 14: Comparação do desvio relativo médio x tarefa para 2 células



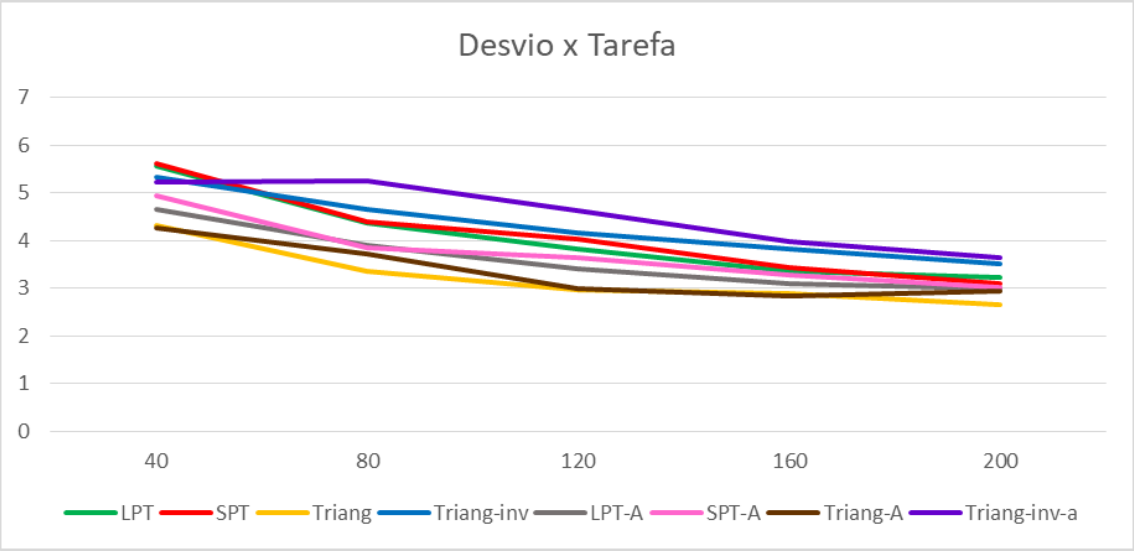
Fonte: Autor, 2018

Figura 15: Comparação do desvio relativo médio x tarefa para 4 células



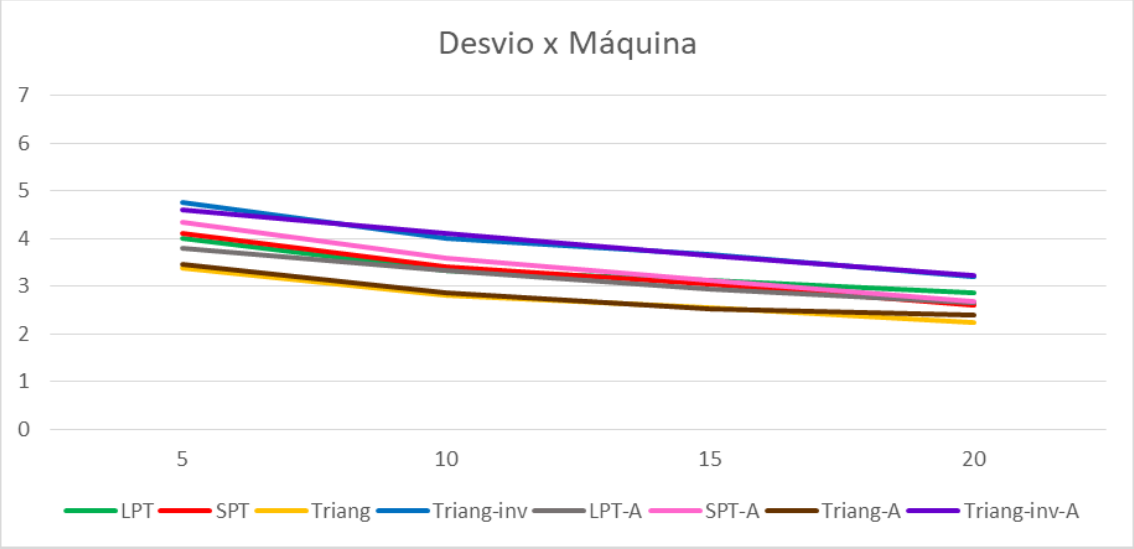
Fonte: Autor, 2018

Figura 16: Comparação do desvio relativo médio x tarefa para 8 células



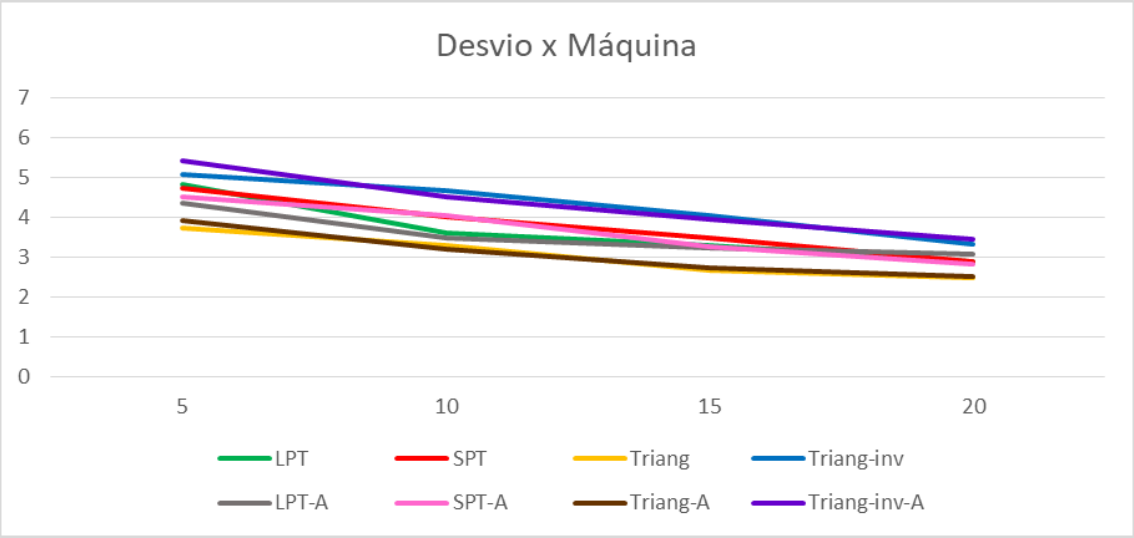
Fonte: Autor, 2018

Figura 17: Comparação do desvio relativo médio x máquina para 2 células



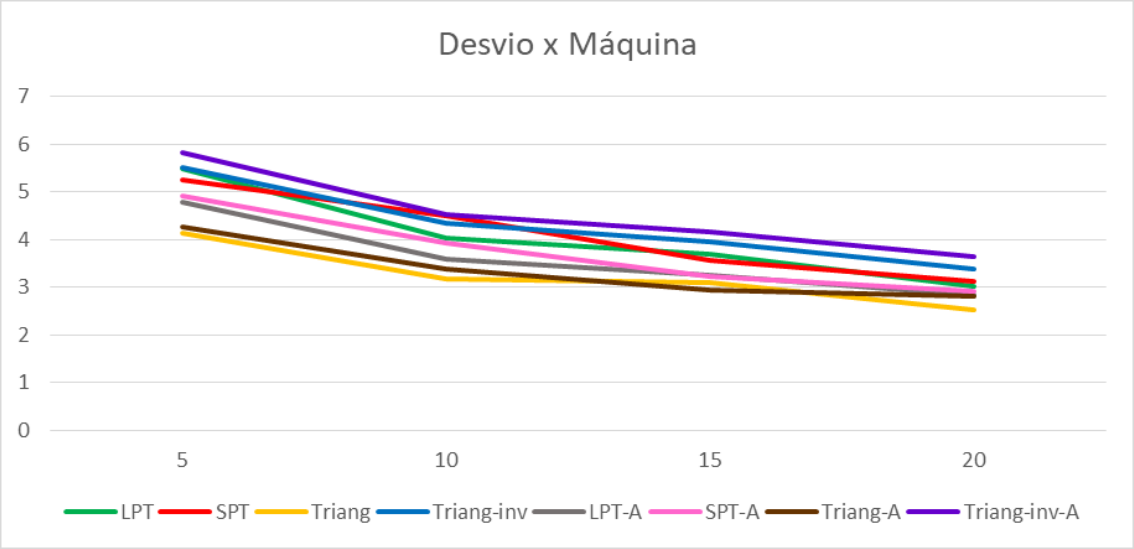
Fonte: Autor, 2018

Figura 18: Comparação do desvio relativo médio x máquina para 4 células



Fonte: Autor, 2018

Figura 19: Comparação do desvio relativo médio x máquina para 8 células



Fonte: Autor, 2018

A análise dos gráficos apresentados nas Figuras 8, 9, 10 e 11 permitem verificar que, apesar de os métodos Triangular e sua variação Triangular-A apresentarem desempenho ligeiramente superior aos demais para este experimento, o desempenho geral das oito heurísticas foi semelhante neste estudo, com nenhuma delas se sobressaindo em relação às demais.

