UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA DE TRANSPORTES E LOGÍSTICA

HENRIQUE ARAÚJO FERNANDES

SIMULATED ANNEALING PARA PROGRAMAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS EM AMBIENTE DE MANUFATURA FLEXÍVEL

Joinville
2021

HENRIQUE ARAÚJO FERNANDES

SIMULATED ANNEALING PARA PROGRAMAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS EM AMBIENTE DE MANUFATURA FLEXÍVEL

Trabalho apresentado como requisito para obtenção do título de bacharel em Engenharia de Transportes e Logística do Centro Tecnológico de Joinville da Universidade Federal de Santa Catarina.

Orientadora: Dra. Silvia Lopes de Sena Taglialenha

Joinville
2021

HENRIQUE ARAÚJO FERNANDES

SIMULATED ANNEALING PARA PROGRAMAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS EM AMBIENTE DE MANUFATURA FLEXÍVEL

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de bacharel em Engenharia de Transportes e Logística, na Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Joinville (SC), 23 de setembro de 2021.

Banca Examinadora:

_____

Orientadora: Dra. Silvia Lopes de Sena Taglialenha
Orientador(a)
Presidente

_____

Profa. Dra. Christiane Wenck Nogueira Fernandes
Membro(a)
Universidade Federal de Santa Catarina

_____

Me. Natan Bissoli
Membro(a)

Dedico este trabalho aos meus pais. Este TCC é a prova de que todo seu investimento e dedicação valeram a pena.

AGRADECIMENTOS

Agradeço aos meus pais, pela base e apoio que me deram e por terem sonhado comigo para que este momento chegasse.
Agradeço à Luíza que divide a vida comigo, me aguenta e me apoia em todas as decisões.
Agradeço aos amigos que fiz nessa jornada, que sempre estiveram ao meu lado e foram um forte auxilio em momentos difíceis e se mostraram fiéis e companheiros pelo caminho.
Agradeço à professora Silvia pela paciência e dedicação ao longo desse tempo.
Agradeço à banca, não só por ter aceitado o convite, mas também por compartilhar deste momento comigo.
Agradeço a todos os professores que tive oportunidade de conhecer ao longo desta jornada, que contribuíram para que eu chegasse até aqui, que me ajudaram a abrir portas na minha vida.
Agradeço, de maneira geral, a todos os colaboradores da UFSC, que me receberam de braços abertos, e aos meus colegas de trabalho, que me ajudam diariamente no meu desenvolvimento profissional e acreditam no meu potencial.
Agradeço à toda minha família por terem acreditado no meu potencial e me ajudado em todo o processo.

RESUMO

Apresenta-se neste trabalho o problema de sequenciamento de tarefas de um único robô móvel autônomo (AMR - *Autonomous Mobile Robot*) responsável por providenciar peças para máquinas de uma linha de produção localizada em um ambiente de manufatura flexível. O problema consiste em programar o AMR de modo que evite que as máquinas interrompam seu funcionamento devido à falta de peças. Um método de solução exata que utiliza o modelo matemático de Programação Linear Inteira Mista (PLIM) é implementada, em linguagem AMPL, e comparada com trabalhos anteriores. O modelo tem como objetivo minimizar o *makespan* - tempo total requerido para a execução de todas as tarefas - em uma linha de produção para um horizonte de planejamento pré-estabelecido. A abordagem leva em consideração tanto as características das máquinas quanto a capacidade de carregamento do robô, e define uma janela de tempo restrita para a realização das tarefas. Como este é um problema *NP-Hard* uma meta-heurística baseada no Simulated Annealing é apresentada, resultando em um aumento significativo da localização de soluções próximas à ótima. Os resultados encontrados apresentam uma redução significativa do *makespan*, em relação ao trabalho comparado. Os resultados melhoram conforme o número total de tarefas aumenta, chegando a encontrar uma redução 16,67% do tempo total de viagem do robô para os cenários comparados, e uma redução média de 6%.

Palavras-chave: *Scheduling*. Sistema de manufatura flexível. *Autonomous Mobile Robot*. Programação Linear Inteira Mista

ABSTRACT

This work considers the task sequencing problem of a single autonomous mobile robot (AMR -Autonomous Mobile Robot) responsible for providing parts for machines on a production line located in a flexible manufacturing environment. The problem is to program the AMR so that it prevents the machines from interrupting their operation due to lack of parts. An exact solution method that uses the mathematical model of Mixed Integer Linear Programming (MILP) is implemented, in AMPL language, and compared with previous works. The model aims to minimize the makespan - total time required to perform all tasks - on a production line for a pre-established planning horizon. The approach takes into account both the characteristics of the machines and the loading capacity of the robot, and defines a restricted time window for carrying out the tasks. As this is an NP-Hard problem, a meta-heuristic based on Simulated Annealing is presented, resulting in a significant increase in the location of close-optimal solutions. The results found show a significant reduction in the makespan, in relation to the work compared. The results become better as the total number of tasks increases, reaching a 16.67% reduction in the total robot travel time for the compared scenarios, and an average reduction of 6%.

Keywords: Scheduling. Flexible Manufacturing System. Autonomous Mobile Robot. Integer Linear Programming

LISTA DE FIGURAS

Figura 1 – Etapas do desenvolvimento do trabalho 14
Figura 2 – Representação de processamento em máquina única 19
Figura 3 – Representação de processamento em máquinas paralelas 19
Figura 4 – Representação de processamento em máquinas em Job Shop . . . 20
Figura 5 – Representação de processamento em máquinas em *Flow Shop* . . 21
Figura 6 – Representação de processamento em máquinas em *Flexible Flow Shop* 21
Figura 7 – Layout da célula da linha de produção 23
Figura 8 – Ótimo Local e Global 30
Figura 9 – Fluxograma do procedimento SA 32
Figura 10 – Representação das soluções via SA 36
Figura 11 – Exemplo de solução inicial 37
Figura 12 – *Status* do AMR ao longo da Solução Inicial 37
Figura 13 – Simulação para cálculo da temperatura inicial 38
Figura 14 – Fluxograma do procedimento para gerar pertubações 39
Figura 15 – Possibilidades de trocas de tarefas 40
Figura 16 – Possibilidades de trocas de rotas 40
Figura 17 – Representação das janelas de tempo utilizada em Dang et al. (2013) e a proposta pelo autor 42
Figura 18 – Codificação da solução ótima do modelo proposto para o cenário 10 45
Figura 19 – Representação gráfica da solução ótima do cenário 10 45
Figura 20 – Solução incumbente e tempo de processamento conforme taxa de resfriamento 47
Figura 21 – Solução incumbente e tempo de processamento conforme o número de iterações dentro do mesmo nível de temperatura 48
Figura 22 – Resultados do SA para cenários maiores 50

LISTA DE TABELAS

Tabela 1 – Níveis de peças e taxas de alimentação 41
Tabela 2 – Tempo de trabalho do robô nos alimentadores 43
Tabela 3 – Tempo de viagem do robô entre as localizações (segundos) 43
Tabela 4 – Cenários analisados 43
Tabela 5 – Comparação dos resultados para os diferentes cenários 44
Tabela 6 – Resultados obtidos pelo SA para o cenário 10 46
Tabela 7 – Definição dos parâmetros para o SA 49
Tabela 8 – Comparação dos resultados para os diferentes cenários 49
Tabela 9 – Resultados do SA para cenários maiores 50

LISTA DE SIGLAS

AGV *Automated Guided Vehicle*

AMPL A Mathematical Programming Language

AMR *Autonomous Mobile Robot*

FMS *Flexible Manufacturing System*

FO Função Objetivo

IG *Iterated Greedy*

ILS *Iterated Local Search*

PLIM programação linear inteira mista

PO Pesquisa Operacional

SA *Simulated Annealing*

SAMR *Sechduling a single AMR*

SLC *Small Loaded Carries*

SUMÁRIO

1 INTRODUÇÃO 11
1.1 Objetivos 13
1.1.1 Objetivo Geral 13
1.1.2 Objetivos Específicos 13
1.2 Metodologia 13
1.3 Estrutura do Trabalho 15

2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 Problema de Sequenciamento 16
2.2 Sequenciamento em Máquinas 18
2.2.1 Modelo de Máquina Única 18
2.2.2 Modelo de Máquina em Paralelo 19
2.2.3 Modelo de *Job Shop* 20
2.2.4 Modelo *Flow Shop* 20
2.2.5 Modelo de *Flexible Flow Shop* 21
2.3 Sequenciamento de tarefas e robôs autônomos 22
2.3.1 Definição do problema 22
2.3.2 Modelagem Matemática para o SAMR 23
2.4 Complexidade do SAMR 27
2.5 Métodos de solução 28
2.5.1 Solução via Programação Matemática 28
2.5.2 Solução via Métodos Heurísticos 29
2.6 *Simulated Annealing* 31

3 MÉTODOS PROPOSTOS 34
3.1 Método de solução via Programação matemática 34
3.2 Método de Solução via *Simulated Annealing* 36
3.2.1 Codificação da solução 36
3.2.2 Solução Inicial 36
3.2.3 Avaliação da solução 37
3.2.4 Cálculo da temperatura inicial 38
3.2.5 Definição das vizinhanças 39

4 RESULTADOS 41
4.1 Dados e parâmetros utilizados 41
4.2 Resultados - solução via modelagem matemática 44
4.3 Resultados - solução via meta-heurística *Simulated Annealing* 46

4.3.1 Resultados para diferentes taxas de resfriamento 46
4.3.2 Resultados para diferentes quantidades de iterações 48

5 CONCLUSÕES 52
5.1 Trabalhos Futuros 53

REFERÊNCIAS 54

1 INTRODUÇÃO

A utilização dos recursos disponíveis de maneira otimizada está se tornando cada vez mais imprescindível para as indústrias, uma vez que a melhoria dos processos, a redução dos custos e o aumento da qualidade do nível de serviço vêm se mostrando elementos decisivos para a consolidação de uma empresa no mercado. Para alcançar uma melhoria no nível de serviço de atendimento aos seus clientes, é natural que empresas procurem a otimização dos processos da cadeia produtiva, a fim de atingir um nível de excelência em suas operações. Variadas técnicas de otimização têm sido desenvolvidas e empregadas em indústrias com o intuito de reduzir ao máximo os custos de produção ou até mesmo aumentar o rendimento de processos produtivos (BOWERSOX; CLOSS, 2011).

Mudanças nas demandas do mercado têm sido cada vez mais imprevisíveis e abruptas nos últimos anos. No entanto, com os desdobramentos das transformações geradas pela quarta revolução industrial (indústria 4.0) é importante ter consciência acerca da velocidade da revolução tecnológica e de seu impacto na sociedade de forma geral. É necessário pensar de forma estratégica sobre as tendências de inovação que moldam nosso futuro. O conhecimento compartilhado facilita o alinhamento de objetivos e valores e bens comuns. A capacidade de reagir a essas mudanças nos sistemas de manufatura tornou-se um tópico de interesse para muitos pesquisadores no campo da tecnologia de fabricação. Como a flexibilidade está se tornando mais importante, um sistema de manufatura flexível (FMS - *flexible manufacturing system*) desempenha um papel significativo, especialmente na era da indústria 4.0 (MACDOUGALL, 2014; LU, 2017; THOBEN; WIESNER; WUEST, 2017). E visualizando combinar a eficiência em alta escala dos sistemas de produção automatizados com a flexibilidade de processos manuais, atualmente as indústrias têm feito o uso de novas tecnologias de automação assistida e robôs móveis autônomos (AMR - *Autonomous Mobile Robot*) (DANG et al., 2013).

Um FMS consiste em estações de trabalho onde várias operações como usinagem, inspeção e montagem são realizadas. Além de contar com sistemas de manuseio automatizado de materiais, dispositivos que transportam componentes entre estações de trabalho, sistema de armazenamento e sistema de controle por computador para coordenar as atividades das estações de trabalho e sistema de manuseio (SHIVANAND; BENAL; KOTI, 2006). Os manuseios de materiais no FMS são frequentemente realizados por veículos guiados automatizados (AGV - *Automated Guided Vehicle*) ou, em alguns casos, por AMR com ou sem braços montados em suas plataformas móveis.

Os AMR são capazes de se movimentar no local de trabalho para transportar componentes entre estações de trabalho e podem executar várias tarefas de valor agregado, incluindo manutenção de máquinas, pré-montagem e inspeção de qualidade em diferentes estações de trabalho.

Por ser um sistema caro de ser implementado seu uso deve ser feito de melhor forma possível. Neste sentido, é conveniente usar a programação (*scheduling*) da produção nos AMR, ou seja, designar onde (em que máquinas) e por quem (quais AMR) as tarefas serão executadas, em que sequência e em que instante de tempo devem ser iniciadas e finalizadas. Uma programação eficiente determina a quantidade mínima de robôs necessários para executar todas as tarefas em tempo mínimo, por exemplo, ou ainda, indica qual a sequência de tarefas deve ser realizada para que o tempo total de execução das mesmas seja o menor possível, e atendendo às restrições operacionais.

O escalonamento de tarefas é um fator crucial para a eficiência produtiva, uma vez que está ligado ao planejamento estratégico da empresa e por ser capaz de reduzir estoques, melhorar datas de entrega e resolver gargalos do sistema. Este problema de sequenciamento de diferentes tarefas nas máquinas e nos robôs possui algumas semelhanças com o problema do agendamento simultâneo de máquinas, pois seus subproblemas, o escalonamento de máquinas e o agendamento de AMR's, se enquadram no conjunto dos problemas NP-Difíceis, ou seja, problemas para os quais não existem algoritmos polinomiais que obtenham as soluções ótimas (FISCHETTI; MARTELLO; TOTH, 1987). Portanto, é necessária a utilização de métodos heurísticos para resolver de forma satisfatória problemas de grande porte (GOLDBERG, 1989; NIELSEN et al., 2014; DANG; NGUYEN; RUDOVá, 2019).

Este trabalho utiliza como base o modelo descrito por Dang et al. (2013), sendo que os dados e a análise das restrições consideradas foram utilizados como base para as etapas seguintes, que consistem na construção do modelo matemático e na identificação do método de resolução a ser implementado. Inicialmente um modelo de programação linear inteira mista (PLIM) foi implementado em linguagem de programação A Mathematical Programming Language (AMPL) utilizando o solver GUROBI na plataforma Neos, para problemas de pequeno porte. Além disso, a complexidade do problema aumenta rapidamente quando o AMR precisa alimentar mais máquinas ou quando o número de tarefas aumenta ou quando se considera um longo horizonte de planejamento. O foco deste trabalho é o desenvolvimento de um método computacional baseado na meta-heurística *Simulated Annealing* para resolver esse problema. Para avaliar a performance do método proposto, comparou-se os resultados encontrados pelo PLIM e a meta-heurística.

1.1 OBJETIVOS

Para resolver a problemática do sequenciamento e agendamento de tarefas de AMR em uma planta industrial, propõe-se neste trabalho os seguintes objetivos.

1.1.1 Objetivo Geral

Desenvolver um método computacional baseado na meta-heurística *Simulated Annealing* para determinar a sequência de realização de tarefas do AMR de melhor *makespan* possível.

1.1.2 Objetivos Específicos

Para alcançar o objetivo geral proposto, tem-se os seguintes objetivos específicos:

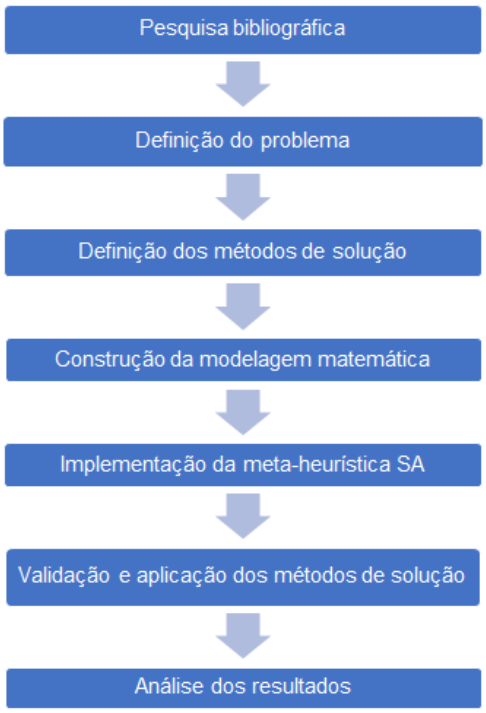
- Apresentar um modelo de programação linear inteira mista para determinar a sequência ótima exata de realização de tarefas para problemas de pequeno porte;
- Indicar modos de codificação para a aplicação de uma meta-heurística;
- Identificar métodos de busca em vizinhança para realização de buscas locais;
- Determinar a sequência de realização de tarefas para minimizar o tempo total de viagem;
- Identificar as restrições práticas necessárias para a modelagem (capacidade do robô, janela de tempo);
- Analisar o impacto das alterações da capacidade de carga do AMR e da janela de tempo no cálculo *mkespan*;
- Aplicar os modelos propostos nos dados do Estudo de Caso proposto em (Dang, 2014);
- Analisar os resultados obtidos com os resultados apresentados em (Dang, 2014).

