



Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Engenharia Elétrica

Natália Antunes

Estudo de heurísticas matemáticas para o problema de escalonamento de máquina
simples

Dissertação de Mestrado

Belo Horizonte
2019

Natália Antunes

Estudo de heurísticas matemáticas para o problema de escalonamento de máquina simples

Versão final

Dissertação apresentada ao Programa de Pós graduação em Engenharia Elétrica, área de concentração: Sistemas de Computação e Telecomunicações e linha de pesquisa: Otimização, da Escola de Engenharia da Universidade Federal de Minas Gerais como requisito parcial para obtenção do grau de Mestre.

Orientador: Prof. Eduardo Gontijo Carrano

Belo Horizonte
2019

A636e

Antunes, Natália.

Estudo de heurísticas matemáticas para o problema de escalonamento de máquina simples [recurso eletrônico] / Natália Antunes. - 2019.

1 recurso online (72 f. : il., color.) : pdf.

Orientador: Eduardo Gontijo Carrano.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f.69-72.

Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia Elétrica - Teses. 2. Programação heurística - Teses.
3. Modelos matemáticos – Teses. I. Carrano, Eduardo Gontijo.
II. Universidade Federal de Minas Gerais. Escola de Engenharia.
III. Título.

CDU: 621.3(043)


"Estudo de Heurísticas Matemáticas para o Problema de Escalonamento de Máquina Simples"

Natália Antunes

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 06 de dezembro de 2019.

Por:



Prof. Dr. Eduardo Gontijo Carrano
DEE (UFMG) - Orientador



Prof. Dr. Ricardo Hiroshi Caldeira Takahashi
DMAT (UFMG)



Prof. Dr. Lucas de Souza Batista
DEE (UFMG)

AGRADECIMENTOS

À Deus, meu primeiro agradecimento, que sempre está comigo e que tem me amparado até aqui.

Aos meus familiares, em especial meus pais, Assis e Denise, e meus irmãos Eduardo e Assis Gabriel, por todo amor, paciência, suporte, motivação e confiança.

Ao Igor pelo carinho, amor e cumplicidade que compartilhamos juntos, e pela capacidade de tornar meus dias mais felizes.

Ao meu orientador, Eduardo Gontijo Carrano, pelos ensinamentos compartilhados e principalmente pela ajuda no decorrer do trabalho. Gostaria de agradecer também aos amigos do mestrado, em especial a Isterândia.

Ao CNPQ, à Universidade Federal de Minas Gerais e ao Programa de Pós Graduação em Engenharia Elétrica pela estrutura necessária ao desenvolvimento deste trabalho.

RESUMO

Este trabalho apresenta um estudo de heurísticas matemáticas para o problema de sequenciamento de tarefas em uma única máquina com data de entrega comum, onde o objetivo é minimizar penalidades decorrentes de atrasos e adiantamentos nas entregas das tarefas. O problema é NP-difícil, tendo sido tratado por diferentes heurísticas e metaheurísticas ao longo do tempo. A finalidade do trabalho é desenvolver um método que combine metaheurísticas e algoritmos exatos, as chamadas heurísticas matemáticas. No decorrer deste trabalho foram definidos e validados dois modelos matemáticos que representam o problema. Ao todo, cinco vizinhanças com métodos de programação matemática foram implementadas utilizando estratégias de fixação de variáveis forte e fraca. As vizinhanças com estratégia de fixação forte foram inspiradas no algoritmo *Fix-and-Optimize* (FixOpt) e no algoritmo *Relaxation Induced Neighborhood Search* (RINS). A vizinhança com estratégia de fixação fraca implementada foi baseada no algoritmo *Local Branching* (LB). Dentre as vizinhanças implementadas, a inspirada no algoritmo RINS foi a que apresentou os melhores resultados, sendo utilizada em conjunto com a metaheurística *Variable Neighborhood Search* (VNS) para obtenção dos resultados finais. Os testes computacionais foram realizados utilizando instâncias *benchmark* do problema e os resultados obtidos no trabalho foram comparados com diferentes resultados reportados na literatura. As principais contribuições deste trabalho foram o estudo dos modelos matemáticos e a proposição de vizinhanças exatas para o problema de sequenciamento de tarefas em máquina simples com data de entrega comum.

Palavras-chave: Sequenciamento de tarefas, Data de entrega comum, Programação inteira mista, Heurística matemática, Metaheurística.

ABSTRACT

This document presents a study of matheuristics for the common due date single machine scheduling problem, where the goal is to minimize earliness and tardiness penalties in the delivery of jobs. The problem is NP-hard, which justifies proposals of heuristics and metaheuristics for solving it over the years. The purpose of the work is to develop a method that combines metaheuristics and exact algorithms, the so-called matheuristics. In the course of this work, two mathematical models for the problem were validated. In all, five exact neighborhoods were implemented using hard fixing and soft fixing. The neighborhoods with hard fixing were inspired in Fix-and-Optimize (FixOpt) and Relaxation Induced Neighborhood Search (RINS). The neighborhood with soft fixing were inspired in Local Branching (LB). Among the implemented neighborhoods, the neighborhood inspired in RINS had the best results and it was combined with Variable Neighborhood Search (VNS) to obtain the final results. Computational tests were performed using benchmark instances of the problem and the results obtained in the work were compared with different results reported in the literature. The main contributions of this work were the study of mathematical models and the proposition of exact neighborhoods for the single machine common due date scheduling problem.

Keywords: Scheduling, Common Due Date, Mixed Integer Programming, Matheuristics, Metaheuristic.

LISTA DE FIGURAS

Figura 1	Exemplo da obtenção de um vizinho da estrutura de vizinhança $V^{(1)}$	31
Figura 2	Exemplo da obtenção de um vizinho da estrutura de vizinhança $V^{(2)}$	32
Figura 3	Exemplo da obtenção de um vizinho da estrutura de vizinhança $V^{(3)}$	32
Figura 4	Exemplo da obtenção de um vizinho da estrutura de vizinhança $V^{(4)}$	33
Figura 5	Exemplo da vizinhança FixOpt 1	38
Figura 6	Exemplo da vizinhança FixOpt 2	40
Figura 7	Abordagem proposta	48

LISTA DE TABELAS

Tabela 1	Tempo médio (em segundos) dos modelos MIP 1 e MIP 2 para as instâncias com 10 tarefas	52
Tabela 2	Valores de função objetivo obtidos pelos modelos MIP 1 e MIP 2 para as instâncias com 20 tarefas limitadas em 5 minutos no software Gurobi	52
Tabela 3	Valores dos GAPs obtidos pelos modelos MIP 1 e MIP 2 para as instâncias com 20 tarefas limitadas em 5 minutos no software Gurobi .	52
Tabela 4	Valores médios de função objetivo obtidos pela vizinhança exata baseada em RINS	56
Tabela 5	Limitantes superiores obtidos pela vizinhança exata baseada em RINS com $h=0,2$ e $h=0,4$	57
Tabela 6	Limitantes superiores obtidos pela vizinhança exata baseada em RINS com $h=0,6$ e $h=0,8$	58
Tabela 7	Tempo médio (em segundos) da vizinhança exata baseada em RINS	59
Tabela 8	Valores médios de função objetivo obtidos pelo VNS	62
Tabela 9	Limitantes superiores obtidos pelo VNS com $h=0,2$ e $h=0,4$	63
Tabela 10	Limitantes superiores obtidos pelo VNS com $h=0,6$ e $h=0,8$	64
Tabela 11	Tempo médio (em segundos) do VNS	65
Tabela 12	Valores de DPR obtidos por algoritmos reportados na literatura ...	66