É possível observar também que a proposta de ordenação inicial alternada não possui influência no valor da duração total da programação, visto que os valores obtidos para cada heurística e suas variações são próximos.

Observa-se também que o desvio relativo médio tende a diminuir à medida que aumentam o número de tarefas e o número de máquinas.

4.1.2 Minimização do Tempo Total de Fluxo

A comparação das heurísticas LPT, SPT, Triangular e Triangular Invertida e suas variações, a LPT-A, SPT-A, Triangular-A e Triangular Invertida-A para minimização do tempo total de fluxo gerou as porcentagens de sucesso de cada heurística por classe, apresentadas nas Tabelas 15, 16 e 17, em pontos percentuais, com o método vencedor em destaque.

Tabela 15: Porcentagem de sucesso por classe para 2 células

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	0	48	1	0	0	**51**	0	0
40x10	0	46	2	0	0	**51**	1	0
40x15	0	43	0	0	0	**55**	2	0
40x20	0	42	1	0	0	**52**	5	0
80x5	0	46	1	0	0	**53**	0	0
80x10	0	**57**	0	0	0	43	0	0
80x15	0	48	1	0	0	**51**	0	0
80x20	0	43	0	0	0	**56**	1	0
120x5	0	**54**	0	0	0	46	0	0
120x10	0	**52**	0	0	0	48	0	0
120x15	0	43	0	0	0	**56**	1	0
120x20	0	48	0	0	0	**52**	0	0
160x5	0	**54**	0	0	0	46	0	0
160x10	0	**53**	0	0	0	47	0	0
160x15	0	**50**	0	0	0	50	0	0
160x20	0	**56**	0	0	0	44	0	0
200x5	0	**50**	0	0	0	50	0	0
200x10	0	**50**	0	0	0	49	1	0
200x15	0	47	0	0	0	**53**	0	0
200x20	0	**54**	0	0	0	46	0	0

Fonte: Autor, 2018

Tabela 16: Porcentagem de sucesso por classe para 4 células

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	0	38	1	0	0	**61**	0	0
40x10	0	44	3	0	0	**51**	2	0
40x15	0	40	0	0	0	**58**	2	0
40x20	0	48	2	0	0	**50**	0	0
80x5	0	49	0	0	0	**51**	0	0
80x10	0	44	3	0	0	**53**	0	0
80x15	0	49	0	0	0	**50**	1	0
80x20	0	**51**	0	0	0	49	0	0
120x5	0	**53**	0	0	0	47	0	0
120x10	0	47	0	0	0	**53**	0	0
120x15	0	49	0	0	0	**50**	1	0
120x20	0	**57**	0	0	0	42	1	0
160x5	0	**51**	0	0	0	49	0	0
160x10	0	**55**	0	0	0	45	0	0
160x15	0	**51**	0	0	0	49	0	0
160x20	0	**53**	0	0	0	47	0	0
200x5	0	41	0	0	0	**59**	0	0
200x10	0	**54**	1	0	0	45	0	0
200x15	0	47	0	0	0	**53**	0	0
200x20	0	**50**	0	0	0	50	0	0

Fonte: Autor, 2018

Tabela 17: Porcentagem de sucesso por classe para 8 células

(continua)

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	0	37	1	0	0	**62**	0	0
40x10	0	28	0	0	0	**71**	1	0
40x15	0	31	1	0	0	**66**	2	0
40x20	0	40	0	0	0	**60**	0	0
80x5	0	49	1	0	0	**50**	0	0
80x10	0	28	2	0	0	**69**	1	0
80x15	0	**51**	0	0	0	49	0	0
80x20	0	38	0	0	0	**62**	0	0
120x5	0	**52**	0	0	0	48	0	0
120x10	0	45	1	0	0	**54**	0	0
120x15	0	47	0	0	0	**53**	0	0
120x20	0	**50**	0	0	0	50	0	0
160x5	0	42	0	0	0	**58**	0	0

Tabela 17: Porcentagem de sucesso por classe para 8 células

(conclusão)

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
160x10	0	**58**	0	0	0	42	0	0
160x15	0	**50**	1	0	0	49	0	0
160x20	0	**54**	0	0	0	46	0	0
200x5	0	43	0	0	0	**57**	0	0
200x10	0	46	0	0	0	**54**	0	0
200x15	0	**53**	0	0	0	47	0	0
200x20	0	48	0	0	0	**52**	0	0

Fonte: Autor, 2018

Os resultados para a soma dos desvios relativos médios são apresentados nas Tabelas 18, 19 e 20, em porcentagem.

Tabela 18: Soma dos desvios relativos médios por classe para 2 células

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	30,67	2,06	13,25	19,70	29,93	**1,90**	12,98	20,06
40x10	21,20	1,52	8,87	15,19	20,51	**1,35**	8,61	14,95
40x15	18,10	1,19	7,44	13,59	17,36	**0,98**	6,65	13,43
40x20	15,53	1,14	6,18	11,44	14,89	**0,84**	6,37	11,71
80x5	28,28	1,83	12,70	17,03	27,89	**1,70**	12,89	16,82
80x10	18,91	**0,95**	8,65	12,16	19,12	1,41	8,29	12,86
80x15	16,44	1,11	6,58	11,49	16,24	**0,93**	6,34	11,14
80x20	14,22	0,86	5,83	10,08	13,94	**0,79**	5,84	10,14
120x5	28,49	**0,84**	13,78	16,62	27,94	1,47	13,55	16,36
120x10	19,58	**1,00**	8,48	12,18	19,39	1,06	8,82	12,03
120x15	14,96	0,82	6,61	9,41	14,88	**0,79**	6,73	9,71
120x20	13,28	0,76	5,89	8,85	13,48	**0,74**	5,98	9,14
160x5	28,82	**0,94**	14,16	16,09	28,85	1,18	13,82	15,94
160x10	19,10	**0,95**	9,08	11,46	18,98	1,03	9,30	11,41
160x15	14,95	0,93	6,80	9,34	14,92	**0,84**	6,80	9,26
160x20	12,97	**0,73**	5,71	8,36	12,74	0,90	5,63	8,50
200x5	29,66	**1,06**	15,01	16,85	29,91	1,13	14,88	16,53
200x10	18,71	**0,80**	9,34	10,69	19,24	0,85	9,27	11,04
200x15	15,08	**0,78**	6,99	9,24	14,93	0,94	7,05	9,16
200x20	13,15	**0,51**	5,62	8,54	13,12	0,82	5,80	8,04

Fonte: Autor, 2018

Tabela 19: Soma dos desvios relativos médios por classe para 4 células

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	30,99	2,52	13,35	21,78	30,42	**1,32**	13,34	22,67
40x10	22,04	1,63	9,38	16,72	21,62	**1,24**	9,05	16,91
40x15	19,67	1,56	7,62	15,34	18,56	**0,98**	7,58	14,83
40x20	16,81	**0,99**	7,50	13,80	17,11	1,06	7,59	13,87
80x5	28,21	**1,69**	12,93	18,10	27,71	1,73	12,46	17,63
80x10	19,36	1,38	8,44	13,85	19,46	**0,92**	8,43	13,63
80x15	17,82	1,07	7,14	13,55	17,56	**1,00**	6,84	12,88
80x20	15,48	0,93	6,06	11,41	15,73	**0,85**	6,10	11,70
120x5	27,36	**1,25**	12,88	17,43	27,43	1,60	12,81	17,18
120x10	19,47	1,18	9,04	13,30	19,53	**0,97**	8,83	12,96
120x15	15,86	1,02	6,54	11,06	15,80	**0,82**	6,57	11,23
120x20	14,88	**0,75**	6,33	10,61	14,96	1,08	6,34	10,66
160x5	28,19	1,46	13,20	16,27	27,81	**1,17**	13,62	16,34
160x10	19,32	**0,94**	9,02	12,58	19,12	1,31	8,69	12,30
160x15	15,71	**0,81**	6,85	10,45	15,65	1,03	6,71	10,55
160x20	13,89	0,76	5,85	9,53	13,83	**0,75**	5,90	9,78
200x5	28,50	1,58	13,97	16,61	28,84	**0,75**	13,98	16,66
200x10	18,67	1,03	8,96	11,56	18,47	**0,98**	8,78	11,80
200x15	15,49	0,87	7,08	10,23	15,60	**0,83**	7,00	10,22
200x20	13,91	1,06	6,15	9,53	13,93	**0,82**	5,92	9,22

