Universidade Federal Fluminense

VITOR TOMAZ DE AQUINO

Proposta de Modelos Matemático e Heurístico para Otimizar o Processo de Formação de Carga em Planta de Recozimento em Caixa

VOLTA REDONDA

2021

VITOR TOMAZ DE AQUINO

Proposta de Modelos Matemático e Heurístico para Otimizar o Processo de Formação de Carga em Planta de Recozimento em Caixa

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Modelagem Computacional e Pesquisa Operacional.

Orientador:

Tiago Araújo Neves

Coorientador:

Weslley Luiz da Silva Assis

Universidade Federal Fluminense

VOLTA REDONDA

2021

Ficha catalográfica automática - SDC/BEM
Gerada com informações fornecidas pelo autor

T655p Tomaz de aquino, Vitor
Proposta de Modelos Matemático e Heurístico para Otimizar o Processo de Formação de Carga em Planta de Recozimento em Caixa / Vitor Tomaz de aquino ; Tiago Araújo Neves, orientador ; Weslley Luiz da Silva Assis, coorientador. Volta Redonda, 2021.
127 p. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Volta Redonda, 2021.

DOI: http://dx.doi.org/10.22409/PPG-MCCT.2021.m.08662273777

1. Programação linear. 2. Algoritmo genético. 3. Recozimento (Metalurgia). 4. Produção intelectual. I. Araújo Neves, Tiago, orientador. II. Luiz da Silva Assis, Weslley, coorientador. III. Universidade Federal Fluminense. Escola de Engenharia Industrial e Metalúrgica de Volta Redonda. IV. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

Proposta de Modelos Matemático e Heurístico para Otimizar o Processo de Formação de Carga em Planta de Recozimento em Caixa

Vitor Tomaz de Aquino

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Modelagem Computacional e Pesquisa Operacional.

Aprovada por:

Prof. Tiago Araújo Neves, D.Sc. / MCCT - UFF
(Orientador)

Prof. Antônio Iacono, D.Sc. / DEQUI-USP

Prof. Celso Luiz Moraes Alves, Dr.-Ing. / UFF

Prof. Eliane da Silva Christo, D.Sc. / MCCT - UFF

Prof. Giselle de Mattos Araújo, D.Sc. / UFF

Prof. Puca Huachi Vaz Penna, D.Sc. / DECOM - UFOP

Volta Redonda, 29 de Março de 2021.

Para minha família, Catiele, Vitória e Theo!

Agradecimentos

Agradeço primeiramente a Deus pelas bençãos e alegrias concedidas ao longo da minha vida sendo Este a base para eu seguir adiante!

A minha família, em especial a minha esposa Catiele S. de O. Aquino, que está comigo em todos os momentos da vida e me presenteou com nossos filhos, Vitória S. de Aquino e Theo Santos de Aquino, este ainda dentro de seu ventre. A minha mãe Ana Lúcia de Jesus pelo apoio, incentivo e paciência nos momentos em que estive ausente.

A todos os amigos de classe e trabalho que me apoiou durante esta importante fase da vida.

Ao amigo Moysés D. Silva que ao longo dos anos vêm contribuindo para carreira pessoal, acadêmica e profissional, cuja capacidade de ensinar e inspirar na busca da solução de problemas me fez acreditar que muitos desafios pudessem ser superados.

Meus sinceros agradecimentos aos orientadores Tiago Araújo Neves e Weslley Luiz da Silva Assis pela sua dedicação e paciência durante as aulas e na orientação deste trabalho. Sempre demonstraram fazer o melhor para meu crescimento pessoal e acadêmico.

Meus agradecimentos à UFF - EEIMVR, todo o corpo docente e apoio pois colaboraram para todo o desenvolvimento.

Por fim, agradeço a todos que direta ou indiretamente fizeram parte das etapas da elaboração desta pesquisa.

Resumo

Investiga-se neste trabalho um problema de alocação em lote que afeta as decisões diárias operacionais de uma planta de recozimento em caixa (RCX) em indústria siderúrgica. O problema consiste em avaliar as bobinas disponíveis no pátio de entrada do RCX e decidir quais delas devem ser alocadas para formar cargas. O objetivo é maximizar o peso da carga selecionada para recozimento, analisando simultaneamente o tempo das bobinas em estoque, visando a redução das perdas por oxidação nas bobinas mais antigas. Características especiais de processo exigem que as bobinas selecionadas sejam todas da mesma especificação, ou seja, para cada carga formada não pode haver bobinas de diferentes tipos. Para resolução do problema utilizaram-se dois métodos. No primeiro método propõe-se um modelo matemático baseado no Problema das Múltiplas Mochilas (PMM). Nesta abordagem foi acoplada ao modelo base as restrições de disjunção que trata o problema de agrupamento de bobinas de mesma especificação, além das restrições adicionais do processo. O segundo método é baseado em algoritmo genético e busca-se solucionar instâncias de dimensões maiores dos problemas em tempo razoável. Um método eficiente de criação da população inicial e dois novos processos de mutação com busca guiada são implementados computacionalmente. Experimentos foram realizados para ajustar os parâmetros ideais de funcionamento do algoritmo genético além da implementação de controle do reinício da população através de medição da sua diversidade. Demonstrou-se a partir dos resultados que ambos métodos são capazes de resolver o problema. O método exato tem limitação quanto ao tempo de resolução em função do tamanho da instância. O algoritmo genético atingiu resultados ótimos globais, e nas maiores instâncias ficou até 6% do ótimo global, além disso, em todas as simulações que o método exato não encontrou solução o algoritmo genético gerou solução com tempo de resolução de poucos segundos. Conclui-se que o modelo matemático e o algoritmo genético implementados são capazes de solucionar problemas reais gerando bons resultados para o processo de formação de carga em planta de recozimento em caixa.

Abstract

This paper investigates a batch allocation problem that affects the daily operational decisions in a steel industry batching annealing process (BAP). The problem is to evaluate the coils available in the BAP inventory and decide which coils should be selected to form loads. The goal is to maximize the weight of load and prioritize oldest coils in stock to reduce the possibility of oxidation in the oldest coils. In addition, respecting physical process constraints. Special process characteristics require that the selected coils to form load must have same specification, i.e. for each load formed there can be no coils of different types. Two methods were used to solve the problem. In the first method, a mathematical model based on the Multiple Knapsack Problem (MKP) is proposed. In this approach, variant of the standard knapsack problem with special disjunctive constraints were used to solve the problem of clustering coils of the same specification. The second method is based on genetic algorithm and aims to solve larger problems in reasonable time. An efficient method of creating the initial population and two new mutation processes with guided search are implemented computationally to escape the evolutionary process from optimal locations. Experiments were carried out to adjust the ideal genetic algorithm parameters, in addition, a population restart control was implemented by measuring its diversity. It was demonstrated from the results that both methods are able to solve the problem. The exact method has limitation in the resolution time depending on the instance size. The genetic algorithm achieved optimal overall results, and in the largest instances was at 6% of the global optimal, in addition, in all simulations the genetic algorithm found solutions that exact method couldn't in resolution time of few seconds. It is concluded that the mathematical model and genetic algorithm implemented can solve real batch allocation problem in BAP with good results.

Palavras-chave

1. Problema das Múltiplas Mochilas
2. Problema de Programação Linear Inteira
3. *Branch and Bound*
4. Algoritmo Genético
5. Recozimento em Caixa

Glossário

AG	:	Algoritmo Genético
BAP	:	*Batch Annealing Process*
B&B	:	*Branch and Bound*
BQ	:	Bobina a Quente
BF	:	Bobina a Frio
Carga	:	Conjunto de bobinas selecionadas para recozimento
DCKP	:	*Disjunctively Constrained Knapsack Problem*
DDE	:	*Discrete Differencial Evolution*
EB	:	Espaço de Busca
EEIMVR	:	Escola de Engenharia Industrial Metalúrgica de Volta Redonda
FO	:	Função Objetivo
GAP	:	*Generalized Assignment Problem*
H_2N_2	:	Hidrogênio e Nitrogênio
HPH	:	*High Performance Hydrogen*
KP	:	*Knapsack Problem*
LTF	:	Laminador de Tiras a Frio
LTQ	:	Laminador de Tiras a Quente
NP	:	Tempo Polinomial não determinístico
PCC	:	Problema da Cobertura de Cliques
PCG	:	Problema de Coloração de Grafos
PLI	:	Programação Linear Inteira
PLI	:	Programação Linear Inteira Mista
PM	:	Problema da Mochila
PMI	:	Problema da Mochila Inteiro
PMMCI	:	Problema das Múltiplas Mochilas com Conflito de Itens
PMM	:	Problema das Múltiplas Mochilas
PMR	:	Problema da Mochila Restrito
PPL	:	Problemas de Programação Linear

Glossário

PRVJT	:	Problema de Roteamento de Veículo com Janela de Tempo
Rand()	:	Função produz um número aleatório no intervalo 0 - 1
RCX	:	Recozimento em Caixa
Tbae	:	Tempo de Bobina Aguardando em Estoque
UFF	:	Universidade Federal Fluminense
UGGA	:	*Undominated Grouping Genetic Algorithm*
VNS	:	*Variable Neighborhood Search*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Nas indústrias de maneira geral muitos processos têm oportunidades para aplicação de técnicas de otimização com o objetivo de reduzir custos, aumentar produtividade ou produção. Ademais, com a evolução computacional possibilitou a criação de algoritmos de otimização para tornar eficientes diversos processos industriais, de engenharia, ciência, etc. Neste sentido, esta dissertação tem foco voltado para otimização de uma fase do processamento de chapa de aço plano em indústria siderúrgica, mais especificamente na formação de carga em planta de recozimento em caixa.

Devido à globalização, o mercado financeiro mundial quebra fronteiras e o setor siderúrgico é cada dia mais afetado devido à concorrência. A China hoje é o pais com maior fabricação de aço no mundo e o Brasil ocupa modesta nona posição conforme (STEEL, 2019). Se os processos nacionais não estiverem otimizados, certamente o aço fabricado na China chegará ao país com menor custo desestimulando o processo de fabricação do aço.

Segundo (AÇO BRASIL, 2019) o ferro, como matéria-prima mineral, pode ser encontrado em toda a crosta terrestre em diferentes formas químicas e/ou mineralógicas. Já o aço como liga metálica Fe-C é encontrado ao redor do globo nas mais diversas aplicações em incontáveis produtos com diferentes níveis de processamento da liga metálica. Abordando especificamente a fabricação de bobinas de aço plano em processo siderúrgico, este consiste em etapas de processamento a quente e dentre as principais estão: o Alto Forno, Aciaria e a Laminação à Quente que finalmente resulta na fabricação de bobinas de aço plano. Em seguida ocorre o processo de Laminação de Tiras à Frio (LTF), que visa dar melhor acabamento superficial e gerar maior valor agregado ao produto. Cada bobina

produzida tem suas próprias características, tais como, peso em toneladas, largura e espessura em milímetros e especificação do aço como por exemplo o aço carbono, inoxidável, etc. O que determina tais características é a aplicação final do produto.

As bobinas produzidas por um LTF são transportadas e posicionadas em um pátio de entrada de planta de Recozimento em Caixa (RCX). Cada indústria siderúrgica tem sua logística definida e, dependendo desta logística, as bobinas podem levar mais ou menos tempo para estarem disponíveis no pátio de entrada da planta de recozimento para seleção e agrupamento. O agrupamento em lote é caracterizado como sendo um processo do tipo batelada baseado em critérios de acordo com a literatura (BORGES; DALCOL, 2002). O foco desta pesquisa está nesta etapa do processo, que trata da seleção das bobinas para a formação de carga e posterior recozimento. O termo carga deve ser entendido como uma pilha de bobinas, ou seja, a carga é composta de um conjunto de bobinas empilhadas uma sobre as outras.

Para aumentar a competitividade deste processo, a formação de carga deve ser a mais otimizada possível. A formação de carga consiste em selecionar as bobinas ideais em estoque para aumentar o produtividade do processo atendendo todas a restrições físicas e de qualidade. Conforme pode ser observado na Figura 1.1 tem-se três exemplos de formação de carga como forma ilustrativa e didática.

A Figura 1.1*a*, é um esquemático de formação de carga com boa ocupação do espaço interno do recipiente que recobre as bobinas, denominado abafador. Observa-se que a ocupação está próxima da largura e altura máxima. A Figura 1.1*b*, contém a representação da formação de uma carga que ultrapassa a capacidade do abafador, violando a restrição de altura máxima. Já a Figura 1.1*c*, apresenta uma carga formada não otimizada pois o espaço interno útil do abafador poderia ser melhor utilizado. Um outro fator a ser analisado é o peso da carga formada, que deve ser otimizado a fim de tornar o processo mais rentável. Contudo, o equipamento tem restrições de peso máximo tolerado, o que também precisa ser observado no processo de formação da carga.

Além das restrições de peso e altura, tem-se a restrição que trata da especificação do material. Neste tipo de processo só é permitido agrupar na mesma carga bobinas de mesmo tipo, ou seja, da mesma especificação. Fazendo uma analogia com a Figura 1.1, se a especificação da bobina for representado por sua cor, então na mesma carga só podem haver bobinas de mesma cor (verde por exemplo no caso da Figura 1.1*c*). É importante mencionar que no programa de produção, montado pelos engenheiros de processo, se encontram as especificações que podem ser agrupadas, logo esta é uma informação conhecida

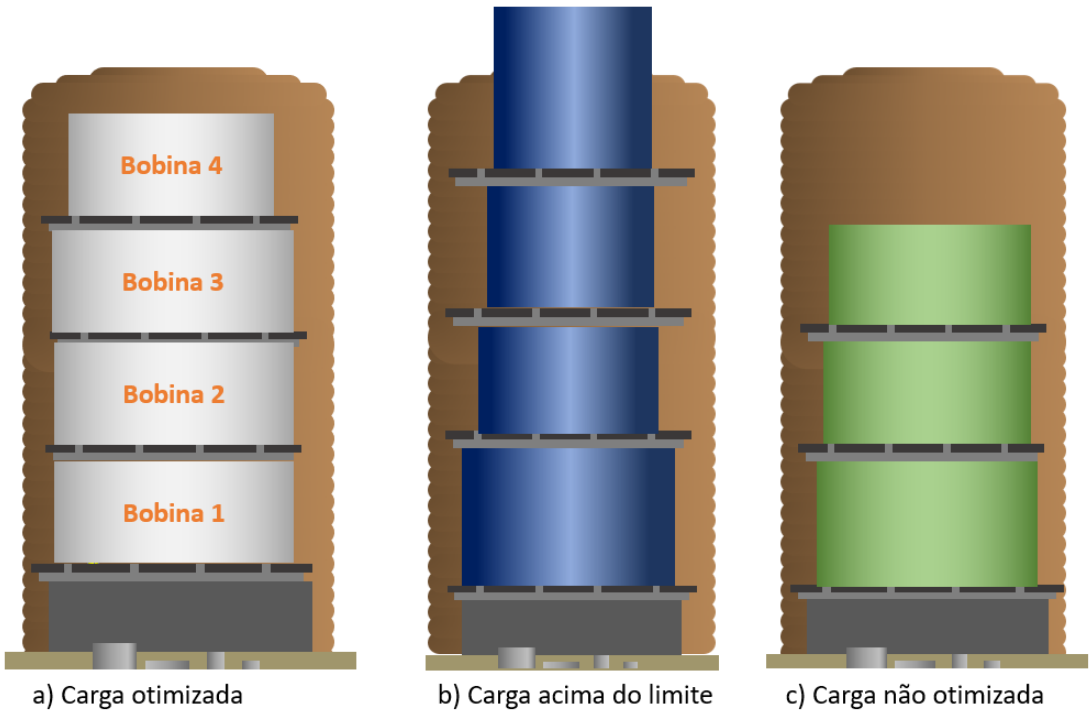

Figura 1.1: Esquema ilustrativo de geração de três cargas com diferentes combinações de bobinas. As cores representam várias especificações das bobinas mas que não podem ser misturadas.

FONTE: Adaptado de (TENOVA, 2020)

e dada como parâmetro para o processo de recozimento em caixa.

Um importante requisito está no tempo em que as bobinas ficam aguardando em estoque até serem direcionadas para a produção. Por se tratar de bobinas de chapa de aço carbono, estas ficam armazenadas em galpões aguardando o direcionamento para produção no equipamento. Porém, este período de espera é crucial para a garantia da qualidade, uma vez que há risco de oxidação caso o tempo de espera em estoque seja demasiadamente longo. Cada empresa neste setor tem seu cronograma de controle e tempo de armazenamento que pode variar por tipo de processo, local e região do armazenamento. Vale ressaltar que nesta etapa do processo, as bobinas não recebem embalagens para proteção contra exposição à agentes externos, somente em caso de exposição à chuva durante transporte, se este for feito fora dos galpões. Desta forma, em caso de defeitos de qualidade gerado por oxidação o prejuízo pode ser irreversível.

Considerando estes e outros cenários, busca-se a otimização do processo de formação de carga de forma a encontrar um ótimo agrupamento de bobinas, atendendo todos os requisitos de qualidade e restrições do processo, eliminando desperdícios e erros que

as operações manuais podem ocasionar. Neste sentido, este trabalho emprega técnicas de Pesquisa Operacional (PO) que, segundo (LOPES; RODRIGUES; STEINER, 2003), é uma área do conhecimento fortemente interdisciplinar, voltada ao desenvolvimento de modelos matemáticos e algoritmos para a resolução de problemas reais complexos. Diversos estudos em PO são realizados e têm sido utilizados com sucesso para a obtenção de soluções otimizadas nos mais variados contextos de problemas (LOPES; RODRIGUES; STEINER, 2003) e (CASADO et al., 2020).

O problema de agrupamento de bobinas descrito é pouco explorado na literatura e tem similaridades com o Problema das Múltiplas Mochilas (PMM). A solução para o PMM é aplicado em diversas situações reais na indústria e levam pesquisadores buscarem formas eficientes para solucionar grandes instâncias de problemas reais.

Tendo-se o detalhamento e as necessidades deste problema propõe-se a construção de modelos matemático para otimizar o processo de formação de carga em uma planta típica de RCX para atender as seguintes necessidades:

- Formar múltiplas cargas com maior peso possível.
- Priorizar as bobinas mais antigas em estoque.
- Respeitar o limite máximo de altura.
- Respeitar o limite máximo de peso.
- Só haver bobinas de mesma especificação na carga formada.
- Formar carga acima de um peso mínimo.
- Formar carga com um número mínimo de bobinas.

1.2 Justificativa

Ao pesquisar a literatura, não foram encontrados trabalhos que abordam o problema descrito, com todos os detalhes considerados, no setor desta pesquisa. Os trabalhos mais atuais encontrados com a mesma característica de otimização em processo siderúrgico estão localizadas em sua maior parte no setor de lingotamento contínuo, então a ausência ou a carência de informação gerou a necessidade de um estudo analítico para otimizar o processo de formação de carga em planta de recozimento em caixa.

1.3 Objetivo

1.3.1 Objetivo Geral

Propor soluções, matemática baseada em programação linear inteira e aproximada por algoritmos genético para otimizar o processo de formação de cargas atendendo às restrições impostas pelo processo.

1.3.2 Objetivos Específicos

- Elaborar um modelo matemático para a otimizar a formação de cargas em planta de recozimento em caixa utilizando como base o Problema das Múltiplas Mochilas.
- Propor e analisar métodos aproximados para os casos onde o modelo matemático não apresentar bom rendimento.
- Realizar comparações entre métodos: aproximado e exato.

1.4 Delimitações do Estudo

Este estudo foi proposto com o intuito de tratar o problema de formação de carga em empresa do setor siderúrgico que necessite gerar um limite de múltiplas cargas com números limitado de bobinas. Delimita-se a utilizar as instâncias geradas com base em valores próximos em ordem de grandeza aos valores de um processo real siderúrgico, ou seja, os dados deste trabalho não são os dados reais de qualquer empresa siderúrgica. Neste trabalho não é considerado gerar um modelo para cálculo de ciclo térmico ou previsão das temperaturas de aquecimento e resfriamento para cada carga formada. Também considera-se para otimização do processo utilização de bobinas de chapa de aço carbono plano, não levando-se em consideração vergalhões, tarugos, etc.

1.5 Estrutura da Dissertação

Este trabalho está estruturado em cinco capítulos contendo a descrição sistemática da pesquisa realizada. No Capítulo 2 apresenta-se a revisão bibliográfica do processo siderúrgico, alguns trabalhos de otimização atualizados na literatura até a data de criação desta pesquisa cuja aplicação está com foco em otimização de processo siderúrgico.

Demonstra-se o referencial teórico do método exato com ênfase no problema das múltiplas mochilas e por fim a revisão do método aproximado baseado em algoritmos genéticos. No Capítulo 3 tem-se a metodologia com a proposta de modelo matemático final aplicado no problema de formação de carga em planta de recozimento em caixa utilizando como base o problema das múltiplas mochilas com conflito de itens. Adicionalmente, a elaboração do banco de dados e análise das instâncias criadas para os testes. Também a descrição detalhada do algoritmo genético proposto como método aproximado. No Capítulo 4 são apresentados dois importantes resultados, sendo um para o método exato e outro para o método aproximado. Nos resultados do método exato, os testes de formação de cargas, a comparação entre a formação de carga em manual, formação de carga n vezes e várias cargas simultaneamente para comprovação do erro de aproximação, além da análise dos tempos de execução e limites computacionais. Para o método aproximado compara-se os resultados e tempos de execução em relação ao método exato. No Capítulo 5, estão as conclusões, bem como, as sugestões para trabalhos futuros. Ao final, encontram-se as referências bibliográficas.

Capítulo 2

Referencial Teórico

Este capítulo é composto por cinco seções e contém a revisão da literatura para composição desta pesquisa.

2.1 Processo Siderúrgico

O processo produtivo do aço é desenvolvido a partir das matérias-primas básicas como o minério de ferro e carvão. Com a utilização de alto-fornos entre outros equipamentos de grande porte, busca-se obter uma produção elevada com segurança e qualidade. A energia predominante para este tipo de usina é o carvão. Após o refino na aciaria, o aço, na forma líquida segue para o lingotamento onde é solidificado em placas que posteriormente são laminadas a quente e a frio, respectivamente, originando os produtos finais, aços longos, perfis, barras, vergalhões, canos, arames, trilhos, pregos e os denominados aços planos (AÇO BRASIL, 2019). Um breve resumo deste processo é apresentado nas próximas seções.

2.1.1 Alto Forno, Aciaria e Lingotamento

A partir do minério de ferro e do coque, o Alto Forno produz ferro gusa e escória líquidos, sendo o primeiro direcionado à Aciaria para refino e tratamento químico que dá origem ao aço, e a segunda encaminhada para reaproveitamento como co-produto.

A Aciaria é a unidade que começa definir a composição química de aço e neste trabalho é denominado "especificação". Existem máquinas e equipamentos voltados para o processo de transformar o ferro gusa proveniente dos alto fornos em diferentes tipos de aço. O principal destes equipamentos é o conversor, que é um tipo de forno, revestido com

tijolos refratários e que transforma misturas de ferro gusa e sucatas em aço. Uma lança sopra oxigênio em alta pressão para o interior do forno, produzindo reações químicas que separam as impurezas, como os gases e a escória.

A Figura 2.1 foram apresentados alguns detalhes do fluxo produtivo do processo siderúrgico.

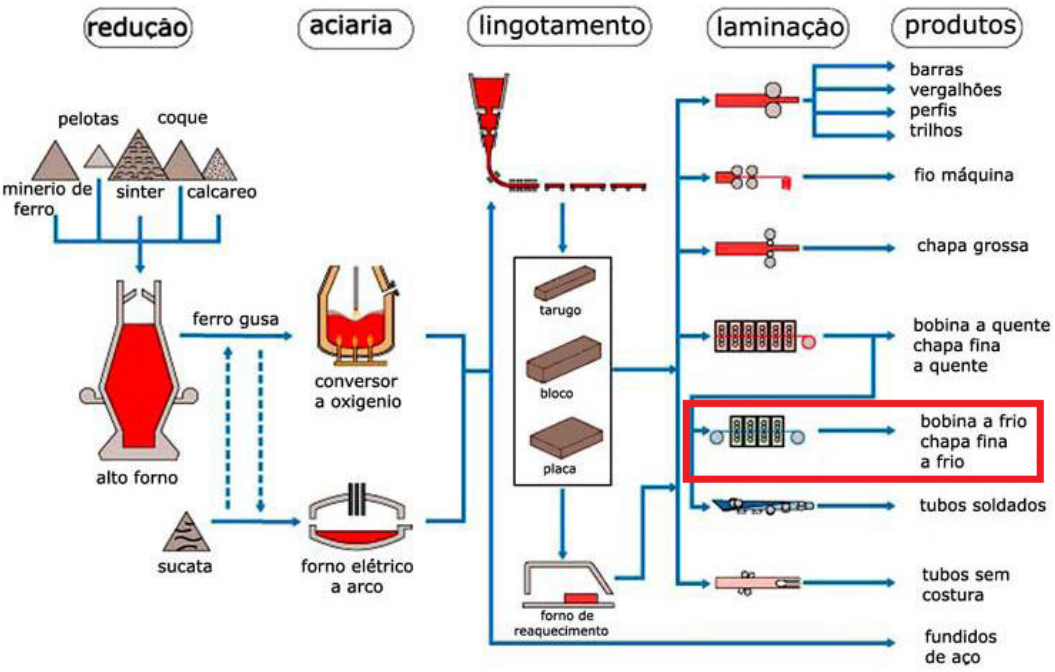

Figura 2.1: Fluxo de produção de bobinas de aço plano.
FONTE:(SCHEID, 2020)

O Lingotamento Contínuo transforma o aço do seu estado líquido para o estado sólido, sendo o aço vazado através de um molde aberto e móvel e solidificado através do seu resfriamento. O aço é moldado, ou seja, é definido no lingotamento a largura final da chapa formando um "tarugo", solidificado de maneira progressiva da superfície para o núcleo do veio. A forma é cortada em comprimentos pré-definidos em função da faixa de peso dos produtos finais, buscando-se maximizar a produtividade dos processos subsequentes. Nesta fase é definida mais uma característica da bobina usada neste trabalho que é a largura do material e também o peso aproximado da bobina (SCHEID, 2020).

2.1.2 Laminação

Na laminação o processo do Laminador de Tiras a Quente (LTQ), consiste na deformação a quente (alta temperatura) do aço através da sua passagem entre cilindros em

vários passes, até atingir a dimensão final do produto a quente. Ao sair da última "cadeira de laminação", a chapa é resfriada com água até uma temperatura pré-determinada e enrolada na forma de bobina de aço. O processo pode ser separado em cinco estágios, sendo eles: reaquecimento, desbaste, acabamento, resfriamento e bobinamento (SCHEID, 2020). Processo de decapagem é por fim realizado até a Laminação a Frio.

A Laminação a Frio é empregada para produzir, a partir de bobinas a quente, folhas e tiras com acabamento superficial superiores. Também a taxa de deformação aplicada nas bobinas garantem espessuras ainda menores quando comparadas com as bobinas produzidas pela laminação a quente. Importante característica da redução de espessura deste processo é o encruamento resultante que é aproveitado para dar maior resistência ao produto final e fornecer energia interna para o posterior processo de recozimento. Trens de laminadores quádruos de alta velocidade e cinco cadeiras são exemplos de laminadores utilizados para a laminação a frio. Normalmente esses trens de laminação são concebidos para terem tração avante e a ré. O objetivo principal de um Laminador de Tiras a Frio (LTF) é reduzir a chapa para espessura desejo do cliente, ou seja, a espessura final da chapa. A redução total atingida por laminação a frio geralmente varia de setenta e noventa por cento (SANTOS, 2017). Quando se estabelece o grau de redução em cada passe ou em cada cadeira de laminação, deseja-se uma distribuição tão uniforme quanto possível nos diversos passes sem haver uma queda acentuada em relação à redução máxima em cada passe. Normalmente, a porcentagem de redução menor é feita no último passe para permitir um melhor controle do aplainamento, bitola e acabamento superficial.

Os pesos das placas são definidos no Lingotamento e são calculados de forma que as bobinas submetidas aos processos posteriores cheguem dentro da faixa de peso desejada. No LTF a dimensão pode ser redefinida, de forma a diminuir o tamanho da bobina, visto que o fluxo de produção tende a ter equipamentos menores no sentido final das linhas. Assim, bobinas de diferentes pesos, seja por problemas de processo ou diversificação de produtos por clientes, acabam tendo em um LTF, suas características físicas redefinidas. Desta forma, os equipamentos posteriores se planejam ou geram o próprio programa de produção conforme matéria prima disponível. Está destacado com um retângulo em vermelho na Figura 2.1 o LTF e o seu fluxo de produção.

2.1.3 Patio de Armazenamento das Bobinas

As bobinas produzidas em um LTF são armazenadas em seu patio de saída e direcionadas para linhas de produção afim de realizar os tratamentos necessários para entrega

ao cliente final. Para o caso das bobinas que saem de um LTF cujo fluxo de produção tem destino uma planta de Recozimento em Caixa (RCX), as bobinas são transportadas e armazenadas no pátio de armazenamento de bobinas. Algumas das características físicas da bobina são apresentadas conforme Figura 2.2.

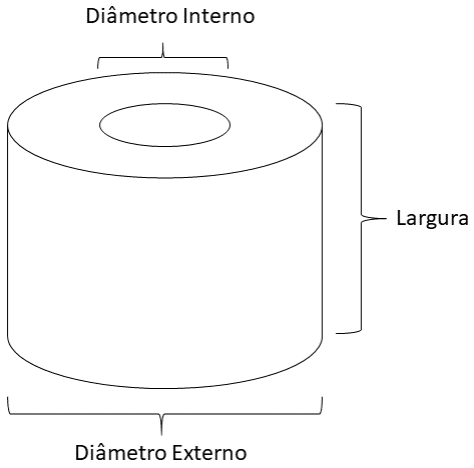

Figura 2.2: Características dimensionais da bobina.
FONTE: Adaptado de (PALMETAL, 2019)

Ao chegarem no pátio de armazenamento, equipamentos transportadores auxiliam na retirada e posicionamento de cada bobina em berços de bobinas. Observa-se que a quantidade de bobinas que chegam no pátio de entrada é na maior parte função da programação de produção das bobinas que saem de um LTF. Também podem haver entregas de bobinas no pátio de armazenamento de um RCX que não sejam direto de LTF o que caracteriza as diferentes qualidades de materiais para processo. A disposição de armazenamento das bobinas é conforme Figura 2.3. Portanto, seguem informações fundamentais das bobinas utilizadas para formação de carga e produção em um RCX.

Figura 2.3: Patio de estocagem na entrada do recozimento em caixa.
FONTE:(PALMETAL, 2019)

- Peso: o peso de cada bobina varia e seu peso máximo é determinado pela característica dos equipamentos subsequentes ao RCX. A limitação de peso do RCX se deve ao somatório dos pesos das bobinas agrupadas na formação da carga. Um fato importante está na limitação de diâmetro externo da bobina, contudo, as regras dos equipamentos anteriores à produção do RCX devem garantir o valor do diâmetro dentro das especificações dos equipamentos.

- Largura: as larguras dos materiais variam de acordo com a necessidade do cliente, sendo que para uma planta típica de recozimento em caixa, não há limitação de largura da bobina, mas há limite de altura da pilha de bobinas. No pátio de entrada da planta, o eixo da bobina está no posicionamento horizontal, mas a bobina deve ser virada para serem empilhadas na base do forno de RCX, e assim gerar a altura da pilha de bobinas sendo esta uma das restrições deste processo o qual é aplicada na proposta do modelo matemático. Detalhe do posicionamento da bobina em uma planta típica de RCX pode ser visto na Figura 2.4.

- Especificação: para o RCX é obrigatório que, em um processo de recozimento, todas as bobinas deste processo sejam do mesmo tipo, ou seja, especificação. Não pode haver, portanto, mistura de diferentes especificações. Esta exigência se deve ao fato de que, ao misturar especificações nem todas as bobinas adquirem as propriedade mecânicas desejadas.

- Tempo máximo de armazenamento: o tempo que a bobina fica aguardando no pátio de entrada para processo é fundamental sob pena de sofrer deterioração e ter sua qualidade afetada.

2.1.4 Planta de Recozimento em Caixa

Segundo (STARCK; MÜHLBAUER; KRAMER, 2005) as primeiras plantas construídas no mundo para processo de recozimento em caixa, tinha sua atmosfera formada por hidrogênio e nitrogênio (H_2N_2) e nos dias atuais usa-se hidrogênio puro em uma atmosfera protetiva para recristalizar e clarear o aço de baixo carbono. Se comparada com antiga planta de recozimento baseado em atmosfera de (H_2N_2), as novas plantas obtêm vantagens de processo em amplos aspectos, dentre os quais pode-se citar uma menor viscosidade e um aumento de condutividade térmica em sete vezes. Sendo assim, há um aumento da transmissão de calor por um fator maior que dois e consequentemente maior performance no aquecimento e resfriamento das bobinas.

