Centro Federal de Educação Tecnológica de Minas Gerais
Programa de mestrado em modelagem matemática e computacional

Modelo e Algoritmos para um problema de sequenciamento de ordens de manutenção preventiva

Joice Martinha Rodrigues

Orientador: Prof. Dr. Gustavo Campos Menezes
Centro Federal de Educação Tecnológica de Minas Gerais

Belo Horizonte
Outubro de 2023

Joice Martinha Rodrigues

Modelo e Algoritmos para um problema de sequenciamento de ordens de manutenção preventiva

Dissertação apresentado ao Programa de Pós-graduação em Modelagem Matemática e Computacional do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Área de concentração: Modelagem Matemática e Computacional.

Linha de pesquisa: Sistemas inteligentes.

Orientador: Prof. Dr. Gustavo Campos Menezes
Centro Federal de Educação Tecnológica de Minas Gerais

Belo Horizonte
Outubro de 2023

Rodrigues, Joice Martinha
R696m Modelo e algoritmos para um problema de sequenciamento de ordens de manutenção preventiva. / Joice Martinha Rodrigues. – – Belo Horizonte, 2023.
50 f. : il.

Dissertação (Mestrado) – Centro Federal de Educação Tecnológica de Minas Gerais, Programa de Pós-Graduação em Modelagem Matemática e Computacional, 2023.
Orientador: Prof. Dr. Gustavo Campos Menezes

Bibliografia

1. Modelos Matemáticos. 2. Algoritmos – Heurística. 3. Manutenção - Programação. I. Menezes, Gustavo Campos. II. Centro Federal de Educação Tecnológica de Minas Gerais. III. Título

CDD 511.8

Elaboração da ficha catalográfica pela Bibliotecária Elisângela Gonçalves Barbieri CRB-6: 2624 / CEFET-MG

Joice Martinha Rodrigues

Modelo e Algoritmos para um problema de sequenciamento de ordens de manutenção preventiva

Dissertação apresentado ao Programa de Pós-graduação em Modelagem Matemática e Computacional do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Prof. Dr. Gustavo Campos Menezes
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Nome do convidado 1
Instituição do convidado 1

Prof. Dr. Nome do convidado 2
Instituição do convidado 2

Belo Horizonte
Outubro de 2023

À minha mãe, que sempre me apoiou em todos os momentos da minha vida. Seu amor incondicional, suas palavras de encorajamento e seu exemplo de determinação e perseverança foram fundamentais para que eu pudesse chegar até aqui.

Agradecimentos

Agradeço, em primeiro lugar, a Deus e ao Espírito Santo, que me concederam força e sabedoria para superar os desafios e concluir este trabalho.

Agradeço também à minha família, que sempre esteve presente, me apoiando e incentivando em todas as etapas do meu percurso acadêmico e pessoal.

Ao meu orientador que não apenas compartilhou seu conhecimento e experiência, mas também incentivou-me a explorar novas ideias e a aprimorar minhas habilidades de pesquisa. Suas contribuições foram inestimáveis, e é uma honra ter tido a oportunidade de trabalhar sob sua orientação

Que todos esses agradecimentos sejam registrados como uma homenagem àqueles que, de alguma forma, colaboraram para que este trabalho pudesse ser concluído com êxito.

"E a luz resplandece nas trevas, e as trevas não a compreenderam" (João 1:5)

Resumo

Esta dissertação teve como objetivo principal desenvolver um modelo matemático capaz de abordar um problema real de sequenciamento de máquinas paralelas em um ambiente *Job Shop*, visando encontrar o menor *makespan* global do sistema. Além da formulação matemática em si, uma análise detalhada da eficácia dos algoritmos desenvolvidos também será realizada. Dois algoritmos foram elaborados: um utilizando uma heurística construtiva parcialmente gulosa e outro com base na técnica de *Simulated Annealing*.

Para avaliar a performance dos algoritmos, foram conduzidos experimentos utilizando conjuntos de instâncias diversificados. Em particular, dois testes foram realizados, variando a mão de obra disponível: um aumento de 2% e outro de 5%. Notavelmente, observou-se que à medida que o aumento da mão de obra crescia, as soluções obtidas tendiam a melhorar.

O primeiro algoritmo produziu resultados para todos os grupos de instâncias definidos. Contudo, as soluções obtidas ficaram significativamente distantes da solução ótima, indicando a complexidade intrínseca do problema. A partir dessas soluções iniciais, o segundo algoritmo foi aplicado, permitindo uma exploração mais profunda do espaço de soluções. Esse processo se mostrou mais eficaz em termos de convergência para soluções mais próximas do ótimo global. O segundo algoritmo, baseado no *Simulated Annealing*, foi capaz de explorar diversas configurações de alocação de tarefas, permitindo uma busca mais ampla por soluções promissoras.

Palavras-chave: Ordens de manutenção; Sequenciamento; *Simulated Annealing*.

Abstract

This dissertation aims to develop a mathematical model capable of addressing a real-world problem of parallel machine scheduling in a Job Shop environment, aiming to find the minimum global makespan of the system. In addition to the mathematical formulation itself, a detailed analysis of the effectiveness of the developed algorithms will also be conducted. Two algorithms were devised: one using a partially greedy constructive heuristic and another based on the Simulated Annealing technique.

To evaluate the performance of the algorithms, experiments were conducted using diversified sets of instances. In particular, two tests were performed, varying the available workforce: a 2% increase and a 5% increase. Notably, it was observed that as the increase in workforce grew, the obtained solutions tended to improve.

The first algorithm produced results for all defined instance groups. However, the obtained solutions were significantly distant from the optimal solution, indicating the intrinsic complexity of the problem. Building upon these initial solutions, the second algorithm was applied, allowing for a deeper exploration of the solution space. This process proved more effective in terms of converging towards solutions closer to the global optimum. The second algorithm, based on Simulated Annealing, was able to explore various task allocation configurations, enabling a broader search for promising solutions.

Keywords: Maintenance orders; Scheduling; Simulated Annealing.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

EDD	*Earliest Due Date*
ERP	*Enterprise Resources Planning*
FIFO	*First In, First Out*
FO	Função Objetivo
HPG	Heurística Parcialmente Gulosa
LIFO	*Last In, First Out*
LS	*Least Slack*
LS	Limite Superior
MFC	*Modified Choice Function*
PEPS	Primeiro que Entra Primeiro que Sai
PMSP	*Parallel Machine Scheduling Problem*
PSTEEM	Problema de Sequenciamento de Tarefas, Equipamentos e Equipes de Manutenção.
RL	Relaxação Linear
SA	*Simulated Annealing*
SAP	Sistemas, Aplicativos e Produtos para Processamento de Dados
SBPO	Simpósio Brasileiro de Pesquisa Operacional
SPT	*Shortest Processing Time*
TFR	Tempo de Folga Restante
UEPS	Último que Entra é o Primeiro que Sai

Lista de Símbolos

Δ	Letra grega Delta
α	Letra grega Alfa
$\in$	Símbolo de pertencimento
$\forall$	Para todo

Sumário

1 Introdução

1.1 Motivação

Com a crescente utilização de softwares e tecnologias da informação, eventuais problemas podem ser solucionados de forma a potencializar a busca de novas possibilidades e resultados para as companhias que necessitam de explorar oportunidades e de se manter nos mercados mundiais. Uma das maneiras de se tornar competitivo é criar soluções inteligentes e inovadoras que venham a agregar nos processos cotidianos. A inovação, segundo Tidd e Bessant (2015), nos últimos anos caminha para ser a peça chave da economia nacional.

Um outro fator que hoje as companhias precisam levar em consideração é a eficiência operacional dos equipamentos. Chiavenato (2002) define esta eficiência como uma medida racional e inteligente de utilização de recursos que a empresa possui, ou seja, para aumentar a performance dos indicadores é ideal que todas as entradas sejam parametrizadas e controladas. Uma forma de fazer a gestão e medir o rendimento das equipamentos é realizar as manutenções periódicas para evitar paradas não programadas.

As manutenções preventivas acarretam diversos benefícios para as organizações, Martins e Laugeni (2005) citam algumas dessas vantagens como: o aumento da vida útil dos equipamentos, a redução dos custos operacionais e de interrupções da produção e também a incentivação de uma mentalidade preventiva nas empresas. Por esse motivo é comum encontrar diversas aplicações de ferramentas que auxiliam e direcionam a criação de rotinas de manutenção capazes de administrar vários fatores que compõem a gestão da manutenção.

O avanço da tecnologia também contribui fortemente para encontrar soluções que condiz com os principais problemas que hoje as empresas enfrentam. Gonçalves (1994) cita os impactos que a informática gera nas organizações com transformações que abrangem as interações pessoais, o ritmo de trabalho e os *skills* dos operadores. Com essa tecnologia é possível, por exemplo, consolidar um grande número de informações de forma prática e rápida e trazer respostas e resultados que atendam as necessidades das organizações.

1.2 Definição do problema de pesquisa

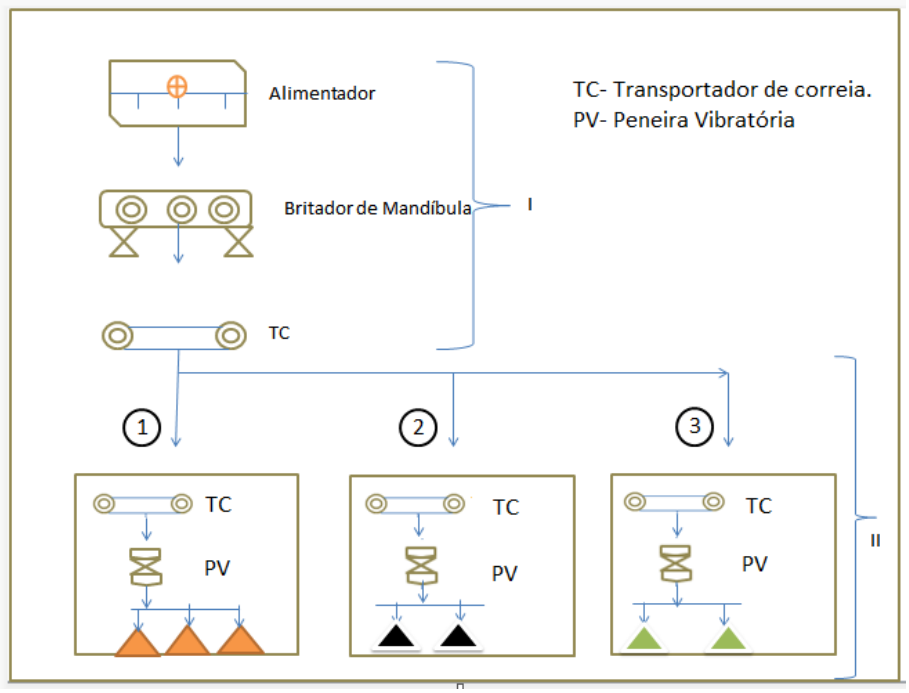
Este trabalho tem como objetivo abordar um desafio real enfrentado por uma pequena indústria do setor de mineração de gnaisse, em que é preciso realizar a correta programação das ordens de tarefas de manutenção, bem como a alocação eficiente de equipamentos e equipes, considerando as particularidades e restrições inerentes ao contexto industrial. Esse problema enquadra-se na classe de sequenciamento e é formalmente denominado como Problema de Sequenciamento de Tarefas, Equipamentos e Equipes de Manutenção (PSTEEM).

Nesta situação existe um conjunto de tarefas não preemptivas com tempos de processamentos conhecidos que devem ser sequenciadas nesses equipamentos, sendo que uma mesma tarefa pode ser executada em diferentes equipamentos e ao mesmo tempo. Existem também intervalos pré-determinados denominados neste trabalho como janelas iniciais e finais em que essas tarefas devem ser executadas. Uma outra característica que foi considerada é referente ao grau de prioridades em que essas máquinas são divididas, sendo que o grau de prioridade é a importância operacional que estas desempenham dentro da planta industrial, portanto aquelas definidas com o grau de prioridade I possuem janelas de tempo menores do que ao de grau II, pois estes são fundamentais para a produção uma vez que uma parada não programada resultaria em um prejuízo enorme tanto do ponto de vista produtivo quanto do financeiro.