1.2 METODOLOGIA

Nesta seção é apresentado o passo a passo usado neste trabalho para alcançar os objetivos propostos. Os conceitos a seguir estão propostos de forma cronológica para melhor compreensão do trabalho.

A metodologia adotada neste trabalho pode ser classificada como uma pesquisa experimental, que, segundo (GIL, 2008) consiste, essencialmente, em determinar um problema de estudo, selecionar as variáveis capazes de influenciá-lo e definir os métodos de controle e solução. Na Figura 1 ilustra-se as etapas da metodologia utilizada.

Figura 1 – Etapas do desenvolvimento do trabalho



Fonte: Autor (2021)

Inicialmente, realiza-se a pesquisa bibliográfica sobre os temas que construiram a fundamentação teórica do trabalho. Os principais temas abordados no referencial são: problemas de *scheduling*, modelos de sequenciamento de tarefas e de máquinas, e os métodos de solução, tanto os exatos quanto os aproximados.

Na etapa seguinte, define-se o problema considerado, baseado em (DANG et al., 2013), os dados usados por Dang et al. (2013) foram utilizados como base para as próximas etapas, que consistem na definição dos métodos de solução.

O primeiro método de solução definido foi o modelo matemático exato. O modelo foi implementado em linguamgem de programação AMPL (*A Mathematical Programming Language*) utilizando o solver Gurobi. O trabalho propõe uma nova abordagem matemática para o problema do sequenciamento de tarefas com janela de tempo restrita, variando em relação ao momento que a tarefa antecessora é realizada.

Em seguida, é proposto uma solução heurística baseada no método *Simulated Annealing*, visando a resolução de problemas de grande porte. O modelo foi validado comparando os seus resultados com os resultados do modelo exato. Por fim, analisa-se os resultados obtidos, comparando com outros métodos existentes na literatura. No próximo capítulo inicia-se a descrição da Etapa 1.

1.3 ESTRUTURA DO TRABALHO

Visando alcançar os objetivos propostos, este trabalho está organizado em cinco capítulos sequencialmente estruturados para melhor compreensão do problema e sua respectiva solução.

No Capítulo 2 apresenta-se os conceitos inerentes aos problemas de sequenciamento e agendamento de tarefa com janela de tempo restrita, principais bases teóricas do modelo utilizado, assim como as respectivas modelagens matemáticas. A capacidade de aplicação de AMR e outras tecnologias similares das indústrias na atualidade é apresentado, finalizando a fundamentação teórica do trabalho.

No Capítulo 3 expõe-se a descrição dos métodos de solução propostos. Explicando-se todas as restrições e função objetivo do modelo matemático, assim como a descrição do método heurístico proposto.

No Capítulo 4, apresenta-se a descrição dos dados utilizados, seguido da validação do modelo de solução exata. Também é reservado à apresentação dos resultados obtidos na aplicação do cenário típico e do método heurístico proposto, bem como as análises pertinentes.

As conclusões e considerações finais obtidas da aplicação dos métodos de solução propostos são apresentadas no Capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresenta-se conceitos utilizados como base no desenvolvimento do trabalho, a respeito do problema de sequenciamento e agendamento de tarefas, bem como do método utilizado para o desenvolvimento do estudo de caso. A seguir será discutido o problema de sequenciamento de tarefas.

2.1 PROBLEMA DE SEQUENCIAMENTO

O Problema de Sequenciamento (também conhecido como *Scheduling Problem*) é um problema de otimização. De acordo com Pape (2015), é dividido em três partes:

- Designação: define quem será o responsável por realizar cada tarefa;
- Sequenciamento: indica em que ordem será realizada cada tarefa; e
- Agendamento ou escalonamento: decide quando será realizada cada tarefa, a fim de satisfazer restrições de tempo e recursos.

Para Hochbaum (1999), os objetivos em problemas de sequenciamento são:

- Minimizar o tempo de execução total, medindo o nível de utilização das máquinas;
- Minimizar o tempo de espera de cada tarefa, que se trata do tempo entre a finalização de uma tarefa até o início da próxima; e
- Minimizar os custos de execução das atividades.

Segundo Pinedo (2008), os problemas de sequenciamento contam com números finitos de tarefas e máquinas. O problema de escalonamento atua na alocação de recursos para a realização de tarefas em uma janela de tempo, tratando-se de um processo de otimização de um ou mais objetivos. Esses objetivos podem ser diversos, como a redução do tempo total de realização de um determinado conjunto de tarefas, ou a redução do número de tarefas realizadas com atraso.

Para Tubino (2007), é função do escalonamento buscar a adequação dos limitados recursos de produção, sejam eles mão de obra, máquinas, instalações ou materiais. Tal adequação dependerá do grau de detalhamento e intensidade que se dá à execução das funções de escalonamento no sistema produtivo em que se encontra inserido.

Segundo Arnold (1998), o *scheduling* estabelece a ordem em que as tarefas são realizadas e as datas de começo e fim de cada operação. Para isso, o programador precisa das informações de roteiro, capacidade produtiva, capacidade de carregamento e tempo de processamento dos centros de trabalho envolvidos.

De acordo com Graves (1981), em *scheduling* o tempo de processamento é formado por cinco elementos:

- Tempo em fila: tempo em que um trabalho aguarda para ser iniciado em um centro de produção;
- Tempo de *setup*: tempo necessário para preparar um centro de trabalho para executar a operação;
- Tempo de corrida: tempo em que a operação é realizada;
- Tempo de espera: tempo de aguardo pelo material para ser transferido ao próximo centro de trabalho; e
- Tempo de movimentação: tempo de trânsito entre os centros de trabalho.

Os objetivos do *scheduling* geralmente estão associados a regras de prioridades. Visa-se selecionar as tarefas que terão prioridade de processamento, alocando assim os recursos de maneira a executar essas tarefas.

Para Joo e Kim (2015), a etapa de sequenciamento de tarefas é de suma importância para o planejamento da produção, uma vez que se trata de uma série de atividades que viabilizam os recursos necessários para a conclusão das tarefas, determinando data e hora para a produção.

Segundo Graves (1981), três informações básicas devem ser consideradas na classificação de problemas de escalonamento de produção:

- Necessidade de produção;
- Complexidade de processamento;
- Critério para escalonamento das tarefas.

Para Santos (1994), a necessidade de produção é determinada diretamente pelas ordens de pedido ou indiretamente pelas ordens de reabastecimento de estoques. Dependendo do modo que os pedidos surgem, pode ser dividida em:

- Produção direta ("Open shop"): onde todas as ordens de produção são diretas do mercado; ou
- Produção estocada ("Closed shop"): as ordens são indiretas, frutos da necessidade de manutenção de estoques.

Ainda segundo Santos (1994), um problema de produção direta resulta em um problema de sequenciamento de tarefas, no qual as ordens de produção são sequenciadas nos equipamentos. No problema de produção estocada estão envolvidas não só decisões de sequenciamento, mas também decisões de controle de estoque.

A complexidade de processamento está associada ao número de processadores envolvidos no processo, assim como a estrutura da planta. Barbosa (2017) realça que a complexidade de um sistema de sequenciamento de produção se dá pela dinâmica dos processos produtivos, pela chegada de pedidos inesperados, flexibilidade ou alteração nas prioridades de entrega e pelas taxas de defeitos aleatórias.

De acordo com Pinedo (2008), o sequenciamento é um processo de tomada de decisões que visa definir a melhor sequência de processamento de tarefas, sujeito a restrições de recursos ao longo do tempo. A classificação de problemas de escalonamento encontra-se ligada ao critério utilizado para avaliar o desempenho do roteiro de fabricação, pela estrutura de processamento e pelas regras de produção que governam o processo. Dentre os critérios de desempenho de sistema estão (REKLAITIS, 1982):

- *Makespan*: tempo total da programação de tarefas;
- *Mean flow-time*: média de tempo de duração do fluxo;
- *Total flow-time*: tempo total de duração do fluxo;
- *Mean tardiness*: atraso médio do fluxo de tarefas;
- *Maximum tardiness*: atraso máximo para a conclusão de tarefas;
- *Tardiness*: soma das penalidades por atraso;
- *Earliness*: soma das penalidades por adiantamento.

Decisões acerca da ordenação das tarefas no sistema devem ser tomadas a partir do momento em que as mesmas tornam-se aptas ao processamento. Tais decisões podem se tornar complexas, gerando lentidão na tomada de decisão (SLACK; CHAMBERS; JOHNSTON, 2002).

Na prática, o problema de sequenciamento de tarefas está relacionado diretamente ao sequenciamento das máquinas, visto que a ordem dos processos está intimamente ligada ao funcionamento da linha de produção. Sendo assim, a seguir será abordado o sequenciamento de máquinas.

2.2 SEQUENCIAMENTO EM MÁQUINAS

De acordo com Baker (1974), o sequenciamento de máquinas estabelece a ordem em que a produção será realizada, respeitando prioridades e restrições impostas pelo processo industrial. Para classificar os modelos de sequenciamento é necessário caracterizar os dados sobre os recursos e o comportamento das operações. Pode haver um ou vários tipos de recursos, que podem ser especializados (executa um determinado tipo de operação) ou paralelos (diversos recursos executando uma mesma operação).

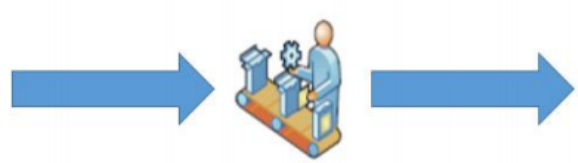
Pinedo (2008) classificou o sequenciamento de máquinas conforme as características das máquinas utilizadas e o ambiente de produção. Os possíveis ambientes de máquinas considerados por Pinedo (2008) estão apresentados a seguir.

2.2.1 Modelo de Máquina Única

Para Pinedo (2008), o Modelo de Máquina Única é o mais simples quando comparado aos demais. Neste modelo, todos os produtos são processados em uma

única máquina especializada. Considera-se a existência de apenas um recurso, podendo ser um único equipamento, uma célula de produção ou um conjunto de recursos modelados como um só. Na Figura 2 é apresentado um esquema deste tipo de modelo.

Figura 2 – Representação de processamento em máquina única



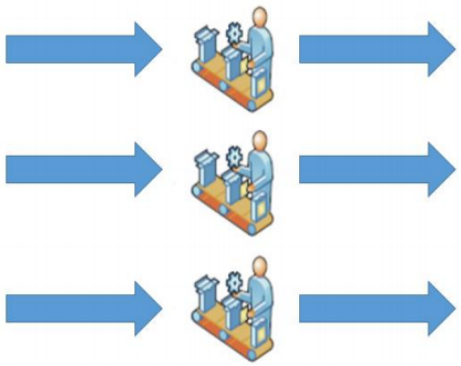
Fonte: Santo (2014)

A Figura 2 ilustra que todos os produtos que entram no ambiente deverão ser processados pela máquina, e somente por ela. Após o processamento, os produtos sairão do ambiente.

2.2.2 Modelo de Máquina em Paralelo

Conforme explicado na seção anterior, o Modelo de Máquinas em Paralelo envolve diversos recursos executando uma mesma operação. Pinedo (2008) ressalta que os tempos de processamento das máquinas não precisam ser iguais. No caso de máquinas idênticas, existem m máquinas e os produtos podem ser produzidos em qualquer máquina sem interferir no *makespan*. Já no caso de máquinas em paralelo com tempos de processamento distintos, a velocidade de processamento de cada máquina interfere diretamente no tempo total de processamento das tarefas, pois, quanto menor o tempo de processamento de uma máquina, maior será a tendência do modelo de alocar tarefas nela. Na Figura 3 é apresentado um exemplo deste modelo.

Figura 3 – Representação de processamento em máquinas paralelas



Fonte: Santo (2014)

Diferente do Modelo de Máquina Única, a Figura 3 ilustra a entrada de

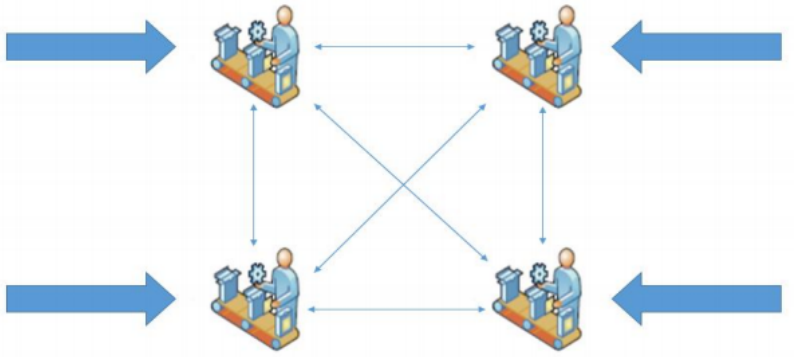
múltiplos produtos em diversas máquinas em paralelo. Assim, os produtos podem ser processados simultaneamente até sairem do ambiente.

2.2.3 Modelo de *Job Shop*

Segundo Pinedo (2008), no modelo *Job Shop* cada produto tem um roteiro pré-determinado em um processo com m máquinas. Os roteiros envolvem linhas de produção em que o produto visita uma sequência de máquinas uma única vez, ou linhas de produção cujas máquinas podem ser visitadas mais de uma vez.

Neste modelo, os produtos não precisam necessariamente passar por todas as máquinas, nem seguir a mesma sequência de produção, consistindo em um processamento de n peças em m máquinas, no qual o processamento de cada peça realiza operações em uma sequência específica, como na Figura 4.

Figura 4 – Representação de processamento em máquinas em Job Shop



Fonte: Santo (2014)

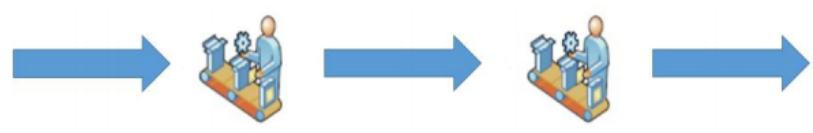
Como apresentado na Figura 4, o produto pode entrar no ambiente por qualquer uma das máquinas e seguir a sequência especificada pelo modelo.

2.2.4 Modelo *Flow Shop*

Pinedo (2008) explica que neste modelo existem m máquinas em série, e cada produto deve ser processado em cada uma das máquinas em uma mesma sequência pré-determinada. Após completar o processamento em uma máquina, os produtos seguem, geralmente, uma fila no esquema *First In First Out* para aguardar o início do processamento na máquina seguinte.

Trata-se de um caso particular de *Job Shop*, em que todos os processos possuem uma mesma sequência tecnológica de produção, com várias operações em série. Cada operação deve ser processada em todas as máquinas da linha em uma ordem pré-estabelecida, como na Figura 5.

Figura 5 – Representação de processamento em máquinas em *Flow Shop*



Fonte: Santo (2014)

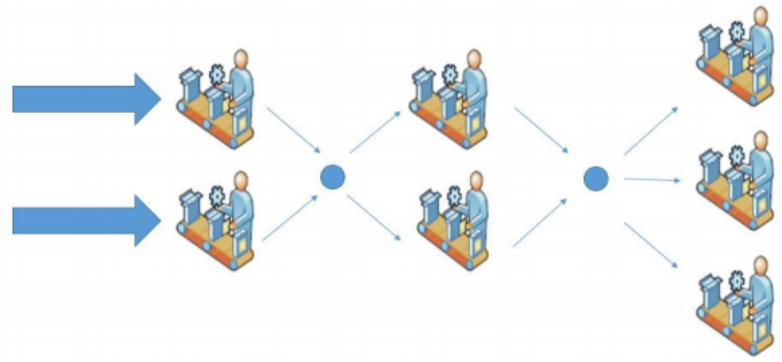
A Figura 5 ilustra duas máquinas em série. O produto entra no ambiente sendo processado pela primeira máquina, seguindo para a segunda, e saindo do ambiente ao final da sequência.

