CURSO DE SISTEMAS DE INFORMAÇÃO

MODELAGEM MONO E MULTIOBJETIVO PARA O PROBLEMA DE SEQUENCIAMENTO DE TAREFAS EM MÁQUINAS PARALELAS NÃO RELACIONADAS

HUGO CAVALCANTE LIMA

Palmas

2018

CURSO DE SISTEMAS DE INFORMAÇÃO

MODELAGEM MONO E MULTIOBJETIVO PARA O PROBLEMA DE SEQUENCIAMENTO DE TAREFAS EM MÁQUINAS PARALELAS NÃO RELACIONADAS

HUGO CAVALCANTE LIMA

Projeto apresentado ao Curso de Sistemas de Informação da Fundação Universidade do Tocantins - UNITINS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do professor Me. Douglas Chagas.

Palmas

2018

UNITINS | GOVERNO DO TOCANTINS

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE SISTEMAS DE INFORMAÇÃO DA UNIVERSIDADE ESTADUAL DO TOCANTINS - UNITINS

Aos **26** dias do mês de **Junho** de **2018**, reuniu-se na Universidade Estadual do Tocantins, às **08:30 horas**, sob a Coordenação do Professor **Douglas Chagas da Silva**, a banca examinadora de Trabalho de Conclusão de Curso em Sistemas de Informação, composta pelos examinadores Professor **Douglas Chagas da Silva** (Orientador), Professor **Marco Antônio Firmino de Sousa** e Professora **Arlenes Delabary Spada**, para avaliação da defesa do trabalho intitulado **"MODELAGEM MONO E MULTIBOJETIVO PARA O PROBLEMA DE SEQUENCIAMENTO DE TAREFAS EM MÁQUINAS PARALELAS NÃO RELACIONADAS"** do acadêmico **Hugo Cavalcante Lima** como requisito para aprovação na disciplina Trabalho de Conclusão de Curso (TCC). Após exposição do trabalho realizado pelo acadêmica e arguição pelos Examinadores da banca, em conformidade com o disposto no Regulamento de Trabalho de Conclusão de Curso em Sistemas de Informação, a banca atribuiu a pontuação: 9,0 .

Sendo, portanto, o Acadêmico: (X) Aprovado ( ) Reprovado

Assinam esta Ata:

Professor Orientador: _____

Examinador: Arlenes Delabary

Examinador: Marco Antonio Firmino de Sousa

Acadêmico: Hugo Cavalcante Lima

Prof. Douglas Chagas da Silva
Coordenador da Banca Examinadora
Coordenação do Curso de Sistemas de Informação

Dados Internacionais da Catalogação na Publicação (CIP)
Biblioteca da Universidade Estadual do Tocantins
Campus Palmas – TO

L732m Lima, Hugo Cavalcante
Modelagem mono e multiobjetivo para o problema de sequenciamento de tarefas em máquinas paralelas não relacionadas / Hugo Cavalcante Lima – Palmas, TO, 2018.
53 fls.; Il.;col.

Inclui CD-ROM

Orientação: Prof. Douglas Chagas da Silva
TCC (Trabalho de Conclusão de Curso). Bacharel em Sistemas de informação. Universidade Estadual do Tocantins. 2018

1. Métodos mono e multiobjetivos. 2. Técnicas de otimização. 3. Métodos determinísticos. 4. Modelagem matemática. I. Silva, Douglas da. II. Título. III. Direito.

CDD: 003.3

Ficha catalográfica elaborada pela Bibliotecária – Maria Madalena Camargo – CRB 2/1527

Este trabalho é dedicado a minha esposa Romilda Almeida.

Agradecimentos

Agradeço a Deus, minha família, amigos e colegas da faculdade por me proporcionar a oportunidade de criar este trabalho. Retribuo de maneira especial a minha esposa pela sua paciência, carinho, amor e dedicação com minha pessoa. Além de reconhecer o apoio e dedicação dos Professores e Colegas da Unitins especificamente aos companheiros de Turma e ao Prof. Douglas Chagas, fundamentais nessa etapa final de minha jornada.

"E a atitude de fé é o oposto do apego à crença."
Alan Watts

Resumo

Diante dos problemas de otimização das linhas da produção os métodos computacionais têm suma importância auxiliando na construção de modelos otimizados para os processos de manufatura. Contudo estes problemas possuem um ou mais objetivos, os quais às vezes podem ser conflitantes entre si. Dessa maneira a resolução dos mesmos necessita do emprego de métodos de otimização classificados conforme os critérios de eficiência aplicados estes conhecidos por função objetivo. Dentre as Técnicas de otimização conhecidas a Programação Linear e Binária permitiu a construção de soluções algorítmicas que proporcionam uma gama de resultados ótimos com consumo de recurso computacional viável.

Palavras-chaves: Métodos mono e multiobjetivos, Técnicas de otimização, Métodos determinísticos, Modelagem Matemática.

Abstract

Given the problems of optimization of production lines, computational methods are extremely important, helping to build models optimized for manufacturing processes. However, these problems have one or more objectives, which can sometimes be conflicting. In this way the resolution of the same requires the use of optimization methods classified according to the applied efficiency criteria known as objective function. Among the techniques of optimization known Linear and Binary Programming allowed the construction of algorithmic solutions that provide a range of optimal results with viable computational resource consumption.

Key-words:Mono and multiobjective methods, Optimization techniques, Methods deterministic, Mathematical Modeling

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

POM - Problema de Otimização Multiobjetivo.

PMNR - Máquinas paralelas não relacionadas

PI - Programação Inteira

PIL - Programação Linear Inteira

PIB - Programação Linear Binária

Sumário

1 Introdução

O mercado empresarial ao perceber que precisa melhorar seu processo produtivo, de modo a conseguir a eficiência e produtividade vem desenvolvendo novas técnicas de otimização de seus modos de produção (COUTINHO, 2008).

Ao considerarmos, por exemplo, o caso de uma empresa que deseja comprar uma máquina nova para melhorar seu processo de produção, mas, contudo, que seja mais barata, estamos diante da chamada *otimização mono-objetivo*, pois, o objetivo é adquirir um bem de menor custo. No entanto, caso a empresa pretenda adquirir, uma máquina mais barata e que consuma menos energia para desempenhar suas atividades, temos dois objetivos que devem ser observados, custo e economia energética, para estas categorias de problemas existe a *otimização multiobjetivo* (CORREIA, 2017).

As empresas têm entre suas metas maximizarem o uso dos seus equipamentos e ativos funcionais, visando lucratividade e diminuindo a ociosidades dos maquinários. Assim, o estudo de modelos de otimização são de fundamental importância. Organizar o sequenciamento de tarefas em diversas máquinas é um problema constante que consiste em definir um grupo de atividades, as quais poderão ser alocadas em dispositivos paralelos, com objetivo de minimização dos tempos de processamentos nas linhas de produção (SILVA, 2008).

Dessa maneira para otimizar os objetivos conflitantes do processo de produção de máquinas faz se necessários estudos dos conceitos básicos de otimização e seus métodos e técnicas. Portanto, vislumbrou-se a necessidade de modelar matematicamente o problema, permitindo em sequência o emprego dos conceitos de programação linear na construção das soluções para cada problema mono e multiobjetivos, com a finalidade de resolver as proposições, além da análise dos resultados. Tais aplicações se restringiram as funções objetivo.

Este trabalho está organizado da seguinte forma: no Capítulo 2 é apresentado uma breve explanação sobre os conceitos básicos de otimização e alguns de seus métodos em conjunto com conceitos básicos de programação de tarefas em máquinas paralelas. No capítulo 3 são apresentados materiais utilizados e pseudocódigos dos métodos desenvolvidos e as características de cada algoritimo. No capítulo 4 os resultados são abordados, e por fim, as considerações finais e trabalhos futuros no Capítulo 5.

1.1 Motivação

Diante da grande concorrência entre as empresas, métodos computacionais de otimização ganham cada vez mais importância. Problemas de máquinas paralelas não relacionadas, são presentes em diversas indústrias. No ramo da pesquisa, por se tratar de um problema da classe NP-difícil, o que impossibilita a obtenção apenas de uma solução ótima para a questão, sendo apresentado, na verdade, para esta problemática, um conjunto de soluções viáveis. O desenvolvimento de uma metodologia eficaz em conjunto com resultados com tempos viáveis, reforça os diversos estudos, nas áreas da Pesquisa Operacional e Ciência da Computação (QUEIROZ, 2011; SECCHI, 2015; XAVIER, 2003).

Dessa forma a principal motivação consiste em apresentar a modelagem matemática para o problema e a aplicação dos conceitos de otimização, possibilitando uma melhor utilização dos recursos no processo de fabricação, reduzindo custos e obtendo ganhos, melhorando o procedimento na sua totalidade.

1.2 Objetivos

Nesta seção são demonstrados o objetivo geral e específicos dessa monografia.

1.2.1 Objetivo geral

Elaborar uma proposta envolvendo a formulação matemática e desenvolver algoritmos mono-objetivos e multiobjetivos para o problema de sequenciamento de máquinas paralelas não relacionadas.

1.2.2 Objetivos específicos

- Compreender os conceitos envolvendo problemas de sequenciamento de máquinas;
- Propor a formulação matemática para o problema abordado;
- Implementar algoritmos mono-objetivos e multiobjetivos para a resolução do problema;
- Demonstrar os resultados obtidos para a problemática.

2 Referencial Teórico

Neste Capítulo, a seção 2.1 apresenta os conceitos básicos de otimização, já na seção 2.1 tem-se os conceitos de otimização, posteriormente a seção 2.3 demonstra-se alguns conceitos envolvendo a programação de máquinas paralelas e suas generalizações.

2.1 Conceitos básicos de otimização

Para NARIÑO, MARTHA e DE MENEZES (2014), otimização é o processo de busca da melhor solução para o aproveitamento dos recursos de um grupo de possíveis soluções com base nas variáveis do projeto.

As técnicas de otimização, pretendem encontrar a melhor solução para problemas de maximização ou minimização, tais aplicações são necessárias nas áreas da Pesquisa operacional, Engenharia e Controle de processos. Proporcionando, por exemplo, soluções para problemas de controle de estoques, planejamento do processo de produção, dimensionamento e otimização de processos além da identificação de sistemas (SECCHI, 2015).

Desse modo, é apresentado um exemplo comum de formulação matemática utilizada em problemas de otimização a seguir:

$$arg\ min/max\ f_{obj}(x) \tag{2.1}$$

de modo que:

$$g_i(x) \leq 0\ para\ i = 1, ..., n_g \tag{2.2}$$

$$h_j(x) \leq 0\ para\ i = 1, ..., n_h \tag{2.3}$$

$$lb_k \leq x_k \leq ub_k\ para\ i = 1, ..., n_x \tag{2.4}$$

De forma que: f_{obj} é a função objetivo a ser otimizada; $g_i(x)$ conjunto das restrições; n_g é número de restrições de desigualdade; h_j conjunto de restrição de igualdade; n_h é o número de restrições de igualdade; x_k é o k-ésimo elemento do vetor de variáveis de projeto; n_x é o número de variáveis de projeto; e lb_k e ub_k são os limites inferior e superior da variável de projeto x_k.

Nas subseções 2.1.1, 2.1.2 e 2.1.3 apresenta-se os componentes principais dos problemas de otimização.

2.1.1 Função objetivo

A função objetivo tem-se a Equação 2.1 o qual é a representação do critério de eficiência aplicado no problema de otimização. A atuação das variáveis de controle do problema, pode determinar a utilização da função objetivo, em problemas com uma única função, ou seja, questões mono-objetivos, ou por várias funções na solução de problemas multiobjetivos. Na formulação de um projeto o Engenheiro ou Projetista, pode pretender, por exemplo, minimizar apenas os esforços na produção, o problema e mono-objetivo, outrora caso a minimização de recursos seja os esforços e os custos, a problemática passa a ser multiobjetivo (NARIÑO; MARTHA; DE MENEZES, 2014).