LISTA DE ABREVIATURAS E SIGLAS

BA *Bees Algorithm*

BB *Branch-and-Bound*

DACO *Dynamical Ant Colony Optimization*

DDE *Discrete Differential Evolution*

DE *Differential Evolution*

DPR *Desvio Percentual Relativo*

DPSO *Discrete Particle Swarm Optimization*

ES *Evolutionary Strategy*

FixOpt *Fix-and-Optimize*

GA *Genetic Algorithm*

GBA *Genetic Bees Algorithm*

HOPS *Hamming Oriented Partition Search*

HS *Harmony Search*

JIT *Just-in-Time*

LB *Local Branching*

MILP *Mixed Integer Linear Programming*

MSE *Memorial Self Evolution*

RINS *Relaxation Induced Neighborhood Search*

SA *Simulated Annealing*

SE *Self Evolution*

SEA *Sequential Exchange Approach*

SMCDD *Single Machine Common Due Date*

SS *Scatter Search*

PHVNS *Permutation-Based Harmony Variable Neighborhood Search*

TA *Threshold Accepting*

TS *Tabu Search*

VND *Variable Neighborhood Descent*

VNS *Variable Neighborhood Search*

SUMÁRIO

1	Introdução	14
1.1	Objetivos	15
1.1.1	Objetivo geral	15
1.1.2	Objetivos específicos	15
1.2	Revisão da Literatura	15
1.3	Estrutura do Trabalho	20
2	O problema SMCDD	22
2.1	Formulações matemáticas	24
2.1.1	MIP 1	24
2.1.2	MIP 2	25
3	Algoritmos	28
3.1	Solução inicial	28
3.2	Metaheurística	30
3.2.1	Estruturas de vizinhança	30
3.2.2	VNS	33
3.2.3	Busca local	35
3.3	Heurísticas matemáticas	36
3.3.1	Fix-and-Optimize	37
3.3.1.1	Vizinhança FixOpt 1	38
3.3.1.2	Vizinhança FixOpt 2	40
3.3.1.3	Vizinhança FixOpt 3	41

3.3.2	Relaxation Induced Neighborhood Search (RINS)	43
3.3.2.1	Vizinhança exata baseada no algoritmo RINS	43
3.3.3	Local Branching	44
3.4	Abordagem proposta	48
4	Resultados Computacionais	50
4.1	Ambiente computacional	50
4.2	Caracterização das instâncias	50
4.3	Validação dos modelos matemáticos	50
4.4	Vizinhança exata baseada em RINS	54
4.5	VNS	59
4.6	Comparação com a literatura	60
5	Considerações Finais	67
5.1	Trabalhos Futuros	68
5.2	Publicações	68
	Referências	69

1 INTRODUÇÃO

Com a crescente concorrência dos mercados internacionais, as empresas precisam oferecer uma grande variedade de produtos aos clientes em prazos pré-determinados. Tendo em vista esta necessidade, alguns princípios e filosofias foram desenvolvidos, tais quais Gestão Administrativa Enxuta - *Lean Management*, Engenharia Simultânea - *Simultaneous Engineering*, Produção no Momento Certo - *Just-in-Time* (JIT), entre outros. O problema de agendamento de tarefas em única máquina com data de entrega comum - *Single Machine Common Due Date* (SMCDD) se alinha com a filosofia JIT, que determina que a quantidade certa de mercadorias deve ser produzida ou entregue exatamente na hora certa, sendo atrasos e adiantamentos vistos de maneira negativa.

No problema SMCDD, somente uma tarefa pode ser concluída exatamente na data de entrega comum. As outras tarefas são finalizadas antes ou depois da data de entrega comum. Entretanto, uma conclusão antecipada da tarefa incorre em uma penalidade por adiantamento e os adiantamentos podem resultar em custos com seguro, deterioração e armazenamento. Por sua vez, uma conclusão tardia da tarefa resulta em uma penalidade por atraso e os atrasos podem causar insatisfação do cliente, penalidades contratuais, perdas de vendas e redução de reputação (LIAO & CHENG, 2007). Sendo assim, o objetivo do problema de agendamento com data de entrega comum é encontrar um agendamento que minimize a soma dos custos de adiantamento e atraso na entrega das tarefas.

Abordagens envolvendo atrasos e adiantamentos em problemas de sequenciamento de produção têm recebido bastante atenção nos últimos anos e diversos algoritmos foram implementados para a resolução do problema SMCDD. Devido à sua natureza NP-difícil (HOOGEVEEN & VELDE, 1991; HALL; KUBIAK & SETHI, 1991), heurísticas e metaheurísticas são as técnicas mais recorrentes, pois apresentam soluções com boa qualidade em tempo computacional razoável.

O presente trabalho propõe o uso de heurísticas matemáticas, técnica que combina metaheurísticas com métodos de programação matemática, para obter a solução

do problema SMCDD. As abordagens combinando os dois métodos ganharam espaço principalmente por causa do aumento na disponibilidade de softwares de otimização, bibliotecas e *frameworks* para programação matemática (TALBI, 2013).

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

O objetivo geral deste trabalho é desenvolver uma heurística matemática para solucionar o problema de sequenciamento de tarefas em uma única máquina com data de entrega comum. O algoritmo proposto deve apresentar um bom desempenho e os testes serão realizados utilizando um conjunto de instâncias *benchmark* disponível na literatura.

1.1.2 OBJETIVOS ESPECÍFICOS

Com a finalidade de alcançar o objetivo geral, os seguintes objetivos específicos são definidos:

- Definir e validar modelos matemáticos que irão representar o problema SMCDD;
- Implementar diferentes vizinhanças utilizando programação matemática, verificando a formulação matemática mais eficiente para cada uma delas;
- Desenvolver uma ferramenta de otimização capaz de solucionar o problema com um tempo computacional satisfatório;
- Avaliar a eficiência do algoritmo através da realização de testes com instâncias *benchmark* do problema;
- Comparar os resultados obtidos com outros métodos presentes na literatura.

1.2 REVISÃO DA LITERATURA

Problemas de agendamento de tarefas têm sido estudado por muito tempo e um dos pioneiros em abordagens considerando data de entrega comum foi Kanet (1981), que propôs um algoritmo para minimizar a soma dos desvios com relação a data de entrega comum considerando as penalidades por atraso e adiantamento das tarefas iguais à um. Desde seu trabalho, muitos algoritmos foram propostos para várias versões do

problema básico. Alguns exemplos são os trabalhos de Hall (1986) e Bagchi, Sullivan e Chang (1986, 1987), que propuseram algoritmos para minimizar a soma do desvio absoluto dos tempos de conclusão das tarefas.

O valor da data de entrega comum foi estudado em alguns trabalhos, pois influencia na complexidade do problema. A data de entrega comum é considerada restrita quando a sequência ótima não pode ser construída sem considerar o seu valor e, considerada irrestrita, caso contrário. Os trabalhos de Hall e Posner (1991), De, Ghosh e Wells (1994), Hoogeveen e Velde (1997), Panwalkar, Smith e Seidmann (1982) e Lee, Danusaputro e Lin (1991) trataram o problema considerando a data de entrega comum irrestrita. Outros trabalhos, realizados antes de 2002, que estudaram problemas de sequenciamento com data de entrega comum podem ser encontrados nas revisões da literatura realizadas por Cheng e Gupta (1989), Baker e Scudder (1990) e Gordon, Proth e Chu (2002).