Fonte: Autor, 2018

Tabela 20: Soma dos desvios relativos médios por classe para 8 células

(continua)

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
40x5	30,93	2,33	14,09	24,94	30,08	**1,35**	12,74	23,85
40x10	23,67	2,50	10,05	19,41	22,25	**0,46**	9,92	18,47
40x15	20,02	1,49	8,70	17,21	19,51	**0,64**	8,05	16,29
40x20	17,91	1,40	8,24	15,43	16,79	**0,71**	7,65	14,90
80x5	28,65	2,10	13,67	20,10	28,13	**1,61**	13,20	20,37
80x10	21,38	1,93	9,59	15,93	20,90	**0,71**	9,72	16,88
80x15	19,92	1,32	8,39	15,63	19,20	**0,80**	8,36	16,25
80x20	17,13	1,33	6,73	13,98	17,10	**0,59**	7,54	14,26
120x5	27,96	**1,35**	12,72	18,54	27,29	1,43	12,35	18,45
120x10	20,72	1,72	9,18	15,47	20,89	**0,97**	9,17	14,49
120x15	17,84	1,26	7,55	13,62	17,67	**0,99**	7,30	13,62
120x20	16,42	0,92	6,82	12,62	16,40	**0,85**	6,99	12,69
160x5	27,68	1,94	13,36	18,28	27,86	**1,47**	13,05	17,83

Tabela 20: Soma dos desvios relativos médios por classe para 8 células

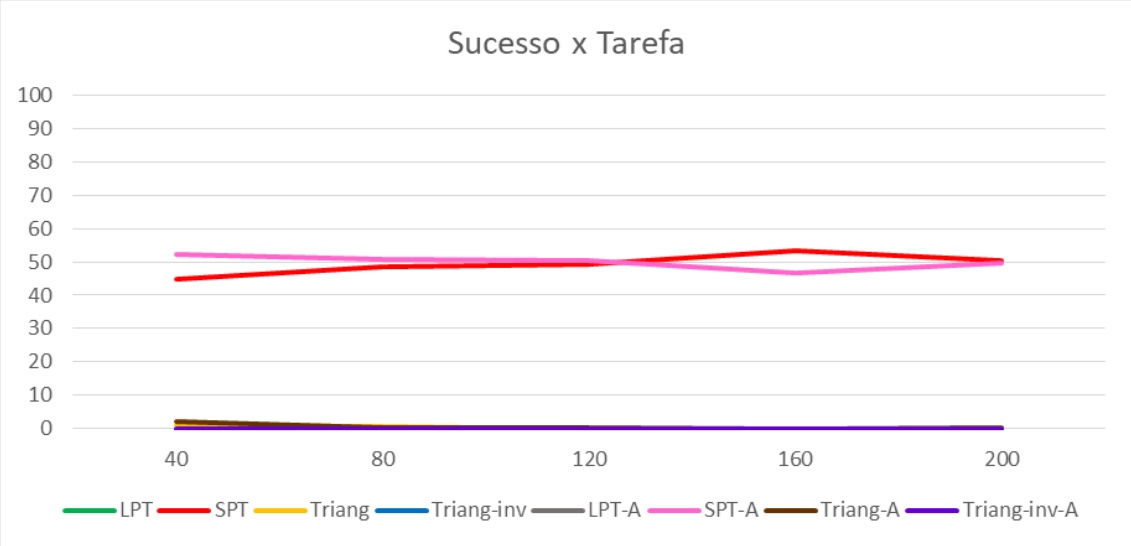
(conclusão)

Classe	LPT	SPT	Triang	Triang-inv	LPT-A	SPT-A	Triang-A	Triang-in-A
160x10	20,68	**0,93**	9,01	14,21	20,24	1,34	8,96	14,34
160x15	17,50	**0,90**	7,50	12,32	17,09	1,17	7,06	12,44
160x20	15,32	0,99	6,39	11,36	15,27	**0,93**	6,35	11,64
200x5	28,29	1,87	14,31	17,65	28,12	**1,02**	13,95	17,84
200x10	19,62	1,35	9,11	13,37	19,40	**0,88**	9,27	13,29
200x15	16,50	**0,83**	7,26	11,83	16,45	0,92	7,54	11,55
200x20	15,12	1,01	6,29	10,52	15,08	**0,74**	6,18	11,07

Fonte: Autor, 2018

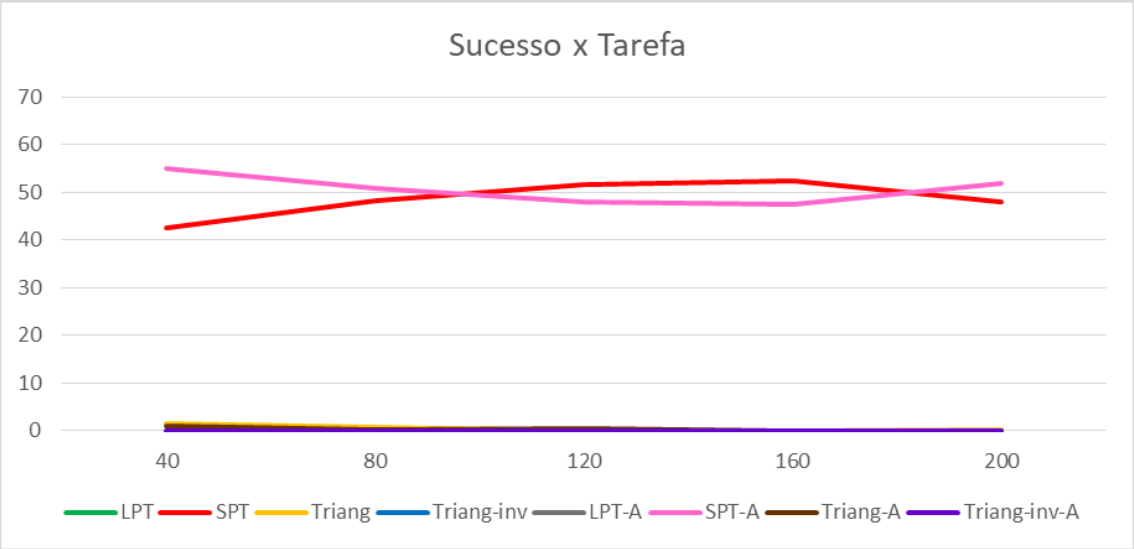
Da mesma forma como para a análise da duração total da programação, foram plotados os gráficos de Sucesso x Tarefa, Sucesso x Máquina, Desvio x Tarefa e Desvio x Máquina, conforme as Figuras 12, 13, 14 e 15.