2.2.5 Modelo de *Flexible Flow Shop*

Ainda de acordo com o Pinedo (2008), o Modelo *Flexible Flow Shop* é uma generalização do Modelo *Flow Shop* e dos ambientes de máquinas em paralelo, porém, ao invés de m máquinas em série, existem c etapas em série e, em cada etapa, há máquinas idênticas em paralelo.

As filas entre os estágios não precisam necessariamente ser processadas de acordo com o esquema de *First In First Out*. A Figura 6 apresenta um exemplo deste modelo.

Figura 6 – Representação de processamento em máquinas em *Flexible Flow Shop*



Fonte: Santo (2014)

Na Figura 6, os círculos representam os pontos intermediários entre as etapas do modelo. Neste modelo, o produto segue uma sequência, como no Modelo *Flow Shop*, porém cada etapa pode conter máquinas em paralelo, como no Modelo de Máquinas em Paralelo.

O problema considerado nesse trabalho, o SAMR (*Sechduling a single AMR*) , considera um ambiente de FMS com sequenciamento de tarefas e um único robô em máquinas em paralelo. Após a identificação do problema e da definição do modelo a ser utilizado, deve-se pensar no método de solução a ser proposto. Este tópico será abordado na próxima seção.

2.3 SEQUENCIAMENTO DE TAREFAS E ROBÔS AUTÔNOMOS

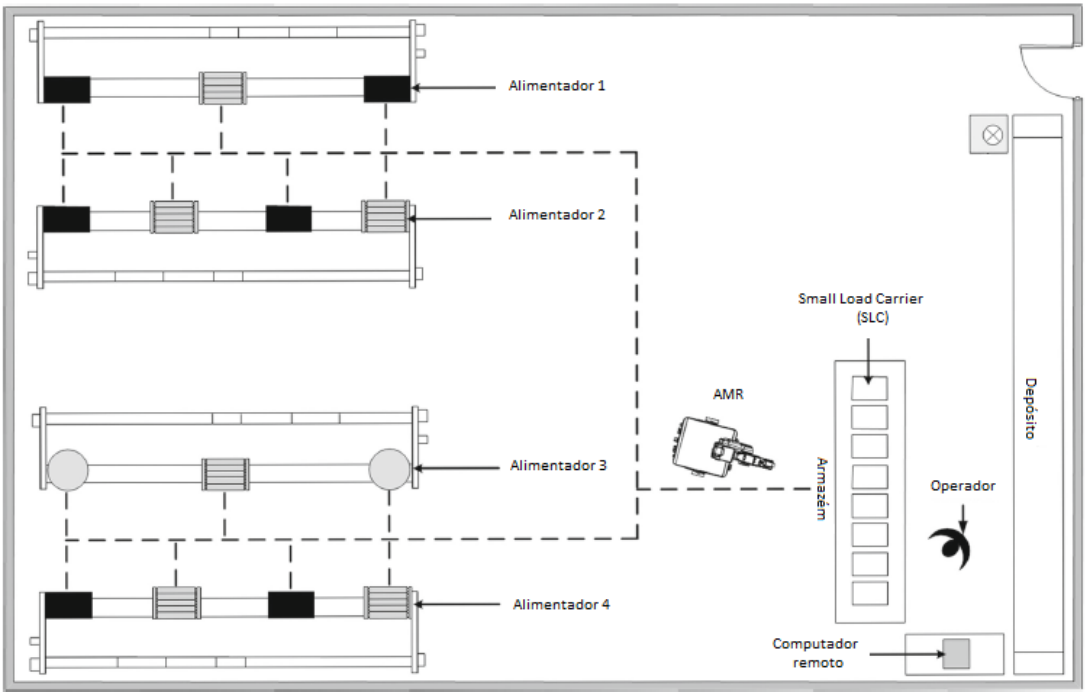
Essa seção é dedicada a definir o modelo estudado por Dang et al. (2013) e descrever a modelagem matemática utilizada para a resolução do problema.

2.3.1 Definição do problema

O problema estudado considera robôs autônomos com braços mecânicos que realizam tarefas em uma linha de produção com máquina em paralelo, transportando, coletando e esvaziando containers de peças no local desejado. Os AMRs devem ser programados para impedir a falta de peças na linha de produção. Para determinação da janela de tempo para realização das tarefas de alimentação pelos AMRs, foi utilizado um método baseado no sistema de inventário de estoque (s, Q), que leva em consideração as características das máquinas. Neste método, quando a quantidade de peças em uma máquina alcança um nível s, um pedido de reposição Q é lançado. Além disso, a capacidade dos robôs também foi levada em consideração. A função objetivo é minimizar o tempo total de viagem de um AMR para um determinado horizonte de planejamento.

O problema apresentado por Dang et al. (2013) trata de uma linha de produção de uma fábrica, representada na Figura 7 que produz impulsores de compressão. Esta linha de produção consiste em um armazém e quatro máquinas que devem ter seus alimentadores servidos pelo AMR. Os alimentadores são responsáveis por suprir automaticamente peças às máquinas. Além disso, alimentadores diferentes são carregados por diferentes tipos de peças.

Figura 7 – Layout da célula da linha de produção



Fonte: Dang et al. (2013)

No caso estudado pelos autores, antes de designar o robô ao ambiente de trabalho, a área da fábrica foi dividida em células de manufatura com uma ou mais linhas de produção, que consistem em diversas máquinas. As tarefas de posicionamento dos paletes ou caixas ao lado das máquinas e de alimentação das peças para os alimentadores são responsabilidade dos AMRs. Para atender aos requisitos declarados Dang et al. (2013) implementou o conceito do bartender.

Neste conceito, todo alimentador é atribuído às seguintes características: nível máximo e mínimo de peças, e uma taxa de alimentação de peças na máquina. Além disso, um operador (bartender) é responsável por colocar as peças em pequenos carregamentos padronizados (SLC - *Small Loaded Carries*) , que ficam localizados no armazém central (o bar), responsável por reunir diferentes peças em uma única área.Durante a operação, o robô pega um ou mais SLCs do armazém, viaja até uma máquina, esvazia todas as peças no alimentador, retorna ao armazém para descarregar todos os SLCs vazios e então carregar novos SLCs e recomeçar a operação.

2.3.2 Modelagem Matemática para o SAMR

A modelagem matemática proposta por Dang et al. (2013) considera que todas as tarefas de alimentação correspondentes às entregas de SLC são conhecidas antecipadamente. Um método para evitar a falta de peças durante o *lead time* de reposição foi baseado no sistema de controle de estoque (s,Q) e implementado para definir a janela de tempo para a realização das tarefas de alimentação de peças. O

AMR é posicionado inicialmente no armazém e seu limite de carregamento é definido. Além disso, outras suposições também foram levadas em consideração:

- Todas as tarefas são periódicas, independentes e atribuídas ao mesmo AMR;
- O tempo de trabalho e de viagem do AMR entre qualquer par de localidades são conhecidas;
- A taxa de alimentação de peças do alimentador para máquina é conhecida e constante;
- Todos os alimentadores devem ser carregados até o volume máximo de peças.

Algumas das variáveis de decisão do modelo PLIM foram restritas apenas a valores inteiros, o que torna o problema de otimização muitos mais difícil de ser resolvido. O tempo de solução aumentam exponencialmente conforme o tamanho do problema aumenta. Portanto, o modelo só pode ser aplicado na prática em pequenas escalas, com poucos alimentadores na linha de produção e um horizonte de planejamento curto. Para estes cenários, o modelo PLIM vai gerar a solução ótima, o que pode ser usado como parâmetro de comparação para a qualificação do método heurístico a ser implementado.

Os parâmetros de entrada do modelo possuem as seguintes notações:

- N: grupo de todas as tarefas ($N = 0, 1, 2, ..., n$, onde 0 é a tarefa no armazém);
- n_i: número de vezes que a tarefa i tem que ser executada;
- R: grupo de todas as possíveis rotas ($R = 1, 2, ..., Rmax$; $Rmax = \sum n_i$,$\forall i \in N \setminus$ {0});
- e_{ik}: horário de liberação da subtarefa k da tarefa i;
- d_{ik}: prazo limite da subtarefa k da tarefa i;
- p_i: tempo de periodicidade da tarefa i;
- w_i: tempo de processamento do AMR na localização i;
- t_{ij}: tempo de viagem do AMR da localização i até a localização j;
- c_i: taxa de alimentação de peça do alimentador i para máquina;
- v_i: volume mínimo de peças no alimentador i;
- u_i: volume máximo de peças no alimentador i;
- Q_m: número máximo de SLC que o AMR consegue carregar;
- T: horizonte de planejamento.

As variáveis de decisão usadas no modelo são:

- $X_{ik}^{jlr} = \begin{cases} 1 & \text{, se o AMR viaja da localização da tarefa } i\text{, subtarefa } k\text{, para a} \\ & \text{localização da tarefa } j\text{, subtarefa } l\text{, na rota } r; \\ 0 & \text{, caso contrário.} \end{cases}$
- Y_{ik}= índice da rota cuja a subtarefa k da tarefa i pertence;
- S_{ik}= tempo de início da subtarefa k da tarefa i.

Devido às características periódicas das tarefas i nos alimentadores, cuja

periodicidade é calculada no conjunto de Equações 1, um número de subtarefas deve ser realizado. Essa quantidade de subtarefas de cada tarefa i é definida como: $n_i = [T/p_i]$. O AMR deve começar o processamento da subtarefa k da tarefa i dentro da janela de tempo associada. Ou seja, o robô não pode chegar ao alimentador depois do limite de tempo superior, e, se chegar antes do limite de tempo inferior, deverá esperar para começar a tarefa. O limite inferior da janela de tempo, ou o horário de liberação, é definido como o tempo em que o número de peças dentro do alimentador cai a um certo nível v_i, o conjunto de Equações 2. Já o limite superior da janela de tempo, ou o prazo limite da subtarefa k da tarefa i, é caracterizado como o tempo em que não há mais nenhuma peça no alimentador, Equações 3.

$$p_i = (u_i - v_i) * c_i, \forall i \in N \backslash \{0\} \tag{1}$$

$$e_{ik} = e_{(ik-1)} + p_i, \forall i \in N \backslash \{0\}, k \in 1, 2, \ldots, n_i, e_{i0} = 0 \tag{2}$$

$$d_{ik} = e_{ik} + (v_i * c_i), \forall i \in N \backslash \{0\}, k = 1, 2, \ldots, n_i \tag{3}$$

Como o tempo de trabalho do robô em cada máquina e a quantidade de vezes que uma tarefa deve ser repetida na máquina são parâmetros pré-estabelecidos do problema, a função objetivo leva em consideração apenas o tempo gasto viajando entre as máquinas para a realização das tarefas.

$$MinZ = \sum_{i \in N} \sum_{k=1}^{n_i} \sum_{j \in N} \sum_{l=1}^{n_j} \sum_{r \in R} t_{ij} * X_{ik}^{jlr} \tag{4}$$

Os conjuntos de restrições usados no problema são descritos e explicados como:

$$e_{ik} < S_{ik} < d_{ik}, \forall i \in N \backslash \{0\}, k \in 1, 2, \ldots, n_i \tag{5}$$

$$\sum_{j \in N \backslash \{0\}} \sum_{l=1}^{n_j} X_{01}^{jl1} = 1 \tag{6}$$

$$\sum_{j \in N \backslash \{0\}} \sum_{l=1}^{n_j} \sum_{r \in R} X_{01}^{jlr} \leq 1 \tag{7}$$

O conjundo de Restrições 5 garante que o tempo de início de processamento da subtarefa k da tarefa i respeite os limites inferior e superior da janela de tempo. As Restrições 6 garantem que o robô comece no armazém no estágio inicial na primeira

rota, e o conjunto de Restrições 7 garante que o estágio incial no armazém seja realizado uma única vez.

$$\sum_{(i,k),(j,l)Z} X_{ik}^{jlr} \leq |Z|-1, \forall r \in R, \forall Z \in Z_T, Z_T = \{(i,k)|i \in N\{0\}, k \in 1,2,\ldots,n_i\} \quad (8)$$

O grupo de Restrições 8 elimina as subrotas entre subtarefas de tarefas, no qual Z é um subconjunto de Z_T, sendo que Z_T é o conjunto de todas as subtarefas de tarefas nos alimentadores e no armazém.

$$\sum_{j \in N} \sum_{l=1}^{n_j} \sum_{r \in R} X_{ik}^{jlr} = 1, \forall i \in N\backslash\{0\}, k \in 1,2,\ldots,n_i \quad (9)$$

$$\sum_{i \in N} \sum_{k=1}^{n_i} \sum_{r \in R} X_{ik}^{jlr} = 1, \forall j \in N\backslash\{0\}, k \in 1,2,\ldots,n_j \quad (10)$$

$$\sum_{i \in N} \sum_{k=1}^{n_i} \sum_{j \in N\backslash\{0\}} \sum_{l=1}^{n_j} X_{ik}^{jlr} \leq Q_m, \forall r \in R \quad (11)$$

Os conjuntos de Equações 9 e 10 garantem que apenas uma subtarefa de uma tarefa será realizada imediatamente após e imediatamente antes da subtarefa k da tarefa i, respectivamente. As Equações 11 garantem que o AMR não carregará, por rota, mais SLC que o permitido por sua capacidade.

$$s_{ik} + (w_i + t_{ij} \sum_{r \in R} X_{ik}^{jlr}) - L(1 - \sum_{r \in R} X_{ik}^{jlr}) + (y_{jl} - y_{ik})(t_{i0} + w_0 + t_{0j} - t_{ij}) \leq s_{jl} \quad (12)$$

$$\forall i,j \in N, k \in 1,2,\ldots,n_i, l \in 1,2,\ldots,n_j$$

$$y_{jl} = \sum_{i \in N} \sum_{k=1}^{n_i} \sum_{r \in R} r * X_{ik}^{jlr}, \forall j \in N\backslash\{0\}, l \in 1,2,\ldots,n_j \quad (13)$$

$$y_{jl} \geq y_{ik} \sum_{r \in R} X_{ik}^{jlr}, \forall i,j \in N, k \in 1,2,\ldots,n_i, l \in 1,2,\ldots,n_j \quad (14)$$

As Restrições 12 asseguram que os requisitos de tempo de viagem sejam obedecidos. Caso duas subtarefas, de tarefas iguais ou não, sejam conectadas porém em rotas diferentes, o robô deve visitar o armazém para descarregar os SLCs vazios e carregar os cheios. Ainda, a mesma restrição, garante que o tempo de início da subtarefa k da tarefa i não seja maior que o da subtarefa l da tarefa j, caso elas ocorram em sequencia. L é um número muito grande. O conjunto de Restrições 13 atribui uma subtarefa de uma tarefa a uma rota e as Restrições 14 garantem que os

índices das rotas utilizadas aumentem sequencialmente. Por fim as restrições 15 e 16 indicam os domínios das variáveis.

$$X_{ik}^{jlr} \in \{0,1\}, \forall r \in R, \forall i,j \in N, k \in 1,2,\ldots,n_i, l \in 1,2,\ldots,n_j \quad (15)$$

$$y_{ik} : \text{variável inteira positiva} \forall i \in N \backslash \{0\}, k \in 1,2,\ldots,n_i \quad (16)$$

2.4 COMPLEXIDADE DO SAMR

No âmbito da PO, modelos exatos fornecem a resposta ótima para problemas de otimização, porém este tipo de modelagem não pode ser aplicada a todos os problemas.Alguns problemas são tão complicados que pode não ser possível encontrar uma solução ótima em tempo polinomial. Nessas situações, ainda é importante encontrar uma boa solução viável, que seja pelo menos razoavelmente próxima da solução ótima (HILLIER; LIEBERMAN, 2013).

