Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Meta-Heurísticas Aplicadas ao Problema de Sequenciamento em Uma Máquina com Penalidade por Antecipação e Atraso da Produção

Bruno Lacerda Pêgo

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:

Prof. Samuel Souza Brito

COORIENTAÇÃO:

Prof. Fernando Bernardes de Oliveira

Dezembro, 2019
João Monlevade–MG

Bruno Lacerda Pêgo

Meta-Heurísticas Aplicadas ao Problema de Sequenciamento em Uma Máquina com Penalidade por Antecipação e Atraso da Produção

Orientador: Prof. Samuel Souza Brito

Coorientador: Prof. Fernando Bernardes de Oliveira

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Dezembro de 2019

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

P376m Pêgo, Bruno Lacerda.
Meta-heurísticas aplicadas ao problema de sequenciamento em uma máquina com penalidade por antecipação e atraso da produção. [manuscrito] / Bruno Lacerda Pêgo. - 2019.
56 f.: il.: color., gráf., tab..

Orientador: Prof. Me. Samuel Souza Brito.
Coorientador: Prof. Dr. Fernando Bernardes de Oliveira.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Sistemas de Informação .

1. Programação heurística. 2. Algoritmos. 3. GRASP (Sistema operacional de computador) . 4. Programação evolutiva (Computação) . I. Brito, Samuel Souza. II. Oliveira, Fernando Bernardes de. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 004.023

Bibliotecário(a) Responsável: Flavia Cristina Miguel Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Bruno Lacerda Pêgo

Meta-heurísticas Aplicadas ao Problema de Sequenciamento em uma Máquina com Penalidade por Antecipação e Atraso da Produção

Membros da banca

Samuel Souza Brito - Mestre - DECSI (UFOP)
Fernando Bernardes de Oliveira - Doutor - DECSI (UFOP)
George Henrique Godim da Fonseca - Doutor - DECSI (UFOP)
Rafael Frederico Alexandre - Doutor - DECSI (UFOP)

Versão final
Aprovado em 17 de Dezembro de 2019

De acordo

Samuel Souza Brito

Documento assinado eletronicamente por **Samuel Souza Brito**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 07/03/2020, às 16:56, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0041699** e o código CRC **C0E41704**.

Referência: Caso responda este documento, indicar expressamente o Processo nº 23109.002147/2020-83 SEI nº 0041699

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Dedico este trabalho aos meus pais e à minha irmã.

Agradecimentos

Agradeço aos meus pais pelo suporte, incentivo, educação e todo esforço que aplicaram em mim, sem vocês nada disso seria possível.

À minha irmã Letícia, por todos os momentos e por sempre confiar em mim como irmão mais velho.

Aos meus amigos da UFOP pelos momentos juntos que sem dúvidas contribuíram para minha chegada até aqui, em especial Bárbara, Raul e amigos da "salinha", pelo companheirismo, ajudas, conselhos, conversas, e todo crescimento pessoal que pude ter ao lado de vocês.

Aos meus amigos de república, em especial Vinícius, pela parceria e amizade ao longo da graduação e pelos diversos momentos de alegria.

Ao Pré-Vestibular: Rumo à Universidade, projeto que pude participar por quase 3 anos, por me ensinar que pequenos gestos e esforços podem transformar a vida de muitas pessoas.

À Universidade Federal de Ouro Preto pelo ensino público, gratuito e de qualidade e pelos valores transmitidos.

A todos os professores que me guiaram nessa jornada, pelo ensino e experiência de vida.

Por fim, agradeço ao meu orientador Samuel Brito e ao meu coorientador Fernando Oliveira, por todo aprendizado e por me guiarem com toda disposição para a conclusão deste trabalho.

"Nascemos e morremos neste mundo sem trazer e nem levar nada, mas cabe a nós decidirmos o que vamos deixar."

— Autor Desconhecido

Resumo

Este trabalho trata do problema de sequenciamento de tarefas em uma máquina com tempo de preparação dependente da sequência de produção, onde cada tarefa possui uma janela de entrega na qual deve ser preferencialmente concluída. O objetivo é minimizar a soma das penalidades por atraso e antecipação da produção, determinando a sequência de execução e a data de início de processamento das tarefas. É proposto um algoritmo de busca populacional baseado em computação evolutiva de duas etapas, denominado GEVITIA. A primeira etapa consiste da construção da população inicial baseada em GRASP, enquanto a segunda combina os procedimentos de Estratégia Evolutiva, VND e um algoritmo para determinar a data ótima de início de processamento (ITIA). O algoritmo proposto se mostrou competitivo quando comparado com os trabalhos presentes na literatura, sendo capaz de gerar soluções de qualidade semelhante e, em alguns casos, superiores.

Palavras-chaves: Problema de sequenciamento de tarefas. Sequenciamento em uma máquina. GRASP. Estratégia Evolutiva. VND. ITIA.

Abstract

This work addresses the single machine scheduling problem with sequence-dependent setup times, where each job has a distinct time window within which it should preferably be completed. The goal is to minimize the value of penalties for tardiness and earliness by determining the execution sequence and the time to start processing the jobs. A population search algorithm based on two-step evolutionary computation called GEVITIA is proposed. The first step is the initial population construction phase based on GRASP, while the second step combines Evolution Strategy, VND and an algorithm for determining the optimal time for completion of each job in a given sequence (ITIA). The proposed algorithm proved to be competitive when compared to the other approaches in the literature, being able to generate solutions of similar and, in some cases, superior quality.

Key-words: Job Sequencing Problem. Single Machine Scheduling Problem. GRASP. Evolution Strategy. VND. ITIA.

Lista de Algoritmos

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADDOIP Algoritmo de Determinação das Datas Ótimas de Início de Processamento das tarefas

AED Algoritmo de Evolução Diferencial

AG Algoritmos Genéticos

ANOVA Análise de Variância

BB *Branch and Bound*

BT Busca Tabu

E/T Problem *Earliness and Tardiness Problem*

EDD *Earliest Due Date*

EE Estratégia Evolutiva

EST *Earliest Starting Time*

GRASP *Greedy Randomized Adaptive Search Procedure*

ILS *Iterated Local Search*

ITIA *Idle Time Insertion Algorithm*

JIT *Just in Time*

MD Método de Descida

MRD Método Randômico de Descida

PG Programação Genética

PLIM Programação Linear Inteira Mista

PSUMAA Problema de Sequenciamento de Tarefas em uma Máquina com Penalidades por Atraso e Antecipação da Produção

SPT *Shortest Processing Time*

TDD *Tardiest Due Date*

VND *Variable Neighborhood Descent*

Lista de símbolos

α	Alfa (Penalidade por antecipação)
β	Beta (Penalidade por atraso)
$\emptyset$	Conjunto Vazio
$\setminus$	Exclusão
Γ	Gama Maiúsculo (Conjunto de números entre 0 e 1)
γ	Gama Minúsculo (Valor aleatório selecionado em Γ)
λ	Lambda (Quantidade de indivíduos filhos)
μ	Mu (Quantidade de indivíduos pais)
$\notin$	Não Pertence
$\in$	Pertence
φ	Phi (Quantidade de unidades de deslocamento)
$\cup$	União

Sumário

1 Introdução

A revolução industrial promoveu um impacto nas cadeias de produção, aumentando a capacidade de manufatura e, consequentemente, gerando uma maior capacidade de entrega. Assim, houve uma expansão no atendimento das demandas da sociedade em processo de industrialização. Novos desafios começaram a surgir com a crescente demanda e competição em grande escala entre os setores produtivos, aumentando a necessidade de planejar e controlar a produção (BUSTAMANTE, 2006).

A produção de bens de consumo é um ciclo interminável, visto que frequentemente novas indústrias são estabelecidas para atender às mais diversas necessidades existentes. Com o crescente desenvolvimento destes setores, muitas empresas passaram a adotar o sistema de produção *Just in Time* (JIT), visando a redução de custos e de desperdícios de recursos, bem como a melhoria do processo produtivo.

O sistema JIT teve início em meados da década de 70 e destaca a importância de se planejar a produção. Sua filosofia consiste em planejar a produção de forma que não haja antecipação nem atrasos da manufatura e entrega dos bens. A produção com antecedência pode gerar demanda de capital, espaço para armazenar os bens produzidos e recursos para gerir e manter o estoque. Em contrapartida, o atraso das entregas pode acarretar em multas, rescisões contratuais e perda de credibilidade.

Neste contexto, a programação da produção se torna crucial para manutenção das indústrias no mercado competitivo. Dentre os cenários encontrados na literatura, Baker e Scudder (1990) destacam que o Problema de Sequenciamento de Tarefas reflete bem os ambientes de produção administrados sob o sistema *Just in Time*.

O Problema de Sequenciamento de Tarefas consiste em sequenciar a ordem de execução da produção, determinando o momento exato em que cada tarefa deve ser executada. Trata-se de um problema da classe NP-difícil (BAKER; SCUDDER, 1990). Na literatura é possível encontrar métodos exatos, heurísticos e populacionais para resolução desse problema. Contudo, os métodos exatos se limitam a problemas pequenos, aquém das necessidades reais, uma vez que tais métodos requerem um extenso tempo computacional para seu processamento. Devido à essa limitação, métodos heurísticos e populacionais são frequentemente utilizados, uma vez que são capazes de gerar boas soluções em tempos computacionais viáveis. Entretanto, tais métodos não garantem a obtenção de soluções ótimas.

Este trabalho propõe o estudo acerca do problema de programação da produção, mais especificamente o Problema de Sequenciamento de Tarefas em uma Máquina com Penalidades por Atraso e Antecipação da Produção (PSUMAA), aplicando a técnica de

Estratégia Evolutiva (EE) com o intuito de obter soluções factíveis e de qualidade em tempos computacionais restritos. O algoritmo implementado consiste em duas etapas. Na primeira etapa, um procedimento baseado na fase construtiva do algoritmo *Greedy Randomized Adaptive Search Procedure* (GRASP) é utilizado para a gerar a população inicial. A etapa seguinte consiste na execução iterativa da EE, que utiliza uma combinação dos procedimentos de *Variable Neighborhood Descent* (VND) e *Idle Time Insertion Algorithm* (ITIA) para refinar as soluções. Os resultados obtidos nos experimentos computacionais mostram que o algoritmo implementado é capaz de obter soluções de qualidade semelhante ou superior àquelas descritas na literatura.

1.1 Objetivos

Nesta seção são apresentados o objetivo geral e específicos, e como foi feita a organização escrita do trabalho.

O objetivo geral deste trabalho é desenvolver um algoritmo capaz de produzir soluções de qualidade para o PSUMAA em tempos computacionais restritos.

Com a finalidade de atingir os objetivos gerais, os seguintes objetivos específicos foram definidos:

1. Revisar a literatura sobre problemas de sequenciamento de tarefas, identificando os diversos métodos utilizados para sua resolução;
2. Estudar e desenvolver um algoritmo baseado em Estratégia Evolutiva para obtenção de soluções factíveis e de qualidade;
3. Aplicar um algoritmo de busca local como método de refinamento durante o processo evolutivo;
4. Implementar o algoritmo de inserção de tempo ocioso de máquina para definição da data ótima de início de processamento das tarefas do problema;
5. Executar experimentos em problemas-teste disponíveis na literatura;
6. Analisar e discutir os resultados obtidos comparando-os com as principais abordagens presentes na literatura.

1.2 Organização do Trabalho

Este trabalho está organizado como segue. O Capítulo 2 apresenta o Problema de Sequenciamento de Tarefas e suas variações, com ênfase no sequenciamento em uma máquina. Este capítulo também descreve detalhadamente as características e restrições do

problema abordado. Uma revisão bibliográfica dos trabalhos relacionados, metodologias aplicadas e resultados obtidos são apresentados no Capítulo 3. Este capítulo demonstra as diversas aplicabilidades do problema e os meios utilizados para sua resolução, utilizando-se de heurísticas, métodos exatos e populacionais. O Capítulo 4 visa a apresentar a representação do problema, estruturas de vizinhança e operadores de mutação, bem como a implementação dos algoritmos propostos. Os resultados obtidos dos experimentos computacionais são apresentados no Capítulo 5. Por fim, o Capítulo 6 apresenta as considerações finais sobre o estudo e os resultados obtidos, além de propor trabalhos futuros sobre o objeto de estudo.

2 Problema Abordado

No contexto industrial ou até mesmo de manufatura de bens, programar a produção consiste em definir em um determinado período de tempo a ordem de execução das tarefas. A programação é um processo de tomada de decisão que, em alguns casos, se constitui como um fator importante para o aumento de competitividade, haja visto que uma boa definição dos processos e ordem de produção podem causar impactos financeiros entre outros. Estes impactos podem ser um fator crucial para o crescimento da empresa, aumentando sua capacidade competitiva no mercado. Definir uma sequência lógica de produção respeitando os critérios e restrições, como prazos e características dos produtos para atendimento de uma demanda caracteriza-se como um Problema de Programação da Produção (BUSTAMANTE, 2006).