Figura 20: Comparação do percentual de sucesso x tarefa para 2 células



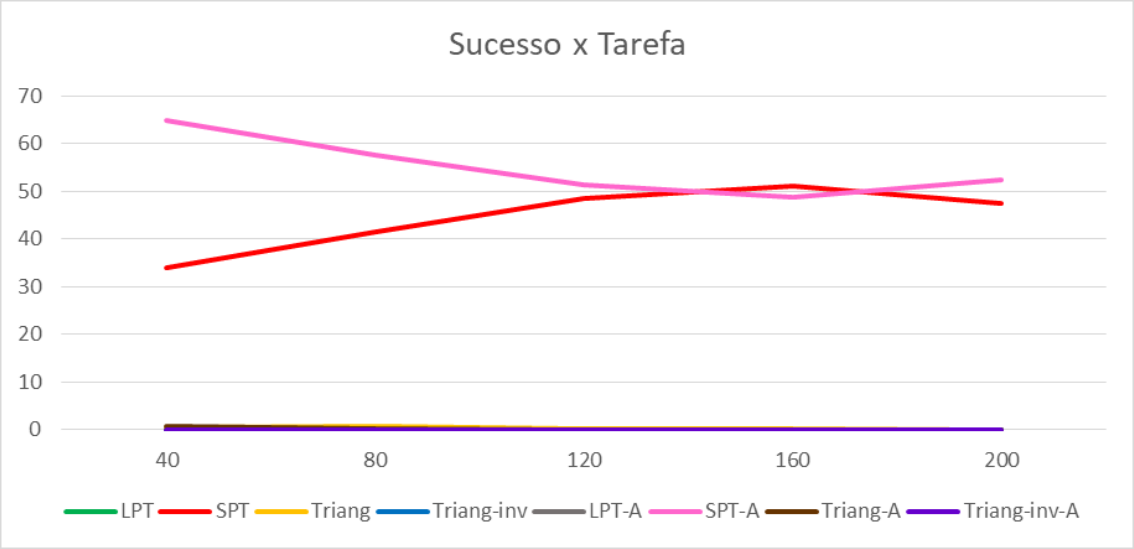
Fonte: Autor, 2018

Figura 21: Comparação do percentual de sucesso x tarefa para 4 células



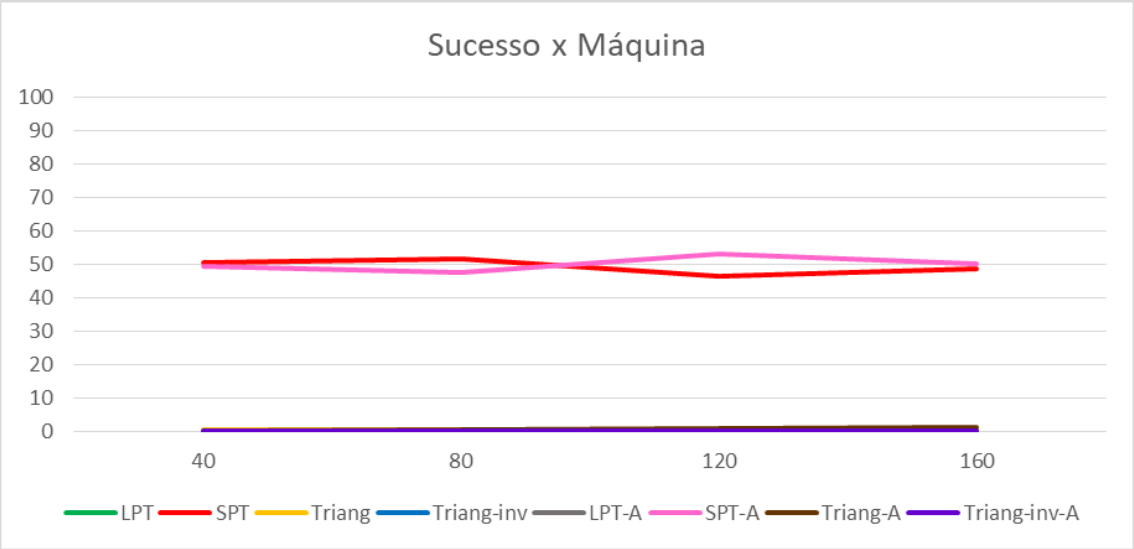
Fonte: Autor, 2018

Figura 22: Comparação do percentual de sucesso x tarefa para 8 células



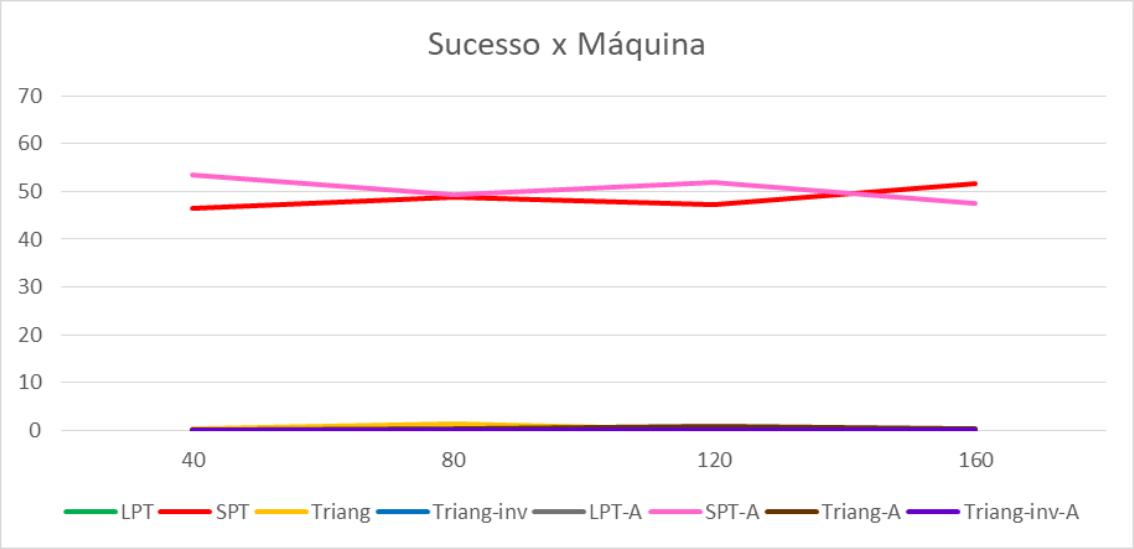
Fonte: Autor, 2018

Figura 23: Comparação do percentual de sucesso x máquina para 2 células



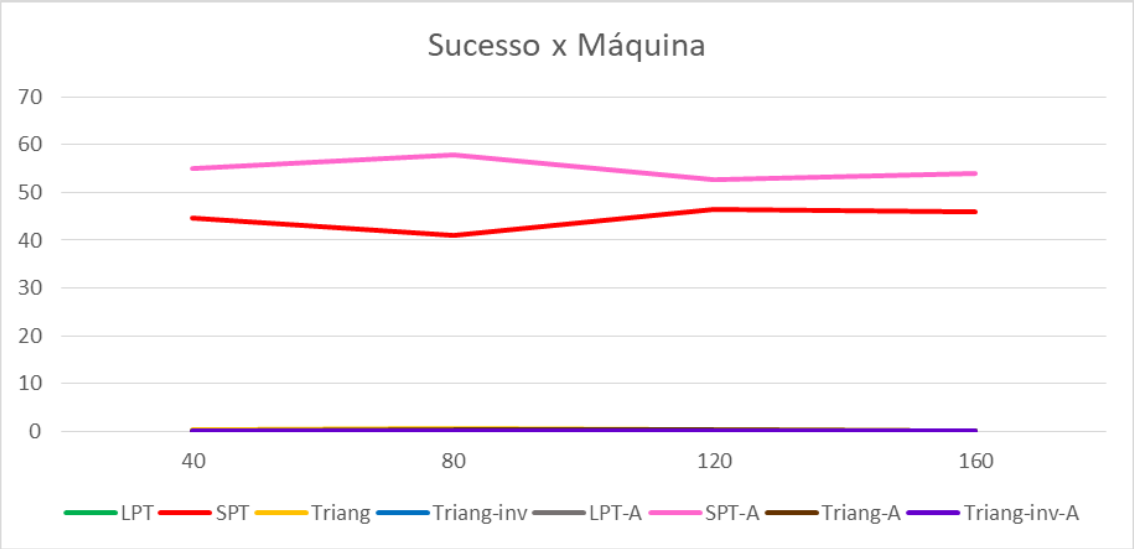
Fonte: Autor, 2018

Figura 24: Comparação do percentual de sucesso x máquina para 4 células



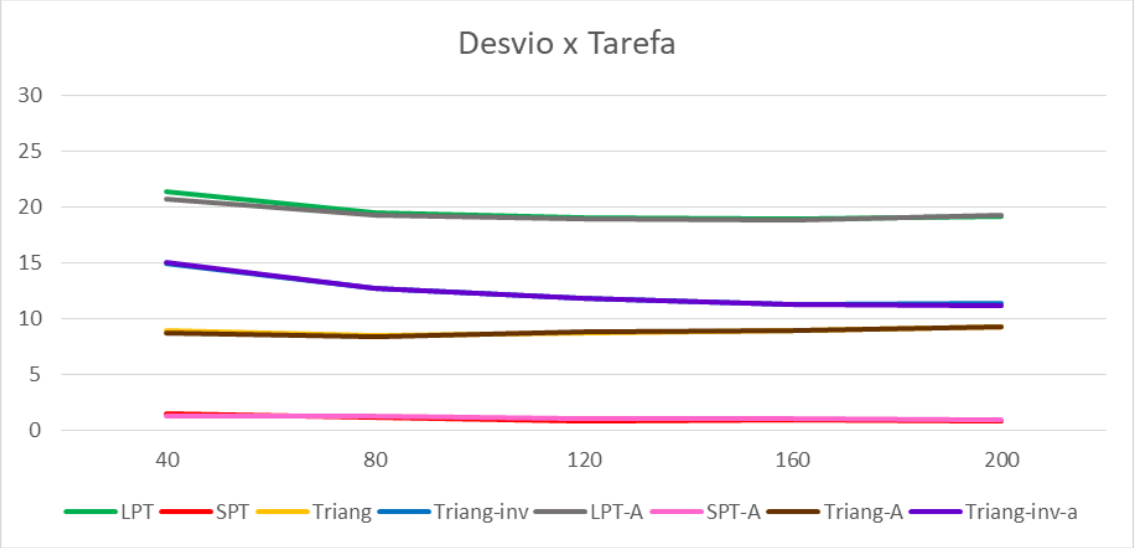
Fonte: Autor, 2018

Figura 25: Comparação do percentual de sucesso x máquina para 8 células



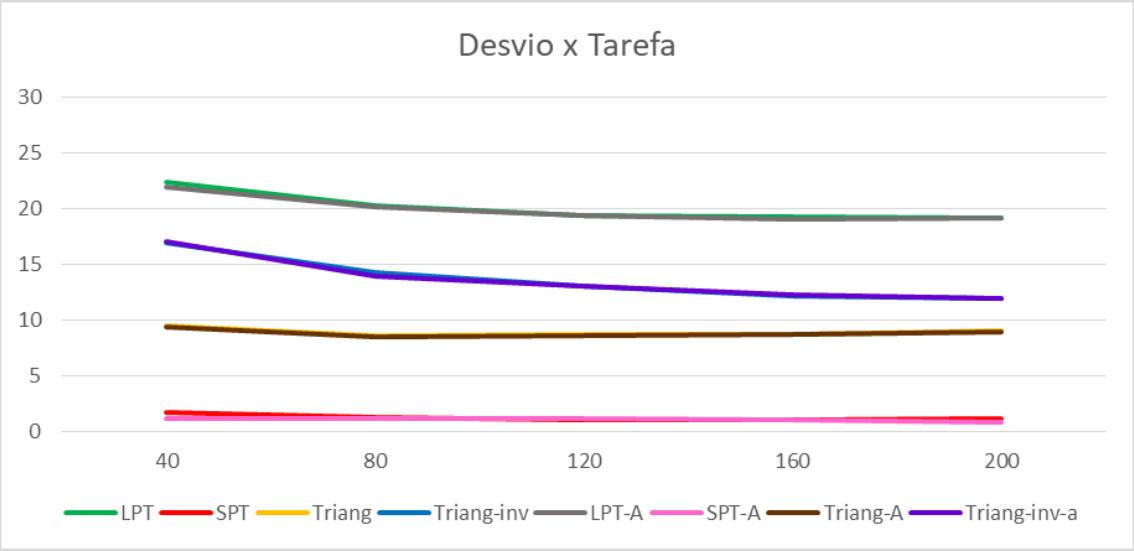
Fonte: Autor, 2018

Figura 26: Comparação do desvio relativo médio x tarefa para 2 células



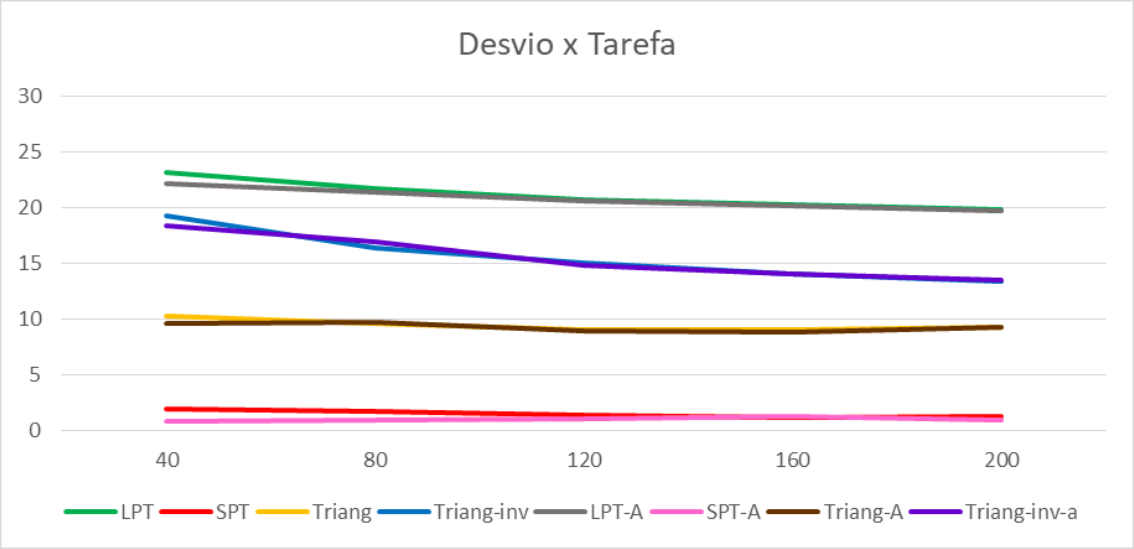
Fonte: Autor, 2018

Figura 27: Comparação do desvio relativo médio x tarefa para 4 células



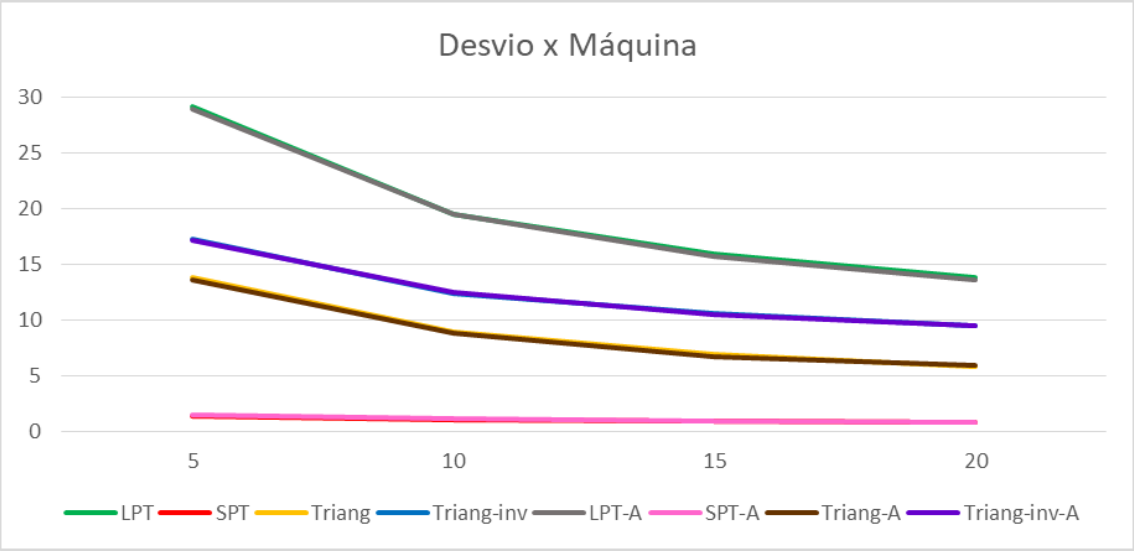
Fonte: Autor, 2018