Em (SCHEUERMANN, 1995) explica-se as funcionalidades das plantas denominadas por *High Performance Hydrogen (HPH) bell-type batch annealing*, o qual podem ser traduzidas como, Processo de Recozimento em Caixa de Alta Performance a Hidrogênio Puro. Muitos desenvolvimentos no processo ao longo dos anos foram realizados de modo a se obter aumento de produtividade. Seguem algumas conclusões importantes a serem citadas: O processo atual possui uma taxa de aquecimento é 72% maior, a taxa de resfriamento é 490% maior e a produtividade da base é 280% maior em relação ao processo anterior, feito com HN.

Figura 2.4: Planta típica de Recozimento em Caixa.
FONTE:(TENOVA, 2020)

Uma planta de RCX, mostrada na Figura 2.4 é composta por vários equipamentos que em conjunto tornam o processo de recozimento possível. Sistema de utilidade com gases são usados para recozimento e proteção, sistema de recirculação de água para controle de refrigeração e temperatura, válvulas de controle além de todo um sistema de automação

e supervisão de processo. Assim, os principais equipamentos, considerados nesta pesquisa são os seguintes:

- Bases: são utilizadas para apoiar as bobinas submetidas ao processo de recozimento. Cada empresa pode ter um número fixo de bases definido durante fase de projeto para atingir determinada produção.
- Abafadores: após empilhamento das bobinas na base, o abafador é posicionado sobre as bobinas. Sua função é isolar o ambiente interno da bobina com o ambiente de aquecimento (forno).
- Fornos: um exemplo de forno de recozimento está representado pela Figura 2.5 caracterizado por ser de alta convecção. As setas representam o fluxo de hidrogênio em alta temperatura. Utiliza-se gás como combustível para o forno e o abafador isola o ambiente das bobinas e a chama. O forno segue um rigoroso controle de temperatura alimentado por um modelo matemático térmico. Cada especificação do aço tem sua curva de aquecimento e resfriamento. Ao término do ciclo de recozimento o forno é retirado e posicionado em outra base para iniciar novo processo de recozimento. A produção desta planta é definida pelo número de fornos em operação.
- Resfriadores: os resfriadores são posicionados na base logo após a retirada do forno para acelerar o processo de resfriamento das bobinas. Estes podem utilizar como método de resfriamento sopradores de ar ou resfriamento com água.

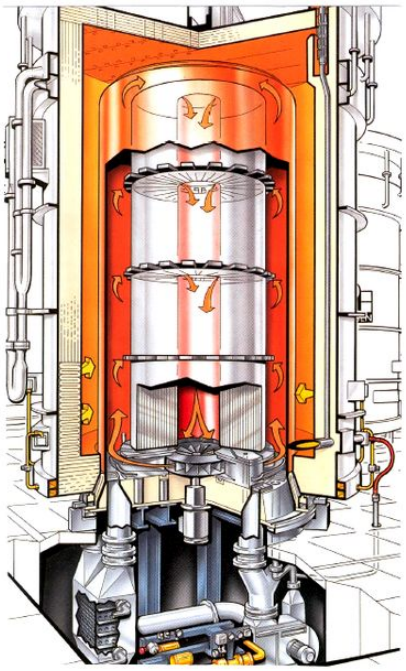

Figura 2.5: Ilustração esquemática de processo de Recozimento em Caixa.
FONTE:(SCHEUERMANN, 1995)

Cada especificação do aço tem um tempo pré-determinado para recozimento (DEUS et al., 2007). Um modelo matemático recebe os dados das bobinas selecionadas e calcula os tempos e taxas de aquecimento afim de garantir que as bobinas atinjam as propriedades mecânicas. Nesta pesquisa não é considerado gerar um modelo para o cálculo do ciclo térmico de recozimento. Tipicamente, os modelos para cálculo de ciclos térmicos para recozimento em caixa, levam em consideração as características físicas das bobinas selecionadas para a formação da carga. Os atributos das bobinas utilizadas como entrada no modelo são: peso total da carga, espessura da menor e largura da maior bobina além da especificação do aço.

Neste trabalho busca-se resolver um problema de otimização para formação de carga e não elaborar um modelo de cálculo dos ciclos térmicos, sendo assim, na próxima seção é feito um resumo dos trabalhos relacionados a otimização aplicado em processo siderúrgico com elaboração a partir de 2015.

2.2 Problemas Relacionados

Trabalhos de pesquisa entre 2016 e 2020 abordam a otimização de parte dos processos siderúrgicos para fabricação de aço utilizando método exato e aproximado e são

apresentados a seguir:

Em (HADIDI; MOAWAD, 2017) tratam de gerenciar o quanto a diversificação de produtos impactam na produção de uma usina siderúrgica no qual o fluxo é sequencial, ou seja, a saída da produção de um equipamento é o início para o outro. A produtividade dos equipamentos é influenciada pela diversificação de material e isso define a capacidade diária, mensal e anual de produção. Devido à expansão de uma planta de recozimento em caixa de uma usina na Arábia Saudita houve desabastecimento no fluxo produtivo. São seis os equipamentos citados, Linha de Decapagem Ácida, Laminador Reversível, Planta de Recozimento em Caixa, Laminador de Encruamento, Linha de Galvanização Contínua e Linha de Corte e Pintura de Chapa. Os autores propõe uma formulação matemática baseado em programação linear inteira (PLI) para resolver o problema de organizar o fluxo de produção dos equipamentos baseado na diversificação dos materiais de forma a eliminar paradas indesejadas da cadeia de produção. A função objetivo maximiza a produção de chegada em cada equipamento sujeito as principais restrições: não ultrapassar a produtividade de cada equipamento, o total de produção dos seis equipamentos deve ser maior que a capacidade de produção mínima planejada de cada equipamento para garantir certa utilização, ter 20% de flexibilização no agrupamento de materiais de mesma característica para dar certa flexibilidade ao modelo e por fim, o fluxo de produção do equipamento anterior deve ser maior ou igual ao do próximo equipamento. O modelo proposto maximiza o fluxo de chegada de material em cada equipamento e atendeu as restrições impostas não excedendo o limite de produção por mês, evitando assim paradas desnecessárias. A validação do modelo foi feito com os dados do ano anterior.

Segundo (GUO; TANG, 2019) propõe resolver o problema de replanejamento de ordem de produção em indústria siderúrgica comparando o método de sequenciamento manual, um método exato e um método heurístico. Os autores consideram os impactos que podem acontecer na linha de produção quando se faz necessário replanejar a produção, como por exemplo, os desvios em termos de produção entre o novo e antigo sequenciamento, as preparações necessárias no equipamento para executar a nova ordem de produção, além de recomendações mínimas para gerar a nova sequencia de produção. O modelo matemático proposto foi baseado em programação linear inteira mista, cita que faz parte da classe dos problemas NP-Difícil e sua implementação foi no software CPLEX. Por se tratar de um problema prático uma solução deve ser apresentada em tempo razoável, ou seja, em poucos minutos, então foi proposto um novo algoritmo de Evolução Diferencial Discreta (DDE - *Discrete Differencial Evolution*), o qual propõe um novo processo de mutação, *crossover* e busca local. Os resultados demonstraram que o algoritmo é eficaz

para resolução do problema com grandes instâncias aplicado em usina siderúrgica.

Em (CASADO et al., 2020) aborda-se o agrupamento de placa de aço compatíveis para a utilização eficiente ou otimizada das facilidades e equipamentos em LTQ de uma usina siderúrgica no norte da Espanha para produção de bobinas de aço. Objetiva-se reduzir o tempo de mudança, ao se trocar a fabricação de um produto por outro, de formar a atender todos os requisitos de produção com mínimo custo. Estudou-se encontrar o melhor agrupamento de bobinas com características similares para minimizar os custos de produção dos pedidos. Também aborda a minimização do custo associado a redução do número de partidas e paradas dos equipamentos, sendo estes custos associados a ajustes obrigatórios para produção de diferentes produtos. Alguns benefícios da pesquisa são apontados que vai além da redução de custo, a minimização de acidentes pelo menor manuseio e redução de armazenamento de materiais. A pesquisa demonstrou que retirando as questões de custo associados a produção do aço, este é um problema de agrupamento o qual a versão simplificada é equivalente ao problema da cobertura de cliques (PCC), também chamado na literatura de partição em cliques (*clique-partitioning problem*). Desta forma, o que é chamado de compatibilidade de bobinas neste problema, na literatura é chamado de clique na teoria de grafos. Também os autores apontam a semelhança deste caso com o problema de coloração de grafos (PCG). Foi proposto um modelo matemático para o problema abordando a relação existente entre o PCC e o PCG. Adicionalmente o desenvolvimento de um método heurístico baseado em busca tabu para encontrar solução aproximada com tempo de resolução razoável. Experimentos computacionais foram realizados para comparar os resultados do método heurístico desenvolvido com o método exato baseando-se na implementação do PCC e PCG utilizando o software CPLEX 12.8. Alguns atributos das bobinas são utilizados para mapear ou fazer tal agrupamento tais como, largura, peso, grau do aço e data desejo. Os experimentos demonstraram o mérito do algoritmo desenvolvido e finalmente sugerem que a solução demonstra ser flexível para absorver novas considerações de custos além de diferentes necessidades como função bi-objetivo.

O trabalho de (SU, 2016) propõe integrar o problema de programação da produção de ferro fundido e fabricação de aço englobando os equipamentos desde a formação da placa ou tarugo até a produção de bobinas em um Laminador de Tiras a Quente (LTQ). A otimização busca o melhor agrupamento de materiais para processo de forma a reduzir o consumo de energia e os custos de produção, enquanto ao mesmo tempo melhora o lucro. O autor propõe a modelagem matemática baseada no Problema de Roteamento de Veículos com Janela de Tempo (PRVJT). O PRVJT consiste no atendimento de um

conjunto de consumidores por intermédio de uma frota de veículos. Estes veículos partem de um ou mais pontos denominados depósitos e a janela de tempo trata do tempo de atendimento de cada consumidor. Este é um problema de otimização combinatória NP-Difícil e para encontrar boa solução em tempo razoável foi proposto um módulo difuso adaptativo integrado ao algoritmo genético (AG) tradicional. Assim, buscou-se reduzir o tempo computacional e a convergência prematura do AG. A codificação do AG foi inspirada na otimização por enxame de partículas e o módulo da lógica difuso adaptativo é utilizado para calcular a probabilidade de mutação. Os resultados demonstraram que o algoritmo genético difuso é um método eficiente e pode ajudar as empresas encontrarem bons planejamentos de produção. Também o tempo para encontrar uma solução foi em média 0,17 minutos e que pode ser usado para encontrar solução de problemas ainda maiores podendo portanto ser usado em processo produtivo real.

Em (LONG et al., 2018) aborda o problema de otimização combinatória para determinação integrada do agrupamento de panelas de aço e sequência de produção em lingotamento contínuo além de informar o momento início de cada corrida de produção. Neste processo a produção vem das panelas da aciaria com aço líquido e são lingotados para formar placas e por fim seguir o fluxo produtivo para um LTQ. Porém, ao realizar a sequência de produção, é necessário, no lingotamento contínuo, conhecer o número de lingotamento (corridas) a serem processadas em cada lingotamento contínuo, que neste caso real são um total de cinco, além de quais são as panelas agrupadas e sua sequência. Por fim, o início de cada corrida para formar placas (tarugo). O processo de lingotamento contínuo permite fazer lingotamento de aços de mesma especificação e em caso de mudança de especificações, custos são adicionados ao processo de preparação. Quando placas com diferentes espessuras precisam ser processadas, o lingotamento precisa ser parado por horas para ajustes da espessura do molde e para evitar paradas para estes ajustes, procura-se agrupar a produção tanto por espessura quanto largura. No caso da largura os ajustes são realizados nas máquinas somente do material mais largo para mais estreito para evitar problemas relacionados a segurança. Porém, existe um limite máximo para as mudanças de largura do molde que deve ser obedecido. Também custos com consumo de combustível são adicionados a cada parada de processo, seja por qualquer ajuste da máquina ou outras necessidades operacionais. Desta forma a função objetivo do problema visa minimizar o salto de largura da placa, minimizar os custos de produção para cada nova corrida com nova especificação e minimizar o número de placas em estoque de modo a produzir o mais próximo da data desejo no LTQ. Este problema é normalmente abordado pela literatura usando técnicas criadas para o problema do caixeiro viajante, porém neste trabalho o

autor utilizou a Pesquisa em Vizinhança Variável (*Variable Neighborhood Search*, VNS) em que a variável de decisão foi codificada da seguinte forma: busca-se encontrar o número de corridas ou placas em cada lingotamento contínuo e o número de panelas do lingotamento. Uma vez encontrado a seleção da carga e a sequencia de produção, um novo problema é proposto e busca decodificar, o início de cada lingotamento. Um modelo matemático baseado em programação linear inteira mista foi proposto e implementado no CPLEX. Também o algoritmo VNS foi desenvolvido em C++ e comparado com a solução ótima através do CPLEX para instâncias menores. Finalmente, uma comparação entre o método proposto e o método de decisão manual é realizado. Os resultados demonstraram a eficiência do novo método proposto baseado no algoritmo VNS e como proposta futura os autores propõe o desenvolvimento de uma aplicação.

Em (KOBLASA; VAVROUSEK; MANLIG, 2016) observa que devido a grande concorrência do mercado, empresas aumentam a diversificação de produtos ou materiais, com isso, propõe-se solucionar o problema de planejamento não só para materiais homogêneos mas também programar materiais com fluxo de trabalhos em máquinas levando-se em consideração agrupar lotes heterogêneos, ou seja, programar famílias de materiais compatíveis e incompatíveis dentro de certos critérios. É um trabalho de alocação inspirado no problema de empacotamento tridimensional onde as peças com diferentes formas, influenciam o tempo e o custo total de produção. É proposto no trabalho agrupar o processamento de lotes heterogêneos na indústria do aço com foco em operações de tratamento térmico, como por exemplo o recozimento. A razão apresentada foi a possibilidade de processar famílias de materiais diferentes ao mesmo tempo. Uma das principais tarefas é modelar o tempo de processamento da operação de tratamento térmico juntamente com a minimização dos custos de operação, ou seja, minimizar o número de pacotes, levando em consideração a data desejo do cliente. Para alocação dos materiais, é dado um conjunto n de itens de forma retangular, cada um caracterizado pela largura l_j, altura a_j e profundidade p_j (j = 1, ..., n), e um número ilimitado de pacotes tridimensionais idênticos (caixas) com largura L, altura A e profundidade P. Assumiu-se que os itens podem ser girados dentro de todos os eixos (x, y, z). São propostos dois algoritmos heurísticos, o algoritmo construtivo e algoritmo evolucionário de chave aleatória simples. Os resultados demonstraram que o algoritmo evolucionário de chaves aleatórias foi promissor para aplicação em ambiente industrial e seu resultado no geral apresentou redução na função objetivo de 64% em alguns casos em comparação com o algoritmo construtivo.

Os trabalhos apresentados demonstram que no período abordado em comum estão as abordagens dos problemas de otimização por programação linear inteira através de

desenvolvimento de modelos matemáticos. Observa-se que foram utilizados na maior parte dos casos, dois métodos sendo que o método exato foi usado para encontrar o ótimo global e posterior comparação com o método aproximado, desta forma, um resumo das pesquisas atuais está na Tabela 2.1.

A primeira coluna da tabela contém a referência dos autores enumeradas conforme abaixo.

1. (HADIDI; MOAWAD, 2017)
2. (GUO; TANG, 2019)
3. (LONG et al., 2018)
4. (KOBLASA; VAVROUSEK; MANLIG, 2016)
5. (CASADO et al., 2020)
6. (SU, 2016)

A coluna problema trata-se de resolver problemas de sequenciamento ou agrupamento. A terceira coluna denominada processo indica se é do tipo batelada ou processo contínuo. Por fim, se os algoritmos utilizados são exatos ou aproximados em conjunto a indicação da implementação, o qual Programação Linear Inteira (PLI), Programação Linear Inteira Mista (PLIM) e *Variable Neighborhood Search* (VNS).

Tabela 2.1: Diferença entre este estudo e assuntos atualizados semelhantes na literatura.

Ref	**Problema**	**Processo**	**Métodos**		**Implementação**
			Exato	**Aproximado**	
1	Seq	Contínuo		x	Alg Genético
2	Seq	Contínuo	x	x	Alg Evolucionário PLI
3	Agrup Seq	Contínuo	x	x	VNS PLIM
4	Agrup	Batelada		x	Alg Evolucionário Chave Aleatória
5	Agrup	Batelada	x	x	Busca Tabu PLIM
6	Agrup	Batelada	x	x	AG Difuso PLI
Este autor	Agrup	Batelada	x	x	Alg Genético PLI

Observa-se algumas semelhanças entre esta pesquisa os trabalhos (SU, 2016) e (CASADO et al., 2020) conforme Tabela 2.1 e para ilustrar as diferenças faz-se necessário algumas explicações sobre o processo a ser otimizado neste trabalho.

Em planta de recozimento em caixa busca-se formar as cargas ou pilhas de bobinas ocupando a maior capacidade de peso possível, visto que a quantidade de recursos gastos como combustível e força de trabalho tem pouca dependência com o peso das cargas formadas e, portanto, podem ser considerados como custo fixo de operação. Entretanto, com a otimização é esperado que sejam necessários menos ciclos de recozimento para tratar um estoque de bobinas, gerando então melhor aproveitamento de recursos, tempo e mão de obra. Além do mais, obedecer a regra de processo com bobinas de mesma especificação de aço, observando o tempo máximo que as bobinas podem ficar aguardando para agrupamento favorece o atendimento aos clientes dentro dos prazos e evitam interferências na qualidade desejada para o produto final.

O processo de formação da carga pode acontecer várias vezes em uma jornada de trabalho, sendo necessário conferir a relação de bobinas disponíveis no estoque para aplicação. Contudo, existem regras de formação de carga que devem ser obedecidas devido as dimensões do compartimento de bobinas, que aqui é chamado abafador, portanto, se faz necessário uma boa análise para determinar quais materiais podem ser agrupados. Após a escolha das bobinas para formação de carga, foco deste trabalho, estas são empilhadas na base conforme Figura 1.1. Em seguida um forno de recozimento recobre o abafador e o processo de recozimento inicia-se até atingir as propriedades mecânicas desejadas. No fim, as bobinas são resfriadas e liberadas para o próximo processo. Na Seção 3.2.1 estão descritos termos reais de processo utilizados em ambiente industrial.

Restrições do problema tratam da necessidade de agrupamento de bobinas de mesma especificação, ou seja, mesmo tipo de aço, não podendo haver misturas pois cada especificação tem tempo pré-determinado de recozimento para atingir sua propriedade mecânica (DEUS et al., 2007). Este tempo é calculado por um modelo matemático que leva em consideração variáveis como especificação do aço, espessura e peso das bobinas selecionadas para a carga. Restrições adicionais como as limitações físicas de altura máxima e peso máximo devem ser observadas pois, no caso da altura, o somatório das larguras das bobinas selecionadas compõe a altura da carga formada e o somatório do peso das bobinas compõe o peso da carga formada. Cada equipamento deste tipo tem sua especificação com limites máximos definidos pelo seu fabricante.

Condições especiais do processo podem exigir priorização das bobinas mais antigas

do estoque para garantia dos requisitos de qualidade pois um processo de oxidação pode ocorrer em função do tempo de armazenamento da bobina, visto que, normalmente, não se controla umidade nos ambiente de armazenamento de bobinas de aço.

Por questões de lucratividade deseja-se que não sejam geradas cargas com poucas bobinas, ou seja, um número mínimo de bobinas é estabelecido e também a formação de carga com um peso mínimo para evitar desperdícios.

Com as exposições do problema abordado é possível apontar as principais diferenças entre este trabalho e a literatura citada. Este trabalho considera a otimização do processo de formação de carga após um LTF. O trabalho de (SU, 2016) considera a otimização da cadeia de produção até o processo de um LTQ, sendo o LTF um equipamento não considerado por (SU, 2016). Já em relação ao trabalho de (CASADO et al., 2020) as diferenças estão na forma de abordagem e no equipamento tratado. Em (CASADO et al., 2020) a modelagem é baseada em partição por cliques enquanto este trabalho apoia-se no PMM. Em termos de método aproximado, (CASADO et al., 2020) utilizou busca tabu enquanto este trabalho utiliza AG. Além disso, o problema abordado por (CASADO et al., 2020) é referente a um LTQ enquanto este trabalho trata de planta de Recozimento em Caixa.

Na literatura, pelo menos que fosse de conhecimento dos autores, não encontrou-se trabalhos que aborde método de solução exata e algoritmos genéticos como métodos para resolução de problema de formação de carga em planta de recozimento em caixa. Em comum observa-se que a maioria dos métodos aproximados empregados atualmente são baseados em algorítimos genéticos que fazem parte dos algoritmos evolutivos e inspirou a sua aplicação nesta pesquisa. Assim, para resolver este problema de pesquisa uma formulação matemática é proposta baseado no Problema da Mochila, cuja função objetivo visa maximizar o peso total da carga ocupada pelas bobinas de mesmas características e priorizar as bobinas mais antigas em estoque além, uma proposta de solução por algoritmos genéticos como método aproximado. As Seções 2.3 e 2.4 contêm a fundamentação teórica utilizada para ambos os métodos.

2.3 Método Exato

O Problema da Mochila (PM) é amplamente estudado em otimização combinatória e faz parte da classe de problemas NP-Difícil (MARTELLO; TOTH, 1990). Sua abordagem é feita na área de Pesquisa Operacional em Engenharia de Produção e outras áreas

de otimização de processo. Segundo (SOUZA; RAFAEL, 2009) este não é somente um problema clássico nesta linha de pesquisa, mas é um importante assunto que apresenta diversas aplicações no mundo real.

Duas direções de pesquisa podem ser seguidas para problemas de otimização combinatória:

- Solução por métodos exatos.
- Solução por métodos aproximados.

Para os problemas combinatórios, algumas das técnicas utilizadas por métodos exatos são programação linear, não-linear e programação linear inteira. Estes podem ser tão complicados de resolver que, mesmo no estado da arte em termos de algoritmos e *hardware*, pode não ser possível obter soluções ótimas ou soluções exatas, sendo necessário fazer o uso de alguma técnica que forneça pelo menos uma solução razoável, preferencialmente próximo da ótima. Neste caso os métodos heurísticos ou em alguns casos os meta-heurísticos geralmente atendem fornecendo "boas" soluções (FOLLY, 2017).

Um algoritmo exato busca encontrar resultado ótimo ou um conjunto de soluções ótimas para um determinado problema (HILLIER; LIEBERMAN, 2010). No entanto, os métodos exatos normalmente consomem um tempo exponencial para sua resolução, o que inviabiliza sua aplicação para problemas de larga escala na vida prática. Assim, o desenvolvimento de métodos heurísticos tem recebido mais atenção nas últimas décadas (FOLLY, 2017). Observa-se que muitos pesquisadores apresentam trabalhos comparando resultados gerados utilizando soluções exatas e algorítmos heurísticos. Nesta Seção são abordados os métodos exatos específicos para resolução do problema da mochila adicionando-se características a partir da formulação básica para sustentar a aplicação deste estudo.

2.3.1 Problema da Mochila na sua Forma Generalizada

De acordo com (MARTELLO; TOTH, 1990) o Problema da Mochila (PM) é apresentado da seguinte forma. Suponha que um viajante tenha que carregar ou encher sua mochila selecionando itens ou objetos visando maximar o conforto. Cada objeto possui um peso p_i e a mochila tem capacidade c. A formulação deste problema é feita enumerando os objetos i de 1 até m através do vetor de variáveis binárias $x_i(i = 1, ..., m)$. Sendo:

- $x_i = \begin{cases} 1, & \text{se o objeto } i \text{ é utilizado} \\ 0, & \text{caso contrário} \end{cases}$

E as seguintes definições,

- O é o conjunto de objetos.
- b_i é uma medida de conforto dado para o objeto i denominado "benefício".
- p_i é o peso do objeto i.
- c é a capacidade da mochila.

O problema está em selecionar os objetos avaliando o vetor binário x_i que satisfaça a restrição de forma que o somatório dos pesos selecionados seja menor que a capacidade da mochila.

A função objetivo (2.1) é de maximização do conforto do viajante e tem a seguinte formulação.

$$Max \quad \sum_{i=1}^{m} b_i x_i \tag{2.1}$$

É importante mencionar que os valores do benefício b_i são dados de entrada do problema.

A restrição (2.2) garante o limite de capacidade da mochila.

$$\sum_{i=1}^{m} p_i x_i \leq c, \tag{2.2}$$

Variantes do Problema da Mochila, mais conhecido como *Knapsack Problem* (KP) foram estudados mais intensamente após o trabalho de Danzig (DANTZIG, 1957).

Em (KELLERER; PFERSCHY; PISINGER, 2004) existem muitos exemplos do problema da mochila e suas variantes dentre os quais cita-se o problema de um despachante de carga aérea no qual o benefício de despachar uma carga é diretamente proporcional ao peso da carga a ser despachada. Neste caso, o peso ótimo da carga a ser despachada pelo avião está ligado ao atingir a carga ideal prevista para o tipo do avião. O problema de otimização resultante neste caso é conhecido como Problema de Soma de Sub-Conjunto

(*Subset Sum Problem*), porque procura-se a soma dos pesos p_i de um sub-conjunto de forma que não ultrapasse a capacidade de peso c. A formulação matemática para o problema é:

$$Max \quad \sum_{i=1}^{m} p_i x_i \tag{2.3}$$

$$s.a \quad \sum_{i=1}^{m} p_i x_i \leq c \tag{2.4}$$

Este exemplo foi citado para demonstrar que os problemas possuem aplicações importantes em vários domínios, especialmente em logística, indústria e gerenciamento financeiro. Em outras palavras, o problema da mochila e suas variações são aplicados em muitas organizações, sejam de pequenas empresas a grandes corporações. Mais especificamente, muitos problemas pertencentes à família do Problema da Mochila geralmente aparecem como uma redução ou subproblema nos procedimentos de solução de problemas mais complexos (ANDRADE; BIRGIN; MORABITO, 2013). Devido a estas características, os modelos teóricos estudados possuem particular importância. Em (WILBAUT; HANAFI; SALHI, 2007) observa-se além do problema clássico, as seguintes aplicações: *two-dimensional KP* (TKP); *multidimensional KP* (MKP); *multi-objective KP*(MOKP); *multiple KP* (M-KP); *precedence KP* (PCKP); *disjunctively constrained KP*(DCKP); *multiple choice KP*(MCKP); *multiple choice MKP* (MCMKP); *multidemand MKP* (MDMKP); *knapsack sharing problem* (KSP); *quadratic KP* (QKP); *max-min KP* (MMKP).

Um resumo das aplicações são apresentadas abaixo:

- KP/TKP: corte de estoque, orçamento financeiro e criptografia.
- MKP/MOKP: orçamento financeiro, problema de carregamento, alocação de recursos e gerenciamento diário de satélite.
- M-KP: carregamento de carga.
- PCKP: gerenciamento de projeto.
- DCKP: problema de localização.
- MCKP/MCMKP: orçamento financeiro, problema de alocação e confiabilidade de sistemas complexos.

- MDMKP: orçamento financeiro e problema de localização.
- KSP: problema de alocação.
- QKP: problema de localização.
- MMKP: orçamento financeiro.

As técnicas desenvolvidas para o problema da mochila podem ser utilizadas para resolver problemas com n dimensões (LOPES; RODRIGUES; STEINER, 2003). O PM de uma única dimensão consiste na representação de único valor para as restrições de cada item. A aplicação deste modelo é encontrada em problemas de corte de barras metálicas. Também, é o caso deste trabalho onde existe a restrição de altura máxima e peso máximo. Para os itens com duas dimensões (altura e largura), a mochila é um espaço bi-dimensional (QUEIROZ; MIYAZAWA, 2012) e (ANDRADE; BIRGIN; MORABITO, 2013). Aplicações são encontradas na indústria têxtil, metalúrgica, etc. Também podem haver aplicações do PM tri-dimencional, quando utilizado para restrições de altura, largura e profundidade. Cada mochila deve ter as três dimensões bem definidas e cada item é associado a um volume. Esta aplicação é encontrada em carregamento de contêiner, navios, e caminhões (KOBLASA; VAVROUSEK; MANLIG, 2016).

Em (HUANG et al., 2015) tem-se a seguinte abordagem quanto a restrição do limite de itens na mochila. Recebe-se um conjunto de itens e um conjunto de mochilas. Tanto o peso quanto o lucro do item estão em função da mochila, e cada mochila tem uma capacidade real positiva. A restrição exige um número de itens "k" admissível a cada mochila. São definidos dois seguintes objetivos: (1) maximizar o lucro total de todas as mochilas (Max-Sum k-GMK); (2) maximizar o mínimo lucro de todas as mochilas (Max-Min k-GMK). Demonstrou-se que os dois problemas são NP-completo quando k é maior ou igual a 4. Para o problema do K-GMK Max-Sum, pode-se encontrar a ótima solução, especialmente quando k=2, utilizando algoritmos de aproximação. Para o problema Max-Min k-GMK, apresentou-se um algoritmo de aproximação 1/(k-1), e quando k=2, encontra-se a solução ótima.

Em (HASAN; KAABI; HARRATH, 2019) trata de um caso especial do problema de empacotamento multi-objetiva 3D. N caixas de diferentes dimensões devem ser inseridas em um número mínimo de caixas maiores idênticas. As caixas menores têm pesos diferentes e podem ser apenas horizontalmente giradas quando colocadas nas caixas maiores. Dois objetivos são simultaneamente considerados: Um número mínimo de caixas maiores para empacotar todas as caixas menores e ter peso total equilibrado em termos de caixas

maiores. O problema investigado é NP-Difícil. Um algoritmo é proposto para resolver esse problema 3D em duas fases. Durante a primeira fase, todas as combinações possíveis de camadas de tipos iguais ou diferentes de caixas menores que podem caber em uma caixa maior são geradas. Essas combinações representam soluções candidatas em termos de volume. Finalmente, na segunda fase as caixas menores são acondicionadas nas caixas maiores de acordo com o melhor uso do volume. O algoritmo para resolver este problema 3D foi validado usando dados do mundo real de uma empresa de correios Fedex gerando ótimos resultados.

No entanto, essa família de problemas é muito ampla e não será totalmente vista aqui. Para obter mais informações recomenda-se consultar (MARTELLO; TOTH, 1990) além de (KELLERER; PFERSCHY; PISINGER, 2004) e (WILBAUT; HANAFI; SALHI, 2007). A próxima seção tem-se uma discussão sobre o Problema da Mochila 0-1, assim como alguns conceitos base para formulação do modelo proposto neste trabalho.

2.3.2 Problema da Mochila 0-1

Na sua forma mais simples, dado a mochila com capacidade c, e m objetos, cada um com peso p_i e benefício b_i busca-se preencher a mochila com os objetos de forma a maximizar o benefício. Em outras palavras, busca-se encontrar o vetor $\vec{x} = (x_1, x_2, ..., x_m)$ onde $x_i \in \{0, 1\}$, tal que a restrição $\sum_{i=1}^{m} x_i p_i \leq c$ seja satisfeita, maximizando o benefício $Z = \sum_{i=1}^{m} x_i b_i$.

O termo "0-1" citado no Problema da Mochila 0-1 se deve ao fato de o item selecionado ter a obrigação de entrar por completo na mochila, ou seja, não fracioná-lo, o que implica em não poder dividir o objeto para caber na mochila mesmo que haja espaço (LAGOUDAKIS, 1996). Esta importante característica é aplicada neste trabalho pois a variável de decisão não pode ser fracionada.

Algumas formas de resolver o problema da mochila são apresentadas (MARTELLO; TOTH, 1990): a cada objeto selecionado busque o que tenha maior benefício b_i e insira na mochila até que não caiba mais novos objetos. Este algoritmo pode dar boa ou má solução que no geral não garante ser a melhor solução possível. Outras soluções podem ser implementadas buscando-se escolher objetos que tragam maior valor agregado fazendo uma relação entre o benefício e o peso do item b_i/p_i.

Por se tratar de um problema de otimização combinatória, a busca pela solução pode gerar uma série de testes para todas as possibilidades de subconjuntos de objetos que po-

dem ser inseridos na mochila ou não (LAGOUDAKIS, 1996). Para elucidar, um exemplo de busca para este problema é dado de forma que para n objetos, existem 2^n possibilidades de busca. Os objetos são hipoteticamente denominados B1 e B2, portanto, para $n = 2$ o espaço de busca (EB) é $2^n = 2^2 = 4$ possibilidades. Na "base" da Figura 2.6, tem-se as decisões referentes a B1 e B2 que comprovam as quatro combinações possíveis.