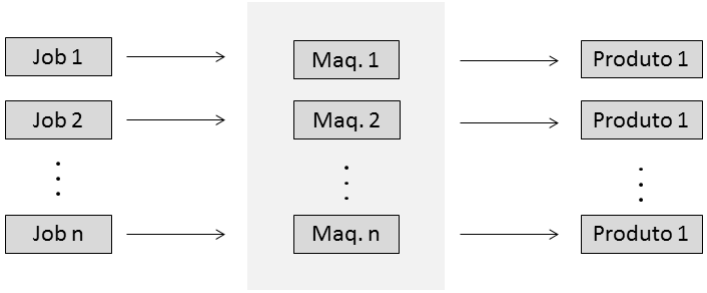
A aplicação prática desta classe de problemas incide nos mais variados tipo industriais, uma vez que o intuito é programar a produção sequenciando a ordem de execução das tarefas dentro do período estabelecido. Dentre esses setores, podem ser citados indústrias têxteis, de tintas, celulose, alimentícia, entre outras. Casos de estudos práticos de sequenciamento e planejamento da produção são encontrados frequentemente na literatura, como os trabalhos de Bustamante (2006), Rodrigues (2012) e Penna (2009). O primeiro trabalho considerou a produção de uma indústria metalúrgica, enquanto o segundo utilizou dados de uma indústria bélica de uma organização militar no qual o principal produto fabricado é o morteiro pesado. O último envolve a aplicação do planejamento de produção em usinas de mineração com minas subterrâneas ou a céu aberto.

Dentro da classe de Problema de Programação da Produção está contido o Problema de Sequenciamento de Tarefas em Máquinas. Com o objetivo de programar a execução das tarefas necessárias, no estudo proposto por Allahverdi et al. (2008) os autores apresentam os principais tipos de problema de sequenciamento, classificando-os como, Sequenciamento em uma máquina, o problema de Máquinas paralelas, *Flow Shop* e *Job Shop*.

Abaixo seguem os exemplos dos tipos de problema encontrados na literatura. Como o foco deste trabalho é o Sequenciamento de Tarefas em uma Máquina, o mesmo terá maior enfoque na seção 2.1 e seção 2.2.

1. **Máquinas Paralelas:** Neste cenário, um conjunto m de máquinas em paralelo podem ser utilizadas na execução de uma tarefa (Figura 1). As máquinas são consideradas idênticas quando possuem a mesma velocidade ou diferentes quando suas velocidades variam. Nesse último caso, as máquinas ainda podem ser consideradas não-uniformes ou uniformes, quando a velocidade depende ou não da tarefa que será executada, respectivamente.

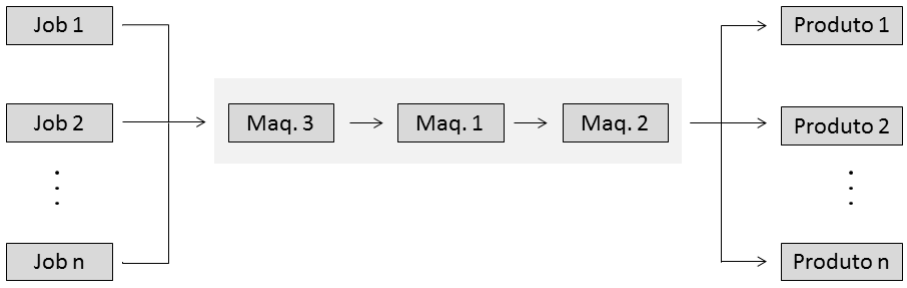
Figura 1 – Exemplo do problema de Máquinas Paralelas.



Fonte: figura elaborada pelo autor (Adaptado de Bustamante (2006))

2. ***Flow Shop:*** Quando um conjunto de tarefas devem ser executados em um conjunto m de máquinas, obedecendo a mesma sequência de produção (Figura 2). Para cada tarefa a ser executada o tempo de processamento demandado nas máquinas pode ser diferente.

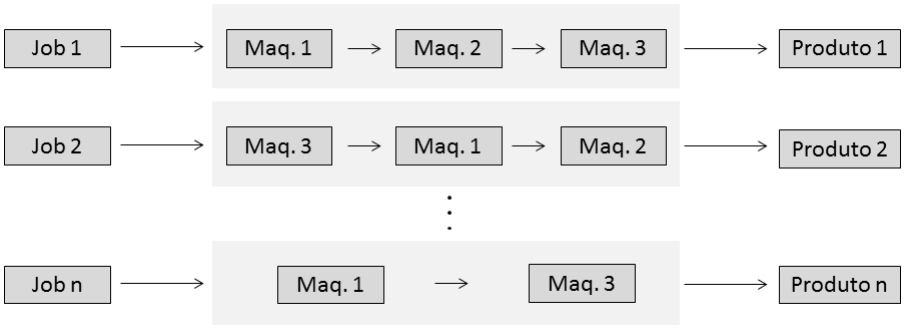
Figura 2 – Exemplo do problema de *Flow Shop*.



Fonte: figura elaborada pelo autor (Adaptado de Bustamante (2006))

3. ***Job Shop:*** Trata-se de um caso mais genérico do *flow shop*, pois os modelos desenvolvidos para resolver problemas de *job shop* também resolvem os de *flow shop* (BUSTAMANTE, 2006). Neste contexto, cada tarefa pode possuir uma sequência lógica de produção, ou seja, para um conjunto m de máquinas, as tarefas podem ser executadas em máquinas distintas (Figura 3).

Figura 3 – Exemplo do problema de *Job Shop*.

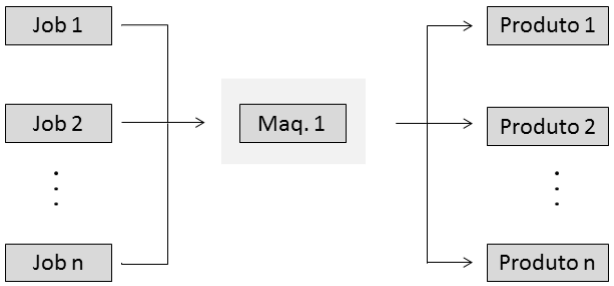


Fonte: figura elaborada pelo autor (Adaptado de Bustamante (2006))

2.1 Problema de Sequenciamento de Tarefas em Uma Máquina

Este modelo considera que apenas uma máquina irá executar todas as tarefas (Figura 4). Um conjunto de máquinas que são dependentes umas das outras também podem ser representadas por este modelo (JÚNIOR, 2007). Um exemplo prático aplicado pode ser encontrado no trabalho de Bustamante (2006), onde um conjunto de laminadores que executam uma sequência de tarefas são considerados uma mesma máquina.

Figura 4 – Exemplo do problema de Uma Máquina.



Fonte: figura elaborada pelo autor (Adaptado de Bustamante (2006))

Programar essa sequência de tarefas sob o conceito da filosofia JIT é um fator crucial, tendo em vista que essa filosofia desencoraja tanto o atraso quanto a antecipação da produção, uma vez que podem surgir custos para manter estoque caso a tarefa seja executada com antecedência ou problemas contratuais e multas devido à um produção tardia (JÚNIOR, 2007). Portanto, sob este modelo, uma solução ideal é aquela na qual a produção terminará na data exata de sua necessidade. Segundo Bustamante (2006),

os problemas que visam minimizar os custos de atraso e antecipação da produção são conhecidos como *Earliness and Tardiness Problem* (E/T Problem)

Acerca da data desejada para a entrega de um ou mais produtos Bustamante (2006) identifica no trabalho de Baker e Scudder (1990) as seguintes classificações:

- Datas de Entregas Comuns (*Common Due Date*): quando a data de entrega é a mesma para todas as tarefas ou produtos;
- Datas de Entregas Distintas (*Distinct Due Date*): para cada tarefa existe uma data de entrega distinta associada;
- Janelas de Entregas Distintas (*Distinct Due Windows*): quando há tolerância para a finalização das tarefas. Neste caso, uma tarefa pode terminar em um determinado intervalo de tempo, não havendo custos quando a finalização ocorre dentro deste período.

Em relação aos custos provenientes de atrasos e antecipações, Bustamante (2006) e Júnior (2007) citam três possíveis classificações:

- Problemas com Custos Diferentes: quando as penalidades de atraso diferem das penalidades de antecipação;
- Problemas com Custos Dependentes: quando as penalidades por atraso e antecipação dependem da tarefa que está sendo executada, a penalidade por antecipação pode ser diferente da penalidade por atraso;
- Problemas com Custos Iguais: quando as penalidades por atraso e antecipação são iguais para todas as tarefas.

Neste trabalho, o objeto de estudo trata do Problema de Sequenciamento de Tarefas em Uma Máquina, com penalidades por atraso e antecipação das tarefas, janelas de entregas distintas e custos dependentes da sequência de produção. Para fim de simplificação no decorrer deste trabalho o problema citado será denominado somente como PSUMAA.

2.2 Características Específicas do PSUMAA

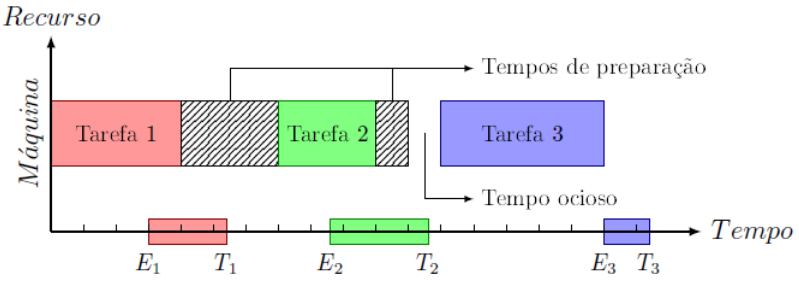
O PSUMAA possui características que devem ser consideradas para resolução do problema:

- Todas as tarefas estão disponíveis para iniciar seu processamento na data 0;
- Uma máquina deve processar um conjunto I de n tarefas;

- Para cada tarefa i está associado um tempo de processamento P_i que deve ser preferencialmente concluída dentro de uma janela de tempo $[E_i, T_i]$, onde E_i representa a data inicial desejada de entrega e T_i a data final de entrega para a tarefa i;
- Caso a tarefa i seja finalizada antes da data E_i, há um custo de α_i por unidade de tempo de antecipação;
- Caso a tarefa i seja finalizada após da data T_i, há um custo de β_i por unidade de tempo de antecipação;
- Se a tarefa for concluída dentro da janela de tempo $[E_i, T_i]$, nenhuma penalidade será aplicada;
- A máquina só pode processar por vez uma única tarefa, e, uma vez iniciado seu processamento, não é permitida a preempção da mesma;
- É permitido tempo ocioso entre o processamento de duas tarefas consecutivas;
- Entre duas tarefas i e $j \in I$ consecutivas é necessário um tempo S_{ij} de preparação da máquina, chamado tempo de *setup*. Assume-se que o tempo de preparação da máquina para o processamento da primeira tarefa na sequência é igual a 0;

A Figura 5 elaborada por Penna (2009) exemplifica uma possível solução para o PSUMAA com janelas de entregas distintas e tempo de preparação de máquina dependentes da sequência de produção.

Figura 5 – Exemplo do problema estudado.



Fonte: Penna (2009)

Neste exemplo a Tarefa 1 tem seu início na data 0 e término dentro da janela especificada. Entre a Tarefa 1 e a Tarefa 2 há o tempo de *setup* e na sequência ocorre o início da execução da Tarefa 2. Para a Tarefa 3 também há o tempo de *setup*, porém caso a tarefa fosse executada imediatamente após o tempo de *setup* entre a Tarefa 2 e 3, a mesma seria concluída antes da janela de conclusão atribuída a ela, gerando assim custos por unidade de tempo antecipadas.

2.3 Considerações Gerais

Determinar a ordem de execução das tarefas para gerar um baixo custo de produção não é trivial, uma vez que tal problema possui alta complexidade e restrições a serem cumpridas. Por ser considerado um problema da classe NP-difícil (BAKER; SCUDDER, 1990), a utilização de estratégias de solução exatas são computacionalmente inviáveis para problemas de um cenário real e em grande escala, como a industrial (BUSTAMANTE, 2006).

Neste trabalho são aplicadas técnicas baseadas em computação evolucionária a fim de se obter boas soluções para o problema abordado. Também é implementado um algoritmo de inserção de tempo ocioso para determinação da data ótima do início de processamento das tarefas a serem executadas.

3 Revisão bibliográfica

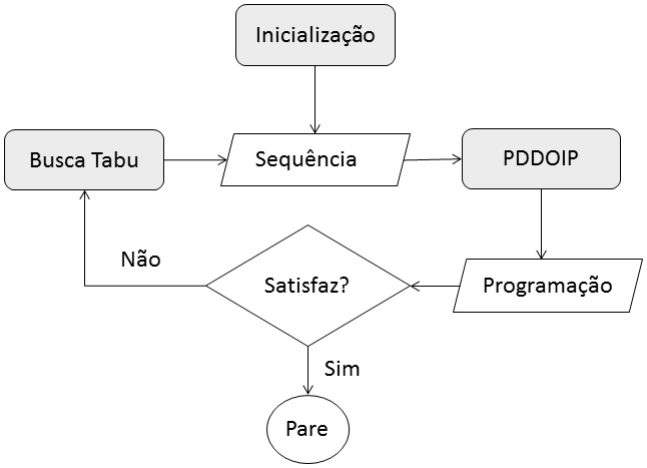
Com o objetivo de resolver essa classe de problemas, muitos trabalhos são desenvolvidos na tentativa de obter resultados de boa qualidade. Devido à sua complexidade e aplicabilidade tem-se grande motivação para resolução do PSUMAA em suas diversas variantes.