2.1.2 Variáveis de decisão e Restrições

Na Equação 2.4 tem-se uma demonstração de variáveis, os quais são os parâmetros do método de otimização que definem o atributos do modelo investigado. As variáveis são o objetivo o qual se deseja solucionar dentro do contexto. Dessa forma, as variáveis podem assumir valores distintos, contudo, podem resultar em soluções inviáveis para o procedimento de otimização.

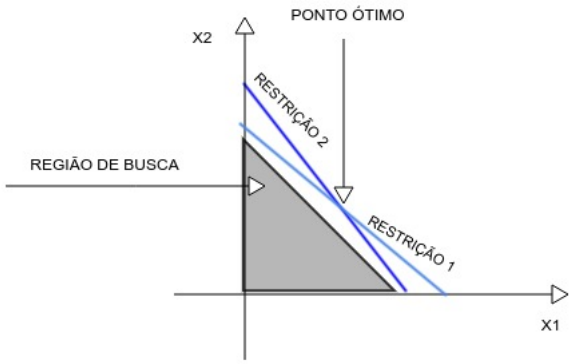
Por outro lado, o conjunto de restrições conforme as Equações 2.2 e 2.3 são as fronteiras estabelecidas ao sistema, estas definidas por condições como o trabalho, os recursos e custos, premissas de otimização submetidas as variáveis de decisão. As restrições são classificadas em equações de igualdade ou inequações de desigualdade (SECCHI, 2015).

2.1.3 Região de busca e Ponto ótimo

A região do domínio conforme Figura 1 contempla às restrições do problema de otimização é chamada de espaço de busca. As extremidades são definidas pelas restrições do sistema e pelo distanciamento da intercorrência das variáveis da proposta a ser otimizada. Contudo, as soluções ótimas, podem estar fora do espaço de busca, tornando as mesmas impraticáveis por estarem infringindo as restrições (NARIÑO; MARTHA; DE MENEZES, 2014).

Observa-se na Figura 1 que o ponto ótimo, que é de maneira prática o vetor das variáveis da proposta que otimiza (minimiza ou maximiza) a função objetivo e atendente as restrições do problema. A função objetivo correlacionada as variáveis são chamadas de valor ótimo. De maneira que, o par constituído pelo ponto ótimo e o valor ótimo é designado solução ótima.

Figura 1 – Representação gráfica - Região de busca e Ponto ótimo



Adaptado de (AZUMA, 2011)

2.1.4 Otimização multiobjetivo

Problemas de otimização multiobjetivo consistem em dois ou mais objetivos, os quais podem ser discrepantes, mais que necessitam serem otimizados concomitantemente. A solução de um problema multiobjetivo é composta por um conjunto de soluções comprometidas com os objetivos. Tal agrupamento de soluções é qualificado como Pareto-ótimo, ou seja, quando cada resolução do conjunto não possuir nenhuma outra solução exequível com capacidade de reduzir o valor de um dos critérios da problemática, sem que não cause nenhum incremento em algum critério (AZUMA, 2011).

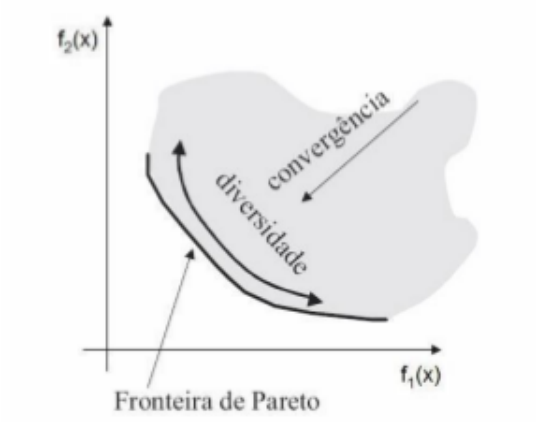
Conforme NARIÑO, MARTHA e DE MENEZES (2014) podem existir casos de otimização multiobjetivo em que se procura obter uma melhor resolução em mais de uma particularidade do sistema e, ao mesmo tempo, atender concomitantemente a vários objetivos. Desta forma, busca-se a melhor resposta em um determinado objetivo apenas, pode resultar uma resposta não esperada em relação aos outros.

Conforme AZUMA (2011), AMORIM (2006) estas são três principais metas da otimização multiobjetivo:

1. Alcançar um conjunto de soluções que esteja o mais próximo da fronteira de Pareto;
2. Atingir um conjunto de soluções com a maior diversidade de valores possíveis;
3. Conseguir atingir as outras duas metas anteriores com o menor gasto de processamento computacional possível.

Ao analisar o desempenho de um algoritmo, é fundamental utilizar medidas que contemplem ambas as metas demostradas na Figura 2. A primeira meta é genérica a

Figura 2 – Metas da otimização multiobjetivo



Fonte: (AZUMA, 2011)

qualquer método de otimização. No entanto, a segunda é específica para a otimização multiobjetivo que trabalha com o espaço de decisão e o espaço dos objetivos. Dessa forma para encontrar soluções que estejam devidamente distribuídas no plano em ambos os espaços, e necessário utilizar uma considerável quantidade de recursos computacionais. Em contrapartida, a necessidade de se criar soluções algorítmicas eficientes se torna evidente a frente dos crescentes e complexos problemas de otimização (AZUMA, 2011).

2.2 Métodos de otimização

Existem diversos métodos de otimização empregados na solução de inúmeros problemas. Os mesmos podem ser classificados conforme a função objetivo e suas restrições que podem ser lineares ou não-lineares. Dessa forma, tem-se os dois grupos de métodos: programação linear e programação não-linear.

Assim, em MINEIRO (2007) os métodos podem apresentar dificuldades no momento da solução, como questões de multimodalidade (vários pontos ótimos), encadeamento das funções a serem otimizadas e suas restrições, são alguns dos problemas apresentados.

Escolher a técnica mais apropriada para cada contexto, considerando as variáveis e o problema em questão, pode viabilizar ou inviabilizar uma solução complexa mediante os altos custos computacionais envolvidos.

2.2.1 Programação linear

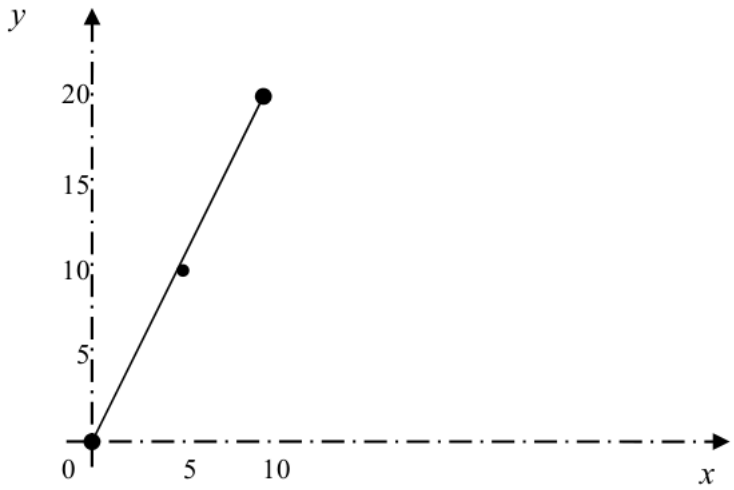
Os problemas de programação linear desde 1950 tem diversos nichos de utilização. De maneira objetiva esse modelo trabalha com soluções para problemas de alocação de recursos

concorrentes, solucionando problemas, por exemplo, na administração agrícola, padrões de embarque, alocação de recursos institucionais e mão-de-obra (HILLIER; LIEBERMAN, 2006).

Conforme MINEIRO (2007) os problemas de programação linear são formados por funções objetivos e conjunto de restrições lineares, ou seja, possuem relações equivalentes.

Na Figura 3 tem-se a representação gráfica da função linear obtida por $y = 2x$, onde x pode admitir valores entre 0 e 10. Dessa forma todos os pontos admitidos pela preposição linear podem ser ligados por uma linha reta.

Figura 3 – Gráfico da função $y = 2x$



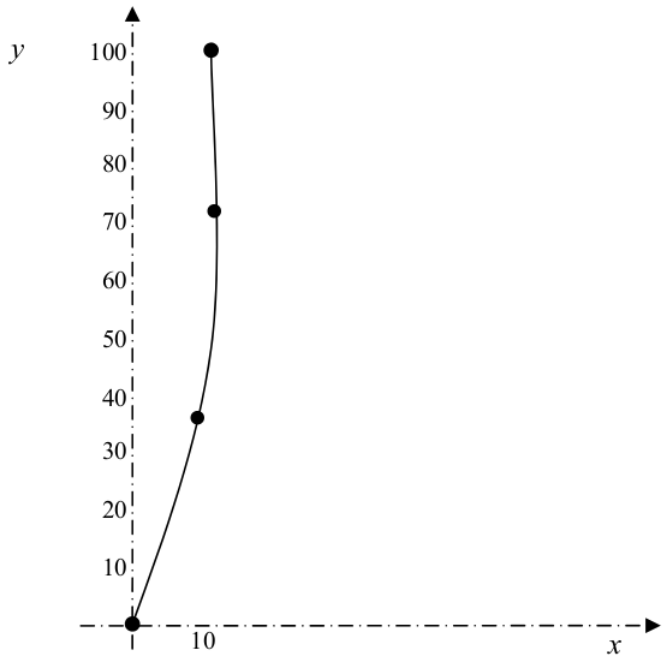
Obtido em: (MINEIRO, 2007)

De acordo com MARTINS (2017), a principal tarefa da programação linear compreende a maximização ou minimização (otimização) de uma função linear respeitando as restrições do problema. Tendo por objetivo encontrar a solução ótima, ou seja, o melhor resultado ou conjunto de resultados dentro da região de busca conforme discutido na seção 2.1.3.

2.2.2 Programação não linear

A programação não-linear atua em problemas que envolvem funções objetivo e restrições com variáveis irregulares, ou seja, não lineares. De acordo com MINEIRO (2007), na programação não-linear a função é classificada como não-linear quando todas as variáveis envolvidas na problemática são desproporcionais. Dessa maneira, os pontos no gráfico não poderão ser relacionados com uma reta. Conforme pode-se observar no exemplo da Figura 4, onde tem-se uma função não-linear $y = x^2$, onde x pode contrair valores entre 0 e 10. Por fim, tem-se uma curva na interligação dos pontos na função representada.

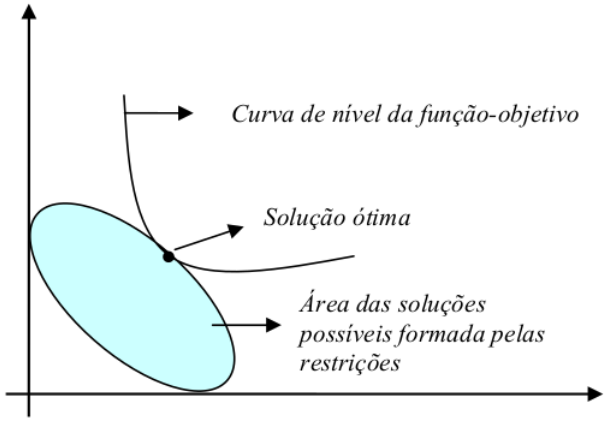
Figura 4 – Gráfico da função $y = 2^x$



Obtido em: (MINEIRO, 2007)

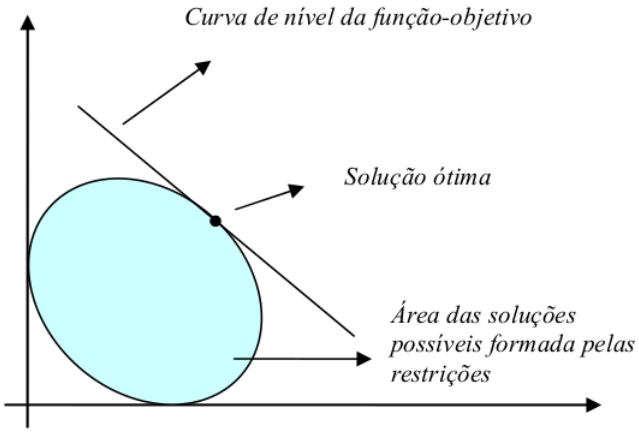
Dessa maneira, na programação linear a solução pode ser encontrada nos vértices do problema, conforme ilustrado na Figura 3, ou seja, na reta da função objetivo e das restrições do problema. Contudo, nos problemas não-lineares conforme os exemplos ilustrados nas Figuras 5 e 6 tem-se soluções ótimas com diferentes formas.