Os materiais produzidos nesse processo são separados em 3 tipos de granulometria, sendo o estágio 1, relativo aos materiais classificados como grossos com granulometria entre 37,5 a 75 mm, o estágio 2 classificados como médios com granulometria entre 19 a 31,5 mm e o estágio 3 classificados como finos com granulometria entre 4,8 a 12,5 mm.

Na Figura 1 é possível verificar a vista da planta industrial da britagem dividida em graus de prioridades e nos tipos de granulometria dos materiais produzidos.

Figura 1 – Visão global da planta industrial



Fonte: Elaborado pelo autor

O conjunto de equipamentos que fazem parte do mesmo tipo de granulometria devem também possuir as mesmas janelas de tempo, uma vez que para a conclusão do processamento desses materiais é necessário o funcionamento de todos equipamentos do conjunto.

Com relação a mão de obra responsável para a execução dessas tarefas, essas são divididas entre as equipes de manutenção que possuem aptidão para a realização dessas tarefas. Ou seja,

para este problema além de garantir a alocação de todas as tarefas é preciso também determinar qual equipe disponível irá executar essas ordens de manutenção.

Para a realização das tarefas é necessário considerar que cada equipe possui um limite de horas disponíveis e existe uma configuração específica para execução dessas tarefas, que podem ser duas hipóteses, a primeira considerando que apenas uma equipe específica pode realizar uma dada tarefa e a segunda possibilidade é quando duas ou mais equipes têm competência para realizar a mesma tarefa.

A Tabela 1 ilustra uma configuração possível da rotina de manutenção levando em consideração os intervalos de tempo para a execução das tarefas de acordo com a importância operacional dos equipamentos.

Tabela 1 – Configuração Inicial: tabela que resume uma programação inicial modelo.

Grau	Granulometria	Equipamento	Janela Inicial (h)	Janela Final (h)
I	1	A	0	3
II	1	B	3	17
II	1	C	3	17
II	2	D	2	18
II	3	E	1	19

Para este mesmo exemplo existe duas equipes disponíveis que devem realizar essas tarefas. A disposição dessas tarefas é evidenciada na Tabela 2.

Tabela 2 – Configuração das equipes: tabela que resume uma programação das equipes do modelo.

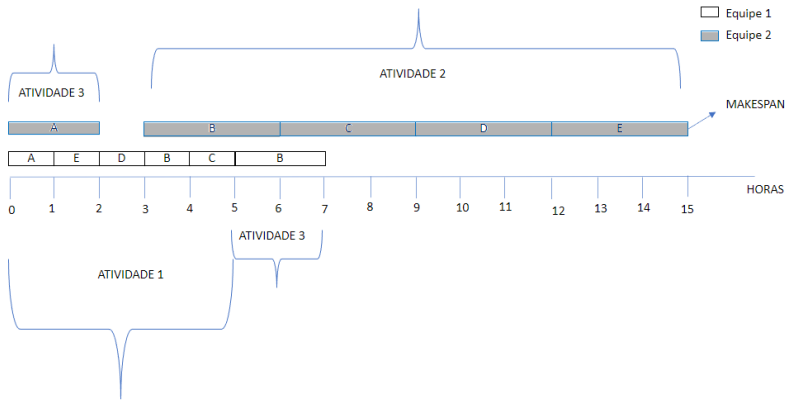
Tarefas	Processamento (h)	Equipamentos	Equipe
1	1	A,B,C,D e E	Equipe apta: 1
2	3	B,C,D,E	Equipe apta: 2
3	2	A,B	Equipe apta: 1 ou 2

A Figura 2 ilustra um possível arranjo para a estrutura acima em que são considerados todos os parâmetros do problema.

Nota-se que neste exemplo básico as janelas de tempo foram todas respeitadas e as equipes foram alocadas de acordo com a disponibilidade. É importante ressaltar que existem diversas maneiras de sequenciar essas tarefas, neste mesmo esboço se a ordenação fosse diferente, o tempo final do sistema poderia ser maior ou até mesmo menor. Na Figura 3 temos um outro exemplo utilizando os mesmos critérios de entrada da Figura 2, no entanto, o tempo final ficou maior apenas realocando a tarefa 3 para a equipe 2.

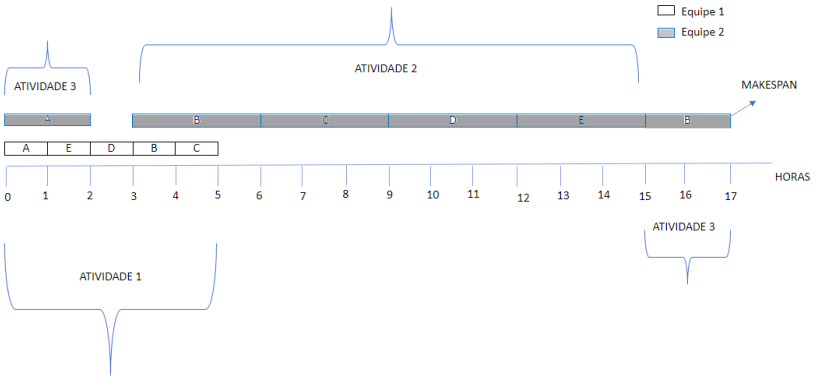
Este problema faz parte da classe de problemas de sequenciamento de máquinas paralelas (PSMP) com grande complexidade computacional. Uma vez que este problema possui vários fatores a serem considerados, é difícil criar um padrão que seja excelente de tal forma que aspectos como eficiência operacional, ociosidade de equipes, disponibilidade de ativos industriais e paradas não programadas sejam todos satisfeitos.

Figura 2 – Configuração Modelo I



Fonte: Elaborado pelo autor

Figura 3 – Configuração Modelo II



Fonte: Elaborado pelo autor

1.3 Objetivos

Nesta seção, serão explorados os objetivos gerais deste trabalho, que possuem um propósito amplo para orientar a pesquisa. Além disso, serão detalhados os objetivos específicos, nos quais serão apresentados os principais meios para alcançar esse propósito de maneira mais precisa e concreta.

1.3.1 Objetivo Geral

Os objetivos gerais desta dissertação são:

- Propor uma nova formulação matemática para o sequenciamento de tarefas em um cenário de equipamentos paralelos considerando como parâmetros as janelas iniciais e finais dos

equipamentos e a disponibilidade das equipes.

- Desenvolver algoritmos que sejam capazes de resolver o PSTEEM.

1.3.2 Objetivo Específico

Para alcançar os objetivos gerais deste trabalho são definidos os seguintes objetivos específicos:

- Medir a eficiência dos resultados obtidos pelo emprego da heurística parcialmente gulosa para a construção da solução e da meta-heurística *simulated annealing*.
- Identificar qual algoritmo apresenta melhores resultados para o cálculo do menor *makespan*.
- Teorizar as principais referências da literatura sobre os problemas de sequenciamento de tarefas e comparar os métodos heurísticos aplicados.

1.4 Organização do trabalho

A dissertação está dividida em capítulos, incluindo este. Os capítulos subsequentes são : Capítulo 2, que apresenta as principais referências e revisões encontradas na literatura acerca do tema proposto. O Capítulo 3, onde são apresentadas as contribuições que a pesquisa proporcionou, os resultados e as realizações alcançadas. Já o Capítulo 4 mostra a formulação matemática e os peseudocódigos dos algoritmos que foram desenvolvidos .

Os resultados estão evidenciados no Capítulo 5, onde ocorre a análise e as interpretações acerca dos experimentos e testes e, por fim, no Capítulo 6 são apresentadas as conclusões e as perspectivas sobre a dissertação.

2 Fundamentação Teórica

Neste capítulo serão abordados os principais assuntos que contribuíram para a construção teórica desta dissertação. A Seção 2.1 diz respeito aos principais tipos de sequenciamento de tarefas encontrados na literatura, categorizando seu fluxo de operação e seus estágios de produção. O segundo assunto, presente na Seção 2.2, não só faz uma conceituação sobre o *makespan* como também evidencia alguns estudos com aplicações reais de métodos que foram capazes de otimizar e reduzir o tempo final do sequenciamento do sistema.

Os próximos dois tópicos são referentes as heurísticas e meta-heurísticas empregadas neste trabalho. A Seção 2.3 faz uma breve explicação sobre as heurísticas construtivas gulosas enquanto a Seção 2.4 descreve sobre a meta-heurística abordada nesta dissertação que é o *simulated annealing*.

2.1 Tipos de sequenciamento de tarefas

Os problemas de sequenciamentos e programação de tarefas encontrados na literatura são classificados de acordo com especificações do fluxo das operações. Nagano, Moccellin e Lorena (2004) destacam os mais comuns, sendo eles:

- *Job Shop*: Existe um conjunto de atividades e um conjunto de máquinas e cada atividade possui uma sequência específica de processamento. Neste tipo de configuração existe um conjunto de tarefas que precisam ser escalonadas nas máquinas de modo a garantir um sequenciamento eficaz. É comum também encontrar algumas variações como por exemplo equipamentos que exigem intervalos de operação e tempos de *setup*. Um exemplo desta aplicação é evidenciado no artigo de Fuchigami, Moura e Branco (2017) que aborda uma programação *job shop* com dois tipos de tempos de *setups*, um incluído nos tempos de processamento das tarefas e outro antecipado;

- *Flow Shop*: Existe um conjunto de atividades e um conjunto de máquinas em que todas as atividades possuem a mesma sequência de processamento. Uma demostração que ilustra esta aplicação é o texto de Wang e Xia (2005) que utiliza uma abordagem heurística para resolver um problema *flow shop* com tempos de processamentos fixos, operações não preemptivas cujo objetivo é encontrar uma permutação que minimize o tempo de conclusão das tarefas;

- *Open Shop*: Existe um conjunto de atividades e um conjunto de máquinas sem uma definição formal de sequenciamento, ou seja, permite uma alocação flexível das tarefas sem seguir um sequenciamento prévio;

- *Flow Shop* Permutacional: Possui as mesmas características do *Flow Shop*, porém em cada máquina a sequência das atividades é a mesma;

- Máquina Única: Existe um conjunto de atividades que devem ser processadas por uma única máquina. Há situações em que existe um conjunto de equipamentos, mas estes por funcionar como uma só máquina se enquadram no sistema de máquina única. Neto (2010) mostra em sua tese uma formulação matemática voltada para minimizar o tempo de termino de um conjunto de tarefas em uma única máquina. Uma diferenciação deste modelo é que para este caso a tarefa pode ser alocada em uma máquina única ou uma máquina terceirizada, sendo que esta última implica em um custo adicional;

- Máquinas Paralelas: Existe um conjunto de atividades que devem ser processadas por mais de uma máquina. Diferente do sistema de máquina única, este considera mais de uma máquina no sistema, sendo assim a alocação de recursos deve ocorrer de forma a balancear o sequenciamento nos equipamentos. É comum encontrar diversas aplicações deste tipo de programação. Um estudo feito por Viegas (2016) exemplifica uma proposta de sequenciamento em máquinas paralelas em um banco de investimento em que a dificuldade maior que a empresa enfrentava era a má distribuição das solicitações para os analistas, que acarretava atrasos e insatisfação dos clientes;

- *Job Shop* com Máquinas Múltiplas: É um sistema *Job Shop* aplicado a um conjunto de máquinas paralelas. Um exemplo básico desta aplicação é o trabalho apresentado por Pereira (2019) que surge devido a uma necessidade encontrada nas empresas de manufatura com questões sustentáveis no que diz respeito a eficiência energética, no qual o autor se preocupa em encontrar métodos que sejam capazes de fornecer uma estimação de consumo eficaz. Neste mesmo contexto foram desenvolvidas diferentes técnicas de aprendizado de máquinas com o intuito de encontrar diferentes programações com diversas taxas de consumo de energia;

- Flow Shop com Máquinas Múltiplas: É um sistema *Flow Shop* aplicado a um conjunto de máquinas paralelas. O artigo apresentado por Moccellin e Nagano (2003) propõe um procedimento heurístico para lidar com o dilema de programação Flow Shop, que envolve máquinas paralelas e gargalos na produção, com o objetivo de minimizar a extensão completa do programa. O autor sugere duas fases para a resolução deste problema: Na primeira, o problema original é convertido em um Flow Shop imaginário (sem estágios críticos), cuja solução é viabilizada na segunda fase através de uma reorganização das tarefas no estágio crítico.