Portanto, os diferentes problemas de otimização possuem diferentes níveis de complexidade. A complexidade de um problema pode ser medida principalmente pelo tempo computacional necessário para se obter uma solução, ou pela quantidade de memória necessária. Contudo, podem ser utilizados outros critérios de desempenho, os quais o analista considere relevante (COLIN, 2013). Os problemas mais comuns se dividem em duas grandes classes: Problemas Polinomiais (P) e Não Polinomiais (NP). Os problemas da classe P são aqueles que possuem solução em tempo polinomial em função da quantidade de entrada de dados. Já os problemas da classe NP são os que não possuem solução em tempo polinomial.

Dentre os mais difíceis problemas da classe NP, estão os problemas NP-Hard e NP-Completos. De acordo com Goldbarg e Luna (2005), os problemas da classe NP possuem um elevado número de combianções possíveis, o que torna os métodos de enumeração de soluções inviáveis. Desta forma, devido à grande explosão na quantidade de combinações ao se acrescentar variáveis, o tempo de solução desta classe de problemas é não polinomial (GOLDBARG; LUNA, 2005). Para ambos os casos o tempo de solução de um algoritmo é definido pelo tempo necessário para solucionar o problema dada uma entrada de dados de tamanho n, no pior caso (COLIN, 2013). Em geral, problemas inclusos nestas classes, possuem elevado tempo de processamento. Devido a este fato, a espera pela solução ótima torna-se inviável.

O problema do SAMR pode ser apresentado como um problema da classe NP-Hard, devido à sua natureza combinatorial. Em seu trabalho, Dang et al. (2013) demonstram os motivos para o problema se comportar deste modo.

2.5 MÉTODOS DE SOLUÇÃO

Apesar dos problemas de programação linear buscarem a otimização, em determinadas situações, dependendo de suas características, podem trazer complexidades que exigem uma abordagem de solução aproximada, como explicado na seção anterior.

2.5.1 Solução via Programação Matemática

A solução via programação matemática busca, por meio de métodos exatos, a resolução de modelos matemáticos. Dentre os métodos que podem ser utilizados, destacam-se o Método *Simplex* e o Método *Branch-and-Bound*, para problemas lineares contínuos e inteiros, respectivamente.

Os métodos exatos determinam a solução ótima dos modelos de programação matemática que representam os problemas e necessitam de *softwares* específicos de otimização, que dependem da estrutura do modelo considerado. Nesse contexto, com avanços computacionais e desenvolvimento de máquinas de alta performance, tornou-se possível resolver problemas reais com grande número de variáveis, o que antes não era possível utilizando métodos exatos.

Devido ao alto tempo de processamento necessário nos métodos de programação matemática e à complexidade computacional demandada pelos problemas de sequenciamento, os estudos realizados para a resolução do problema de sequenciamento de tarefas em máquinas, utilizando o método exato, consideram apenas casos pequenos com poucas máquinas e poucas tarefas. Em geral, empregando linguagens específicas para modelagem matemática, como AMPL (2018) (*A Mathematical Programming Language*), é possível utilizar recursos computacionais (LINGO, GUROBI, CEPLEX, dentre outros) como ferramentas de solução de problemas gerais de otimização (ANAND; AGGARWAL; KUMAR, 2017).

Yang e Chern (1995) resolveram um problema de sequenciamento de tarefas com duas máquinas em ambiente *Flow Shop* com restrição de tempo de espera. Os autores consideraram um tempo limite, no qual o tempo entre máquinas, de cada trabalho, não poderia ser superior. Yang e Chern (1995) propuseram o uso do algoritmo de *Branch-and-Bound*, cujo o objetivo era minimizar o *makespan*, que é o tempo total que um produto passa na linha de produção.

Petrović, Miljković e Jokić (2019) analisaram o problema de *scheduling* para um único robô móvel, assim como neste trabalho. Contudo, Petrović, Miljković e Jokić (2019) propuseram um modelo matemático no qual o robô era responsável apenas pelo transporte de matérias-primas, mercadorias e peças, sem precisar realizar nenhuma tarefa de valor agregado. Os autores conseguiram obter resultados para cenários com até 14 tarefas.

Chen et al. (2020) estudaram a eficiência dos terminais de contêineres automatizados. Para isso, consideraram o problema de programação de AGV como um problema de coordenação e programação de multi-robôs, integrando os AGVs aos guindastes dos terminais. Utilizando solvers comerciais os resultados do estudo indicaram que a abordagem proposta poderia encontrar soluções de forma eficiente dentro de lacunas de otimalidade de 2

Riazi e Lennartson (2021) formularam um modelo matemático para resolver o problema de roteamento e programação de AGVs. Os autores consideraram instâncias pequenas e foram testados dois solvers diferentes, sendo um com restrições comerciais e um de código aberto. De acordo com os resultados, ambos os solucionadores de propósito geral podem resolver efetivamente os modelos propostos.

Como explicado anteriormente, os métodos de solução via programação matemática, apesar de serem capazes de encontrar a solução ótima, demandam muito tempo e esforço computacional quando se trata de problemas com elevado número de restrições e variáveis. A melhor maneira de se contornar este problema é com a utilização de heurísticas, que são capazes de encontrar soluções próximas da solução ótima em tempo viável para problemas de grande porte. A seguir será explicado mais sobre as soluções via métodos heurísticos.

2.5.2 Solução via Métodos Heurísticos

De acordo com Hillier e Lieberman (2013), heurísticas são procedimentos utilizados dentro da PO na busca de soluções viáveis para um problema. Estes métodos devem ser capazes de lidar com problemas de grande porte, muito embora não possuam garantia de otimalidade, sendo assim chamados de métodos aproximados.

Os métodos aproximados surgiram como meios de contornar os problemas de tempo de processamento dos métodos exatos, viabilizando a utilização da PO para soluções de problemas de grande porte. Seu uso tem sido muito disseminado, já que foram capazes de encontrar soluções muito próximas as soluções exatas, porém, demandando um esforço computacional muito menor (HILLIER; LIEBERMAN, 2013).

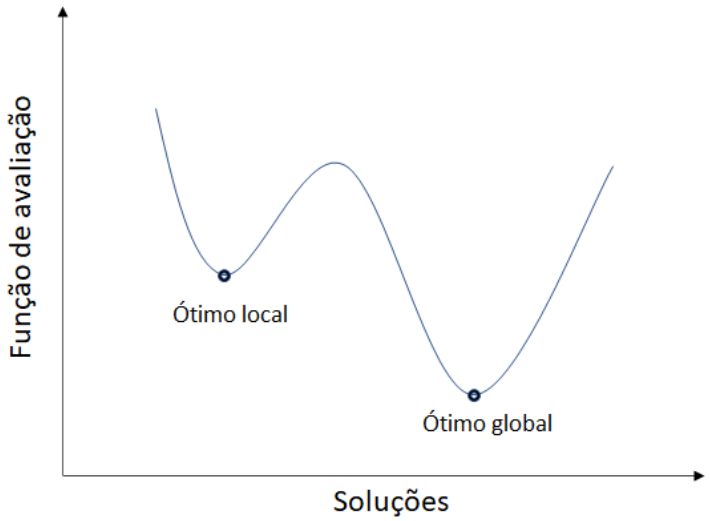
Segundo Bodin et al. (1983), as heurísticas podem ser classificadas em três categorias:

- Construtiva: método que busca construir a solução;
- Melhoria: tem por objetivo melhorar uma solução inicial; e
- Composta: combina as duas classificações anteriores, com objetivo de melhorar uma solução construída.

O principal problema do uso dos métodos heurísticos são os ótimos locais, que são conjuntos de valores para as variáveis do problema que maximizam, ou minimizam, a função objetivo em um subespaço do espaço de busca. Na Figura 8 é possível notar

a diferença entre o ótimo local e global. Com as meta-heurísticas, que são heurísticas melhoradas, criou-se estratégias de alto nível capaz de escapar dos ótimos locais e realizar uma busca consistente de soluções viáveis (HILLIER; LIEBERMAN, 2013).

Figura 8 – Ótimo Local e Global



Fonte: Autor (2021)

Alguns exemplos de heurísticas e meta-heurísticas, são: Vizinho mais Próximo; Algoritmo Genético; *Simulated Annealing*; Busca Tabu; Colônia de Formigas. Por serem métodos capazes de solucionar problemas de grande porte, são úteis para problemas reais. Diversos autores utilizaram das heurísticas e meta-heurísticas para a resolução de problemas de sequenciamento de tarefas em máquinas. Alguns deles são apresentados a seguir.

Dong et al. (2018) investigou o problema de sequenciamento para uma máquina com *job delivery* para múltiplos clientes. Neste problema, cada tarefa precisa ser processada por uma única máquina e, então, ser entregue por um único veículo ao cliente. O objetivo é minimizar o *makespan*, no qual todas as tarefas são processadas e entregues ao respectivo cliente, e o veículo retorna à máquina. Os autores propõem, para isso, um algoritmo *2-approximation* para casos gerais, e um algoritmo *5/3-approximation* para casos com apenas dois clientes.

Uma outra abordagem pode ser encontrada em Santos (2016), em que o objetivo é encontrar a otimização do sequenciamento de tarefas em cada máquina de tal maneira que os desgastes das máquinas sejam reduzidos, minimizando assim o tempo de conclusão de todas as tarefas. A autora utiliza algoritmos baseados nas meta-heurísticas *Iterated Local Search* (ILS) e *Iterated Greedy* (IG). Os algoritmos são comparados entre si e também com um algoritmo *Simulated Annealing* (SA).

Fidelis e Arroyo (2017) sugerem o uso de duas heurísticas *Adaptative Large Neighborhood Search* (ALNS) e *Simulated Annealing* (SA), com avaliação

de desempenho através de experimentos computacionais, comparando com duas heurísticas da literatura *Memetic Algorithm* e *Variable Neighbordhood Search*, para a resolução de problemas de sequenciamento de tarefas em máquinas paralelas idênticas de processamento em lotes. Um lote consiste em um grupo de tarefas que podem ser processadas simultaneamente na mesma máquina, a fim de determinar os lotes de tarefas a sequenciar nas máquinas de maneira a minimizar o atraso total ponderado. Diferente deste trabalho, Fidelis e Arroyo (2017) não consideraram um operador (um AMR, no caso deste trabalho), responsável por processar as tarefas e ter que viajar entre as máquinas.

Em Dang, Nguyen e Rudová (2019) foi explorado um problema de um FMS que contém robôs móveis autônomos capazes de realizar tarefas de valor agregado ao produto, além do transporte. A inovação deste trabalho foi a consideração de três sub-problemas que devem ser resolvidos, sendo eles: o sequenciamento de operações nas máquinas, a atribuição dos robôs para o transporte, e atribuição dos robôs para o processamento dos produtos. Os autores então desenvolveram uma heurística híbrida que combinou o Algoritmo Genético com a Busca Tabu para solucionar um problema de minimização do *makespan*.

A seguir será contextualizado e explicado a meta-heurística do *Simulated Annealing*, método de solução que foi escolido para resolver os cenários de grande porte do problema estudado neste trabalho.

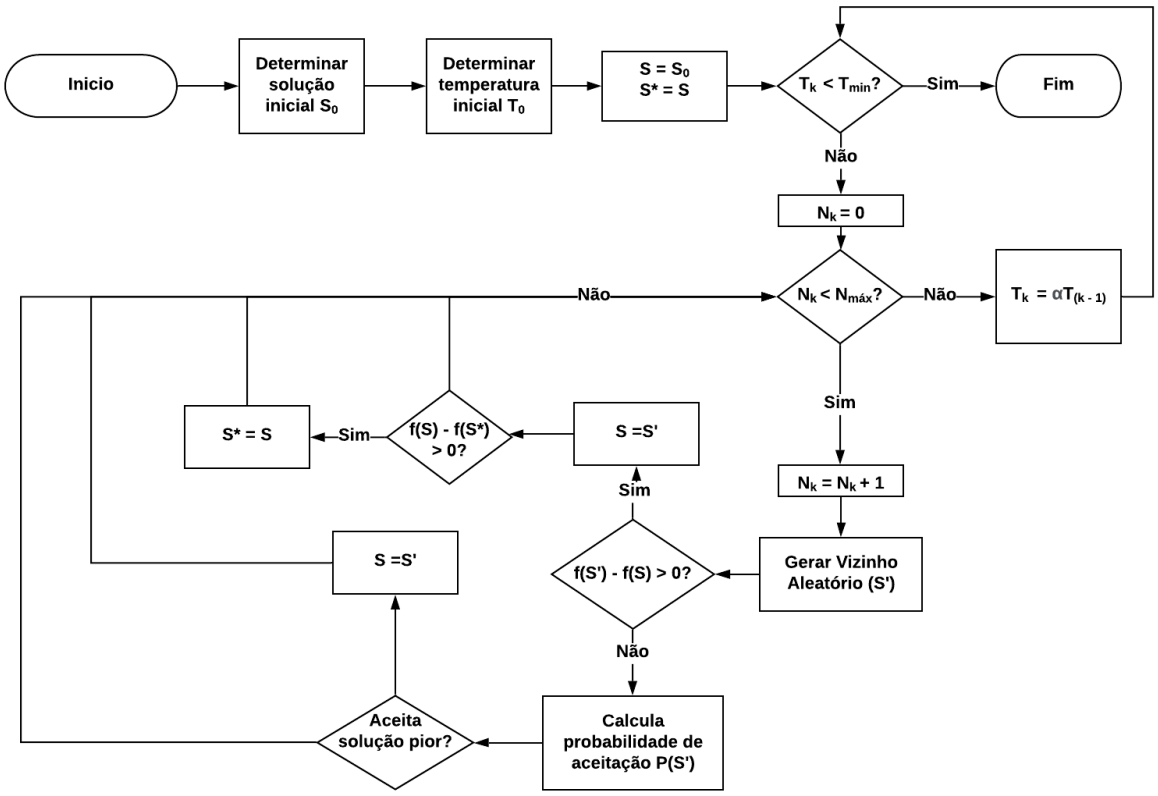
2.6 *SIMULATED ANNEALING*

A meta-heurística do *Simulated Annealing* proposta por Kirkipatrick, Gelatt e Vecchi (1983) consiste em simular o processo físico de recozimento aplicado à problemas combinatoriais. O processo de recozimento, na metalurgia, consiste em aquecer, em geral, um metal à altas temperaturas, para posteriormente resfriá-lo de maneira controlada, afim de se conseguir o melhor posicionamento dos átomos no material. Caso o resfriamento não ocorra de forma lenta, o sólido é congelado em uma estrutura localmente ótima e não atinge o estado de energia zero e, consequentemente, seu estado ótimo global (ARENALES et al., 2015).

No fluxograma apresentado na Figura 9 descreve-se as etapas do SA. Os parâmetros utilizados no fluxograma são apresentados a seguir e explicados mais adiante nessa seção.

- S_0: Solução inicial;
- T_0: Temperatura inicial;
- N_mx: Número de iterações para cada temperatura;
- $f(S)$: Função objetivo para determinada solução S;
- $P(S)$: Probabilidadde de aceitação da solução S.

Figura 9 – Fluxograma do procedimento SA



Fonte: Adaptado de Araujo (2001)

O procedimento SA, descrito na Figura 9, parte de uma solução inicial S_0, gerada aleatoriamente, e uma temperatura inicial T_0. Considerando ΔE^+ como a média aritimética da diferença de n soluções aleatórias e ϵ_0 = 0.8 um valor empirico. Tal temperatura inicial pode ser determinada pela Equação 17.

$$T_0 = -\frac{\Delta E^+}{ln(\epsilon_0)} \tag{17}$$

Para determinar o valor de ΔE^+ são geradas n soluções aleatórias e calculadas as variações entre as soluções, da primeira para a segunda, da segunda para a terceira e assim por diante. A média das variações das soluções é o valor de ΔE^+.

A cada iteração k do procedimento, é determinada uma temperatura T_k que é atualizada por um fator de resfriamento α, o qual varia entre $\{0, 1\}$. É realizada uma perturbação na solução inicial que gera uma nova solução S'. As perturbações são movimentos válidos dentro das restrições do problema. Então calcula-se

$$\Delta E = f(S) - f(S') \tag{18}$$

o qual é a energia do movimento. No caso do problema proposto neste trabalho, a função objetivo é de minimização, por isso se ΔE é positivo, a nova solução S' é aceita e se torna a solução incubente. Quando ΔE for negativo o caso deve ser tratado

probabilisticamente, segundo a Equação 19, afim de se escapar de ótimos locais, permitindo que a função objetivo tenha um valor de piora.

$$P(S') = \begin{cases} 1, & se \Delta E \leqslant 0 \\ exp(-\Delta E/T_k), & c.c. \end{cases} \tag{19}$$

Para cada temperatura são realizadas N_{max} iterações, na qual são realizadas as perturbações e geradas novas soluções S. A Figura 9 esquematiza o procedimento do algoritmo SA para um problema de minimização. O procedimento termina quando uma temperatura minima t_{min} é alcançada.