Um marco na resolução do problema SMCDD considerando penalidades por atrasos e adiantamentos ocorreu quando Biskup e Feldmann (2001) criaram um gerador de problemas para geração de instâncias para o problema SMCDD, que resultou na criação de 280 instâncias. Estas instâncias têm sido utilizadas desde então para avaliar o desempenho dos algoritmos propostos. Além do gerador de problema, Biskup e Feldmann propuseram um modelo matemático para o problema e desenvolveram duas heurísticas. As heurísticas foram utilizadas para identificar limitantes superiores para as instâncias e diversas comparações com os valores obtidos por estas heurísticas foram realizadas ao longo dos anos.

Os trabalhos que serão descritos a seguir utilizam heurísticas e metaheurísticas para resolver o problema SMCDD. O primeiro deles é o de Feldmann e Biskup (2003), onde três metaheurísticas são propostas: estratégia evolucionária - *Evolutionary Strategy* (ES), recozimento simulado - *Simulated Annealing* (SA) e limite de aceitação - *Threshold Accepting* (TA). Modificações destas metaheurísticas também foram implementadas e, ao todo, cinco abordagens foram realizadas: ES, ESD (ES com uma fase de desestabilização), TA, TAR (TA com um “passo atrás”) e SA. Os resultados de todas as abordagens foram próximos, entretanto a metaheurística TAR apresentou um desempenho ligeiramente melhor.

O trabalho de Hino, Ronconi e Mendes (2005) propôs uma abordagem híbrida utilizando busca tabu - *Tabu Search* (TS) e algoritmo genético - *Genetic Algorithm* (GA). As metaheurísticas foram analisadas individualmente e combinadas de forma

sequencial para obtenção dos resultados. As duas versões híbridas do algoritmo foram a HTG, onde a busca tabu é utilizada seguida do algoritmo genético, e a HGT, onde o algoritmo genético é utilizado seguido da busca tabu. Os resultados obtidos pelos quatro algoritmos propostos foram similares, entretanto, as versões híbridas foram mais eficientes do ponto de vista de esforço computacional.

Pan, Tasgetiren e Liang (2006) utilizaram o algoritmo de otimização por enxame de partículas discreto - *Discrete Particle Swarm Optimization* (DPSO) em conjunto com um algoritmo de busca de vizinhança. A busca local foi realizada utilizando o movimento *swap*, onde duas tarefas da sequência trocam suas posições. Nesta abordagem, as tarefas foram escolhidas aleatoriamente, sendo uma delas atrasada e a outra adiantada.

O trabalho de Lin, Chou e Chen (2007) utiliza os algoritmos GA e SA combinados com uma busca local gulosa - *greedy local search*. A busca local gulosa consiste em selecionar uma tarefa adiantada ou uma tarefa atrasada e efetuar a troca com a tarefa do outro conjunto que conduz à maior melhoria no valor de função objetivo.

Lin, Chou e Ying (2007) propuseram um algoritmo denominado de abordagem de troca sequencial - *Sequential Exchange Approach* (SEA). Neste algoritmo, as tarefas são alocadas em dois conjuntos, atrasadas e adiantadas. As tarefas adiantadas são selecionadas uma a uma e efetua-se a troca com a tarefa do conjunto das tarefas atrasadas que conduz à maior melhoria no valor de função objetivo. O mesmo processo ocorre com cada uma das tarefas do conjunto das atrasadas.

O trabalho de Pham et al (2007) utiliza o algoritmo das abelhas - *Bees Algorithm* (BA) para solucionar o problema SMCDD. Este algoritmo é inspirado no comportamento natural das abelhas e executa uma espécie de busca local combinada com uma busca aleatória. O artigo é o primeiro a relatar a aplicação do algoritmo de abelhas em um problema combinatório.

Liao e Cheng (2007) propuseram um algoritmo híbrido que combina busca tabu e busca em vizinhança variável - VNS. O VNS implementado utiliza diferentes abordagens dependendo do valor da data de entrega comum e a busca tabu é usada para evitar que o algoritmo fique preso em um mínimo local. Duas listas tabu foram criadas, uma para cada estrutura de vizinhança do VNS. Os resultados mostraram que a versão híbrida forneceu melhores valores de função objetivo se comparado com o valor obtido somente pelo VNS.

O trabalho de Tasgetiren et al (2007) apresenta um algoritmo de evolução diferen-

cial discreto - *Discrete Differential Evolution* (DDE). Um novo operador de mutação binário denominado de *Bswap* é proposto. Esta abordagem utiliza uma busca local para refinar o resultado obtido pelo DDE baseada no movimento *swap*. Nearchou (2008) também utiliza um algoritmo de evolução diferencial - *Differential Evolution* (DE) para solucionar o problema SMCDD. A abordagem considera as propriedades do problema e, no conjunto de soluções, todas as soluções são ordenadas em forma de V^1 .

Em 2008, os autores Weng e Fujimura propuseram duas abordagens para o problema. A primeira pode ser encontrada no trabalho Weng e Fujimura (2008a) e utiliza um algoritmo de autoevolução com memória - *Memorial Self Evolution* (MSE). Neste tipo de algoritmo, somente a mutação é aplicada e cada indivíduo evolui a partir de suas características. Um banco de dados é utilizado para armazenar as melhores soluções e estas são usadas como soluções iniciais na próxima execução do algoritmo. A segunda abordagem pode ser encontrada no trabalho Weng e Fujimura (2008b) e os autores utilizaram o algoritmo de autoevolução - *Self Evolution* (SE) para solucionar o problema SMCDD. A abordagem deste algoritmo é similar ao do outro trabalho, entretanto as melhores soluções não são armazenadas. As duas abordagens obtiveram resultados parecidos, sendo que a MSE obteve um desempenho ligeiramente melhor.

Lee, Lin e Ying (2008) utilizaram o algoritmo de otimização por colônia de formigas dinâmico - *Dynamical Ant Colony Optimization* (DACO) para solucionar o problema SMCDD. Neste algoritmo, um parâmetro da heurística é ajustado dinamicamente. Além disto, heurísticas adicionais são incorporadas ao DACO como busca local para escapar de ótimos locais.

O trabalho Talebi et al (2009) apresenta um algoritmo de busca dispersa - *Scatter Search* (SS). Este algoritmo apresenta cinco etapas: gerador de diversificação, método de melhoria, atualização do conjunto de referência, geração do subconjunto e método de combinação das soluções. Esta abordagem apresentou os melhores resultados reportados na literatura para instâncias com menos de 100 tarefas e com data de entrega comum² calculada com $h=0,6$ e $h=0,8$.

Liu e Zhou (2013) propuseram uma metaheurística híbrida denominada *Permutation-Based Harmony Variable Neighborhood Search* (PHVNS) que combina um algoritmo

¹As tarefas que estão adiantadas são sequenciadas em ordem decrescente de p_i/α_i e as tarefas que estão atrasadas são sequenciadas em ordem crescente de p_i/β_i . Sendo que p_i é o tempo de processamento da tarefa i , α_i é a penalidade por adiantamento da tarefa i e β_i é a penalidade por atraso da tarefa i .