As ordens do sequenciamento podem variar de acordo com a regra de prioridade de cada sistema produtivo. Tubino (2000), Dobos, Tamás e Illés (2016) e Lustosa, Mesquita e Oliveira (2008) destacam alguma delas

- FIFO (*First In, First Out*): Também conhecido como PEPS - primeiro que entra é o primeiro que sai. Nesse tipo de sequenciamento, a prioridade da sequência será sempre dada à primeira atividade que chega no sistema para ser processada, ou seja, a atividade que chega primeiro também é a primeira a ser processada. Um exemplo clássico dessa regra são as montadoras de automóveis, que trabalham com a produção dos carros de acordo com a ordem de chegada dos pedidos.

- LIFO (*Last In, First Out*): Nesse tipo de programação a última tarefa a entrar no sistema será a primeira a ser processada. Nessa estratégia o UEPS, nomenclatura também encontrada na literatura (último que entra é o primeiro que sai) é indicado em situações em que a rotatividade e a urgência sejam ocorrências comuns e que a maior prioridade seja a entrega.

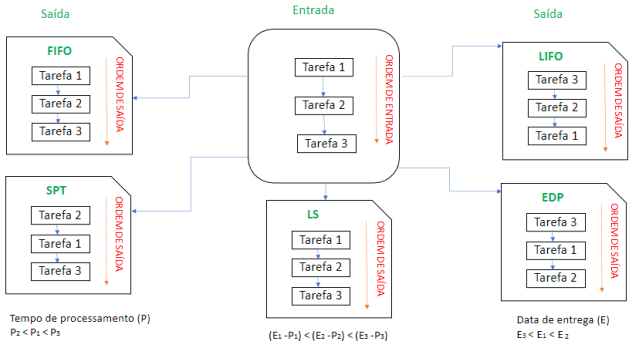
- EDD (*Earliest Due Date*): O sequenciamento é realizado sempre com o intuito de evitar atrasos, por isso a principal prioridade deste sequenciamento são as tarefas que possuem a data de entrega mais próxima de expirar.

- SPT (*Shortest Processing Time*): Nesta regra a prioridade do sequenciamento sempre será as tarefas que apresentam tempos de processamento menores, ou seja, atividades que demandam tempo maiores são sempre alocadas para as últimas posições da sequência. Com essa aplicação a principal vantagem é diminuir a ociosidade das tarefas subsequentes, mas é importante também garantir a execução das tarefas de longos prazos, por isso são sugeridas também outras regras complementares para assegurar a dinâmica da operação.

- LS (*Least Slack*): Diferente do EDD a programação que tem como referência o *least slack* tem como prioridade a tarefa que apresenta a menor folga (diferença entre o processamento total das atividades e a data de término prevista). A maior vantagem desta técnica é garantir a efetividade na entrega.

A Figura 4 ilustra os principais tipos de estratégias utilizadas para o sequenciamento de tarefas, sendo E_j e P_j data de entrega e tempo de processamento da tarefa j.

Figura 4 – Estratégia de sequenciamento



Fonte: Elaborado pelo autor

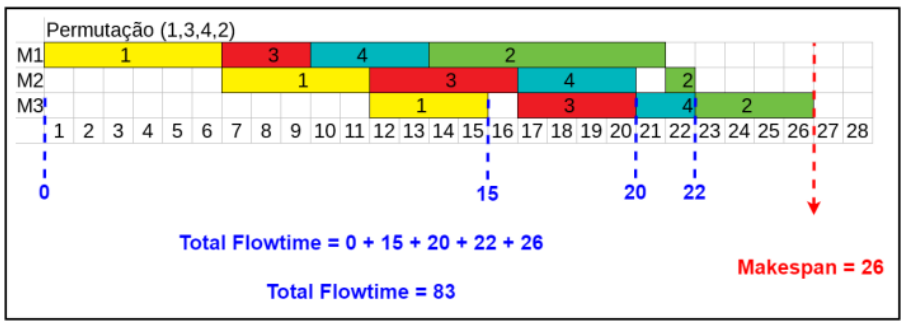
2.2 *Makespan*

No que se refere aos tempos de processamentos que compõem as técnicas de sequenciamento, um termo bastante comum encontrado na literatura é o *makespan* que também é conhecido como C_{max}. Pinedo (2008) define o *makespan* como o tempo de conclusão da última tarefa a ser processada no sistema, neste caso considera-se por exemplo um conjunto de atividades $I = \{1,2,3....,n\}$ com n tarefas que serão sequenciadas, sendo que cada tarefa possui um instante de término representado por C . Ou seja o C_{max} será igual ao maior tempo de conclusão das tarefas sequenciadas, conforme Equação (1):

$$C_{max} = max\,(C_1, C_2, C_3....C_n) \tag{1}$$

Na Figura 5 é possível observar um exemplo representativo de Júnior, Almeida e Venske (2020) em que é realizado o sequenciamento de um conjunto de atividades em três máquinas paralelas em um ambiente *flow shop* permutacional.

Figura 5 – Representação do cálculo do *makespan*



Fonte: Júnior, Almeida e Venske (2020)

Nesta ilustração a última tarefa a ser alocada no sistema termina no tempo 26, sendo este o *makespan* deste problema. A nível de eficiência na produção é essencial aplicar técnicas que sejam capazes de construir uma ordenação que entregue um *makespan* menor.

Encontra-se diversos exemplos reais na literatura em que se busca encontrar soluções preferíveis de sequenciamento com o intuito de minimizar o *makespan*. O trabalho de Flizicoski (2017) por exemplo busca reduzir os atrasos totais da programação de produção em um sistema *flow shop* composto por uma única máquina, sendo esta responsável pelo processamento de várias tarefas. Esta máquina pode apenas realizar um processo por vez, suas tarefas são não preemptivas e o tempo de *setup* é levado em consideração.

O trabalho de Melo (2010) já é voltado para a programação de produção naval com o principal objetivo de minimizar o tempo total de fabricação (*makespan*) das submontagens estruturais em oficinas de estaleiros de construção naval. Além de garantir o tempo mínimo de montagem, fatores como recursos, prazos limites de tarefas e tempos de processos também devem ser respeitados.

2.3 Algoritmos Gulosos

As heurísticas ou algoritmos gulosos são procedimentos que geram soluções com base em alguma decisão estratégica em que a seleção ocorre por algum fator específico. Esses fatores irão alternar de acordo com o objetivo em que se pretende alcançar, temos como exemplos de estratégias: arestas que possuem peso menor, vértices com graus maiores ou até mesmo tarefas com tempos de processamentos maiores ou menores (GONZALEZ, 2007).

Na Figura 6 é apresentado o pseudocódigo do algoritmo de seleção gulosa.

Figura 6 – Algoritmo Guloso

```
1  função guloso(C: conjunto) : conjunto
2     {C é o conjunto de todos os candidatos}
3     S := {} {Conjunto que constitui a solução, inicialmente vazio}
4     Enquanto S não é uma solução e C não vazio
5        x := Melhor elemento de C
6        C := C - {x}
7        Se x é viável então S := S U {x}
8     Se S é uma solução
9        retornar S
10    Senão
11       retornar Não existe uma solução
```

Fonte: Rocha, Alvarenga e Silva (2006)

Percebe-se que existe um conjunto que abrange todos os candidatos, mas que o critério de escolha é regido por uma função, ou seja, a cada seleção de um novo elemento ocorre a avaliação dos elementos candidatos e identificação do melhor elemento encontrado e em caso positivo o elemento é adicionado na solução corrente.

Uma grande vantagem dos algoritmos gulosos é que estes tentam encontrar soluções com base nas informações e padrões estabelecidos. É comum encontrar este tipo de heurística em situações em que é preciso construir soluções que sejam viáveis para posteriormente serem refinadas com outros métodos mais apropriados. O trabalho de Rizzo e Urrutia (2011), por exemplo, tem como objetivo realizar um sequenciamento de carros sem violar nenhuma limitação das estações nas quais os carros eram alocados.

A heurística escolhida pelo autor foi o *GRASP* que precisou gerar uma solução inicial a cada iteração. Devido a essa necessidade foi empregada uma heurística gulosa randômica que elabora uma ordem de veículos, e acrescenta a cada iteração um veículo ainda não incluído na última posição da ordem parcial que está sendo construída. O critério que diz qual veículo adicionar em cada passo, leva em conta o número de violações máximas que a inclusão do veículo gera na ordem em construção.

Um outro exemplo de aplicação é evidenciado no trabalho de Siqueira, Souza e Souza (2012) que busca encontrar o menor *makespan* de um problema de sequenciamento em um ambiente *flow shop* híbrido e flexível. Para atingir este objetivo é empregado um método guloso para gerar soluções iniciais e um procedimento de busca local.

A heurística gulosa para esta condição procura as duas tarefa que possuem os maiores tempos médios de processamento e aloca na máquina que consegue executar essas tarefas mais cedo evitando assim conflitos futuros como por exemplo atrasos na execução. Após todas as atividades serem alocadas é executado um outro algoritmo de busca local para refinar a solução.

Em geral, as soluções dos algoritmos gulosos não conseguem alcançar os ótimos globais, contudo devido a facilidade de implementação é uma boa opção para criar uma solução inicial viável e até mesmo para encontrar ótimos locais.

2.4 *Simulated Annealing*

A técnica *simulated annealing* faz referência aos princípios da termodinâmica nos processos de aquecimento de metal em que ocorrem as técnicas de aquecimento e resfriamento do sistema. Essa heurística foi desenvolvida por Kirkpatrick, Jr e Vecchi (1983) e ela se baseia no algoritmo de Metropolis et al. (1953) onde existe uma função custo que deve ser otimizada.

A funcionalidade deste algoritmo se baseia no controle da temperatura efetiva. O processo do *simulated annealing* inicia com uma alta temperatura efetiva do sistema a ser otimizado, que a cada estágio é lentamente resfriada. Dentro desses estágios são geradas várias configurações com diversos números de rearranjos a fim de obter soluções melhores.

Devido as variadas possibilidades de rearranjos, uma vantagem na utilização desta heurística é justamente evitar ótimos locais, pois ela consegue alcançar vizinhanças mais distantes da solução atual. No processo de geração de vizinho, o valor da função objetivo é sempre avaliado, sendo denominado Δ. É calculado subtraindo a função objetivo da solução vizinha $f(s')$, pela função objetivo da solução corrente $f(s)$, conforme expresso pela Equação (2) em um problema de minimizaçao.

$$\Delta = f(s') - f(s), \tag{2}$$

Em que:

- $\Delta < 0$ Quando a solução atual é melhor que a anterior, nesse caso ocorreu uma redução de energia.
- $\Delta > 0$ A solução atual é pior que a anterior, nesta situação deve ser calculada a probabilidade de se aceitar uma outra solução. Esta probabilidade é expressa pela equação de Boltzmann (2012). O Parâmetro T se refere a temperatura do sistema responsável por regular soluções com pior custo, como indicado na Equação (3).

$$e^{(-\Delta/T)} \tag{3}$$

No Algoritmo 1 temos o pseudocódigo básico do *simulated annealing* desenvolvido por Aguiar, Mauri e Silva (2018).