Nota-se que o algoritmo SA permite uma determinada flexibilidade na escolha das novas soluções resultantes de pertubações aleatórias. Desta maneira, para um mesmo problema, pode haver diferentes formas de escolha de pertubações que podem levar a soluções melhores ou não.

A definição dos parâmetros é relativa a cada problema, não existindo uma fórmula única, porém existem algumas diretrizes e métodos para determinar esses parâmetros.Uma série de simulações realizadas com o intuito de otimizar a definição dos parâmetros chegaram a algumas diretrizes (PINHEIRO et al., 2019):

- O valor de α deve ser grande o suficiente para permitir que a temperatura diminua de forma lenta;
- Iniciar com uma boa solução que usa algum conhecimento é sempre preferível que iniciar com uma solução aleatória;
- Em altas temperaturas quase todos os movimentos são aceitos, portanto não é necessário se gastar muito tempo nelas;
- Além do problema em questão, a parametrização depende também do tipo e do tamanho da instância que está sendo considerada.

Destaca-se que na literatura alguns autores, como Santos (1994), Fidelis e Arroyo (2017) e Dang, Nguyen e Rudová (2019), estudaram problemas de sequenciamento resolvidos com uso de métodos heurísticos. Neste trabalho será apresentada uma heurística de SA, validada por um modelo exato, para minimizar o *makespan* de um sistema de manufatura flexível, com janela de tempo para realização das tarefas baseada no horário de realização da tarefa anterior.

3 MÉTODOS PROPOSTOS

Neste capítulo apresentam-se os métodos de solução propostos. O primeiro, considera a Programação matemática, com a proposição de uma nova modelagem, baseada em Dang et al. (2013), e será descrito na próxima seção. O segundo método considera um método aproximado, baseado na Meta-heurística Simulated Anealing.

3.1 MÉTODO DE SOLUÇÃO VIA PROGRAMAÇÃO MATEMÁTICA

Nesta seção é apresentado o modelo matemático desenvolvido baseado no modelo explicado na seção 2.3.2. A principal diferença entre os dois modelos é na formação da janela de tempo restrita. No modelo de Dang et al. (2013) os tempos de início e fim da janela de tempo são pré-definidos, baseados somente no volume máximo e mínimo de peças que cada alimentador comporta e na taxa de alimentação de peça para máquina, como representado pelas Equações 1. Já no método aqui proposto, a janela de tempo continua sendo restrita, ou seja, a tarefa deve obrigatoriamente ser realizada dentro deste intervalo de tempo, porém o tempo de início e fim deste intervalo podem variar de acordo com o momento em que a tarefa anterior foi realizada.

Para isso, foi necessário acrescentar alguns parâmetros, variáveis e restrições, além de ter que fazer algumas adaptações em partes do modelo anterior. Parte das restrições foram mantidas sem alterações.

Abaixo estão os parâmetros e variáveis alterados ou acrescentados. Os parâmetros e variáveis não listados continuaram os mesmos:

- N : grupo de todas as tarefas ($N = 1, 2, 3, ..., n$ no qual 1: tarefa no armazém), por causa da linguagem de programação utilizada para modelar o método, AMPL, foi necessário trocar o índice das tarefas realizadas no armazém de 0 para 1;
- SLC_i: este novo parâmetro indica a quantidade de peças que há em cada caixa para cada alimentador i;
- F_{ik}: esta nova variável indica quantas peças existem no alimentador i, logo após a subtarefa k.

A primeira mudança realizada no modelo foi na função objetivo. A alteração foi feita para a função objetivo não levar em consideração somente o tempo de viagem do robô entre as máquinas, mas também entre as máquinas e o armazém central quando for necessário. Com isso a função objetivo ficou da seguinte maneira:

$$MinZ = \sum_{i \in N} \sum_{k=1}^{n_i} \sum_{j \in N} \sum_{l=1}^{n_j} t_{ij} + (y_{jl} - y_{ik})(t_{i1} + t_{1j} - t_{ij})) * \sum_{r \in R} X_{ik}^{jlr} \tag{20}$$

Como explicado anteriormente, a janela de tempo proposta nesse modelo não se baseia somente no volume máximo e mínimo de peças de cada alimentador, por isso o conjunto de Equações 1 não será necessária neste modelo. Já as Equações 2 e 3 foram alteradas para que a janela de tempo seja baseada na quantidade de peças que foram adicionadas ao alimentador na subtarefa anterior e no momento em que a subtarefa anterior foi realizada. Deste modo as Equações 2 e 3 foram substituidas pelas Equações 21 e 22, respectivamente.

$$e_{ik} = (F_{ik-1} - v_i) * c_i + S_{ik-1}, \forall i \in N\backslash\{0\}, k \in 1, 2, \ldots, n_i, e_{i1} = 0 \tag{21}$$

$$d_{ik} = (F_{ik-1} - 0) * c_i + S_{ik-1}, \forall iN\backslash\{0\}, k = 1, 2, \ldots, n_i \tag{22}$$

De maneira análoga, os conjuntos de restrições 5 a 16 foram mantidos com o mesmo objetivo explicado na seção anterior. Alterou-se apenas o índice das tarefas do armazém, quando necessário. Além dessas restrições, foram adicionadas outras três, representadas nas Equações 23, 24 e 25. A primeira indica que após a primeira subtarefa ser realizada o número de peças no alimentador i será igual ao número de peças existentes no SLC_i. Isto se deve ao fato de que consideramos que todas as máquinas estão vazias no começo do problema. O grupo seguinte de restrições indica que em nenhum momento o número de peças no alimentador pode exceder a sua capacidade máxima. Já as restrições 25 são responsáveis por calcular o número de peças no alimentador após uma subtarefa, da segunda subtarefa em diante.

$$F_{i1} = SLC_i, \forall i \in N\backslash\{0\} \tag{23}$$

$$F_{ik} \leq u_i, \forall i \in N\backslash\{0\}, k = 1, 2, \ldots, n_i \tag{24}$$

$$F_{ik} = [F_{ik-1} - (S_{ik} - S_{ik-1} - w_i)/c_i] + SLC_i, \forall i \in N\backslash\{0\}, k = 2, 3, \ldots, n_i \tag{25}$$

Como explicado anteriormente, o método de solução via programação matemática, apesar de encontrar a solução ótima, é incapaz de resolver cenários com grandes instâncias em tempo polinomial, para isso é necessário utilizar métodos heurísticos. A seguir apresenta-se o detalhamento do método meta-heurístico proposto, baseado no SA.

O modelo foi implementado em linguagem AMPL a ser resolvido utilizando o solver GUROBI disponível no site NEOS Server.

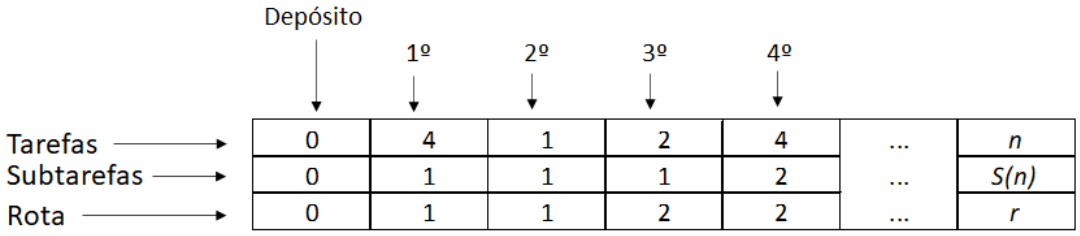
3.2 MÉTODO DE SOLUÇÃO VIA *SIMULATED ANNEALING*

Nesta seção descreve-se o detalhamento do algoritmo SA proposto para a resolução do problema considerado, seguindo o fluxograma geral da Figura 9.

3.2.1 Codificação da solução

Para facilitar a aplicação do SA, fez-se necessário utilizar uma codificação da solução, como representado na Figura 10. Utilizou-se uma matriz de ordem (3 x n), na qual a primeira linha representa a sequência das máquinas em que as tarefas são realizadas, a segunda linha indica qual a subtarefa da respectiva tarefa, e por fim, a última linha indica em qual rota a tarefa é realizada. Sendo, n o número total de tarefas, $S(n)$ o total de subtarefas da tarefas n e r o índice da última rota. Ressalta-se que na primeira coluna da matriz sempre indica a tarefa no armazém.

Figura 10 – Representação das soluções via SA



Fonte: Autor (2021)

A escolha desta representação para o problema se deu pela facilidade em realizar as perturbações na solução no processo de busca local do algoritmo SA. Entretanto, devido às restrições 21 e 22, que definem as janelas de tempo de realizações das tarefas, as pertubações precisaram ser controladas para que seja possível considerar apenas soluções viáveis.

3.2.2 Solução Inicial

Após a inicialização dos parâmetros, aplicou-se uma heurística construtiva aleatória controlada para obtenção da solução inicial. Um exemplo de solução inicial, utilizando a codificação explicada na seção anterior, está respresentada na Figura 11.

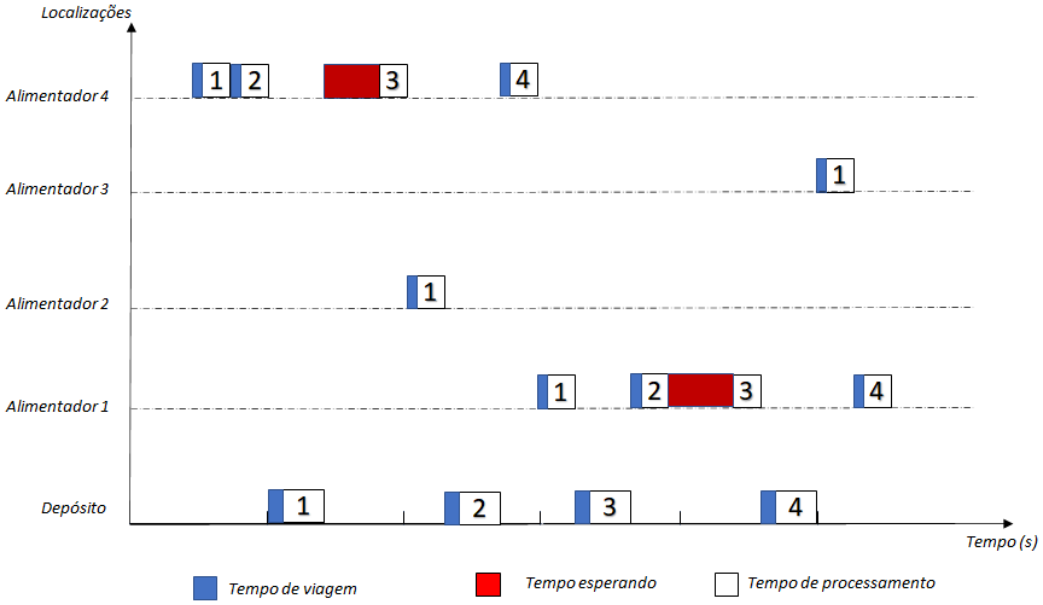
Figura 11 – Exemplo de solução inicial

0	4	4	4	2	4	1	1	1	3	1
0	1	2	3	1	4	1	2	3	1	4
0	1	1	2	2	3	3	4	4	5	5

Fonte: Autor (2021)

A fim de se obter uma melhor vizualização, no gráfico, representado na Figura 12, está indicado o *status* o AMR ao longo do horizonte de planejamento. As linhas horizontais do gráfico indicam os alimentadores e o depósito, enquanto o eixo das abscissas indica o tempo. Os números apontam a subtarefa e cada cor representa um *status* diferente.

Figura 12 – *Status* do AMR ao longo da Solução Inicial



Fonte: Autor (2021)

3.2.3 Avaliação da solução

Para avaliar a qualidade da solução encontrada primeiramente é verificado se a solução atende a todas as restrições, de capacidade e de janela de tempo de execução das tarefas. Caso não atenda, a solução é automaticamente descartada e uma nova solução é gerada.

Em caso de solução factível, a qualidade da solução é calculada considerando o tempo necessário para realizar toda a sequência de tarefas. Ou seja, soma-se o tempo de viagem entre os alimentadores, o tempo de viagem até o deposito quando necessário, o tempo de processamento do AMR em cada tarefa e o tempo que o AMR se encontra parado. No caso da solução representada na Figura 11 o tempo total

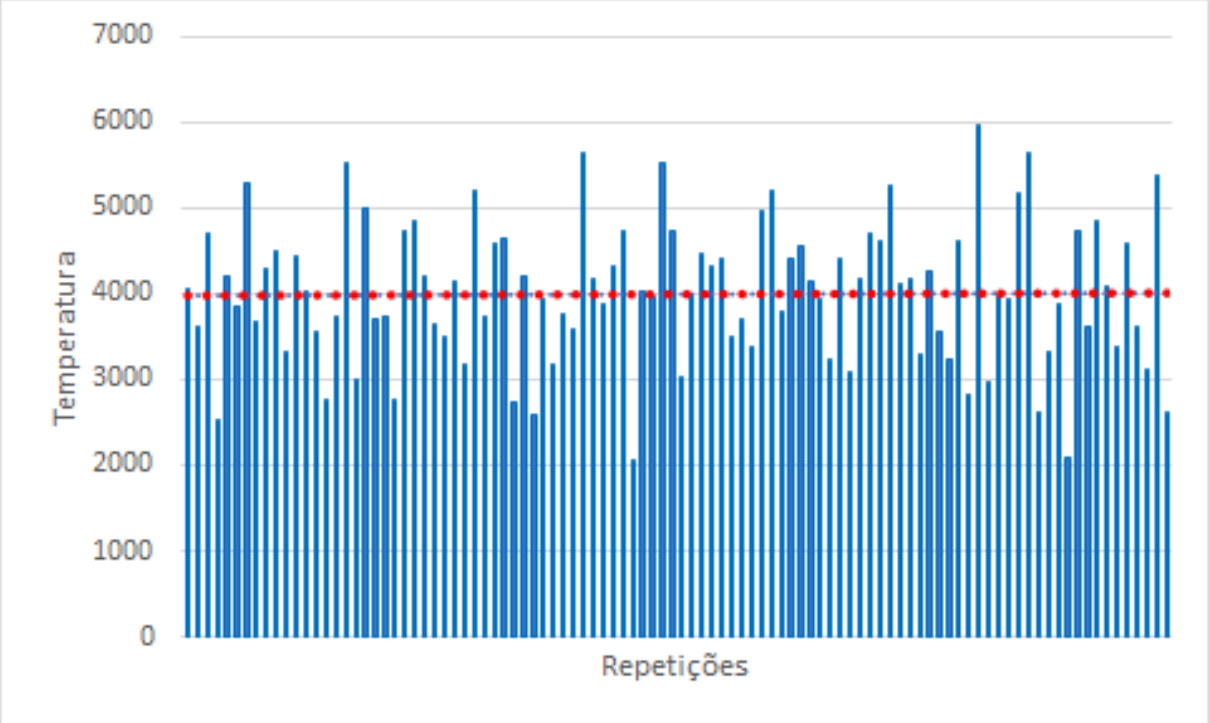
necessário para realizar toda a sequência de tarefas é de 2717,5 segundos.

3.2.4 Cálculo da temperatura inicial

Para determinar a temperatura inicial gerou-se 10 soluções aleatórias e encontrou-se a média na variação de energia. Desta forma, com o auxílio da Equação 17, foi possível determinar uma temperatura inicial adequada. O modelo foi testado com diferentes taxas de resfriamento. Iniciou-se o procedimento com taxas altas, e posteriormente com taxas mais baixas.

Na Figura 13 está apresentado o resultado das simulações para determinação da temperatura inicial, calculada utilizando-se a equação 1. A simulação foi repetida cem vezes para que fosse encontrada a média das temperaturas iniciais geradas. Nesse caso verifica-se que a maior temperatura encontrada foi de 5949 e o valor médio das temperaturas ficou em 4002.