Lee e Choi (1995) em seu trabalho utilizaram Algoritmos Genéticos (AG) aplicados ao PSUMAA com datas de entrega distintas. A busca pela solução foi dividida em duas partes: determinar a melhor data de processamento das tarefas e determinar a ordem de processamentos das mesmas. Para determinar a melhor data de processamento das tarefas é proposto um algoritmo determinístico com complexidade polinomial. Os autores utilizaram as técnicas *Earliest Due Date* (EDD) e *Earliest Starting Time* (EST) para geração inicial da população e, em seguida, aplicaram o algoritmo para determinar a data ótima de processamento das tarefas. Como método de reprodução na geração de novos indivíduos foram utilizados os operadores de *crossover* e mutação. Neste estudo foram resolvidos problemas teste com até 80 tarefas. O AG proposto conseguiu obter soluções ótimas para problemas com até 20 tarefas nos quais os resultados foram verificados através de um algoritmo *Branch and Bound* (BB).

Wan e Yen (2002) abordaram o PSUMAA com janelas de entrega distintas e tempo de *setup* independentes da sequência de produção. Neste trabalho, os autores propuseram um procedimento de determinação de data ótima de conclusão das tarefas, baseando-se no algoritmo proposto por Lee e Choi (1995), permitindo a ociosidade das máquinas. Também é apresentada uma formulação matemática para o problema e proposta uma combinação entre os algoritmos de Busca Tabu (BT) e o Algoritmo de Determinação das Datas Ótimas de Início de Processamento das tarefas (ADDOIP) citado acima representado pela Figura 6. Foram gerados aleatoriamente instâncias com 15, 20, 30, 50 e 80 tarefas para aplicação do algoritmo de BT e ADDOIP. Os resultados foram comparados com os obtidos através do método *Branch and Bound* (BB) para efeito de validação. O algoritmo dos autores obtiveram soluções ótimas em mais de 20% das instâncias com 15 e 20 tarefas, os valores foram validados com os obtidos através do método exato aplicado.

Em Bustamante (2006) foram desenvolvidos modelos de Programação Linear Inteira Mista (PLIM) para resolver o PSUMAA com tempo de preparação de máquina dependentes da sequência de produção. O trabalho consistiu na resolução de um problema real que era sequenciar tarefas de um conjunto de laminadores de uma indústria siderúrgica. Devido a técnica utilizada de métodos exatos e seu custo computacional, a aplicação da modelagem proposta se restringe a problemas em escala reduzida, tendo em vista que

Figura 6 – Procedimento de Busca Tabu.



Fonte: figura elaborada pelo autor (Adpatado de Wan e Yen (2002))

obter resultado ótimos para problemas de grande escala demandaria um tempo inviável. Tendo em vista essa limitação os métodos propostos pelo autor foram capazes de resolver em sua otimalidade problemas com até 10 tarefas.

Em seu trabalho, Júnior (2007) propõe um modelo baseado em PLIM para representar o problema. O foco do trabalho foi a resolução do PSUMAA com tempo de preparação de máquina dependentes da sequência e janelas de entrega distintas. Também são propostos métodos heurísticos baseado nas meta-heurísticas *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Iterated Local Search* (ILS). Foram utilizados três métodos de busca local, Método Randômico de Descida (MRD), Método de Descida (MD) e *Variable Neighborhood Descent* (VND). Todos eles utilizaram duas estruturas de vizinhança: realocação e troca de tarefas. O autor desenvolveu também o Algoritmo de Determinação das Datas Ótimas de Início de Processamento das tarefas (ADDOIP) que possibilitou encontrar soluções ótimas na maioria dos problemas de pequenas dimensões. Os algoritmos propostos foram aplicados em problemas testes de 6 até 75 tarefas, gerados através de métodos pseudo-aleatórios. Para problemas pequenos de 8 a 12 tarefas, os métodos conseguiram quase sempre obter resultados ótimos e nos problemas de dimensões maiores de 15 a 75 tarefas obtiveram pequenos desvios em relação a melhor solução encontrada.

No trabalho de Penna (2009) foi proposto um algoritmo heurístico de três fases. A primeira fase é a de geração da solução inicial, onde foram utilizados GRASP e VND. A segunda fase é baseada em Busca Tabu para refinamento, enquanto a terceira fase utiliza Reconexão por Caminhos como estratégia de pós-otimização. Para cada solução gerada foi utilizado o ADDOIP. O autor também propõe um modelo de programação matemática para resolver na otimalidade problemas de pequenas dimensões. Os problemas

teste utilizados foram os mesmos propostos por Júnior (2007). O modelo matemático apresentado foi capaz de obter em todos os testes soluções ótimas para os problemas de 6 até 10 tarefas. Com o algoritmo proposto foi possível superar os resultados da literatura em até 5,57%.

Rosa (2009) trata do PSUMAA com tempo de preparação de máquina dependentes da sequência e janelas de entrega. O autor propõe duas novas formulações matemáticas para o problema, sendo uma delas um aperfeiçoamento do modelo proposto por Júnior (2007). Além das formulações matemáticas foi proposto um algoritmo heurístico composto de duas fases, sendo a primeira fase a geração da solução inicial com base nas meta-heurísticas GRASP, Princípio da Otimalidade Próxima e VND. A segunda fase realiza um pós-refinanamento aplicando novamente o VND. O algoritmo foi denominado GPV. Também foi implementado o ADDOIP proposto por Júnior (2007). Na resolução do problema foram utilizados os problemas testes propostos por Júnior (2007) e mais outras duas bases de dados. Em questão de tempo computacional o algoritmo GPV demandou um tempo médio muito inferior quando comparado com Penna (2009) e Ribeiro (2009) sendo, respectivamente, 800% e 360% menor o tempo médio necessário para execução do GPV. Contudo a qualidade das soluções encontradas não foram superiores aos resultados de Penna (2009) e Ribeiro (2009) .

Gonçalves (2010) propôs um algoritmo denominado GTSPR-M que combina os procedimentos heurísticos GRASP, VND, Busca Tabu e Reconexão por Caminhos. Foi explorado neste trabalho o conceito de processamento paralelo, aplicando GRASP na primeira fase para gerar a solução inicial e distribuindo o processamento entre várias *threads*. A melhor solução encontrada entre as *threads* é utilizada como solução inicial para o algoritmo. Na segunda fase é feito um refinamento pelo procedimento de Busca Tabu, também executada de forma paralela. Como técnica de pós-otimização, o autor utilizou o procedimento de Reconexão por Caminhos. A solução proposta através da exploração de processamento paralelo se mostrou superior ao processamento sequencial no quesito de tempo médio de processamento gerando todas as soluções ótimas para problemas de até 12 tarefas e comparado com Rosa (2009) o algoritmo GTSPR-M demandou menos tempo para gerar boas soluções em problemas-teste de até 50 tarefas.

Ramos e Oliveira (2010) desenvolveram um algoritmo evolutivo híbrido aplicado ao problema de sequenciamento em uma máquina com penalidades por atraso e antecipação da produção. Em comparação ao algoritmo genético clássico, houve melhora nos resultados encontrados pelo algoritmo evolutivo híbrido desenvolvido, atingindo melhora superior a 74% para alguns casos de teste. Em relação à literatura, os resultados obtidos apresentaram melhora para alguns casos de teste com 20 e 25 tarefas entre 1% e 26%, considerando 10.000 gerações para o algoritmo evolutivo.

No trabalho de Alves (2016) foi aplicado um Algoritmo de Evolução Diferencial

(AED) ao PSUMAA com tempo de preparação dependente da sequência de produção e janela de entrega. A solução inicial foi gerada de modo aleatório. Para a geração de novos indivíduos foram utilizados os operadores de mutação e o cruzamento. O método Primeiro de Melhora foi implementado como busca local na intenção de obter melhores resultados. Comparando os resultados obtidos com os da literatura, o AED encontrou soluções próximas ao ótimo para instâncias menores, enquanto para instâncias maiores os valores obtidos ficaram distantes. De acordo com o autor, esse resultado é devido ao fato de que os trabalhos comparados apresentavam a implementação do ADDOIP.

4 Metodologia e Algoritmo Proposto

Neste capítulo serão apresentados todos os conceitos e métodos utilizados para a resolução do problema de estudo. A seção 4.1 apresenta a forma de representação de uma solução. Na seção 4.2 são apresentados todos os operadores de mutação e vizinhança, tais operadores serão utilizados posteriormente na fase de evolução e refinamento das soluções. Um algoritmo de duas fases denominado GEVITIA é proposto e implementado. Este algoritmo encapsula a geração da população inicial, que será detalhada na subseção 4.4.2, o processo evolutivo na subseção 4.4.3 e a implementação do algoritmo de inserção de tempos ociosos (ITIA) na subseção 4.4.4.

4.1 Representação de uma Solução

Uma solução para o Problema de Sequenciamento de Tarefas em uma Máquina com Penalidades por Atraso e Antecipação da Produção pode ser representado por um vetor v de n posições, onde cada elemento v_i indica a ordem de produção da i-ésima tarefa. Sendo assim, para uma sequência de 5 tarefas $v = \{2, 3, 1, 4, 5\}$ a tarefa 2 é a primeira a ser executada, a tarefa 3 a segunda, e assim por diante. Cada tarefa possui um conjunto de características que são descritos na Tabela 1.

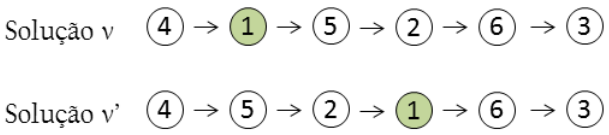
4.2 Vizinhança e Operadores de Mutação

Algoritmos de Estratégia Evolutiva (EE) usualmente utilizam a mutação como operador para geração de novos indivíduos a partir de uma população já existente (LUKE, 2013). Para gerar novos indivíduos foram implementados seis operadores de mutação. Devido às suas características, esses operadores também serviram como movimentos para exploração do espaço de soluções no algoritmo de busca local implementado neste trabalho.

Os tipos de movimentos implementados foram:

N^R - Este movimento consiste em realocar uma determinada tarefa para depois de outra, como no exemplo da Figura 7, onde a tarefa 1 foi realocada para depois da tarefa 2.

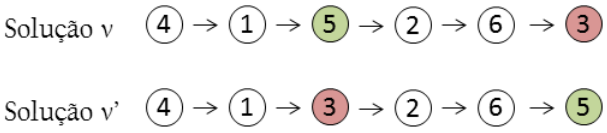
Figura 7 – Movimento de Realocação - N^R.



Fonte: figura elaborada pelo autor

N^T - Esta estrutura de vizinhança consiste em trocar a ordem de processamento de duas tarefas. Por exemplo, a solução $v' = \{4, 1, 3, 2, 6, 5\}$ é vizinha de $v = \{4, 1, 5, 2, 6, 3\}$ segundo a vizinhança N^T, como demonstra a Figura 8.

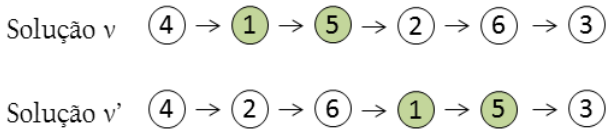
Figura 8 – Movimento de Troca - N^T.



Fonte: figura elaborada pelo autor

N^{RB} - Similar à estrutura N^R, neste modelo um bloco composto por duas tarefas são realocados para depois de uma determinada tarefa. Como demonstrado na Figura 9 onde o bloco $B = \{1, 5\}$ foi realocado para depois da tarefa 6.

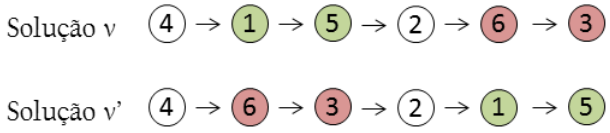
Figura 9 – Movimento de Realocação de Bloco - N^{RB}.



Fonte: figura elaborada pelo autor

N^{TB} - Este movimento realiza a troca entre dois blocos, onde cada bloco é composto por duas tarefas, como mostra o exemplo da Figura 10.

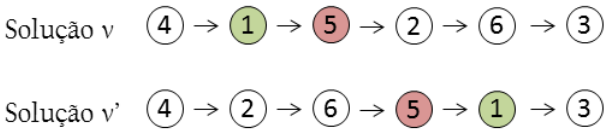
Figura 10 – Movimento de Troca de dois Blocos - N^{TB}.



Fonte: figura elaborada pelo autor

N^{TRB} - Nesta estrutura de vizinhança ocorre a troca da ordem de processamento de duas tarefas num bloco e a realocação deste bloco na sequência de produção. Para o exemplo da Figura 11, primeiramente foi selecionado o bloco $B = \{1, 5\}$, após a seleção é feita a troca das tarefas dentro do bloco, resultando em $B' = \{5, 1\}$. Por fim o bloco B' é realocado para depois da tarefa 6.