Algoritmo 1: Algoritmo base do simulated Annealing

```
1  Entrada: α, SAmax, T0, Tc e S
2  Saída: O melhor valor da função objetivo f(S*)
3  S* ← S
4  IterT ← 0
5  T ← T0
6  while T > Tc do
7      while IterT < SAmax do
8          IterT ← IterT + 1
9          gerar um vizinho S' ∈ N(S)
10         Δ ← f(S') − f(S)
11         if Δ < 0 then
12             S ← S'
13             if f(S') < f(S*) then
14                 S* ← S'
15         else
16             tomar x ∈ [0,1]
17             if x < e^(−Δ/T) then
18                 S ← S'
19     T ← α · T
20     IterT ← 0
```

Os parâmetros de entrada são: α que se refere a taxa de resfriamento, o número de iterações representado por $SAmax$, o T_0, T_C e S, sendo, respectivamente, a temperatura inicial do sistema, a temperatura de congelamento e a solução inicial.

O Algoritmo começa com as variáveis sendo inicializadas (linhas 1 a 5). O primeiro laço de repetição (linha 6) ocorre para garantir que enquanto a temperatura T for maior que a temperatura de congelamento T_C a iteração irá acontecer. O segundo laço de repetição (linha 7) garante a geração de uma solução vizinha enquanto a contabilização das iterações $IterT$ for menor que o número de iteração estabelecido inicialmente na variável $SAmax$.

Uma vez que é escolhido um vizinho S' dentro do espaço de solução $N(S)$, ocorre o cálculo do Δ para verificar se houve uma melhora da função objetivo do problema. Se o Δ tiver um valor negativo em uma problema de minimização, por exemplo, é um indicativo de melhora da solução, então nessa situação a nova solução é atualizada e comparada com as demais soluções para verificar se esta vizinhança é a melhor encontrada dentro da busca que foi realizada, e assim, é atualizada como a melhor solução do sistema (linha 9 a 14). Nas situações em que ocorreu uma piora da solução com o Δ positivo é possível que a solução seja aceita desde que esta atenda a seguinte condição: o número aleatório gerado entre o intervalo de 0 a 1 deve ser menor que o fator de Boltzmann (2012): $x < e^{(-\Delta/T)}$, senão o processo de busca é encerrado (linha 15 a 18).

A temperatura T em cada ciclo é atualizada (linha 14) e assim que ela alcança a temperatura de congelamento T_C o algoritmo termina.

3 Trabalhos Relacionados

Nesta seção serão apresentados os trabalhos que se assemelham com o problema abordado nesta dissertação, assim como também as principais heurísticas e meta-heurísticas que são comumente empregadas em situações de programação de atividades e problemas de ordenação de tarefas.

Na Seção 3.1 são apresentados alguns problemas similares de sequenciamento de tarefas como a dissertação de Aquino (2018) que utiliza uma abordagem heurística para resolver um problema de planejamento de ordens de manutenção de longo prazo e o trabalho de Sant'anna (2022) que apresenta um problema de programação de manutenções em linhas de transmissão.

Também encontram-se os trabalhos de Barbosa (2022) que apresenta uma programação de tarefas que trabalha sobre previsões de demanda, Dahinten (2014) como uma estratégia de sequenciamento fundamentada em uma priorização do tempo de folga restante e Silva (2018) que com uma matriz de elegibilidade realiza um sequenciamento para minimizar o tempo de *setup* em uma indústria química.

A Seção 3.2 apresenta aplicações de algoritmos a problemas de sequenciamento e quais as principais estratégias usadas para aproximar os valores dos ótimos globais. Os métodos heurísticos apresentados nesta seção são: algoritmos colônia de formiga, algoritmos genéticos e a seper-heurística *Modified Choice Function* do trabalho de Andrade, Tanaka e Tanaka (2023), as heurísticas construtivas de Vieira e Taglialenha (2022) e por fim o *simulated annealing* apresentada na dissertação Vivan (2010).

3.1 Problemas de sequenciamentos de tarefas

O problema abordado por Aquino (2018) tem o intuito de minimizar a quantidade de mão de obra necessária para a realização das ordens de manutenção. Nessa situação existe um conjunto de atividades de manutenção e essas são executadas por um conjunto de equipes disponíveis de trabalho.

Em seu modelo matemático, existe uma penalidade que é aplicada a não execução dessas tarefas, ou seja, além de construir um planejamento de execução a longo prazo das ordens de manutenção é essencial também atingir um número de cumprimento satisfatório. Cada manutenção deve igualmente respeitar um intervalo de tempo para ser processada denominadas no trabalho como janelas de tempo iniciais e finais e não podem ultrapassar também o limite disponível de horas que cada equipe possui.

A dissertação de Sant'anna (2022) tem alguns fundamentos similares, porém diferentes do estudo anterior, o problema é limitado em desenvolver uma solução capaz de otimizar a programação das manutenções das linhas de transmissão energizadas e desenergizadas priorizando os atendimentos mais críticos.

O modelo matemático possui também restrição de disponibilidade de mão de obra, impossibilitando assim que a quantidade de equipes seja maior que a quantidade disponível para aquele período. As manutenções possuem janelas finais, então não é permitido que estas manutenções seja executada em atraso e existe uma quantidade mínima de linhas de transmissão que uma equipe pode trabalhar por dia.

Há também situações que visam atender a mais de um objetivo, estes são chamados de multiobjetivo. No trabalho de Barbosa (2022) temos um problema *Flow Shop* permutacional aplicado dentro da industria que consiste em executar tarefas em uma determinada ordem em todas as máquinas, buscando otimizar dois ou mais objetivos - como tempo de execução, atrasos, antecipações ou atrasos por máquina.

Nesse tipo de estratégia, o sequenciamento precisa considerar um agendamento de tarefas que tenha um *makespan* mínimo e ao mesmo tempo tenta evitar sempre que possível atrasos ou antecipação de tarefas.

Nos problemas de sequenciamento de tarefas nas indústrias é comum também encontrar programação de tarefas que seja orientada para uma produção sob demanda, ou seja, a cadeia produtiva só inicializa após uma ordem de produção ser gerada. No artigo de Dahinten (2014) a situação acontece em uma fábrica de elevadores, cuja a produção é condicionada a uma encomenda personalizada de acordo com a necessidade do cliente. Neste contexto de ambiente de produção do tipo *Job Shop*, cada peça tem sua própria programação para ser processada em diferentes máquinas, sendo que estas são detalhadas nas chamadas Ordens de Fabricação (OF) que são emitidas pelo Departamento de Planejamento e Controle de Produção (DPCP) da fábrica e repassadas para a equipe de produção. Esta indústria ainda tem um conjunto de de máquinas que processam essas peças e que posteriormente são direcionadas para a montagem final dos componentes do elevador.

A estratégia usada para este sequenciamento é conhecida como Tempo de Folga Restante (TFR), ela consiste basicamente em priorizar os itens que possui o menor TFR para a alocação desses nas máquinas disponíveis. O cálculo do TFR é evidenciado na Equação (4), onde d_j representa a data que o item j deve estar disponível na montagem, ppd_j o número de processos pós dobra do item j e t o dia atual.

$$TFR = d_j - ppd_j - t \tag{4}$$

Após definida a estratégia, a alocação irá acontecer primeiro considerando a capacidade da máquina. Ou seja, será dada preferência à máquina cuja soma do tempo de processamento for menor dentre as opções disponíveis para processar o item.

Uma outra opção de programação de tarefas de produção é quando para um produto há uma lista de máquinas que são elegíveis para que este produto seja processado. Nesta ocasião, o sequenciamento ocorre sempre em concordância com uma matriz de elegibilidade. Para exemplificar esta condição, Silva (2018) aplica este método em uma indústria química que tem como principal objetivo minimizar a soma dos tempos de *setup*.

Na Figura 7 é possível verificar um exemplo de configuração de elegibilidade de cada produto.

Figura 7 – Matriz de elegibilidade dos produtos

Produto	Máquina	Tempo de processamento (horas)	Produto	Máquina	Tempo de processamento (horas)
A	M1, M2, M3	38,03	N	M1, M2, M3	42,28
B	M1, M2, M3	33,17	O	M1, M2, M3	40,93
C	M1, M2, M3	36,00	P	M1, M2, M3	36,67
D	M1	20,32	Q	M1, M2, M3	42,42
E	M1, M2, M3	32,83	R	M1, M2, M3	40,37
F	M1, M2, M3	34,50	S	M1, M2, M3	46,43
G	M1	30,04	T	M1, M2, M3	23,93
H	M1, M2, M3	34,85	U	M1, M2, M3	96,67
I	M1, M2, M3	32,28	V	M2	73,00
J	M1, M2, M3	28,77	W	M2	38,17
K	M1, M2, M3	46,17	X	M2	49,25
L	M1, M2, M3	45,90	Y	M1, M2, M3	40,65
M	M1, M2, M3	38,49			

Fonte: Silva (2018)

Como é possível verificar, o produto A por exemplo deve ser alocado em uma máquina que seja $M1$, $M2$ ou $M3$. Há também situações como os produtos D, G, V, W, X que só podem ser alocados em apenas uma máquina específica, e nesse tipo de situação essas máquinas tem como prioridade receber esses produtos, uma vez que, para os demais existe uma arbitragem na escolha.

3.2 Métodos Heurísticos para problemas de sequenciamento

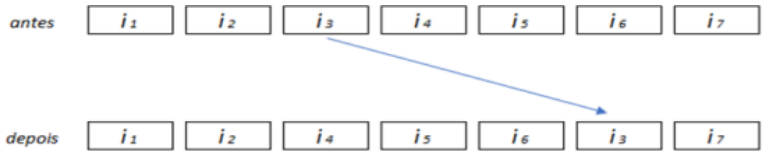
Os problemas de *scheduling*, em sua maioria, são complexos e apresentam dificuldades em encontrar soluções ótimas, principalmente por apresentar várias particularidades (NAPIERALA, 2000) . Por essa razão, são utilizadas abordagens aproximadas que visam encontrar soluções viáveis, porém não necessariamente ótimas, de maneira mais eficiente em termos de tempo e recursos computacionais. Assim como nesta dissertação, a utilização de meta-heurísticas são bons instrumentos para encontrar boas as soluções em um tempo computacional viável. Andrade, Tanaka e Tanaka (2023) apresentam em seu artigo além dos algoritmos colônia de formiga e algoritmos genéticos, uma super-heurística denominada *Modified Choice Function* (MFC) que destaca-se por sua vantagem em operar com bom desempenho em várias configurações, em contraste com outras heurísticas/meta-heurísticas.

Nesse contexto o Algoritmo Genético fornece ao MFC um resultado melhor em comparação com o antes e depois da aplicação da heurística selecionada pelo próprio MFC, permitindo avaliar o desempenho das heurísticas escolhidas.

Uma outra maneira muito comum de resolução de problemas de ordenação de atividades ou programação de processos, é utilizar heurísticas construtivas para em seguida realizar trocas e realocações na estrutura de vizinhança. Vieira e Taglialenha (2022) apresentam um problema de programação de robôs móveis em uma linha de manufatura, em que o ponto de partida

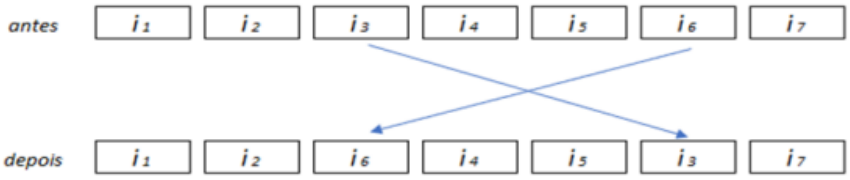
foi uma heurística construtiva capaz de fornecer dados iniciais, como o horário de liberação de uma subtarefa e o prazo limite da execução desta mesma subtarefa. Após a avaliação da função objetivo ocorre uma busca na estrutura de vizinhança com 4 pertubações apresentadas nas Figura 8, Figura 9 e Figura 10.