Figura 13 – Simulação para cálculo da temperatura inicial



Fonte: Autor (2021)

Quanto mais elevada a temperatura inicial mais o modelo aceita resposta que pioram a função objetivo. Para evitar que o modelo se torne muito aleatório foi definido como temperatura inicial a média encontrada nas 100 repetições, logo $T_0 = 4002$.

Para o critério de parada foi considerado uma temperatura de congelamento, que, pela definição do SA, é quando a possibilidade de melhoria do resultado corrente é muito baixa. Portanto, a temperatura de congelamento deve ser muito próxima de zero, e neste trabalho adotou-se o valor de $0,001$.

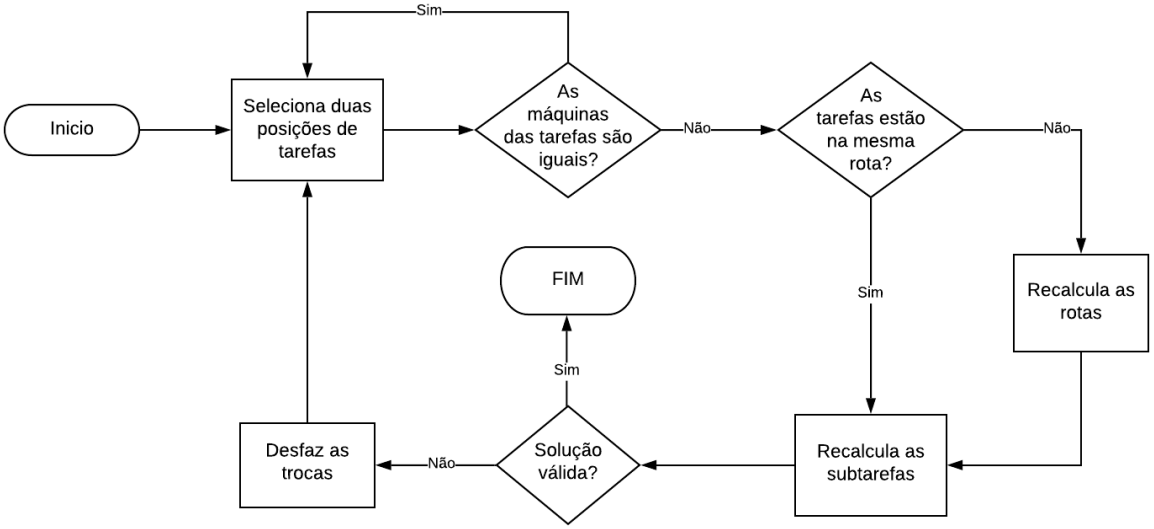
3.2.5 Definição das vizinhanças

A partir da solução inicial, e considerando-se a representação da Figura 10, aplicou-se aleatoriamente três diferentes tipos de perturbação, ou seja, três regras de definição de vizinhanças, definidas da seguinte forma:

- N_1: Troca de tarefas na mesma rota (Figura 15(1));
- N_2: Troca de tarefas em rotas diferentes (Figura 15(2));
- N_3: Troca de rotas (Figura 16).

As vizinhanças são escolhidas de forma aleatória, seguindo o fluxograma da Figura 14, na qual destaca-se que são aceitas somente soluções vizinhas viáveis.

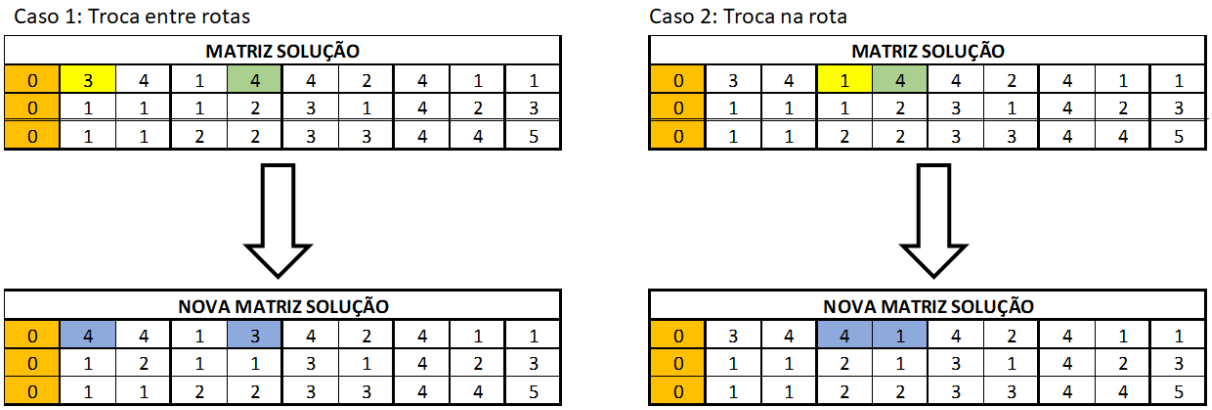
Figura 14 – Fluxograma do procedimento para gerar pertubações



Fonte: Autor (2021)

Para a escolha das trocas de tarefas nas vizinhanças, selecionam-se duas posições aleatórias na matriz de solução representada na Figura 10. As posições escolhidas devem possuir tarefas em máquinas diferentes e podem estar (N_1 Figura 15(1)), ou não na mesma rota (N_2 Figura 15(2)).

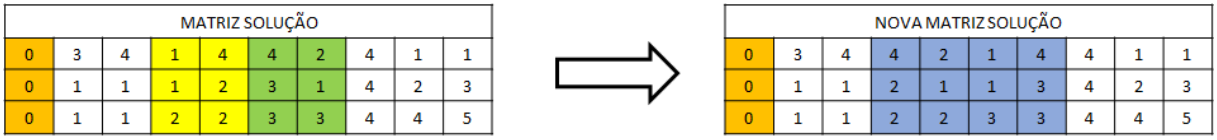
Figura 15 – Possibilidades de trocas de tarefas



Fonte: Autor (2021)

Para a troca de rotas (N_3) escolhe-se duas rotas aleatórias e trocam-se todas as tarefas realizadas entre estas rotas (Figura 16).

Figura 16 – Possibilidades de trocas de rotas



Fonte: Autor (2021)

Se a troca sugerida satisfizer as restrições da janela de tempo a troca é efetivada, caso contrário, a troca é descartada e então sorteia-se novas tarefas (novas posições na matriz). A busca local nas vizinhanças da solução atual poderá ocasionar aumento ou diminuição no tempo de viagem.

No capítulo seguinte apresentam-se os dados utilizados, os resultados obtidos com a aplicação dos modelos propostos.

4 RESULTADOS

Neste capítulo são apresentados os dados utilizados, os resultados obtidos e as análises quanto ao desempenho computacional dos métodos e qualidade das soluções obtidas com a aplicação dos métodos propostos.

O computador utilizado na aplicação dos métodos possui as especificações a seguir:

1. Processador Inter(R) Core(TM) i5-7200U CPU @ 2.50GHz; e
2. Memória RAM 8.00 GB (7,88 GB usáveis); e
3. Tipo de sistema 64-bit; e
4. Sistema operacional Windows 10.

4.1 DADOS E PARÂMETROS UTILIZADOS

Os dados utilizados neste trabalho foram os mesmos utilizados em Dang et al. (2013).

Destaca-se que o número de peças por SLC que alimentam as máquinas 1 e 4 é de 125 peças, com aproximadamente 2kg por caixa. E o número médio de peças nos SLC que alimentam as máquinas 2 e 3 é de 1100 peças, com aproximadamente 1kg por caixa.

Na Tabela 1 detalham-se o número máximo e mínimo de peças em cada alimentador, e a taxa de alimentação de peça do alimentador para máquina. Portanto, alimentadores 1 e 4 alimentam a máquina com uma peça a cada 4,5 segundos. Já os alimentadores 2 e 3 carregam a máquina com uma palheta a cada 1,5 segundo, três palhetas a cada 4,5 segundos.

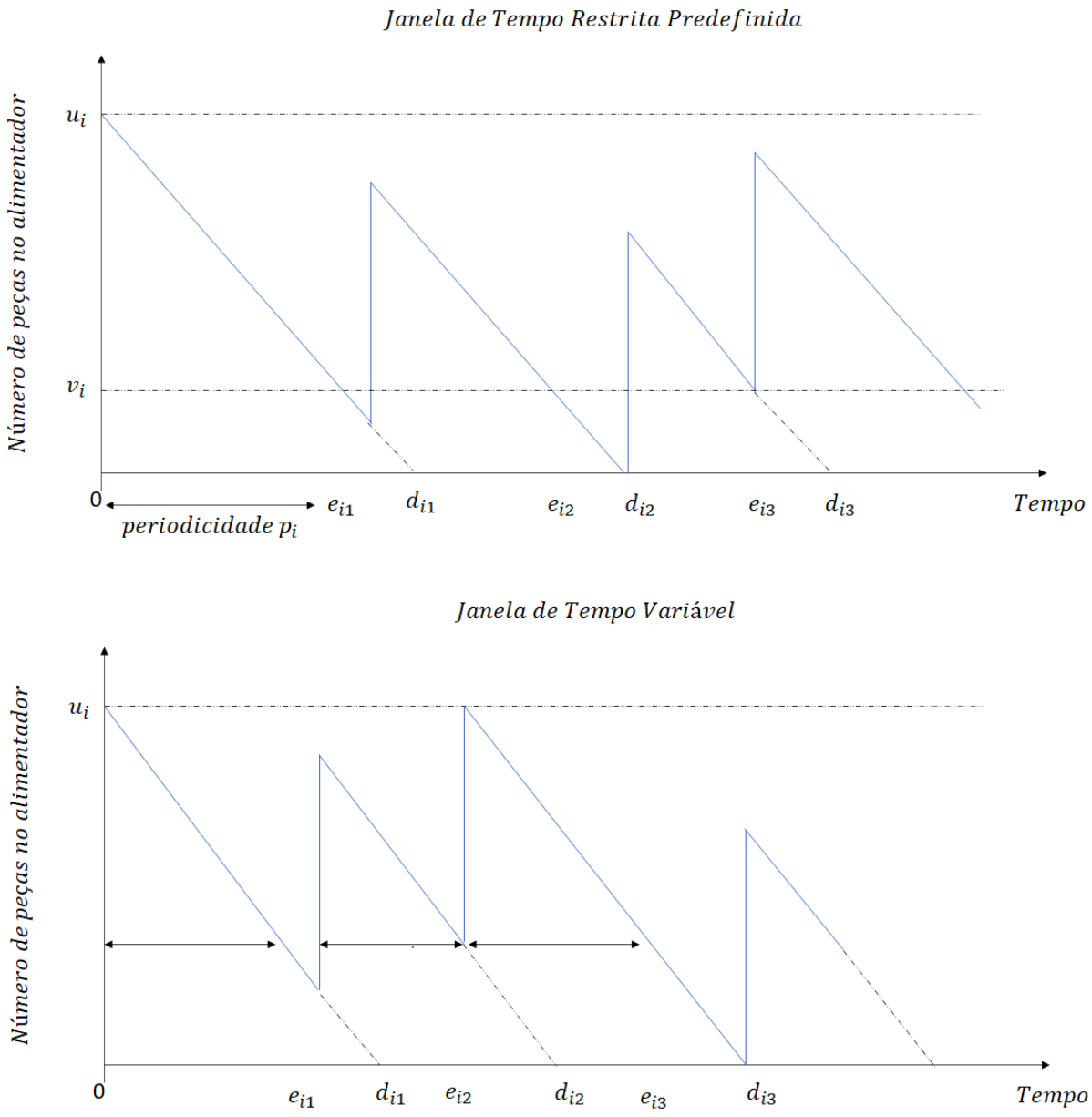
Tabela 1 – Níveis de peças e taxas de alimentação

Alimentador	1	2	3	4
Nível Máximo (peças)	250	2000	2000	250
Nível Mínimo (peças)	125	900	900	125
Taxa de Alimentação (peça/segundos)	4,5	1,5	1,5	4,5

Fonte: Dang et al. (2013)

Os dados da Tabela 1 junto com os conjuntos de equações 1, 2 e 3 são usados para calcular a periodicidade da tarefa (p_i), o limite inferior (e_ik) e o limite superior (d_ik) da janela de tempo do modelo de Dang et al. (2013) . Já para calcular a janela de tempo do modelo proposto os dados da Tabela 1 são usados juntos com as Equações 21 e 22. A diferença entre as duas janelas de tempo podem ser observadas na Figura 17.

Figura 17 – Representação das janelas de tempo utilizada em Dang et al. (2013) e a proposta pelo autor



Fonte: Autor (2021)

Como no modelo porposto a janela de tempo depende da hora de início da subtarefa anterior, podemos observar na Figura 17 que o AMR tende a manter o número de peças no alimentador o mais perto do máximo possível. Além disso, calculando-se a janela de tempo desta forma não é possível ter uma periodicidade, por isso foi chamado de janela de tempo variável.

Os tempos de trabalho do robô nos alimentadores estão apresentados na Tabela 2.

Tabela 2 – Tempo de trabalho do robô nos alimentadores

Localização	depósito	A1	A2	A3	A4
Tempo de Trabalho do Robô (segundos)	90	42	42	42	42

Fonte: Dang et al. (2013)

Na Tabela 3 apresentam-se o tempo de viagem do robô entre todos os pares de localização. Para ambas as tabelas os valores são apresentados em segundos e o índice 1 se refere ao depósito central.

Tabela 3 – Tempo de viagem do robô entre as localizações (segundos)

Origem/Destino	Depósito	A1	A2	A3	A4
Depósito	0	34	37	34	40
Alimentador 1	39	0	17	34	50
Alimentador 2	35	17	0	35	49
Alimentador 3	34	33	35	0	47
Alimentador 4	36	47	48	46	0

Fonte: Dang et al. (2013)

Foram simulados diferentes cenários considerando-se os dados apresentados.Para testar a eficiência do método heurístico proposto considerou-se os cenários apresentados na Tabela 4, considerando-se, inicialmente diferentes quantidades de alimentadores (2 a 4) e tarefas (5 a 10), e em seguida, com variação capacidade máxima de carregamento do AMR (2 a 3) e tarefas (5 a 10). Os parâmetros de cada cenário pode ser observado na Tabela 4.

Tabela 4 – Cenários analisados

Cenário	Número de alimentadores	Capacidade	Total de subtarefas
1	2	2	5
2	2	2	6
3	2	3	5
4	2	3	6
5	3	2	7
6	3	2	9
7	3	3	7
8	3	3	9
9	4	2	8
10	4	2	10
11	4	3	8
12	4	3	10

Fonte: Adaptado de Dang et al. (2013)

Os resultados para estes cenários utilizando os métodos de solução via

modelagem matemática são explicados à seguir.

4.2 RESULTADOS - SOLUÇÃO VIA MODELAGEM MATEMÁTICA

Na Tabela 5 apresenta-se a comparação dos resultados obtidos utilizando o modelo de Dang et al. (2013) e o proposto nesse trabalho.

Tabela 5 – Comparação dos resultados para os diferentes cenários

	Dang et al (2014)		Modelo Proposto		
Cenário	Variáveis	FO (s)	Variáveis	FO (s)	Diferença (%)
1	310	891,0	319	891,0	0,00
2	432	1046,5	444	1023,5	2,20
3	310	770,0	319	770,0	0,00
4	432	888,0	444	888,0	0,00
5	574	1342,5	586	1374,5	-2,38
6	918	1415,5	930	1374,5	2,90
7	574	1380,5	586	1367,5	0,94
8	918	1418,5	930	1369,5	3,45
9	736	1654,5	748	1379,0	16,65
10	1120	1754,0	1132	1462,0	16,65
11	736	1617,5	748	1372,0	15,18
12	1120	1692,0	1132	1410,0	16,67

Fonte: Autor (2021)

Destaca-se que os cenários analisados nesse trabalho foram também resolvidos considerando-se o modelo proposto por Dang et al. (2013), conforme apresentado na Tabela 5.

É possível observar na Tabela 5 que conforme o número de alimentadores no cenário aumenta a diferença entre os modelos também aumenta. O mesmo pode ser observado em relação ao número de subtarefas. O modelo proposto gerou uma melhora média no tempo total do AMR para a realização de todas as tarefas do processso de fabricação de 6,02% em relação ao modelo de Dang et al. (2013) para as instâncias consideradas. A maior diferença entre os modelos resultou em uma redução de 16,67% do tempo total de viagem.