²Nas instâncias *benchmark* do problema, o valor de data de entrega em comum é calculado para quatro valores de h , que são iguais à $h = 0,2, 0,4, 0,6$ e $0,8$.

de busca harmônica - *Harmony Search* (HS) com uma variante do VNS. O algoritmo utiliza uma permutação baseada no algoritmo HS, que é combinado com o VNS para coordenar a busca com diversificação e intensificação. Duas estratégias de busca local específicas do problema SMCDD são utilizadas para percorrer regiões com soluções mais promissoras.

Finalizando as abordagens de heurísticas e metaheurísticas na resolução do problema SMCDD, o trabalho de Yuce et al (2017) apresenta uma versão híbrida do algoritmo das abelhas, chamado algoritmo genético das abelhas - *Genetic Bees Algorithm* (GBA). Os operadores dos algoritmos genéticos, cruzamento e mutação, são incluídos nesta versão para aumentar a taxa de convergência, pois o algoritmo básico das abelhas pode ter limitações para convergir na escala de tempo desejada.

Os trabalhos descritos anteriormente demonstram que diversas heurísticas e metaheurísticas foram empregadas na resolução do problema SMCDD ao longo dos anos. Várias estratégias diferentes foram exploradas, inclusive com abordagens utilizando algoritmos híbridos. O objetivo deste trabalho é estudar heurísticas matemáticas e aplicá-las ao problema. Devido à evolução das ferramentas de otimização, a técnica tem ganhado espaço nos últimos anos e aplicações de heurísticas matemáticas em problemas de *scheduling* podem ser encontradas na literatura com resultados relevantes. Alguns destes trabalhos são citados a seguir.

O trabalho de Croce, Salassa e Tkindt (2014) utiliza heurísticas matemáticas na resolução de um problema de agendamentos de tarefas em uma única máquina, onde a tarefa apresenta um tempo de processamento e uma hora de liberação. O algoritmo Fixa e Otimiza - FixOpt foi utilizado para a resolução do problema obtendo bons resultados.

Camargo, Toledo e Almada-Lobo (2014) utilizaram heurística matemática para resolver um problema de dimensionamento e agendamento de dois estágios em uma fábrica de fiação. O método utilizado é denominado *Hamming Oriented Partition Search* (HOPS) e consiste em um procedimento que utiliza o algoritmo *Branch-and-Bound* (BB) e incorpora uma melhoria baseada em FixOpt. Um dos diferenciais do algoritmo é a proposição de uma partição para determinar as variáveis que serão fixadas no FixOpt; a partição utiliza um ranking que considera a frequência das mudanças das variáveis nas iterações anteriores. Os resultados obtidos pelo HOPS foram melhores que os reportados na literatura anteriormente.

No trabalho de Lin e Wing (2016), um problema de otimização de makespan sem

espera entre as tarefas e com várias máquinas foi resolvido. O problema foi convertido em um problema de caixeiro viajante assimétrico e um algoritmo foi usado para melhorar a solução inicial. Em seguida, a solução obtida anteriormente foi definida como limite superior do modelo matemático correspondente ao problema do caixeiro viajante assimétrico transformado e o modelo resultante foi resolvido através de um software de programação inteira. Os resultados obtidos por esta abordagem foram melhores que os reportados na literatura anteriormente.

Toffolo et al. (2016) resolveram um problema de agendamento multiprojeto de recursos multimodo cujo objetivo é determinar o agendamento de várias tarefas dentro de um limite estipulado com recursos renováveis e não-renováveis. A solução inicial do problema é construída utilizando uma heurística de programação inteira e refinada com uma busca local híbrida. A busca local consiste em um método de melhoria para frente e para trás (*forward-backward*) híbrido com um modelo de programação inteira para mudar os modos de execução da tarefa e um procedimento de reconstrução parcial. Os resultados obtidos foram melhores que os reportados na literatura anteriormente.

Tendo em vista os bons resultados obtidos através das abordagens com heurísticas matemáticas em problemas de *scheduling* e as poucas contribuições utilizando esta técnica no problema SMCDD, neste trabalho será realizado um estudo de heurísticas matemáticas para o problema. Os resultados obtidos neste trabalho serão comparados com outros trabalhos da literatura que propuseram algoritmos para a resolução do problema SMCDD.

1.3 ESTRUTURA DO TRABALHO

O Capítulo 1 apresentou uma introdução sobre este trabalho, onde os principais objetivos foram apresentados e foi realizada uma contextualização do problema resolvido. Também foi realizada uma revisão da literatura e as principais contribuições e técnicas utilizadas para a resolução do problema SMCDD foram destacadas.

No Capítulo 2 são apresentadas as características do problema SMCDD e as premissas utilizadas para a sua resolução. Além disso, duas formulações matemáticas do problema são estudadas.

No Capítulo 3 são mostrados os algoritmos que foram implementados neste trabalho: a heurística utilizada para a construção da solução inicial, a metaheurística VNS e as vizinhanças exatas. A abordagem proposta para a resolução do problema também

é apresentada.

No Capítulo 4 são apresentados e discutidos os resultados obtidos e o desempenho dos modelos matemáticos é analisado. A eficiência da abordagem proposta é testada utilizando instâncias *benchmark* disponíveis na literatura e os resultados obtidos são comparados com os reportados em diferentes trabalhos.

Por fim, no Capítulo 5 são apresentadas as principais conclusões do trabalho e sugestões de trabalhos futuros.

2 O PROBLEMA SMCDD

O problema SMCDD tem a finalidade de determinar o sequenciamento de um conjunto de tarefas que apresentam uma data de entrega comum e devem ser processadas por uma única máquina. O objetivo é minimizar a soma das penalidades decorrentes de atrasos ou adiantamentos na entrega das tarefas.

A data de entrega em comum, d , influencia na complexidade do problema e pode ser uma variável de decisão (cujo valor deve ser determinado na otimização) ou pode ser conhecida a priori. Neste trabalho, d é calculada como em Biskup e Feldmann (2001), onde foram propostas as instâncias *benchmark* consideradas para este problema. O valor de d na referência é um parâmetro de entrada e é restritiva, conforme a Equação 2.1,

$$d = \left\lceil h \sum_{i=1}^N p_i \right\rceil, \quad (2.1)$$

onde d equivale ao maior número inteiro menor ou igual a $h \sum_{i=1}^N p_i$, h é um escalar entre 0 e 1, p_i é o tempo de processamento da tarefa i e N é a quantidade de tarefas. A variação do valor de h na Equação 2.1 torna o problema mais ou menos restritivo. Na resolução das instâncias *benchmark* são utilizados os valores de $h = 0,2, 0,4, 0,6$ e $0,8$.

Neste problema, um conjunto de N tarefas devem ser processadas por uma única máquina a partir de um tempo inicial zero. Todas as tarefas estão disponíveis no tempo zero, são independentes e executadas em uma única operação da máquina. Além disso, não existe tempo ocioso entre as tarefas e não é permitido preempção (após iniciar o processamento da tarefa, não é permitido interrompê-lo). Sendo assim, somente uma tarefa pode ser concluída na data de entrega comum, sendo as outras tarefas finalizadas antes ou depois dela.