Figura 5 – Problema com função-objetivo e restrições não lineares



Obtido em: (CORRAR; THEÓPHILO; BERGMANN, 2004)

Figura 6 – Problema com função-objetivo linear e restrições não lineares



Obtido em: (CORRAR; THEÓPHILO; BERGMANN, 2004)

De acordo com CORREIA (2017) as técnicas de programação não-linear são separadas em dois grupos: métodos determinísticos (clássicos) e não-determinísticos (evolucionários). Os métodos clássicos, são caracterizados pela conversão de várias funções objetivas em apenas uma função de maneira que o problema seja solucionado como uma única questão de otimização. Contudo, os métodos de otimização evolucionários são baseados na teoria da evolução das espécies, utilizando princípios que procuram imitar processos encontrados na natureza como a seleção e evolução.

Na Tabela 1 tem-se uma representação que compara as programações linear e não-linear. Desta forma, diversos quesitos são analisados, desde a garantia de obtenção de uma solução ótima, até ser mais viável na programação linear em relação a não-linear, custos computacionais envolvidos, complexidade e cuidado na análise dos dados são premissas que diferenciam as problemáticas.

Tabela 1 – Comparação entre a Programação linear e não-linear

Quesito	Programação linear	Programação não-linear
Representação do problema	Restrita, porque não considera aspectos causadores de não linearidade, tais como:eficiência e ineficiência de produtos em escalas diferentes, efeitos da quantidade de venda nos preços unitários.	Abrangente à medida que tenta incorporar esses aspectos desconsiderados no modelo linear.
Nível de complexidade	Simplificado, devido à abordagem restrita do problema.	Complexo,em virtude da riqueza e abrangência abordada.
Custo de processamento	Baixo	Alto
Aplicabilidade	Quando o problema tem limitada área de soluções possíveis e existe boa noção sobre o posicionamento da solução ótima, possibilitando aproximação linear adequada.	Ao contrário, quando o problema tem ampla área de soluções possíveis e inexiste boa noção sobre o posicionamento da solução ótima, dificultando aproximação linear adequada.
Nível de cautela na análise dos resultados	Alto	Baixo, devido ao maior esforço de incluir os aspectos que causam não-linearidade.

Fonte: Adaptado de (CORRAR; THEÓPHILO; BERGMANN, 2004)

2.2.3 Programação linear inteira

Na programação Linear Inteira todas as variáveis do problema pertencem ao conjunto dos números inteiros. Por exemplo, quando uma varável x representa a quantidade de trabalhadores de uma obra. Assim sendo, fica sem sentido atribuir o valor $2,5$ para os trabalhadores. Portando um número inteiro e necessário como resposta. Quando as variáveis do problema são números inteiros o modelo é designado como **Programação Inteira Pura**, de outro modo, é denominado **Programação Inteira Mista** (YUNES, 2000).

Nos problemas onde uma série de decisões ("sim"ou "não") devem ser usados os conceitos envolvendo programação inteira. Assim, perguntas como: "É viável um investimento fixo?"; Possam ser respondidas e representadas. Consequentemente pode-se representar uma decisão caracterizada por $\boldsymbol{j}$ onde as variáveis binárias são representadas com apenas dois valores, por exemplo, 0 e 1 conforme a Equação 2.5.

$$x_j = \begin{cases} 0 & se\ a\ decisao\ \boldsymbol{j}\ for\ nao \\ 1 & se\ a\ decisao\ \boldsymbol{j}\ for\ sim \end{cases} \tag{2.5}$$

Dessa forma, quando todas as variáveis do problema forem binárias pode-se classificar o modelo como **Programação Linear Inteira Binária** (HILLIER; LIEBERMAN, 2013).

Na Programação Linear Inteira (PLI) os problemas são considerados difíceis, não há algoritmos polinomiais conhecidos para resolver um problema genérico desse método. Deste modo é recomendado, recorrer a alguns recursos algorítmicos que possuem complexidade exponencial com um diferencial de percorrer os espaços de soluções de maneira cautelosa (YUNES, 2000).

2.2.4 Método Branch and Bound

A metodologia do *Branch-and-Bound* é alicerçada no conceito de relacionar todas as possíveis soluções viáveis de um problema de otimização. O conceito foi apresentado inicialmente por Land e Doig em 1960. Os termos *Branch* tange ao processo de particionamento e *bound* aos limites inferiores dos problemas de minimização os quais são utilizados para conceber resultados otimizados sem efetuar processamentos exaustivos (ALBUQUERQUE; SANTOS; FILHO, 2011).

De acordo com KAWAMURA (2006) o método emprega o conceito básico de dividir e conquistar, ou seja, um algoritmo de enumeração implícita, isto é, que mesmo ao não testar categoricamente todas as soluções possíveis, consegue uma solução ótima para o problema. De forma que ao otimizar subconjuntos mínimos, considera-se o resultado total ao contrário de tratar integralmente o conjunto de soluções. De acordo com (REIS; DUQUE; VILLELA, 2010) o método pode ser divido em duas operações básicas:

1. **Particionamento**: decompor o problema principal em sub-problemas menores de maneira a facilitar a exploração, eliminando as soluções inviáveis, sem empenhar a integridade dos resultados;

2. **Poda**: extinguir as soluções de baixa qualidade através de comparações com os limites do problema. Estes limites são: superior e inferior.

   Nos problemas de minimização o limite superior é um valor viável da função objetivo, não obrigatoriamente o valor ótimo, utilizado para servir de parâmetro de avaliação das soluções. Dessa forma quaisquer resultados que sejam superiores ao limitante superior serão descartados por se tratarem de soluções piores do que a atualmente conhecida.

   O limite inferior nos problemas de minimização e considerado uma estimativa da função objetivo pegando como base a solução parcial ate então obtida. De forma que o limite inferior é sempre menor ou igual que o valor da função objetivo, devido ao fato que seu resultado é baseado em um subconjunto da solução enquanto a função objetivo é calculada considerando a solução completa. Consequentemente, é possível extinguir as soluções que tenham limitantes inferiores piores do que os atuais limitantes superiores conhecidos.

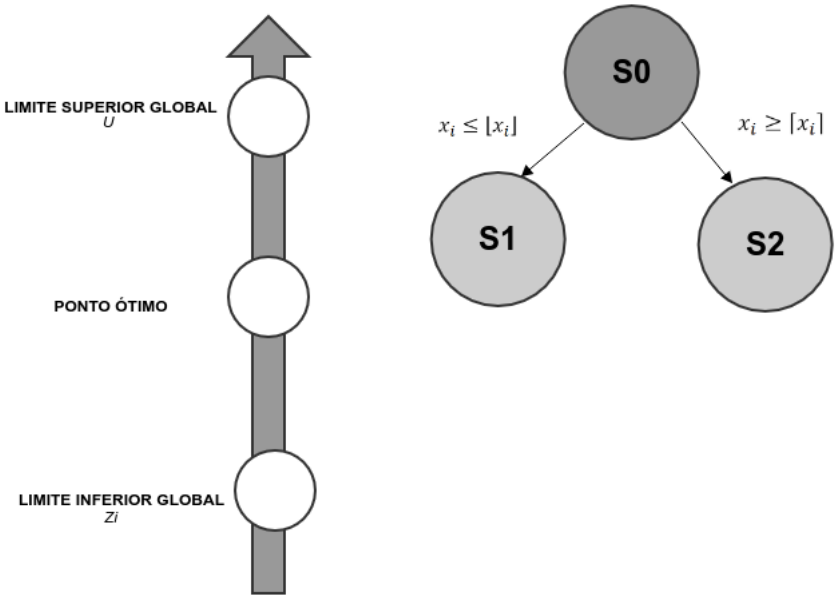
Conforme (ALBUQUERQUE; SANTOS; FILHO, 2011) o particionamento sucessivo do espaço de busca em sub-conjuntos i (ramos) partindo dos mais fáceis e após os cálculos dos limites inferiores (z_i) para cada subproblema. Tais ações permitem manter o limite superior de referência U atualizado ao encontrar resultados viáveis. Dessa forma tem-se a solução ótima entre os dois limites demonstrados na Figura 7a.

A pesquisa no método se inicia no (nó) raiz, que condiz com o problema de programação linear original. De forma que primeiramente e solucionada a relaxação linear do problema, que retorna uma solução x. Assim caso x seja uma solução inteira, então o problema terá sido solucionado. Se a solução x for fracionada, então ramos são abertos com variáveis xi composta por valores inteiros. Assim gerando dois subproblemas conforme a Figura 7b com suas restrições.

Figura 7 – Limites e Particionamento - Branch and Bound

(a) Limites no método e a solução ótima

(b) Particionamento no método



Fonte: Adaptado de (ALBUQUERQUE; SANTOS; FILHO, 2011)

Assim, em (ALBUQUERQUE; SANTOS; FILHO, 2011) após a resolução dos subproblemas, tem-se os casos onde é necessário ou não retirar um ramo da árvore de busca pelos seguintes motivos:

- $z_i = U$: A solução ideal foi processada acontece a poda por resolução;
- $z_i > U$: O limite inferior e maior do que o limite superior global. Dessa forma o ramo não pode produzir sendo a melhor solução ser podado por limitação;

- $zi < U$: O ramo pode conter uma solução mais adequada, portanto, deve ser particionado;
- Diante de um subproblema com solução impraticável ocorre a poda.

2.3 Programação de tarefas em máquinas paralelas

Inicialmente tem-se a programação de tarefas (*job scheduling*), que de acordo com ETCHEVERRY (2012) trata-se do processo de ordenamento das atividades e escolha do recurso mais indicado para a execução de uma tarefa, considerando as limitações de tempo e as interações entre as execuções.

Os problemas de programação podem envolver uma série de possibilidades de combinações na utilização dos recursos, resultando em inúmeras soluções com custos diferentes. A programação é encarregada da alocação das atividades nas máquinas para serem executadas em um intervalo de tempo delimitado. Ao observar o ambiente real de uma empresa, por exemplo, pode-se abstrair os recursos e atividades. Os recursos através de uma pista de corrida e unidades de processamento de um *data center*. As tarefas mediante os *pit stops* nas pistas de corrida e a execução de um sistema de computador (QUEIROZ, 2011).

Dessa maneira, problemas de programação de tarefas em máquinas paralelas nada mais são que questões de otimização, onde o objetivo fundamental é alocar um número (n) determinado de atividades, para uma quantidade de máquinas paralelas (m), com tempos de execução relacionados. Assim depois que as tarefas são distribuídas entre os dispositivos, a totalidade dos tempos das atividades da máquina com maior quantidade de tarefas (*makespan*), seja o mínimo possível (MÜLLER; DIAS, 2002).

Os ambientes de produção formados por máquinas paralelas de acordo com (QUEIROZ, 2011; MÜLLER; DIAS, 2002) podem ser classificados em relação à máquina utilizada: máquinas paralelas idênticas (*identical parallel machines*), máquinas paralelas uniformes (*uniform parallel machines*) e máquinas paralelas não relacionadas (*unrelated parallel machines*).

As máquinas paralelas são **idênticas** quando uma tarefa pode ser processada em qualquer máquina utilizando o mesmo período de processamento, Caso uma empresa resolva, por exemplo, iniciar a implantação de linhas de produção, no qual, após um procedimento de testes baseados nos processos de simulação, resulta na identificação de uma série de empecilhos. Dessa maneira, ao constatar a proliferação do problema na linha de produção percebeu-se a probabilidade que as máquinas sejam idênticas (QUEIROZ, 2011; MÜLLER; DIAS, 2002).

Nas máquinas paralelas **uniformes**, quando o tempo de processamento de uma

tarefa qualquer poderá necessitar de um tempo menor ou maior para as atividades as quais estão submetidas. Dessa maneira, por exemplo, quando empresa de produção de veículos, cujo aumento da fabricação, demandou a instalação de novas máquinas, as quais têm atributos iguais, contudo, devido aos aprimoramentos tecnológicos, possuem velocidade de processamentos distintos (QUEIROZ, 2011; MÜLLER; DIAS, 2002).