Figura 8 – Inserção Interna



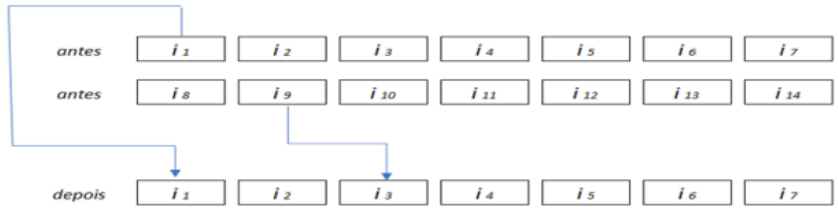
Fonte: Vieira e Taglialenha (2022)

Figura 9 – Troca Interna



Fonte: Vieira e Taglialenha (2022)

Figura 10 – Inserção Externa



Fonte: Vieira e Taglialenha (2022)

A cada pertubação é realizado um movimento de troca ou realocação em que é sempre avaliado se houve uma melhora ou piora da solução.

Encontra-se também a aplicação da meta-heurítica *simulated annealing* em problemas de sequenciamento da produção. A dissertação de Vivan (2010), por exemplo, emprega esta meta-heurística em um sistema de produção de fabricação de seringas médicas, cujo objetivo é minimizar o tempo de execução total de todas as tarefas.

A primeira etapa do método foi a geração dos dados em que as variáveis de entrada foram definidas de maneira aleatória, sendo geradas principalmente por meio de uma distribuição uniforme, enquanto que em alguns casos foram geradas por meio de uma distribuição normal. Os parâmetros de entrada foram moldados de acordo com a situação real, entretanto os limites do modelo assumiram suposições mais amplas.

Em comparação com o resultado do solver, o *simulated annealing* apresentou resultados satisfatórios para este problema, no entanto o tempo computacional para alguns testes consumiram um esforço maior do que do próprio método exato.

4 Abordagem da Solução

Neste capítulo, serão apresentados os procedimentos adotados para alcançar os objetivos propostos. A formulação matemática utilizada para modelar o problema é discutida na Seção 4.1. Em seguida, na Seção 4.2, é descrita a construção de uma solução inicial que permite explorar novas possibilidades de soluções. Por fim, a Seção 4.3 aborda a aplicação da meta-heurística *Simulated Annealing* para o modelo proposto.

4.1 Modelo Matemático

Foi proposto um modelo matemático, sendo este uma adaptação da formulação matemática desenvolvida por Aquino (2018) que é capaz de descrever a realidade desta indústria a fim de solucionar e encontrar soluções viáveis.

Os conjuntos, os parâmetros e as variáveis de decisão estão descritos a seguir:

Este problema é formado por 3 conjuntos, considera-se $I = \{1, 2, 3, ..., n\}$ o conjunto de n tarefas de manutenção que devem ser processadas em um conjunto $A = \{1, 2, 3, ..., m\}$ de m equipamentos por um conjunto $K = \{1, 2, 3, ..., p\}$ com p equipes de manutenção.

Para realizar essas ordens de manutenções é necessário observar critérios como os intervalos das janelas de tempo dos equipamentos $[e_a, l_a]$ com $a \in A$ e o instante de término da disponibilidade da equipe de manutenção h_k em que $k \in K$.

Para cada tarefa i de manutenção realizada no equipamento a é atribuído um tempo de processamento predefinido denominado p_{ia}. Para cada tarefa i existem uma ou mais equipes k que são qualificadas para executar a manutenção, conhecidas como u_{ik}. O valor de u_{ik} igual a 1 indica que a equipe é capaz de realizar a manutenção, enquanto o valor 0 indica incompatibilidade da equipe com a tarefa.

As variáveis e os parâmetros do modelo são esclarecidos na Tabela 3 e Tabela 4.

Tabela 3 – Variáveis do modelo.

Variáveis	Descrição
y_{iak}	1, se a tarefa i é executada pela equipe k no equipamento a; 0, caso contrário.
t_{ia}^{k}	Tempo de término da tarefa i no equipamento a executada pela equipe k.
C_{max}	Tempo de conclusão do processamento das tarefas.

Tabela 4 – Parâmetros do modelo.

Parâmetros	Descrição
e_a	Instante de início da janela de tempo do equipamento a.
l_a	Instante de término da janela de tempo do equipamento a.
h_k	Instante de término da disponibilidade da equipe de manutenção k.
p_{ia}	Tempo de processamento da tarefa i no equipamento a.
u_{ik}	Matriz de configuração de tarefas por equipes, sendo 1 se a tarefa i pode ser alocada na equipe k e 0, caso contrário.

A formulação matemática para o PSTTEM é apresentado abaixo:

$$Min\, C_{max} \tag{5}$$

S.a

$$\sum_{k=1}^{p} y_{iak} \cdot u_{ik} = 1, \forall i \in I, a \in A \tag{6}$$

$$t_{ia}^{k} \leq C_{max}, \forall i \in I, a \in A, k \in K \tag{7}$$

$$t_{ia}^{k} \geq (e_a + p_{ia}) \cdot y_{iak}, \forall i \in I, a \in A, k \in K \tag{8}$$

$$t_{ia}^{k} \leq l_a \cdot y_{iak}, \forall i \in I, a \in A, k \in K \tag{9}$$

$$t_{ia}^{k} \geq ti_{(a-1)}^{k} + y_{iak} \cdot p_{ia}, \forall i \in I, a \neq 0, k \in K \tag{10}$$

$$t_{ia}^{k} \geq t_{(i-1)a^k} + y_{iak} \cdot p_{ia}\, \forall i \neq 0\, a \in A, k \in K \tag{11}$$

$$t_{ia}^{k} \geq t_{(i+1)(a-1)}^{k} + y_{ik} \cdot p_{ia}, \forall i \in, a \neq 0, k \in K \tag{12}$$

$$t_{(i-1)a^k} \geq t_{i(a-1)}^{k} + y_{(i-1)k} \cdot p_{(i-1)a}, \forall i \neq 0, a \neq 0, k \in K \tag{13}$$

$$t_{ia}^{k} \leq h_k, \forall i \in I, a \in A, k \in K \tag{14}$$

$$y_{iak} \in \{0,1\}, \forall i \in I, k \in K \tag{15}$$

$$t_{ia^k \geq 0}, \forall i \in I, a \in A, k \in K \tag{16}$$

A função objetivo (5) procura reduzir o *makespan* das tarefas de manutenção considerando sempre a alocação de todas essas tarefas nos equipamentos. A equação (6) assegura que apenas uma equipe de manutenção será alocada na tarefa. Com o intuito de evitar que o tempo de manutenção seja superior ao tempo de processamento total das tarefas (*makespan*) é utilizada a equação (7). As equações (8) e (9) obrigam que as tarefas sejam executadas dentro das janelas de tempo dos equipamentos.

O sequenciamento das ordens de manutenção ocorre nas equações (10), (11), (12) e (13). Sendo a equação (10) impõe que o tempo de término da tarefa i no equipamento a executado pela equipe k deve ser maior ou igual ao tempo de término do equipamento anterior ($i_{(a-1)}$) no mesmo equipamento a e pela mesma equipe k, somado ao tempo de processamento da tarefa i no equipamento a. A equação (11) relaciona o tempo de término da tarefa i no equipamento a executado pela equipe k deve ser maior ou igual ao tempo de término da tarefa anterior $(i-1)$ no mesmo equipamento a e pela mesma equipe k. A equação (12) impõe que o tempo de término da tarefa i no equipamento a executado pela equipe k deve ser maior ou igual ao tempo de término da tarefa seguinte $(i+1)$ no equipamento anterior $(a-1)$ e pela mesma equipe k. Por fim a equação (13) garante que o tempo de término da tarefa anterior $(i-1)$ no equipamento a executado pela equipe k deve ser maior ou igual ao tempo de término da tarefa i no equipamento anterior $(a-1)$ e pela mesma equipe k.

A equação (14) garante que nenhuma manutenção ultrapasse a janela final da disponibilidade da equipe de manutenção.

Os domínios das variáveis são referenciados nas equações (15) e16).

As principais adaptações que foram realizadas comparadas com o modelo de Aquino (2018) são: A ausência de uma variável binária específica para indicar a precedência das tarefas em um mesmo equipamento com equipes diferentes, sendo que isso ocorre porque o modelo proposto adotou quatro restrições (10), (11), (12) e (13) para realizar esse sequenciamento. No modelo original existe também janelas de tempo para as tarefas de manutenções, já na formulação proposta nesta dissertação são os equipamentos que possuem intervalos de tempo para que as tarefas aconteçam.

Uma outra diferença entre os modelos são em relação as equações tanto da função objetivo quanto das restrições. Enquanto a equação (5) tenta reduzir o *makespan*(c_{max}), no modelo original a função objetivo tem a finalidade de minimizar a quantidade de equipe disponível. Um outro ponto importante é que no modelo apresentado neste trabalho existe uma obrigatoriedade de execução de todas as tarefas manutenção, enquanto no trabalho de Aquino (2018) algumas tarefas podem não ser selecionadas (gerando uma penalidade pela não realização).

4.2 Heurística Parcialmente Gulosa

Com o intuito de desenvolver uma solução viável para este modelo em que atendesse a todos os parâmetros, foi desenvolvida uma heurística parcialmente gulosa (HPG) capaz de construir a solução inicial para cada conjunto de instância.

No Algoritmo 2 temos a representação desta heurística.

Algoritmo 2: Heurística Parcialmente Gulosa

```
1  Entrada: máquinas, tarefas, tempos_processamento, janela_inicial_maquinas,
     janela_final_maquinas, janela_final_equipes
2  Saída: maior_equipe
3  equipes_ordenadas_decrescente ← [] Ordenar(janela_final_equipes)
4  maquinas_ordenadas_crescente ← [] Ordenar( janela_final_maquinas)
5  tarefas_ordenadas_crescente ← [] Ordenar(tarefas)
6  for cada máquinas em maquinas_ordenadas_decrescente do
7      for cada tarefas em tarefas_ordenadas_crescente do
8          tarefas_manutencao ← Sortear Tarefas (tempos_processamento)
9          if janela_inicial_maquinas > tempos_processamento then
10             tempos_processamento ← tempos_processamento + janela_inicial_maquinas
11         else
12             tempos_processamento ← tempos_processamento + 0
13         if equipes_ordenadas_decrescente = 1 then
14             equipe_escolhida ← equipes_ordenadas_decrescente
15             equipe_escolhida ← equipe_escolhida -tempos_processamento
16         else
17             Escolher a próxima equipe
18     maior_equipe ← MaiorEquipe(equipe_escolhida)
19 retornar maior_equipe
```

Na linha 1, o algoritmo inicia recebendo como entrada as máquinas, as tarefas, os tempos de processamento, as janelas iniciais e finais das máquinas e o instante de término das equipes de manutenção. A saída esperada é o tamanho da maior equipe após a alocação das tarefas, conforme indicado na linha 2.

Nas linhas 3, 4 e 5, são realizadas, respectivamente, a ordenação das equipes em ordem decrescente. Ou seja, as equipas com o maior tempo de disponibilidade ocupam a primeira posição. A ordenação em ordem crescente das janelas finais das máquinas é feita para garantir que as máquinas com tempo de término menor estejam nas primeiras posições. Por fim, ocorre o arranjo da prioridade das tarefas, com as tarefas mais críticas sendo realocadas para as primeiras posições.

O primeiro laço de repetição na linha 6 executa o comando para cada máquina dentro das máquinas que foram ordenadas. O segundo laço, na linha 7, executa o comando para todas as tarefas. Para cada tarefa escolhida, um tempo de processamento é sorteado e alocado no vetor

$tarefas\_manutencao$, conforme evidenciado na linha 8.

Na linha 9, é feita a verificação se a janela inicial do equipamento é maior que o tempo de processamento. Se for o caso, na linha 10, o tempo inicial desta máquina é acrescentado ao tempo de processamento. Caso contrário, o tempo de processamento permanece nulo, como mostrado nas linhas 11 e 12.