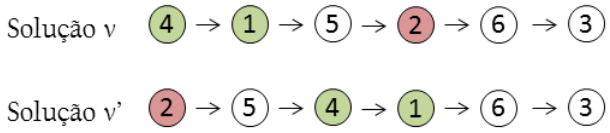
Figura 11 – Movimento de Realocação e Troca de Bloco - N^{TRB}.



Fonte: figura elaborada pelo autor

N^{TBU} - Similar à estrutura N^{TB}, é selecionado um bloco que será trocado de posição com apenas uma tarefa, como demostrado na Figura 12, onde é feita uma troca entre o bloco $B = \{4, 1\}$ e a tarefa 2.

Figura 12 – Movimento de Troca de Bloco com Uma Tarefa - N^{TBU}.



Fonte: figura elaborada pelo autor

Diversos trabalhos encontrados na literatura, como por exemplo o trabalho de Bustamante (2006), Júnior (2007), Rosa (2009) e Gonçalves (2010), utilizaram somente três estruturas de vizinhança. Neste trabalho, como o conceito de movimentos na vizinhança converge com o conceito de mutação, foram implementados seis movimentos com o objetivo

de gerar diversidade no processo evolutivo do algoritmo implementado, que será detalhado na subseção 4.4.3.1.

4.3 Função de Avaliação

Para determinar a qualidade das soluções uma função f é aplicada sobre um indivíduo representado por um vetor v, no qual cada posição do vetor representa a ordem de processamento das tarefas. A Equação 4.1 representa a função f, na qual deve ser minimizada:

$$f(v) = \sum_{i=1}^{n} (\alpha_i e_i + \beta_i t_i) \tag{4.1}$$

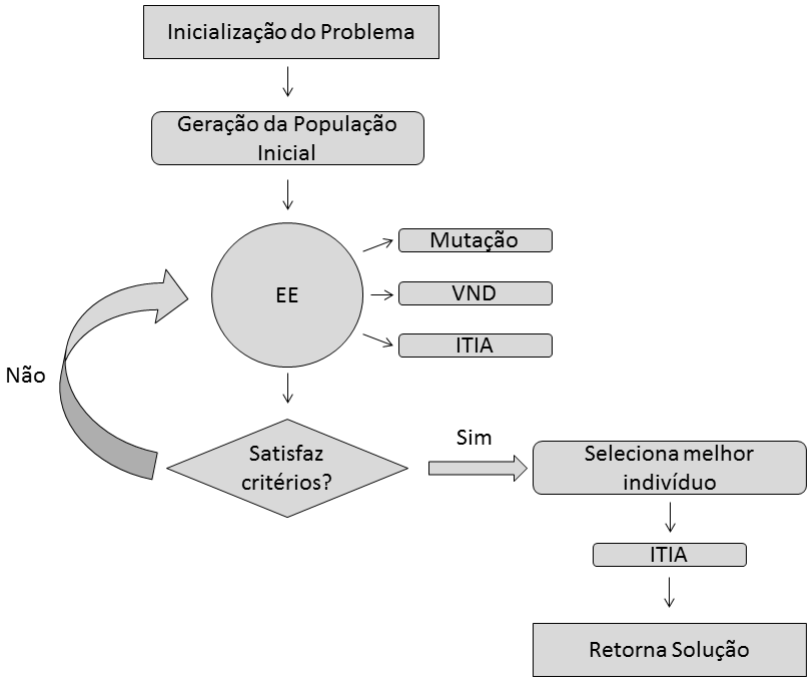
onde α_i e β_i representam, respectivamente, as penalidades de antecipação e atraso da tarefa i, enquanto e_i e t_i representam o tempo de antecipação e atraso em relação à execução desta tarefa.

4.4 Métodos de Resolução Aplicados

4.4.1 Algoritmo GEVITIA

O algoritmo proposto, denominado GEVITIA, é composto de duas etapas. A primeira etapa consiste na geração da população inicial baseada no procedimento de construção da meta-heurística GRASP. Na segunda etapa, utiliza-se da combinação da Estratégia Evolutiva (EE) com o método de busca local VND e o ITIA como procedimentos para o refinamento das soluções. Por fim, após todos os critérios serem satisfeitos, o melhor indivíduo é selecionado e o ITIA é aplicado para determinar a data ótima de início de processamento das tarefas. Uma exemplificação do algoritmo proposto é apresentada na Figura 13. A explicação detalhada dos procedimentos utilizados será feita nas seções a seguir.

Figura 13 – Exemplificação do Algoritmo Proposto.



Fonte: figura elaborada pelo autor

4.4.2 Geração da População Inicial

Antes da geração da população inicial ocorre a inicialização do problema P a ser programado. A inicialização ocorre através da leitura dos dados de uma instância do problema, onde são obtidos todas as informações relacionadas às tarefas que devem ser programadas. A Tabela 1 representa um exemplo de instância do problema. Cada instância possui um conjunto de n tarefas a serem sequenciadas, representado por $P_i = \{1, 2, 3, n\}$, onde cada tarefa i contida em P possui:

- Um tempo de processamento denominado por P_i;
- Uma janela de entrega $[E_i, T_i]$, sendo a data de início e fim da janela, respectivamente;
- Uma penalidade de antecipação α_i, que é aplicada quando a tarefa termina seu processamento antes da data E_i;
- Uma penalidade de atraso α_i, que é aplicada quando a tarefa termina seu processamento após a data T_i.

Neste mesmo processo de inicialização é carregada uma matriz que representa o tempo de preparação de máquina (tempo de *setup*), que indica o tempo necessário para

preparar a máquina entre uma tarefa e outra. No exemplo demonstrado pela Tabela 1 pode-se observar que, a execução da Tarefa 4 após a Tarefa 1 existe um tempo de preparação de 2 unidades de tempo, enquanto o tempo de preparação da Tarefa 3 após a execução da Tarefa 1 é de uma unidade de tempo. Dessa forma, é possível perceber o tempo de preparação de máquina é dependente da sequência de produção programada.

Tabela 1 – Exemplo de Problema de Entrada.

	Dados do Problema					**Tempo de Preparação**			
Tarefas	P_i	E_i	T_i	α_i	β_i	**1**	**2**	**3**	**4**
1	3	14	15	2	4	0	2	1	2
2	4	22	24	7	9	1	0	2	3
3	4	9	12	7	8	1	3	0	1
4	3	5	7	1	4	1	2	2	0

Fonte: elaborado pelo autor

A geração da população inicial ocorre através da fase construtiva baseada na meta-heurística GRASP. Nesta etapa, é gerada uma população de indivíduos distintos utilizando-se de quatro métodos heurísticos encontrados na literatura que se adaptam bem para este modelo de problema (GONÇALVES, 2010). Os métodos heurísticos utilizados são descritos a seguir:

- *Earliest Due Date* (EDD): as tarefas são ordenadas pela data entrega, sendo a mais recente a primeira;
- *Tardiest Due Date* (TDD): as tarefas são ordenadas pela data entrega, sendo a mais tardia a primeira;
- *Shortest Processing Time* (SPT): as tarefas são ordenadas pelo tempo de processamento, sendo a com menor tempo a primeira;
- Aleatório: a seleção das tarefas ocorre de modo aleatório.

Os Algoritmos 1 e 2 representam a construção da população inicial que será posteriormente utilizada pelo procedimento de Estratégia Evolutiva. Inspirado na fase construtiva do GRASP, os procedimentos constroem cada individuo da população inicial de forma parcialmente gulosa. Esta construção ocorre de forma gradativa no qual, para uma instância é formada uma lista restrita de candidatos (LRC). O tamanho da LRC é calculado baseado no parâmetro $\gamma \in [0, 1]$ tal que, quanto mais próximo de 1 menor o nível de gulosidade e maior o de aleatoriedade, dentro desta lista (LRC) uma tarefa é selecionada e adicionada à programação final, até que todas as tarefas tenham sido sequenciadas.

O Algoritmo 1 inicia-se obtendo o fator de gulosidade γ que será utilizado para geração da LRC (linha 3) no Algoritmo 2. Após, inicia o procedimento de construção e sequenciamento das tarefas do problema (linha 4 e 5), nesta etapa é escolhido aleatoriamente uma heurística construtiva para ser utilizada, sendo elas, EDD, TDD, SPT e totalmente aleatória, são passados os parâmetros γ, e o problema a ser sequenciado e a heurística escolhida. Concluído o sequenciamento, verifica-se então se o indivíduo gerado já existe na população inicial (linha 6), se existir, o indivíduo é descartado e gera-se outro, este processo continua até que sejam gerados μ indivíduos distintos para a população inicial.

O procedimento apresentado pelo algoritmo 2 representa a programação inicial do problema baseado em alguma das heurísticas construtivas implementadas. De acordo com a heurística selecionada previamente, é feita a programação sequencial do indivíduo (linhas 1 até 12), neste momento considera-se que o sequenciamento foi efetuado de forma gulosa, contudo, para gerar uma população diversificada esse indivíduo é reprogramado de forma gradativa e parcialmente gulosa (linhas 13 até 18).

Após todo o processo de geração, a população inicial é armazenada para a próxima etapa do algoritmo proposto (GEVITIA). A segunda etapa consiste em aplicar técnicas de computação evolutiva, busca local e determinação de datas ótimas de processamento como técnica de refinamento dos indivíduos/soluções geradas. Este processo evolutivo será melhor detalhado na seção seguinte.

Algoritmo 1: Geração da População inicial

Entrada: $\mu \leftarrow$ quantidade de indivíduos pais a serem gerados , $problema \leftarrow$ dados da instância
Saída: $pop \leftarrow$ população inicial

1 $i \leftarrow 0$;
2 **enquanto** $i < \mu$ **faça**
3 escolha $\gamma \in \Gamma$;
4 $tipoheuristica \leftarrow selecionaAleatoriamente([EDD, TDD, SPT, Aleatorio])$;
5 $ind \leftarrow constroiNovoIndividuo(\gamma, tipoheuristica, problema)$;
6 **se** $ind \notin pop$ **então**
7 $pop \leftarrow pop \cup \{ind\}$;
8 $i \leftarrow i + 1$;
9 **fim**
10 **fim**
11 **retorna** pop;

Algoritmo 2: Constrói Novo Indivíduo

Entrada: γ, $tipoHeuristicaConst$, $problema \leftarrow$ dados da instância
Saída: $individuo$

$tamLista \leftarrow qtdTarefas(problema)$;
$vInicial \leftarrow \emptyset$;
$vFinal \leftarrow \emptyset$;
se $tipoHeuristicaConst = 1$ **então**
    $vInicial \leftarrow$ sequencia tarefas pela heurística construtiva EDD ;
senão se $tipoHeuristicaConst = 2$ **então**
    $vInicial \leftarrow$ sequencia tarefas pela heurística construtiva TDD ;
senão se $tipoHeuristicaConst = 3$ **então**
    $vInicial \leftarrow$ sequencia tarefas pela heurística construtiva SPT ;
senão
    $vInicial \leftarrow$ sequencia tarefas de forma aleatória ;
fim
enquanto $vInicial \neq \emptyset$ **faça**
    $tamLRC \leftarrow max(1, (\gamma \times tamLista))$;
    $tarefa \leftarrow selecionaTarefaAleatoria(vInicial, tamLRC)$;
    $vFinal \leftarrow vFinal \cup \{tarefa\}$;
    $vInicial \leftarrow vInicial \setminus \{tarefa\}$;
fim
$individuo \leftarrow vFinal$;
retorna $individuo$;

4.4.3 Estratégia Evolutiva

As técnicas de computação evolutiva são inspiradas nos conceitos da evolução das espécies, adotando diversas características e termos vindo da área de biologia. Cunha Antonio Gaspar, Takahashi Ricardo e Antunes Carlos Alberto Henggeler de Carvalho (2012) dizem que segundo a Teoria da Evolução de Darwin, a evolução das espécies ocorrem devido à seleção natural e a adaptação ao ambiente em que vivem. Esta evolução contínua pode ser vista como um processo de otimização na qual a adaptação e seleção implica na resultância de indivíduos com melhores características e maior probabilidade de sobrevivência no cenário em que se encontram.

Métodos populacionais diferem de métodos heurísticos tradicionais (não evolucionários) por se basearem na busca em uma coleção de indivíduos diferentemente de um solução com somente um candidato. Essa ideia evolutiva vinda da biologia inspirou no desenvolvimento de técnicas computacionais de busca populacional, entre elas, Estratégia Evolutiva (EE), Algoritmos Genéticos (AG) e Programação Genética (PG). As princi-

pais diferenças entre tais métodos são as formas iterativas de evolução e os métodos de adaptação aplicados na geração de novos indivíduos (LUKE, 2013).

A implementação aplicada neste trabalho foi de Estratégia Evolutiva devido à sua capacidade de gerar indivíduos diversos através de técnicas de mutação, e de forma iterativa caminhar no espaço de soluções. Esta diversidade conduz no processo de refinamento encontrar indivíduos de boa qualidade e que, através do processo de mutação passem suas características aos seus descendentes levando a soluções melhores para o problema abordado.