Os problemas de programação de tarefas em máquinas paralelas **não relacionadas** (PMNR), são considerados ser os mais difíceis. Estes problemas de programação linear, tratam da alocação de tarefas e otimização, ou seja, (maximização ou minimização) das variáveis na função objetivo (SERAFINI; ANZANELLO; KAHMANN, 2016).

Nas máquinas paralelas não relacionadas o tempo de processamento das atividades depende, simultaneamente, das características da tarefa e velocidade da máquina. Quando, por exemplo, tem-se um processo de readequação da linha de produção de uma montadora de veículos, onde existem diversas máquinas com tarefas distintas, como montar bancos, colocar portas e capuzes. Assim, as máquinas podem ser mais rápidas ou mais lentas para determinados procedimentos, resultando na impossibilidade de se estabelecer uma associação de velocidade entre as tarefas. No contexto da programação de máquinas algumas funções objetivo vem ganhando relevância nos estudos, tais soluções necessitam de um tempo de processamento exponencial, com intuito de obter a solução ótima ao passo que o número de equipamentos ou atividades aumenta. Dessa forma, diversas propostas de algoritmos para reduzir os tempos de processamento tem gerado estudos e soluções pertinentes (ROCHA, 2006; ETCHEVERRY, 2012).

Conforme HADDAD (2012), ROCHA (2006), ETCHEVERRY (2012) relaciona-se às quatro funções objetivo, que são abordadas no estudo:

1. **Atraso total das tarefas** (*makespan*): significa minimizar o maior tempo de encerramento de um conjunto de atividades. A otimização (minimização) do *makespan* demanda em num melhor aproveitamento das máquinas;

2. **Atraso total ponderado** (*tardiness*): estabelecer um sequenciamento onde a soma ponderada dos atrasos das tarefas faça-se mínima;

3. **Adiantamento total ponderado** (*lateness*): estabelecer um sequenciamento com um valor mínimo para o adiantamento total ponderado;

4. **Soma ponderada de atrasos e adiantamentos**: estabelecer um sequenciamento com um valor mínimo para o adiantamento e atraso das atividades.

3 Metodologia

Este capítulo aborda o modelo matemático e os métodos implementados para resolução do problema de programação de máquinas paralelas não relacionadas descrito na seção 2.3. Inicialmente na seção 3.1 será descrito o problema a ser estudado e seus principais aspectos, em seguida na seção 3.2 tem-se a formulação matemática do problema, na seção 3.3 são apresentados os materiais que auxiliarão na resolução da questão e por fim, na seção 3.4 são apresentados os algoritmos dos métodos implementados.

3.1 Descrição do Problema

3.1.1 Problema de máquina paralelas não relacionadas

Conforme descrito por ETCHEVERRY (2012) os problemas de sequenciamento de tarefas em máquinas paralelas não relacionadas são formados por conjuntos $N = \{1, ..., n\}$ de tarefas indivisíveis e por grupos $M = \{1, ..., m\}$ de máquinas, de forma que todas as N atividades possam ser processadas por alguma das máquinas M.

O problema em estudo de sequenciamento contém as seguintes restrições, estabelecidas conforme (ROCHA, 2006; QUEIROZ, 2011; AZUMA, 2011):

1. Cada tarefa j, possui um tempo de execução t_{ij} e deve ser processada uma única vez em cada máquina i;
2. As tarefas contam com uma mesma data ideal d de entrega;
3. Qualquer tarefa j que for entregue adiantada ou atrasada em relação à data ideal d de entrega sofre uma penalidade w_j proporcional a cada dia.

Parâmetros de Entrada:

- $\boldsymbol{N}$: Número de tarefas a serem executados;
- $\boldsymbol{M}$: Número de máquinas paralelas;
- $\boldsymbol{i}$: Indexação das máquinas, $i = 1, ..., M$;
- $\boldsymbol{j}$: Indexação de cada tarefa a ser processada, $i = 1, ..., N$;
- $\boldsymbol{d_j}$: Data de entrega da tarefa j;
- $\boldsymbol{L_j}$: Atraso na execução da tarefa j;

- $\boldsymbol{T_j}$: Antecipação na execução da tarefa;
- $\boldsymbol{t_{ij}}$: tempo de processamento da tarefa j pela máquina i;
- $\boldsymbol{w_j}$: Penalidade empregada à tarefa j na condição de adiantada ou atrasa em relação a d_j;
- $\boldsymbol{C_j}$: Tempo de conclusão da tarefa j;
- $\boldsymbol{Z}$: Tempo necessário para o término de todas as tarefas, conhecido como *makespan*.

Variáveis de Decisão:

- $\boldsymbol{x_{ij}}$: tarefa j processada na máquina i.

Variáveis iniciais do problema:

- A data de entrega de cada tarefa (t_{ij}): 6;
- O número de máquinas (M): 5;
- O número de tarefas (N): 25.

Os tempos de processamento de cada tarefa por máquina (em dias) e os pesos referentes aos problemas de adiantamento e atraso foram escolhidos e organizados de maneira determinística, a partir dos estudos realizados em (HADDAD, 2012; ROCHA, 2006; ETCHEVERRY, 2012) é podem ser observados na Tabela 2.

Tabela 2 – Tempos de processamento de tarefas por máquina

Tarefas	**Máquinas**					**Peso**
	1	**2**	**3**	**4**	**5**	
1	2	1	4	7	8	8
2	8	3	2	1	5	5
3	8	8	8	4	1	7
4	4	9	10	4	5	10
5	9	10	7	5	3	2
6	3	3	4	3	8	5
7	9	1	1	8	3	2
8	10	6	4	9	6	8
9	9	8	1	1	9	10
10	6	1	4	10	6	6
11	10	10	6	5	9	3

Tarefas	Máquinas					Peso
	1	**2**	**3**	**4**	**5**	
12	4	7	6	2	6	7
13	9	5	3	6	2	2
14	4	7	3	8	1	7
15	2	9	10	8	6	2
16	5	8	2	6	9	10
17	7	8	7	1	8	1
18	6	9	1	8	9	1
19	1	5	8	8	10	6
20	3	2	7	9	4	1
21	1	6	7	9	10	1
22	10	8	4	4	9	9
23	6	2	9	3	8	5
24	2	1	1	6	5	10
25	1	9	3	10	6	3

3.2 Formulação matemática proposta

Esta seção apresenta a formulação das equações obtidas a partir do estudo dos diversos trabalhos relacionados HADDAD (2012), XAVIER (2003), COUTINHO e DE OLIVEIRA (2009), adaptadas e alteradas para o tratamento do problema proposto, de forma a possibilitar a aplicação de algoritmos lineares e, por conseguinte, permitir a validação da modelagem. Conforme descrito nas seções 2.1.1 e 2.1.4 pode-se constatar que o problema possui diversas variáveis de controle descritas na seção 3.1.1, Desta forma, concebendo abordagens mono e multiobjetivos para a solução. Assim, o estudo possibilitou a criação de quatro conjuntos de equações (3.1, 3.4, 3.5 e 3.6) para os problemas mono-objetivos e dois para os problemas multiobjetivos (3.7 e 3.8).

(a) Agrupamento de equações para os problemas mono-objetivos:

O Conjunto de Equações 3.1 para resolução do problema de minimização do tempo total de entrega:

$$\begin{cases} 1. \;\; c_1 = min\ Z \\ 2. \;\; \sum_{i=1}^{M} \sum_{j=1}^{N} \; t_{ij} \cdot x_{ij} \leq c_1 \\ 3. \;\; \sum_{i=1}^{M} \; x_{ij} = 1, j = 1, ..., N \\ 4. \;\; x_{ij} = 0 \; ou \; 1, i = 1, ..., m; \;\; j = 1, ..., N \end{cases} \tag{3.1}$$

No conjunto de Equações 3.1, o objetivo da linha (1) é a variável c_1 receber a minimização do *Makespan*. As restrições na linha (2) garantem que a tarefa processada pela máquina t_{ij} relacionada pelo tempo de processamento da tarefa x_{ij}, seja menor ou igual ao *Makespan*. As restrições na linha (3) garantem que cada tarefa seja processada pela máquina apenas uma vez. Finalmente, as restrições da linha (4) definem o domínio das variáveis.

O atraso L_j de cada tarefa e apresentado pela Equação 3.2:

$$L_j = max(C_j - d_j, 0) \tag{3.2}$$

A equação 3.2 tem-se o atraso da execução das tarefas L_j resultante do tempo de conclusão da tarefa C_j subtraído pela data de entrega da tarefa d_j.

O adiantamento T_j de cada tarefa e apresentado pela Equação 3.3:

$$T_j = max(d_j - C_j, 0) \tag{3.3}$$

A equação 3.2 tem-se o adiantamento da execução das tarefas T_j resultante da data de entrega da tarefa d_j subtraído pelo tempo de conclusão da tarefa C_j.

O Conjunto de Equações 3.4 para resolução do problema de minimização da soma ponderada dos atrasos:

$$\begin{cases} 1. \;\; c_2 = min\ (\sum_{j=1}^{N}\ w_j \cdot L_j) \\ 2. \;\; \sum_{i=1}^{M} \sum_{j=1}^{N}\ (w_j \cdot L_j) \cdot x_{ij} \leq c_2 \\ 3. \;\; \sum_{i=1}^{M}\ x_{ij} = 1, j = 1, ..., N \\ 4. \;\; x_{ij} = 0\ ou\ 1, i = 1, ..., m;\ \ j = 1, ..., N \end{cases} \tag{3.4}$$

No conjunto de Equações 3.4, o objetivo da linha (1) é a variável c_2 receber a minimização da soma ponderada dos atrasos (*tardiness*). As restrições na linha (2) garantem que a penalidade emprega para cada tarefa w_j relacionada pelo atraso de cada tarefa L_j, seja menor ou igual ao *tardiness*. As restrições na linha (3) garantem que cada tarefa seja processada pela máquina apenas uma vez. Enfim, as restrições da linha (4) definem o domínio das variáveis.

O Conjunto de Equações 3.5 para resolução do problema de minimização da soma ponderada dos adiantamentos:

$$\begin{cases} 1. \;\; c_3 = min\ (\sum_{j=1}^{N}\ w_j \cdot T_j) \\ 2. \;\; \sum_{i=1}^{M} \sum_{j=1}^{N}\ (w_j \cdot T_j) \cdot x_{ij} \leq c_3 \\ 3. \;\; \sum_{i=1}^{M}\ x_{ij} = 1, j = 1, ..., N \\ 4. \;\; x_{ij} = 0\ ou\ 1, i = 1, ..., m;\ j = 1, ..., N \end{cases} \tag{3.5}$$

No conjunto de Equações 3.5, o objetivo da linha (1) é a variável c_3 receber a minimização da soma ponderada dos adiantamentos (*lateness*). As restrições na linha (2) garantem que a penalidade emprega para cada tarefa w_j relacionada pelo adiantamento de cada tarefa T_j, seja menor ou igual ao *Lateness*. As restrições na linha (3) garantem que cada tarefa seja processada pela máquina apenas uma vez. Enfim, as restrições da linha (4) definem o domínio das variáveis.

O Conjunto de Equações 3.6 para resolução do problema de minimização da soma ponderada de atrasos e adiantamentos:

$$\begin{cases} 1. \;\; c_4 = min\ (\sum_{j=1}^{N}\ w_j \cdot (L_j \cdot T_j)) \\ 2. \;\; \sum_{i=1}^{M} \sum_{j=1}^{N}\ (w_j \cdot (L_j + T_j)) \cdot x_{ij} \leq c_4 \\ 3. \;\; \sum_{i=1}^{M}\ x_{ij} = 1, j = 1, ..., N \\ 4. \;\; x_{ij} = 0\ ou\ 1, i = 1, ..., m;\ j = 1, ..., N \end{cases} \tag{3.6}$$

No conjunto de Equações 3.6, o objetivo da linha (1) é a variável c_4 receber a minimização da soma ponderada dos atrasos e adiantamentos, onde temos a penalidade empregada a tarefa pelo produto do relacionamento do adiantamento e dos atrasos. As restrições na linha (2) garantem que a penalidade emprega para cada tarefa w_j relacionada pelo produto da soma dos adiantamentos e dos atrasos seja menor ou igual ao objetivo. As restrições na linha (3) garantem que cada tarefa seja processada pela máquina apenas uma vez. Por fim, as restrições da linha (4) definem o domínio das variáveis.