A verificação das equipes para as tarefas ocorre na linha 13, onde o valor 1 indica que a equipe está apta para realizar a tarefa. Sendo assim, o tempo de processamento é alocado e subtraído do tempo final da equipe, conforme mostrado na linha 15. Caso a equipe não seja apta, é realizada a escolha da nova equipe, conforme indicado na linha 17.

Na linha 18, após a alocação de todas as tarefas de manutenção em suas respectivas máquinas, é obtida uma maior equipe existente na lista $maior\_equipe$.

Por fim, o tamanho da $maior\_equipe$, que neste caso é o *makespan*, é retornado como saída do algoritmo.

4.3 Simulated Annealing para o problema proposto

Uma vez que foi possível desenvolver uma solução inicial viável, foi necessário explorar outras soluções e para este problema foi utilizado o *Simulated Annealing*. O Algoritmo 3 apresenta o código base com as alterações que atendam ao modelo proposto.

No Algoritmo 3, na linha 1, temos a definição de algumas variáveis: T_0 e T_c, representando, respectivamente, as temperaturas iniciais e de congelamento; o parâmetro α; o número de iterações T ($inteT$) e de iterações máximas ($iter_{max}$); a solução inicial; e o tempo de processamento das tarefas nas máquinas.

O algoritmo começa inicializando, nas linhas 3, 4 e 5, os valores de $IterT$ como 0, a temperatura T recebe o valor da temperatura inicial T_0, e a solução corrente recebe a solução inicial.

Em seguida, na linha 6, ocorre o laço de repetição principal, que verifica se a temperatura T ainda é maior do que a temperatura de congelamento T_c. Dentro desse laço principal, temos outro *loop*, na linha 7, que é executado um número máximo de iterações $iter_{max}$.

Dentro desse segundo laço de repetição, são realizados movimentos de realocação de tempo de processamento entre equipes, conforme evidenciado na linha 11. A equipe com o maior tempo de término é selecionada como a equipe com maior energia (linha 8), enquanto, na linha 9, a equipe com o menor tempo de término é selecionada como a equipe com menor energia. Na linha 10, o tempo de processamento é selecionado aleatoriamente da equipe com maior energia e realocado para a equipe com menor energia (linha 11).

Após cada movimento, na linha 12, é calculada a diferença entre a solução corrente e a solução inicial, representada por Δ. Se essa diferença for menor que zero (linha 13), significa que a solução corrente é melhor que a solução inicial e, portanto, a melhor solução é atualizada (linha 14).

Algoritmo 3: Algoritmo de Simulated Annealing

```
1  Entrada: α, Iter_max, IterT, T_0, T_c, solução_inicial, tempo_processamento
2  Saída: melhor_solução
3  IterT ← 0;
4  T ← T_0;
5  solução_corrente ← solução_inicial;
6  while T > T_C do
7      while Iter_T < Iter_max do
8          equipe_maior_tempo ← encontrar equipe com maior tempo de término;
9          equipe_menor_tempo ← encontrar equipe com menor tempo de término;
10         tempo_processamento ← selecionar o tempo_processamento da
             equipe_maior_tempo;
11         realocar tempo_processamento da equipe_maior_tempo para
             equipe_menor_tempo;
12         Δ = solução_corrente - solução_inicial;
13         if Δ < 0 then
14             melhor_solução ← solução_corrente;
15         else
16             tomar x ∈ [0,1] ;
17             if x < e^(−Δ/T) then
18                 solução_inicial ← solução_corrente;
19     T ← α * T;
20     IterT ← 0;
21 Retornar melhor_solução;
```

No entanto, se a diferença for maior ou igual a zero, é aplicada a equação de Boltzmann para determinar a probabilidade de aceitar uma solução pior. Na linha 16, um número aleatório x é gerado entre 0 e 1, e se esse número for menor que $e^{-\Delta/T}$ (linha 17), a solução corrente é aceita como a nova solução inicial (linha 18).

Após o término do segundo *loop* na linha 19, a temperatura T é atualizada multiplicando-se por α, e na linha 20, o valor de $IterT$ é redefinido como 0. Isso simula o resfriamento lento do sistema.

O algoritmo continua executando até que a temperatura T atinja a temperatura de congelamento T_c, momento em que a melhor solução encontrada é retornada como resultado.

5 Resultados

Neste capítulo, serão discutidos todos os procedimentos e métodos que foram determinantes para a elaboração desta dissertação. Na Seção 5.1 , serão esclarecidos os processos de coleta e tratamento de dados, bem como as características do ambiente computacional que foram usados para os testes. Na Seção 5.2, são apresentados os resultados obtidos através de cada método empregado e as conclusões que foram obtidas.

5.1 Coleta de dados e Ambiente computacional

As instâncias que foram pensadas para este experimento são oriundas das ordens de manutenção de um software de gestão empresarial do tipo ERP (*Enterprise Resources Planning*), comumente conhecido como SAP (*Sistemas, Aplicativos e Produtos para Processamento de Dados*) de uma indústria de mineração de médio porte que tem como tarefa principal o beneficiamento de gnaisse.

Esses documentos contêm os tempos de processamento das tarefas de cada máquina e as equipes que são habilitadas para a execução dessas tarefas. As ordens de manutenção podem ser classificadas em semanais, mensais ou anuais, mas para este trabalho só foram consideradas as ordens de manutenção semanais e mensais, uma vez que todos os parâmetros foram dimensionados levando em consideração um horizonte de planejamento de curto a médio prazo, com o intuito de obter resultados prósperos e imediatos.

Todas as instâncias foram solucionadas no solver IBM CPLEX *Optimizer* na versão 12.9.0, utilizando a API *Python* do CPLEX. A execução do código foi realizada em um computador com sistema operacional *Windows* 11, processador *AMD Ryzen* I5, 8GB de memória RAM e sistema de 64 bits.

Os dados para o problema foram separados em três grupos, sendo eles: as instâncias pequenas usadas apenas para a validação do modelo matemático com informações que não condizem com a realidade, e as instâncias médias e grandes que foram extraídas das ordens de manutenção e representam o contexto real da empresa.

Na Tabela 5 é possível verificar as características de cada grupo de instâncias.

Os parâmetros de entrada do *simulated annealing* são representados na Tabela 6.

5.2 Resultados computacionais

Os testes iniciais foram realizados no *s*olver CPLEX, onde foi possível não apenas validar o modelo proposto, mas também encontrar o ótimo global nas instâncias de dimensões pequenas e médias.

Tabela 5 – Características das Instâncias.

	Número da instância	Número de tarefas	Número de Equipamentos	Número de Equipes
PEQUENAS	1	3	4	2
	2	5	9	3
	3	9	9	5
	4	10	15	5
	5	13	18	4
	6	18	20	4
	7	18	23	6
	8	18	25	6
	9	20	28	7
	10	20	30	8
MÉDIAS	11	27	30	8
	12	35	38	8
	13	35	40	9
	14	35	42	9
	15	50	55	9
	16	65	55	10
	17	58	65	10
	18	70	60	11
	19	70	75	11
	20	80	75	11
GRANDES	21	80	90	11
	22	110	100	19
	23	120	100	19
	24	140	110	19
	25	140	110	20
	26	150	120	20
	27	160	120	20
	28	160	130	21
	29	165	130	21
	30	165	140	21

Tabela 6 – Parâmetros *simulated annealing*.

Parâmetros	Valores
α	0,975
$Iter_{max}$	número de tarefas multiplicado pelo número de máquinas $(i * a)$
T_0	100
T_C	0,01

Os resultados dessa aplicação estão discriminados na Tabela 7.

Tabela 7 – Resultados do Solver

N. Instância	Solução Inteira		Relaxação Linear		*gap*
	Tempo(s)	FO(min)	Tempo(s)	FO(min)	
1	0,09	15,00	0,01	14,46	3,60
2	0,59	24,00	0,01	23,00	4,17
3	0,08	54,00	0,01	48,72	9,78
4	0,05	80,00	0,01	68,81	13,99
5	0,19	61,00	0,12	56,00	8,20
6	0,13	64,00	0,17	60,00	6,25
7	0,75	53,00	0,23	41,00	22,64
8	0,81	58,00	0,70	41,00	29,31
9	1,94	68,00	1,05	49,72	26,88
10	1,00	74,00	1,03	59,48	19,62
11	3,67	74,00	1,02	62,19	15,96
12	1,64	183,00	1,03	123,09	32,74
13	3,42	101,00	1,03	100,89	0,11
14	4,19	105,00	1,03	103,22	1,70
15	9,67	137,00	2,30	105,32	23,12
16	11,30	136,00	3,20	108,00	20,59
17	10,03	156,00	3,25	104,89	32,76
18	12,09	151,00	4,02	113,23	25,01
19	22,78	177,00	4,36	114,57	35,27
20	27,08	177,00	4,37	117,00	33,90
21	-	-	4,37	102,00	-
22	-	-	5,60	127,00	-
23	-	-	6,30	175,00	-
24	-	-	7,48	186,00	-
25	-	-	10,36	217,00	-
26	-	-	-	-	-
27	-	-	-	-	-
28	-	-	-	-	-
29	-	-	-	-	-
30	-	-	-	-	-

Com base nos resultados apresentados na Tabela 7, podemos fazer uma explicação comparativa entre a solução inteira (obtida pelo *solver*) e a solução relaxada (resultante da relaxação linear) para o problema em questão.

A coluna (ii) mostra o tempo em segundos que o *solver* levou para encontrar a solução inteira para cada instância do problema. Observamos que o tempo varia de acordo com a complexidade da instância, sendo que problemas maiores demandam um tempo de processamento maior devido o aumento do número de tarefas, máquinas e equipes.

A coluna (iv) indica o valor da função objetivo (*makespan*) em minutos, obtido a partir da solução inteira encontrada pelo *solver*. Esses valores representam o tempo mínimo necessário

para executar todas as tarefas do problema, de acordo com a solução encontrada pelo *solver*.

Por outro lado, as colunas (iii) e (v) apresentam o tempo em segundos e o valor da função objetivo (*makespan*), respectivamente, provenientes da solução relaxada através da relaxação linear. Nesse caso, as restrições que impõem que as variáveis sejam inteiras são relaxadas, permitindo a obtenção de uma solução inicial.

É importante notar que, em alguns casos, a solução relaxada pode ser encontrada mais rapidamente do que a solução inteira, pois a relaxação linear reduz a complexidade do problema ao permitir valores fracionários para as variáveis. No entanto, a solução relaxada tende a ser inferior à solução inteira em termos de valor da função objetivo.

Comparando os valores da função objetivo das colunas (iv) e (v), podemos observar que a solução inteira FO da coluna (iv) tende a ser igual ou superior à solução relaxada FO da coluna (v), pois a relaxação linear oferece apenas um limite inferior para a função objetivo. A diferença entre esses valores é o que chamamos de *gap* de integralidade, representado pela coluna (VI), que indica o quão distante a solução relaxada está da solução ótima (inteira) e que é obtida pela Equação (17).

$$gap = \left(\frac{LS - RL}{LS} \right) \times 100 \tag{17}$$

Considera-se LS o limite superior e RL a relaxação linear da solução.

Portanto, em geral, espera-se que os valores da função objetivo na coluna (iv) sejam maiores ou iguais aos valores na coluna (v), uma vez que a relaxação linear fornece um limite inferior para o *makespan* do problema. A análise do *gap* de integralidade é essencial para avaliar a qualidade das soluções encontradas, bem como a eficiência do modelo matemático e do *solver* na abordagem do problema de otimização.

O terceiro grupo, composto por instâncias classificadas como grandes, o *solver* não conseguiu atender todas as instâncias do modelo devido às extensas dimensões de variáveis e parâmetros. Não foi possível processar e explorar todas as possibilidades de arranjos, resultando em erros de memória em todos os casos.