Figura 28: Comparação do desvio relativo médio x tarefa para 8 células



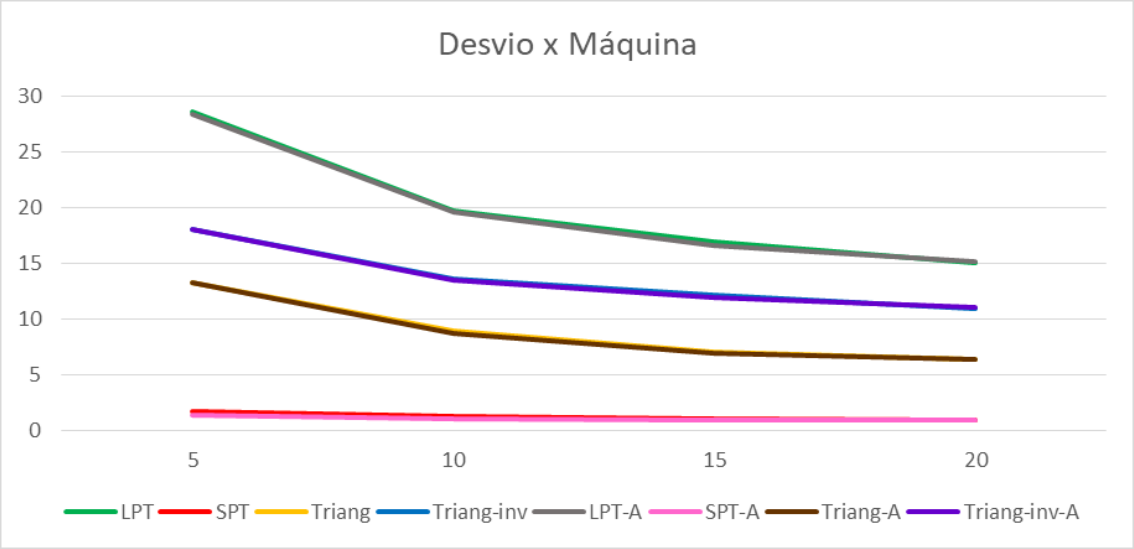
Fonte: Autor, 2018

Figura 29: Comparação do desvio relativo médio x máquina para 2 células



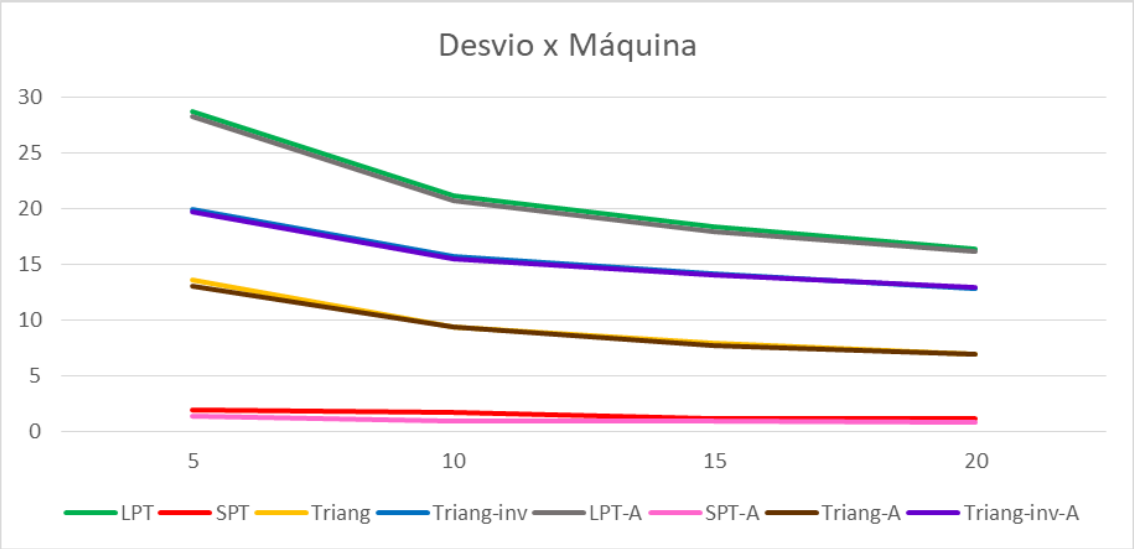
Fonte: Autor, 2018

Figura 30: Comparação do desvio relativo médio x máquina para 4 células



Fonte: Autor, 2018

Figura 31: Comparação do desvio relativo médio x máquina para 8 células



Fonte: Autor, 2018

A análise dos gráficos apresentados nas Figuras 12, 13, 14 e 15 permite afirmar que, para este estudo, a heurística SPT e sua variação SPT-A apresentaram desempenho superior às demais para a minimização do tempo total de fluxo, sendo que as duas somaram praticamente a totalidade dos sucessos e apresentando baixo desvio relativo médio.

Da mesma forma como na minimização da duração total da programação, verificou-se que a proposta de ordenação alternada também não apresentou impacto significativo para redução do tempo total de fluxo neste estudo, com os valores de cada heurística e sua variação próximos.