Não é possível determinar previamente se uma tarefa será entregue antes ou depois da data de entrega comum, então calcula-se o adiantamento e o atraso para todas as

tarefas. Se o tempo de conclusão da tarefa i , C_i , é menor ou igual a d , a tarefa está adiantada e o adiantamento é dado por $E_i = \max\{d - C_i, 0\}$. Se o tempo de conclusão da tarefa i for maior que d , a tarefa está atrasada e o atraso é dado por $T_i = \max\{C_i - d, 0\}$.

Além do tempo de processamento, p_i , as tarefas apresentam penalidades por adiantamento e atraso, α_i e β_i , respectivamente. Essas penalidades são definidas por unidade de tempo e também podem influenciar na complexidade do problema. Um estudo da influência de α_i e β_i no problema SMCDD pode ser encontrado no trabalho de Biskup e Feldmann (2001).

O objetivo do problema é minimizar simultaneamente as penalidades por atraso e adiantamento, sendo a função objetivo dada pela Equação 2.2,

$$f(S) = \sum_{i=1}^N \alpha_i E_i + \sum_{i=1}^N \beta_i T_i \quad (2.2)$$

onde S representa um cronograma viável das tarefas. O cronograma S contém todas as informações necessárias para a produção das tarefas: a sequência π de produção, o tempo de início da primeira tarefa e a data de entrega em comum.

A sequência ótima apresenta algumas propriedades que podem auxiliar na resolução e formulação do problema. As propriedades da sequência ótima quando d é restritiva são:

- Propriedade 1: na sequência ótima não existe tempo ocioso entre o processamento de duas tarefas consecutivas.
- Propriedade 2: na sequência ótima, as tarefas são ordenadas em forma de V, onde as tarefas que estão adiantadas são sequenciadas em ordem decrescente de p_i/α_i e as tarefas que estão atrasadas são sequenciadas em ordem crescente de p_i/β_i .
- Propriedade 3: existe uma sequência ótima em que o processamento da primeira tarefa inicia no instante de tempo zero ou uma tarefa termina o seu processamento exatamente na data de entrega comum.

As demonstrações destas propriedades podem ser vistas nos trabalhos de Cheng e Kahlba-Cher (1991), Baker e Scudder (1990) e Hoogeveen e Velde (1991).

2.1 FORMULAÇÕES MATEMÁTICAS

Nessa seção serão descritos os modelos matemáticos para o problema SMCDD que foram utilizados neste trabalho. Estes modelos apresentam abordagens diferentes do problema, com parte das variáveis de decisão distintas, o que pode ser interessante para a elaboração de vizinhanças. As diferentes características das variáveis de decisão oferecem uma flexibilidade na fixação das variáveis no modelo matemático, permitindo que vizinhanças distintas sejam implementadas.

Além disso, diferentes modelos matemáticos podem apresentar tempos de execução distintos para uma mesma vizinhança. Isso ocorre devido às características do modelo. Sendo assim, é interessante que as vizinhanças sejam implementadas com os diferentes modelos, de forma que o modelo escolhido seja aquele que apresente o melhor desempenho.

O primeiro modelo matemático de programação linear inteira mista - *Mixed Integer Linear Programming* (MILP) descrito neste trabalho é um modelo de precedência e será descrito ao longo do trabalho como MIP 1. Já o segundo, é um modelo posicional e será referenciado ao longo do trabalho como MIP 2.

Algumas informações sobre as tarefas são disponibilizadas a priori, já que constituem os parâmetros de entrada do problema:

α_i	Penalidade por adiantamento da tarefa i
β_i	Penalidade por atraso da tarefa i
p_i	Tempo de processamento da tarefa i
d	Data de entrega em comum (<i>due-date</i>)

2.1.1 MIP 1

O modelo matemático MIP 1 é apresentado por Biskup e Feldmann (2001) e, nesta formulação, o tempo de início de cada uma das tarefas é determinado com o auxílio de uma variável binária, x_{ij} , que assume valor 1 quando a tarefa i é processada antes da tarefa j (não necessariamente imediatamente) e assume 0 caso contrário. As variáveis do modelo são:

T_i	Atraso da tarefa i
E_i	Adiantamento da tarefa i
s_i	Instante de início da tarefa i
x_{ij}	Variável binária que indica a precedência ou não entre duas tarefas

A restrição 2.3 determina o atraso e o adiantamento de cada uma das N tarefas.

$$T_i - E_i = s_i + p_i - d \quad \forall i \in N \quad (2.3)$$

As restrições 2.4 e 2.5 determinam o instante de início de cada uma das tarefas. Essas restrições são disjuntivas e garantem que ou a tarefa i será processada antes da tarefa j ou a tarefa j será processada antes da tarefa i . Nesta formulação, geralmente o valor de M é igual a soma do tempo de processamento de todas as tarefas.

$$s_i + p_i \leq s_j + M(1 - x_{ij}) \quad i = 1, \dots, N-1; j = i+1, \dots, N \quad (2.4)$$

$$s_j + p_j \leq s_i + Mx_{ij} \quad i = 1, \dots, N-1; j = i+1, \dots, N \quad (2.5)$$

Por fim, as restrições 2.6 e 2.7 são restrições de não negatividade e integralidade.

$$E_i, T_i, s_i \geq 0 \quad \forall i, j \in N \quad (2.6)$$

$$x_{ij} \in \{0,1\} \quad \forall i, j \in N \quad (2.7)$$

2.1.2 MIP 2

O modelo matemático MIP 2 é baseado no modelo matemático posicional apresentado por Keha, Khowala e Fowler (2009) para diferentes problemas de agendamento de tarefas em uma única máquina. Algumas alterações foram realizadas no modelo original para adequação ao objetivo do problema SMCDD. Nesta formulação, o tempo de conclusão de cada uma das tarefas é determinado com o auxílio de uma variável binária, x_{ij} , que equivale a 1 se a tarefa i é atribuída à posição j da sequência ou 0 caso contrário. As variáveis do modelo são:

T_i	Atraso da tarefa i
E_i	Adiantamento da tarefa i
C_i	Tempo de conclusão da tarefa i
γ_j	Tempo de conclusão da tarefa na posição j
x_{ij}	Variável binária que determina a posição j em que a tarefa i está alocada

As restrições 2.8 e 2.9 determinam o atraso e o adiantamento, respectivamente, de cada uma das N tarefas.

$$T_i \geq C_i - d \quad \forall i \in N \quad (2.8)$$

$$E_i \geq d - C_i \quad \forall i \in N \quad (2.9)$$

As restrições 2.10 e 2.11 são necessárias para garantir que as tarefas sejam alocadas em apenas uma posição e que cada posição receba somente uma tarefa.

$$\sum_{j \in N} x_{ij} = 1 \quad \forall i \in N \quad (2.10)$$

$$\sum_{i \in N} x_{ij} = 1 \quad \forall j \in N \quad (2.11)$$

As restrições 2.12 e 2.13 determinam o tempo de conclusão da tarefa na posição j .

$$\gamma_1 \geq \sum_{i \in N} p_i x_{i1} \quad (2.12)$$

$$\gamma_j \geq \gamma_{j-1} + \sum_{i \in N} p_i x_{ij} \quad j = 2, \dots, N \quad (2.13)$$