Para essas situações em que não foi possível obter soluções completas, foram empregados dois algoritmos, nos quais foram realizados 30 testes para cada grupo de instâncias. O primeiro algoritmo foi uma heurística parcialmente gulosa, com o objetivo de construir uma solução inicial viável para todos os casos, e o segundo, uma meta-heurística denominada *simulated annealing*.

Os valores resultantes dessas abordagens são exibidos na Tabela 8.

Tabela 8 – Resultados Heurística

N. Instância	HPG			SA		
	Tempo (s)	FO (min)	gap	Tempo (s)	FO (min)	gap
1	0,01	16,00	6,25	0,01	15,00	0,00
2	0,01	51,00	52,94	0,01	48,00	50,00
3	0,01	84,00	35,71	0,01	77,00	29,87
4	0,01	84,00	4,76	0,03	81,00	1,23
5	0,01	125,00	51,20	0,20	117,00	47,86
6	0,01	125,00	48,80	0,30	123,00	47,97
7	0,01	133,00	60,15	0,30	89,00	40,45
8	0,01	136,00	57,35	0,30	91,00	36,26
9	0,01	137,00	50,36	0,30	139,00	51,08
10	1,43	145,00	48,97	0,30	143,00	48,25
11	2,34	158,00	53,16	2,46	95,00	22,11
12	2,56	292,00	37,33	3,14	278,00	34,17
13	2,56	264,00	61,74	3,27	203,00	50,25
14	2,56	264,00	60,23	3,45	220,00	52,27
15	3,46	290,00	52,76	4,10	198,00	30,81
16	5,60	301,00	54,82	6,36	145,00	6,21
17	7,03	398,00	60,80	8,02	267,00	41,57
18	7,12	397,00	61,96	8,02	209,00	27,75
19	7,18	404,00	56,19	8,04	367,00	51,77
20	7,41	407,00	56,51	8,18	345,00	48,70
21	12,24	412,00	-	12,40	378,00	-
22	10,45	441,00	-	12,10	387,00	-
23	12,48	477,00	-	13,06	387,00	-
24	14,00	502,00	-	14,50	441,00	-
25	16,30	505,00	-	17,03	434,00	-
26	17,10	592,00	-	17,05	421,00	-
27	17,00	525,00	-	17,54	500,00	-
28	17,60	510,00	-	17,55	498,00	-
29	17,60	576,00	-	17,55	476,00	-
30	17,65	556,00	-	17,55	497,00	-

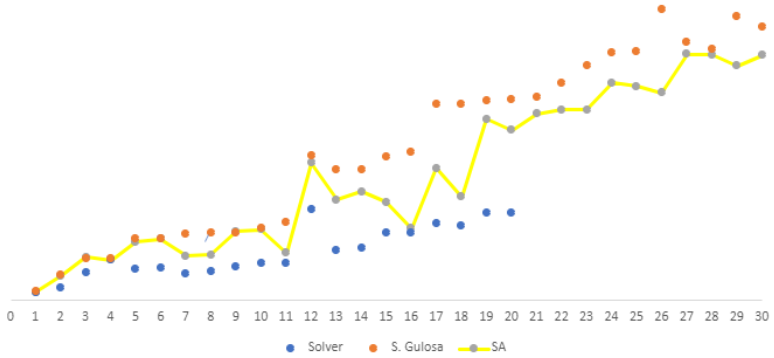
A Tabela 8 apresenta os resultados das duas heurísticas aplicadas a diferentes instâncias do problema, onde a função objetivo (*makespan*) é minimizada. As heurísticas são denominadas Heurística Parcialmente Gulosa (HPG) e *Simulated Annealing* (SA).

Ambas as heurísticas visam encontrar soluções aproximadas para o problema em questão. A heurística HPG utiliza uma abordagem parcialmente gulosa, onde cada decisão é tomada com base na melhor escolha local em cada etapa, enquanto a heurística SA se inspira no processo de resfriamento simulado, buscando explorar o espaço de soluções e, eventualmente, aceitar movimentos que levem a soluções piores para escapar de mínimos locais e alcançar uma solução mais próxima do ótimo global.

É importante também destacar que após a implementação do algoritmo *simulated annealing*, foi possível melhorar a solução e aproximar os resultados do ótimo. No entanto,

devido à complexidade do problema, não foi viável explorar todos os movimentos de realocação conforme demonstrado na Figura 11 onde é possível verificar que alguns valores estão distantes do ótimo.

Figura 11 – Comparação das soluções encontradas II



Fonte: Elaborado pelo autor

Com o intuito de ampliar o espaço de possibilidades e facilitar mais movimentos de realocações, foram realizados aumentos de 2% e 5% nas equipes de manutenção.

Esses resultados estão apresentados na Tabela 9 e na Tabela 10.

Com o aumento de 2% e 5% na equipe, confrontou-se que as soluções encontradas tanto pela Heurística Parcialmente Gulosa (HPG) quanto pelo Simulated Annealing (SA) apresentaram uma redução no tempo total, representada pela função objetivo (*makespan*). Isso indica que o acréscimo na equipe permitiu a conclusão das tarefas de forma mais rápida, adquiridas em *makespans* menores.

Além disso, aumentando o número de equipes e mantendo o número de máquinas e tarefas, ambos os algoritmos foram capazes de encontrar novas maneiras de realizar essas realocações,e assim, reduzir significativamente o valor do *gap*. Esse resultado indica que as soluções encontradas pelas heurísticas ficaram mais próximas das soluções ideais, ou seja, mais próximas do ótimo global.

É possível também inferir que o aumento de 5% na equipe da equipe apresentou resultados superiores em relação ao aumento de 2%, conforme é evidenciado nas Figura 12 e Figura 13. Os acréscimos permitiram uma extensão significativa das possibilidades de alocação de recursos enquanto a disponibilidade da configuração inicial das equipes pode ter sido insuficiente para promover uma diversificação adequada nas opções de solução, limitando o desempenho do modelo.

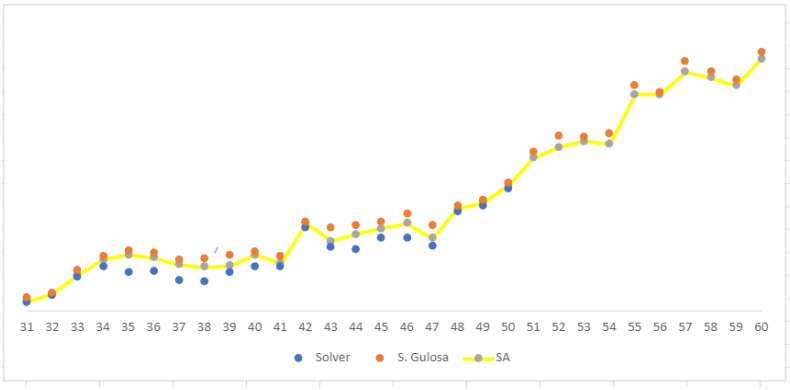
Tabela 9 – Aumento de 2% na equipe

N. Instância	Makespan (min)			GAP	
	Solver	HPG	SA	HPG	SA
31	12,00	18,00	12,00	20,00	0
32	20,00	24,00	23,00	16,67	4,76
33	46,00	56,00	49,00	17,86	6,12
34	60,00	74,00	70,00	18,92	7,69
35	52,00	82,00	77,00	27,78	8,77
36	54,00	80,00	73,00	26,03	8,47
37	41,00	70,00	63,00	28,07	4,65
38	40,00	72,00	60,00	23,08	9,09
39	52,00	76,00	62,00	21,21	8,77
40	61,00	81,00	77,00	24,69	8,96
41	60,00	74,00	65,00	18,92	7,69
42	114,00	122,00	120,00	6,56	5,00
43	88,00	114,00	96,00	22,81	8,33
44	84,00	117,00	105,00	28,21	6,67
45	100,00	123,00	113,00	18,70	9,09
46	100,00	134,00	120,00	25,37	9,09
47	89,00	117,00	100,00	23,93	9,18
48	136,00	145,00	140,00	6,21	2,86
49	145,00	153,00	149,00	5,23	2,68
50	169,00	177,00	173,00	4,52	2,31
51	-	220,00	211,00	-	-
52	-	241,00	226,00	-	-
53	-	240,00	234,00	-	-
54	-	245,00	230,00	-	-
55	-	311,00	299,00	-	-
56	-	302,00	299,00	-	-
57	-	345,00	330,00	-	-
58	-	330,00	322,00	-	-
59	-	320,00	311,00	-	-
60	-	358,00	348,00	-	-

Tabela 10 – Aumento de 5% na equipe

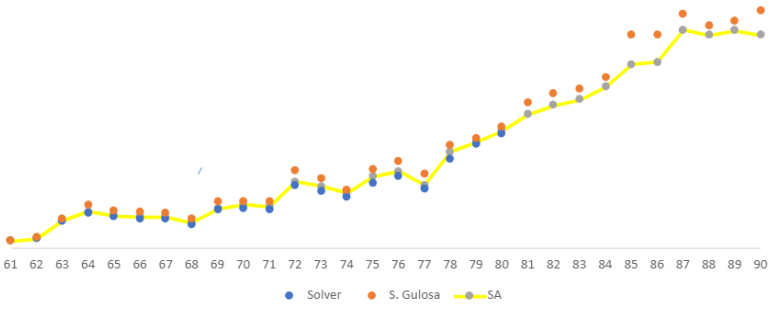
N. Instância	Makespan (min)			GAP	
	Solver	HPG	SA	HPG	SA
61	9,00	9,00	9,00	0,00	0
62	12,00	14,00	12,00	14,29	0
63	34,00	38,00	36,00	10,53	5,56
64	45,00	56,00	47,00	19,64	4,26
65	40,00	48,00	42,00	16,67	4,76
66	38,00	46,00	40,00	17,39	5,00
67	37,00	45,00	40,00	17,78	7,50
68	30,00	37,00	33,00	18,92	9,09
69	49,00	61,00	51,00	19,67	3,92
70	52,00	60,00	56,00	13,33	7,14
71	50,00	61,00	54,00	18,03	7,41
72	82,00	101,00	86,00	18,81	4,65
73	74,00	91,00	80,00	18,68	7,50
74	66,00	76,00	71,00	13,16	7,04
75	85,00	102,00	93,00	16,67	8,60
76	94,00	113,00	100,00	16,81	6,00
77	77,00	96,00	82,00	19,79	6,10
78	116,00	134,00	125,00	13,43	7,20
79	136,00	143,00	138,00	4,90	1,45
80	149,00	158,00	153,00	5,70	2,61
81	-	189,00	174,00	-	-
82	-	201,00	186,00	-	-
83	-	207,00	194,00	-	-
84	-	223,00	210,00	-	-
85	-	278,00	240,00	-	-
86	-	278,00	243,00	-	-
87	-	306,00	285,00	-	-
88	-	290,00	278,00	-	-
89	-	296,00	284,00	-	-
90	-	310,00	278,00	-	-

Figura 12 – Resultados obtidos com o acréscimo de 2%



Fonte: Elaborado pelo autor

Figura 13 – Resultados obtidos com o acréscimo de 5%



Fonte: Elaborado pelo autor

6 Conclusão

Esta dissertação teve como objetivo desenvolver uma formulação matemática capaz de modelar um problema real de sequenciamento de ordens de manutenção, levando em consideração restrições de tempo e disponibilidade de mão de obra, a fim de obter um arranjo que resultasse no menor *makespan* possível. Através desse modelo, foram obtidos valores ótimos para o problema nas instâncias de tamanho pequeno e médio. No entanto, para as instâncias maiores, foi necessário implementar uma heurística construtiva para obter uma solução inicial viável.

A heurística desenvolvida foi capaz de produzir resultados para todos os grupos de instâncias. No entanto, observou-se que, à medida que o espaço de busca aumentava, os resultados obtidos se distanciavam da solução ótima. Para melhorar essas soluções, foi empregada a meta-heurística *Simulated Annealing*.

O objetivo do *Simulated Annealing* era explorar o espaço de solução de forma mais eficiente e encontrar soluções mais próximas da ótima. Essa abordagem foi aplicada em todas as instâncias, inclusive nas de maior dimensão, onde a busca exaustiva não era viável devido à complexidade do problema.