O processo evolutivo, também denominado de geração, atua sobre a população de indivíduos aplicando operadores genéticos de recombinação e/ou mutação, estes operadores permitem controlar a diversidade da população e exploração do espaço de busca. A seleção dos indivíduos mais aptos ocorre através de uma função de avaliação que determina a aptidão e qualidade, selecionando para as próximas gerações os melhores indivíduos.

A EE possui dois tipos de estratégia, representados por (μ, λ) e $(\mu + \lambda)$. Os indivíduos pais μ sofrem durante o processo evolutivo mutações gerando para cada pai uma quantidade λ de indivíduos filhos. Na primeira estratégia (μ, λ), é gerado uma população contendo todos os λ descendentes que são avaliados através da função de aptidão e somente os μ melhores indivíduos entre os descendentes sobrevivem para a próxima geração. Na outra estratégia a população final consiste na união dos indivíduos pais e seus descendentes gerados, ou seja, a população final se dá por $\mu + \lambda$, neste caso os indivíduos pais competem com os filhos no processo evolutivo. Neste trabalho, a técnica implementada foi o modelo $(\mu + \lambda)$, portanto, a cada geração a população é composta pelos indivíduos pais e filhos e após o processo de avaliação de aptidão, somente os μ melhores indivíduos sobrevivem para a próxima geração. A estratégia continua sua execução até que todos os critérios de parada definidos sejam satisfeitos.

4.4.3.1 Aplicação de EE ao PSUMAA

A aplicação da técnica de Estratégia Evolutiva no problema de sequenciamento em uma máquina é representada pelo pseudocódigo apresentado no algoritmo 3. O algoritmo recebe a população inicial gerada previamente pelos algoritmos 1 e 2. A geração é explicada na subseção 4.4.2, a taxa de probabilidade de aplicação do ITIA que tem sua funcionalidade explanada na subseção 4.4.4, o tipo de busca local aplicado definido pelo parâmetro $tipoVnd$ e o parâmetro *problema* que contém todas as informações da instância. As linhas 1 até 3 inicializam as variáveis necessárias para avaliação do critério de parada. O critério de parada é o numero de evoluções sem melhora e o tempo máximo de execução do algoritmo. Cada indivíduo da população inicial gera λ descendentes que são gerados através dos operadores de mutação apresentados na seção 4.2 (linhas 6 até 13). Estes indivíduos são aplicado ao procedimento ITIA de acordo com a taxa de probabilidade (linha 12). Após

a geração de todos os descendentes é feita a união $(\mu + \lambda)$, e selecionados os μ melhores indivíduos que continuarão no processo de evolução (linha 17).

Neste trabalho foram implementados duas estratégias de busca local, utilizando a Descida em Vizinhança Variável, do inglês (*Variable Neighborhood Descent* (VND)) proposto por Mladenović e Hansen (1997). Os métodos implementados foram variações clássicas conhecidas na literatura, sendo utilizado o método de Primeiro de Melhora e Método Randômico de Descida (MRD). As estruturas de vizinhança adotadas nestes procedimentos são explanadas na seção 4.2. Para o MRD o número máximo de exploração sem melhora em uma estrutura de vizinhança foi de uma iteração. Como aplicar os dois métodos de busca local nos μ melhores indivíduos selecionados demanda alto custo computacional, a execução final do algoritmo restringiu-se apenas ao MRD. A seção 5.1 explana o processo de definição dos parâmetros e o porquê de tal método ser escolhido. De acordo com o $tipoVnd$ o método escolhido é aplicado durante o processo de evolução (linha 18 até 22).

Após a aplicação da busca local, o melhor indivíduo é avaliado (linhas 23 até 29), e o processo evolutivo continua até que os critérios de parada sejam satisfeitos (linha 6). Satisfazendo o critério de parada, o melhor indivíduo é então submetido ao ITIA novamente, tendo em vista que durante o processo de evolução essa aplicação ocorre de acordo com uma probabilidade, e o melhor indivíduo pode não ter tido seu sequenciamento de datas ótimas realizado. Por fim, o algoritmo retorna o melhor indivíduo selecionado

como solução final.

Algoritmo 3: Estratégia Evolutiva

Entrada: $pctgItia$, $tipoVnd$, $populacaoInicial$, μ, λ, $problema$
Saída: $melhorIndividuo$

avalie todos os indivíduos em $populacaoInicial$;
$melhorIndividuo \leftarrow$ melhor indivíduo em $populacaoInicial$;
$numIterSemMelhora \leftarrow 0$;
enquanto $(numIterSemMelhora < 4 \times qtdTarefas(problema))$ *e* $(tempoDeExecucao < 20\ minutos)$ **faça**
    $pai \leftarrow 0$;
    enquanto $pai < \mu$ **faça**
        $descendente \leftarrow 0$;
        enquanto $descendente < \lambda$ **faça**
            clone o pai;
            $tipoMutacao \leftarrow selecionaAleatoriamente([N^R, N^T, N^{RB}, N^{TB}, N^{TRB}, N^{TBU}])$;
            muteDescendente($tipoMutacao$);
            submeta o descendente gerado ao ITIA de acordo com a probabilidade de aplicação $pctgItia$;
            $descendente \leftarrow descendente + 1$
        fim
        $pai \leftarrow pai + 1$;
    fim
    selecione os μ melhores indivíduos de $(\mu + \lambda)$;
    se $tipoVnd = 1$ **então**
        submeta os μ melhores indivíduos selecionados à busca local VND com Primeiro de Melhora;
    senão se $tipoVnd = 2$ **então**
        submeta os μ melhores indivíduos selecionados à busca local VND com Método Randômico de Descida;
    fim
    $melhorIndividuoAux \leftarrow$ obtenha o melhor indivíduo;
    se $melhorIndividuoAux$ *for melhor que* $melhorIndividuo$ **então**
        $melhorIndividuo \leftarrow melhorIndividuoAux$;
        $numIterSemMelhora \leftarrow 0$;
    senão
        $numIterSemMelhora \leftarrow numIterSemMelhora + 1$;
    fim
fim
submeta $melhorIndividuo$ ao ITIA ;
retorna $melhorIndividuo$;

Fonte: pseudocódigo desenvolvido pelo autor

4.4.4 *Idle Time Insertion Algorithm*

O problema de sequenciamento em uma máquina pode ser dividido em dois sub-problemas, determinar a ordem no qual as tarefas devem ser executadas e determinar a data de início de cada tarefa na qual a soma das penalidades por antecipação e atraso devem ser minimizadas. Para tal motivo, Rosa et al. (2017) propuseram um algoritmo de complexidade $O(n^2)$ denominado ITIA para determinar a data ótima do início de processamento inserindo tempo ocioso de máquina.

Este processo de ordenação e inserção de tempo ocioso começa da última tarefa até a primeira sucessivamente, verificando em cada passo se a inserção de tempo ocioso é vantajosa. Para melhor entendimento do algoritmo são apresentados abaixo algumas definições e conceitos:

- Considere uma sequência X de tarefas, uma subsequência dessas tarefas é denominado *Bloco* B, seja $X = \{2, 3, 1, 4, 5\}$, para que a subsequência $B = \{4, 5\}$ seja considerada um bloco, não pode haver tempo ocioso entre as tarefas x_4 e $x_5 \in X$;
- C_{x_i} representa a data de conclusão da tarefa $x_i \in X$;
- $S_{(x_i)(x_i+1)}$ representa o tempo de preparação de máquina entre a tarefa x_i e a próxima tarefa na ordem programada em X;
- P_{x_i} representa o tempo de processamento da tarefa $x_i \in X$;
- E_{x_i} representa a data de início da janela de entrega da tarefa $x_i \in X$;
- T_{x_i} representa a data final da janela de entrega da tarefa $x_i \in X$;
- Para identificar se o deslocamento do bloco é vantajoso é feito o cálculo do custo marginal do bloco, representado pela Equação 4.2;

$$MC(B) = \sum_{x \in B : C_x \geq T_x} \beta_x - \sum_{x \in B : C_x < E_x} \alpha_x \tag{4.2}$$

- Para calcular a quantidade de unidades de tempo de deslocamento utiliza-se de duas funções, m_1 e m_2. A quantidade de unidades de tempo que podem ser deslocadas dentro da janela de tempo é calculada por m_1, enquanto m_2 verifica quantas unidades podem ser deslocadas até o início da janela caso haja antecipação, tais funções são representadas pela Equação 4.3 e Equação 4.4;

$$m_1 = \min_{x \in B: E_x \leq C_x < T_x} (T_x - C_x) \tag{4.3}$$

$$m_2 = \min_{x \in B: C_x < E_x} (E_x - C_x) \tag{4.4}$$

- Se $MC(B) \geq 0$, deslocar as tarefas contidas no bloco B não traz benefícios, porque o aumento dos custos dos atrasos serão maiores que a redução dos custos de antecipação. Caso contrário, se $MC(B) < 0$ é considerado vantajoso deslocar o bloco B; e
- O valor unitário do deslocamento é representado por $\varphi = min(m_1, m_2)$, caso x_i seja a última tarefa ($x_n \in X$). Caso contrário, o cálculo do deslocamento é representado por $\varphi = min(C_{(x_i)+1} - P_{(x_i)+1} - S_{(x_i)(x_i+1)}, m_1, m_2)$.

O pseudocódigo apresentado no Algoritmo 4 exemplifica, utilizando os conceitos acima o processo de inserção de tempo ocioso e determinação das datas ótimas de início de processamento das tarefas. São consideradas duas estruturas, o vetor X que representa a sequência já programada na ordem de execução das tarefas e um vetor de blocos (linha 1 e 2). O processo começa pela última tarefa (linha 3) até a primeira, sucessivamente até terminar a execução. Se houve antecipação para a tarefa x_i (linha 7) então calcula-se o custo marginal para identificar se o deslocamento para a direita é benéfico (linha 8). Caso seja benéfico, o valor necessário para o deslocamento é calculado (linha 10 e 11), se x_i não for a última tarefa programada (linha 12) o cálculo do deslocamento leva em conta também o limite imposto pelo início da tarefa seguinte $x_i + 1$, uma vez que o término da tarefa x_i não pode sobrepor a data de início da próxima tarefa, este valor de deslocamento é então atribuído à φ (linha 13). Se a tarefa x_i for a última, o cálculo irá considerar o deslocamento dado por m_1 e m_2 (linha 15). Após o cálculo, a programação da hora de início e fim das tarefas do bloco corrente são recalculadas (linha 17 até 19). Neste cenário, caso haja bloco sucessor e, o início do bloco sucessor aconteça exatamente após o término do bloco corrente, então, por definição, junta-se os blocos (linha 20 até 22). O custo marginal para o novo bloco é calculado e reinicia-se o processo enquanto for viável

até a primeira tarefa de X.

Algoritmo 4: IDLE TIME INSERTION ALGORITHM

Entrada: *individuo*, *problema* ← objeto que possui todos os dados da instância
Saída: *individuo*

1 inicializa o vetor X com a sequência de tarefas do *individuo* ;
2 $Blocos \leftarrow \emptyset$;
3 **para** $i = n, n-1, ..., 2, 1$ **faça**
4 $bloco \leftarrow \emptyset$;
5 adiciona $x_i \in X$ no início de *bloco* ;
6 adiciona *bloco* no início do vetor *Blocos* ;
7 **se** $C_{x_i} < E_{x_i}$ **então**
8 $mc = calculaCustoMarginal(bloco)$;
9 **para** $mc < 0$ **faça**
10 $m_1 = calculaM_1(bloco)$;
11 $m_2 = calculaM_2(bloco)$;
12 **se** $x_i \neq x_n$ **então**
13 $\varphi \leftarrow min(C_{(x_i)+1} - P_{(x_i)+1} - S_{(x_i)(x_i+1)}, m_1, m_2)$;
14 **senão**
15 $\varphi \leftarrow min(m_1, m_2)$;
16 **fim**
17 **para** $x \in bloco$ **faça**
18 $C_x = C_x + \varphi$;
19 **fim**
20 **se** $C_{x_i} + S_{(x_i)(x_i+1)} + P_{x_i+1} = C_{x_i+1}$ **então**
21 junte *bloco* ao bloco sucessor ;
22 **fim**
23 $mc = calculaCustoMarginal(bloco)$;
24 **fim**
25 **fim**
26 **fim**
27 X recebe sequência programada em *Blocos* ;
28 atualiza *individuo* com a programação definida no vetor X;
29 **retorna** *individuo*;

Fonte: Algoritmo adaptado de Rosa et al. (2017)

4.5 Considerações Gerais

Diversas metodologias para a resolução do Problema de Sequenciamento de Tarefas em uma Máquina com Penalidades por Atraso e Antecipação da Produção são propostas ao longo do tempo. Este trabalho tem como intuito explorar novos cenários através de um método pouco utilizado para a resolução desta classe de problema, implementando técnicas de busca populacional utilizando Estratégia Evolutiva. Após a implementação do algoritmo proposto, torna-se necessário validar a implementação através de testes e análises para identificar se o mesmo é capaz de obter soluções de qualidade e competitivas com trabalhos já presentes na literatura. Para isso, os testes são executados com base nos parâmetros que definem sua execução. Os experimentos computacionais são apresentados no Capítulo 5 assim como a análise feita para determinar a capacidade de resolução do algortimo proposto.