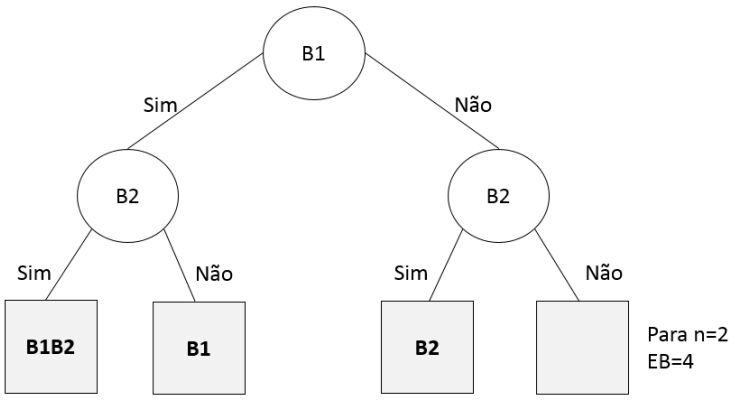

Figura 2.6: Espaço de busca para $n = 2$ geram quatro possibilidades.
FONTE: Adaptado de (HILLIER; LIEBERMAN, 2010)

Para o caso de crescimento do número de objetos, suponha $n = 10$ dará $2^n = 1024$ e se $n = 30$, $2^n = 1.073.781.824$ possibilidades. Portanto, o tamanho da instância a se testar no problema da mochila tem tempo de execução que cresce exponencialmente e demanda grande custo computacional.

Segundo (SHAMAKHAI, 2017) ao abordar o problema da mochila 0-1 é dito que este problema é conhecido como um problema NP-Difícil e nunca foi provado poder ser resolvido por algoritmo em tempo polinomial. Também acrescenta que este é um problema de otimização combinatória e que já foi resolvido por alguns algoritmos tais como, programação dinâmica, *Branch and Bound* ($B\&B$) e Algoritmo Genético (AG).

Em (MOR, 2007) apresenta-se um estudo para resolver o problema da mochila com a técnica de roubo de tarefas (*workstealing*) em programas paralelos tomando-se como base a paralelização do algoritmo *Branch-and-Bound* para resolver o problema da mochila. Segundo ele, pelo fato de o problema da mochila ser NP-Completo, a complexidade exponencial pessimista é $O(2^n)$. Assim, utiliza-se como principal ferramenta uma implementação da especificação MPI (*Message-Passing Interface*) com objetivo de avaliar o impacto da solução apresentada sobre diferentes aspectos como tempo de execução, consumo de memória, balanceamento de carga, complexidade algorítmica, etc.

Segundo (TUCKER; CRAMPTON; SWIFT, 2005) os algoritmos que percorrem todo

o espaço de busca, como métodos exatos, tem tempo de execução exponencial em função do tamanho dos dados de entrada, assim, procedimento de heurística e métodos aproximados são usados para resolver problemas reais.

A seguir algumas características dos métodos de solução para o problema da mochila (KELLERER; PFERSCHY; PISINGER, 2004).

- Programação Linear Inteira (PLI): é um método exato e garante soluções ótimas, mas só para instâncias pequenas ou médias, pois para grandes instâncias a característica exponencial impõe elevado custo computacional de modo a tornar inviável a obtenção de soluções práticas. Como método de resolução utiliza-se, por exemplo, *Branch and Bound* ou *Branch and Cut*.

- Algoritmos de aproximação: São algoritmos que não necessariamente acham soluções ótimas mas garantem que a solução encontrada está próxima da solução ótima. A necessidade vem da impossibilidade de resolver satisfatoriamente a classe de problemas NP-Difíceis. Portanto, é razoável sacrificar uma solução ótima em troca de uma solução aproximada em tempos computacionais razoáveis.

- Heurística: São algoritmos que não garantem encontrar ótimas soluções pois pode não haver uma formalidade matemática para o problema, mas podem gerar boas soluções com tempo de execução razoável.

Como informado, para resolver os problemas PLI, o algoritmo ($B\&B$) pode ser usado e é feito uma breve descrição na Seção 2.3.2.1. Caso o problema a ser resolvido não envolva variáveis inteiras, o método Simplex pode ser usado como técnica de resolução (HASEEN et al., 2014).

2.3.2.1 *Branch and Bound*

Segundo (MARTELLO; TOTH, 1990) nos anos cinquenta a teoria da programação dinâmica foi primeiro introduzida por Bellman para resolver o problema da mochila 0-1. Logo, Dantzig apresentou o método eficiente para determinar a solução de um problema com as variáveis não inteiras e então criou-se o termo *Upper Bound*. Nos anos sessenta o problema da mochila foi muito estudado por Gilmore e Gomory e finalmente Kolesar apresentou o algoritmo de $B\&B$. Nos anos setenta o algoritmo de $B\&B$ foi desenvolvido e provado ser o método capaz de resolver os Problema de Programação Linear (PPL) para um número maior de variáveis. O método mais conhecido no período foi o de Horowitz

e Sahni. A partir de então buscou-se por métodos que pudessem reduzir o procedimento para solução visto que a árvore de busca cresce exponencialmente e Ingargiola e Korsh apresentaram um algoritmo de pré-processamento para redução do número de variáveis. Nos anos oitenta a busca era para se resolver problemas maiores, no qual a ordenação das variáveis já consumia boa parte do tempo de execução dos algoritmos. Então, Balas e Zemel geraram subproblemas em que ordenava-se apenas parte das variáveis.

Em (SHAMAKHAI, 2017) cita-se que o algoritmo $B\&B$ é aplicado para resolver problemas de programação linear (PPL) buscando-se garantir a integralidade dos valores das variáveis de decisão $x_1, x_2, ..., x_n$ de forma a maximizar/minimizar a função objetivo do problema. Relaxação das variáveis fazem parte do método de resolução de maneira que é possível "andar"pelo espaço contínuo de soluções, podendo analisar os vértices do poliedro até obter a solução ótima.

Para (HILLIER; LIEBERMAN, 2010), pode parecer que os problemas de Programação Linear Inteira (PLI), derivados do PPL, sejam relativamente fáceis de resolver. Afinal, PPL podem ser resolvidos com extrema eficiência e a única diferença é que os problemas de PLI têm muito menos soluções a serem consideradas. De fato, problemas puros de PLI com região viável limitada são garantidos terem apenas um número finito de soluções viáveis. Para o (HILLIER; LIEBERMAN, 2010) existem duas falácias para esta ideia.
A primeira falácia está no fato de haver um número finito de soluções viáveis que garantem que o problema seja solucionável. Números finitos podem ser muito grandes. Por exemplo o caso simples de problemas de PLI com variáveis binárias. Com n variáveis, existem 2^n soluções a serem consideradas, no qual algumas dessas soluções podem subsequentemente ser descartadas por violar as restrições funcionais. Assim, cada vez que o valor de n é aumentada em uma unidade, o número de soluções é dobrado.
A segunda falácia é que remover algumas soluções viáveis das regiões com valores das variáveis não inteiras de um problema de programação linear facilitará a solução. Pelo contrário, segundo ele, é pelo fato de haver todas as soluções viáveis não inteiras que se pode garantir encontrar a ótima solução do problema. Isso se deve ao fato de que todas essas soluções viáveis existem para garantir que haverá uma solução viável nos vértices e, portanto, uma solução viável básica ideal para o problema geral. Essa garantia é a chave para a notável eficiência do método simplex. Como resultado, problemas de programação linear geralmente são consideravelmente mais fáceis de resolver do que problemas PLI. Consequentemente, os algoritmos mais bem-sucedidos para PLI incorporam um algoritmo de PPL, como o método simplex e/ou método dual simplex, relacionando partes do problema de PLI em consideração ao PPL correspondente. Esse método geralmente é

chamado de relaxamento do PPL.
Portanto, como já apresentado na Figura 2.6 o problema cresce exponencialmente. Hoje os melhores algoritmos de PLI são muito superiores à enumeração exaustiva. Houve notória melhoria nas últimas duas ou três décadas. Os problemas de PLI Booleana que exigiriam muito de tempo de computação para serem resolvidos há 25 anos, agora podem ser resolvidos em segundos com os melhores softwares comerciais. Essa significativa evolução no desempenho se deve a três grandes fatores:
1) melhorias nos algoritmos de PLI Booleana, assim como, em outros algoritmos de PLI,
2) melhorias nos algoritmos de PPL que são muito usados nos algoritmos de PLI,
3) e a evolução da performance dos elementos de *hardware*, incluindo computadores de mesa. Adicionalmente, (MOR, 2007) introduziu o estudo do emprego da técnica de roubo de tarefas *(Workstealing)* em programas paralelos, tomando-se como base a paralelização do algoritmo *Branch and Bound* para resolução do Problema da Mochila.

O espaço de busca do algoritmo B&B pode ser representado por uma árvore, chamada habitualmente de árvore de busca, onde cada um nó representa o resultado de soluções de problemas lineares que necessitam ser resolvidos. Com a evolução dos algoritmos e das definições matemáticas dos PPL, foram definidos os limites superiores e inferiores de modo e reduzir o tempo computacional em busca de soluções. A palavra *Branch* indica ramificação a partir de um nó de modo a descobrir outras respostas para maximizar ou minimizar a função objetivo. A busca sempre vai depender da fração da árvore a ser percorrida. Já a palavra *Bound* propõe dois limites, um inferior e outro superior, que são usados, juntamente com a função objetivo, no processo de exploração na árvore de busca. De fato, uma vez encontrado um nó cujo valor da função objetivo é menor do que o limite inferior já encontrado, no caso de maximização, não é necessário mais ramificar este nó reduzindo o tempo computacional e também a porção do espaço de busca explorado. Para a finalização do processo de busca em um ramo três critérios devem ser atendidos (HASEEN et al., 2014):

- Todos os valores encontrados para as variáveis com restrição de integralidade são inteiros e significa que foi encontrado uma solução inteira.
- Encontrar uma solução fracionária, cujo valor da função objetivo não é melhor que o de uma solução inteira já encontrado *(lower bound)*.
- Encontrar uma solução inviável.

Veja na Figura 2.7 um breve exemplo ilustrativo da árvore de busca para o seguinte

problema de maximização.

$$Max \quad z = 5x_1 + 8x_2 \tag{2.5}$$

$$S.a \quad -x_1 + x_2 \leq 6, \tag{2.6}$$

$$5x_1 + 9x_2 \leq 45, \tag{2.7}$$

$$x_1, x_2 \geq 0 \tag{2.8}$$

$$x_1, x_2 \; inteiros \tag{2.9}$$

A função objetivo é definida por (2.5). Dois conjuntos de restrições de capacidade são definidas por (2.6) e (2.7). A expressão (2.8) trata da não negatividade e a integralidade das variáveis está representada em (2.9).

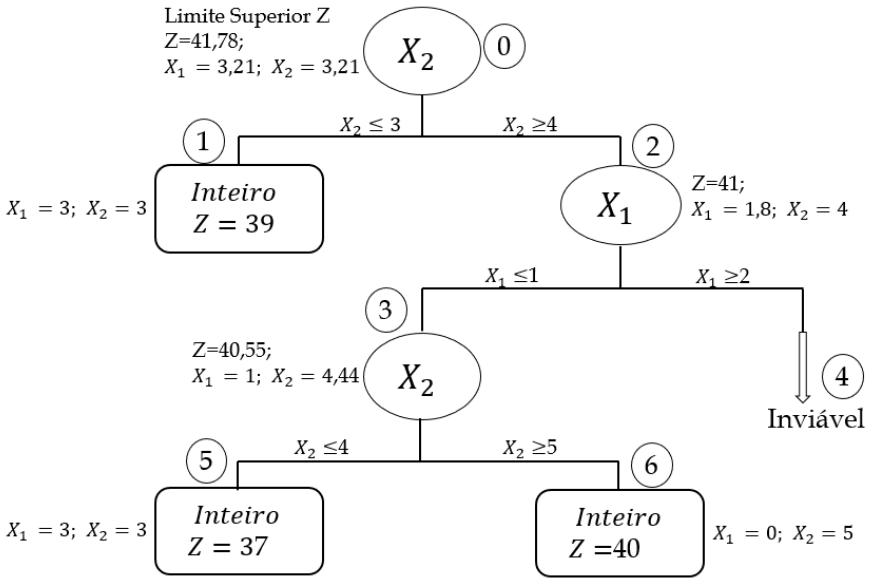

Figura 2.7: Árvore de busca do *Branch and Bound* pelo método de Dakin
FONTE:(HASEEN et al., 2014)

Ao buscar o melhor valor para função objetivo garantindo a variável x_1 e x_2 como inteiro, o simplex calcula $Z = 41,78$. Neste passo nem x_1 ou x_2 tornaram-se inteiros. Em seguida toma-se uma das variáveis, no caso foi o valor x_2 e adiciona-se as restrições de forma que $x_2 \leq 3$ e $x_2 \geq 4$. No passo um encontra-se valores das variáveis, inteira, com $Z = 39$ não necessitando fazer mais iterações neste lado da árvore. No passo dois novamente encontra-se variável não inteira x_1 mas o valor da função objetivo $Z = 41$ é maior que no passo um. Estes passos são seguidos até encontrar o melhor valor da função objetivo para x_1 ou x_2 inteiros. O melhor valor neste problema se deu no passo seis de

forma que a função objetivo é $Z = 40$ com as duas variáveis x_1 e x_2 inteiras.

2.3.3 O Problema das Múltiplas Mochilas

Este trabalho tem foco na variante do Problema da Mochila cuja variável de decisão é binária e se tratando do PM seria naturalmente encaixado na classe do Problema da Mochila 0-1. Contudo, uma variante do PM chamado Problema das Múltiplas Mochilas (PMM) tem por definição selecionar um subconjunto de objetos de tamanho máximo m e dividi-los em n mochilas de modo a maximizar o benefício total do conjunto selecionado, desde que a soma dos pesos dos objetos associados uma mochila j não ultrapasse a capacidade c_j.

Segundo (THOMAS; KHURI; HEITKOTTER, 1994) no PM cada objeto i possui um único valor associado, sendo um caso unidimensional. Para o caso de mais de uma mochila cada objeto i pode ser atribuído a uma mochila j, gerando-se um problema com variável bidimensional (m x n) de forma que um objeto i escolhido está associado a uma mochila j.

Segue a formulação do PMM, onde:

- $x_{ij} = \begin{cases} 1, & \text{se o objeto } i \text{ é associado a mochila } j \\ 0, & \text{caso contrário} \end{cases}$
- m é o número de objetos.
- n é o número de mochilas.
- b_i é o benefício do objeto i.
- p_i é o peso do objeto i.
- c_j é a capacidade da mochila j.

$$Max \quad \sum_{j=1}^{n} \sum_{i=1}^{m} b_i \, x_{ij} \tag{2.10}$$

$$S.a.$$

$$\sum_{i=1}^{m} x_{ij} \, p_i \; \leq \; c_j, \forall j = 1, ..., n \tag{2.11}$$

$$\sum_{j=1}^{n} x_{ij} \leq 1, \forall i = 1, ..., m \tag{2.12}$$

$$x_{ij} \in \{0, 1\}, \forall i = 1, ..., m, \forall j = 1, ..., n \tag{2.13}$$

A função objetivo (2.10) visa associar cada objeto i na mochila j de forma a maximizar o benefício total. As restrições (2.11) garantem que o somatório dos pesos dos objetos p_i na mochila j não ultrapasse a capacidade da mochila c_j. As restrições (2.12) garantem que um objeto i só pode estar alocado a no máximo uma mochila j e em (2.13) a definição da variável x_{ij} pertencer ao conjunto de valores binários zero ou um.

Algumas generalizações são feitas para o problema tais como (SHAMAKHAI, 2017): 1) Pesos e benefícios dos objetos e capacidade da mochila são positivos inteiros conforme (2.14):

$$p_i > 0; b_i > 0; c_j > 0; \tag{2.14}$$

2) Devido à forte interação entre as variáveis, peso p_i de cada objeto associado à mochila e capacidade da mochila c_j, há uma restrição em que não pode haver peso de um objeto maior que a capacidade de uma mochila como mostrado em (2.15):

$$p_i \leq \; max\{c_j; \forall j \in n\}; \forall i \in m; \tag{2.15}$$

3) É demonstrado em (2.16) que a capacidade de qualquer mochila deve ser maior que o menor peso individual de um objeto:

$$c_j \geq \; min\{p_i; \forall i \in m\}; \forall j \in n; \tag{2.16}$$

4) Como complemento a expressão (2.17) implica que o somatório dos pesos dos objetos

é maior que as capacidades das mochilas:

$$\sum_{i=1}^{m} p_i > c_j \; \forall \; j = 1, ..., n \tag{2.17}$$

Em (LOPES; RODRIGUES; STEINER, 2003) cita-se o PMM como problema de otimização combinatória, cujo vasto espaço de busca tem representação matemática $EB = n.2^m$ sendo n o número de mochilas, m o número de objetos o qual gera uma função exponencial de base 2. Pelo fato de o PM já ser da classe de problemas NP-Difícil, aumentando o número de mochilas este problema torna-se ainda mais difícil de resolver. Em contra partida no caso $n = 1$ volta-se ao clássico Problema da Mochila 0-1 mencionado na Seção 2.3.2.

Segundo (WANG; XING, 2009) considera-se resolver o problema das múltiplas mochilas com o objetivo de maximizar o benefício total com capacidades diferentes e também com as capacidades iguais através de algoritmo de aproximação. Seu objetivo é resolver o problema da mochila em tempo polinomial mas observa haver erro de aproximação em relação ao valor ótimo. Assim, o método de resolução para uma mochila n vezes ao invés de resolver o problema das múltiplas mochilas uma vez gera um erro de aproximação a ser demonstrado. Considera-se resolver o problema com duas ou três mochilas. Na tabela 2.2 pode ser vista a resolução, para o pior caso do problema das múltiplas mochilas, quando comparado com a resolução do problema da mochila uma por vez. Considera-se z o resultado do algoritmo de aproximação e z^* o valor ótimo. Na primeira linha $z/z^* = 3/4$ no pior caso para as capacidades das mochilas iguais.

Tabela 2.2: Perda na resolução do PM em relação ao PMM no pior caso. (WANG; XING, 2009)

Caso	Subcaso	Relação
$c_1 = c_2 = c_3$		3/4
$c_1 = c_2 \leq c_3$	$c_2 \leq 1/2\ c_3$	5/7
	$c_2 > 1/2\ c_3$	2/3
$c_1 < c_2 = c_3$	$c_2 \geq 3\ c_1$	3/4
	$4/3\ c_1 < c_2 < 3\ c_1$	7/10
	$c_2 \leq 4/3\ c_1$	8/11
$c_1 < c_2 < c_3$	$c_1 + c_2 \leq c_3$	2/3
	$2c_1 \leq c_2$	2/3
	$c_1 + c_2 > c_3$ & $2\ c_1 > c_2$	3/5

Como exemplo prático, observa-se uma instância criada com quatro itens na Tabela 2.3. Busca-se inserir os itens em duas mochilas e resolver uma mochila n vezes, ou seja, preencher uma mochila de cada vez e posteriormente resolver o problema das múltiplas

mochilas para duas mochilas com capacidade 10.

Tabela 2.3: Instância de teste para comparação de solução do PM e PMM

Item	**1**	**2**	**3**	**4**
Peso	4	4	6	6
Benefício	5	5	3	3

Então, são selecionados os itens um e dois, de forma que o benefício totalizado seja dez, portanto, o maior valor possível e a capacidade utilizada oito. Restam o item três e o item quatro. Os dois itens restantes ultrapassam a capacidade da mochila dois ficando portanto apenas a possibilidade de seleção do item três ou do item quatro. A Tabela 2.4 apresenta a solução com valor da função objetivo totalizando treze.

Tabela 2.4: Seleção de itens para cada mochila n vezes maximizando o benefício.

Mochila1			Mochila2		
Item	Peso	Benefício	Item	Peso	Benefício
1	4	5	3 ou 4	6	3
2	4	5			
Total	8	10	Total	6	3

Na Tabela 2.5 está contido o resultado do PMM com maximização do benefício tendo a seleção do item um e três na primeira mochila e os itens dois e quatro na segunda mochila totalizando o benefício dezesseis, sendo o erro de aproximação neste exemplo $z/z^* = 13/16$ para duas mochilas de pesos iguais;

Tabela 2.5: Seleção de itens para PMM uma vez maximizando o benefício.

Mochila1			Mochila2		
Item	Peso	Benefício	Item	Peso	Benefício
1	4	5	2	4	5
3	6	3	4	6	3
Total	10	8	Total	10	8

Portanto, embora o PMM seja considerado fortemente NP-Difícil (WANG; XING, 2009), optar-se-á por implementar este trabalho pelo problema das múltiplas mochilas ao lugar de uma mochila n vezes em busca do maior valor da função objetivo conforme apresentado nesta simulação.

2.3.4 Problema das Múltiplas Mochilas com Conflito de Itens

O Problema das Múltiplas Mochilas com conflito de itens (PMMCI) consiste em maximizar o benefício b_i, selecionando itens com diferentes pesos p_i, em uma mochila de capacidade limitada c_j, enquanto restringe inserir na mochila itens considerados em conflito.

Segundo (SALEH, 2015) o PMMCI conhecida como *The Disjunctively constrained Knapsack Problem (DCKP)* é uma variante do clássico problema das múltiplas mochilas com especial restrição de disjunção. As seguintes citações e aplicações elucidam esta variação do problema da mochila.

Em (QUEIROZ; MIYAZAWA, 2012) trata-se do Problema da Mochila 0-1 Bidimensional com restrições de Disjunção. É proposto uma heurística baseada *Greedy Randomized Adaptative Search Procedure (GRASP)*. Este gera uma solução inicial obtida através de procedimento que opera numa lista de possíveis candidatos, logo após, é executado um procedimento de busca local, gerando pertubações a vizinhança da solução inicial. Diferentes estratégias são propostas para gerar soluções diversificadas e escapar de ótimos locais. O trabalho propôs comparar uma solução heurística com um modelo de programação linear inteira utilizando o CPLEX (IBM, 2013). Os resultados computacionais encontrados pela heurística proposta foram em média de 23% piores, em relação ao valor ótimo resolvido pelo CPLEX.

Em (SADYKOV; VANDERBECK, 2012) é citato o problema do empacotamento com conflito de itens. O objetivo é empacotar itens do mesmo tipo, em um número mínimo de pacotes respeitando a sua capacidade. O estudo *"Bin-Packing With Conflict: A Generic Branch-and-Price Algorithm"* propõe a implementação de um procedimento próprio *Branch and Price* baseado em algoritmo de programação dinâmica o qual provou ser mais eficiente na prática quando comparado com o solver do CPLEX.

Para (YAMADA; SEIJI; KOHTARO, 2002) a restrição de disjunção aplicada no problema da mochila é caracterizada por um conjunto I contendo m objetos, um conjunto E de pares incompatíveis que não podem ser unidos em uma mesma mochila de capacidade c. Sem perda de generalidade é assumido que w_i, p_i e c são positivos inteiros, $\sum_{i=1}^{m} p_i \geq c$, $w_i < c \; \forall \; i = 1, ..., m$, e o conjunto de pares incompatíveis é $E \subseteq \{(i,j) \mid 1 \leq i \neq j \leq m\}$. Portanto, se $(i,j) \in E$ não se permite que seja incluído na mochila simultaneamente por se tratar de itens incompatíveis. Esta relação é recíproca para $(i,j) \in E \Leftrightarrow (j,i) \in E$,

sendo então denominada relação de disjunção formulada conforme Equação (2.18).

$$x_i + x_j \leq 1,\ \forall (i,j) \in E \tag{2.18}$$

Por se tratar de um problema de otimização combinatória, este se reduz ao clássico PMM citado na Seção 2.3.3, quando $E = \{\}$, ou seja, não é encontrado o conflito entre os itens (SALEH, 2015).

A formulação do PMMCI então se torna:

- $x_{ij} = \begin{cases} 1, & \text{se o objeto } i \text{ é associado a mochila } j \\ 0, & \text{caso contrário} \end{cases}$
- m é o número de objetos.
- n é o número de mochilas.
- b_i é o benefício do objeto i.
- p_i é o peso do objeto i.
- c_j é a capacidade da mochila j.
- E é o conjunto de itens conflitantes.

$$Max \quad \sum_{j=1}^{n} \sum_{i=1}^{m} b_i\, x_{ij} \tag{2.19}$$

$S.a.$

$$\sum_{i=1}^{m} x_{ij}\, p_i \ \leq\ c_j, \forall j = 1, ..., n \tag{2.20}$$

$$\sum_{j=1}^{n} x_{ij} \leq 1, \ \ \forall\ \ i = 1, ..., m \tag{2.21}$$

$$x_{ij} + x_{kj} \leq 1,\ \forall (i,k) \in E, \ \ \forall\ \ j = 1, ..., n \tag{2.22}$$

$$x_{ij} \in \{0, 1\}, \ \ \forall\ \ i = 1, ..., m, \ \ \forall\ \ j = 1, ..., n \tag{2.23}$$

A função objetivo (2.19) visa associar cada objeto i na mochila j de forma a maximizar o benefício total b_i. As restrições (2.20) garantem que o somatório dos pesos dos objetos p_i na mochila j não ultrapasse de capacidade da mochila c_j. As restrições (2.21) garantem

que um objeto i só pode estar alocado a no máximo uma mochila j e em (2.23) a definição da variável x_{ij} pertencer ao conjunto de valores binários zero ou um. Observa-se a inclusão das restrições (2.22) que representa o conjunto de restrições de disjunção o qual garante a compatibilidade dos itens selecionados no problema das múltiplas mochilas.

Na teoria de complexidade computacional, o problema da mochila pertence à classe de problemas NP-Difícil, o que implica não haver solução computacional eficiente. Muitos algoritmos de aproximação foram criados nas últimas décadas, no entanto, interesse por este problema permanece tão forte que ainda é pesquisado intensamente, uma vez que cada pequena melhoria pode gerar enormes valores econômicos na produção das indústrias (LAM et al., 2017).

Ainda em (LAM et al., 2017), foi abordado problema de empacotamento bidimensional (2D). Este trata o problema de otimização combinatória relacionando à alocação de vários objetos em recipientes retangulares de dimensões conhecidas com o objetivo de minimizar o número de caixas necessárias. Cita também que uma das abordagens mais populares para esse problema é combinar uma estratégia de escolha com uma técnica de otimização. Assim, em seu trabalho cita muitos pesquisadores utilizando algoritmos genéticos para resolver este problema possibilitando encontrar boas soluções.

Desta forma, após apresentado o referencial teórico para o problema da mochila como um método exato, na próxima seção é demonstrado um segundo método baseado em algoritmos genéticos a ser estudado como método aproximado.

2.4 Métodos Aproximados

2.4.1 Heurísticas e Metaheurística

De acordo com (ABBASS; SARKER; NEWTON, 2002) a palavra heurística é originária do grego antigo $\varepsilon\upsilon\rho\iota\tau\kappa\omega$, e significa descobrir. Existem, entretanto, significados mais direcionados e ligados ao contexto no qual a palavra é aplicada, embora todos sejam essencialmente similares.

Nas ciências que estudam o comportamento humano, por exemplo, entende-se por heurística a maneira com que o ser humano cognitivamente julga e decide com base em intuição, analogia ou experiência, formando estratégias a partir de informações incertas, incompletas e imprecisas sobre eventos complexos com a finalidade de tornar o processo decisório factível, tudo isso em detrimento do rigor absoluto e admitindo que a solução

concebida pode não ser a melhor alternativa possível para a situação ou adversidade depurada, mas que oferece uma "boa" tratativa dado as limitações do agente (CASH, 2019).

Analogamente, no âmbito da computação e informática, mais precisamente no ramo de otimização, métodos heurísticos são estratégias que encontram soluções para problemas específicos sem garantir que são ótimos globais (BOUARARA; HAMOU; RAHMANI, 2018), em contrapartida, o desenvolvimento de soluções através desses métodos se destaca pela maior efetividade em problemas de elevada complexidade, quando comparados aos métodos exatos (LEE; EL-SHARKAWI, 2008).

Essa maior efetividade se refere à resolução de problemas das classes NP-Completo ou NP-Difícil, ou seja, os métodos determinísticos não são capazes de conceber soluções em tempo polinomial, enquanto as estratégias não determinísticas baseadas em heurísticas ou meta-heurística geram boas soluções e conseguem checá-las em tempo polinomial (SACHDEVA; GOEL, 2014). Isso significa que elevando-se a complexidade com o aumento do tamanhos das instâncias em problemas dessas categorias, a característica exponencial dos métodos exatos faz com que o tempo de execução rapidamente se torne impraticável, sendo esse um fator limitante em muitas aplicações reais.

As abordagens heurísticas oferecem então, uma maneira alternativa praticável de encontrar soluções que se aproximam do ótimo global para problemas do tipo NP-Completo ou NP-Difícil, conforme discutido por (BALLOU, 2006). Assim, heurísticas sendo convenientemente direcionadas aos problemas corretos, fazem com que o espaço de busca seja explorado de maneira inteligente, contribuindo com seu sucesso na resolução de problemas difíceis (KAGAN et al., 2009).

As metaheurísticas, por sua vez, são algoritmos mais gerais que guiam as heurísticas na busca pelo aprimoramento das soluções. As meta-heurística podem se beneficiar de fatores aleatórios para evitar o aprisionamento em máximos e mínimos locais durante seu processo de busca e convergência a uma solução, gerando portanto uma aproximação de maneira progressiva da solução buscada (KAGAN et al., 2009).

Heurísticas e meta-heurística pertencem a classe de métodos aproximados (não determinísticos), em conjunto com outros casos especiais. Segundo (NETO, 2005), os métodos aproximados de solução de problemas podem ser classificados de acordo com o fluxograma da Figura 2.8

Trabalhos são publicados e exploram as formas não determinísticas de solução para

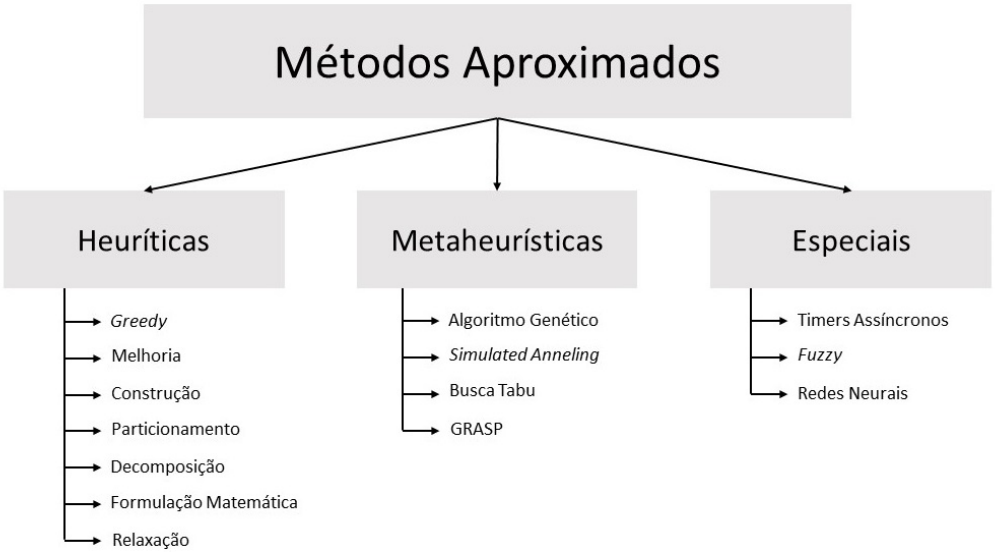

Figura 2.8: Classificação e exemplos de métodos aproximados de solução de problemas de otimização.
FONTE:(RODRIGUES, 2000 apud NETO, 2005)

problemas de otimização do tipo NP-Completo ou NP-Difícil. Em (NETO, 2005) verificou-se que algoritmos genéticos são eficientes na solução do problema do contêiner, (TOMAR, 2014) constatou que é possível resolver o problema da cobertura de vértices mínima em tempo hábil com o uso de uma metodologia gulosa (*Greedy*) melhorada, (ALKHEDHAIRI, 2008) utilizou *Simulated Annealing* para resolver o problema da P-Média, (PEREZ; GARCIA; RAMIREZ, 2005) apresentou um algoritmo genético para resolução de problema de empacotamento 2-D, (YAKAWA; IIMA, 2003) implementou um algoritmo genético para o problema do *Bin Packing Problem* e (FUKUNAGA, 2008) aplicou o algoritmo genético para resolução do problema das múltiplas mochilas.

Em (FOGEL, 1994) cita-se que existem três principais tipos de pesquisa para processo de evolução simulada: algoritmos genéticos, estratégias de evolução e programação evolutiva. Cada método enfatiza diferentes partes da evolução natural. Os algoritmos genéticos enfatizam os operadores cromossômicos. Estratégias de evolução enfatizam mudanças comportamentais no nível do indivíduo. A programação evolucionária enfatiza mudanças comportamentais no nível das espécies.

Para esse trabalho, há o interesse em explorar soluções mediante o uso de algoritmos genéticos para o problema das múltiplas mochilas pois foi visto ser um método eficiente para resolver este tipo de problema. Portanto, a Seção 2.4.2 se dedica a discorrer um resumo introdutório destes conceitos e o problema da mochila solucionado por algoritmos genéticos. Recomenda-se o livro de (LINDEN, 2012), (EIBEN; SMITH, 2015), que foram

usados como base para este referencial teórico.