(b) Agrupamento de equações para os problemas multiobjetivos:

O Conjunto de Equações 3.7 para resolução do problema da minimização do *makespan*

× minimização da soma ponderada dos atrasos e adiantamentos:

$$\begin{cases} 1. \;\; c_1 = min\ Z \\ 2. \;\; c_4 = min\ \left(\sum_{j=1}^{N}\ w_j \cdot (L_j + T_j)\right) \\ 3. \;\; \sum_{i=1}^{M}\ \sum_{j=1}^{N}\ (w_i \cdot T_j)\ x_{ij} \leq c_1 \\ 4. \;\; \sum_{i=1}^{M}\ \sum_{j=1}^{N}\ (w_i \cdot T_j)\ x_{ij} \leq c_4 \\ 5. \;\; \sum_{i=1}^{M}\ x_{ij} = 1, j = 1, ..., N \\ 6. \;\; x_{ij} = 0\ ou\ 1, i = 1, ..., m;\ \ j = 1, ..., N \end{cases} \tag{3.7}$$

No conjunto de Equações 3.7, o objetivo da linha (1) é a variável c_1 receber a minimização do makespan. Na linha (2) o objetivo e a variável c_4 receber a minimização da soma ponderada dos atrasos e adiantamentos. As restrições da linha (3) e (4) respectivamente seja menor ou igual aos objetivos c_1 e c_4. As restrições na linha (5) garantem que cada tarefa seja processada pela máquina apenas uma vez. Enfim, as restrições da linha (6) definem o domínio das variáveis.

O Conjunto de Equações 3.8 para resolução do problema da minimização da soma ponderada dos atrasos × minimização da soma ponderada dos adiantamentos:

$$\begin{cases} 1. \;\; c_2 = min\ \left(\sum_{j=1}^{N}\ w_j \cdot L_j\right) \\ 2. \;\; c_3 = min\ \left(\sum_{j=1}^{N}\ w_j \cdot T_j\right) \\ 3. \;\; \sum_{i=1}^{M}\ \sum_{j=1}^{N}\ (w_i \cdot T_j)\ x_{ij} \leq c_2 \\ 4. \;\; \sum_{i=1}^{M}\ \sum_{j=1}^{N}\ (w_i \cdot T_j)\ x_{ij} \leq c_3 \\ 5. \;\; \sum_{i=1}^{M}\ x_{ij} = 1, j = 1, ..., N \\ 6. \;\; x_{ij} = 0\ ou\ 1, i = 1, ..., m;\ \ j = 1, ..., N \end{cases} \tag{3.8}$$

No conjunto de Equações 3.8, o objetivo da linha (1) é a variável c_2 receber a minimização da soma pondera dos atrasos. Na linha (2) o objetivo e a variável c_3 receber a minimização da soma ponderada dos adiantamentos. As restrições da linha (3) e (4) respectivamente sejam menor ou igual aos objetivos c_2 e c_3. As restrições na linha (5) garantem que cada tarefa seja processada pela máquina apenas uma vez. Enfim, as restrições da linha (6) definem o domínio das variáveis.

3.3 Materiais

Para a implementação e avaliação dos métodos descritos utilizou-se o software Matlab® na versão R2017a[1]. Para processamento dos resultados utilizou-se o software

[1] https://www.mathworks.com/products/matlab.html

RStudio® versão 1.0.153[2] livre que possui um ambiente integrado com a liguagem R[3], tendo foco principal de uso criação de gráficos e cálculos estatísticos. As rotinas foram desenvolvidas e executadas em um Notebook HP® Pavilion dv4-2173nr[4], com processador Intel® Core i5-430M de 2.26GHz, 8GB de memória RAM e sistema Linux Ubuntu 16.04 64bits.

Os problemas de tempo de processamento das tarefas conforme detalhado na seção 3.1, foram desenvolvidos utilizando a *toolbox* de otimização presente no Matlab®, o qual apresenta diversas funções voltadas para resolução de questões de otimização (PACHAMANOVA; FABOZZI, 2010). Na ferramenta existem diversas funções, como por exemplo, *fmincon*[5], *quadprog*[6], as quais geram os mesmos resultados, porem solicitam bem mais dados, visando clareza e simplicidade, será utilizada a função *intlingprog*[7] que utiliza o método *Branch-and-Bound* "vide 2.2.4"para o processamento dos dados.

3.3.1 Função *intlingprog*

A *intlingprog* soluciona problemas do tipo como pode ser observado na Equação 3.9:

$$\begin{cases} min \quad f(x)(inteiros) \\ s.t \quad A \cdot x \leq b \\ \quad\quad Aeq \cdot x \doteq beq \\ \quad\quad lb \leq x \leq ub \end{cases} \tag{3.9}$$

A função deve ser executada com pelo menos quatro parâmetros:

- Os coeficientes de "$\boldsymbol{f}$"da função linear operada;
- A matriz "$\boldsymbol{A}$"com os coeficientes do sistema de desigualdades, formado por $\boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}$;
- O vetor coluna $\boldsymbol{b}$ contendo os termos independentes do sistema de desigualdades;
- O vetor $\boldsymbol{intcon}$ contendo a quantidade de variáveis que possuem valores inteiros.

Alguns parâmetros opcionais podem ser incluídos na função como as restrições de igualdade, limites superiores e inferiores de x. Estas novas restrições podem ser passadas pelos seguintes parâmetros:

[2] https://www.rstudio.com/
[3] https://www.r-project.org/
[4] https://support.hp.com/
[5] https://www.mathworks.com/help/optim/ug/fmincon.html
[6] https://www.mathworks.com/help/optim/ug/quadprog.html
[7] https://www.mathworks.com/help/optim/ug/intlinprog.html

- A matriz de "$\boldsymbol{Aeq}$"contendo os coeficientes dos sitema de desigualdades, formado por $\boldsymbol{Aeq} \cdot \boldsymbol{x} \leq \boldsymbol{beq}$;
- O vetor coluna "$\boldsymbol{beq}$"com os termos independentes do sistema de desigualdades;
- O vetor coluna $\boldsymbol{lb}$ contendo os limites inferiores que os valores de x podem assumir;
- O vetor coluna $\boldsymbol{ub}$ contendo os limites superiores que os valores de x podem assumir.

Uma ilustração da utilização da função pode ser observado em 3.10:

$$x = intlinprog(f, A, b, Aeq, beq, lb, ub) \tag{3.10}$$

Ao passar os argumentos a função deve-se ter cuidado com a ordem disposta dos parâmetros os quais não podem mudar. Contudo, caso não exista alguma restrição para passar como argumento, deve se utilizar um vetor vazio na posição.

3.4 Implementação dos métodos

As equações matemáticas propostas na seção 3.2 foram implementadas utilizando o Matlab®, conforme citado anteriormente. A problemática em estudo se converte em problemas mono e multiobjetivos de programação linear inteira e binária. Dessa forma utilizou-se a função *intlinprog* do conjunto de ferramentas de otimização do Matlab, detalhado na seção 3.3.1, para criação das soluções.

Os algoritmos foram projetados para gerar saídas de dados em arquivos, os quais foram tratados utilizando a ferramenta Rstudio®, onde para fins estatísticos foram empregados os cálculos da média, mediana, variância, desvio padrão, com níveis de confiança de 95%. Os gráficos e tabelas foram criados também utilizando a ferramenta.

3.4.1 Pseudocódigos

3.4.1.1 Método do atraso total das tarefas (*Makespan*)

Algoritmo 1: Algoritmo para o atraso total das tarefas (*Makespan*)

Entrada: Tempo de Processamento de cada Tarefa por Máquina em dias (F),Número de Testes (It) e Interações (Zt), Máquinas (M), Tarefas (N), Coeficientes de desiqualdades(Aeq e beq), Limites inferiores e Superiores (lb e ub), Quantidade de variáveis inteiras ($intcon$)

Saída: Resultado vetor de soluções ótimas (x) e a Solução otimizada ($result$)

```
1  início
2      intcon ← (N * M); // Incializa variável com quantidade de dados inteiros
3      beq ← ones(N, 1); // Inicializa o vetor de desiqualdades
4      lb ← zeros(intcon, 1); // Inicializa o vetor de Limites inferiores
5      ub ← ones(intcon, 1); / Inicializa o vetor de Limites superiores
6      Aeq = kron(eye(N), ones(1, M)); // Inicilializa a matriz com os coefiencientes
         de desiqualdades
7      // Encontra e armazena a solução ótima para i de 1 até It faça
8          para j de 1 até Zt faça
9              [x, result] = intlinprog(F, intcon, [ ], [ ], Aeq, beq, lb, ub);
10         fim
11     fim
12 fim
```

No algoritmo 1 para o atraso total das tarefas as entradas são os tempos de processamento de cada máquina da problemática, representado pelo vetor F, os limites superiores lb e inferiores lu, a quantidade de números inteiros da preposição $intcon$ e a matriz com os coeficientes de desigualdade Aeq e beq.

Quando a matriz com os tempos de processamento é inserida em conjunto com a quantidade de máquinas e respectivas tarefas, o algoritmo em seguida gera uma matriz de coeficientes utilizando a função $kron$[8]. Contudo, os limites superiores e inferiores foram carregados com 0 e 1 de forma que todas as restrições fossem consideradas.

Desta maneira, os laços de repetição de testes e resultados serão repetidos baseados nas variáveis It e Zt diversas vezes. Dessa forma a função $intlingprog$ irá otimizar o grupo de dados com os tempos de execução F de cada máquina. Durante o cálculo do problema tem-se um conjunto de soluções mono-objetivas x binárias e um grupo de soluções ótimas $result$.

Algoritmo 2: Algoritmo para o atraso total ponderado (*Tardiness*)

Entrada: Tempo de Processamento de cada Tarefa por Máquina em dias (F),Número de Testes (It) e Interações (Zt), Máquinas (M), Tarefas (N),Penalidade de cada tarefa w atrasada ou adiantada, Coeficientes de desiqualdades(Aeq e beq), Limites inferiores e Superiores (lb e ub), Quantidade de variáveis inteiras ($intcon$), (fl) variável auxiliar e a variável com data de entrega de cada tarefa dd

Saída: Resultado vetor de soluções ótimas (x) e a Solução otimizada ($result$)

início
- $intcon \leftarrow (N * M)$; // Incializa variável com quantidade de dados inteiros
- $beq \leftarrow ones(N, 1)$; // Inicializa o vetor de desiqualdades
- $lb \leftarrow zeros(intcon, 1)$; // Inicializa o vetor de Limites inferiores
- $ub \leftarrow ones(intcon, 1)$; / Inicializa o vetor de Limites superiores
- $Aeq = kron(eye(N), ones(1, M))$; // Inicilializa a matriz com os coefiencientes de desiqualdades
- $fl = f - dd$; // Criando novo vetor de tarefas com restrição de atrasos
- // Laço que vai empregar a restricao do atraso de cada tarefa
- **para** *j de 1 até intcon* **faça**
  - **para** *i de 1 até M* **faça**
    - $fl(M * (j - 1) + i) = w(j) * fl(M * (j - 1) + i)$;
  - **fim**
- **fim**
- $fl(fl < 0) = 0$; // Remove todos os valores inferiores a zero.
- // Encontra e armazena a solução ótima **para** *i de 1 até It* **faça**
  - **para** *j de 1 até Zt* **faça**
    - $[x, result] = intlinprog(fl, intcon, [\,], [\,], Aeq, beq, lb, ub)$;
  - **fim**
- **fim**

fim

3.4.1.2 Método do atraso total ponderado (*Tardiness*)

Algoritmo 2 tem-se as variáveis iniciais os quais são carregadas com os tempos de processamento de cada máquina F, os limites superiores lb e inferiores lu, a quantidade de números inteiros processados $intcon$, as matrizes com os coeficientes de desigualdades Aeq e beq e com a matriz de penalidades para os atrasos ou adiantamentos w.