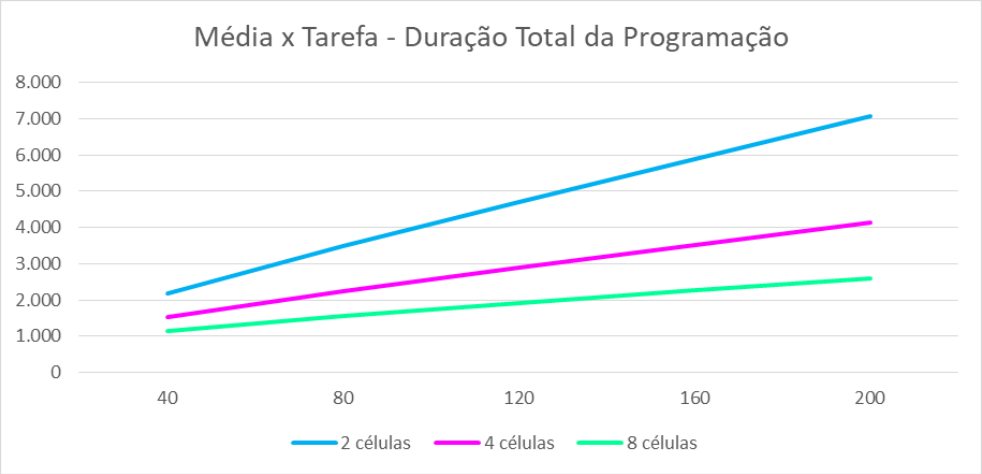
Nota-se neste caso que o número de tarefas não influencia no desvio relativo médio; porém, este tende a diminuir à medida que aumenta o número de máquinas do sistema.

4.1.3 Comparação entre Número de Células de Processamento

Este estudo foi realizado para um problema *flowshop* permutacional distribuído, ou seja, um problema onde existe um número maior que 1 de células para processamento das tarefas. O número de células definido para o estudo foi de 2, 4 e 8 células, com intuito de verificar se o número de células impacta significativamente na minimização das funções objetivo definidas.

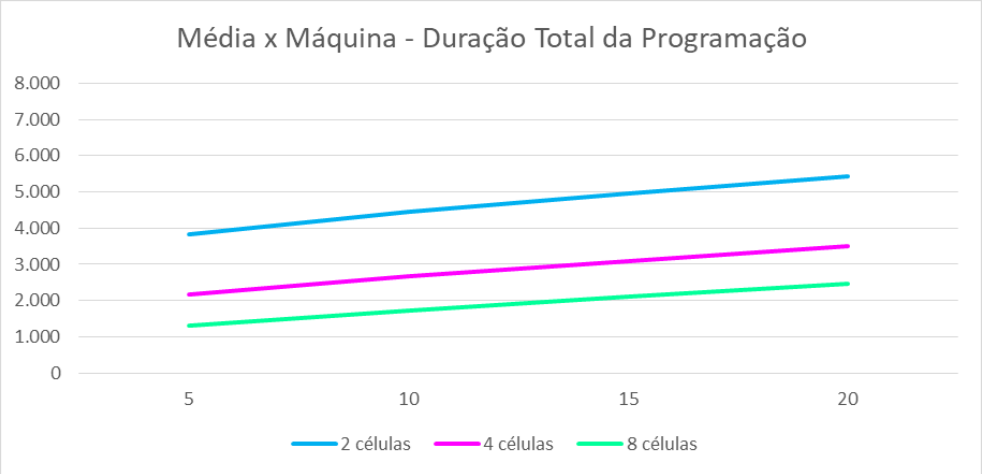
Para esta comparação, foi calculada a média dos valores obtidos para cada função objetivo e para cada número de células de processamento. Assim, foi possível analisar graficamente o impacto da variação do número de células do sistema no valor das funções objetivo, conforme apresentados pelas Figuras 16 e 17 para a duração total da programação e pelas Figuras 18 e 19 para tempo total de fluxo.