2.4.2 Algoritmos Genéticos

Segundo (EIBEN; SMITH, 2015) os Algoritmos Genéticos (AG) são baseados na teoria da evolução de Darwin e oferece uma explicação da origem da diversidade biológica e da estrutura dos seus mecanismos. A seleção natural de um indivíduo desempenha um papel central no que às vezes é chamado de visão macroscópica da evolução. Dado um ambiente que pode abrigar um número limitado de indivíduos com o instinto básico de reprodução, a seleção se torna inevitável se o tamanho da população não deve crescer exponencialmente. São dois alicerces para que haja o progresso evolutivo utilizando a seleção baseada em competição.

O primeiro é a seleção natural que favorece aqueles indivíduos que dados os recursos, competem de maneira mais eficaz, em outras palavras, aqueles que são adaptados ou adequados às melhores condições ambientais.

Outra força primária identificada por Darwin resulta de variações fenotípicas entre os membros da população. Traços fenotípicos são características comportamentais e físicas de um indivíduo que afetam diretamente sua resposta ao meio ambiente (incluindo outros indivíduos), determinando sua aptidão. Esse fenômeno também é conhecido como sobrevivência do mais apto. Este termo deve ser entendido como, o indivíduo que melhor se encaixa, pode sobreviver, visto que a natureza tem incertezas e não há garantia de sobrevivência. Cada indivíduo representa uma combinação única das características de seus traços fenotípicos que são avaliadas pelo ambiente. Se essa combinação tiver avaliação favorável, então o indivíduo tem maior chance de gerar filhos e os bons traços fenótipos dos indivíduos podem ser propagados através da geração de seus filhos, senão, o indivíduo é descartado. A visão de Darwin foi que variações pequenas e aleatórias (mutações) nas características fenotípicas ocorrem durante a reprodução de geração a geração. Através dessas variações, novas combinações de características ocorrem e são avaliadas. Os melhores sobrevivem e se reproduzem, e assim, há evolução. Para resumir esse modelo básico, uma população consiste em um número de indivíduos. Esses indivíduos são as unidades de seleção, ou seja, dizem que seu sucesso reprodutivo depende de quão bem eles estão adaptados em seu ambiente em relação ao restante da população. Como os indivíduos mais bem sucedidos se reproduzem, mutações ocasionais dão origem a novos indivíduos para serem testados. Assim, com o passar do tempo, há uma mudança na constituição da população, ou seja, a população evolui a cada geração.

Ainda segundo (EIBEN; SMITH, 2015) a visão microscópica da evolução natural é oferecida pela genética molecular. Isto lança luz sobre os processos abaixo do nível das características fenotípicas visíveis, em particular relacionado à hereditariedade. A observação fundamental da genética é que cada indivíduo é uma entidade dupla, sendo elas, as propriedades fenotípicas (externas) e genotípico representado em um nível interno. Em outras palavras, o genótipo de um indivíduo codifica seu fenótipo. Os genes são as unidades funcionais de herança codificando características fenotípicas. Por exemplo, propriedades visíveis como a cor da pele ou o comprimento da cauda podem ser determinados pelos genes. Aqui é importante distinguir genes e alelos. Um alelo é um dos possíveis valores que um gene pode ter. Nos sistemas naturais, a codificação genética não é individual: um gene pode afetar mais características fenotípicas (pleiotropia) e, por sua vez, uma característica fenotípica pode ser determinada por mais de um gene (poligênese). Variações fenotípicas são sempre causada por variações genotípicas, que por sua vez são as consequências de mutações de genes ou recombinação de genes por reprodução sexual. Em computação evolucionária a combinação das características de dois indivíduos para gerar filhos é frequentemente chamado de *crossover*.

Em um dos primeiro livros publicados sobre algoritmos genéticos (HOLLAND, 1992) cita que estes ocupam um importante papel nos estudos de sistemas adaptativos complexos, teoria econômica e dispositivos complexos apresentando fundamentos teóricos e práticos. Utiliza conceitos de sistemas artificiais e continua a explorar seu uso no estudo em ampla gama de processos complexos e de ocorrência natural explicando os principais efeitos da coadaptação, coevolução e surgimento de esquemas que quando recombinados e transmitidos às gerações seguintes inovam e fornecem melhorias. Seu trabalho intitulado de *Adaptation in Natural and Artificial Systems*, tem objetivo de formalizar matematicamente os processos de adaptação natural em sistemas artificiais que mantenham os mecanismos originais.

Em (IYODA, 2000) aborda-se o algoritmo evolutivo, que vem da ideia de programação evolutiva, como sendo uma técnica para criar inteligência artificial em problemas de otimização e neste caso são inseridas as estratégias evolutivas. Apresenta-se a seguinte estrutura de algoritmo evolutivo que tem o princípio da sobrevivência natural, sendo: cria-se e seleciona uma população de indivíduos, alguns sofrem transformações, enquanto evoluem e competem pela sobrevivência.

Segundo (LINDEN, 2012) a Figura 2.9 apresenta um esquema similar podendo ser resumido através dos seguintes passos:

1. Inicialize a população.
2. Avaliação de cada cromossomo da população.
3. Seleção dos pais para geração de novos filhos (cromossomos).
4. Aplicação dos operador de recombinação e mutação aos pais de forma a gerar os indivíduos da nova geração.
5. Substituir os piores indivíduos da população por novos.
6. Avalie todos os novos cromossomos e insira na população.
7. Se atender o critério de parada, retorne o melhor cromossomo, senão, volte ao passo 3.

Esta é uma visão de alto nível e esconde algumas complexidades do processo de obtenção de alguns elementos como a representação do cromossomo adequado ao problema. Para (IYODA, 2000) um cromossomo representa um indivíduo da população o qual contém a codificação (genótipo) de uma possível solução para o problema (fenótipo). Sua implementação é normalmente na forma de vetores em que cada posição do vetor é conhecido como gene e os possíveis valores que o gene pode assumir é chamado de alelo. Os cromossomos podem ser implementados como uma cadeia de bits, aqui tratado como codificação do cromossomo, e comumente utiliza-se de um vetor como estrutura de armazenamento. Para saber o quanto este indivíduo está adaptado ao meio é utilizado uma função objetivo que podem ser para maximizar ou minimizar. Em 2.4.2.1 é apresentado o quão versátil um algoritmo genético é e se adapta em diferentes processos além da biológica.

2.4.2.1 Função Objetivo ou *Fitness*

Em um primeiro momento, a descrição apresentada na Seção 2.4.2 pode aparentar limitar o escopo de aplicação da técnica à problemas de otimização, entretanto é possível aplicá-la em uma ampla variedade de problemas de outros tipos, visto que, muitas vezes, realizando algumas adaptações, a situação possa ser reconhecida como um problema de otimização sendo necessário, portanto, criar um modelo que descreva o quão apta uma possível combinação de dados de entrada é de ser solução de um problema.

Por exemplo, no problema da mochila, descrito na Seção 2.3.1, fica evidente que pode-se prontamente encontrar uma função para maximização, tendo em vista que o

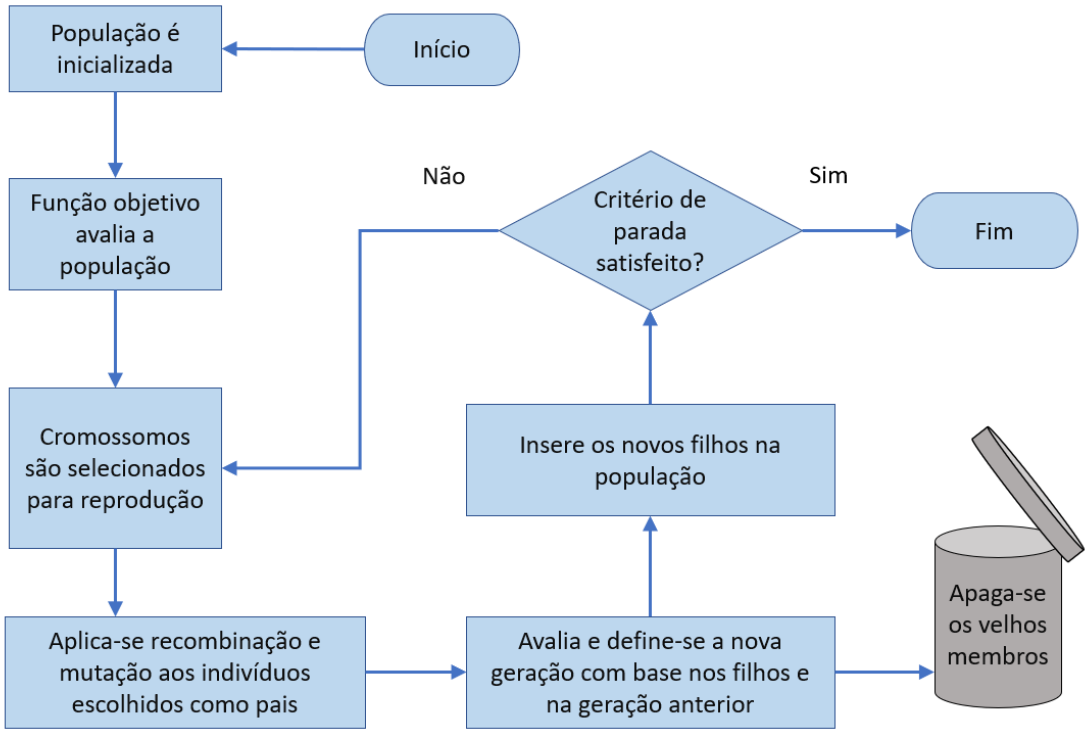

Figura 2.9: Esquema genérico de um algoritmo genético.
FONTE: Adaptado de (LINDEN, 2012)

objetivo do problema é essencialmente maximizar o conforto proporcionado pelos itens a serem escolhidos. Em contrapartida, existem outros exemplos cuja aplicação do método é menos óbvia, como o problema matemático do quadrado mágico, em que o objetivo é fazer com que um conjunto de números sejam alocados em espaços de uma matriz quadrada de modo que a soma dos elementos das linhas, das colunas ou das diagonais sejam sempre um valor fixo, isso está ilustrado na Figura 2.10, em que tem-se um exemplo em que as somas devem resultar em 15 e os números a serem alocados são os inteiros de 1 a 9, sem repetição.

A princípio esse não é um problema de otimização, entretanto se o objetivo for adaptado tal que passe a desejar-se que seja minimizado a diferença entre o módulo da soma dos elementos em cada linha, coluna e diagonal e o valor de interesse 15, conforme descrito pela Equação (2.24), então ele se torna um problema propício para ser resolvido por um algoritmo genético. Para a equação que avalia a aptidão das soluções é atribuído o nome de "Função Objetivo" ou "*Fitness*".

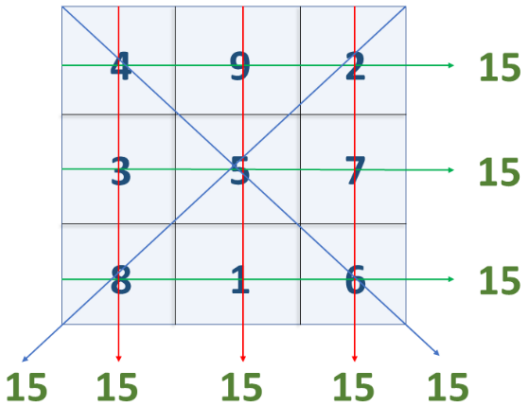

Figura 2.10: Possível solução para problema do quadrado mágico 3x3.
Fonte: (WEISSTEIN, 2020)

$$
\begin{aligned}
Min\ Z = |x_{11} + x_{12} + x_{13} - 15| \ + \ |x_{21} + x_{22} + x_{23} - 15| \ + \\
|x_{31} + x_{32} + x_{33} - 15| \ + \ |x_{11} + x_{21} + x_{31} - 15| \ + \\
|x_{12} + x_{22} + x_{32} - 15| \ + \ |x_{13} + x_{23} + x_{33} - 15| \ + \\
|x_{11} + x_{22} + x_{33} - 15| \ + \ |x_{13} + x_{22} + x_{31} - 15|
\end{aligned}
\tag{2.24}
$$

Estas funções são implementadas em algoritmos que são inspirados pelo conhecimento do processo evolucionário dos seres vivos, trazendo alguns atributos característicos em comum, como os conceitos de: gene, cromossomo e população, além dos processos de: seleção, *crossover* e mutação. Na Seção 2.4.2.2 são apresentados alguns conceitos básicos de algoritmo genético.

2.4.2.2 População, Cromossomo e Gene

Um dos passos iniciais da implementação de um algoritmo genético é a criação da população inicial sendo formada por um conjunto de cromossomos que representam possíveis soluções de um problema. (FOGEL, 1994) cita que uma solução candidata deve ser iniciada sujeita a aplicação de certas restrições do problema. Tipicamente, cada cromossomo deve ser codificado em um vetor com x elementos sendo cada elemento denominado gene e cada posição no gene possuem valores específicos que se chamam valor do alelo. (HOLLAND, 1992) sugere que a representação de uma solução seja binária, sendo esta uma representação muito explorada na literatura.

Segundo (LINDEN, 2012) o tamanho da população precisa ser escolhido de modo a atingir um equilíbrio entre a efetividade e eficiência. Populações constituídas de baixo número de indivíduos podem não ser adequadas para permitir um boa exploração do espaço de busca, fazendo com que seja mais recorrente a estagnação em pontos ótimos locais, ao passo que populações superdimensionadas podem tornar o processo excepcionalmente lento, inviabilizando a obtenção da solução em tempo aceitável.

Um conjunto de cromossomos ou indivíduos, formam uma população, ou em outras palavras, uma população representa um conjunto de possíveis soluções para o problema de otimização. A Figura 2.11 sumariza o vínculo entre os conceitos de população, cromossomo e gene (TALBI, 2009).

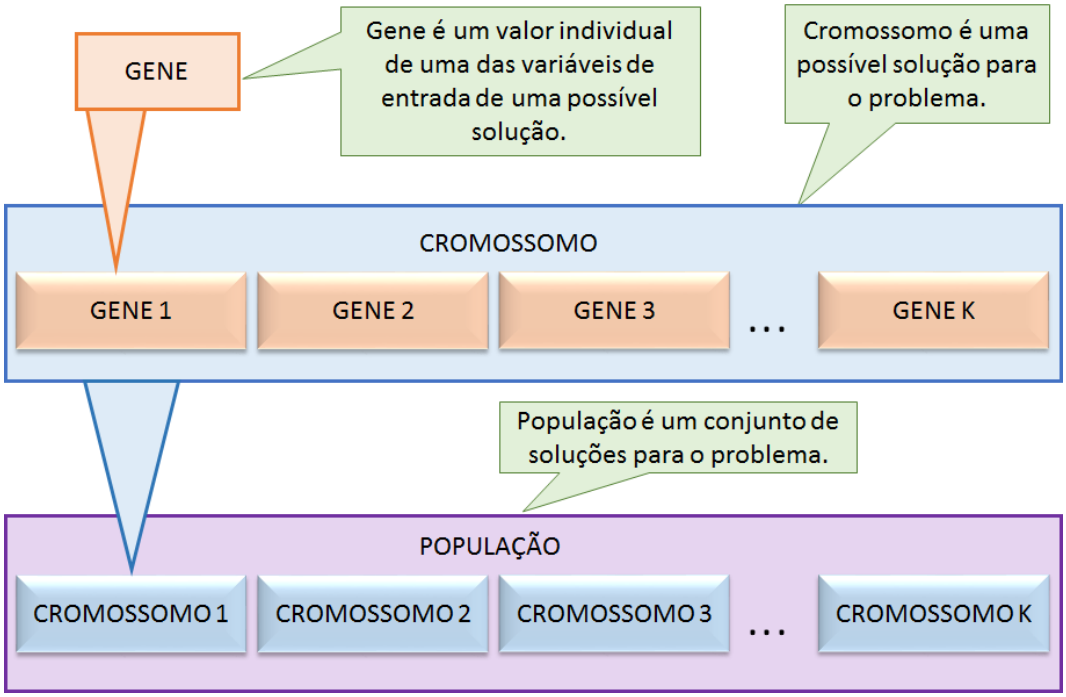

Figura 2.11: Relação entre os conceitos de população, cromossomo e gene, dentro do contexto de algoritmo genético.
FONTE: Adaptado de (TALBI, 2009)

Segundo (FOGEL, 1994), cada cromossomo, $x_i, i = 1, ..., n$ na população deve ser decodificado de forma que a função de aptidão (*fitness*) possa sumarizar o resultado, tendo cada, uma probabilidade de ser selecionado para um processo reprodutivo. A possibilidade de seleção é proporcional ao valor do indivíduo calculado pela função de aptidão em relação aos outros cromossomos da população. Se a aptidão de cada individuo a ser maximizado é um número positivo, então frequentemente utiliza-se o método da roleta para o sorteio. De acordo com a probabilidade de reprodução associada, uma nova população de

cromossomos é gerada a partir da seleção dos indivíduos da população atual. Os cromossomos selecionados geram filhos a partir de uso de operadores genéticos, como *crossover* e mutação. O operador de *crossover* é normalmente aplicado em dois cromossomos, denominados na literatura como "pais", e são gerados dois novos cromossomos denominados "filhos" através da seleção de parte do cromossomo de cada um dos cromossomos pais para gerar o filho um e o restante do cromossomo dos pais formam o outro filho. Operadores mais sofisticados serão vistos posteriormente. O processo de mutação acontece gerando a chance de um gene mudar o valor do seu alelo, e caso a codificação seja binária, poderá mudar *bit* de estado. Este processo é repetido até que um critério de parada seja atingido.

Em (LINDEN, 2012) comenta-se que existem diversas abordagens quanto ao critério de parada do processo evolucionário, entre elas se destacam aqueles baseados no número de avaliações da população feitas pela função objetivo. Outras formas podem utilizar o relógio do computador para definir um tempo fixo de execução.

2.4.2.3 Representação ou Codificação de Indivíduos

Segundo (IYODA, 2000), afirma-se que a codificação do cromossomo é um dos fatores mais importantes na implementação do algoritmo genético. Em (LINDEN, 2012) explica-se que a codificação é maneira como se traduz a informação de forma que possa ser tratada pelo computador. Assim, a qualidade da codificação está diretamente ligada a quão adequada o algoritmo genético está para resolver o problema.

Em (HOLLAND, 1992) é apresentado a codificação binária com motivação vinda da teoria dos esquemas. A representação em esquema é capaz de descrever diversos cromossomos simultaneamente construído inserindo o caractere *don't care* * no alfabeto de gene. Um exemplo é [* 0 0 1 0 0 1] que representa [**1** 0 0 1 0 0 1] ou [**0** 0 0 1 0 0 1] podendo o caractere * estar em qualquer posição e quantos forem necessários, sendo o esquema [* * * * * * *], a representação possível de todos os cromossomos de comprimento 7. Exemplo, para o esquema binário [* 0 0 1 0 0 1] tem-se 2^1 cromossomos possíveis e para a representação [* * * * * * *] equivale a 2^7 combinações. Portanto, o número de cromossomos no esquema são 2^n sendo n o número de genes.

Conforme (LINDEN, 2012), em seu livro ressalta que a representação cromossomial é completamente arbitrária pois não há qualquer tipo de obrigação em adotar a representação binária. Informa ainda que muitos autores a utilizam pela simplicidade de implementação e representação de um problema.

Por outro lado, existem problemas que adotando-se a representação binária dificultam a implementação como o caso de problemas de múltiplas dimensões de variáveis contínuas que requerem grande quantidades de bits para atingir tal precisão, então, cromossomos se tornam grandes demais dificultando a operação do AG (HERRERA; LOZANO; VERDEGAY, 1998).

As seguintes recomendações são apresentadas no livro de (LINDEN, 2012) para uma boa codificação :
a) A representação deve ser a mais simples possível, que parte do princípio da simplicidade *KISS - keep it simple* devendo este ser um procedimento básico da informática e não do AG.
b) Havendo soluções proibidas para o problema, então não se deve ter uma representação. O problema do caxeiro viajante é um bom exemplo apresentado, pois deseja-se visitar cinco cidades e se a codificação binária for utilizada, sendo cada gene representado com 2 *bits*, só se poderia visitar quatro cidades inviabilizando a codificação e se três *bits* forem utilizadas oito cidades são possíveis de serem visitadas. Neste caso as cidades zero, seis e sete são exceção de modo que, dois cromossomos podem se cruzar e gerar esta solução que é proibida e não deveria estar contida dentro da representação.
c) Se houverem restrições, estas devem ser tratadas na construção da representação, exemplo o caso anterior em que as cidades seis e sete não existem.
Uma representação sugerida para este problema é a que utiliza números inteiros em cada gene, por exemplo a representação [1 2 3 4 5], que pode significar, primeiro visita-se a cidade um, depois a dois e assim sucessivamente. Caso uma nova representação fosse [5 1 2 3 4] primeiro visita-se a cidade cinco, depois a cidade um e assim sucessivamente. Observa-se que neste caso não há a ocorrência dos problemas apresentados na representação binária. Portanto, sugere-se nem sempre recorrer a representação binária. O livro de (LINDEN, 2012) no capítulo dez contém vários exemplos. Pode-se citar: AG baseado em ordem, representação numérica, valores categóricos e representações híbridas, sendo boa parte destas representações com exemplo de seus operadores de *crossover* e mutação.

Em (RESENDE, 2012) apresenta-se o estudo com *Biased Random-Key Genetic Algorithm* (BRKGA) que significa algoritmo genético com chaves aleatórias. A chave aleatória é um número aleatório real em um intervalo contínuo entre zero e um [0 - 1] utilizados nos cromossomos que formam uma população. Cada elemento do vetor é chamado de chave e seu valor é gerado aleatoriamente na população inicial. A adaptabilidade (*fitness*) do cromossomo é definida pelo custo da solução fornecida por uma heurística que recebe como um de seus dados de entrada o vetor de chaves do cromossomo e devolve uma solu-

ção viável para o problema. Este método de codificação não é usado neste trabalho mas ressalta-se que é um novo método com vários trabalhos já apresentados na literatura o qual os resultados apresentam ótima performance em termos de custo computacional e solução de problemas de otimização.

Portanto, observa-se que a codificação do problema a ser resolvido é uma das principais causas de sucesso e insucesso na implementação de um AG.

2.4.2.4 Seleção

Seleção consiste na sobrevivência do cromossomo mais apto ou assegurar que a informação passe adiante, para isso utiliza-se a função objetivo para avaliar a aptidão de cada elemento da população, em seguida ordena-se de acordo com os resultados e logo, aplica-se procedimentos de recombinação para, baseado no percentual de elitismo, os cromossomos com melhor desempenho possam ser copiados para a próxima geração, eliminando-se os restantes. Este procedimento assegura um desempenho crescente através das gerações. O princípio básico da etapa de seleção é que deve-se fazê-la levando em consideração a avaliação de cada cromossomo no momento de determinar os participantes do processo de recombinação e mutação. Para isso, existe o método conhecido como roleta viciada, que consiste em designar uma probabilidade de seleção a cada cromossomo da população de maneira proporcional ao valor da função de aptidão de cada um, como solução para o problema tratado, ou em outras palavras, aqueles que possuem valores superiores de avaliação pela função objetivo, tem mais chances de participar do processo de recombinação, ao passo que não se elimina completamente a possibilidade que os cromossomos menos aptos também sejam selecionados ocasionalmente (LINDEN, 2012).

Adicionalmente, (EIBEN; SMITH, 2015) cita que no método de seleção por roleta viciada (*Roulette Wheel*), um dos mais conhecidos na literatura, cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão. Assim, para indivíduos com mais alta aptidão é dada uma porção maior da roleta, enquanto aos indivíduos de aptidão mais baixa, é dada uma porção relativamente menor e desta forma, os indivíduos com o maior aptidão, tem uma maior chance de ser escolhido para gerar descendentes.

Em (LINDEN, 2012) define-se como **pressão seletiva** uma "força" que o método faz para impelir os esquemas contidos nas melhores soluções para próxima geração. Dependendo do método de seleção dos pais utilizados, pode-se acelerar ou retardar a convergência genética, caindo em ótimos locais com convergência pré-matura. É sugerido medir

a intensidade desta pressão através da melhoria obtida calculando-se a média da função de aptidão da população em relação ao melhor indivíduo da população, pois quanto mais próximos os valores, menor é a melhoria causada pelo método de seleção usada e também pelos operadores genéticos. Como pode ser visto na Figura 2.12 os valores tendem a se aproximar devido à convergência genética natural pela baixa diversidade ao fim do processo.

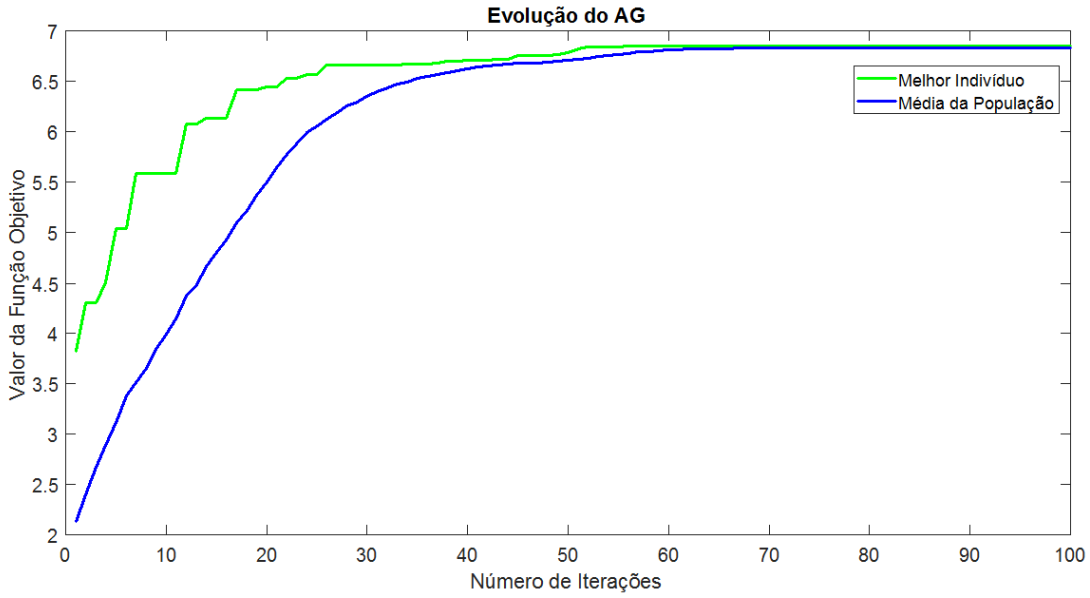

Figura 2.12: Evolução do melhor indivívuo e a média da população.
FONTE: Adaptado de (FOGEL, 1994)

Outros métodos de seleção são sugeridos para evitar que só pais bons sejam escolhidos, gerando convergências pré-matura, desta forma, se existirem esquemas bons em indivíduos ruins, de alguma forma possa haver chance deste indivíduo participar e deixar seu traço para as próximas gerações, ou mesmo permitir que indivíduos com avaliações ruins participem do processo reprodutivo, de tal forma que, esquemas ruins não desapareçam da população. Os métodos sugeridos além do método da roleta viciada são, seleção proporcional a função de aptidão, seleção por ranking, método do torneio, seleção uniforme dos pais entre outros (LINDEN, 2012) e (EIBEN; SMITH, 2015).

2.4.2.5 *Crossover*

Hereditariedade é uma forma de transmissão de informações genéticas de uma geração para a seguinte e os algoritmos genéticos realizam processos de *Crossover*. O processo de *Crossover* é responsável pela combinação de esquemas e também uma forma reprodução

existente neste processo. Este processo pode ajudar eliminar tendências simplesmente tornando aleatória a escolha do pai que da origem a cada um dos genes que formam os filhos. O indivíduo ou cromossomo escolhido possui um conjunto de características denominadas genes. Cada posição nesta cadeia de caracteres ou cromossomo é designada como um locus, que contém um valor ou alelo. Desse modo, para cada um dos genes, um dos pais é selecionado e o gene presente no locus correspondente é copiado para o filho. Em seguida repete-se para todos os locus(LINDEN, 2012).

Estes operadores podem ser categorizados pelo mecanismo básico usado para combinar informações dos pais e os mais comuns são classificados como: *Crossover* uniforme, cruzamento de ponto e dois pontos (EIBEN; SMITH, 2015).

O cruzamento de ponto numa representação binária inicia pela escolha de dois indivíduos (pais) na população com certa probabilidade. Um ponto de corte é escolhido aleatoriamente nos cromossomos dos pais e os segmentos entre os pais são trocados a partir do ponto de corte para geração dos filhos. A Figura 2.13 apresenta na parte de cima um processo de *crossover* de um ponto. Já o *crossover* de dois pontos, cada pai é dividido em três partes. O primeiro filho é formado através da escolha de duas partes do material genético do primeiro pai e uma parte do material genético do centro do segundo pai. Para a geração do segundo filho, o papel dos pais se inverte, usando duas partes do segundo pai e uma parte do primeiro.

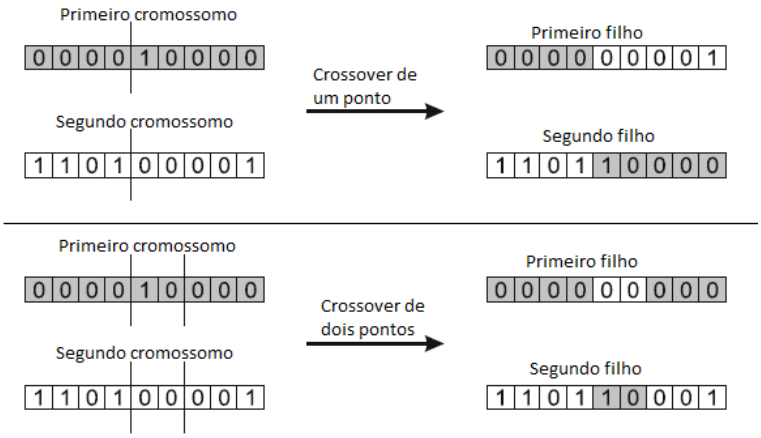

Figura 2.13: *Crossover* de ponto em cima e *Crossover* de n pontos em baixo para $n = 2$.
FONTE: (EIBEN; SMITH, 2015)

O *crossover* uniforme tem a capacidade de combinar todo esquema existente pois trabalha escolhendo entre as posições de cada um dos pais que formam cada posição do filho. Então, para cada gene é sorteado qual dos pais cede o gene ao filho, assim, se o

primeiro pai foi sorteado, o filho herda e gene do primeiro pai, caso contrário do segundo pai, continuando este processo até o último gene do cromossomo. É citado que devido ao fato de fazer sorteio para cada posição, seu desempenho tende a ser pior se comparado com o *crossover* de dois pontos (LINDEN, 2012).

Em (IYODA, 2000) são relatados diversos experimentos com operadores de *crossover* sendo que não há diferença clara de desempenho de um processo de *crossover* em relação a outro, assim, cada operador de *crossover* se torna eficiente para uma determinada uma classe de problemas e extremamente ineficiente para outras.

2.4.2.6 Mutação

O operador de mutação é o grande responsável pela garantia da diversidade genética na população enquanto o operador de *crossover* contribui para a igualdade entre os indivíduos. Pode-se pensar no operador de mutação como uma heurística exploratória, criando novos cromossomos e inserindo na população. Isso permite que novos indivíduos sejam explorados fora da população inicial criada (LINDEN, 2012).

O processo de mutação é feito através da modificação aleatória de um dos genes de cada cromossomo. A probabilidade de haver a mutação em um gene é chamado de taxa de mutação ou mesmo probabilidade de mutação, o qual é atribuído uma pequena probabilidade de alterar um dos genes do cromossomo. Para o caso de um esquema mais básico binário escolhe-se o gene a ser alterado com certa probabilidade e altera o valor do gene de zero para um ou vice-versa (IYODA, 2000).

A importância do processo de mutação se torna mais evidente quando se analisa o efeito da **pressão seletiva** sobre a população comentada na Seção 2.4.2.4, porque a pressão seletiva mostra que ao longo das gerações o valor da aptidão do melhor indivíduo e a média da função de aptidão da população tendem a se aproximar. Esse processo se chama convergência genética e sua ocorrência pode limitar a evolução pelo processo de *crossover*, restando apenas a esperança que mutações aleatórias possam reinserir diversidade para continuar a evoluir (EIBEN; SMITH, 2015).

Existem na literatura, exemplos de trabalhos que consideraram algum tipo de mutação adaptativa. Uma dessas técnicas é a variação da taxa de mutação de acordo com a diversidade da população. Também é sugerida a mutação dirigida que começa a trabalhar somente depois que a população começa a estagnar criando diversidade genética (LINDEN, 2012).