5 Experimentos Computacionais

Nesta seção serão discutidos o processo de parametrização e resultados obtidos do algoritmo desenvolvido. Sendo a proposta deste trabalho implementar um algoritmo baseado na fase construtiva do GRASP, Estratégia Evolutiva, Busca Local com VND e um algoritmo para inserção de tempo ocioso e determinação da data ótima de início de processamento das tarefas (ITIA) para o problema de Sequenciamento de tarefas em uma máquina. O objetivo desta fase é verificar se o algoritmo proposto é capaz de gerar soluções viáveis e se o mesmo é competitivo com demais algoritmos presentes na literatura

O algoritmo proposto foi implementado na linguagem C++ e os experimentos foram executados em um computador com processador Intel(R) Core(TM) i7-4790 de 3.60GHz, 16 GB de memória RAM e Sistema Operacional Ubuntu 14.04 de 64bits. O processo de parametrização assim como a estrutura de teste adotada no trabalho é detalhado na seção 5.1. Após a definição de parâmetros os testes são executados e avaliados na seção 5.2. Por fim, na seção 5.3 são discutidos os resultados e as considerações finais deste capítulo.

As instâncias utilizadas neste trabalho foram propostas por Júnior (2007) que foram baseadas no trabalho de Wan e Yen (2002), instâncias que serviram de referência para diversos trabalhos posteriores, nos quais este também se inclui. Esta base de dados é constituída por grupos de problemas-teste com 8, 9, 10, 11, 12, 15, 20, 25, 30, 40, 50 e 75 tarefas.

5.1 Definição dos Parâmetros

Esta fase tem como intuito testar diferentes configurações de parametrização para definir o melhor cenário final de execução. Os parâmetros μ (número de indivíduos pais), λ (número de indivíduos filhos), número máximo de execuções sem melhora e tempo máximo de execução do $GEVITIA$ foram definidos empiricamente a partir de testes preliminares. Os valores definidos para cada parâmetro acima são descritos na Tabela 2. É importante destacar que os dois últimos parâmetros não são parâmetros de entrada, mas sim o critério utilizado para a parada do algoritmo.

Tabela 2 – Parâmetros definidos empiricamente.

Parâmetro	**Valor**
μ	200
λ	20
Nº Máx. Evoluções Sem Melhora	$4\times$ $quantidadeTarefas(instancia)$
Tempo Máx. Execução	20 minutos

Fonte: elaborado pelo autor

No caso do parâmetro de número de evoluções sem melhora é estipulado um valor máximo utilizando a fórmula $4 \times quantidadeTarefas(instancia)$ demonstrada na Tabela 2, ou seja, para uma instância que contém 10 tarefas, caso o algoritmo atinja $4 \times 10 = 40$ evoluções sem melhora, o algoritmo para e retorna o melhor indivíduo da população corrente como solução final. Tanto este parâmetro quanto o de tempo máximo de execução foram selecionados com a intenção de finalizar o processo de evolução caso houvesse convergência dos resultados ou caso o algoritmo estivesse demandando muito tempo de processamento.

Visando definir os valores para os parâmetros de estratégia do VND e porcentagem de aplicação do ITIA, foi executado um planejamento fatorial 2×5. Foram utilizadas 11 instâncias de 8 a 50 tarefas, 2 configurações de VND, sendo definido como VND_1 o método de Primeiro de Melhora e VND_2 o Método Randômico de Descida e 5 valores para a porcentagem de probabilidade da aplicação do ITIA, sendo as probabilidades $20\%, 40\%, 60\%, 80\%, 100\%$. Cada combinação de instância, tipo VND e porcentagem de aplicação do ITIA foi executada 30 vezes, gerando um total de $2 \times 5 \times 11 \times 30 = 3300$ replicações. As instâncias utilizadas para a parametrização são descritas na tabela abaixo:

Tabela 3 – Instâncias utilizadas para parametrização.

Instância	**Número de Tarefas**
INST0801	8
INST0901	9
INST1001	10
INST1101	11
INST1201	12
INST1501	15
INST2001	20
INST3001	30
INST4001	40
INST5001	50

Fonte: elaborado pelo autor

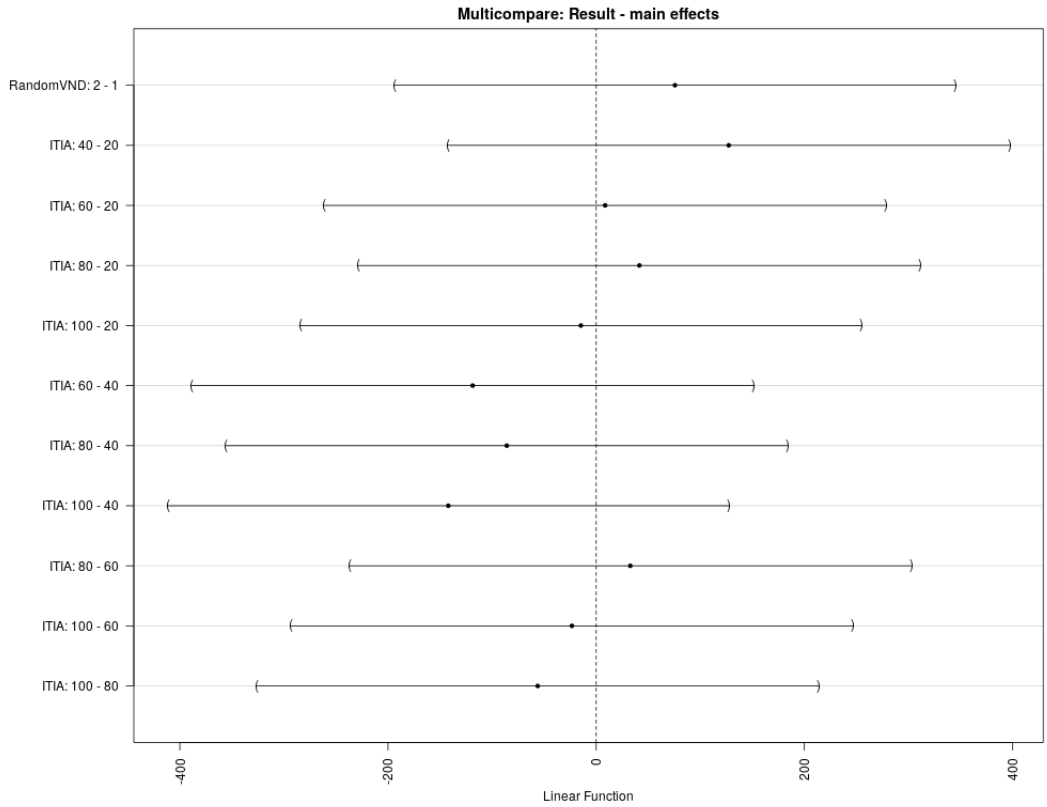
Considerando que o computador pode estar utilizando em momentos distintos uma quantidade maior ou menor de recursos é adotado a aleatorização das execuções, sendo aleatorizado a ordem de execução das observações e a configuração dos parâmetros, esta aleatorização busca eliminar possíveis problemas de seleção e regressão estatística, ou seja, evitar que certas características momentâneas do computador afetem observações com configurações semelhantes.

A análise estatística dos resultados foi feita por meio da Análise de Variância (ANOVA). A variância é uma medida de dispersão que indica a distanciação da média pelos valores obtidos. Esta análise é utilizada para verificar se existe diferença estatística entre as configurações de parâmetros adotada.

Para comparar as configurações adotadas e validar sua significância na execução dos problemas, foi executado o teste de Tukey demonstrado pelos gráficos na Figura 14 e Figura 15.

A Figura 14 representa a análise dos parâmetros em função da qualidade das soluções obtidas, representando para cada combinação de configuração sua significância estatística.

Figura 14 – Teste de Tukey em relação ao resultado obtido.

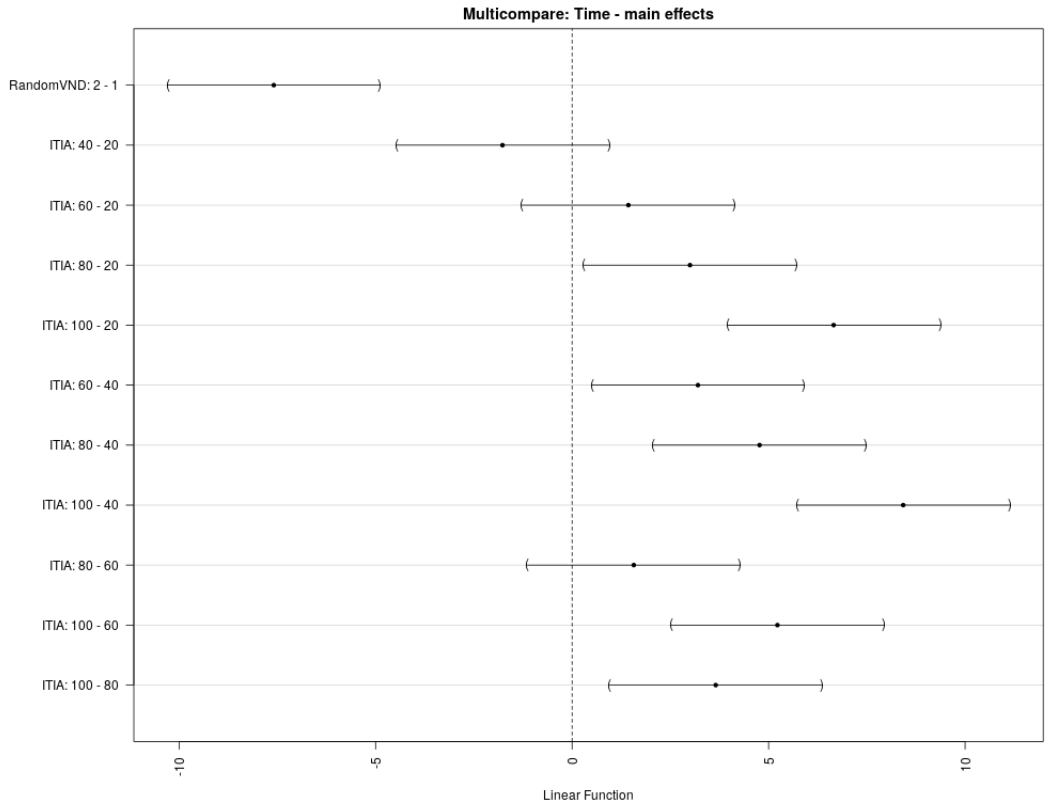


Fonte: elaborado pelo autor

É possível observar na Figura 14 que estatisticamente não há diferença entre os valores de parâmetros testados quando comparados em relação à qualidade das soluções obtidas pelo algoritmo desenvolvido. Ou seja, as diversas combinações possíveis apresentam a mesma capacidade na geração de boas soluções.

A Figura 15 representa a análise das configurações dos parâmetros em função do tempo demandado para execução do algoritmo. O intuito desta análise é identificar a melhor combinação de parâmetros que demandam menos tempo de processamento computacional e que possuam significância estatística plausível para a obtenção de bons resultados.

Figura 15 – Teste de Tukey em relação ao Tempo.



Fonte: elaborado pelo autor

Em relação ao tempo, existe uma diferença entre os valores de parâmetros definidos. Para estratégia de Busca Local, como pode ser observado, o VND_2 (Método Randômico de Descida) demanda menos tempo em relação ao VND_1 (Primeiro de Melhora). No caso do algoritmo de inserção de tempo ocioso (ITIA), é possível observar que as taxas menores de probabilidade de aplicação de 20% até 60% demandam menos tempo em relação às taxas de 80% e 100%.

Com base nos resultados apresentados, foi possível observar que não há diferença estatística das configurações de parâmetros utilizadas quando em função da qualidade das soluções obtidas, porém há configurações que tornam o processo de resolução dos problemas-teste mais rápido. Como o intuito da otimização é obter os melhores resultados preferencialmente no menor tempo possível, foram escolhidos os seguintes parâmetros: O método de busca local adotado foi o VND_2 (Método Randômico de Descida) e a taxa de probabilidade de aplicação do ITIA durante o processo de evolução de 20%, uma vez que tais configurações demonstraram ser mais rápidas como pode ser observado pelo gráfico da Figura 15. A Tabela 4 representa a definição dos valores adotados na execução final dos experimentos.

Tabela 4 – Parâmetros Finais.