Para definir os valores do atraso e do adiantamento é necessário determinar o tempo de conclusão da tarefa i . As restrições 2.14 e 2.15 determinam os tempos de conclusão de cada uma das N tarefas. Nesta formulação, o valor de M deve ser maior que a soma do tempo de processamento de todas as tarefas.

$$C_i \geq \gamma_j - M(1 - x_{ij}) \quad \forall i, j \in N \quad (2.14)$$

$$\sum_{j \in N} \gamma_j = \sum_{i \in N} C_i \quad (2.15)$$

Por fim, as restrições 2.16 e 2.17 são restrições de não negatividade e integralidade.

$$E_i, T_i, C_i, \gamma_j \geq 0 \quad \forall i, j \in N \quad (2.16)$$

$$x_{ij} \in \{0,1\} \quad \forall i, j \in N \quad (2.17)$$

3 ALGORITMOS

Este capítulo apresenta os algoritmos implementados neste trabalho. A Seção 3.1 apresenta a heurística utilizada para gerar a solução inicial. A Seção 3.2 apresenta a metaheurística VNS implementada e as estruturas de vizinhanças utilizadas. E a Seção 3.3 apresenta as vizinhanças com algoritmos exatos implementadas neste trabalho.

3.1 SOLUÇÃO INICIAL

A solução inicial foi gerada utilizando uma das heurísticas propostas por Biskup e Feldmann (2001), a Heurística II. A ideia desta heurística é sequenciar primeiramente as tarefas com penalidades de atraso relativamente altas. O pseudocódigo da solução inicial pode ser visto no Algoritmo 1.

Os parâmetros de entrada do algoritmo são as penalidades por adiantamento (α) e por atraso (β), o tempo de processamento das tarefas (p), a quantidade de tarefas (N) e a data de entrega comum (d). Os parâmetros de saída são a sequência das tarefas (s), o valor de função objetivo (f) e o instante de início do processamento da primeira tarefa (t_0). Ao final desta heurística são criados dois conjuntos, o $conjA$ e o $conjB$, sendo o $conjA$ formado pelas tarefas atrasadas e o $conjB$ pelas tarefas adiantadas. O gap, nesta heurística, corresponde ao instante de início do processamento da primeira tarefa e é determinado através da Equação 3.1.

$$gap = d - \sum_{i \in conjB} p_i \quad (3.1)$$

Inicialmente, as tarefas são ordenadas em ordem decrescente de β_i/α_i (linha 2) e, caso existam tarefas em que o valor de β_i/α_i é o mesmo, estas são ordenadas em ordem decrescente de β_i . Em seguida, a primeira tarefa atribuída à Pab é adicionada ao $conjB$, excluída de Pab e o valor do gap é calculado através da Equação 3.1 (linhas 6 à 8). Esse processo é repetido até que: (a) o valor do gap seja pequeno para alocar

a primeira tarefa de Pab ou (b) $N/2$ tarefas tenham sido adicionadas ao $conjB$.

Algoritmo 1: Solução inicial

Entrada: α, β, p, N, d

Saída: $s, f, t0$

```

1  início
2     $Pab \leftarrow \text{sort}(\beta_i/\alpha_i, \text{decrecente})$  ;
3     $conjB \leftarrow [ ]$ ;
4     $conjA \leftarrow [ ]$ ;
5    repita
6       $conjB \leftarrow Pab(1)$  ;
7      Calcule o  $gap$ ;
8      Exclua  $Pab(1)$  de  $Pab$ ;
9    até  $gap < p(Pab(1)) \vee |conjB| > N/2$ ;
10   se  $0 < gap < p(Pab(1)) \wedge |conjB| < N/2$  então
11     repita
12        $S \leftarrow \{\exists i \in Pab \mid p(Pab(i)) \leq gap\}$ ;
13        $conjB \leftarrow S(1)$  ;
14       Calcule o  $gap$ ;
15       Exclua  $S(1)$  de  $Pab$ ;
16     até  $S \leftarrow \emptyset \vee |conjB| \geq N/2$ ;
17   fim
18    $conjA \leftarrow Pab$ ;
19    $s \leftarrow [conjB \ conjA]$  em forma de V;
20    $t0 \leftarrow gap$ ;
21   se  $|conjB| \geq N/2$  então
22     repita
23        $s' \leftarrow s$ ;
24        $conjB \leftarrow Pab(1)$ ;
25       Exclua  $Pab(1)$  de  $Pab$ ;
26        $conjA \leftarrow Pab$ ;
27        $s \leftarrow [conjB \ conjA]$  em forma de V ;
28        $t0 \leftarrow gap$ ;
29     até  $f(s) > f(s')$ ;
30   fim
31 fim

```

Caso ocorra a situação (a) e o valor do gap seja maior que zero (linha 10), é realizada uma busca das tarefas de Pab cujo valor de p é menor que o gap (linha 12); a primeira tarefa encontrada é movida para o $conjB$ (linhas 13 à 15); este processo é repetido até que nenhuma tarefa tenha p menor que o gap ou que $N/2$ tarefas tenham sido adicionadas ao $conjB$ (linha 16). As tarefas remanescentes do conjunto Pab são alocadas no $conjA$ (linha 18). As tarefas nos $conjA$ e $conjB$ são sequenciadas em forma de V e o valor do gap é atribuído à $t0$ (linhas 19 à 20).

Se $N/2$ tarefas forem adicionadas ao $conjB$ (situação (b)), uma busca local é efetuada: a primeira tarefa do conjunto Pab é adicionada ao $conjB$ se o valor de função objetivo melhorar com a adição (linhas 23 à 28). O processo se repete até que o valor de função objetivo piore (linha 29).

3.2 METAHEURÍSTICA

Nesta seção são apresentadas a metaheurística implementada neste trabalho, a Busca em Vizinhança Variável - VNS, e os movimentos que definem as estruturas de vizinhança utilizadas.

3.2.1 ESTRUTURAS DE VIZINHANÇA

As estruturas de vizinhança são formadas por todas as soluções obtidas a partir de determinado movimento. Os movimentos utilizados neste trabalho serão descritos a seguir e o primeiro passo para determiná-los consiste em dividir a solução s em dois conjuntos, sendo um conjunto formado pelas tarefas atrasadas, o S_T , e o outro conjunto pelas tarefas adiantadas, o S_E .

Neste trabalho foram implementadas quatro estruturas de vizinhança, $V^{(1)}$, $V^{(2)}$, $V^{(3)}$ e $V^{(4)}$, que estão ordenadas conforme foram utilizadas no VNS.

- **Realocação de uma tarefa em outro conjunto**

Neste movimento, uma tarefa da sequência é realocada. Caso a tarefa esteja adiantada na solução incumbente, ela é inserida no final de S_T . Se a tarefa estiver atrasada, ela é inserida no início de S_E .

A Figura 1 ilustra um exemplo do funcionamento deste movimento. Nesta figura, um vizinho da solução incumbente, representada na Figura 1(a), é obtido. Inicialmente,

a solução incumbente é dividida nos conjuntos S_T e S_E , sendo que S_E está representado em amarelo e S_T está representado em rosa na Figura 1(a). Em seguida, uma tarefa é selecionada para ser realocada em outro conjunto, a tarefa selecionada está representada na Figura 1(b) em vermelho. Como a tarefa selecionada está atrasada, ela é realocada no início de S_E , conforme pode ser visto na Figura 1(c) em cinza. Por fim, após a realocação, a sequência obtida é reordenada em forma de V (conforme a propriedade 2 descrita no Capítulo 2) e o vizinho, representado na Figura 1(d), é gerado.