Figura 32: Comparação da média x tarefa para duração total da programação



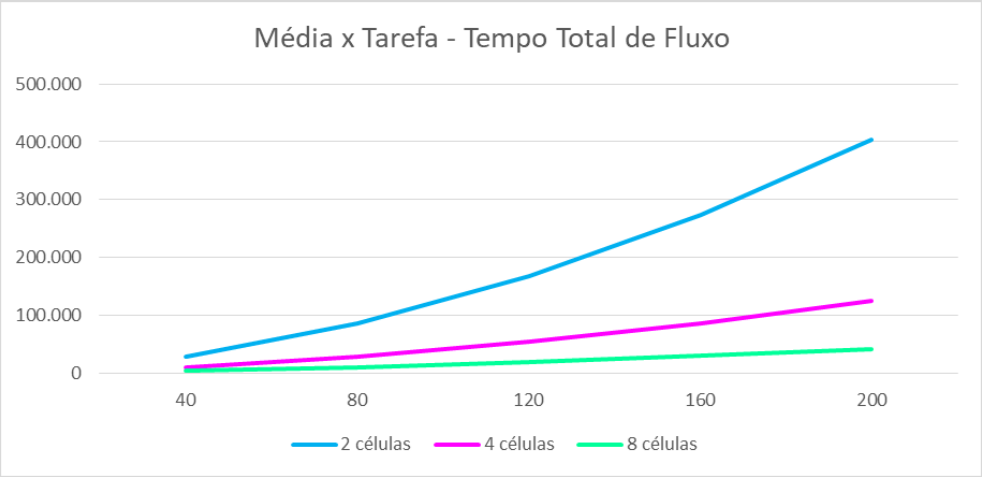
Fonte: Autor, 2018

Figura 33: Comparação da média x máquina para duração total da programação



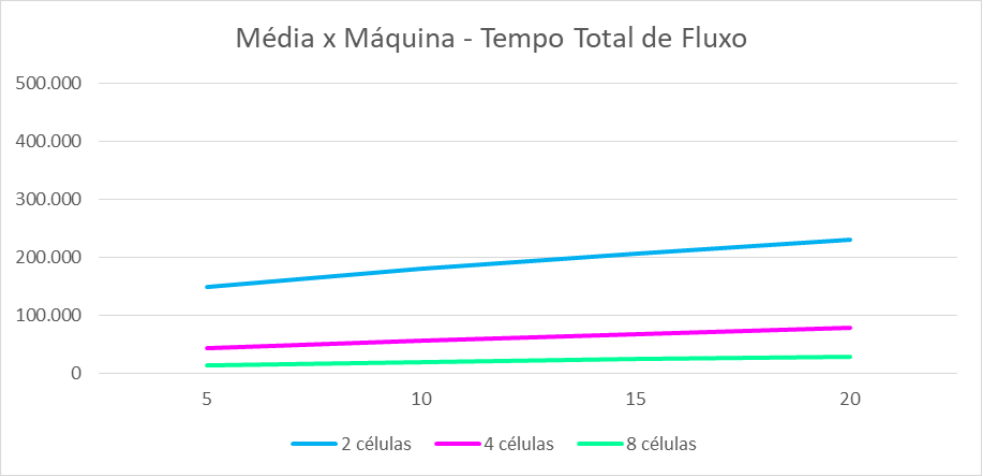
Fonte: Autor, 2018

Figura 34: Comparação da média x tarefa para tempo total de fluxo



Fonte: Autor, 2018

Figura 35: Comparação da média x máquina para tempo total de fluxo



Fonte: Autor, 2018

A análise dos gráficos apresentados permite afirmar que, para este estudo, tanto para a duração total da programação quanto para o tempo total de fluxo, o aumento do número de máquinas reduz significativa e linearmente os valores para as funções objetivo, mostrando que a distribuição das tarefas em células, princípio do problema *flowshop* permutacional distribuído, minimiza as funções objetivo propostas de forma satisfatória.

5 CONCLUSÕES

Os problemas de programação de tarefas em máquinas, pelo seu caráter experimental, não trazem soluções finais nas experimentações computacionais, porém são capazes de trazer soluções de boa qualidade e a busca por melhoramentos nas heurísticas e, consequentemente, das soluções, permitem uma grande variedade de estudos dentro do *scheduling*.

A implementação das heurísticas LPT, SPT, Triangular e Triangular Invertida, adaptadas para um problema *flowshop* permutacional distribuído através de suas variações LPT-A, SPT-A, Triangular-A e Triangular Invertida-A, foram comparadas para verificar seu desempenho em relação a duas funções objetivo definidas.

O objetivo deste trabalho foi de verificar, dentre heurísticas conhecidas na literatura, qual possui o melhor desempenho na minimização da duração total da programação e do tempo total de fluxo num sistema de produção clássico. Após a realização da experimentação computacional, para este estudo pode-se verificar que as heurísticas aplicadas não influenciaram significativamente nos valores para a duração total da programação. Porém, quando se tratou do tempo total de fluxo, notou-se a superioridade do método SPT, juntamente com sua variante SPT-A, na minimização da função objetivo. A proposta de distribuição alternada das tarefas entre as células de processamento, porém, não teve influência significativa nos valores para ambas as funções objetivo.

Por outro lado, observa-se que os aumentos da quantidade de células de processamento influenciam significativamente os valores das funções objetivo, apresentando na duração total da programação reduções de 38% a aumentar de 2 para 4 células, e 33% aumentando de 4 para 8 células; já no tempo total de fluxo, a redução foi de 67% aumentando de 2 para 4 células, e 64% amentando de 4 para 8 células.

Os resultados obtidos neste trabalho são importantes para estudos futuros, com a implementação de heurísticas alternativas às já estudadas, adaptando-as para o problema em questão e verificando seu desempenho para a otimização destas e outras funções objetivo.

REFERÊNCIAS BIBLIOGRÁFICAS

BAKER, Kenneth R.; TRIETSCH, Dan. **Principles of Sequencing and Scheduling.** Nova Jérsei: John Wiley & Sons, 2009. 509 p.

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

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

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

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

BOBELIN, Laurent; MARTINEAU, Patrick; HE, Haiwu. **Shortest Processing Time First and Hadoop**. In: 3rd IEEE International Conference on Cyber Security and Cloud Computing (CSCloud 2016), 2016.

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

BOIKO, Thays J. P; MOCCELLIN, João Vitor. **Métodos Heurísticos para a Programação da Produção em Flow Shop Permutacional com Tempos de Setup Separados dos Tempos de Processamento e Independentes da**

Sequência de Tarefas. In: XLII Simpósio Brasileiro de Pesquisa Operacional, 2010, Bento Gonçalves. **Anais...** 2010.

CARNEIRO, Felipe M. **Avaliação de Métodos Heurísticos para a Solução do Problema de Programação Flowshop com Tempos de Setup Assimétricos e Dependentes da Sequência**. 2010. 185 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

CHIAVENATO, Idalberto. **Iniciação à administração da produção.** São Paulo: Makron, Mcgraw Hill, 1991.

CORRÊA, Henrique L.; CORRÊA, Carlos A. **Administração de produção e de operações: manufaturas e serviços:** Uma abordagem estratégica. 2. ed. São Paulo: Atlas, 2013.

CRUZ, Fábio E. V.; BRANCO, Fábio J. C. **Análise de Métodos Para Solução do Problema de Programação de Operações Flow Shop**. In: XVI SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO, 2009, Bauru.

DONG, Xingye; HUANG, Houkuan; CHEN, Ping. **An improved NEH-based heuristic for the permutation flowshop problem**. Computers & Operations Research. Elsevier, p. 3962-3968. 2008.

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.

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa.** 4. ed. São Paulo: Atlas, 2002. 175 p.

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

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

HUANG, Kai; YU, Xianyu; ZHANG, Yulin**. Minimizing the makespan for scheduling problems with general deterioration effects**. Mathematical Problems in Engineering, 2013, 8p.