Parâmetro	**Valor**
μ	200
λ	20
Nº Máx. Evoluções Sem Melhora	$4 \times quantidadeTarefas(instancia)$
Tempo Máx. Execução	20 minutos
VND	Método Randômico de Descida
Probabilidade de aplicação do ITIA na EE	20%

Fonte: elaborado pelo autor

5.2 Análise dos resultados

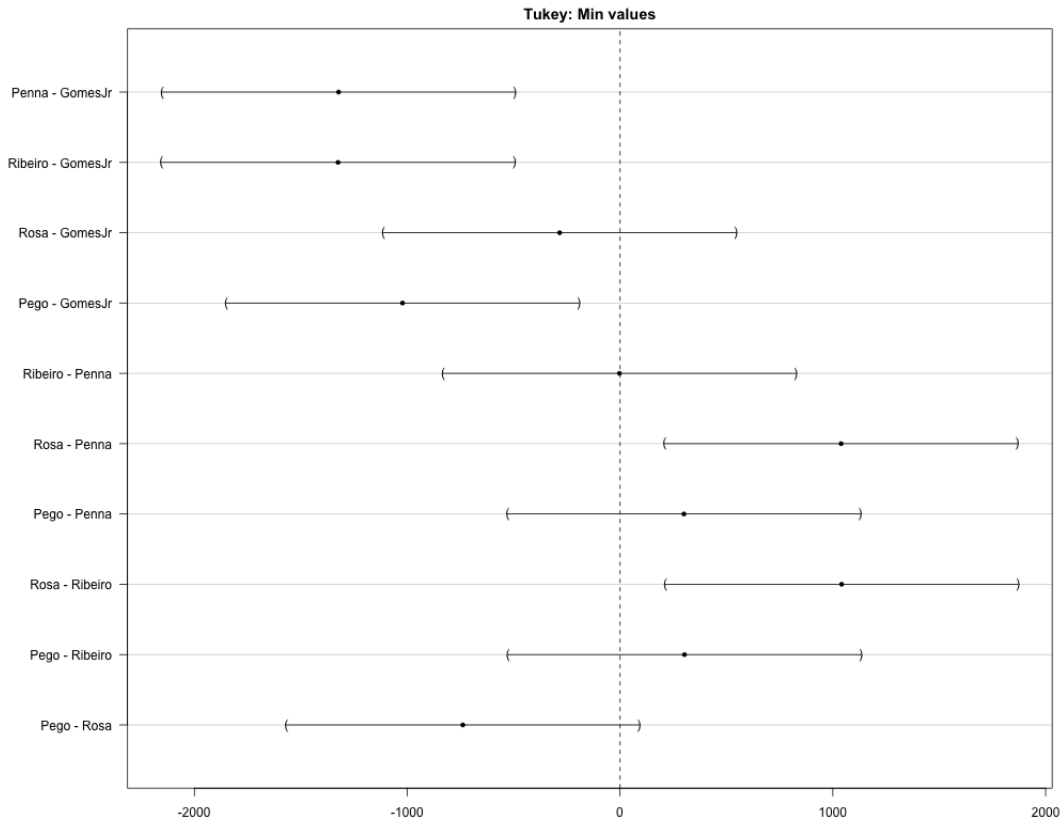
A execução final dos experimentos seguem as mesmas premissas empregadas na fase de parametrização. Para a execução final a base de dados é constituída por grupos de problemas-teste com 8, 9, 10, 11, 12, 15, 20, 25, 30, 40, 50 e 75 tarefas. Cada grupo de problema-teste possui 12 variações, ou seja, para o grupo com 8 tarefas há 12 instâncias de 8 tarefas, totalizando em 144 instâncias a serem programadas. Para cada umas dessas 144 instâncias foram feitas 30 execuções, totalizando em 4320 observações. Os parâmetros utilizados para esta execução foram os escolhidos pelo método de parametrização descrito na seção anterior. Tais parâmetros estão descritos na Tabela 4.

Os resultados obtidos pelo algoritmo proposto neste trabalho (GEVITIA) foram comparados com os resultados de Júnior (2007), Rosa (2009), Ribeiro (2009) e Penna (2009). Tais autores também utilizaram a mesma base de dados de problemas-teste em seus respectivos estudos. Dentre os autores citados, somente Ribeiro (2009) utilizou busca populacional como método de resolução, desenvolvendo um algoritmo genético adaptativo. Os demais autores utilizaram em seus trabalhos métodos de busca não populacionais.

A análise comparativa entre o resultado do algoritmo proposto neste trabalho e demais autores demandava conhecimento de todos resultados obtidos (resultados das 144 instâncias de 8 até 75 tarefas) por cada autor. Tais resultados foram cedidos por Rosa (2009) para critério de comparação desta monografia. Os trabalhos de Gonçalves (2010) e Ramos e Oliveira (2010) se mostraram competitivos como explicado no Capítulo 3, porém não foi possível obter os valores de todas 144 instâncias para critério de comparação, contudo ambos autores também comparam seus estudos com alguns dos autores comparados neste trabalho. Alves (2016) aplicou em seu estudo um Algoritmo de Evolução Diferencial (AED), comparando seus resultados com o de Júnior (2007), porém seu algoritmo foi capaz de encontrar resultados próximos ao ótimo para instâncias menores e para instâncias maiores os resultados ficaram mais distantes, demonstrando assim não ser competitivo ao algoritmo proposto por Júnior (2007). Portanto, o trabalho de Alves (2016) também não entrou na comparação dos resultados desta monografia.

Nesta fase também foi utilizado a Análise de Variância e o teste de Tukey para validação estatística das qualidades dos resultados finais obtidos. Comparando os resultados da nossa abordagem com o da literatura pode-se observar que o algoritmo proposto se mostra competitivo com os demais trabalhos encontrados na literatura. A Figura 16 mostra o gráfico da comparação dos valores mínimos obtidos com os valores da literatura e a Figura 17 compara o valor das médias obtidas das 30 replicações para cada instância.

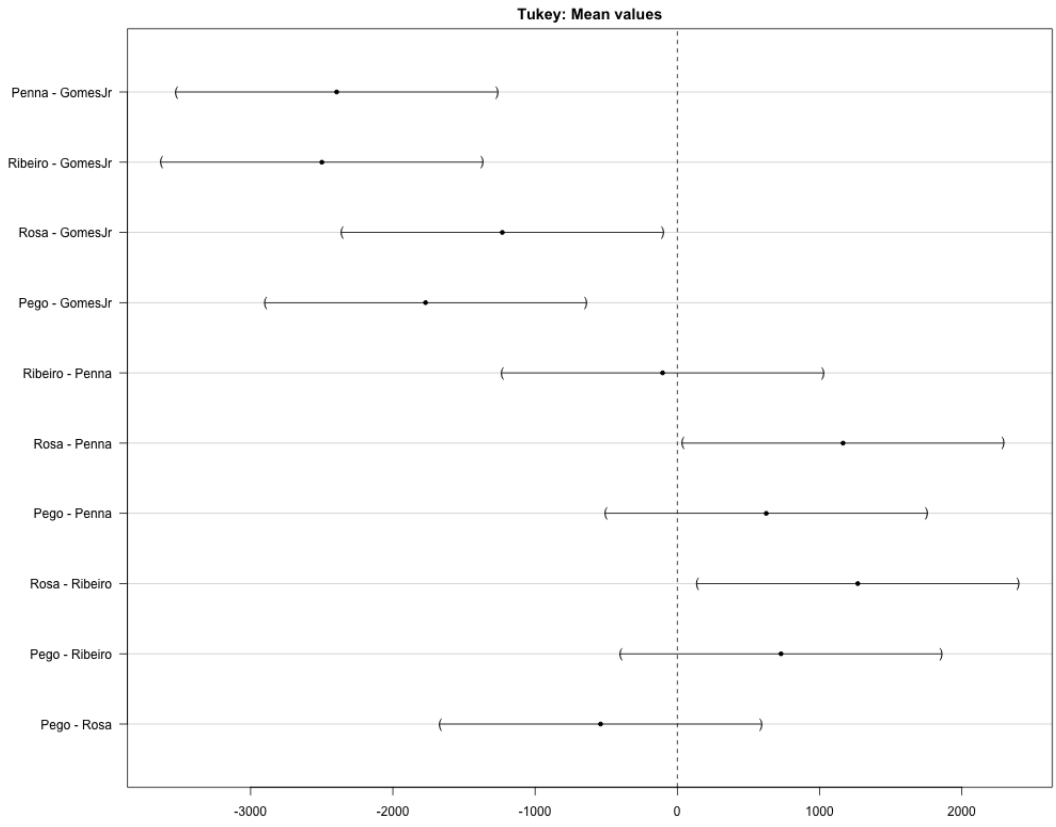
Figura 16 – Comparação dos valores mínimos obtidos.



Fonte: elaborado pelo autor

Em relação às melhores soluções encontradas é possível observar na Figura 16 que os resultados encontrados neste trabalho, por Penna (2009) e Ribeiro (2009) são estatisticamente melhores que o de Júnior (2007). Ao comparar os valores mínimos deste trabalho com os de Penna (2009), Ribeiro (2009) e Rosa (2009), estatisticamente possuem a mesma significância.

Figura 17 – Comparação da média dos valores obtidos.



Fonte: elaborado pelo autor

Considerando a média dos valores das soluções obtidas, pode-se observar no gráfico da Figura 17 que os resultados encontrados neste trabalho, por Penna (2009) e Ribeiro (2009) também são estatisticamente melhores que o de Júnior (2007). Assim como os valores mínimos obtidos, quando comparado a média dos resultados, estatisticamente não há diferença entre o GEVITIA proposto neste trabalho e os algoritmos propostos por Penna (2009), Ribeiro (2009) e Rosa (2009).

A tabela Tabela 5 do Apêndice A apresenta os resultados dos experimentos, o melhor valor obtido pelo GEVITIA e o da literatura, além destes valores é apresentado o gap(%) entre os valores. O calculo do gap é representado pela Equação 5.1.

$$gap = \frac{Best_{GEVITIA} - Best_{Literatura}}{Best_{Literatura}} \times 100 \tag{5.1}$$

5.3 Considerações Gerais

Conforme apresentado na seção 5.1 e seção 5.2, os resultados deste trabalho são comparados com os melhores valores encontrados na literatura. O algoritmo proposto

se mostrou competitivo e capaz de obter soluções de qualidade utilizando métodos de Computação Evolutiva, tal metodologia foi pouco abordada na resolução desta classe de problemas.

Como pode ser observado na Tabela 5 do Apêndice A o GEVITIA foi capaz de obter os mesmos valores da literatura para os problemas-teste de 8, 9, 10, 11, 12, 15, 20 e 25 tarefas obtendo um gap de 0% para tais problemas. Para os problemas de 30 tarefas o algoritmo proposto obteve gap 0 em 10 das 12 instâncias com a mesma quantidade de tarefas, equiparando neste cenário com a literatura em 83,33% dos problemas com 30 tarefas. A duas instâncias que não alcançaram os mesmos resultados da literatura apresentaram um gap de 0,78% e 0,03%.

Para os problemas com 40 tarefas o algoritmo proposto foi capaz de obter gap 0 em 58,33% das observações, sendo o maior gap obtido de 0,39%. Para os problemas de 50 e 75 tarefas o algoritmo alcançou gap 0 em 41,66% das instâncias. O maior gap para os problemas de 50 tarefas foi de 4,40% e para os problemas de 75 tarefas de 7,87%. Houve melhora em relação à literatura para três instâncias de 75 tarefas, obtendo uma melhora de até 1,42%.

Com esses dados e análise estatística feita, é possível afirmar que a metodologia aplicada neste trabalho é competitiva com as comparadas na literatura, sendo capaz de gerar soluções de alta qualidade e, em alguns casos, obter melhores resultados.

6 Conclusão

Este trabalho teve como foco o estudo do Problema de Sequenciamento de Tarefas em uma Máquina com Penalidades por Atraso e Antecipação da Produção (PSUMAA) e aplicações de métodos heurísticos e computação evolutiva para resolução desta classe de problema. O intuito era identificar a capacidade do algoritmo implementado de competir com os métodos presentes na literatura e abordar um método pouco utilizado para esta classe de problemas através de busca populacional utilizando Estratégia Evolutiva.

O Capítulo 2 apresentou os Problemas de Programação da Produção e suas variantes dentro do contexto abordado. Estes problemas motivam estudos para resolução dos mesmos que consiste em planejar em um determinado período de tempo as ações necessárias para a produção e entrega dos bens produzidos. Foram apresentadas as variações do problema de sequenciamento de tarefas, e dentre esses foram enfatizados o Problema de Sequenciamento de Máquinas, e a vertente abordada nesta monografia que é o sequenciamento em uma máquina com penalidades por antecipação e atraso da produção com janelas de entrega distintas e tempo de preparação de máquina dependentes da sequência de produção.

No Capítulo 3 foram apresentados os trabalhos presentes na literatura que abordam o PSUMAA, adotando diversas variações do problema e metodologias para sua resolução. Os trabalhos discutidos apresentam em sua maioria técnicas heurísticas não populacionais para a resolução do problema.

A metodologia e algoritmo proposto (GEVITIA) são apresentadas no Capítulo 4, assim como a representação do problema e operadores de mutação que pelo conceito também foram considerados como estrutura de vizinhança para o método de busca local implementado. O objetivo deste capítulo foi elucidar os passos necessários através da construção da população, evolução e refinamento e aplicação do *Idle Time Insertion Algorithm* (ITIA).

Por fim, os resultados foram demonstrados no Capítulo 5, neste capítulo é realizado o processo de parametrização responsável por definir a configuração ideal para a execução final dos experimentos. A análise estatística dos resultados foram feitas através da Análise de Variância (ANOVA) e Teste de Tukey.