Quando os valores são inicializados o algoritmo em seguida cria um vetor com os valores de atraso de cada tarefa conforme a Equação 3.2, posteriormente as restrições da formulação são empregas pelo laço de repetição preparando a nova matriz para ser otimizada.

A matriz fl foi submetida a um processamento em relação à restrição de valores maiores que zero, posteriormente o laço final será executado baseado nas variáveis It e Zt diversas vezes. Dessa forma a função $intlingprog$ irá otimizar o grupo de dados com os

[8] https://www.mathworks.com/help/matlab/ref/kron.html

tempos de atraso de cada máquina. Durante o cálculo do problema tem-se um conjunto de soluções mono-objetivas x binárias e um grupo de soluções ótimas $result$.

3.4.1.3 Método do adiantamento total ponderado (*Lateness*)

Algoritmo 3: Algoritmo para o adiantamento total ponderado (*Lateness*)

Entrada: Tempo de Processamento de cada Tarefa por Máquina em dias (F),Número de Testes (It) e Interações (Zt), Máquinas (M), Tarefas (N),Penalidade de cada tarefa w atrasada ou adiantada, Coeficientes de desiqualdades(Aeq e beq), Limites inferiores e Superiores (lb e ub), Quantidade de variáveis inteiras ($intcon$), (ft) variável auxiliar e a variável com data de entrega de cada tarefa dd

Saída: Resultado vetor de soluções ótimas (x) e a Solução otimizada ($result$)

```
início
    w; // Carrega vetor com penalidades
    dd; // Data de entrega de cada tarefa em dias
    intcon ← (N * M); // Incializa variável com quantidade de dados inteiros
    beq ← ones(N, 1); // Inicializa o vetor de desiqualdades
    lb ← zeros(intcon, 1); // Inicializa o vetor de Limites inferiores
    ub ← ones(intcon, 1); / Inicializa o vetor de Limites superiores
    Aeq = kron(eye(N), ones(1, M)); // Inicilializa a matriz com os coefiencientes
     de desiqualdades
    ft = dd − f; // Criando novo vetor de tarefas com restrição de adiantamentos
    // Laço que vai empregar a restricao do adiantamento de cada tarefa
    para j de 1 até intcon faça
        para i de 1 até M faça
            ft(M * (j − 1) + i) = w(j) * ft(M * (j − 1) + i);
        fim
    fim
    // Encontra e armazena a solução ótima para i de 1 até It faça
        para j de 1 até Zt faça
            [x, result] = intlinprog(ft, intcon, [ ], [ ], Aeq, beq, lb, ub);
        fim
    fim
fim
```

Algoritmo 3 tem-se as variáveis iniciais os quais são carregadas. Quando os valores forem inicializados o algoritmo em seguida cria um vetor com os dados de adiantamento de cada tarefa conforme a Equação 3.3.

A nova matriz ft será submetida no laço de repetição que emprega as restrições de valores conforme Equação 3.5, posteriormente o laço final será executado baseado nas variáveis It e Zt diversas vezes. Dessa forma a função $intlingprog$ irá otimizar o grupo de dados com os tempos de adiantamento de cada máquina. Resultando um conjunto de soluções mono-objetivas x binárias e um grupo de soluções ótimas $result$.

3.4.1.4 Método da soma ponderada de atrasos e adiantamentos

Algoritmo 4: Algoritmo para a soma ponderada de atrasos e adiantamentos

Entrada: Tempo de Processamento de cada Tarefa por Máquina em dias (F),Número de Testes (It) e Interações (Zt), Máquinas (M), Tarefas (N),Penalidade de cada tarefa w atrasada ou adiantada, Coeficientes de desiqualdades(Aeq e beq), Limites inferiores e Superiores (lb e ub), Quantidade de variáveis inteiras ($intcon$), (ftl) variável auxiliar e a variável com data de entrega de cada tarefa dd

Saída: Resultado vetor de soluções ótimas (x) e a Solução otimizada ($result$)

```
1  início
2      N; // Quantidade de tarefas
3      M; // Quantidade de máquinas
4      F; // Carrega vetor de tempo de processamento por máquina
5      w; // Carrega vetor com penalidades
6      dd; // Data de entrega de cada tarefa em dias
7      intcon ← (N * M); // Incializa variável com quantidade de dados inteiros
8      beq ← ones(N, 1); // Inicializa o vetor de desiqualdades
9      lb ← zeros(intcon, 1); // Inicializa o vetor de Limites inferiores
10     ub ← ones(intcon, 1); / Inicializa o vetor de Limites superiores
11     Aeq = kron(eye(N), ones(1, M)); // Inicilializa a matriz com os coefiencientes
        de desiqualdades
12     ftl = abs(f − dd); // Criando novo vetor com valores absolutos das tarefas
13     // Laço que vai empregar a restrição a cada tarefa
14     para j de 1 até intcon faça
15         para i de 1 até M faça
16             ftl(M * (j − 1) + i) = w(j) * ftl(M * (j − 1) + i);
17         fim
18     fim
19     // Encontra e armazena a solução ótima para i de 1 até It faça
20         para j de 1 até Zt faça
21             [x, result] = intlinprog(ft, intcon, [ ], [ ], Aeq, beq, lb, ub);
22         fim
23     fim
24 fim
```

Algoritmo 4 tem-se as variáveis iniciais as quais são carregadas. Quando os valores são inicializados o algoritmo em seguida cria um vetor com os dados de atrasos e adiantamentos de cada tarefa, dando sequência a variável ftl recebe os valores absolutos dos tempos já com as restrições de execução de cada tarefa.

A nova matriz ftl será submetida no laço de repetição que emprega as restrições de valores conforme Equação 3.6, processando os tempos de execução para o adiantamento e atrasos das tarefas, posteriormente o laço final será executado, resultando saídas binárias e os valor ótimo da soma dos tempos de adiantamento e atrasos das máquinas.

3.4.1.5 Método da soma ponderada de atrasos e adiantamentos x Makespan

Algoritmo 5: Algoritmo para a soma ponderada de atrasos e adiantamentos x Makespan

Entrada: Tempo de Processamento de cada Tarefa por Máquina em dias (F),Número de Testes (It) e Interações (Zt), Máquinas (M), Tarefas (N),Penalidade de cada tarefa w atrasada ou adiantada, Coeficientes de desiqualdades(Aeq e beq), Limites inferiores e Superiores (lb e ub), Quantidade de variáveis inteiras ($intcon$), (ftl) variável auxiliar e a variável com data de entrega de cada tarefa dd

Saída: Resultado vetor de soluções ótimas (x) e a Solução otimizada ($result$)

início
- N; // Quantidade de tarefas
- M; // Quantidade de máquinas
- F; // Carrega vetor de tempo de processamento por máquina
- w; // Carrega vetor com penalidades
- dd; // Data de entrega de cada tarefa em dias
- $intcon \leftarrow (N * M)$; // Incializa variável com quantidade de dados inteiros
- $beq \leftarrow ones(N, 1)$; // Inicializa o vetor de desiqualdades
- $lb \leftarrow zeros(intcon, 1)$; // Inicializa o vetor de Limites inferiores
- $ub \leftarrow ones(intcon, 1)$; / Inicializa o vetor de Limites superiores
- $Aeq = kron(eye(N), ones(1, M))$; // Inicilializa a matriz com os coefiencientes de desiqualdades
- //funcao makespan
- $fm = fmakespan(f, N, M)$;
- //funcao soma adiantamentos e atrasos
- $ftl = fsomaAd(f, N, M, dd)$;
- **para** *i de 1 até It* **faça**
  - **para** *j de 1 até Zt* **faça**
    - // Empregas as retrições da relação entre makespan, atrasos e adiantamentos
    - $fx = Restric(fm, ftl)$;
    - $[x, result] = intlinprog(fx, intcon, [\,], [\,], Aeq, beq, lb, ub)$;
  - **fim**
- **fim**

fim

Algoritmo 5 as entradas são os tempos de processamento de cada máquina da formulação, representado pelo vetor F, os limites superiores lb e Inferiores lu, a quantidade de números inteiros da preposição $intcon$ a matriz com os coeficientes de desigualdade Aeq e beq e a matriz de penalidades para os atrasos ou adiantamentos w.

O próximo passo após o carregamento das informações iniciais, e a execução da função $fmakespan$, criada com o objetivo de preparar os dados de forma bruta e não otimizada com as primeiras restrições da equação proposta na seção 3.7. Carregando o vetor fm com dados prontos para serem processados em sequência.

Posteriormente a função $fsomaAd$ prepara dos dados com as demais restrições do problema, inicializando o vetor ftl com os dados da somada ponderada dos atrasos e adiantamentos.

Nos laços de repetição de testes e resultados tem-se a função *Restric* o qual recebe os vetores com os dados prontos e emprega a restrições dos objetivos pelas máquinas em execução. Dessa forma, a cada loop a variável fx será carregada com um conjunto multiobjetivo de restrições, seguidamente o vetor é empregado na execução da função de otimização *intlingprog*, resultando no conjunto de valores otimizados.

3.4.1.6 Método da soma ponderada de atrasos x adiantamentos

Algoritmo 6: Algoritmo para a soma ponderada de atrasos x adiantamentos

Entrada: Tempo de Processamento de cada Tarefa por Máquina em dias (F),Número de Testes (It) e Interações (Zt), Máquinas (M), Tarefas (N),Penalidade de cada tarefa (w), Coeficientes de desiqualdades(Aeq e beq), Limites inferiores e Superiores (lb e ub), Variáveis inteiras ($intcon$), (ftl) variável auxiliar e a variável com data de entrega de cada tarefa (dd)

Saída: Resultado vetor de soluções ótimas (x) e a Solução otimizada ($result$)

```
1  início
2      N; // Quantidade de tarefas
3      M; // Quantidade de máquinas
4      F; // Carrega vetor de tempo de processamento por máquina
5      w; // Carrega vetor com penalidades
6      dd; // Data de entrega de cada tarefa em dias
7      intcon ← (N * M); // Incializa variável com quantidade de dados inteiros
8      beq ← ones(N, 1); // Inicializa o vetor de desiqualdades
9      lb ← zeros(intcon, 1); // Inicializa o vetor de Limites inferiores
10     ub ← ones(intcon, 1); / Inicializa o vetor de Limites superiores
11     Aeq = kron(eye(N), ones(1, M)); // Inicilializa a matriz com os coefiencientes
        de desiqualdades
12     //funcões da soma ponderada dos atrasos e soma adiantamentos
13     ft = ftardiness(f, N, M, dd);
14     fl = flateness(f, N, M, dd);
15     para i de 1 até It faça
16         para j de 1 até Zt faça
17             // Empregas as retrições da relação entre atrasos e adiantamentos
18             fx = Restric2(ft, fl);
19             [x, result] = intlinprog(fx, intcon, [ ], [ ], Aeq, beq, lb, ub);
20         fim
21     fim
22 fim
```

Algoritmo 6 após o carregamento das informações iniciais, tem-se a execução da função $ftardiness$, inicializando o vetor ft com os dados de forma bruta da soma ponderada

dos atrasos. Posteriormente a função $flateness$ carrega os dados com as restrições da soma dos adiantamentos no vetor fl.

Nos laços de repetição de testes e resultados tem-se a função $Restric2$ o qual recebe os conjuntos de dados dos problemas multiobjetivos e emprega as restrições do conjunto de equações 3.8. Desta maneira, a cada loop a variável fx será carregada com um conjunto multiobjetivo de restrições, seguidamente o vetor e empregado na execução da função de otimização *intlingprog* resultado nas soluções ótimas.

4 Resultados

4.1 Introdução

Neste capítulo são demonstrados e discutidos os conjuntos de dados resultantes do processamento das funções desenvolvidas, conforme modelagem matemática proposta em 3.2. Os códigos propostos entregam duas saídas de dados composta por um valor ótimo em que representa a execução das tarefas em dias conforme a formulação matemática proposta, e um conjunto de elementos em forma de uma matriz ($N\ x\ M$) contendo os valores binários resultantes do processamento das restrições.