2.4.2.7 Controles Gerais em Algoritmos Genético

Esta Seção trata de alguns controles importantes listados na literatura quando se trabalha com algoritmos genéticos. A escolha de uma forma visual e dos parâmetros ideais são importantes para seu bom funcionamento. Alguns parâmetros básicos como a probabilidade de execução dos operadores de *crossover* e mutação, além da escolha do tamanho da população e o percentual de elitismo podem afetar significativamente a eficiência dos códigos.

A representação visual ajuda analisar a evolução do algoritmo genético se comparar a função de avaliação ou função de aptidão ou *fitness* do melhor indivíduo e a média dos valores da população. Conforme Figura 2.12 a forma visual ajuda a determinar a qualidade de um indivíduo como solução do problema e também analisar se há convergência prématura. A convergência pré-matura é observada se logo no início do algoritmo a média da população vai acompanhando o valor da função de aptidão sem deixar nenhum espaço logo nas primeiras gerações. Este comportamento leva a uma condição de baixa diversidade na população, o que não significa que o algoritmo está na melhor solução possível, mas pode ser que esteja em um ótimo local.

O conceito de elitismo foi introduzido com objetivo de preservar os melhores cromossomos ao longo das gerações. Isso é feito de tal maneira que um certo percentual de elitismo é definido e a partir dele o restante da população é substituída pelos filhos gerados. Logo, as boas soluções não são apagadas a menos que hajam soluções ainda melhores para assumir seu lugar (LINDEN, 2012). Normalmente o elitismo pode ser entendido como os indivíduos "elites" da geração.

Porém, ao deletar a porção com menor desempenho da população, é imposto uma grande pressão seletiva o que faz com que ocorra uma rápida perda da diversidade. Pode-se salientar aqui que para resguardar a performance do AG é essencial que hajam mecanismos que contribuam com a manutenção do parâmetro de diversidade. São conhecidas diversas abordagens para a manutenção da diversidade, ou para evitar a convergência genética precoce: uma das táticas para contornar esse problema é deletar cromossomos baseando-se na idade da string, também existem outros estudos que propõem grandes populações ou altas taxas de mutação. Além disso, uma outra forma interessante de atenuar a queda da diversidade é através da aplicação de uma política de "não-cópias", que consiste em recusar os filhos gerados se eles forem meras cópias de qualquer um dos outros cromossomos presentes na população no momento. Esse processo de recusa de clones, em alguns casos, impõe certo custo computacional relevante, visto que é necessário

comparar cada filho gerado com os outros cromossomos afim de determinar se ocorreu repetição, entretanto os benefícios proporcionados podem ser apreciados em muitos casos por impactar drasticamente na diversidade.(LINDEN, 2012)

Até este ponto no levantamento da literatura, pôde ser observado que os parâmetros fixos não são unanimidade entre os pesquisadores. O tamanho da população não é bem definido pois afeta a relação de eficiência e eficácia conforme apresentado na Seção 2.4.2.2. (LINDEN, 2012) não recomenda um valor específico para as variáveis pois, podem ser que para um caso o valor seja bom para uma aplicação mas não para outra. Adiante, é apresentado um trabalho que controla melhor algumas variáveis em busca de melhores resultados do algoritmo genético.

No trabalho de (XING et al., 2007) são apresentados métodos para superar a convergência pré-matura através um novo algoritmo genético adaptativo baseado na manutenção da diversidade. O primeiro ponto está na medição da diversidade individual do *fitness* da população ou da diversidade do gene para ajustar a probabilidade de cruzamento de forma adaptativa. Em segundo lugar, restringir a falta de genes eficazes em certos locus pois, as probabilidades de mutação de todos os alelos em cada locus variam adaptativamente dependendo da diversidade do gene no locus correspondente. O método utilizado para medição da diversidade do gene depende da codificação usada. Para uma codificação binária de uma população, o número mais perto de 0 no gene é o número 1 em cada locus, então a diversidade mais alta no gene será relacionado ao seu respectivo locus. Assim, a diversidade do gene pode ser medida da seguinte maneira: Suponha que o tamanho de uma população é N, sendo que a população na geração t consiste de ${x_t}^1, {x_t}^2, ..., {x_t}^N$. O valor $d_{i,j}^k$ representa a diferença entre o indivíduo x_i e x_j no locus k conforme Equação (2.25).

$$d_{i,j}^k = \begin{cases} 1, & \text{para } g_{ik} \neq g_{jk} \\ 0, & \text{caso contrário} \end{cases} \tag{2.25}$$

Então a diversidade do gene no locus k pode ser expresso através da Equação 2.26,

$$D_g^k(t) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^k \tag{2.26}$$

$D_g^k(t)$ representa a diversidade do gene no locus k.

Quando $D_g^k(t) = 0$ significa que todos os alelos do locus k são idênticos, ou seja,

todos são 0 ou todos são 1 e a diversidade do gene se torna baixa. Então baseado em (2.26) a diversidade da população baseada em todos os genes pode ser calculado conforme Equação 2.27

$$D_g(t) = \frac{\sum_{k=1}^{L} D_g^k(t)}{1/2.LN(N-1)} \tag{2.27}$$

onde L é o comprimento do cromossomo.

2.4.2.8 AG Aplicado no Problema da Mochila

Nesta Seção são apresentados alguns trabalhos de algoritmos genéticos aplicados para resolução do problema da mochila que podem ser usada como conceitos base para resolução deste tipo de problema.

É apresentado por (PEREZ; GARCIA; RAMIREZ, 2005) um algoritmo genético para resolver problemas de empacotamento 2-D de formas poligonais em uma tela retangular. Apresenta-se como codificar os parâmetros de forma e uma função de adequação baseada na transformação de eixo para avaliar indivíduos de uma população de algoritmos genéticos.

Em (YAKAWA; IIMA, 2003) é desenvolvido um algoritmo genético para o problema de empacotamento unidimensional, visando minimizar a quantidade de pacotes (*Bin-Packing Problem*). A principal contribuição foi o desenvolvimento do AG de tal forma que os filhos herdem fatores importantes dos pais através da codificação ou representação. O número do item empacotado corresponde ao gene. O genótipo é expresso como a sequencia de itens inseridos em cada mochila. Por exemplo, uma instância com M=15, ou quinze itens, um genótipo pode ser expresso gl: (1,3,10) (2,9,11) (5,7,13,15) (4,6,14) (8,12) estão associados à cinco pacotes M1, M2, M3, M4 e M5. É notório que o número de genes é variável no pacote, mesmo se o locus de um gene é trocado o genótipo tem a mesma solução como a seguir g2: (8,12) (2,9,11) (5,7,13,15) (4,6,14) (1,3,10), ou seja, uma solução como g1 e g2 é considerada igual. Para obter uma melhor solução é proposta um tipo de busca local no processo de *crossover* e mutação. Este trabalho foi aplicado em instâncias da literatura e mostrou melhores resultados em relação aos trabalhos utilizados para comparação.

Em (REZOUG; BADER-EL-DEN; BOUGHACI, 2017) é apresentado um método heurístico híbrido (memético) nomeado algoritmo genético guiado para resolver o problema da mochila multidimensional O algoritmo é composto por duas etapas e utilizam a pré-análise na aplicação no AG. Na pré-análise os dados do problema são tratados usando

um método eficiente para extração de informação útil. Este conhecimento prévio é inserido no AG como um conhecimento guiado. A primeira etapa é aplicada na geração da população inicial e a segunda ao avaliar os filhos produzidos através da função objetivo. Os resultados apresentados demonstraram que este método pré-guiado melhorou se comparado com o algoritmo genético tradicional. Nesta implementação, a codificação dos genes utilizam a própria identificação do item para compor os possíveis valores do lócus neste gene e o tamanho do cromossomo é definido pelo número de genes necessário para representar uma solução do problema. Detalhes conforme a Figura 2.14.

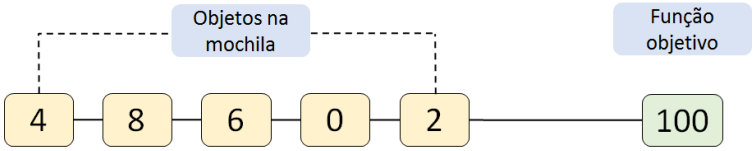

Figura 2.14: Exemplo de desenvolvimento de um cromossomo. Os objetos(itens) são inseridos na mochila e apresentados pelos seus identificadores. O valor da função objetivo é realizado pela soma dos benefícios de todos os objetos.
FONTE: Adaptado de (REZOUG; BADER-EL-DEN; BOUGHACI, 2017)

No trabalho de (FUKUNAGA, 2008) é apresentado o algoritmo genético aplicado ao problema das múltiplas mochilas utilizando conceito de dominância no espaço de soluções. É proposto um algoritmo evolutivo *Undominated Grouping GA* (UGGA), que difere significativamente de outros algoritmos em que gera apenas soluções candidatas que garanta não ser dominada de acordo com um critério de dominância particular. O UGGA pesquisa um espaço de soluções ótimo local de acordo com uma noção particular específica de otimalidade (dominância).

Seja um conjunto de itens g = i1, ..., ik para ser inserido nas mochilas. Uma solução é válida para o problema das múltiplas mochilas se um item é incluído no máximo em uma mochila e também não exceder a capacidade máxima de uma das mochilas. A codificação do problema é da seguinte forma: Considere quatro mochilas e 13 itens variando de a, b, c, ..., m com uma possível solução da seguinte forma: g1 = [(a, b, c), (d, e, f), (g, h, i), (j, k)] no qual a primeira mochila contém os itens a, b, c, a segunda mochila contém os itens d, e, f e assim sucessivamente. Seja um segundo candidato g2 = [(a, d, i), (b, c, g), (e, f, k), (j, h)] em que nem o item l ou m estão nas duas possíveis soluções. Assumindo que os pais g1 e g2 foram marcados para um crossover uniforme então a solução é c1 = [(a, b, c), (b, c, g), (g, h, i), (j, h)] e c2 =[(a, d, i), (d, e, f), (e, f, k), (j, k)]. Os itens b, c, g, h, aparecem duplicados em c1. É sugerido a remoção dos itens duplicados então c1 = [(a), (b, c, g), (h, i), (j)], ou seja, não descartando c1 por completo e gerando uma busca

local com critério particular, completando a mochila com os itens de maior benefício que ficaram como restante d, e, f, k, l, m, n.
O processo de mutação acontece nos indivíduos. Uma mochila é sorteada com certa probabilidade, é esvaziada e aplicada uma heurística com seleção dos candidatos com maior benefício.
Os resultados mostraram que o método maximiza as buscas por soluções candidatas.

Capítulo 3

Metodologia

3.1 Processo Metodológico

O processo metodológico está seguindo as orientações do livro de (GIL, 2002) e (SILVA; MENEZES, 2005) que definem que a pesquisa é um procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas. O desenvolvimento é feito mediante a conhecimentos disponíveis e a utilização cuidadosa de métodos, técnicas e procedimentos para desenvolver desde a adequada formulação do problema até a apresentação dos resultados.

Esta pesquisa é de natureza aplicada pois busca solucionar um problema real. Considera-se que a forma de abordagem é através de números e/ou estatísticas pois pode-se quantificar o resultado. Quanto ao objetivo deste estudo observa-se que tem característica exploratória pois possui familiaridade com o problema da mochila de forma a torná-lo explícito e envolve levantamento bibliográfico, troca de informações com pessoas com experiência do problema e análise de exemplos que estimulem a compreensão.

Esta pesquisa seguiu as seguintes etapas:

- Pesquisa bibliográfica: Buscou-se nesta etapa avaliar as variações dos métodos exatos e heurísticos, trabalhos atuais para resolução de problemas similares aplicados em indústria siderúrgica e aplicações para resolução do problema real de formação de carga em planta de recozimento em caixa.
- Escolha da estratégia a ser utilizada: A escolha da estratégia relaciona-se diretamente com o problema tratado. Desta forma, por ser um problema de maximização e alocação de objeto em vários recipientes, utilizou-se o problema das múltiplas

mochilas e suas variações como ponto de partida. Em seguida foram analisadas as divergências e convergências entre os problemas para só então propor um modelo matemático inicial, que posteriormente foi aperfeiçoado, e também de um modelo heurístico.

- Construção da solução: Após definido o modelo e suas regras gerais foram definidas as ferramentas para implementação do modelo matemático inicial. Após o aperfeiçoamento do modelo, optou-se por utilizar também um modelo heurístico baseado em AG para análise dos casos onde a solução exata apresentava dificuldades na resolução.

- Aplicação do método e obtenção dos resultados: Por meio da implementação de algoritmos para resolver o problema por método exato e método aproximado, foram realizados simulações e gerados resultados para sua análise.

- Análise dos resultados: Nesta fase, são avaliados os resultados obtidos, os tempos computacionais e validação do objetivo principal proposto para este trabalho.

A Seção 3.2 tem por objetivo demonstrar a modelagem final do Problema das Múltiplas Mochilas com Conflito de Itens Aplicado em Planta de Recozimento em Caixa utilizando conceitos de programação linear inteira com as variantes do Problema da Mochila. A Seção 3.3 contém o desenvolvimento do algoritmo genético aplicado.

3.2 Modelo para o Problema das Múltiplas Mochilas com Conflito de Itens

3.2.1 Definições e Nomenclaturas do Problema Real

Esta seção descreve a formulação matemática do problema das múltiplas mochilas aplicada em RCX assumindo os nomes típicos ou similar utilizados na indústria:

- $x_{ij} = \begin{cases} 1, \text{ se a bobina } i \text{ é associado ao abafador } j, \\ 0, \text{ caso contrário.} \end{cases}$

- m é o número total de bobinas.

- n é o número de abafadores.

- p_i é o peso da bobina i.
- l_i é a largura da bobina i.
- c_j é a capacidade de peso máximo permitido para cada abafador j.
- a_j é a altura máxima de cada abafador j.

Dadas estas definições, uma possível adaptação da formulação matemática da Seção 2.3.3 para o problema de formação de carga no RCX é apresentada a seguir:

$$Max \quad \sum_{i=1}^{m}\sum_{j=1}^{n} p_i \, x_{ij} \tag{3.1}$$

$S.a.$

$$\sum_{i=1}^{m} p_i \, x_{ij} \;\leq\; c_j, \forall \, j = 1, ..., n \tag{3.2}$$

$$\sum_{i=1}^{m} \; l_i \, x_{ij} \;\leq\; a_j, \forall \, j = 1, ..., n \tag{3.3}$$

$$\sum_{j=1}^{n} x_{ij} \leq 1, \;\; \forall \;\; i = 1, ..., m \tag{3.4}$$

$$x_{ij} \in \{0, 1\}, \;\; \forall \;\; i = 1, ..., m, \;\; \forall \;\; j = 1, ..., n \tag{3.5}$$

Nesta formulação observa-se que a função objetivo é definida em (3.1) e maximiza o peso das bobinas i associadas ao abafador j sendo este um caso do problema da Soma de Sub-Conjunto, o qual o benefício está associado ao um maior peso da carga. O conjunto de restrições (3.2) garantem que o somatório dos pesos das bobinas i selecionadas não ultrapasse a capacidade do abafador c_j. O conjunto de restrições (3.3) garantem que o limite de altura do abafador a_j não seja ultrapassado pelo somatório das larguras das bobinas l_i selecionadas. O conjunto de restrições (3.4) asseguram que a bobina selecionada só seja incluída em um abafador por vez. A restrição (3.5) é a declaração da variável de decisão x_{ij} fazer parte do conjunto de variáveis booleana.

Nesta pesquisa, o problema tem necessidades especiais de processo que vão além de gerar cargas maximizando o peso. Condições especiais de garantia da qualidade necessitam ser atendidas. O tempo de espera das bobinas aguardando em estoque podem gerar desvios de qualidades e consequentemente perda de valor agregado no produto. Assim,

a priorização das bobinas mais antigas devem fazer parte da solução do problema e é apresentado na Seção 3.2.2 um melhor detalhamento.

3.2.2 Função Objetivo

Problemas das áreas de engenharia, finanças, telecomunicações entre outras utilizam a simulação computacional para quantificar condições reais de processo com objetivo de aumentar performance, lucros, diminuir custos e tempo de processo. Se tratando de otimização, uma função objetivo (FO) faz a medição da qualidade da simulação e em casos reais podem ser necessários profissionais com experiência e conhecimento específico de processo para avaliação do resultado.

Em (KARGUPTA; GOLDBERG, 1997) os autores citam que alguns modelos não podem ser gerados por um modelo matemático analítico e a FO acaba se tornando do tipo "caixa preta". Os testes são realizados passando-se os dados simulados no processo para um modelo que avalia os dados em sua entrada x de forma que a saída da "caixa preta" retorne $f(x)$. Alguns exemplos citados de aplicação são otimização de forma, calibração de um modelo (físico ou biológico) e calibração de parâmetros.

Muitos exemplos práticos se encaixam na classe de problemas de otimização interativa, onde o resultado se deve à ajustes de preferências particulares, como por exemplo, conjunto de cores da interface do usuário ou imagens em evolução, formas animadas em 3D, várias artes e *designs* (TALBI, 2009). Deseja-se mostrar que a avaliação dos resultados de uma FO é uma etapa importante pois traduz o resultado da simulação possibilitando ao tomador de decisão influenciar no resultado de forma atingir a otimização ideal.

Em (TALBI, 2009) é apresentado a dificuldade em gerar soluções ao se trabalhar com funções multiobjetivo. Função mono-objetivo é mais bem explorada nos problemas da literatura se comparados com problemas multiobjetivos. As soluções dos problemas multiobjetivos são parciais e precisam da escolha do tomador de decisão. Também comenta que solução ótima para problemas multiobjetivo é um conjunto de soluções definidas como solução Pareto ótimas. Uma solução é Pareto ótima se não é possível melhorar um certo objetivo sem deteriorar pelo menos outro objetivo. O número de soluções Pareto ótimas aumentam de acordo com o número de funções objetivos e tamanho da instância do problema. Recomenda-se ao leitor o texto de (TALBI, 2009) e (DEUTZ A.H., 2018) para mais detalhes sobre Função Multiobjetivo.

Neste trabalho é aplicado o conceito de função mono-objetivo. A FO do clássico

Problema da Mochila, Equação (3.1), visa gerar cargas com o maior benefício possível. Associando-se este conceito neste estudo de caso, o benefício está em formar cargas maiores também priorizando bobinas mais antigas. Existe uma degradação natural da qualidade da bobina por tempo parado em estoque aguardando a formação da carga. Neste caso, é adicionado ao modelo matemático o conceito do "Tempo de Bonina Aguardando em Estoque" ($Tbae$) que caracteriza um benefício para, além de gerar cargas com o maior peso possível, priorizar as bobinas mais antigas em estoque, preservando as características de qualidade. Simulações são geradas na Seção 4.3 para correlacionar a priorização do $Tbae$ no peso total das cargas geradas.

O cálculo do tempo que a bobina fica aguardando no estoque é definido pela diferença entre o dia em que a bobina foi produzida no equipamento anterior e o dia atual e pode ser representado pela Equação (3.6).

- Tbae: Tempo de bobina aguardando em estoque. O valor é no mínimo um.
- Dpb: Dia de Produção da Bobina.
- Da: Dia atual.

$$Tbae = Da - Dpb \tag{3.6}$$

Devido a grande variabilidade do cenário deste problema e também as mudanças diárias de um processo siderúrgico, nesta dissertação, criou-se fatores com objetivo de dar ao tomador de decisão do processo a seguinte possibilidade de escolha: a) Priorizar a formação de carga maximizando peso. b) Priorizar as bobinas mais antigas em estoque. Desta forma, pode-se julgar, através dos resultados da simulação computacional o valor dos fatores ideais. Um fator α é aplicado na parcela da função objetivo que prioriza o peso da carga e outro fator β é aplicado na parcela do $Tbae$. Na Seção 3.2.3 é apresentado a função objetivo final aplicado neste estudo de caso.

3.2.3 Modelo Proposto Final

Apresenta-se a formulação matemática final utilizando conceitos de problema de programação linear inteira (PLI) no problema das múltiplas mochilas com conflitos de itens aplicada em planta de recozimento em caixa.

- $x_{ij} = \begin{cases} 1, \text{ se a bobina } i \text{ é associado ao abafador } j, \\ 0, \text{ caso contrário.} \end{cases}$

- M é o conjunto de bobinas.

- N é o conjunto de abafadores.

- m é o número total de bobinas.

- n é o número de abafadores.

- p_i é o peso da bobina i.

- l_i é a largura da bobina i.

- $Spec_i$ é a especificação da bobina i.

- $Tbae_i$ é o tempo da bobina i aguardando em estoque.

- c_j é a capacidade de peso máximo permitido para cada abafador j.

- a_j é a altura máxima de cada abafador j.

- Pm_j é o peso mínimo aceitável para gerar uma carga j.

- Nm_j é o número mínimo de bobinas aceitável para gerar uma carga j

- α fator entre zero e um para priorização do peso da carga j.

- β fator entre zero e um para priorização do $Tbae$.

Seja a formulação matemática composta pelo conjunto de equações:

$$Max \quad \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}(\alpha\ p_i + \beta\ Tbae_i\ ) \tag{3.7}$$

S.a.

$$\sum_{i=1}^{m} p_i\ x_{ij}\ \leq\ c_j, \forall\ j = 1, ..., n \tag{3.8}$$

$$\sum_{i=1}^{m} l_i\ x_{ij}\ \leq\ a_j, \forall\ j = 1, ..., n \tag{3.9}$$

$$\sum_{i=1}^{m} x_{ij}\ \geq\ Nm_j, \forall\ j = 1, ..., n \tag{3.10}$$

$$\sum_{i=1}^{m} p_i\ x_{ij}\ \geq\ Pm_j, \forall\ j = 1, ..., n \tag{3.11}$$

$$\sum_{j=1}^{n} x_{ij} \leq 1,\ \ \forall\ \ i = 1, ..., m \tag{3.12}$$

$$x_{ik} + x_{jk}\ \leq\ 1, \forall k \in N, \forall (i,j) \in [(i,j) : i \in M, j \in M, Spec_i \neq Spec_j] \tag{3.13}$$

$$x_{ij} \in \{0, 1\},\ \ \forall\ \ i = 1, ..., m,\ \ \forall\ \ j = 1, ..., n \tag{3.14}$$

A formulação final tem adicionado à função objetivo (3.7), a parcela que prioriza o tempo da bobina aguardando em estoque. Observa-se que o peso de cada bobina tem unidade *Kilogramas* ou toneladas e o *Tbae* tem unidade de dias no estoque. Para unificar esta diferença de unidades, foi adotado uma normalização dos dados e também um "certo controle" ao se buscar a priorização seja em formar maiores cargas ou priorizar o *Tbae* através dos fatores α e β. O conjunto de restrições (3.8) garantem que o somatório dos pesos das bobinas i selecionadas não ultrapasse a capacidade do abafador c_j. O conjunto de restrições (3.9) garantem que o limite de altura do abafador a_j não seja ultrapassado pelo somatório das larguras das bobinas l_i selecionadas. Em (3.10) o conjunto de restrições para garantir que só forma-se cargas com um número mínimo de bobinas. Já em (3.11) só podem ser formadas cargas acima de um peso mínimo para atender uma questão de lucratividade. O conjunto de restrições (3.12) asseguram que a bobina selecionada só seja incluída em um abafador por vez. O conjuntos de restrições (3.13) trata o conflito entre os itens, ou seja, as bobinas são incluídas na carga, desde que, sejam todas da mesma especificação. Em (3.14) a declaração da variável de decisão x_{ij} fazer parte do conjunto de variáveis booleana.

Em (NAYAK; MISRA; BEHERA, 2012) é citado que a normalização de dados é um

fator fundamental na etapa de pré-processamento. Encontrar um método para normalizar os dados é uma das etapas mais críticas em um processo de manipulação de dados. Dentre os vários métodos apresentados a normalização *Min-Max* dos dados é utilizada neste trabalho. Nesta técnica os dados de entrada são mapeados dentro de uma faixa de valores [0,1] ou [-1,1]. O método *Min-Max* normaliza os valores utilizando o atributo A dos dados de acordo com os valores mínimo e máximo. O valor a do atributo A torna-se $\hat{a}$ na faixa de valores *[baixo, alto]*. A recomendação é que os valores $minA$ e $maxA$ estejam contidos dentro da faixa dos dados a serem normalizados. Veja a Equação (3.15).

$$\hat{a} = baixo + \frac{(alto - baixo) * (a - minA)}{maxA - minA} \tag{3.15}$$

Portanto, a normalização dos dados é utilizada somente na função objetivo, de forma que, os pesos das bobinas em cada instância do problema tem seu valor transformado entre zero e um, assim como, os valores do atributo da bobina chamado $Tbae$. Desta forma a diferença da grandeza do peso da bobina em tonelada e o $Tbae$ em dias passam a trabalhar em mesma escala adimensional. Vale ressaltar que o peso das bobinas utilizada na restrição de peso máximo da carga permanecem em toneladas.

Na Seção 4.1.1 define-se os dados de testes para analisar a influência da priorização do tempo de bobina aguardando no estoque em relação ao peso da carga gerada através dos parâmetros α e β.

3.3 Descrição do Algoritmo Genético Proposto

Nesta seção é apresentada a implementação do algoritmo genético para resolução do problema das múltiplas mochilas com conflito de itens aplicado em planta de recozimento em caixa. O funcionamento deste AG inclui o conteúdo apresentado na revisão bibliográfica e está acrescido de novas necessidades do processo. A função objetivo deve, além de maximizar o peso da carga, também ter a flexibilidade de priorizar as bobinas mais antigas em estoque. Novas restrições são propostas: garantir que bobinas de mesma especificação sejam obrigatoriamente agrupadas em mesma carga e respeitar o limite máximo de altura, além das tradicionais restrições do PMM, como, uma bobina só deve estar em uma carga por vez e respeitar a capacidade de peso. Adicionam-se duas questões de lucratividade: só podem ser formadas cargas com no mínimo 3 bobinas e máximo 5 e peso mínimo de 60t.

3.3.1 Representação do Cromossomo Utilizado no Algoritmo Genético

A representação do cromossomo utilizado neste trabalho foi baseado em (FUKUNAGA, 2008) e pode ser vista na Figura 3.1 e basicamente consiste nos genes representar as bobinas e cada conjunto de cinco bobinas representam cargas ou múltiplas mochilas M = $\{m_1, m_2, ..., m_k\}$. O conteúdo do gene, ou valor do alelo, é o identificador da bobina contida em um vetor de posições $id\_bobina = \{b_1, b_2, ..., b_n\}$. Na Figura 3.1, um exemplo de cromossomo que contém no primeiro gene a bobina 7, seguida pelas bobinas 9, 3, 4 e 5. Estas 5 bobinas formam uma primeira carga no cromossomo. Os números nas posições de 6 a 10 representam uma segunda carga no cromossomo, e assim sucessivamente.

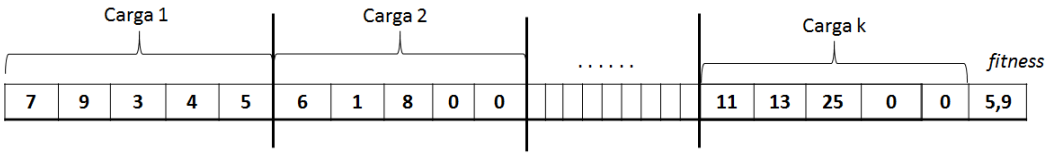

Figura 3.1: Modelo de cromossomo com k cargas representando um indivíduo com máximo de cinco bobinas por carga.
FONTE:(Próprio autor)

Assim uma restrição de máximo cinco bobinas por carga é imposta, fixando o tamanho da carga em cinco genes. Observa-se que alguns genes contém o valor zero, que significa nesta posição não há bobina associada. O valor zero foi utilizado ao lugar de vazio para melhor manipulação dos dados nas operações de *crossover* e mutação.

O tamanho do cromossomo varia em função do número de cargas, ou seja, caso queira-se resolver o PMM para 3 cargas, então o tamanho do cromossomo são de 5 genes vezes 3 cargas mais 1 totalizando 16 genes, visto que a última posição do cromossomo é reservada para armazenar o valor do *fitness* deste indivíduo, que é abordado na seção 3.3.4. Os atributos peso, largura, especificação das bobinas estão contidos na Tabela 4.1.

3.3.2 Processo Evolucionário

O pseudocódigo do programa principal do algoritmo genético pode ser visto no Algoritmo 1. Porém, antes de definir o processo evolucionário, um experimento para encontrar os parâmetros próximos dos valores ideais foi realizado, ou seja, os valores dos parâmetros que o algoritmo entrega o melhor resultado. Para isso, foi elaborado uma heurística para armazenar o resultado dos testes em arquivo de bloco de notas com os seguintes dados:

número de bobinas, tamanho da carga, elitismo, probabilidade de mutação por gene o por carga, número de cromossomos, função objetivo, tempo de execução. Extraiu-se deste banco de dados os parâmetros de probabilidade mutação por gene (Pmg), probabilidade de mutação por carga (Pmc), elitismo e a relação entre o tamanho da população em função do número de bobinas que desempenharam o melhor valor para a função objetivo.

- *critério de parada* = 150; O AG termina o processo evolucionário quando ultrapassar *critério de parada* gerações sem ocorrer melhora do melhor indivíduo da população.
- *reinício população* = 50; Número de gerações sem evolução para que ocorra o reinício da população. Detalhes deste mecanismo são explicados na seção 3.3.7
- Quantidade de cargas ($QtdCargas$) = 3; Número de cargas a serem formadas no problema (número de mochilas), sendo este exemplo 3.
- α = Coeficiente adimensional que quantifica a relevância do peso das bobinas para a função objetivo, quanto maior o valor (no intervalo de 0 a 1), tendência maior para formar cargas mais pesadas. Aumentar esse valor implica em diminuir a prioridade dada ao tempo de estoque das bobinas. O valor deste parâmetro deve ser escolhido de acordo com a experiência ou necessidade do processo.
- β = 1 - α; Coeficiente adimensional que quantifica a relevância do $Tbae$ para a função objetivo, quanto maior o valor (no intervalo de 0 a 1), maior a prioridade para a escolha de bobinas mais velhas para formar carga.
- quantidade de filhos desejados ($QtdFilhosDesejados$)= N*(1-elitismo) = 70% da população com novos filhos criados a cada geração arredondados para baixo. O elitismo é 0,3, ou seja, 30% dos melhores indivíduos não morrem ao terminar uma geração. N = (numero de bobinas) * 0,62, ou seja, o número de cromossomos nesta população é configurado para 62% do número de bobinas da instância.
- probabilidade mutação por gene (Pmg)= 0,08; Probabilidade de ocorrência de mutação de um gene em um filho gerado (troca de uma bobina).
- probabilidade de mutação por carga (Pmc)= 0,005; Probabilidade de ocorrência de mutação por carga, que significa a troca de uma carga inteira em um filho gerado.
- *população* = criado a partir da função *população inicial*, explicado na seção 3.3.3.

Algoritmo 1 *Algoritmo Genético*

1: **Entrada**:(*critério de parada, reinício população, QtdCargas,* α*,* β*, QtdFilhosDesejados, Pmg, Pmc, população*)
2: Salvar Melhor Indivíduo (MI) da *população*; ▷ MI = Primeiro cromossomo
3: **enquanto** *critério de parada* não satisfeito **faça**
4: Incrementa contador de geração;
5: Criar Conjunto de Filhos (CF) vazio baseado em *QtdFilhosDesejados*;
6: **enquanto** *TotFilhosVal* (CF) $<$ *QtdFilhosDesejados* **faça**
7: Gerar a partir da *população* Filhos por *Crossover* e inserir no CF;
8: Expor CF a probabilidade aleatória de mutação por gene: *Pmg*;
9: Expor CF a probabilidade aleatória de mutação por carga: *Pmc*;
10: Excluir cromossomos repetidos no CF;
11: **fim enquanto**
12: Calcular o valor da FO de cada cromossomo $\in CF$;
13: Substituir os piores indivíduos da *população* pelo CF mesmo sem evolução;
14: Ordenar decrescente a *população* pelo valor da FO;
15: **se** (MI) geração atual $>$ (MI) **então**
16: Guardar qual geração houve a evolução;
17: Atualizar MI;
18: **fim se**
19: **se** condição de *reinício da população* for satisfeita **então** ▷ Algoritmo 7
20: Criar e validar cromossomos da nova população;
21: Calcular a FO da nova população;
22: Inserir MI na nova população;
23: Ordenar a nova população decrescente pelo valor da FO;
24: Atualizar o último reinício da nova população;
25: Atualizar *população* $\leftarrow$ nova população;
26: **fim se**
27: **fim enquanto**
28: **Exibir** MI;

O processo evolucionário depende dos métodos *crossover*, mutação por gene e mutação por carga na busca de gerar indivíduos melhores. A quantidade de filhos que são criados a cada geração é dependente do elitismo fixado no parâmetro "qtd filhos desejados". Na próximas seções são apresentados cada processo utilizado. Nesta implementação vale ressaltar que o crossover utilizado só geram filhos válidos, já que na geração da população inicial foram criados apenas filhos válidos mas não necessariamente bons.