Para os problema-testes de 8 até 25 tarefas o algoritmo foi capaz de obter os mesmos valores da literatura, para os problemas de 30 tarefas apenas 2 instâncias não obtiveram valores iguais, contudo a maior diferença foi de 0,78%. Para os problemas-teste de 50 e 75 tarefas o maior GAP foi de 7,87%, sendo capaz de obter valores idênticos aos da literatura para estes problemas em 41,66% das instâncias. O GEVITIA foi capaz de apresentar melhora para 3 problemas-teste de 75 tarefas apresentando um taxa de

melhora de até 1,42%. Comparando os resultados deste trabalho com os da literatura foi possível observar que o algoritmo implementado (GEVITIA) se mostrou competitivo com os demais.

6.1 Trabalhos Futuros

Com a capacidade do GEVITIA e melhorar seu desempenho, propõe-se para trabalhos futuros:

- Aplicação de paralelismo na execução do algoritmo, uma vez que pode-se aproveitar do potencial atual de multi-processamento dos computadores existentes, visando obter em tempo mais hábil soluções factíveis e aptas para as características do mundo real que possuem problemas grandes e necessitam de uma boa solução ao curto prazo;
- Estudar a eficiência individual de cada método utilizado para evolução e estrutura de vizinhança, identificando os movimentos que produzem com mais eficiência soluções com melhores qualidades; e
- Implementar diferentes formulações de busca local e avaliar qual delas produzem melhores resultados.

Referências

ALLAHVERDI, A. et al. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, v. 187, 02 2008. Citado na página 17.

ALVES, Â. B. M. d. C. Análise de representações para o algoritmo de evolução diferencial no contexto discreto. Universidade Federal de Ouro Preto, 2016. Citado 2 vezes nas páginas 25 e 46.

BAKER, K. R.; SCUDDER, G. D. Sequencing with earliness and tardiness penalties: A review. *Operations Research*, Informs, v. 38, n. 1, p. 22–36, 1990. Citado 3 vezes nas páginas 14, 20 e 22.

BUSTAMANTE, L. d. M. Minimização do custo de antecipação e atraso para o problema de sequenciamento de uma máquina com tempo de preparação dependente da sequência: aplicação em uma usina siderúrgica. Programa de Pós-Graduação em Engenharia da Produção. Departamento de Engenharia de Produção, Escola de Engenharia, Universidade Federal de Minas Gerais., 2006. Citado 8 vezes nas páginas 14, 17, 18, 19, 20, 22, 23 e 29.

CUNHA ANTONIO GASPAR, c.; TAKAHASHI RICARDO, c.; ANTUNES CARLOS ALBERTO HENGGELER DE CARVALHO, c. *Manual de computação evolutiva e metaheurística*. Coimbra: Imprensa da Universidade de Coimbra, 2012. ISBN 978-989-26-0583-8 (PDF). Disponível em: <https://digitalis.uc.pt/handle/10316.2/5655>. Citado na página 34.

GONÇALVES, F. A. d. C. A. Sequenciamento em uma máquina: Otimização heurística via multiprocessamento paralelo. Programa de Mestrado em Modelagem Matemática. Diretoria de Pesquisa e Pós-Graduação, Centro Federal de Educação Tecnológica de Minas Gerais., 2010. Citado 4 vezes nas páginas 25, 29, 32 e 46.

JÚNIOR, A. d. C. G. Problema de sequenciamento em uma máquina com penalidades por antecipação e atraso: Modelagem e resolução. Programa de Pós-Graduação em Engenharia da Produção. Departamento de Engenharia de Produção, Escola de Engenharia, Universidade Federal de Minas Gerais., 2007. Citado 9 vezes nas páginas 19, 20, 24, 25, 29, 42, 46, 47 e 48.

LEE, C. Y.; CHOI, J. Y. A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. *Computers & Operations Research*, Elsevier, v. 22, n. 8, p. 857–869, 1995. Citado na página 23.

LUKE, S. *Essentials of Metaheuristics*. second. [S.l.]: Lulu, 2013. Disponível em https://cs.gmu.edu/∼sean/book/metaheuristics/. Citado 2 vezes nas páginas 27 e 35.

MLADENOVIć, N.; HANSEN, P. Variable neighborhood search. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 24, n. 11, p. 1097–1100, nov. 1997. ISSN 0305-0548. Disponível em: <http://dx.doi.org/10.1016/S0305-0548(97)00031-2>. Citado na página 36.

PENNA, P. H. V. Um algoritmo heurístico híbrido para minimizar os custos com a antecipação e o atraso da produção em ambientes com janelas de entrega e tempos de

preparação dependentes da sequência. Programa de Pós-Graduação em Engenharia Mineral. Departamento de Engenharia de Minas, Escola de Minas, Universidade Federal de Ouro Preto., 2009. Citado 7 vezes nas páginas 17, 21, 24, 25, 46, 47 e 48.

RAMOS, R. d. S.; OLIVEIRA, F. B. d. Uma abordagem ao problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção por meio de algoritmos evolutivos. 2010. Citado 2 vezes nas páginas 25 e 46.

RIBEIRO, F. F. Um algoritmo genético adaptativo para a resolção do problema de sequenciamento em uma máquina com penalização por antecipação e atraso da produção. Programa de Mestrado em Modelagem Matemática. Diretoria de Pesquisa e Pós-Graduação, Centro Federal de Educação Tecnológica de Minas Gerais., 2009. Citado 4 vezes nas páginas 25, 46, 47 e 48.

RODRIGUES, G. S. O problema de sequenciamento em uma única máquina, com tempos de preparação dependentes da sequência e penalidades por antecipação e atraso: Estudo de caso de um processo de fabricação por usinagem. Programa de Pós-Graduação em Engenharia de Produção. Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro., 2012. Citado na página 17.

ROSA, B. F. Heurísticas para o problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção. Programa de Mestrado em Modelagem Matemática. Diretoria de Pesquisa e Pós-Graduação, Centro Federal de Educação Tecnológica de Minas Gerais., 2009. Citado 5 vezes nas páginas 25, 29, 46, 47 e 48.

ROSA, B. F. et al. Algorithms for job scheduling problems with distinct time windows and general earliness/tardiness penalties. *Computers & Operations Research*, v. 81, p. 203 – 215, 2017. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S030505481630332X>. Citado 2 vezes nas páginas 38 e 40.

WAN, G.; YEN, B. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, Elsevier, v. 142, n. 2, p. 271–281, 2002. Citado 3 vezes nas páginas 23, 24 e 42.

APÊNDICE A – Tabela de Resultados

A tabela abaixo representa os valores encontrados pelo algoritmo proposto neste trabalho, denominado GEVITIA. Para cada instância processada pelo algoritmo, é comparado o GAP entre o melhor valor obtido pelo GEVITIA e o da literatura. A explicação do algoritmo proposto é feita no Capítulo 4 e o processo de parametrização e análise dos resultados no Capítulo 5.

Tabela 5 – Comparação dos resultados obtidos com as melhores soluções da literatura.

Instância	**GEVITIA**	**Literatura**	**gap (%)**	**Instância**	**GEVITIA**	**Literatura**	**gap (%)**
INST0801	4928	4928	**0,00**	INST2001	16294	16294	**0,00**
INST0802	2739	2739	**0,00**	INST2002	18107	18107	**0,00**
INST0803	4074	4074	**0,00**	INST2003	20249	20249	**0,00**
INST0804	17149	17149	**0,00**	INST2004	29941	29941	**0,00**
INST0805	6195	6195	**0,00**	INST2005	59158	59158	**0,00**
INST0806	4004	4004	**0,00**	INST2006	31053	31053	**0,00**
INST0807	10689	10689	**0,00**	INST2007	40438	40438	**0,00**
INST0808	14059	14059	**0,00**	INST2008	29186	29186	**0,00**
INST0809	13705	13705	**0,00**	INST2009	67827	67827	**0,00**
INST0810	19855	19855	**0,00**	INST2010	34283	34283	**0,00**
INST0811	22774	22774	**0,00**	INST2011	54538	54538	**0,00**
INST0812	6184	6184	**0,00**	INST2012	79599	79599	**0,00**
INST0901	11801	11801	**0,00**	INST2501	23397	23397	**0,00**
INST0902	8561	8561	**0,00**	INST2502	48540	48540	**0,00**
INST0903	4734	4734	**0,00**	INST2503	18503	18503	**0,00**
INST0904	31100	31100	**0,00**	INST2504	25645	25645	**0,00**
INST0905	2986	2986	**0,00**	INST2505	35865	35865	**0,00**
INST0906	13845	13845	**0,00**	INST2506	14189	14189	**0,00**
INST0907	16400	16400	**0,00**	INST2507	37313	37313	**0,00**
INST0908	20817	20817	**0,00**	INST2508	44638	44638	**0,00**
INST0909	14431	14431	**0,00**	INST2509	12839	12839	**0,00**
INST0910	25664	25664	**0,00**	INST2510	51415	51415	**0,00**
INST0911	33202	33202	**0,00**	INST2511	44808	44808	**0,00**
INST0912	13068	13068	**0,00**	INST2512	34197	34197	**0,00**
INST1001	14411	14411	**0,00**	INST3001	43673	43673	**0,00**
INST1002	8349	8349	**0,00**	INST3002	41983	41983	**0,00**
INST1003	11605	11605	**0,00**	INST3003	21951	21951	**0,00**
INST1004	12486	12486	**0,00**	INST3004	64943	64943	**0,00**
INST1005	5679	5679	**0,00**	INST3005	74100	74100	**0,00**
INST1006	2897	2897	**0,00**	INST3006	29829	29829	**0,00**
INST1007	5515	5515	**0,00**	INST3007	74336	74336	**0,00**
INST1008	9534	9534	**0,00**	INST3008	70312	69770	0,78
INST1009	9461	9461	**0,00**	INST3009	21341	21335	0,03
INST1010	22273	22273	**0,00**	INST3010	73702	73702	**0,00**
INST1011	20514	20514	**0,00**	INST3011	35190	35190	**0,00**
INST1012	45554	45554	**0,00**	INST3012	83976	83976	**0,00**
INST1101	16530	16530	**0,00**	INST4001	57086	57086	**0,00**
INST1102	13252	13252	**0,00**	INST4002	49306	49306	**0,00**

Tabela 5 – Comparação dos resultados obtidos com as melhores soluções da literatura. (cont.)

Instância	**GEVITIA**	**Literatura**	**gap (%)**	**Instância**	**GEVITIA**	**Literatura**	**gap (%)**
INST1103	15251	15251	**0,00**	INST4003	28643	28643	**0,00**
INST1104	17573	17573	**0,00**	INST4004	99871	99828	0,04
INST1105	9347	9347	**0,00**	INST4005	29863	29863	**0,00**
INST1106	15737	15737	**0,00**	INST4006	32303	32303	**0,00**
INST1107	12311	12311	**0,00**	INST4007	115345	115312	0,03
INST1108	17361	17361	**0,00**	INST4008	45016	44847	0,38
INST1109	18202	18202	**0,00**	INST4009	77378	77378	**0,00**
INST1110	28886	28886	**0,00**	INST4010	93591	93225	0,39
INST1111	30328	30328	**0,00**	INST4011	105484	105285	0,19
INST1112	76689	76689	**0,00**	INST4012	127269	127269	**0,00**
INST1201	8137	8137	**0,00**	INST5001	108874	105426	3,27
INST1202	9848	9848	**0,00**	INST5002	72291	69244	4,40
INST1203	8002	8002	**0,00**	INST5003	60259	60259	**0,00**
INST1204	29466	29466	**0,00**	INST5004	97957	96837	1,16
INST1205	8468	8468	**0,00**	INST5005	80597	80140	0,57
INST1206	10982	10982	**0,00**	INST5006	64500	64500	**0,00**
INST1207	26939	26939	**0,00**	INST5007	129832	125350	3,58
INST1208	11335	11335	**0,00**	INST5008	76565	76565	**0,00**
INST1209	28610	28610	**0,00**	INST5009	54601	54234	0,68
INST1210	23406	23406	**0,00**	INST5010	163308	162543	0,47
INST1211	23858	23858	**0,00**	INST5011	181141	175894	2,98
INST1212	41983	41983	**0,00**	INST5012	83686	83686	**0,00**
INST1501	18276	18276	**0,00**	INST7501	253302	253362	**-0,02**
INST1502	19622	19622	**0,00**	INST7502	108823	106674	2,01
INST1503	11505	11505	**0,00**	INST7503	85514	85514	**0,00**
INST1504	15164	15164	**0,00**	INST7504	305900	300068	1,94
INST1505	12924	12924	**0,00**	INST7505	114722	114722	**0,00**
INST1506	9396	9396	**0,00**	INST7506	139306	139119	0,13
INST1507	46544	46544	**0,00**	INST7507	313595	318107	**-1,42**
INST1508	24899	24899	**0,00**	INST7508	204268	193600	5,51
INST1509	14457	14457	**0,00**	INST7509	175679	175759	**-0,05**
INST1510	33128	33128	**0,00**	INST7510	337740	320983	5,22
INST1511	42522	42522	**0,00**	INST7511	200137	185530	7,87
INST1512	12793	12793	**0,00**	INST7512	125199	122721	2,02