Dessa forma a matriz otimizada resultante demonstra uma sequência ótima para a execução das tarefas por máquina para os problemas mono-objetivos. Contudo, nos problemas multiobjetivos os resultados serão relacionados por problemáticas e demonstrados em gráficos para análise dos pontos ótimos na fronteira de Pareto, com a finalidade de mostrar a visualização do conjunto de soluções viáveis obtidas.

As informações das execuções foram exportadas para diversos arquivos, no formato e *csv*, seguidamente importados para a ferramenta RStudio®, utilizada na construção de tabelas e gráficos.

4.2 Apresentação de resultados

4.2.1 Problemas mono-objetivos

Algoritmo 1 implementado em Matlab® para o problema de minimização do tempo total de entrega (*makespan*) resultou como saída um número inteiro correspondente a 48 dias e um vetor com 25 posições com a sequência ótima de execução das tarefas por máquina considerando as restrições do problema. Assim, pode-se observar o emprego de uma das restrições ao analisar cada linha tem-se apenas um único valor (1) que representa a máquina que executou a tarefa no respectivo intervalo. Na Tabela 3 tem-se os dados do vetor ótimo resultante do respectivo problema.

Tabela 3 – Resultados para o problema de *makespan*

Tarefas	**Máquinas**				
	1	**2**	**3**	**4**	**5**
1	0	1	0	0	0
2	0	0	0	1	0

Tarefas	**Máquinas**				
	1	**2**	**3**	**4**	**5**
3	0	0	0	0	1
4	1	0	0	0	0
5	0	0	0	0	1
6	0	0	0	1	0
7	0	1	0	0	0
8	0	0	1	0	0
9	0	0	0	1	0
10	0	1	0	0	0
11	0	0	0	1	0
12	0	0	0	1	0
13	0	0	0	0	1
14	0	0	0	0	1
15	1	0	0	0	0
16	0	0	1	0	0
17	0	0	0	1	0
18	0	0	1	0	0
19	1	0	0	0	0
20	0	1	0	0	0
21	1	0	0	0	0
22	0	0	1	0	0
23	0	1	0	0	0
24	0	1	0	0	0
25	1	0	0	0	0

Para os problemas de minimização da soma pondera dos atrasos (*tardiness*) utilizando o Algoritmo 2 desenvolvido, empregamos tanto as restrições com os tempos de entrega de cada tarefa e com o tempo de atrasos por execução de tarefa, resultando no tempo **0** de tarefas atrasadas. Na Tabela 4 tem-se a matriz otimizada com o sequenciamento das 25 tarefas para as 5 máquinas conforme problema, correspondendo a um valor de atrasos igual a zero.

Tabela 4 – Resultados para o problema de *tardiness*

Tarefas	**Máquinas**				
	1	**2**	**3**	**4**	**5**
1	0	0	0	0	1
2	1	0	0	0	0

Tarefas	**Máquinas**				
	1	**2**	**3**	**4**	**5**
3	0	0	1	0	0
4	0	1	0	0	0
5	0	0	1	0	0
6	0	0	0	0	1
7	0	0	0	1	0
8	0	0	0	0	1
9	0	0	0	0	1
10	0	0	0	0	1
11	0	0	0	0	1
12	0	0	0	0	1
13	0	0	0	1	0
14	0	1	0	0	0
15	0	0	0	0	1
16	0	0	0	0	1
17	0	0	0	0	1
18	0	0	0	0	1
19	0	0	0	0	1
20	0	0	0	1	0
21	0	0	0	0	1
22	0	0	0	0	1
23	0	0	0	0	1
24	0	0	0	1	0
25	0	0	0	1	0

No que se refere aos problemas de minimização da soma pondera dos adiantamentos (*lateness*) o Algoritmo 3 desenvolvido subtraiu de cada tarefa executada o tempo de processamento das tarefas e posteriormente empregou a restrição de tempo em cada atividade. Os resultados obtidos para o problema de sequenciamento resultam em dados otimizados. Assim, tem-se um vetor com 25 linhas (tarefas) e 5 colunas (máquinas) demonstrado na Tabela 5, onde a sequência ótima resulta num valor de adiantamento igual a **0**.

Tabela 5 – Resultados para o problema de *lateness*

Tarefas	**Máquinas**				
	1	**2**	**3**	**4**	**5**
1	0	0	1	0	0

Tarefas	**Máquinas**				
	1	**2**	**3**	**4**	**5**
2	0	0	0	0	1
3	0	0	0	1	0
4	0	0	0	0	1
5	0	0	0	1	0
6	0	0	0	1	0
7	0	0	0	0	1
8	0	0	0	0	1
9	0	0	0	1	0
10	0	0	0	0	1
11	0	0	0	1	0
12	0	0	0	0	1
13	0	0	0	0	1
14	0	0	0	0	1
15	0	0	0	0	1
16	0	0	0	1	0
17	0	0	0	1	0
18	0	0	1	0	0
19	0	1	0	0	0
20	0	0	0	0	1
21	0	1	0	0	0
22	0	0	0	1	0
23	0	0	0	1	0
24	0	0	0	0	1
25	0	0	1	0	0

No que tange aos problemas de minimização do tempo total de entrega o Algoritmo 4 considera a soma de atrasos e adiantamentos por meio dos valores absolutos da soma do tempo de atrasos e adiantamentos das tarefas e posteriormente emprega a restrição de tempo em cada atividade. Dessa maneira, resulta-se em uma matriz com a sequência ótima para o processamento de adiantamentos e atrasos, conforme Tabela 6. Contudo, para este problema quando as restrições da soma dos atrasos e adiantamentos e relacionada pelas penalidades do problema. Assim, resulta num valor ótimo considerável de **112** dias.

Tabela 6 – Resultados para o problema de minimização do tempo total da soma ponderada de atrasos e adiantamentos

Tarefas	**Máquinas**				
	1	**2**	**3**	**4**	**5**
1	0	0	0	1	0
2	0	0	0	0	1
3	0	0	0	1	0
4	0	0	0	0	1
5	0	0	1	0	0
6	0	0	0	0	1
7	0	0	0	1	0
8	0	1	0	0	0
9	0	1	0	0	0
10	0	0	0	0	1
11	0	0	1	0	0
12	0	0	1	0	0
13	0	0	0	1	0
14	0	1	0	0	0
15	0	0	0	0	1
16	0	0	0	1	0
17	1	0	0	0	0
18	1	0	0	0	0
19	0	1	0	0	0
20	0	0	1	0	0
21	0	1	0	0	0
22	0	0	0	1	0
23	1	0	0	0	0
24	0	0	0	1	0
25	0	0	0	0	1

4.2.2 Problemas multiobjetivos

Nos problemas multiobjetivos os dados resultantes das funções foram analisados de maneira descritiva. As funções foram executadas gerando resultados os quais foram exportados para arquivos e analisados pela ferramenta RStudio empregando cálculos estatísticos especificamente: cálculo da média, mediana, variância, desvio padrão e intervalo de confiança a 95 % para os conjuntos de saídas resultantes. Para cada execução dos métodos: a) minimização do makespan versus minimização da soma ponderada dos

atrasos e adiantamentos (MULT-1); b) minimização da soma ponderada dos atrasos versus minimização da soma ponderada dos adiantamentos (MULT-2); foram obtidos 50 pontos ótimos, ou seja, 50 soluções para cada problema como pode ser observado na Tabela 7.

Tabela 7 – Resultados para os problemas multiobjetivos

	MULTI-1[a]		**MULTI-2[b]**	
1	48	521	0	143
2	131	122	0	143
3	150	112	167	0
4	104	165	21	91
5	98	182	167	0
6	107	158	167	0
7	119	134	167	0
8	147	113	80	37
9	131	122	162	1
10	119	134	80	37
11	150	112	162	1
12	150	112	80	37
13	142	115	65	47
14	73	311	21	91
15	137	118	0	143
16	119	134	162	1
17	150	112	0	143
18	104	165	10	113
19	142	115	0	143
20	131	122	162	1
21	98	182	167	0
22	142	115	21	91
23	48	521	10	113
24	91	212	167	0
25	48	521	167	0
26	150	112	0	143
27	150	112	0	143
28	63	381	167	0
29	119	134	21	91
30	63	381	167	0
31	131	122	167	0
32	150	112	167	0

	MULTI-1[a]		MULTI-2[b]	
33	101	173	80	37
34	150	112	162	1
35	119	134	80	37
36	98	182	162	1
37	142	115	80	37
38	150	112	65	47
39	137	118	21	91
40	48	521	0	143
41	119	134	162	1
42	82	257	0	143
43	98	182	10	113
44	48	521	0	143
45	150	112	162	1
46	150	112	167	0
47	119	134	21	91
48	150	112	10	113
49	150	112	167	0
50	137	118	167	0

[a] Soma ponderada dos atrasos e adiantamentos x Makespan

[b] Soma ponderada dos adiantamentos x soma ponderada dos atrasos

Note, conforme apresentado na Tabela 7, a agregação de pontos para o problema a minimização do makespan versus minimização da soma ponderada dos atrasos e adiantamentos (MULTI-1) e de 16 pontos, para o problema minimização da soma ponderada dos atrasos versus minimização da soma ponderada dos adiantamentos (MULTI-2) de 8 pontos. Observa-se também que os maiores números de dias concentraram-se no ponto [150,112] este com 13 frequências para o problema (MULTI-1). Contudo, estes valores não representam os pontos ótimos resultantes do problema. Os quais são os pontos mínimos (ótimos) de cada mono-objetivo, ou seja, os valores [48,122]. No problema (MULTI-2), os pontos [0,167] tem 7 frequências, no entanto, da mesma forma, não podemos considerar estes resultados apenas por quantidade de vezes que o ponto se relaciona na matriz, mais sim analisar os valores otimizados de cada objetivo. Assim, têm-se os pontos [0,0] como valores ótimos para o problema (MULTI-2).

A Tabela 8 evidencia os resultados estatísticos calculados a partir dos dados da Tabela 7.

Tabela 8 – Dados estatísticos para os problemas multiobjetivos

	MULTI-1		MULTI-2	
Média	117.1	186.9	88.2	55.04
Mediana	125	128	80	37
Variância	1126.139	16453.89	5333.306	3396.366
Desvio Padrão	33.558	128.2727	73.02949	58.27835
Intervalo de Confiança	107.5229	150.4453	67.44525	38.47748
	126.5971	223.3547	108.95475	71.60252
Mínimo	**48**	**112**	**0**	**0**
Máximo	150	521	167	143

Na tabela 8, pode-se observar as medidas de tendência central da amostra, onde temos a média, ou seja, o ponto de equilíbrio dos dados para os problemas mono-objetivos. Ao analisar a mediana observam-se os pontos centrais dos objetivos. Assim, ao analisar as medidas de dispersão, constata-se que no desvio padrão e variância os pontos estão bastantes dispersos, algo esperado, pois, um dos objetivos da otimização multiobjetivo é atingir um conjunto de soluções com a maior diversidade de valores possíveis (vide 2.1.4). Por fim, têm-se os valores máximos alcançados pelas amostras e os dados mínimos que correspondem aos resultados ótimos dos objetivos.

Os métodos multiobjetivos obtiveram diversos pontos na fronteira de Pareto, posteriormente foi possível descobrir as soluções ótimas de cada problema. No método da minimização do makespan versus minimização da soma ponderada dos atrasos e adiantamentos (MULTI-1) tem-se **16** pontos bem dispersos na fronteira de Pareto, conforme pode ser observado na Figura 8 e descrito também nas Tabelas 7 e 8. A solução ótima deste método é o ponto **[48,112]** apontado como solução para os problemas de otimização mono-objetivos *makespan* e soma ponderada dos atrasos e adiantamentos especificamente. Em relação ao método da minimização da soma ponderada dos atrasos versus minimização da soma ponderada dos adiantamentos (MULTI-2) constatou-se **8** diferentes pontos na fronteira, o quais podem ser observados na Figura 9. Portanto, para os problemas mono-objetivos da minimização da soma ponderada dos atrasos e minimização da soma ponderada dos adiantamentos tem-se como a solução ótima para este problema o ponto **[0,0]**, ou seja, a solução utópica para o problema.