O método proposto, como explicado na seção anterior, requer um maior número de variáveis e restrições, o que leva a um aumento significativo de iterações necessárias para se encontrar a solução ótima de um cenário quando comparado com o modelo original. Por esse motivo, o método proposto não foi capaz de encontrar, em tempo polinomial, o resultado para alguns dos cenários, como já era esperado, pois consideram-se variáveis de decisão inteiras, o que torna o problema de otimização

muito mais difícil de ser resolvido.

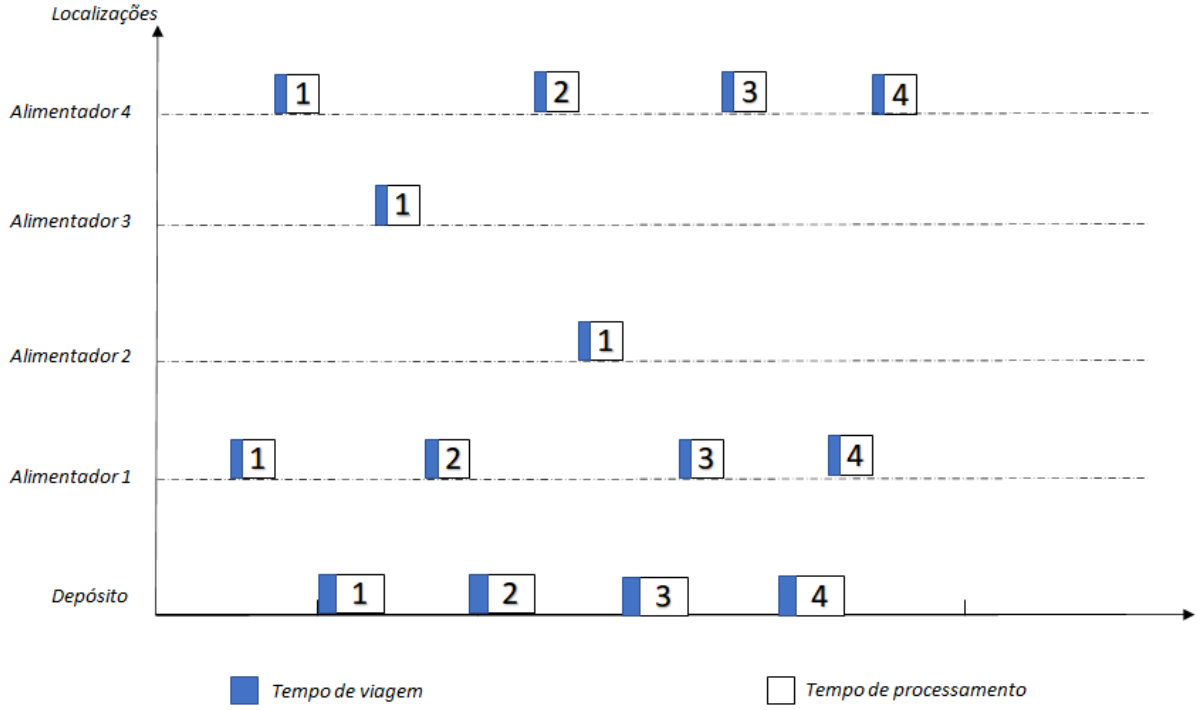
Considerando-se os dados apresentados na seção anterior, a solução ótima para o cenário 10 encontrada pelo modelo proposto pode ser observada nas Figuras 18 e 19.

Figura 18 – Codificação da solução ótima do modelo proposto para o cenário 10

0	1	4	3	1	4	2	1	4	1	4
0	1	1	1	2	2	1	3	3	4	4
0	1	1	2	2	3	3	4	4	5	5

Fonte: Autor (2021)

Figura 19 – Representação gráfica da solução ótima do cenário 10



Fonte: Autor (2021)

Conforme é possivel ser observado nas Figuras 18 e 19, o AMR inicia a operação saindo do depósito e indo realizar a primeira subtarefa do alimentador 1 e, em seguida, vaija para o alimentador 4 para processar sua primeira subtarefa. Após processar a tarefa no alimentador 4, o AMR precisa retornar ao depósito devido à sua capacidade. Então, o AMR viaja até o depósito, substitui os SLCs vazios por outros cheios e segue para realizar as tarefas da rota 2. Por fim, o AMR consegue realizar todas as subtarefas desse cenário em um total de 1462 segundos.

A memória utilizada e o tempo de solução aumentam exponencialmente conforme o tamanho do problema aumenta, portanto o modelo só pode ser aplicado na prática em pequenas escalas com poucos alimentadores na linha de produção e um tempo de planejamento curto. Para esses cenários, o modelo PLIM vai gerar a solução

ótima, o que pode ser usado como parâmetro de comparação para a qualificação de métodos heurísticos a serem considerados.

4.3 RESULTADOS - SOLUÇÃO VIA META-HEURÍSTICA *SIMULATED ANNEALING*

Como explicado na seção 3.2, os principais parâmetros que interferem na qualidade do resultado encontrado via meta-heurística SA, são: a temperatura inicial, a temperatura final, a taxa de resfriamento e o número de iterações realizadas para cada temperatura.

A temperatura inicial e final já foi calculada na seção 3.2.4, considerando um valor de 4002 e 0,001, respectivamente. Na próxima subseção é apresentado os testes realizados a fim de se encontrar o valor ideal para a taxa de resfriamento e o número de iterações para cada temperatura.

4.3.1 Resultados para diferentes taxas de resfriamento

Para se definir a taxa de resfriamento, α, a partir da solução inicial encontrada, foram realizadas 1.000 repetições do algoritmo SA, com 5 taxas de resfriamento diferentes.

O valor de α deve estar muito próximo de 1 para permitir que a temperatura decresça lentamente e é usualmente definido entre 0,8 e 0,99 (ARAUJO, 2001). Portanto, consideraram-se cinco valores diferentes para o fator de resfriamento a fim de analisar o comportamento do algoritmo quando submetido a diferentes quedas de temperatura. Os valores definidos para foram 0,80; 0,85; 0,90; 0,95 e 0,98. A Tabela 6 apresenta os resultados encontrados para as diferentes taxas de resfriamento, considerando o valor das temperaturas inical e final definidos anteriormente.

Tabela 6 – Resultados obtidos pelo SA para o cenário 10

Taxa de resfriamento (α)	0,8	0,85	0,9	0,95	0,98
Solução Incumbente (s)	1467,5	1466,0	1464,0	1462,0	1462,0
Temperatura da solução incumbente	0,0940	0,033	0,057	0,049	0,001
Tempo de processamento (s)	83,69	111,92	161,35	313,33	536,63

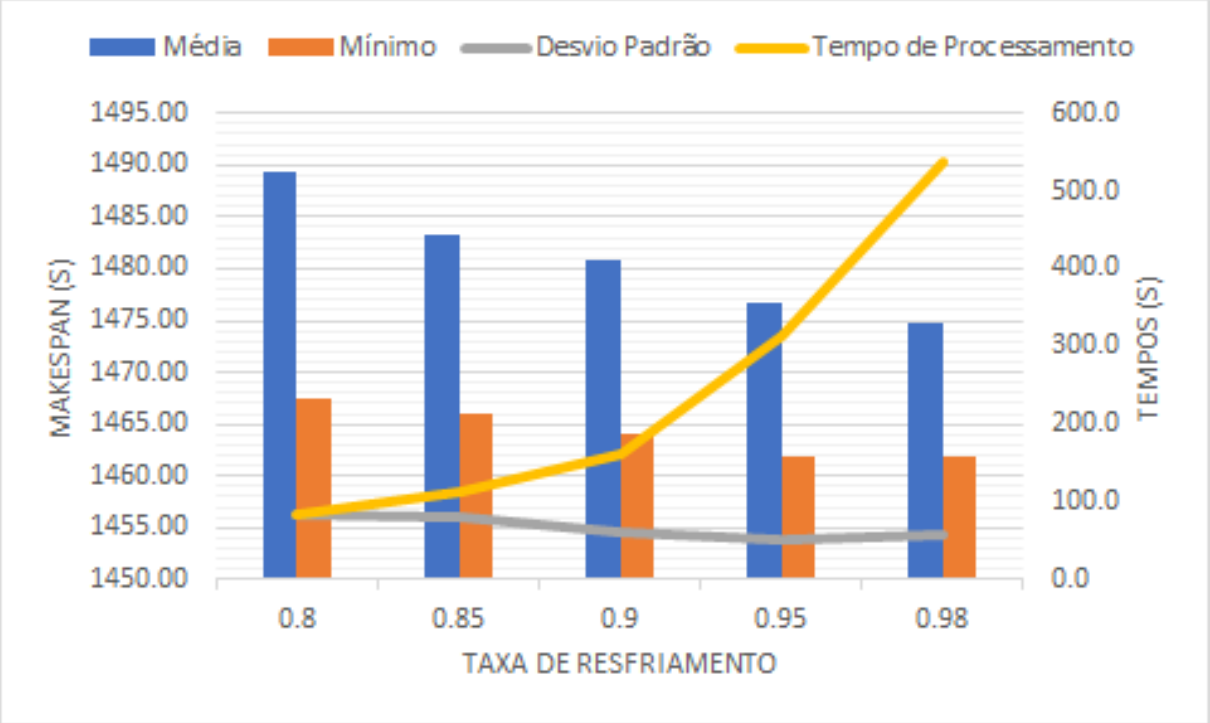
Fonte: Autor (2021)

Para o cenário 10, usando os dados descritos na seção 4.1, as taxas de resfriamento α =0,98 e α =0,95 conseguiram encontrar a melhor solução com o tempo total do *makespam* de 1462 segundos, a mesma encontrada pelo modelo exato. Para todas as taxas de resfriamento consideradas pode-se observar que a temperatura em que a melhor solução é encontrada está muito próxima de zero, sendo menor do que 0,1 em todos os casos.

No início do algoritmo, quando a temperatura está mais elevada, a meta-heurística aceita diversas soluções que pioram o valor da função objetivo. Isso acontece devido as equações 18 e 19, que definem que quanto mais próximo de zero menor a probabilidade de aceitar soluções ruins, portanto conforme o decréscimo da temperatura existe a tendência de que as soluções aceitas sempre sejam de melhoria na função objetivo.

Analogamente é possível constatar que, quanto mais se aumenta a taxa de resfriamento, maior é a chance de encontrar melhores soluções. Tal conclusão pode ser melhor observada pelo gráfico da Figura 20.

Figura 20 – Solução incumbente e tempo de processamento conforme taxa de resfriamento



Fonte: Autor (2021)

Destaca-se que, para os dados considerados neste trabalho, as taxas de resfriamento de 0,95 e 0,98 possibilitaram determinar a solução ótima, o que gera um maior tempo de processamento do algoritmo.

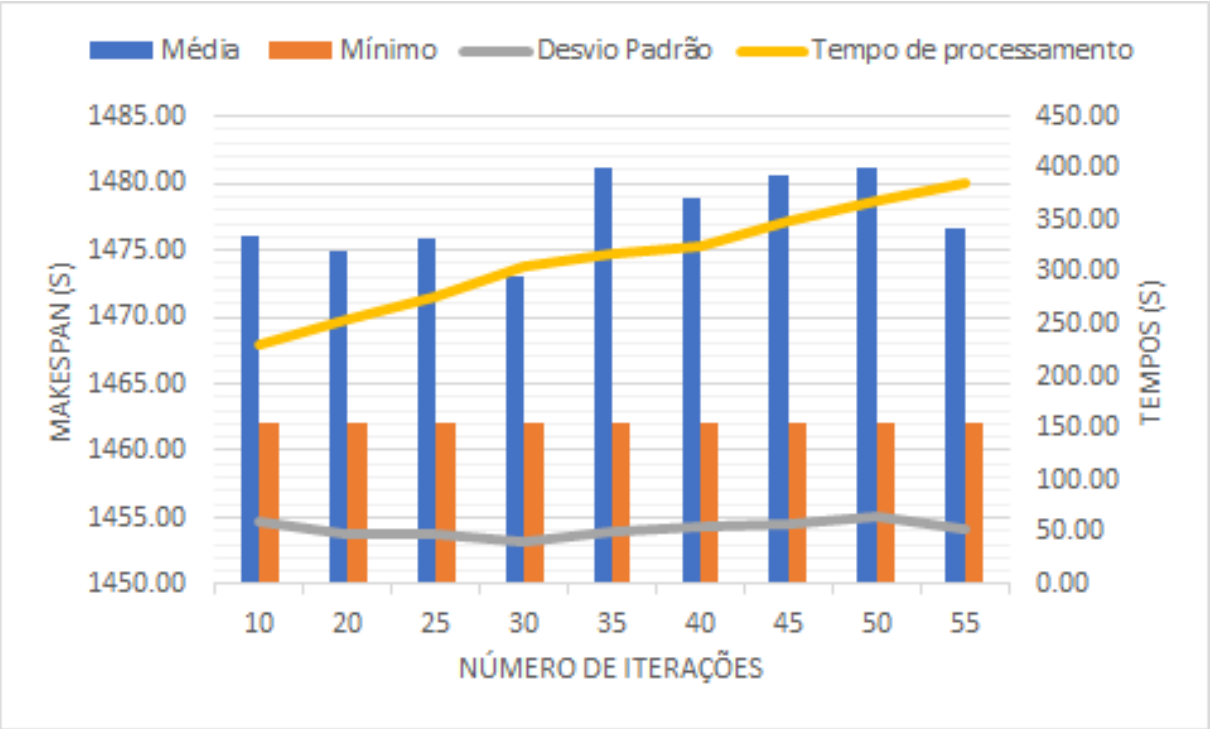
Para questões operacionais, na qual se quer resultados satisfatórios de forma rápida, o uso de uma taxa de resfriamento de α=0,98 se mostrou desnecessária, pois o tempo de processamento para essa temperatura é aproximadamente 1,7 vezes maior que o tempo de processamento para α=0,95, enquanto ambas foram capazes de encontrar a solução ótima e o desvio padrão das soluções encontradas foi praticamente o mesmo. A taxa de resfriamento de 0,98 só se mostrou um pouco melhor na média das soluções encontradas nas 1000 simulações testadas. Por isso, o valor de α escolhido

foi o de 0,95.

4.3.2 Resultados para diferentes quantidades de iterações

Por fim, para encontrar o melhor valor do número de iterações, SA_{max}, em cada temperatura foram realizados testes similares aos que foram feito para achar α. Foram testadas simulações com o 10, 20, 25, 30 ,35, 40, 45, 50 e 55 iterações por temperatura. A Figura 21 apresenta os resultados encontrados nesses testes.

Figura 21 – Solução incumbente e tempo de processamento conforme o número de iterações dentro do mesmo nível de temperatura



Fonte: Autor (2021)

Observa-se no gráfico da Figura 21, que quanto maior o número de iterações maior o tempo de processamento, entretanto, como pode-se observar na Figura 21 isso não significa um resultado melhor. Todos os testes, independente do número de iterações, conseguiram em alguma simulação alcancar a solução ótima, porém a que teve a menor média foi o teste com 30 iterações por temperatura, com um *makespan* médio de 1472 segundos e um desvio padrão de 40 segundos.

Ainda na Figura 21 observa-se que os testes com o maior número de iterações obtiveram as piores médias. Esse fato pode ser explicado, pois quanto maior o número de iterações maior a aleatoriedade no inicio da heurística, quando a temperatura está mais elevada, aceitando um número significativo de soluções que pioram a função objetivo. Porém, quando o número de iterações é baixo podemos notar o problema inverso, no qual o número de tentativas não é o suficiente para se encontrar uma

solução melhor que a atual mesmo em temperaturas altas.

A Tabela 7 indica os valores ideiais para os parâmetros: temperatura inicial, temperatura final, taxa de resfriamento e número de iterações realizadas para cada temperatura.

Tabela 7 – Definição dos parâmetros para o SA

T_0	SA_{max}	T_f	α
4002	30	0,001	0,95

Fonte: Autor (2021)

Para avaliar a qualidade dos resultados obtidos com o SA foram realizados testes com os cenários descritos na Tabela 4 e utilizando os dados apresentados na Tabela 7 como parâmetros para o modelo. Os resultados destes testes são apresentados na na Tabela 8.