Assim, a combinação da formulação matemática, da heurística construtiva e do *Simulated annealing* permitiu avançar na busca de soluções para o problema de sequenciamento de ordens de manutenção, obtendo resultados ótimos para instâncias menores e melhorando as soluções para instâncias maiores.

Por fim foi realizado um o aumento na disponibilidade de mão de obra em que teve um impacto positivo nas soluções encontradas pelas heurísticas. O acréscimo de recursos de mão de obra permitiu a realização das tarefas de forma mais rápida e eficiente, resultando em *makespans* menores e soluções mais próximas do ótimo global. Essa análise demonstra a importância da capacidade de mão de obra na otimização do problema e sugere que um gerenciamento adequado desses recursos pode levar a melhores resultados no planejamento e execução das tarefas.

6.1 Trabalhos futuros

Embora esta dissertação tenha alcançado resultados promissores, há oportunidades para futuras explorações e aprimoramentos. Um ponto fundamental reside na melhoria da formulação matemática do PSTEEM, dada a complexidade intrínseca deste problema. Uma investigação mais detalhada do modelo se faz necessária, com o intuito de incorporar restrições adicionais de maneira mais precisa.

Uma estratégia potencial para aprimorar ainda mais as soluções consiste na exploração de diversos tipos de movimentos de troca e realocação. Embora o algoritmo atual esteja focado em um conjunto específico de movimentos, explorar alternativas pode abrir novas perspectivas.

Por fim, seria altamente benéfico explorar outras heurísticas, como algoritmos genéticos

e busca tabu, entre outros. Essas abordagens ofereceriam diferentes formas de explorar o espaço de soluções, permitindo superar limitações específicas do algoritmo atual. Ao considerar tais estratégias alternativas, será possível ampliar a compreensão do problema e proporcionar soluções mais robustas e eficientes.

6.2 Trabalhos Publicados e Aceitos em Congresso

A partir dos resultados parciais deste estudo, um artigo foi elaborado e apresentado no Simpósio Brasileiro de Pesquisa Operacional (SBPO):

RODRIGUES, J. M.; MENEZES, G. C. **Aplicação de um Modelo Matemático para o Sequenciamento de Tarefas de Manutenção em uma Indústria de Mineração de Gnaisses**. Galoá, Anais do Simpósio Brasileiro de Pesquisa Operacional, 2022.

Adicionalmente, um outro artigo foi aceito para apresentação no Simpósio Brasileiro de Pesquisa Operacional (SBPO) em 2023:

RODRIGUES, J. M.; MENEZES, G. C. ***Simulated Annealing*: uma abordagem eficiente para problemas de sequenciamento de ordens de manutencão**

Referências

AGUIAR, M. O.; MAURI, G. R.; SILVA, R. F. **Introdução aos Métodos Heurísticos de Otimização com Python**. 1. ed. Porto Alegre: Caufes, 2018. Citado na página 25.

ANDRADE, J. V. da C.; TANAKA, S. S.; TANAKA, S. A. Desenvolvimento de uma hiper-heurística aplicada ao escalonamento em problemas de job shop. **Revista Terra & Cultura**, v. 39, n. especial, p. 129–135, 2023. Citado 2 vezes nas páginas 27 e 29.

AQUINO, R. D. **Abordagem Exata e Heurísticas para o Problema de Planejamento de Ordens de Manutenção de Longo Prazo**: um estudo de caso industrial de larga escala. 2018. 95 p. Dissertação (Mestrado em Ciência da computação) — Universidade Federal de Ouro Preto, Minas Gerais, 2018. Citado 3 vezes nas páginas 27, 32 e 34.

BARBOSA, L. C. **Um algoritmo simheurístico para a resolução do Problema de Flow Shop Permutacional multiobjetivo.** 2022, 47 p. Monografia (Bacharel em Engenharia de Produção) — Universidade Federal de Ouro Preto, Ouro Preto, 2022. Citado 2 vezes nas páginas 27 e 28.

BOLTZMANN, L. **Theoretical physics and philosophical problems: Selected writings**. Boston: Springer Science & Business Media, 2012. v. 5. Citado 2 vezes nas páginas 24 e 25.

CHIAVENATO, I. **Teoria geral da administração**. São Paulo: Elsevier Brasil, 2002. Citado na página 14.

DAHINTEN, A. K. **Proposição de sequenciamento Job shop em uma fábrica de elevadores**. 2014, 20 p. Monografia (Bacharelado em engenharia de produção) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2014. Citado 2 vezes nas páginas 27 e 28.

DOBOS, P.; TAMÁS, P.; ILLÉS, B. Decision method for optimal selection of warehouse material handling strategies by production companies. **IOP Conference Series: Materials Science and Engineering**, IOP Publishing, v. 161, n. 1, p. 012100, nov 2016. Disponível em: <https://dx.doi.org/10.1088/1757-899X/161/1/012100>. Citado na página 20.

FLIZICOSKI, A. L. V. **Aplicação da minimização do atraso total em ambiente de máquina única com tempos de setup dependentes da sequência**. 2021. 67 p. Monografia (Bacharelado em Engenharia de produção) — Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2017. Citado na página 22.

FUCHIGAMI, H. Y.; MOURA, M. A. S.; BRANCO, F. J. C. Modelos matemáticos para programação de job shop com tempos de setup independentes da sequência. **Revista Produção Online**, v. 17, n. 1, p. 245–267, 2017. Citado na página 19.

GONÇALVES, J. E. L. Os impactos das novas tecnologias nas empresas prestadoras de serviços. **Revista de Administração de Empresas**, v. 34, p. 63–81, 1994. Citado na página 14.

GONZALEZ, T. F. **Handbook of approximation algorithms and metaheuristics**. Santa Bárbara: Chapman and Hall/CRC, 2007. Citado na página 23.

JÚNIOR, V. F.; ALMEIDA, C.; VENSKE, S. Algoritmo híbrido para o problema flow shop de permutação multiobjetivoalgoritmo híbrido para o problema flow shop de permutação multiobjetivo. In: ENCONTRO NACIONAL DE INTELIGêNCIA ARTIFICIAL E

COMPUTACIONAL, 44., 2020, Porto Alegre. **Anais[...]**. Porto Alegre: SBC, 2020. p. 82–93. Disponível em: <https://sol.sbc.org.br/index.php/eniac/article/view/12119>. Citado na página 22.

KIRKPATRICK, S.; JR, C. D. G.; VECCHI, M. P. Optimization by simulated annealing. **science**, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983. Citado na página 24.

LUSTOSA, L.; MESQUITA, M. A.; OLIVEIRA, R. J. **Planejamento e controle da produção**. 4. ed. Rio de Janeiro: Elsevier Brasil, 2008. Citado na página 20.

MARTINS, P. G.; LAUGENI, F. P. **Administração da Produção.** São Paulo: Saraiva, 2005. Citado na página 14.

MELO, S. E. G. de. **Planejamento de processos de fabricação e montagem integrada à programação da produção em estaleiros de construção naval**. 2010. 93 p. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2010. Citado na página 22.

METROPOLIS, N. et al. Equation of state calculations by fast computing machines. **The journal of chemical physics**, American Institute of Physics, v. 21, n. 6, p. 1087–1092, 1953. Citado na página 24.

MOCCELLIN, J. V.; NAGANO, M. S. Flow shop híbrido com estágios gargalos. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 35., 2003, Natal. **Anais[...]**. Natal: SOBRAPO, 2003. v. 35, p. 47 – 59. Citado na página 20.

NAGANO, M. S.; MOCCELLIN, J. V.; LORENA, L. A. N. Programaçao da produçao flow shop permutacional com minimizaçao do tempo médio de fluxo: proposta metodológica. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL., 36., 2004, Rio de Janeiro. **Anais[...]**. Rio de Janeiro: SOBRAPO, 2004. v. 3, n. 4, p. 81–91. Citado na página 19.

NAPIERALA, H. O algoritmo genético para solucionar a programação de produção do sistema de fluxo permutacional. **Ciências sociais aplicadas em Revista**, v. 13, n. 24, p. 175–189, 2000. Citado na página 29.

NETO, R. F. T. **Proposta de solução de problemas de scheduling considerando possibilidade de terceirização usando a técnica de otimização por colônia de formigas**. 2010, 148 p. Tese (Doutorado em Engenharia de Produção) — Universidade Federal de São Carlos, São Carlos (SP), 2010. Citado na página 20.

PEREIRA, M. S. **Predição do consumo de energia elétrica em sistemas Job shop utilizando técnicas de aprendizado de máquina**. 2019. 134 p. Dissertação (Mestrado em Engenharia Mecânica) — Centro Universitário FEI, São Bernardo do Campo, 2019. Citado na página 20.

PINEDO, M. L. **Scheduling**: theory, algorithms and systems. 3. ed. New York: Springer, 2008. Citado na página 22.

RIZZO, L. M.; URRUTIA, S. Uma heurística grasp para o problema estendido de sequenciamento de carros. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 2011, Ubatuba. **Anais[...]**. Ubatuba: SOBRAPO, 2011. p. 8. Citado na página 23.

ROCHA, M. L.; ALVARENGA, F. V. de; SILVA, R. B. Aplicações de heurísticas à sobrevivência de redes. In: SIMPÓSIO DE PESQUISA OPERACIONAL E LOGÍSTICA DA MARINHA, 8., 2006, Palmas. **Anais[...]**. Palmas: CEULP/ULBRA, 2006. p. 133–142. Citado na página 23.

SANT'ANNA, C. P. C. **Modelos de programação linear inteira mista para o problema de programação de manutenções em linhas de transmissão energizadas e desenergizadas**. 2022. 78 p. Dissertação (Mestrado em Engenharia de Produção) — Universidade Federal de Pernambuco, Recife, 2022. Citado na página 27.

SILVA, M. K. A. **Otimização do problema de sequenciamento da produção em uma indústria química com tempos de setup assimétricos e dependentes**. 2018. 57 p. Monografia (Bacharelado em Engenharia de Produção) — Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2018. Citado 3 vezes nas páginas 27, 28 e 29.

SIQUEIRA, E. C. de; SOUZA, S. R. de; SOUZA, M. J. F. Um algoritmo baseado em iterated greedy para minimização do makespan no problema de sequenciamento flowshop híbrido e flexível. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 44., 2012, Rio de Janeiro. **Anais[...]**. Rio de Janeiro: SOBRAPO, 2012. p. 12 – 24. Citado na página 23.

TIDD, J.; BESSANT, J. **Gestão da inovação-5**. Porto Alegre: Bookman Editora, 2015. Citado na página 14.

TUBINO, D. F. **Planejamento e controle da produção**: teoria e prática . São Paulo: Editora Atlas SA, 2000. Citado na página 20.

VIEGAS, V. A. **O problema de sequenciamento de tarefas em máquinas paralelas: um estudo de caso em um departamento de credenciamento**. 2016, 45 p. Monografia (Bacharelado em Engenharia de produção) — Universidade Federal Fluminense, 2016. Citado na página 20.

VIEIRA, R.; TAGLIALENHA, S. Lopes de S. Heurística de vizinhança variável para programação de robôs móveis autônomos em ambiente de manufatura flexível. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUCÃO, 42., 2022, Foz do iguaçu. **Anais[...]**. Foz do iguaçu: ABREPO, 2022. p. 15 – 30. Citado 4 vezes nas páginas 27, 29, 30 e 31.

VIVAN, C. J. Aplicação do método simulated annealing em um problema real de sequenciamento da produção. **Universidade Federal do Paraná, Pós-Graduação em Métodos Numéricos em Engenharia**, 2010. Citado 2 vezes nas páginas 27 e 31.

WANG, J.; XIA, Z. Flow-shop scheduling with a learning effect. **Journal of the Operational Research Society**, Taylor & Francis, v. 56, n. 11, p. 1325–1330, 2005. Citado na página 19.