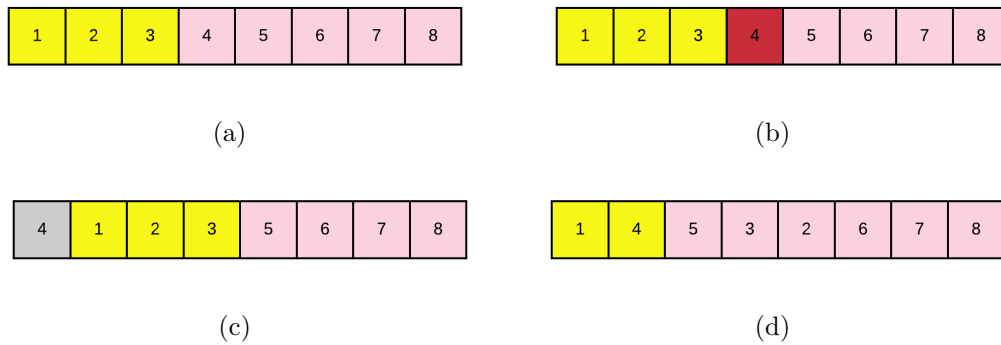


Figura 1: Exemplo da obtenção de um vizinho da estrutura de vizinhança $V^{(1)}$

A cardinalidade da vizinhança $V^{(1)}$ é $O(N)$, onde N é a quantidade de tarefas que serão otimizadas.

- **Troca de ordem de duas tarefas de conjuntos diferentes**

Neste movimento, duas tarefas trocam suas posições, sendo que uma delas deve pertencer à S_E e a outra deve pertencer à S_T . A tarefa adiantada é inserida no final de S_T e a tarefa atrasada é inserida no início de S_E .

A Figura 2 ilustra um exemplo do funcionamento deste movimento. Nesta figura, um vizinho da solução incumbente, representada na Figura 2(a), é obtido. Inicialmente, a solução incumbente é dividida nos conjuntos S_T e S_E , sendo que S_E está representado em amarelo e S_T está representado em rosa na Figura 2(a). Em seguida, duas tarefas são selecionadas, estas estão representadas na Figura 2(b) em vermelho. A tarefa atrasada é realocada no início de S_E e a tarefa adiantada é realocada no final de S_T , conforme pode ser visto na Figura 2(c) em cinza. Por fim, após a realocação, a sequência obtida é reordenada em forma de V e o vizinho, representado na Figura 2(d), é gerado.

A cardinalidade da estrutura de vizinhança $V^{(2)}$ depende da quantidade de tarefas que estão alocadas em S_T e S_E e, no pior caso, quando os dois conjuntos têm a mesma quantidade de tarefas, é dada por $O(N^2/4)$.

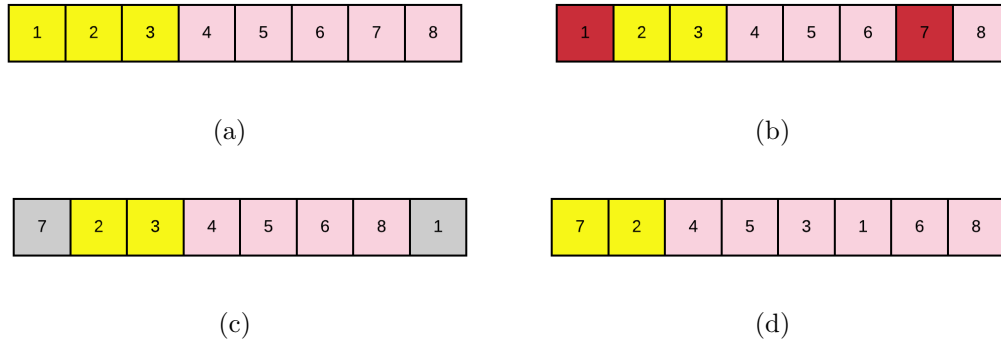


Figura 2: Exemplo da obtenção de um vizinho da estrutura de vizinhança $V^{(2)}$

- **Troca de ordem de três tarefas da solução, sendo uma do conjunto de tarefas atrasadas e duas do conjunto de tarefas adiantadas**

Neste movimento, três tarefas trocam suas posições, sendo que duas delas devem pertencer à S_E e uma delas deve pertencer à S_T . As tarefas adiantadas são inseridas no final de S_T e a tarefa atrasada é inserida no início de S_E .

A Figura 3 ilustra um exemplo do funcionamento deste movimento. Nesta figura, um vizinho da solução incumbente, representada na Figura 3(a), é obtido. Inicialmente, a solução incumbente é dividida nos conjuntos S_T e S_E , sendo que S_E está representado em amarelo e S_T está representado em rosa na Figura 3(a). Em seguida, três tarefas são selecionadas, estas estão representadas na Figura 3(b) em vermelho. A tarefa atrasada é realocada no início de S_E e as tarefas adiantadas são realocadas no final de S_T , conforme pode ser visto na Figura 3(c) em cinza. Por fim, após a realocação, a sequência obtida é reordenada em forma de V e o vizinho, representado na Figura 3(d), é gerado.

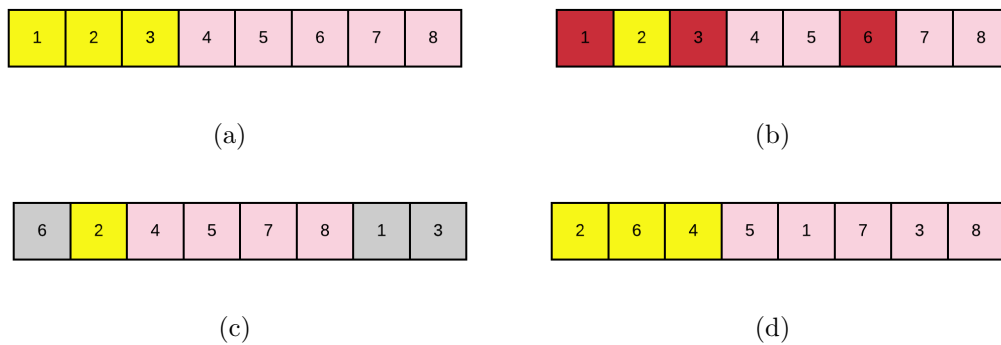


Figura 3: Exemplo da obtenção de um vizinho da estrutura de vizinhança $V^{(3)}$

A cardinalidade da estrutura de vizinhança $V^{(3)}$ depende da quantidade de tarefas

que estão alocadas em S_T e S_E e, no pior caso, quando os dois conjuntos têm a mesma quantidade de tarefas, é dada por $O((N^3 - 2N^2)/16)$.

- **Troca de ordem de três tarefas da solução, sendo duas do conjunto de tarefas atrasadas e uma do conjunto de tarefas adiantadas**

Neste movimento, três tarefas trocam suas posições, sendo que uma delas deve pertencer à S_E e as outras duas devem pertencer à S_T . A tarefa adiantada é inserida no final de S_T e as tarefas atrasadas são inseridas no início de S_E .