O critério de parada utilizado é: atingir 150 gerações desde a última evolução do melhor indivíduo, portanto, um certo controle é feito para que o código não finalize em ótimo local, mas com a população sem evolução, não havendo portanto, um número fixo de gerações para que o código finalize. Tal controle torna o tempo de execução do algoritmo variável porém, permite que o espaço de busca seja melhor explorado.

[0] *TotFilhosVal* => Função que identifica o total de filhos válidos em *CF*.

3.3.3 Geração da População Inicial

A inicialização da população é feita com escolhas aleatórias das bobinas, conforme Tabela 4.1, que compõe as cargas, buscando obter soluções válidas, ou seja, permitir apenas soluções que atendam as restrições de capacidade máxima de peso e altura e que as bobinas sejam todas da mesma especificação. Seguindo a recomendação de (LINDEN, 2012), neste momento, busca-se gerar soluções explorando o espaço de busca, ou seja, gerar soluções com boa distribuição. Embora a explicação anterior pareça óbvia, a inicialização com apenas indivíduos válidos, sem se preocupar com a aptidão dos indivíduos, tornou mais eficiente a inicialização e demonstrou gerar soluções relativamente boas. Vale ressaltar que neste momento não se preocupa em testar soluções com indivíduos repetidos pois a própria evolução natural se encarrega de eliminá-los.

Os parâmetros de entrada da função para gerar a população inicial estão a seguir:

- número de bobinas = Comprimento do vetor *id_bobina*; Contém o número de bobinas do vetor *id_bobina*. Também são usados os vetores da Tabela 4.1 com as características das bobinas tais como, largura, peso e especificação.
- N = (numero de bobinas) * 0,62; O número de cromossomos nesta população é configurado para 62% do número de bobinas da instância.
- L = 5*(*QtdCargas*); Número de genes em um cromossomo vezes a quantidade de cargas solicitadas pelo usuário. Lembrando que o número 5 trata-se da restrição de número máximo de bobinas por carga.
- altura máxima = 4500; Altura máxima da carga em milímetros. As somas das larguras das bobinas não podem ultrapassar a altura máxima.
- peso máximo = 81000; Peso máximo da carga em Kg.

O pseudocódigo de geração da população inicial está representado no Algoritmo 2. Este retorna uma matriz denominada população em que cada linha representa o cromossomo com tamanho relacionado ao número de cargas desejadas definido anteriormente em N.

A função que cria a carga aleatória retorna um vetor com 5 elementos representando as bobinas que formam uma carga válida, ou seja, que atendem as restrições. Inicialmente, busca-se no vetor de especificação identificar o número de diferentes especificações

Algoritmo 2 *Função população inicial*

1: **PopInit**(*N, QtdCargas, especif, largura, peso*)
2: Inicializa matriz população(N, L);
3: **para** *cromossomo* $\leftarrow 1$ **até** N **faça**
4: **para** *carga* $\leftarrow 1$ **até** *QtdCargas* **faça**
5: população(cromossomo, 5*carga-4 : 5*carga) = carga aleatória(especif, largura, peso, população(cromossomo,:));
6: **fim para**
7: **fim para**
8: **retorna** população

existentes e cria-se um novo vetor $E_1 = [a\ b\ c\ d\ e\ f\ g....]$ com apenas as especificações existentes exclusivas. Sorteia-se em E_1 uma especificação para formar uma carga com as bobinas exclusivas desta especificação. Caso uma especificação já tenha sido sorteada e tenha bobinas desta especificação no cromossomo atual, somente as bobinas restantes podem ser usadas para evitar repetição de bobinas. O número de bobinas para formar a carga também é sorteada entre o mínimo de 3 e máximo de 5 bobinas. É importante salientar que não é permitida a reutilização de uma bobina já incluída anteriormente, e este quesito é tratado neste ponto do código. O máximo de 5 bobinas faz parte da restrição imposta pelo processo. Caso o número de bobinas disponíveis de determinada especificação seja 3 ou 4, é sorteado um número entre 3 e 4 para serem inseridas. O número mínimo 3 é utilizado para atender a condição de lucratividade estabelecida e caso haja uma especificação com número de bobina menor que três, sorteia-se outra especificação. Se as bobinas selecionadas para formar a carga atenderem as restrições de altura e peso máximo, então retorna esta carga para a função de geração da população inicial ou qualquer outra parte do programa em que foi chamada a função "carga aleatória". O pseudocódigo é apresentado no Algoritmo 3.

Um detalhe neste processo de criação de carga está na ordenação do vetor de cinco posições do maior para o menor valor. Supondo que as bobinas 3, 5 e 10 são selecionadas para formação da carga no vetor de cinco posições. A função gerar carga aleatória além de escolher as bobinas de mesma especificação no banco de dados, ordena decrescente as bobinas escolhidas para devolver a carga para a função que a chamou: [10 5 3 0 0]. É melhor explicado o benefício desta operação no cálculo da diversidade da população na seção 3.3.7.

Enquanto não formar carga válida, o processo de criar carga aleatória é repetido. Após todos os cromossomos serem preenchidos com as cargas, ou seja, a população inicial ser criada, a função de avaliação é chamada para analisar a aptidão de todos os indivíduos.

Algoritmo 3 *Função Carga Aleatória*

1: **Entrada**:(*especificação, largura, peso, cromossomo*)
2: Inicializar E com valores distintos da *especificação*;
3: Inicializar *carga* $\leftarrow \emptyset$;
4: **enquanto** teste de restrições da *carga* for inválido **faça**
5: $BDE \leftarrow \emptyset$; ▷ Vetor Bobinas Disponíveis da Especificação ;
6: **enquanto** Conteúdo de BDE $<$ 3 ítens **faça** ▷ Garantir a lucratividade.
7: Sortear uma especificação em E;
8: Salvar em BDE os itens sorteados com especificação igual a E;
9: **fim enquanto**
10: Sortear um *número* entre 3 e 5 para criar a *carga*; ▷ Garantir restrições.
11: Criar *carga* a partir de BDE com *número* de itens não repetidos em *cromossomo*;
12: **se** *carga* é válida **então**
13: Salvar *carga* e finalizar *loop*;
14: **fim se**
15: **fim enquanto**
16: Ordenar *carga* decrescente;
17: **retorna** *carga*;

Os detalhes desta operação são mostrados a seguir.

3.3.4 Função de Avaliação

A função de avaliação utiliza a Equação (3.7) definida no método exato. Esta função deve receber um cromossomo por vez para ser avaliado, assim como os parâmetros α como coeficiente que quantifica a relevância do peso para a função objetivo, o parâmetro β como coeficiente que quantifica a relevância do tempo da bobina aguardando no estoque, além dos valores dos pesos e $Tbae$ normalizados conforme Equação (3.15). Vale ressaltar que é ignorado pela função objetivo o possível valor zero em um dos genes do cromossomo.

Após calculado o valor da função objetivo é inserido na matriz população o valor do *fitness*. Assim, a função objetivo sempre vai receber o cromossomo com os genes a serem avaliados, calcular o *fitness* correspondente ao valor de cada cromossomo e inserir este valor na última posição (coluna) da linha que corresponde ao cromossomo em avaliação. Em seguida ordena-se a matriz população em ordem decrescente em função do *fitness*. Nesta implementação a população também é re-ordenada após cada nova geração, pois é o momento em que os filhos são inseridos.

3.3.5 Operador de *Crossover*

O processo de *crossover* inicia com a necessidade de gerar 70% de novos filhos em relação a população, conforme definido pelo elitismo, que visa completar a "qtd filhos desejados".

A seleção dos cromossomos candidatos para geração dos novos filhos utiliza o método da roleta viciada *(Roulette wheel selection)*, sendo assim a probabilidade de seleção do cromossomo é proporcional à sua aptidão. Dessa forma, o indivíduo com maior *fitness* tem maior chance de ser escolhido para gerar descendentes ao passo que os indivíduos menos aptos também participam do processo mas com menor chance.

Baseando-se em (FUKUNAGA, 2008) o processo de *crossover* utilizado é uniforme, visto que cada carga pode ser considerada um "gene". Uma atenção deve ser tomada aqui para não confundir o gene, que é a bobina na população, com a sugestão deste autor ao tratar uma carga no processo de crossover com gene.

A ideia por trás do processo do *crossover* implementado é da seguinte forma: selecionam-se dois indivíduos considerados "pais" intercale as cargas dos pais para gerar os filhos. Neste mecanismo, um dos filhos herdará a primeira carga de c1, a segunda de c2, a terceira de c1 e assim sucessivamente até formar uma solução completa. Aqui cabe ressaltar que uma carga corresponde a um conjunto de bobinas (de 3 a 5 bobinas). Para exemplificar, suponha dois cromossomos de três cargas: c1[a, b, c] e c2[d, e, f]. Estes cromossomos são selecionados para serem os pais de dois filhos: f1 [a, e, c] e f2[d, b, f]. O Algoritmo 4 exibe o pseudocódigo da função de *crossover*.

A vantagem da codificação utilizada é o menor custo computacional ao se fazer trocas de posições fixas entre os cromossomos selecionados, não necessitando de um processo adicional de sorteio de um ponto de corte ou dois pontos de corte, como é o caso do *crossover* de ponto(s). A aleatoriedade do algoritmo genético implementado fica por conta do método da roleta no processo de *crossover* e dos novos processos de mutação a serem apresentados.

Neste processo de *crossover* a única validação é verificar se não houve repetição de bobinas no filho1 e filho2 por vez, desconsiderando os valores zero, visto que todas as cargas contida na população já são válidas. Caso haja repetição, o filho é descartado e o processo de *crossover* continua até que se gere todos os filhos definidos em "qtd filhos desejados". O processo de comparação e validação contabiliza o número de elementos únicos (não repetidos) e diferentes de zero contra o número de elementos diferentes de

Algoritmo 4 *Função Crossover*

```
1: Entrada:(QtdCargas, população, QtdFilhosDesejados)
2: Criar Conjunto de Filhos (CF) com tamanho QtdFilhosDesejados;
3: CFG ← 0;                                          ▷ Contador de Filhos Gerados
4: enquanto CFG < QtdFilhosDesejados faça
5:     Sorteia na população o cromossomo1 pelo método da roleta viciada;
6:     Sorteia na população o cromossomo2 pelo método da roleta viciada;
7:     Inicializar Filho1 ← cromossomo1;
8:     Inicializar Filho2 ← cromossomo2;
9:     para i ← 2 até QtdCargas passo 2 faça
10:        Filho1[5*i-4 : 5*i] ← Carga[5*i-4 : 5*i] do cromossomo2;
11:        Filho2[5*i-4 : 5*i] ← Carga[5*i-4 : 5*i] do cromossomo1;
12:    fim para
13:    se (não existe repetição de gene no Filho1) e (CFG < QtdFilhosDesejados) então
14:        Incrementa CFG;
15:        Inserir o Filho1 no CF;
16:    fim se
17:    se (não existe repetição de gene no Filho2) e (CFG < QtdFilhosDesejados) então
18:        Incrementa CFG;
19:        Inserir o Filho2 no CF;
20:    fim se
21: fim enquanto
22: retorna CF;
```

zero. Se houver repetição a igualdade é falsa, caracterizando a repetição de bobina no cromossomo gerado.

Este processo também tem a característica de aceitar piora através das gerações não impedindo de inserir um indivíduo supostamente ruim nos novos filhos gerados. Este filho pode até ser um indivíduo com baixo *fitness* mas não significa que traços de uma boa genética não existam, de tal forma que sua combinação com outro filho possa ser reaproveitada em futuras gerações. Se o indivíduo for ruim, o próprio processo evolucionário se encarrega de descartá-lo. Este comportamento pode ser observado através da Figura 3.2 observando a média do *fitness* da população. A diversidade começar a baixar, ou seja, a média do *fitness* da população estagna próximo ao valor do melhor indivíduo, então, há oscilações em torno na média para mais e para menos, indicando que indivíduos ruins estão participando do processo evolutivo e sendo inseridos na população.

O "vale" apresentado na Figura 3.2 é um recurso utilizado para reiniciar a população pois, trata a estagnação do melhor indivíduo desde a sua última evolução. Aqui se chama de "*reset* da população".

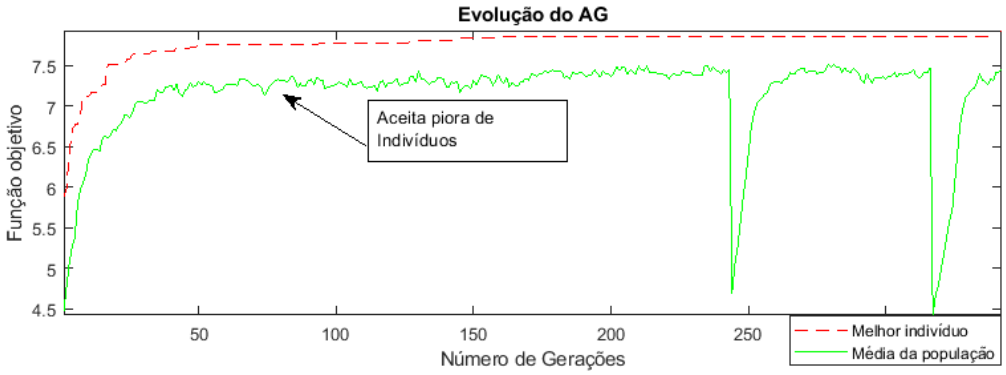

Figura 3.2: Processo evolucionário do algoritmo genético implementado.
FONTE: (Próprio autor)

3.3.6 Operador de Mutação

O operador de mutação é fundamental para um AG, pois garante a continuidade da diversidade genética da população. Como visto, o operador de *crossover* acaba contribuindo para um processo de igualdade entre os indivíduos. Nesta seção, busca-se criar uma heurística exploratória de forma que novos cromossomos sejam inseridos na população, ou seja, indivíduos até então inexistentes na população inicial.

O parâmetro que define a probabilidade de mutação deve ser escolhido de forma que a diversidade não fique tão baixa, ou seja, com baixa probabilidade de mutação, pois tende a estagnar em ótimo local. Por outro lado, se a probabilidade de mutação for alta, ocorre o que se chama na literatura de *random walk* que é uma ideia de busca para qualquer lado de forma não guiada (LINDEN, 2012).

Dois novos operadores de mutação foram criados, pois durante os testes observou-se que o tradicional operador de mutação, gerava muitas cargas inválidas, pelo fato de fazer a inserção de bobinas em uma carga que não pertencia a especificação desejada, com isso, havia um certo desperdício eliminando-se filhos válidos. Outra condição, inerente aos dados reais de processo, citado na Figura 4.6 é, a especificação que contém um número menor de bobinas mas, que podem formar excelentes cargas não entravam ou conseguiam ser escolhidos como possíveis candidatos. O motivo é o vasto espaço de busca proporcionado pela combinação 2^n, visto que a probabilidade de ocorrer o sorteio de um número pequeno de bobinas de mesma especificação em relação à uma base de dados maior com mesma especificação, se torna menos provável.

Assim, esta seção trada da criação de um processo de mutação guiada, através de certa

probabilidade de inserir, em um gene ou uma carga, as bobinas com mesma especificação. Uma nova heurística local precisou ser implementada para que fosse possível aumentar a eficiência na criação de filhos válidos, ou seja, superar os problemas inerentes aos ótimos locais.

3.3.6.1 Processo de Mutação por Carga

Todos filhos gerados no processo de *crossover*, passam pela possibilidade de sofrer a mutação por carga, ou seja, uma das cargas de todos os cromossomos podem sofrer a mutação de forma que, todas as bobinas correspondente a uma determinada especificação sejam trocadas. Esta operação viabiliza a inserção de especificações com menor número de bobinas mas, que podem formar ótimas cargas. A heurística utilizada no processo de criação de uma nova carga é a função cria carga aleatória comentada da seção 3.3.3.

O Algoritmo 5 exibe o pseudocódigo da função de *mutação por carga*.

Algoritmo 5 *Função Mutação por carga*

```
1: Entrada:(CF, Pmc, QtdCargas, peso, altura, especificação)
2: para i ← 1 até Tamanho CF faça
3:     para j ← 1 até QtdCargas faça
4:         se Rand() < Pmc então
5:             CF[i, 5*j-4 : 5*j] = carga aleatória(especificação, altura, peso, CF(i,:));
6:         fim se
7:     fim para
8: fim para
9: retornar CF;
```

3.3.6.2 Processo de Mutação por Gene

O processo de mutação por gene possibilita que cada gene dos filhos criados pelo processo de *crossover*, tenha a probabilidade de sofrer uma mutação. Ao "varrer" a matriz dos novos filhos, cada gene é exposto a probabilidade de mutação. Se o teste de probabilidade der verdadeiro, é encontrada a especificação da bobina presente no "vetor carga" visto que basta olhar a especificação de uma bobina. Pode acontecer que neste cromossomo, já exista uma carga formada com mesma especificação, assim, é necessário que haja no banco de dados mais bobinas disponíveis desta especificação de forma que uma nova bobina seja selecionada aleatoriamente e substituída no gene selecionado para mutação.

O Algoritmo 6 exibe o pseudocódigo da função de *mutação por gene*.

Algoritmo 6 *Função Mutação por gene*

```
1: Entrada:(CF, Pmg, peso, altura, especificação)
2: para cromossomo ∈ CF faça
3:     para gene ∈ cromossomo faça
4:         se (Rand() < Pmg) e (conteúdo do gene ≠ 0) então
5:             Encontrar no BD genes de mesma especificação, excluindo os genes da carga
   atual;
6:             Criar o conjunto M com resultado da busca anterior;
7:             se M ≠ vazio então
8:                 Trocar o gene por outro ∈ M;
9:                 Gerar nova carga que sofreu a mutação;
10:                se nova carga é válida então
11:                    Ordenar os dados da nova carga mutante decrescente pelo valor gene;
12:                    Substituir carga do cromossomo pela nova carga mutante;
13:                fim se
14:            fim se
15:        fim se
16:    fim para
17: fim para
18: retornar CF;
```

Para aumentar a eficiência do código algumas validações são feitas antes do processo de mutação: o valor do gene deve ser diferente de zero e precisa haver bobinas disponíveis da especificação sorteada. Após, o processo de mutação expõe a carga ao teste de restrição de altura e peso máximo. Ao fim deste processo, o vetor que contém as bobinas de cada carga é ordenado. Esta ordenação organiza em cada gene a posição da bobina no banco de forma decrescente, e pode ser visto na Figura 3.3, visando organizar os dados na matriz população para auxiliar o cálculo da diversidade. Então, o cromossomo é devolvido com o novo filho mutante.

Esta nova heurística permite explorar o espaço de soluções em busca do ótimo global. É um método "guiado" que aumenta a chance de encontrar a solução ótima. É um novo tipo de busca totalmente direcionado para este problema.

3.3.7 Módulos de Controle da População

Durante o processo evolucionário alguns módulos de controle foram implementados.

O primeiro é o módulo que elimina todos os indivíduos da atual geração pelos novos filhos criados através do processo de *crossover* e mutação. Este é o principal módulo de controle evolucionário. Como a população é ordenada do maior valor do *fitness* para o menor, os novos filhos substituem os piores conforme elitismo, assegurando que os

indivíduos considerados elites permaneçam a cada geração.

O segundo é o controle de convergência genética que demonstrou ser efetivo para encontrar soluções melhores em relação ao processo evolucionário tradicional que utiliza um número fixo de gerações para finalizar o algoritmo. A medição da diversidade da população foi inspirada no trabalho de (XING et al., 2007) demonstrada na seção 2.4.2.7. É uma forma de "*feedback*" do processo de evolucionário. Para (LINDEN, 2012) o cálculo da diversidade é considerado um algoritmo exaustivo, visto que testa-se todos os elementos uns contra os outros para verificar a diferença entre eles, porém nesta implementação, a codificação utilizada permitiu que o cálculo da diversidade fosse implementada a cada geração sem afetar o desempenho do processo evolucionário.

Como pode ser observado na Figura 3.3, há um exemplo extraído após finalizar a execução do AG. Observa-se que existem cinco indivíduos, ou cromossomos, sendo a linha superior a melhor solução encontrada e a última coluna o valor do *fitness*. Na carga 1 de todos os cinco cromossomos, têm-se quatro cargas iguais e apenas uma diferente. Este é um comportamento natural do AG ao fim do processo evolutivo, pois as soluções tendem a ser parecidas mas, nesta implementação, nunca são iguais.

Observa-se que a diversidade das bobinas na carga 1 é muito baixa se comparada com a carga 3. Como já mencionado, a ordenação decrescente por cargas dentro do cromossomo é utilizado neste ponto para o cálculo da diversidade. O objetivo desta ideia é contabilização das variações que acontecem por gene.

Carga 1					Carga 2					Carga 3					Fitness
103	102	70	6	0	99	85	81	4	0	135	91	55	53	37	7,548
103	102	70	6	0	85	81	75	4	0	56	53	39	37	30	7,547
103	102	70	6	0	99	95	69	4	0	135	86	56	55	53	7,536
104	103	102	6	0	127	124	69	7	0	135	56	55	53	37	7,500
103	102	70	6	0	85	82	81	4	0	56	53	39	37	30	7,498

Figura 3.3: Exemplo dos cinco melhores indivíduos da população. Cada linha representa um cromossomo. As bobinas são inseridas em cada carga de um total de três. A diversidade da carga 1 é baixa e a diversidade da carga 3 é maior devido às diferentes bobinas.
FONTE: (Próprio autor)

A cada nova geração, se o valor da função *fitness* aumentar, ou seja, o melhor cromossomo da geração atual evoluir, atualiza qual geração ocorreu a "ultima evolução". Da mesma forma, a variável "ultimo *reset*" é atualizada com o valor da iteração da geração atual. Então, ocorre o *reset* da população se ultrapassarem 50 gerações sem melhoras no

melhor indivíduo e passarem 50 gerações deste o último *reset* e o cálculo da diversidade estiver abaixo de 35% conforme o pseudocódigo no Algoritmo 7.

A condição de reinício da população pode ser observado na Figura 3.4. A última evolução antes do *reset* aconteceu na geração 164, então, logo após 50 gerações sem melhoras e com a diversidade abaixo de 35% acontece o primeiro *reset* próximo a geração 214.

Algoritmo 7 Teste para condição de reinício da população

```
se (geração atual - última evolução > 50 gerações) &
   (geração atual - último reset    > 50 gerações) &
   (diversidade da geração atual    < 0,35) então
   Reinício da população mantendo melhor indivíduo
fim se
```

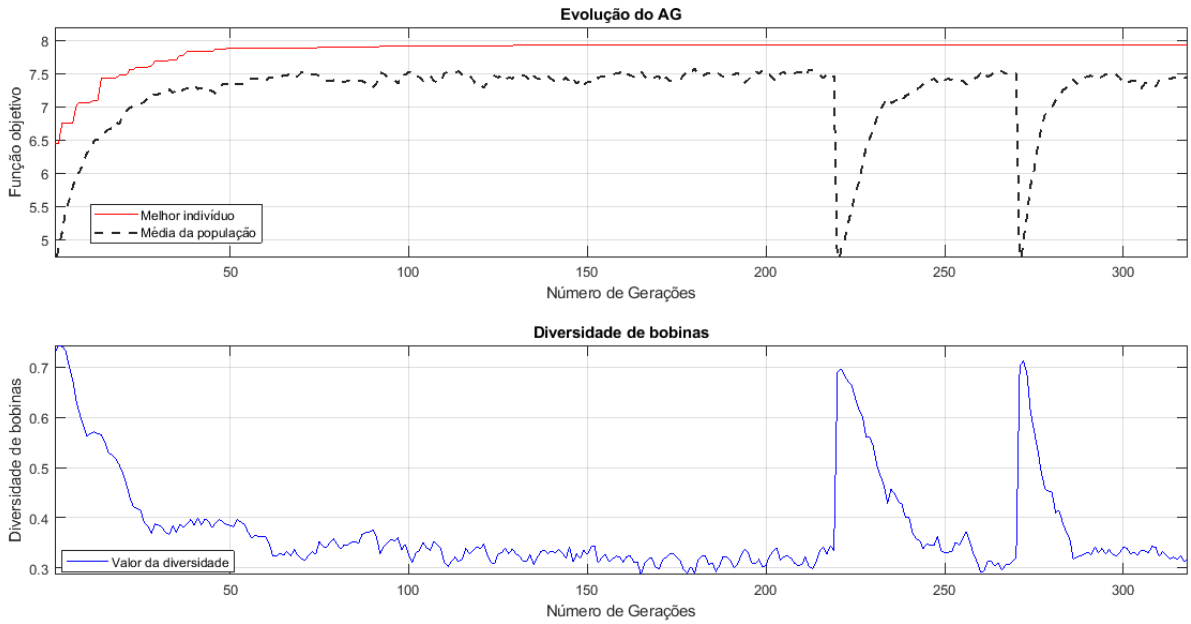

Figura 3.4: Gráfico do processo evolucionário do melhor indivíduo em cima e a diversidade de bobinas na população em baixo.
FONTE: (Próprio autor)

Para uma melhor visualização da geração com última evolução a Figura 3.5 tem a ampliação da imagem no ponto próximo ao 164. Um novo *reset* da população ocorreu 50 gerações depois, devido não ocorrer mais melhoras visto que neste exemplo encontrou-se o ótimo global.

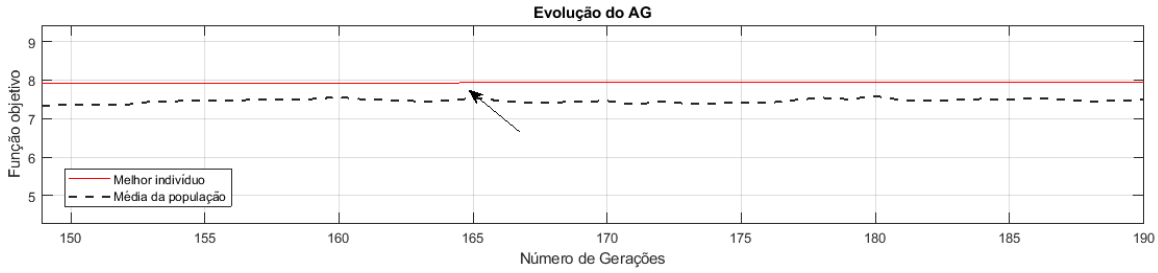

Figura 3.5: Gráfico do processo evolucionário do melhor indivíduo na geração 164.
FONTE: (Próprio autor)

Portanto, o número de gerações em busca do melhor indivíduo é variável neste trabalho e por consequência o tempo de execução até que o algoritmo seja finalizado.

Capítulo 4

Resultados

Nesta seção são apresentados e analisados alguns experimentos numéricos comparando os resultados obtidos pelas propostas apresentadas no Capítulo 3 para o problema de formação de carga em planta de recozimento em caixa.

4.1 Experimentos Computacionais

Experimentos numéricos foram conduzidos em uma máquina Intel Core i5-3230M de terceira geração com 16Gb de memória RAM, quatro CPUs de 2.6GHz e armazenamento SSD Samsung 860 QVO 1TB rodando no sistema operacional Windows 7, service pack 1 64 bits. O modelo de programação linear inteira foi implementado no *Solver IBM ILOG CPLEX Optimization Studio* versão 12.6.0 (IBM, 2013). Foram utilizadas configurações padronizadas do CPLEX.

O tempo para execução do método exato até encontrar a melhor solução é livre, contudo, um tempo teórico de 12 minutos (720 segundos) foi estabelecido como limite por questões práticas. Isso se deve ao fato de o operador ter outras atividades diárias concomitantes com a atividade de formação de carga através do algoritmo. Por exemplo, este mesmo operador faz a identificação das bobinas nos pátios de bobinas para transporte, gerencia o transporte, verifica questões de qualidade de cada bobina a ser enfornada entre outras atividades.

Neste trabalho, o CPLEX foi utilizado até a parada em duas situações: 1) quando a solução encontrada é ótima, e neste caso o resultado obtido é exposto e utilizado; 2) quando é reportado uma parada do tipo *out of memory*, indicando que memória do computador é insuficiente para solucionar a instância em questão e, neste caso, as soluções

intermediárias não são expostas nem utilizadas para comparação.

4.1.1 Geração do Banco de Dados e das Instâncias de Testes

Um banco de dados foi gerado com oitocentas bobinas a partir de valores aleatórios dentro de intervalos de largura, diâmetro interno e peso das bobinas. Apenas distribuição de probabilidade de ocorrência dentro do intervalo do peso é uniforme, assegurando uma grande variedade dos pesos de bobinas. Os dados de diâmetro externo das bobinas foram calculados a partir Equação (4.1). Seguem as faixas de valores das bobinas utilizadas com base em (BIGERELLE; HAOUAM, 2017).

- Peso entre 7000kg e 25000kg.
- Largura entre 600mm e 1600mm.
- Diâmetro interno igual a 500mm.
- Diâmetro externo entre 1000mm e 1800mm.

O Histograma visto na Figura 4.1 tem a distribuição de probabilidade uniforme dos pesos de 800 bobinas com faixa de variação dos dados em intervalos de 2000. O limite inferior referencia o valor mínimo de peso de 9000kg e o limite superior o valor de 25000kg sendo o superior incluído na contagem da classe. O número de intervalo de classes utiliza a regra de Sturges (SEWARD; DOANE, 2014) e totalizaram nove intervalos.

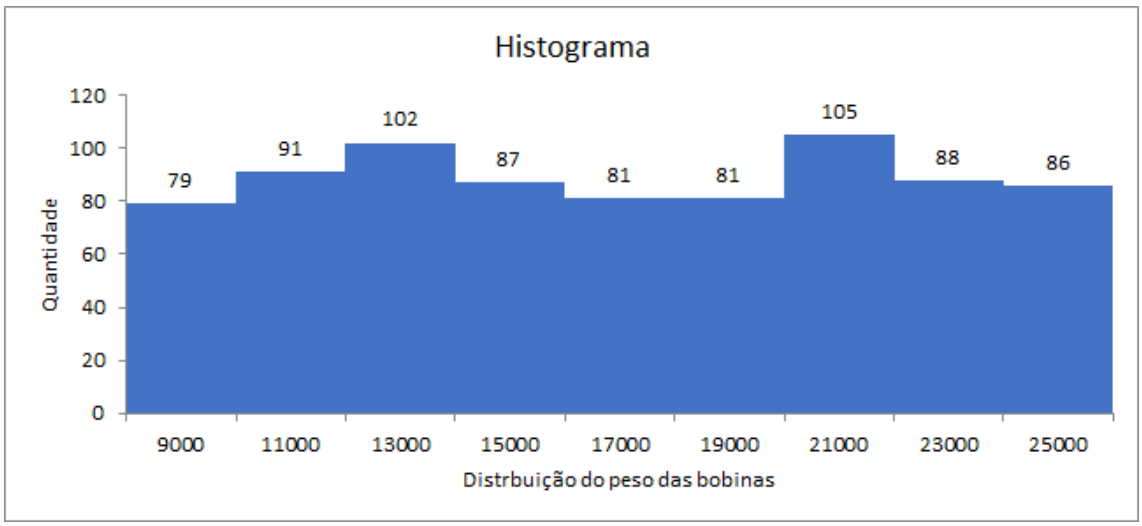

Figura 4.1: Distribuição do peso das bobinas em intervalo de 2000kg.
FONTE: (Próprio autor)

O Histograma da Figura 4.2 tem a distribuição de frequência das larguras no conjuntos de 800 bobinas com faixa de variação dos dados em intervalos de 100. O limite inferior referencia o valor mínimo de largura de 600mm e o limite superior o valor de 1600mm sendo este o superior incluído na contagem da classe. Observa-se que a distribuição de frequência das 800 bobinas têm os maiores valores localizados para o fim da faixa de largura.

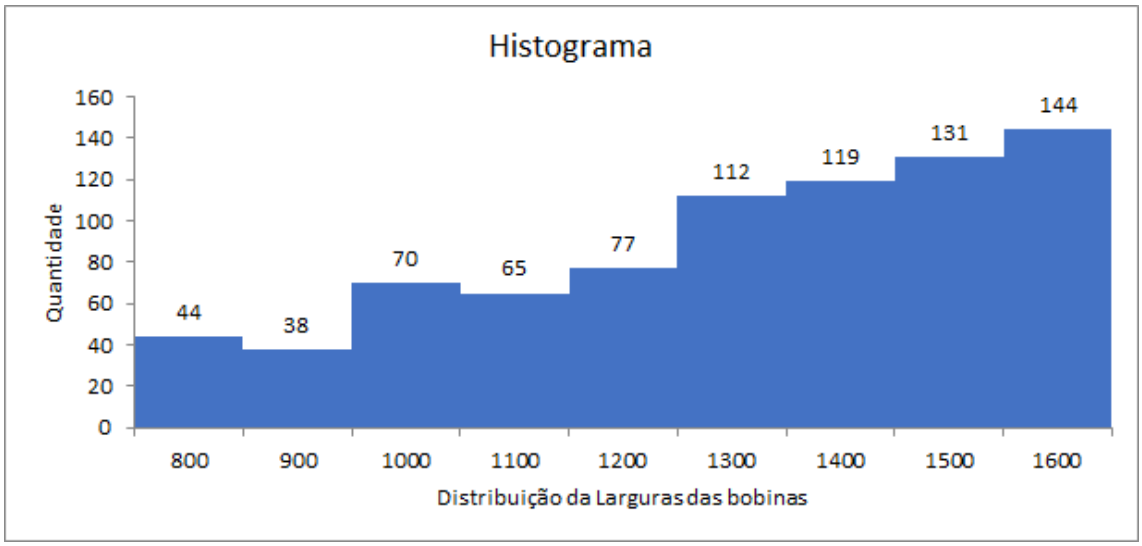

Figura 4.2: Distribuição de larguras em intervalos de 100.
FONTE: (Próprio autor)

A distribuição dos dados do diâmetro externo pode ser visto na Figura 4.3 em que os dados foram gerados utilizando a Equação 4.1 para averiguar as faixas de valores proposta

por (BIGERELLE; HAOUAM, 2017) e observa-se que estão entre os limites apresentado na referência.