Figura 8 – Fronteira de pareto: Soma ponderada dos atrasos e adiantamentos x Makespan

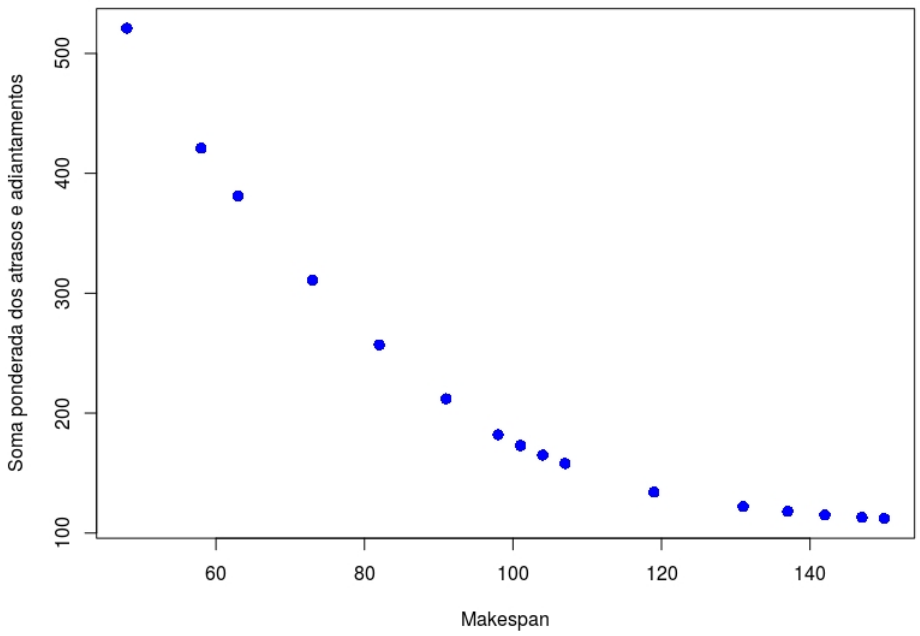
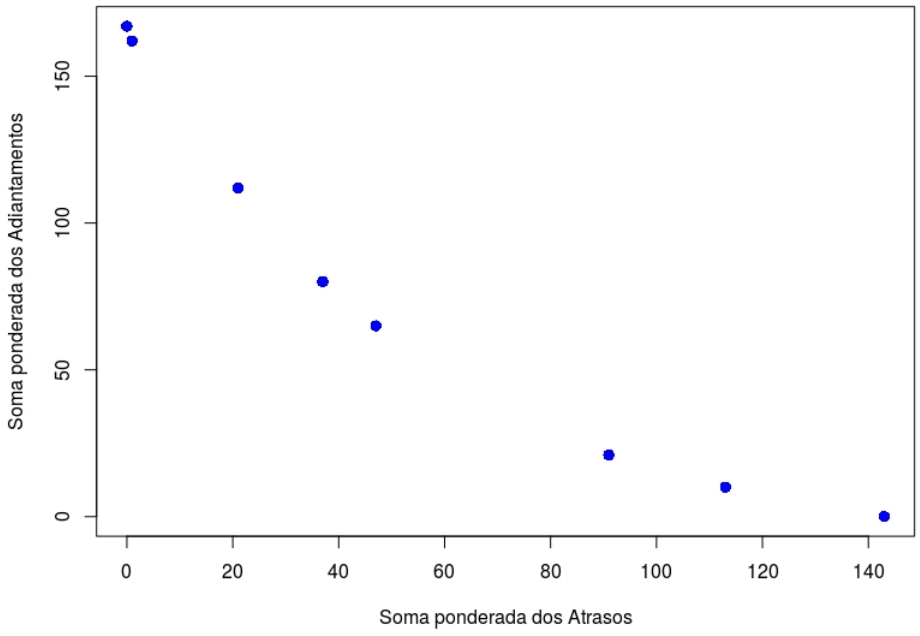


Figura 9 – Fronteira de pareto: Soma ponderada dos adiantamentos x Soma ponderada dos atrasos



5 Conclusão

Diante dos problemas enfrentados nas linhas de produção das empresas, os métodos computacionais de otimização tem suma importância. No caso dos problemas de máquinas paralelas não relacionadas pertencerem à classe de questões NP-díficil, inviabilizaram o encontro de apenas uma solução ótima sendo demonstrados um conjunto de soluções viáveis.

Entretanto, após o emprego da formulação matemática foco deste trabalho, as funções algorítmicas foram desenvolvidas para avaliar e testar a modelagem proposta. Em virtude do correto funcionamento dos métodos, a partir do uso das toolbox, acredita-se que a modelagem apresentada esteja em conformidade com o problema estudado.

Dessa maneira, para a construção dos algoritmos buscaram-se técnicas e métodos de programação linear e binária com baixo nível de complexidade e custos de processamento relativamente pequenos. Assim, o método *Branch-and-Bound* foi escolhido principalmente por suas estratégias de dividir e conquistar aliada ao conceito de considerar apenas resultados viáveis, economizando tempo computacional. Proporcionando a resolução do problema abordado neste trabalho.

Consequentemente, após o emprego das técnicas no problema de sequenciamento de operações das máquinas, teve-se por resultado os valores mínimos de 48 dias para o total de atrasos das tarefas (*makespan*) e para a soma ponderada de atrasos e adiantamentos o valor de 112 dias. No caso dos valores da fronteira de Pareto para os problemas multiobjetivos estes foram alcançados utilizando uma abordagem ponderada associada com a solução dos problemas mono-objetivos. Por consequência derivando 16 diferentes soluções para o problema da minimização do *makespan* versus minimização da soma ponderada dos atrasos e adiantamentos (MULTI-1) e 8 soluções para o problema da minimização da soma ponderada dos atrasos versus minimização da soma ponderada dos adiantamentos (MULT-2) na fronteira de Pareto.

Por fim, considera-se que este trabalho atingiu todas as metas da otimização multiobjetivo elencadas por NARIÑO, MARTHA e DE MENEZES (2014). Alçando conjuntos de soluções próximas da fronteira de Pareto e com maior diversidade de valores possíveis com o menor gasto de processamento computacional possível. Contudo, para melhorar os resultados obtidos e recomendado aperfeiçoar a formulação matemática e analisar outros métodos para o mesmo problema.

5.1 Trabalhos futuros

Como trabalhos futuros pretende-se:

- Testar outros métodos de programação linear como Simplex, Dual Simplex, de modo a avaliar a saídas de dados verificando a possibilidade de melhorias dos resultados.
- Realizar estudo da resolução das funções multiobjetivos deste trabalho empregando métodos de apoio a decisão multicritério. Buscando explorar as possibilidades dos problemas e verificar se os métodos de otimização proporcionam a melhor solução ou apenas soluções que atendam aos critérios estabelecidos para o problema ser resolvido.
- Avaliar o desempenho e viabilidade dos métodos de apoio a decisão, criando funções algorítmicas para posterior verificação dos resultados.

Referências

ALBUQUERQUE, M. C. et al. Heuristica de Programação Matemática para o Problema de Fluxo Multiproduto Binario. In: *Simpósio Brasileiro de Pesquisa Operacional*. [S.l.: s.n.], 2011. p. 2147–2156.

AMORIM, E. d. A. *Fluxo de potência ótimo em sistemas multimercados através de um algorítmo evolutivo multiobjetivo*. Tese (Doutorado) — Universidade Estadual Paulista, 2006.

AZUMA, R. M. et al. *Otimização multiobjetivo em problema de estoque e roteamento gerenciados pelo fornecedor*. 1–113 p. Tese (Mestrado) — Universidade Estadual de Campinas, 2011.

CORRAR, L. J. et al. *Pesquisa operacional para decisão em contabilidade e administração: contabilometria*. São Paulo: Atlas, 2004.

CORREIA, A. M. P. Monografia, *Análise de métodos escalares aplicados a problema multiobjetivos*. [S.l.]: Universidade Estadual do Tocantins, 2017. 1–63 p.

COUTINHO, B. C. *Proposta de Algoritmo Híbrido para o Problema de Tarefas em Ambientes Distribuídos Homogêneos*. Tese (Mestrado) — Universidade Federal do Espírito Santo, 2008.

COUTINHO, B. C.; DE OLIVEIRA, E. S. Fixação de variáveis do modelo matemático aplicado na solução do problema de escalonamento de tarefas em ambientes distribuídos homogêneos. *SBPO - Simpósio Brasileiro de Pesquisa Operacional*, 2009.

ETCHEVERRY, G. V. *Programação de tarefas em máquinas paralelas não-relacionadas com tempos de setup dependentes da sequência*. 1–54 p. Tese (Mestrado) — Universidade Federal do Rio Grande do Sul, 2012.

HADDAD, M. N. *Algoritmos heurísticos híbridos para o problema de sequenciamento em máquinas paralelas não-relacionadas com tempos de preparação dependentes da sequência*. 2012.

HILLIER, F. S.; LIEBERMAN, G. J. *Introduçao a Pesquisa Operacional*. 8. ed. São Paulo: McGraw-Hill Interamericana do Brasil Ltda, 2006. ISBN 85-86804-68-1.

_____. *Introdução à pesquisa operacional*. [S.l.: s.n.], 2013.

KAWAMURA, M. S. *Aplicação do método branch-and-bound na programação de tarefas em uma única máquina com data de entrega comum sob penalidades de adiantamento e atraso.* Tese (Tese de Doutorado) — Universidade de São Paulo, 2006.

MARTINS, R. C. *Análise de algoritmos evolucionários na resolução de funções de benchmark irrestritas.* [S.l.]: Universidade Estadual do Tocantins, 2017.

MINEIRO, A. *Aplicação de programação não-linear como ferramenta de auxílio à tomada de decisão na gestão de um clube de investimento*. 91 p. Tese (Mestrado) — UNIFEI, Itajubá, 2007.

MÜLLER, F. M.; DIAS, O. B. Algoritmo para o problema de seqüenciamento em máquinas paralelas não-relacionadas. *Revista Produção*, v. 12, n. 2, p. 6–17, 2002. ISSN 0103-6513.

NARIÑO, G. A. R. et al. *Otimização de risers em catenária com amortecedores hidrodinâmicos*. 18 p. Tese (Mestrado) — PUC-Rio, 2014.

PACHAMANOVA, D. A.; FABOZZI, F. J. *Simulation and Optimization in Finance: Modeling with MATLAB,@ RISK, or VBA*. [S.l.]: John Wiley & Sons, 2010. 787 p. ISBN 9780470371893.

QUEIROZ, M. M. de. *Métodos heurísticos aplicados ao problema de programação da frota de navios PLVs*. Tese (Doutorado) — Universidade de São Paulo, 2011.

REIS, D. C. et al. Problema da Alocação de Monitores de Qualidade de Energia Elétrica em Redes de Transmissão. *XVIII Congresso Brasileiro de Automática*, p. 1–6, 2010.

ROCHA, P. L. *Um problema de sequenciamento em máquinas paralelas não-relacionadas com tempos de preparação dependentes de máquina e da sequência:: modelos e algoritmos exato*. 1–70 p. Tese (Mestrado) — Universidade Federal de Minas Gerais, 2006.

SECCHI, A. R. *COQ-897 - Otimização de Processos*. Rio de Janeiro: [s.n.], 2015.

SERAFINI, L. et al. Heurística para minimização do atraso total de tarefas baseada em curvas de aprendizado e aspectos ergonômicos. *Revista Produção Online*, Florianópolis, p. 550–574, 2016.

SILVA, V. V. et al. Uma heurística híbrida para o problema de escalonamento de tarefas periódico em máquinas paralelas. *XXVIII Encontro Nacional de Engenharia de Produção*, p. 1–13, 2008.

XAVIER, E. C. et al. *Algoritmos de aproximação para problemas de escalonamento de tarefas em maquinas*. 1–148 p. Tese (Mestrado) — Unicamp, 2003.

YUNES, T. H. et al. *Problemas de escalonamento no transporte coletivo: programação por restrições e outras tecnicas*. Tese (Mestrado) — Unicamp, 2000.