Como é possível observar na Tabela 8, para todos os cenários o SA encontrou, como melhor resultado, o mesmo resultado obtido pelo método exato via programação matemática. Apesar de alguns cenários terem a média dos resultados encontrados relativamente altos em comparação com a solução ótima, essa diferença ficou em média 2,76% maior. Com isso, é possível afirmar que o SA gerou resultados confiáveis com um baixo tempo de processamento (TP).

Tabela 8 – Comparação dos resultados para os diferentes cenários

	Modelo	Matemático		*Simulated*	*Annealing*	
Cenário	Variáveis	FO(s)	Mínimo(s)	Média(s)	Desv. Pad.	TP(s)
1	319	891,0	891,0	967,0	83,2	284,10
2	444	1023,5	1023,5	1036,5	83,4	280,03
3	319	770,0	770,0	866,5	70,0	274,88
4	444	888,0	888,0	897,0	117,1	278,56
5	586	1374,5	1374,5	1392,5	16,0	260,54
6	930	1374,5	1374,5	1397,2	39,2	269,12
7	586	1367,5	1367,5	1387,6	74,2	276,56
8	930	1369,5	1369,5	1379,8	25,6	270,00
9	748	1379,0	1379,0	1385,5	82,5	297,00
10	1132	1462,0	1462,0	1473,0	38,3	307,11
11	748	1372,0	1372,0	1383,2	127,3	303,14
12	1132	1410,0	1410,0	1446,0	123,2	298,81

Fonte: Autor (2021)

A fim de se observar o desempenho do SA para o problema do SAMR em grandes instâncias foram criados cenários com o total de 16, 32 e 48 subtarefas, e com a capacidade do AMR (Q_m) de 2, 3 ou 4, como o descrito na Tabela 9. Foram utilizados os parâmetros da Tabela 7.

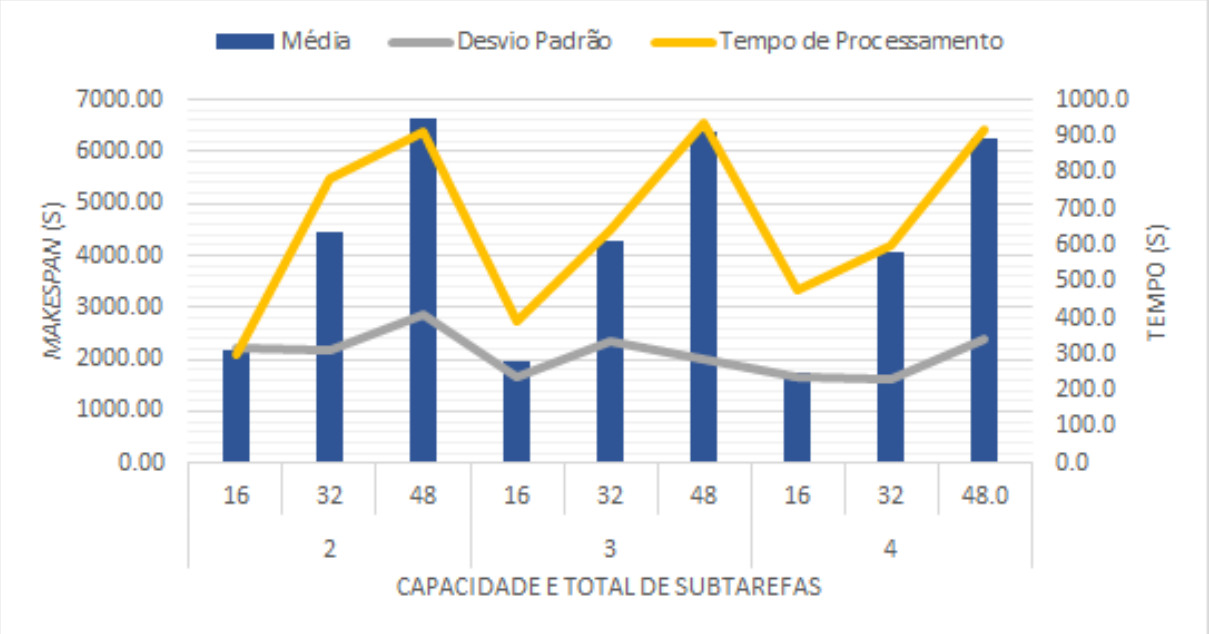
Tabela 9 – Resultados do SA para cenários maiores

Q_m	Subtarefas	Mínimo(s)	Média(s)	Desv. Pad.	TP(s)
2	16	2031.0	2193.6	315.5	298.18
	32	4271.0	4471.3	307.9	783.80
	48	6335.5	6657.4	411.8	913.25
3	16	1796.0	1947.1	239.5	387.70
	32	4049.5	4262.6	334.7	643.55
	48	6205.2	6526.3	342.7	920.20
4	16	1556.0	1749.9	236.6	474.40
	32	3928.5	4084.8	231.9	601.85
	48	6100.7	6397.8	285.4	939.00

Fonte: Autor (2021)

Analisando os resultados da Tabela 9 observa-se que quanto maior a capacidade do AMR menor o *makespan*, pois o AMR precisa retornar ao depósito uma menor quantidade de vezes, porém maior é o tempo de processamento. Também é possível perceber que o desvio padrão diminui conforme a capacidade de carga do AMR aumenta, o que indica uma solução mais confiável, porém aumentou em relação aos apresentados na Tabela 8, o que mostra uma menor uniformidade nos resultados obtidos. Os resultados podem ser melhor visualizados no gráfico da Figura 22.

Figura 22 – Resultados do SA para cenários maiores



Fonte: Autor (2021)

Em relação ao número de subtarefas totais, é possível observar pelo gráfico da Figura 22, que o *makespan* aumenta consideravelmente conforme a quantidade

de subtarefas aumenta, para as instâncias estudadas. É possível notar que o desvio padrão também aumenta, o que torna a solução menos confiável a medida que a instância analisada aumenta.

Neste capítulo foram apresentados os resultados obtidos para o problema do SAMR, comparando os métodos de solução via programação matemática e meta-heurístico. As considerações finais acerca dos resultados obtidos serão apresentados no próximo capítulo.

5 CONCLUSÕES

Este trabalho apresenta o resultado do estudo do problema de *scheduling* de um robô autônomo responsável por executar tarefas de alimentação de peças em máquinas de uma linha de produção em um ambiente de manufatura flexível. A fim de completar todas as tarefas no horizonte de planejamento esperado e respeitando o tempo de bateria de robo, é importante para os responsáveis pelo planejamento da produção determinar o sequenciamento das tarefas que minimize o tempo total de viagem e de execução das tarefas do AMR, levando-se em consideração uma série de restrições práticas. Para isso apresentou-se um método de solução utiliznado-se modelo de programação linear inteira mista (PLIM) para problemas de pequeno porte e um método de solução via a meta-heurística do *Simulated Annealing* para problemas de grande porte.

No PLIM proposto considera-se, simultaneamente, a janela de tempo restrita para a realização das tarefas e a limitação de capacidade de carregamento em um único robô. Destacando-se, como a principal contribuição do trabalho, a alteração da forma de calculo da janela de tempo. Diferentemente do modelo desenvolvido por Dang et al (2013), a janela de tempo restrita não é predefinida. No modelo proposto neste trabalho a janela de tempo varia de acordo com o horário em que a tarefa anterior foi realizada, o que levou a uma melhora de até 16,67% do tempo viagem total do AMR nos cenários estudados.

Devido a característica NP-hard do problema, o método de solução via modelagem matemática é apenas aplicável a problemas de pequena escala com poucos alimentadores e um horizonte de planejamento curto. Um método de solução via meta-heurística do *Simulated Annealing* foi proposto para encontrar soluções próximas ao ótimo para grandes instâncias. Destaca-se que, os métodos de perturbação utilizados na busca local do SA foram propostos pelo autor.

A qualidade das soluções obtidas com a aplicação do SA proposto foi avaliada utilizando-se como base de comparação as soluções do PLIM para quantificar a escala de benefícios. Utilizando o estudo de caso apresentado por Dang et al. (2013), o método gerou resultados confiáveis, sendo capaz de encontrar a solução ótima para todos os cenários de pequeno porte. Os resultados mostraram que o uso da heurística proposta foi significativamente mais rápida na obtenção de soluções quase ótimas.

Como Trabalho de Conclusão de Curso, o estudo realizado foi fundamental para o meu desenvolvimento acadêmico, me possibilitando colocar em prática os conhecimentos adquiridos durante o curso e ir além, pesquisando em materiais de relevância acadêmica. Infelizmente, a Pandemia da Covid-19 dificultou a obtenção de

dados para a realização de um estudo de caso.

5.1 TRABALHOS FUTUROS

Para trabalhos futuros propõe-se:

- Estudar o impacto da solução inicial na solução final da meta-heurística proposta, elaborando uma heurística para se obter uma solução inicial;
- Apresentar novos critérios de pertubação para o método de busca local do SA;
- Propor um modelo que considere mais de um AMR;
- Aplicar o modelo proposto para um estudo de caso real.

REFERÊNCIAS

AMPL, N. G. S. **User's Guide to the NEOS Server**. 2018. Disponível em: https://neos-guide.org/content/users-guide.

ANAND, R.; AGGARWAL, D.; KUMAR, V. A comparative analysis of optimization solvers. **Journal of Statistics and Management Systems**, Taylor Francis, v. 20, n. 4, p. 623–635, 2017. Disponível em: https://doi.org/10.1080/09720510.2017.1395182.

ARAUJO, H. A. d. **Algoritmo Simulated Annealing: Uma Nova Abordagem**. 2001. Dissertação (mathesis) — Universidade Federal de Santa Catarina, Florianópolis.

ARENALES, M. et al. **Pesquisa Operacional**. Rio de Janeiro: Elsevier, 2015.

ARNOLD, J. R. **Introduction to Materials Management**. . Columbus, Ohio: Prentice Hall, 1998.

BAKER, K. R. **Introduction to Sequencing and Scheduling**. New York: John Wiley, 1974.

BARBOSA, L. B. **Otimização do sequenciamento de tarefas em máquinas paralelas com tempos de processamento diferentes**. 2017. TCC (Graduação) – Curso de Engenharia de Transportes e Logística, Universidade Federal de Santa Catarina, Joinville.

BODIN, L. et al. Routing and scheduling of vehicles and crew : the state of the art. **Computers Operations Research**, v. 10, n. 2, p. 63–211, 1983.

BOWERSOX, D. J.; CLOSS, D. J. **Logística empresarial** : o processo de integração da cadeia de suprimento. São Paulo: Atlas, 2011.

CHEN, X. et al. Yard crane and agv scheduling in automated container terminal: A multi-robot task allocation framework. **Transportation Research Part C: Emerging Technologies**, v. 114, p. 241–271, 2020. ISSN 0968-090X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0968090X19313531.

COLIN, E. C. **Pesquisa Operacional** : 170 aplicações em Estratégia, Finanças, Logística, Produção, Marketing e vendas. Rio de Janeiro: LTC, 2013.

DANG, Q. et al. Scheduling a single mobile robot for part-feeding tasks of production lines. **Journal Of Intelligent Manufacturing**, v. 25, p. 1271–1287, 2013.

DANG, Q. V.; NGUYEN, C. T.; RUDOVá, H. Scheduling of mobile robots for transportation and manufacturing tasks. **Journal of Heuristics**, v. 25, p. 175–213, 2019.

DONG, J. et al. Single machine scheduling with job delivery to multiple customers. **Journal of Scheduling**, v. 21, p. 337–348, 2018.

FIDELIS, M. B.; ARROYO, J. E. C. **Heurísticas para o problema de sequenciamento de lotes de tarefas em máquinas paralelas**. 2017. XLIX Simpósio Brasileiro de Pesquisa Operacional, Blumenau.

FISCHETTI, M.; MARTELLO, S.; TOTH, P. The fixed job schedule problem with spread-time constraints. **Operations Research**, v. 35, n. 6, p. 849–858, 1987.

GIL, A. C. **Métodos e técnicas de pesquisa social**. 6. ed. São Paulo: Atlas, 2008.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização Combinatória e Programação Linear** : modelos e algoritmos. 2. ed. Rio de Janeiro: Elsevier, 2005.

GOLDBERG, D. E. Genetic algorithms in search. **Optimization, and Machine Learning**, 1989.

GRAVES, S. C. Review of production scheduling. **Operations Research**, v. 29, n. 4, p. 646–675, 1981.

HILLIER, F. S.; LIEBERMAN, G. J. **Introdução à pesquisa operacional**. 9. ed. Porto Alegre: AMGH, 2013.

HOCHBAUM, D. S. The scheduling problem. **RIOT: Remote Interactive Optimization Testbed**, 1999.

JOO, C. M.; KIM, B. S. Machine scheduling of time-dependent deteriorating jobs with determining the optimal number of rate modifying activities and the position of the activities. **Journal of Advanced Mechanical Design, Systems, and Manufacturing**, v. 9, n. 1, 2015.

KIRKIPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **Science**, v. 4598, n. 220, p. 671–680, 1983.

LU, Y. Industry 4.0 : a survey on technologies, applications and open research issues. **Journal of Industrial Information Integration**, v. 6, p. 1–10, junho 2017.

MACDOUGALL, W. **Industrie 4.0** : Smart Manufacturing for the Future. Berlin, 2014.

NIELSEN, I. et al. Scheduling of mobile robots with preemptive tasks. **Distributed Computing and Artificial Intelligence**, Springer, Switzerland, p. 19–27, 2014.

PAPE, C. L. **Constraint-based scheduling** : A tutorial. [S.l.], 2015.

PETROVIć, M.; MILJKOVIć, Z.; JOKIć, A. A novel methodology for optimal single mobile robot scheduling using whale optimization algorithm. **Applied Soft Computing**, v. 81, p. 105520, 2019. ISSN 1568-4946. Disponível em: https://www.sciencedirect.com/science/article/pii/S156849461930290X.

PINEDO, M. L. **Scheduling - Theory, Algorithms, and Systems**. 3. ed. Nova Iorque: Springer, 2008.

REKLAITIS, G. V. Review of scheduling of process operations. **Alche Symposium Series**, v. 78, n. 214, p. 119–133, 1982.

RIAZI, S.; LENNARTSON, B. Using cp/smt solvers for scheduling and routing of agvs. **IEEE Transactions on Automation Science and Engineering**, v. 18, n. 1, p. 218–229, 2021.

E SANTO, L. C. do. **Sequenciamento de máquinas através de algoritmos genéticos**. 2014. Trabalho de Conclusão de Graduação (Graduação) – Curso de Ciência da Computação, Universidade Estadual de Londrina, Londrina.

SANTOS, E. J. **Análise de Tempos de Estabelecimento e Ordem de Manufatura no Sequenciamento de Tarefas em Processos Batelada**. 1994. Tese – Faculdade de Engenharia Quínlica, UNICAMP, Campinas.

SANTOS, V. L. A. **Sequenciamento de tarefa em máquinas paralelas com desgastes dependentes da sequência: resolução heurística**. 2016. Dissertação (PósGraduação) - Curso Ciência da Computação, Universidade Federal de Viçosa, Viçosa.

SHIVANAND, H. K.; BENAL, M. M.; KOTI, V. **Flexible Manufacturing System**. Nova Deli: New Age International, 2006.

SLACK, N.; CHAMBERS, S.; JOHNSTON, R. **Administração da produção**. 2. ed. São Paulo: Atlas, 2002.

THOBEN, K. D.; WIESNER, S.; WUEST, T. "industrie 4.0"and smart manufacturing - a review of research issues and application examples. **International Journal of Automotive Technology**, v. 11, n. 1, p. 4–16, janeiro 2017.

TUBINO, D. F. **Manual de Planejamento e Controle da Produção**. 2. ed. São Paulo: Atlas, 2007.

YANG, D. L.; CHERN, M. S. A two-machine flowshop sequencing problem with limited waiting time constraints. **Computers Industrial Engineering**, v. 28, n. 1, p. 63–70, 1995. ISSN 0360-8352. Disponível em: https://www.sciencedirect.com/science/article/pii/036083529400026J.