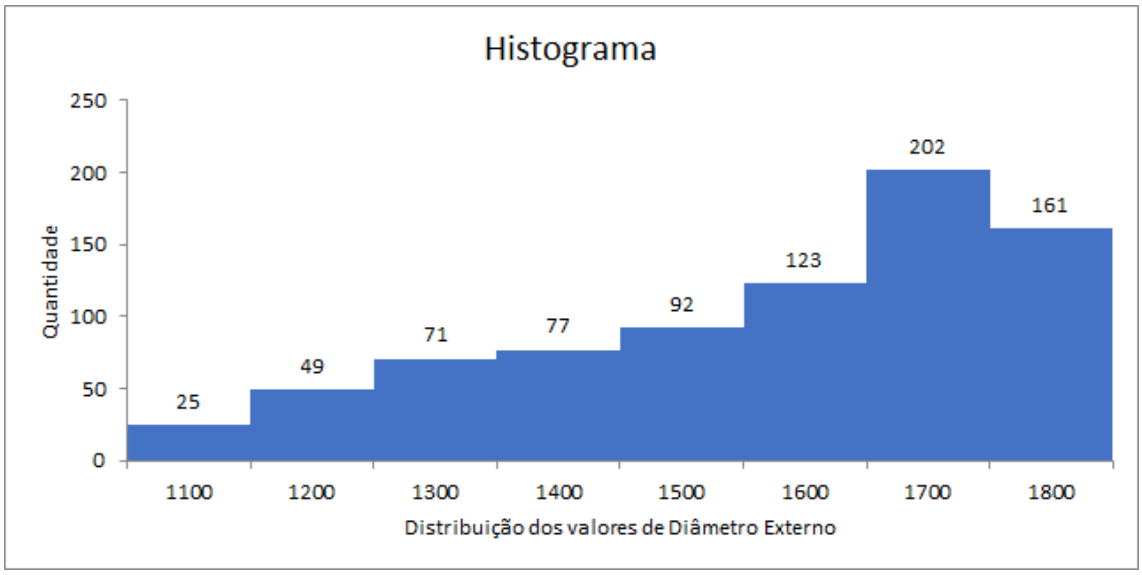

Figura 4.3: Distribuição do diâmetro externo das bobinas em intervalos de 1000.
FONTE: (Próprio autor)

Para o cálculo do diâmetro externo da bobina considera-se que a bobina é um cilindro de certo volume com o centro oco. O cálculo do volume foi feito pela diferença entre a área externa e a área interna multiplicado pela largura da bobina. O peso da bobina é a relação direta entre o volume pelo peso específico do aço. Seja:

- $Peso$ = massa da bobina (kg).
- D_{ext} = Diâmetro externo da bobina (m).
- D_{int} = Diâmetro interno da bobina (m).
- $largura$ = largura da bobina (m).
- ρ = peso específico do aço 7860 (kg/m3).
- π = 3,1415 (valor aproximado).

Tendo-se o peso, arbitrou-se o diâmetro interno em 500mm e isola o diâmetro externo para realização dos cálculos.

$$Peso = \frac{largura.\rho.\pi}{4} (D_{ext}^2 - D_{int}^2) \tag{4.1}$$

Para a característica da bobina relacionada ao tempo que a bobina fica aguardando em estoque para formar a carga, tem-se o *Tbae* o qual foi definido com a distribuição conforme consta na Figura 4.4. Observa-se que existem mais bobinas novas do que velhas no banco. Isso ocorre devido ao fluxo de trabalho em uma indústria, onde as bobinas entram no estoque e vão sendo removidas para o recozimento priorizando, se possível, as mais velhas. Desse modo, a maioria das bobinas tendem a ser direcionadas ao processo antes de envelhecerem no estoque pois todos os esforços são direcionados a não permitir que o produto venha a se degradar.

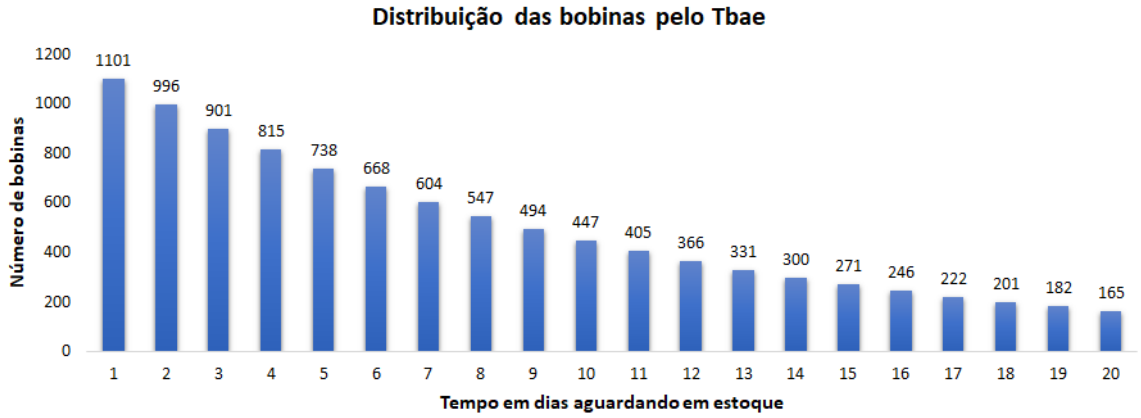

Figura 4.4: Tempo em dias que a bobina está aguardando em estoque.
FONTE: (Próprio autor)

Já a distribuição da especificação das bobinas, conforme Figura 4.5, também utilizou o mesmo comportamento de distribuição da Figura 4.4. Busca-se ter um maior número de bobinas de determinada especificação em relação a outra para que seja possível verificar a capacidade do algoritmo lidar com o problema de haver poucas bobinas de uma determinada especificação mas que podem formar ótima carga. Este comportamento é mais bem explorado no método heurístico no tratamento de ótimos locais. Desta forma, busca-se analisar se o algoritmo é capaz de explorar o espaço de busca quando apresentam um número comparativamente limitado de bobinas de uma especificação em relação as outras.

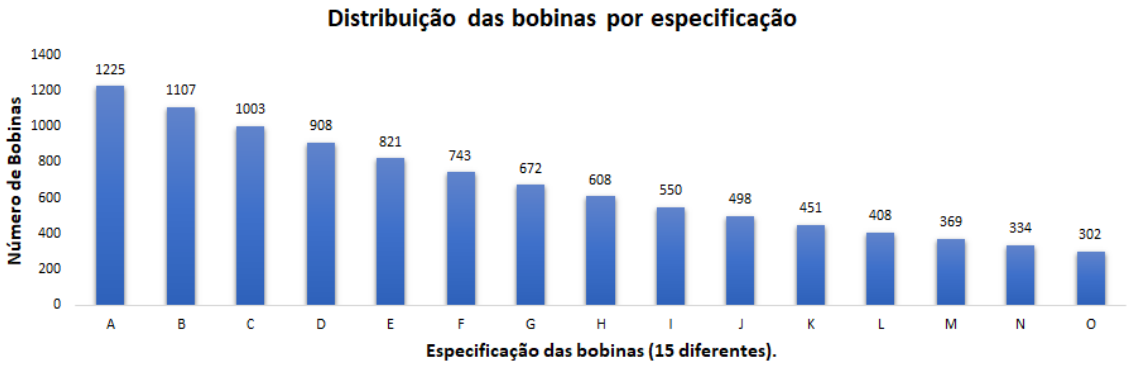

Figura 4.5: Distribuição do número de bobinas por especificação.
FONTE: (Próprio autor)

Portanto, após gerados os dados, apresenta-se uma amostra das primeiras bobinas na Tabela 4.1 a partir do banco de dados criado. A coluna bobina tem a identificação do nome da bobina com valores de um a oitocentas. A coluna massa tem o peso das bobinas em unidade de kg com distribuição de probabilidade uniforme. A coluna largura contém as larguras das bobinas dentro da faixa de valores especificada. A coluna diâmetro externo contém os valores calculados dos diâmetros externo de cada bobina. A coluna *Tbae* contém a distribuição das informações do tempo em que cada bobina aguarda em estoque e por fim a especificações dos materiais são aqui representados por letras.

Tabela 4.1: Banco de dados com oitocentas bobinas.

Bobina	Massa (kg)	Largura (mm)	Diâmetro Externo (mm)	Tbae	Especificação
1	7769	1144	1162	12	G
2	18793	1261	1632	13	A
3	18762	1017	1800	5	B
4	9859	1286	1221	1	E
5	12106	1367	1298	8	F
6	15375	991	1662	7	I
7	8097	1286	1127	17	E
8	8026	1226	1145	12	E
9	14793	1484	1366	6	F
10	9755	918	1404	2	K
⋮	⋮	⋮	⋮	⋮	⋮
800	18763	1018	1800	4	A

FONTE: (Próprio autor)

A partir do banco de dados gerado foram criadas instâncias do problema tratado de forma facilitar a compreensão dos resultados simulados. Portanto, cada instância criada inclui sempre as bobinas a partir da primeira, ou seja, a Inst1 tem as primeiras 50 bobinas, a Inst2 tem as primeiras 100 bobinas até a Inst5 com 800 bobinas. Inst5 trata-se de

uma instância de tamanho elevado para avaliar a capacidade de solução dos algoritmos. Também a divisão do banco de dados em instâncias de diferentes tamanhos servem para simular os problemas práticos em planta siderúrgica, pois muitas interferências no chão de fábrica podem variar a quantidade disponível de bobinas para formar cargas, como por exemplo, limitações logísticas, espaços para armazenamento, programação de produção, etc. Portanto, a Tabela 4.2 tem o nome da instância com suas respectivas quantidades de bobinas.

Tabela 4.2: Instâncias criadas a partir de banco de dados com 800 bobinas.

Instância	Qtd Bobina
Inst1	50
Inst2	100
Inst3	200
Inst4	400
Inst5	800

4.1.2 Análise da Instância Inst4 com 400 Bobinas

Por questões didáticas, são apresentados os dados da Instância Inst4 que contém as primeiras 400 bobinas, ou seja, da bobina identificada como "1"até a bobina "400". O objetivo é verificar que as características gerais do problema são abordadas pelas instâncias tratadas.

Com relação a característica “especificação” das bobinas, observa-se na Figura 4.6, que há um maior número de bobinas da especificação “A” seguida pela “E, “B” e assim sucessivamente. Buscar-se-á analisar qual a influência na formação da carga por haver quantidade maior de uma especificação em relação as outras.

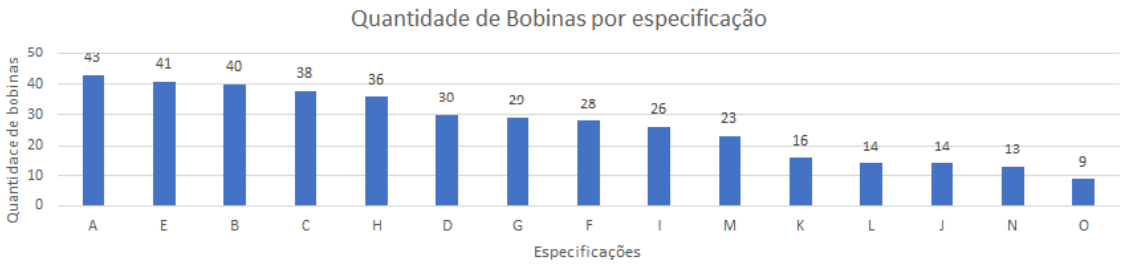

Figura 4.6: Quantidade de bobinas por especificações da instância de 400 bobinas.
FONTE: (Próprio autor)

É apresentado através da Figura 4.7 a quantidade de bobinas com *Tbae* maior que 15 dias agrupada por especificações. Para as simulações, os dados do *Tbae* tem distribuição

entre o valor um e vinte. Considera-se bobinas antigas todas que apresentam tempo de estoque superior a 15 dias. Na Figura 4.7 observa-se a distribuição das bobinas antigas agrupadas por tempo de estoque e separadas por especificação.

Este gráfico deve ser observado ao se formar cargas com priorização do *Tbae*. As bobinas mais antigas devem ser priorizadas para atender um dos requisitos de qualidade apresentados como exigência neste trabalho.

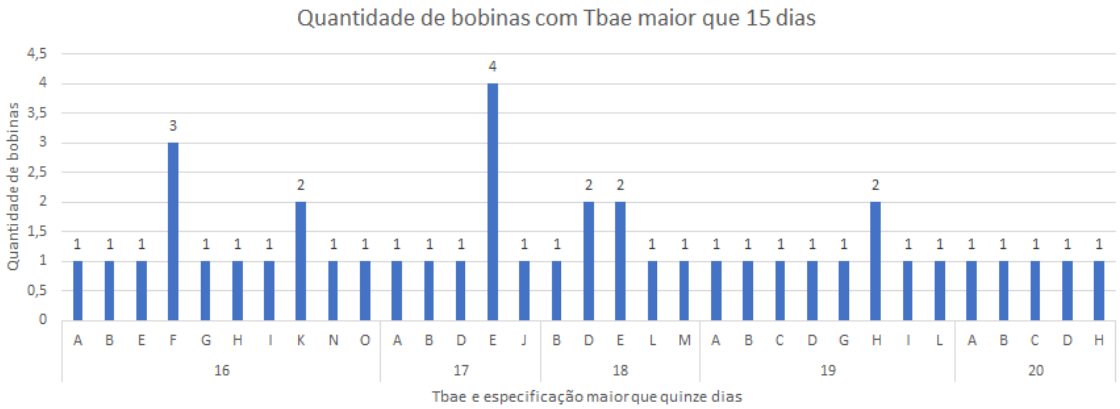

Figura 4.7: Quantidade de bobinas com *Tbae* maior que quinze dias por especificações. FONTE: (Próprio autor)

4.2 Resultados da Função Objetivo e Tempos de Execução

A Tabela 4.3 contém os valores da função objetivo e dos tempos de execução para comparações dos resultados. O tempo de execução no CPLEX retorna o tempo real e é denominado *wall clock time*. Podem ser observados que os testes iniciam sempre por duas cargas devido tratar-se de um trabalho focado no problema das múltiplas mochilas. Ademais, fora apontado o benefício de se trabalhar com o PMM ao lugar do PM aplicado neste estudo de caso na Tabela 4.11. Os casos onde não apresentam valores e contém o campo escrito "Memória", significa que o método exato não conseguiu resolver o problema proposto, acusando o erro de falta de memória para continuar a execução do código. Os parâmetros $\alpha = 0{,}8$ e $\beta = 0{,}2$ foram utilizados para todos os testes.

Por se tratar de um problema prático, o tempo de execução do código para solucionar uma certa instância, tem tempo sugerido limite em 720 segundos. A Tabela 4.3, é montada de forma a se avaliar o tempo computacional para resolução do problema até o limite de 800 bobinas. Para determinado número de bobinas, o número de cargas variam de duas

até sete, para assim, poder ser possível avaliar a capacidade do algoritmo resolver os problemas. Assim, verifica-se que um dos limites observados na execução do algoritmo está no número maior que 100 bobinas e 6 cargas, pois o problema atinge um tempo próximo do máximo aceitável para este processo através do método exato.

Tabela 4.3: Função objetivo e tempo de execução no Cplex

Nº itens	Nº cargas	FO - Cplex	Tempo(s)
50	2	4,8664	8,62
	3	7,1987	8,84
	4	9,3763	9,98
	5	11,5515	10,45
	6	13,433	18,90
	7	15,2749	36,60
100	2	5,6327	9,67
	3	8,1836	12,60
	4	10,7125	78,32
	5	13,1424	154,89
	6	15,5561	640,42
	7	17,9528	1693,36
200	2	6,1667	15,19
	3	9,1645	33,55
	4	12,0495	549,94
	5	14,9305	10141,09
	6	Memória	Memória
	7	Memória	Memória
400	2	6,9191	44,00
	3	10,2182	412,73
	4	13,4933	1695,73
	5	16,717	7520,90
	6	Memória	Memória
	7	Memória	Memória
800	2	7,1858	147,37
	3	10,6648	1845,65
	4	Memória	Memória
	5	Memória	Memória
	6	Memória	Memória
	7	Memória	Memória

A Figura 4.8 contém as informações dos tempos de execução computacionais variando-se o número de bobinas em 50, 100, 200 e 400 bobinas. É possível observar que se trata de um problema com tempo de execução exponencial ao se aumentar tanto o número de bobinas tendo como característica a expressão 2^n sendo o n o número de bobinas e $m2^n$ ao se aumentar o número de cargas sendo m o número de cargas. Porém, é possível

observar que para resolver o problema com 2 ou 3 cargas e máximo de 400 bobinas, este problema foi resolvido dentro do tempo pré-estabelecido sendo assim, considerado um bom resultado em termos de tempo computacional se o problema a ser resolvido puder ser limitado a estes valores.

Figura 4.8: Tempo de execução em segundos para n bobinas x n cargas.
FONTE: (Próprio autor)

4.3 Análise do Impacto dos Parâmetros α e β na Formação das Cargas

Esta Seção tem as simulações do modelo exato apresentado pelas formulações de (3.7) até (3.14) o qual todos testes são realizados com uma base de dados total de 400 bobinas chamada de (Inst4). Esta instância foi arbitrariamente escolhida para os testes desta seção afim de apresentar a possibilidade de formação de maiores cargas pelo fato de um maior número de bobinas gerar também uma maior diversidade.

A Tabela 4.4 é o resultado ou relação de bobinas que o algoritmo selecionou para formar duas cargas, ou seja, duas pilhas de bobinas. As duas cargas formadas têm as características das bobinas que obtiveram o maior valor da função objetivo para $\alpha = 0{,}8$ e $\beta = 0{,}2$, ou seja, 80% de priorização para formar cargas mais pesadas e 20% de priorização do $Tbae$.

O modelo exato encontrou como melhor solução as bobinas enumeradas na coluna "Bobinas". Observa-se que esta simulação, com a Inst4 tem as primeiras quatrocentas bobinas criadas no banco de dados, com isso, não podiam haver bobinas com numeração maior que o valor quatrocentos. O conjunto de restrições imposto em (3.12) garantem que, para as duas cargas formadas, não podem haver repetição de bobina, ou seja, se a bobina foi incluída em uma carga não pode estar em outra.

Para a coluna peso o importante é observar a soma dos pesos das bobinas selecionadas, pois não podem ultrapassar o peso máximo admitido para cada carga conforme (3.8), com valor máximo de 81 toneladas. Os somatórios do peso das cargas foram aproximadamente 73t e 79t, que representam valores próximos da capacidade máxima.

A coluna altura segue a mesma ideia pois a soma das larguras das bobinas, compõe a altura máxima permitida, conforme (3.9), que neste caso é 4500mm. Observa-se que as duas cargas formadas atenderam a restrição de altura pois a carga 1 ocupou próximo de 94% e a carga 2 99,2%.

A coluna especificação deve demonstrar que as cargas formadas possuem apenas bobinas de mesma especificação devido às restrições de conflito de itens tratadas em (3.13). Ademais, observa-se que embora haja um maior número de bobinas com a especificação "A" e "E" o algoritmo encontrou como melhor solução a carga de especificação "B" e "H". Isso se deve ao fato de haver também uma pequena parcela de priorização do $Tbae$, ou seja, tempo que as bobinas estão aguardando em estoque. Observa-se na coluna $Tbae$ os valores em número de dias das bobinas selecionadas. Pode-se notar que houve a inclusão de pelo menos uma das bobinas consideradas antigas (com mais de 15 dias) nas cargas formadas.

Apresentar-se-á uma nova simulação variando os valores para $\alpha = 0{,}5$ e $\beta = 0{,}5$. A Tabela 4.5 é o resultado da seleção para duas cargas e com os novos valores de α e β, que priorizam de forma idêntica o peso da carga e o tempo em estoque.

Pode-se observar que muitas bobinas foram trocadas em comparação com a Tabela 4.4. As bobinas de número 32 e 247 foram incluídas para a especificação "H" tendo respectivamente 19 e 12 dias devido à priorização das bobinas com $Tbae$ mais alto. É bem evidente a priorização do $Tbae$ observando o total em dias da coluna $Tbae$ visto que a soma na Tabela 4.4 foram de 135 dias se comparado com a Tabela 4.5 com 153 dias. Estes números somados não tem importância prática mas geram uma boa ideia de como o algoritmo se comporta em torno das priorizações das bobinas mais antigas.

Tabela 4.4: Formação de duas carga para α=0,8 e β=0,2

Bobina	**Carga**	**Peso(kg)**	**Altura(mm)**	**Especificação**	**Tbae(dias)**
274	1	11794	733	H	20
278	1	12859	727	H	19
346	1	12025	721	H	15
357	1	24234	1324	H	12
368	1	12386	719	H	14
Total		73298	4224		80

Bobina	**Carga**	**Peso(kg)**	**Altura(mm)**	**Especificação**	**Tbae(dias)**
293	2	20230	1210	B	20
317	2	18340	1092	B	11
326	2	19883	1096	B	12
336	2	18574	1066	B	12
Total		77027	4464		55

Outra importante característica está na substituição da especificação “B” pela especificação “E”. A especificação “B” tinha um carga ótima em termos de peso total da carga. Com a priorização do *Tbae* a carga que tinha total de 55 dias passou para 68 dias, porém, o peso total piorou bastante mas, ainda assim, é considerada uma boa carga pois ocupa próximo de 84% do peso máximo admitido.

Considera-se a altura da carga formada também relevante visto que as bobinas incluídas tem somatório próximo da altura máxima admitida para a carga 1 com 99,3% e carga 2 com 96,6%. Assim, percebe-se que o tomador de decisão deve avaliar o cenário real do processo e escolher quais das cargas melhor atende a necessidade do processo.

Finalmente, uma terceira simulação variando os valores para $\alpha = 0{,}2$ e $\beta = 0{,}8$. A Tabela 4.6 é o resultado da seleção de bobinas para duas cargas cujas características das bobinas levaram ao maior valor da função objetivo com 20% de priorização do peso da carga e 80% de priorização do *Tbae*. Observar-se que algumas bobinas foram trocadas em comparação com a Tabela 4.5. A bobina de número 247 foi substituída pela bobina 368 para a especificação “H” tendo respectivamente acrescido o *Tbae* da bobina trocada de 12 por outra de 14 dias, visto que, a soma na Tabela 4.5 foram de 153 dias contra 157 dias se comparado com a Tabela 4.6. A carga formada pelas bobinas da especificação “E” teve um maior impacto negativo em termos de peso total da carga pois ao formar nova carga o peso aproximado de 68 toneladas com uma pilha de 5 bobinas tornou-se, após priorização do parâmetro β o peso próximo de 54 toneladas e apenas 4 bobinas. Assim, atesta-se que houve priorização das bobinas mais antigas aplicando a priorização através do parâmetro

Tabela 4.5: Formação de duas cargas para $\alpha = 0{,}5$ e $\beta = 0{,}5$

Bobina	**Carga**	**Peso(kg)**	**Altura(mm)**	**Especificação**	**Tbae(dias)**
24	1	11386	1030	E	18
375	1	14230	887	E	14
385	1	14779	821	E	18
388	1	9668	705	E	9
393	1	17892	1027	E	9
Total		67955	4470		68

Bobina	**Carga**	**Peso(kg)**	**Altura(mm)**	**Especificação**	**Tbae(dias)**
32	2	14952	1018	H	19
247	2	17831	1148	H	12
274	2	11794	733	H	20
278	2	12859	727	H	19
346	2	12025	721	H	15
Total		69461	4347		85

β. Portanto, foi testado e comprovado que a funcionamento da função objetivo atende a necessidade de formar maiores cargas ou priorizar as bobinas mais antigas do processo.

Tabela 4.6: Formação de duas cargas para $\alpha = 0{,}2$ e β=0,8

Bobina	**Carga**	**Peso(kg)**	**Altura(mm)**	**Especificação**	**Tbae(dias)**
32	1	14952	1018	H	19
274	1	11794	733	H	20
278	1	12859	727	H	19
346	1	12025	721	H	15
368	1	12386	719	H	14
Total		64016	3918		87

Bobina	**Carga**	**Peso(kg)**	**Altura(mm)**	**Especificação**	**Tbae(dias)**
7	2	8097	1286	E	17
24	2	11386	1030	E	18
40	2	19964	1261	E	17
385	2	14779	821	E	18
Total		54226	4398		70

A Tabela 4.7 é um resumo das simulações e cada linha representa o tempo de cada um dos testes apresentados na execução do código. Observa-se que a priorização do parâmetro β também prioriza as bobinas mais antigas e pode ser visto na coluna *Tbae* que contêm o somatório dos dias das bobinas selecionadas em cada execução. Porém, degrada o peso total da carga gerada, o que era esperado. Outro fator importante está

no tempo de execução desta instância que apresentou tempo razoável e que tendeu variar com a alteração dos parâmetros α e β.

Tabela 4.7: Execução para formação de duas cargas com banco de dados de 400 Bobinas

Execução	Tempo(s)	Alpha	Beta	Peso Total (kg)	Tbae(dias)
1	70	0,8	0,2	150325	135
2	53	0,5	0,5	137416	153
3	46	0,2	0,8	118242	157

Para entender o fenômeno de variação do tempo de execução em função dos parâmetros, apresenta-se a Figura 4.9 com o histograma das primeiras 400 bobinas do banco de dados e observa-se que há uma distribuição uniforme para os pesos das bobinas porém, com relação ao tempo das bobinas esperando em estoque, há um número de bobinas antigas menor em relação as novas.

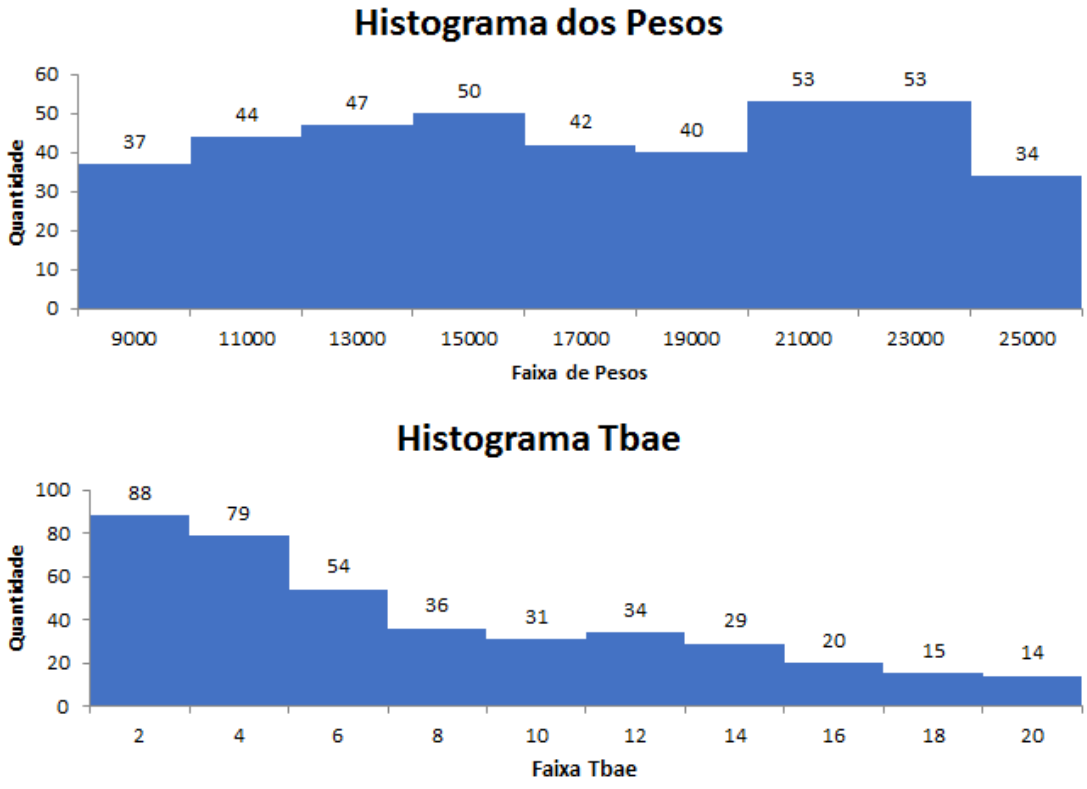

Figura 4.9: Histograma de peso e Tbae para as 400 bobinas
FONTE: (Próprio autor)

Baseado em (IBM, 2013), sugere-se algumas questões adicionais do quanto alguns fatores podem afetar no tempo de execução de um programa.

- O tamanho do problema: para uma determinada classe de problemas, geralmente o tempo de execução varia quando o número de variáveis e / ou restrições aumenta. No entanto, não somente o tamanho do problema impacta. O CPLEX resolve problemas de Programação linear Inteira (PLI) com centenas de milhares de variáveis e restrições em questão de minutos, por outro lado, existem pequenos problemas de PLI com algumas centenas de variáveis que nem o CPLEX nem outros solucionadores resolveram ainda. Resolver problemas de programação linear é geralmente mais previsível, com o tempo de execução mais fortemente correlacionado com o número de restrições do que com o número de variáveis.

- As características do modelo: um modelo com variáveis binárias e inteiras é mais difícil de resolver do que um modelo que possui apenas variáveis contínuas. A maioria dos problemas, no entanto, pode ser modelada de várias maneiras diferentes e a escolha do modelo afeta muito a capacidade de resolvê-lo.

- As características numéricas dos dados: o CPLEX pode encontrar dificuldades numéricas ao resolver alguns problemas devido à natureza dos dados do problema. Um efeito dessas dificuldades é uma desaceleração nos algoritmos de busca.

- As configurações de parâmetros do CPLEX: as configurações de parâmetro padrão funcionam bem para a maioria dos problemas. No entanto, alguns problemas podem se beneficiar de ajuste de parâmetros.

- O hardware usado: o fator mais importante no *hardware* é a velocidade do processador em cálculos de números inteiros e de ponto flutuante.

- Máquina com vários núcleos ou processadores: considera-se usar o recurso de otimização simultânea do CPLEX conforme padrão executando vários algoritmos de programação linear em diferentes threads. O CPLEX interromperá a otimização assim que um dos algoritmos terminar com uma solução ideal.

Dados os fatores que influenciam a execução do tempo do software descrito pelo próprio fabricante é perceptível que a causa do fenômeno pode não ser encontrada de forma trivial. Neste sentido, mais análises precisam ser feitas para determinar a causa deste fenômeno. Tais análises serão objeto de estudos futuros.

4.4 Formação de Carga Manual e Comprovação do Erro de Aproximação

Nesta seção busca-se avaliar os resultados gerados para a formação de cargas de três modos diferentes: 1) cargas formada manualmente, ou seja, montadas por um especialista do processo; 2) Cargas formadas usando um método exato para uma carga (PM) repetido n vezes; e 3) cargas formadas por um método exato que forma n cargas simultaneamente.

Para isso, a Tabela 4.8 é proposta e está ordenada por peso em ordem decrescente pela coluna massa, formada pelas dezesseis primeiras bobinas de mesma especificação "A" do banco de dados. Adotou-se os parâmetros $\alpha = 1$ e $\beta = 0$, peso máximo da carga de 81t e 4500mm de altura máxima, com objetivo de maximizar o peso das cargas. Ressalta-se que a soma das larguras das bobinas compõe a altura da carga. Além disso, as cargas só podem ser formadas se atenderem o número mínimo de três bobinas e o peso mínimo de 60t.

Tabela 4.8: Bobinas disponíveis para formar três cargas classificadas por peso em ordem decrescente.

Bobina	Peso(kg)	Largura(mm)	Tbae(dias)	Especificação
114	24129	1558	20	A
90	23915	1480	6	A
69	21611	1354	4	A
64	21552	1350	1	A
84	19696	1446	13	A
2	18793	1261	13	A
59	18029	1189	1	A
86	17017	1081	12	A
109	16429	1570	13	A
118	15686	906	1	A
99	13510	999	7	A
65	12836	773	5	A
48	12102	1193	1	A
49	10419	1312	8	A
47	9458	1329	19	A
19	7009	1442	2	A

4.4.1 Formação de n Cargas Manualmente

Esta seção visa demonstrar a dificuldade de solução deste problema em condições reais de tal forma que a escolha de quais bobinas irão compor a carga é feita por um

especialista, ou seja, de forma manual.