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

KALCZYNSKI, Pawel J.; KAMBUROWSKI, Jerzy. **On the NEH heuristic for minimizing the makespan in permutation flowshops**. The International Journal Of Management Science. Elsevier, p. 53-60. 2007.

MOCCELLIN, João Vitor; NAGANO, Marcelo S. **Flow Shop com máquinas paralelas genéricas**. In: XXXV Simpósio Brasileiro de Pesquisa Operacional, 2003, Natal. **Anais...** 2003.

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

PINEDO, Michael L. **Planning and Scheduling in Manufacturing.** Nova Iorque: Springer, 2005.

PINEDO, Michael L. **Scheduling:** Theory, Algorithms and Systems. 4. ed. Nova Iorque: Springer-Verlag, 2012. 676 p.

SAPKAL, Sagar U.; CHAKRAVORTY, Arindam. **A new heuristic for minimizing total completion time objective in permutation flow shop scheduling**. International Journal Of Advanced Manufacturing Technology, Londres: Springer-Verlag, v. 38, p.1327-1338, 2013.

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

SCARDOELLI, Lucas Y. **Novos métodos heurísticos para a programação de operações no-wait flow shop com critério de minimização do tempo total de fluxo.** 2006. 159 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2006.

SLACK, Nigel et al. **Administração da produção - Edição Compacta.** São Paulo: Atlas, 1999. 523 p.

SILVA, Augusto A. da. **Heurística Evolutiva para problemas de programação em no-wait flowshop com tempos de setup.** 2012. 120 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2012.

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

APÊNDICE A - Código das heurísticas LPT, SPT, Triangular, Triangular Invertida e variações

```
program TCC;

uses Windows, SysUtils;

type
   arrn = array[0..200] of integer;

var i,ii,j,jj,m,n,a,z,aa,ee,xx,mk,ft,fof,result: integer;
   ma,mb : array [0..200,0..200] of integer;
   s,t,w,w1,w2,w3,w4,w5,w6,w7,w8,w9 : array [0..200] of integer;
   arquivo, arquivo1 : text;

{------------------------------------------------------------------------------}

//Procedure Makespan Classico

procedure fo(ax:arrn;b:integer);
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;
    mk := mb[m,b];
    ft := 0;
    for ii := 1 to b do ft := ft + mb[m,ii];
    fof := ft;                                          // Para makespan, mk;
para flow time, ft
end;

{------------------------------------------------------------------------------}

begin
    assign(arquivo1,'C:\Users\eduar\Documents\UTFPR\2018-1\EP30A          -
Trabalho de Conclusão de Curso II\Program\Resultados\Results.txt');
```

```
    rewrite(arquivo1);
       for xx := 1 to 2000 do
          begin
             assign(arquivo,'C:\Users\eduar\Documents\UTFPR\2018-1\EP30A  -
Trabalho de Conclusão de Curso II\Program\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;

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

{--------------------------------------------------------------------------}

// LPT e SPT

{
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;        // Para LPT <
; Para 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;
}

{--------------------------------------------------------------------------}

// TRIANGULAR                                                    // para Triangular,
SPT; para Triangular Invertida, LPT

{
```

```
aa := n div 2;
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];                                    // Para SPT e LPT,
aberto; para triangular, bypass

{-------------------------------------------------------------------------------}

// SEQUENCIA SIMPLES

   // 2 celulas

{
z := n div 2;

for i := 1 to z do w2[i]:= w1[2*i-1];
for i := 1 to z do w3[i]:= w1[2*i];
}

   // 4 celulas

{
z := n div 4;

for i := 1 to z do

    begin
        w2[i] := w1[4*i-3];
        w3[i] := w1[4*i-2];
        w4[i] := w1[4*i-1];
        w5[i] := w1[4*i];
    end;
}

   // 8 celulas

{
z := n div 8;
```

```
for i := 1 to z do

    begin
        w2[i]:= w1[8*i-7];
        w3[i]:= w1[8*i-6];
        w4[i]:= w1[8*i-5];
        w5[i]:= w1[8*i-4];
        w6[i]:= w1[8*i-3];
        w7[i]:= w1[8*i-2];
        w8[i]:= w1[8*i-1];
        w9[i]:= w1[8*i];
    end;
}

{-------------------------------------------------------------------------------}

// SEQUENCIA ALTERNADA

   // 2 celulas

{
z := n div 2;

for i := 1 to z do
    if (i mod 2 = 0) then
       begin
           w2[i]:= w1[2*i];
           w3[i]:= w1[2*i-1];
       end
    else
       begin
           w2[i]:= w1[2*i-1];
           w3[i]:= w1[2*i];
       end;
}

   // 4 celulas

{
z := n div 4;
```

```
for i := 1 to z do
    if (i mod 2 = 0) then
       begin
           w2[i] := w1[4*i];
           w3[i] := w1[4*i-1];
           w4[i] := w1[4*i-2];
           w5[i] := w1[4*i-3];
       end
    else
       begin
           w2[i] := w1[4*i-3];
           w3[i] := w1[4*i-2];
           w4[i] := w1[4*i-1];
           w5[i] := w1[4*i];
       end;
}

   // 8 celulas

{
z := n div 8;

for i := 1 to z do
    if (i mod 2 = 0) then
       begin
           w2[i]:= w1[8*i];
           w3[i]:= w1[8*i-1];
           w4[i]:= w1[8*i-2];
           w5[i]:= w1[8*i-3];
           w6[i]:= w1[8*i-4];
           w7[i]:= w1[8*i-5];
           w8[i]:= w1[8*i-6];
           w9[i]:= w1[8*i-7];
       end
    else
       begin
           w2[i]:= w1[8*i-7];
           w3[i]:= w1[8*i-6];
           w4[i]:= w1[8*i-5];
           w5[i]:= w1[8*i-4];
```

```
            w6[i]:= w1[8*i-3];
            w7[i]:= w1[8*i-2];
            w8[i]:= w1[8*i-1];
            w9[i]:= w1[8*i];
        end;
}

{-------------------------------------------------------------------------}

   // FO para 2 celulas

{
for i := 1 to n do write(w1[i],' ');
    fo(w1,n);
    writeln;
for i := 1 to z do write(w2[i],' ');
    fo(w2,z);
    result := fof;
    writeln;
for i := 1 to z do write(w3[i],' ');
    fo(w3,z);
            if result <= fof then
            result := fof;
    writeln;
}

   // FO para 4 celulas

{
for i := 1 to n do write(w1[i],' ');
    fo(w1,n);
    writeln;
for i := 1 to z do write(w2[i],' ');
    fo(w2,z);
    result := fof;
    writeln;
for i := 1 to z do write(w3[i],' ');
    fo(w3,z);
            if result <= fof then
            result := fof;
    writeln;
```

```
for i := 1 to z do write(w4[i],' ');
    fo(w4,z);
            if result <= fof then
            result := fof;
    writeln;
for i := 1 to z do write(w5[i],' ');
    fo(w5,z);
            if result <= fof then
            result := fof;
    writeln;
}

   // FO para 8 celulas

{
for i := 1 to n do write(w1[i],' ');
    fo(w1,n);
    writeln;
for i := 1 to z do write(w2[i],' ');
    fo(w2,z);
    result := fof;
    writeln;
for i := 1 to z do write(w3[i],' ');
    fo(w3,z);
            if result <= fof then
            result := fof;
    writeln;
for i := 1 to z do write(w4[i],' ');
    fo(w4,z);
            if result <= fof then
            result := fof;
    writeln;
for i := 1 to z do write(w5[i],' ');
    fo(w5,z);
            if result <= fof then
            result := fof;
    writeln;
for i := 1 to z do write(w6[i],' ');
    fo(w6,z);
            if result <= fof then
            result := fof;
```

```
    writeln;
for i := 1 to z do write(w7[i],' ');
    fo(w7,z);
            if result <= fof then
            result := fof;
    writeln;
for i := 1 to z do write(w8[i],' ');
    fo(w8,z);
            if result <= fof then
            result := fof;
    writeln;
for i := 1 to z do write(w9[i],' ');
    fo(w9,z);
            if result <= fof then
            result := fof;
    writeln;
}

    //readln;
    close(arquivo);
    writeln(arquivo1,result);

end;
close(arquivo1);
end.
```