Observa-se não ser uma boa prática olhar para Tabela 4.8 e buscar uma solução direta sem qualquer estratégia. Um algoritmo guloso foi sugerido por (MARTELLO; TOTH, 1990) na seção 2.3.2 o qual recomenda-se inserir na mochila itens com o maior benefício possível, que neste exemplo trata-se do peso da bobina. Um algoritmo guloso escolhe o que perece ser a melhor solução local na esperança que este seja a melhor solução global.

O resultado obtido pelo procedimento sugerido está na Tabela 4.9. Nesta tabela estão presentes a identificação de cada bobina selecionada, assim como a explicitação da carga em que ela foi alocada. Também estão descritas as características dimensionais das bobinas e as contribuições normalizadas para a função objetivo. Ao somar as contribuições individuais obtém-se a função objetivo da solução. Adota-se os valores da massas já classificadas em ordem decrescente conforme Tabela 4.8, inicia-se o processo de inclusão das bobinas até ultrapassar a restrição de capacidade de peso ou altura máxima, retira-se então a bobina que o fez obtendo-se então a primeira carga. A segunda carga foi gerada com o mesmo procedimento anterior, porém a terceira carga não permitiu o mesmo procedimento pois os pesos das bobinas tornaram-se menores no banco de dados e antes que atingisse o peso mínimo de 60t já havia rompido a restrição de altura máxima, então optou-se por uma seleção aleatória das bobinas. Observa-se que o banco de dados residual tornou o problema intratável pois, caso deseje-se formar mais uma carga, qualquer combinação atinge a restrição de altura máxima antes mesmo que o peso mínimo de 60t seja atingido.

O procedimento manual gerou a carga um com valor aproximado de 69t e outras duas cargas com peso próximo de 60t, valor este definido como bem próximo do limite inferior para geração de uma carga. Fato é que existem mais soluções para esta aplicação e por se tratar de um exemplo prático objetiva atestar a complexidade do problema de formação de carga de forma manual. A situação real em campo tende a se formar cargas com resultados relativamente piores devido as complexidades que foram retiradas para montar este exemplo, tais como, um maior número de bobinas disponíveis, diferentes especificações em banco de dados não podendo, de forma alguma haver misturas de diferentes especificações, garantir que as bobinas mais antigas sejam incluídas além do tempo em que se pode demorar para realização da atividade.

O valor da função objetivo está na última coluna da Tabela 4.9. Não é uma prática operacional, visto que a maximização do peso final da carga gerada é o que importa necessitando que o especialista analise o peso total da carga, porém, a coluna FO deixa

mais claro como a função objetivo funciona nesta aplicação. A normalização acontece entre o valor do peso máximo e mínimo do banco de dados. Para o caso de formação de carga manual, não tem utilidade, contudo estes dados são utilizados no final da seção para comparações dos resultados.

Tabela 4.9: Formação de três cargas manuais

Bobina	**Carga**	**Peso(kg)**	**Largura(mm)**	**FO**
114	1	24129	1558	1,00000
90	1	23915	1480	0,98875
69	1	21611	1354	0,86763
Total		**69655**	**4392**	**2,85638**
64	2	21552	1350	0,76696
84	2	19696	1446	0,71949
2	2	18793	1261	0,67932
Total		**60041**	**4057**	**2,16577**
59	3	18029	1189	0,59521
86	3	17017	1081	0,55615
99	3	13510	999	0,44176
48	3	12102	1193	0,40633
Total		**60658**	**4462**	**1,99945**
Peso Total		**190354**		**7,02589**

Banco de dados residual

Bobina	**Carga**	**Peso(kg)**	**Largura(mm)**	**FO**
109		16429	1570	0,59521
118		15686	906	0,55615
65		12836	773	0,40633
49		10419	1312	0,27926
47		9458	1329	0,22874
19		7009	1442	0,10000

4.4.2 Formação de Múltiplas Cargas de Uma Vez (PMM)

No exemplo de criação de carga manual, observou-se que se faz necessário pré-tratamento dos dados para facilitar a criação da carga. O fato de se retirar algumas restrições e usar o algoritmo guloso, facilitou encontrar três possíveis soluções, sendo duas com pesos bem próximo do mínimo exigido. Já para o algoritmo do PMM para formação de três cargas, com o mesmo banco de dados da Tabela 4.8, gerou-se a Tabela 4.10. Houve expressiva melhora no peso total das cargas formadas, em relação a formação de carga em manual,

com 190t contra as 207t do algoritmo, um aumento de aproximadamente 8%. A função objetivo teve valor de 7,02 para 7,91 e tempo execução menor que 20 segundos. Também, no caso da formação de carga em manual, foi incluída uma bobina a menos em relação a formação de carga pelo algoritmo do PMM. Portanto, este problema, mesmo simplificado, demonstrou ser uma atividade complexa e com grande possibilidade de desperdício de recursos de materiais, combustíveis e mão de obra quando abordado sem o auxílio de ferramentas computacionais. No caso da inclusão das demais características, como diferentes especificações de aço e o tempo de armazenamento, a tarefa se torna ainda mais complexa e suscetível a desperdícios.

Tabela 4.10: Três cargas formadas pelo PMM

Bobina	**Carga**	**Peso(kg)**	**Largura(mm)**	**FO**
84	1	19696	1446	0,7670
90	1	23915	1480	0,9888
114	1	24129	1558	1,0000
Total		**67740**	**4484**	**2,7557**
59	2	18029	1189	0,6793
64	2	21552	1350	0,8645
99	2	13510	999	0,4418
118	2	15686	906	0,5562
Total		**68777**	**4444**	**2,5418**
2	3	18793	1261	0,7195
65	3	12836	773	0,4063
69	3	21611	1354	0,8676
86	3	17017	1081	0,6261
Total		**70257**	**4469**	**2,6196**
Total Geral		**206774**		**7,9170**

4.4.3 Formação de Cargas n Vezes (PM) e Comprovação do Erro de Aproximação

Nesta seção compara-se o resultado obtido por um algoritmo que, de forma exata, constrói n cargas individualmente, uma por vez, com o resultado obtido pelo algoritmo que forma n cargas simultaneamente conforme apresentado na Tabela 4.10.

A Tabela 4.11 contém os resultados do algoritmo que constrói as cargas individualmente. Observa-se a maximização dos pesos das cargas formadas em relação aos testes anteriores e também a geração de apenas duas cargas ao invés de três. As bobinas res-

tantes não permitem gerar nova carga atendendo todas as restrições e a necessidade da lucratividade. O resultado da função objetivo da primeira carga foi 2,8564 com três bobinas e para a carga 2 já obteve-se o valor 2,6165 com quatro bobinas."

Tabela 4.11: Duas cargas formadas pelo PM e Banco de dados residual

Bobina	**Carga**	**Peso(kg)**	**Largura(mm)**	**FO**
69	1	21611	1354	0,8677
90	1	23915	1480	0,9888
114	1	24129	1558	1,0000
Total		**69655**	**4392**	**2,8564**
2	2	18793	1261	0,7198
64	2	21552	1350	0,8646
65	2	12836	773	0,4063
86	2	17017	1081	0,6261
Total		**70198**	**4465**	**2,6165**
Total Geral		**139853**		**5,4728**

Banco de dados residual

Bobina	**Carga**	**Peso(kg)**	**Largura(mm)**	**FO**
19		7009	1442	0,1000
47		9458	1329	0,2287
48		12102	1193	0,3677
49		10419	1312	0,2792
59		18029	1189	0,6793
84		19696	1446	0,7669
99		13510	999	0,4418
109		16429	1570	0,5952
118		15686	906	0,5562

Deixa-se evidente como a função objetivo priorizou, para cada carga gerada, as bobinas com maior peso. Pode-se observar que foram incluídas na primeira carga, exatamente as bobinas mais pesadas que haviam no banco de dados, assim como foi feito na primeira simulação de carga manual. Para a segunda carga, o algoritmo gerou uma carga ainda maior, com quatro bobinas e dentro dos testes realizados é a maior carga gerada. Porém, dadas as bobinas que foram selecionadas nas duas primeiras cargas, fica impossível montar uma terceira carga que atenda a todos os requisitos. De fato, este algoritmo, ao gerar uma carga por vez, deixa de aproveitar de maneira otimizada os recursos disponíveis. Ou seja, ao invés de buscar a melhor formação para as três cargas, ele forma cargas maiores individualmente sem olhar para todo o cenário, fazendo com que seu resultado final não seja ótimo.

O erro de aproximação neste exemplo foi $z/z^* = 5,4728/7,917 = 0,6913$, que é pior que erro demonstrado por (WANG; XING, 2009). Em (WANG; XING, 2009) os autores demonstram que a razão z/z^* entre a solução do PM executado n vezes e a solução do PMM, ambos em suas versões clássicas, pode chegar a $0,75$. O resultado deste trabalho mostra que as restrições extra do problema tratado (lucratividade e peso mínimo) tornam erro de aproximação ainda maior e que, portanto, a formação de cargas simultâneas é o método mais adequado para o problema de formação de carga em Planta de Recozimento em Caixa.

Apesar de a indicação e ganhos possíveis com a aplicação do PMMCI, (KELLERER; PFERSCHY; PISINGER, 2004) e (MARTELLO; TOTH, 1990), tratam este como um problema fortemente NP-Difícil o que exige alto custo computacional para resolução, indicando que o uso de heurísticas pode ser necessário para resolver instâncias de grande porte. A seção 4.5 demonstra os resultados obtidos pelo algoritmo genético descrito na seção 3.3.

4.5 Comparação do Método Exato com Algoritmo Genético

As Tabelas desta seção contém as soluções de geração de duas até sete cargas na coluna "Nº cargas", "CplexFO" corresponde a solução ótima retornada pelo método exato com os parâmetros α e β respectivamente $0,8$ e $0,2$, seguido de seu respectivo tempo de execução em segundos "CplexT(s)". Para comparação com método heurístico tem-se "GAmedia" que contém a média de 30 execuções da função objetivo retornado pelo algoritmo genético com mesmo valores de α e β, seguido pelo seu valor máximo "GA_Max" encontrado entre as trinta execuções. O tempo de execução médio do algoritmo genético é "GA_T(s)". Para encontrar os valores referente a distância entre a solução ótima e a média da solução encontrada pelo AG utilizou-se a Equação (4.2) e finalmente a coluna "Acerto" enumerando a quantidade de vezes que o AG acertou a ótima solução do problema nesta execução. Os campos da Tabela que contém o texto "Memória" significa que o método exato não encontrou a solução do problema por ter exaurido a capacidade de memória do computador utilizado para as simulações.

$$GAP(\%) = \frac{(Solucao\ Otima\ Cplex - Media\ Solucao\ AG)}{Solucao\ Otima\ Cplex} * 100 \qquad (4.2)$$

Os resultados encontrados na Tabela 4.12 para as primeiras cinquenta bobinas do

banco de dados são considerados ótimos pela qualidade da solução gerada e tempo de execução em ambos algoritmos. Pode-se observar que o método heurístico foi aproximadamente quatro a cinco vezes mais rápido se comparado com o tempo de execução do método exato. O número de acertos que o algoritmo genético atingiu ótimo global foram expressivos para os testes com cinquenta bobinas. Ressalta-se que o tempo de execução tanto para o método exato como para o AG não atingiu tempo maior que 37 segundos para até sete cargas o que representa um ótimo tempo para resolução dos problemas.

Tabela 4.12: Tabela Resultados para 50 bobinas variando entre duas e sete cargas

Nº cargas	CplexFO	CplexT(s)	GAmedia	GA_Max	GA_T(s)	GAP(%)	Acertos
2	4,8664	8,6	4,866	4,8664	1,6	0,00	30
3	7,1987	8,8	7,199	7,1987	1,8	0,00	30
4	9,3763	10,0	9,366	9,3763	2,1	0,11	24
5	11,5515	10,5	11,478	11,5515	2,9	0,64	23
6	13,433	18,9	13,399	13,4330	3,1	0,25	17
7	15,2749	36,6	15,238	15,2749	2,8	0,24	20

A Tabela 4.13 observa-se aumento exponencial no tempo de resolução do problema ao utilizar 100 bobinas pelo método exato, porém no algoritmo genético demonstrou-se resolver em poucos segundos. O campo "GAmedia" observa-se que a média dos valores apresentam certa distância do valor ótimo global, o qual pode ser visto pelo "GAP". Para saber o quão próximo chegou-se do melhor valor em uma das trinta execuções, o campo "GA_Max" é mais evidente comparando-se com "CplexFO". Observa-se que devido ao aumento do número de possibilidades aumentando-se o número de bobinas, o número de acertos foi muito menor se comparado com a Tabela 4.12. Mesmo assim, é considerado boa a qualidade da solução gerada pelo AG.

Tabela 4.13: Tabela Resultados para 100 bobinas variando entre duas e sete cargas

Nº cargas	CplexFO	CplexT(s)	GAmedia	GA_Max	GA_T(s)	GAP(%)	Acertos
2	5,6327	9,7	5,449	5,6327	2,3	3,26	6
3	8,1836	12,6	7,959	8,1836	3,5	2,74	2
4	10,7125	78,3	10,375	10,7125	4,7	3,15	1
5	13,1424	154,9	12,761	13,1291	6,2	2,90	0
6	15,5561	640,4	14,986	15,5561	7,1	3,67	1
7	17,9528	1693,4	17,327	17,8179	9,4	3,49	0

A Tabela 4.14 é o resultado da simulação do algoritmo para um total de 200 bobinas.

Para duas cargas, o AG acertou o ótimo global por duas vezes. O GAP para duas cargas é o menor entre as simulações, ainda assim, o valor do GAP não foi maior que 4,57% para 200 bobinas e 5 cargas. Portanto, um limite é encontrado para este problema no método exato porém, o método heurístico é capaz de resolver obtendo uma solução próximo da ótima em tempo razoável. Observa-se que o método exato só seria capaz de resolver até quatro cargas para o caso do limite de tempo de 720 segundos imposto.

Tabela 4.14: Tabela Resultados para 200 bobinas variando entre duas e sete cargas.

Nº cargas	CplexFO	CplexT(s)	GAmedia	GA_Max	GA_T(s)	GAP(%)	Acertos
2	6,1667	15,2	5,990	6,1667	4,7	2,87	2
3	9,1645	33,6	8,808	9,0517	7,7	3,89	0
4	12,0495	549,9	11,545	12,0440	9,8	4,19	0
5	14,9305	10141,1	14,248	14,7502	14,0	4,57	0
6	Memória	Memória	16,871	17,6158	16,5	——	0
7	Memória	Memória	19,485	20,1737	21,5	——	0

A Tabela 4.15 contém as informações para um total de 400 bobinas. O campo Acerto foi retirado devido ao fato de que o AG não atingiu o ótimo global em nenhuma das tentativas. Observa-se que o método exato só foi capaz de entregar a solução dentro do tempo estipulado para um total de três cargas, ao passo que o AG gerou soluções para todos os problemas com tempo menor que um minuto. Ademais o GAP não ultrapassou 5,84% na média das execuções.

Tabela 4.15: Tabela Resultados para 400 bobinas variando entre duas e sete cargas.

Nº cargas	CplexFO	CplexT(s)	GAmedia	GA_Max	GA_T(s)	GAP(%)
2	6,9191	44,0	6,585	6,7580	10,5	4,83
3	10,2182	412,7	9,651	9,9695	15,9	5,55
4	13,4933	1695,7	12,726	13,1933	22,0	5,69
5	16,717	7520,9	15,741	16,2592	29,8	5,84
6	Memória	Memória	18,569	19,2407	40,3	——
7	Memória	Memória	21,683	22,4046	54,3	——

No último exemplo, apresentado na Tabela 4.16, tem a simulação para um total de 800 bobinas. O método exato gerou resultado em tempo razoável para um total de três cargas deixando cada vez mais evidente o limitador apresentado na literatura para resolver os problemas de programação linear inteira, mesmo em tempos de *hardware* de alta velocidade e considerável quantidade de memória. Já o AG apresentou tempos

menores que três minutos para todos os casos. O maior gap apresentado nesta instância é de $4,6\%$ sendo que o método exato não consegue fornecer solução para os problemas.

Tabela 4.16: Tabela Resultados para 800 bobinas variando entre duas e sete cargas.

Nº cargas	CplexFO	CplexT(s)	GAmedia	GA_Max	GA_T(s)	GAP(%)
2	7,1858	147,4	7,037	7,1059	26,6	2,08
3	10,6648	1845,7	10,174	10,6648	43,1	4,60
4	Memória	Memória	13,382	13,8835	59,5	——
5	Memória	Memória	16,332	16,9172	82,1	——
6	Memória	Memória	19,453	20,1112	114,6	——
7	Memória	Memória	22,581	23,3979	141,7	——

Capítulo 5

Conclusões e Trabalhos Futuros

5.1 Conclusões

Neste trabalho propõe-se soluções matemática e heurística para otimizar o processo siderúrgico de formação de cargas em planta de recozimento em caixa com objetivo de maximizar o peso em tonelagem da carga utilizando um parâmetro α e também possibilitar priorizar as bobinas mais antigas aguardando no estoque através de um parâmetro β, respeitando as restrições física impostas pelo processo como limites de altura e peso além de garantir apenas o agrupamento de bobinas de mesma especificação na carga.

Foi feita a revisão da literatura sobre o tema com o intuito de buscar problemas similares e possíveis abordagens. A revisão da bibliográfica apontou que este problema tem similaridades com o problema da mochila e as técnicas usadas para resolução são modelagem matemática e heurísticas.

Duas propostas de resolução do problema foram apresentadas, uma formulação matemática baseada no Problema das Múltiplas Mochilas, descrita em detalhes na seção 3.2 e uma meta-heurística baseada em Algoritmo Genético, descrita em detalhes na seção 3.3.

Para encontrar a solução ótima dos problemas através da implementação da formulação matemática foi utilizado o método exato. Portanto, algumas conclusões para este método é demonstrado:

- Conclui-se que o modelo matemático implementado é capaz de solucionar os problemas reais, em um tempo computacional aceitável, para um número limitado de bobinas e cargas.
- Com a variação dos parâmetros α e β conclui-se que o modelo matemático proposto

atende às necessidades práticas, de forma que, um especialista do processo consiga decidir, em função da condição operacional, se prioriza cargas com pesos maiores ou a necessidade de incluir as bobinas mais antigas, sem perder o foco da produtividade.

- O método exato demonstrou limitações no tempo computacional para encontrar soluções em função do tamanho das instâncias (número de bobinas e cargas). Para superar este problema, foi proposto o algoritmo genético que comprovou encontrar bons resultados nas maiores instâncias ficando até 6% do ótimo global, além disso, em todas as simulações que o método exato não encontrou solução ao atingir o limite de memória disponível o algoritmo genético gerou soluções com tempo de resolução de poucos segundos.

Também foi possível corroborar a afirmação de (WANG; XING, 2009), mostrando que, executar o algoritmo exato N vezes para resolução do problema da mochila única, introduz perda na função objetivo quando comparado ao algoritmo exato que resolve o problema das múltiplas mochilas de uma só vez.

Portanto pode-se concluir que as propostas do trabalho possibilitam gerar soluções eficientes para o processo de formação de carga em Planta de Recozimento em Caixa, mostrando que é possível obter um melhor aproveitamento dos recursos dentro de plantas industriais do ramo.

5.2 Trabalhos Futuros

São listadas as seguintes intenções para trabalhos futuros.

- Devido a eficiente codificação utilizada foi possível encontrar soluções com baixo tempo de execução utilizando algoritmos genético, porém propõe-se aprimorar a heurística exploratória dos processos de mutações guiadas. Segundo (LINDEN, 2012) o operador de mutação é quem garante a continuidade da existência de diversidade genética na população, com isso, pode ser possível encontrar as soluções ótimas para os casos de instâncias com dimensões maiores.

- Outro possível trabalho futuro é o uso de algoritmo baseado em programação dinâmica em busca de superar o problema do alto custo computacional apresentado pelo método exato e encontrar melhores soluções que o algoritmo genético.

Referências

ABBASS, H.; SARKER, R.; NEWTON, C. *Data Mining: A heuristic approach*. Londres: IGP - Idea Group Publishing, 2002. v. 27. 310 p. ISBN 1930708254.

ALKHEDHAIRI, A. Simulated annealing metaheuristic for solving p-median problem. *Int. J. Contemp. Math. Sciences*, v. 3, p. 1357–1365, 01 2008.

ANDRADE, R.; BIRGIN, E. G.; MORABITO, R. Two-stage two-dimensional guillotine cutting stock problems with usable leftover. *Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo*, p. 1–21, 2013.

AÇO BRASIL. *Processo Siderúrgico*. 2019. Disponível em: <www.acobrasil.org.br>. Acesso em: 27 jul. 2019.

BALLOU, R. H. *Gerenciamento da cadeia de suprimentos/logística empresarial*. Porto Alegre: Bookman, 2006. v. 5. 616 p. ISBN 978-85-363-0591-2.

BIGERELLE, M.; HAOUAM, A. Industrial validation of thermal model applied to steel coils annealed under hydrogen gas. In: . Croatia: IEEE, 2017. p. 3–6.

BORGES, F. H.; DALCOL, P. R. T. Indústrias de processo: Comparações e caracterizações. In: *XXII Encontro Nacional de Engenharia de Produção - ABEPRO*. Curitiba: ENEGEP, 2002. p. 1–9.

BOUARARA, H.; HAMOU, R.; RAHMANI, A. *Advanced metaheuristic methods in big data retrieval and analytics*. Hershey PA, USA: IGI Global, 2018. ISBN 9781522573395.

CASADO, S. et al. Grouping products for the optimization of production processes: A case in the steel manufacturing industry. *European Journal of Operational Research*, v. 286, n. 1, p. 190 – 202, 2020. ISSN 0377-2217.

CASH, A. *Psicologia para leigos: Tradução da 2a Edição*. Rio de Janeiro: Alta Books, 2019. ISBN 9788550812014.

DANTZIG, G. B. Discrete-variable extremum problems. *Operations Research*, v. 5, n. 2, p. 266–288, 1957.

DEUS, V. S. de et al. Desenvolvimento de novo controle e gerenciamento para o processo dos fornos de recozimento em caixa número 1. *Tecnologia em Metalurgia e Materiais*, v. 3, n. 3, p. 46–51, 01 2007.

DEUTZ A.H., E. M. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. Springer, p. 585–609, 09 2018.

EIBEN, A.; SMITH, J. *Introduction to evolutionary computing*. London: Springer, 2015.

FOGEL, D. B. An introduction to simulated evolutionary optimization. IEEE transactions on neural networks, v. 5, n. 1, p. 12, 1994.

FOLLY, J. R. *Otimização do Problema de Alocação de Aulas no Âmbito do Ensino Superior*. Dissertação (Mestrado) — Universidade Federal Fluminense, 2017.

FUKUNAGA, A. S. A new grouping genetic algorithm for the multiple knapsack problem. In: *Congress on Evolutionary Computation Intelligence*. China: IEEE, 2008. p. 2225–2232.

GIL, A. C. *Como Elaborar Projetos de Pesquisa*. 4. ed. São Paulo: Editora Atlas, 2002.

GUO, Q.; TANG, L. Modelling and discrete differential evolution algorithm for order rescheduling problem in steel industry. *Computers and Industrial Engineering*, v. 130, p. 586 – 596, 2019. ISSN 0360-8352.

HADIDI, L. A.; MOAWAD, O. A. The product-mix problem for multiple production lines in sequenced stages: a case study in the steel industry. In: *The International Journal of Advanced Manufacturing Technology*. China: IEEE, 2017. p. 1–10.

HASAN, J.; KAABI, J.; HARRATH, Y. Multi-objective 3d bin-packing problem. In: *8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*. Bahrain: IEEE, 2019. p. 1–5.

HASEEN, S. et al. Integer programming: Naz cut and a-t cut. *International Journal of Engineering Science and Technology*, v. 06, p. 128–137, 04 2014.

HERRERA, F.; LOZANO, M.; VERDEGAY, J. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, v. 12, p. 265–319, 08 1998.

HILLIER, F.; LIEBERMAN, G. *Introduction to Operations Research*. USA: McGraw-Hill Higher Education, 2010. 1047 p. ISBN 9780071267670.

HOLLAND, J. H. *Adaptation in natural and artificial systems*: An introductory analysis with applications to biology, control, and artificial intelligence. Michigan: A Bradford Book, 1992.

HUANG, B. et al. Approximation algorithms for the generalized multiple knapsack problems with k restricted elements. In: *7th International Conference on Intelligent Human-Machine Systems and Cybernetics*. China: IEEE, 2015. v. 1, p. 470–474.

IBM. *ILOG CPLEX V.12.6 User Manual IBM Corp*. 2013.

IYODA, E. M. *Inteligência computacional no projeto automático de redes neurais híbridas e redes neurofuzzy heterogêneas*. Dissertação (Mestrado) — Unicamp, 2000.

KAGAN, N. et al. *Métodos de otimização aplicados a sistemas elétricos de potência*. São Paulo: Blucher, 2009. ISBN 9788521215165.

KARGUPTA, H.; GOLDBERG, D. Blackbox optimization: Implications of search. Los Alamos, p. 3–16, 10 1997.

KELLERER, H.; PFERSCHY, U.; PISINGER, D. *Knapsack problems*. New York - USA: Springer, 2004.

KOBLASA, F.; VAVROUSEK, M.; MANLIG, F. Three-dimensional bin packing problem with heterogeneous batch constraints. In: *Mathematical Methods in Economics*. Hradec Králové: ICMME, 2016. p. 406–412.

LAGOUDAKIS, M. G. The 0 – 1 knapsack problem: an introductory survey. University of Southwestern, Louisiana, p. 1–13, 1996.

LAM, G. T. et al. Considerations on using genetic algorithms for the 2d bin packing problem: A general model and detected difficulties. In: *21st International Conference on System Theory, Control and Computing (ICSTCC)*. Alemanha: IEEE, 2017. p. 303–308.

LEE, K. Y.; EL-SHARKAWI, M. A. *Applications to power system scheduling*. Canada: IEEE, 2008. ISBN 978-0471-45711-4.

LINDEN, R. *Algoritmos genéticos*. Rio de Janeiro: Ciência Moderna, 3ª Edição, 2012. ISBN 8539901951.

LONG, J. et al. Variable neighborhood search for integrated determination of charge batching and casting start time in steel plants. *Journal of intelligent and fuzzy systems*, v. 17, p. 3821–3832, 01 2018.

LOPES, H. S.; RODRIGUES, L. C. A.; STEINER, M. T. A. *Meta-Heurística em pesquisa operacional*. Curitiba: Omnipax, 2003.

MARTELLO, S.; TOTH, P. *Knapsack Problems*: Algorithms and computer implementations. New York: John Wiley Sons, 1990.

MOR, S. D. K. *Emprego da técnica de workstealing*: Estudo de caso com o problema da mochila e mpi. Monografia — UFRGS, 2007.

NAYAK, S.; MISRA, B.; BEHERA, D. H. Evaluation of normalization methods on neuro-genetic models for stock index forecasting. In: . India: WICT, 2012. p. 602–607. ISBN 978-1-4673-4806-5.

NETO, L. L. R. *Um Algorítmo Genético para solução do problema de carregamento de container*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, 2005.

PALMETAL. *Bobinas em aço*. 2019. Disponível em: <https://blog.palmetal.com.br/da-serie-materias-primas-chapas-e-bobinas-de-aco-inox/>. Acesso em: 01 agosto 2019.

PEREZ, P. A.; GARCIA, A. P.; RAMIREZ, V. A. Bin-packing using genetic algorithms. In: *15th International Conference on Electronics, Communications and Computers (CONIELECOMP)*. Salamanca, Mexico: IEEE, 2005. p. 311–314.

QUEIROZ, T. A. de; MIYAZAWA, F. K. Problema da mochila 0-1 bidimensional com restrições de disjunção. In: . Rio de Janeiro: SBPO, 2012. p. 1–12.

RESENDE, M. G. C. Biased random-key genetic algorithms: A tutorial. In: *SBPO*. Rio de Janeiro: SBPO, 2012. p. 1–521.

REZOUG, A.; BADER-EL-DEN, M.; BOUGHACI, D. Guided genetic algorithm for the multidimensional knapsack problem. *Memetic Computing*, v. 10, p. 1–21, 05 2017.

RODRIGUES, R. D. F. *Complexidade Computacional*. Dissertação (Mestrado) — Universidade Federal Amazonia, 2000.

SACHDEVA, C.; GOEL, S. An improved approach for solving 0/1 knapsack problem in polynomial time using genetic algorithms. In: *International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*. India: IEEE, 2014. p. 1–4.

SADYKOV, R.; VANDERBECK, F. Bin packing with conflicts: A generic branch-and-price algorithm. *Informs Journal on Computing*, v. 25, p. 1–26, 01 2012.

SALEH, S. A. *O método simplex e o método gráfico na resolução de problemas de otimização*. Dissertação (Mestrado) — l'Ecole Doctorale en Sciences Technologie et Santé, France, 2015.

SANTOS, R. M. dos. *Influência das variáveis de processo de recozimento em caixa e laminação de encruamento nas perdas magnéticas de aços ao carbono laminados a frio*. 121 p. Dissertação (Mestrado), 2017.

SCHEID, A. *A Elaboração do Aço*. 2020. 82 p.

SCHEUERMANN, W. New developments in hydrogen annealing of steel coils. *MPT Metallurgical Plant and Technology*, v. 1, p. 1–7, 1995.

SEWARD, L.; DOANE, D. *Estatística aplicada à administração e economia - 4.ed.* São Paulo: Mc Graw Hill, 2014. ISBN 9788580553949.

SHAMAKHAI, H. A. *The 0 -1 multiple knapsack problem*. Dissertação (Mestrado) — The Faculty of Graduate Studies Laurentian University, Canada, 2017.

SILVA, E. L. da; MENEZES, E. M. *Metodologia de pesquisa e elaboração de teses e dissertações*. 4. ed. Santa Catarina: UFSC, 2005. 138 p.

SOUZA Éfren Lopes de; RAFAEL, E. A. L. *Abordagens para resolver o problema da mochila 0-1*. Amazonas: UFA, 2009. 8 p.

STARCK, A. von; MÜHLBAUER, A.; KRAMER, C. *Handbook of Thermoprocessing Technologies: Fundamentals, Processes, Components, Safety*. Alemanha: Vulkan-Verlag, 2005. ISBN 3-8027-2933-1.

STEEL, W. *Crude Steel Production*. 2019. Disponível em: <https://www.worldsteel.org/internet-2017/steel-by-topic/statistics/steel-data-viewer/MCSP_crude_steel_monthly/CHN/IND>.

SU. Integrated batch planning optimization based on fuzzy genetic and constraint satisfaction for steel production. In: *Int. J. Simul. Model.* China: Semantic Scholar, 2016.

TALBI, E. G. *Metaheuristics*: from design to implementation. EUA: John Wiley and Sons, 2009. 618 p. ISBN 978-0-470-27858-1.

TENOVA. *Planta Típica de um RCX*. 2020. Disponível em: <https://www.tenova.com/product/hph%C2%AE-bell-type-annealing-furnaces-for-stainless-steel-strip/>. Acesso em: 14 Abril. 2020.

THOMAS, B.; KHURI, S.; HEITKOTTER, J. The zero/one multiple knapsack problem and genetic algorithms. ACM, p. 1–8, outubro 1994.

TOMAR, D. An improved greedy heuristic for unweighted minimum vertex cover. In: *Sixth International Conference on Computational Intelligence and Communication Networks*. India: IEEE, 2014. p. 618–622.

TUCKER, A.; CRAMPTON, J.; SWIFT, S. An efficient representation and crossover for grouping genetic algorithms. *Evolutionary Computation*, MIT Press Journals, v. 13, n. 4, p. 477–500, 2005.

WANG, Z.; XING, W. A successive approximation algorithm for the multiple knapsack problem. *J. Comb. Optim.*, v. 17, p. 347–366, 05 2009.

WEISSTEIN, E. W. *Magic Square*. 2020. Disponível em: <https://mathworld.wolfram.com/MagicSquare.html>.

WILBAUT, C.; HANAFI, S.; SALHI, S. A survey of effective heuristics and their application to a variety of knapsack problems. *Ima Journal of Management Mathematics*, v. 19, p. 227–244, 03 2007.

XING, X. et al. A novel genetic algorithm based on individual and gene diversity maintaining and its simulation. In: *International Conference on Automation and Logistics*. China: IEEE, 2007. p. 2754–2758.

YAKAWA, T.; IIMA, H. A new design of genetic algorithm for bin packing. In: *Congress on Evolutionary Computation*. Japão: IEEE, 2003. v. 2, p. 1044–1049.

YAMADA, T.; SEIJI, K.; KOHTARO, W. Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *IPSJ*, v. 43, n. 9, p. 2864–2870, 09 2002.