A Figura 4 ilustra um exemplo do funcionamento deste movimento. Nesta figura, um vizinho da solução incumbente, representada na Figura 4(a), é obtido. Inicialmente, a solução incumbente é dividida nos conjuntos S_T e S_E , sendo que S_E está representado em amarelo e S_T está representado em rosa na Figura 4(a). Em seguida, três tarefas são selecionadas, estas estão representadas na Figura 4(b) em vermelho. As tarefas atrasadas são realocadas no início do conjunto das tarefas adiantadas e a tarefa adiantada é realocada no final do conjunto das tarefas atrasadas, conforme pode ser visto na Figura 4(c) em cinza. Por fim, após a realocação, a sequência obtida é reordenada em forma de V e o vizinho, representado na Figura 4(d), é gerado.

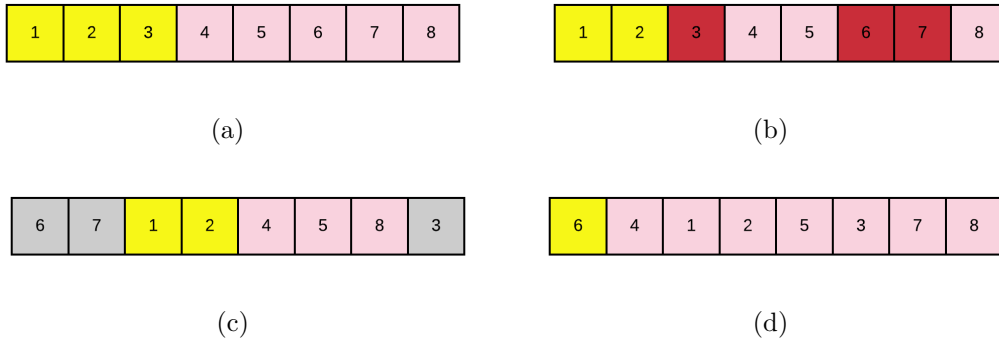


Figura 4: Exemplo da obtenção de um vizinho da estrutura de vizinhança $V^{(4)}$

A cardinalidade da estrutura de vizinhança $V^{(4)}$ depende da quantidade de tarefas que estão alocadas em S_T e S_E e, no pior caso, quando os dois conjuntos têm a mesma quantidade de tarefas, é dada por $O((N^3 - 2N^2)/16)$.

3.2.2 VNS

A metaheurística VNS foi proposta por Mladenović e Hansen (1997) e consiste em um método de busca local que explora o espaço de busca através de mudanças

sistemáticas realizadas por diferentes estruturas de vizinhança. O pseudocódigo do VNS é apresentado no Algoritmo 2. Os parâmetros de entrada do VNS são a solução inicial (s_0), a quantidade de estruturas de vizinhança utilizadas (k_{max}) e as estruturas de vizinhança $V^{(k)} = \{V^{(1)}, V^{(2)}, V^{(3)}, V^{(4)}\}$. As estruturas de vizinhança são pré-ordenadas conforme foi visto na seção anterior e a variável k indica qual delas está sendo empregada.

Algoritmo 2: VNS

Entrada: Solução inicial s_0 , k_{max} , $V^{(k)}$

Saída: Melhor solução s

```

1  início
2       $s \leftarrow s_0$ ;
3      repita
4           $k \leftarrow 1$ ;
5          repita
6              Gere um vizinho  $s' \in V^{(k)}(s)$ ;
7               $s'' \leftarrow$  Busca local ( $s'$ ) (Algoritmo 3);
8              se  $f(s'') < f(s)$  então
9                   $s \leftarrow s''$ ;
10                  $k \leftarrow 1$ ;
11             senão
12                  $k \leftarrow k + 1$ ;
13         fim
14     até  $k > k_{max}$ ;
15 até Critério de parada seja satisfeito;
16 Retorne  $s$ ;
17 fim

```

O VNS parte de uma solução inicial e da primeira estrutura de vizinhança (linhas 2 e 4). Um vizinho pertencente à estrutura de vizinhança k é escolhido aleatoriamente (linha 6) e, em seguida, uma busca local é aplicada neste vizinho (linha 7). Se a solução obtida através da busca local for melhor que a solução incumbente, a solução incumbente é atualizada e a busca recomeça com $k = 1$ (linhas 8 à 10). Quando o processo de busca local não produz uma solução melhor, o algoritmo altera a estrutura de vizinhança por outra que ainda não foi utilizada (linha 12). O processo se repete, considerando a nova estrutura de vizinhança e a solução incumbente, até que o critério

de parada seja satisfeito. A saída do algoritmo consiste na melhor solução s encontrada (linha 16).

3.2.3 BUSCA LOCAL

A busca local no VNS pode ser implementada utilizando diferentes algoritmos. Neste trabalho, a busca local foi realizada utilizando o algoritmo Descida em Vizinhança Variável - *Variable Neighborhood Descent* (VND) proposto por Mladenović e Hansen (1997). O espaço de soluções no VND é explorado por meio de trocas de estruturas de vizinhança, onde são aceitas somente as soluções que apresentam melhoria em relação à solução atual. O método sempre retorna para a primeira vizinhança quando uma solução melhor é encontrada.

O pseudocódigo do VND pode ser visto no Algoritmo 3. Os parâmetros de entrada do VND são a solução incumbente s (vizinho obtido no VNS), a quantidade de estruturas de vizinhança utilizadas (k_{max}) e as estruturas de vizinhança $V^{(k)} = \{V^{(1)}, V^{(2)}, V^{(3)}, V^{(4)}\}$. As estruturas de vizinhança são pré-ordenadas e a variável k indica qual delas está sendo empregada.

Algoritmo 3: Busca Local (VND)

Entrada: Solução s , k_{max} , $V^{(k)}$

Saída: Melhor solução s

```

1  início
2       $k \leftarrow 1$ ;
3      repita
4          Encontre o melhor vizinho  $s' \in V^{(k)}(s)$  - First Improvement;
5          se  $f(s') < f(s)$  então
6               $s \leftarrow s'$ ;
7               $k \leftarrow 1$ ;
8          senão
9               $k \leftarrow k + 1$ ;
10         fim
11     até  $k > k_{max}$ ;
12     Retorne  $s$ 
13 fim
```

O VND parte de uma solução s e da primeira estrutura de vizinhança (linha 2). A

exploração da vizinhança é realizada com a finalidade de encontrar a melhor solução vizinha (linha 4). Se a solução obtida for melhor que a solução s , a solução s é atualizada e a busca recomeça com $k = 1$ (linhas 5 à 7). Quando o processo não produz uma solução melhor, o algoritmo altera a estrutura de vizinhança por outra que ainda não foi utilizada (linha 9). O processo se repete até que todas as estruturas de vizinhança sejam exploradas (linha 11). A saída do algoritmo consiste na melhor solução s encontrada (linha 12).

A busca pelo melhor vizinho s' (linha 4) é realizada utilizando heurísticas de refinamento, sendo que os métodos mais utilizados são o *first improvement*, o *best improvement* e o método de subida/descida aleatória. O método utilizado neste trabalho foi o *first improvement*, ou método de primeira melhora, onde a busca é interrompida quando a primeira solução de melhora é encontrada. Apenas no pior caso (ótimo local), toda a vizinhança é explorada.

3.3 HEURÍSTICAS MATEMÁTICAS

As heurísticas matemáticas são algoritmos que combinam heurísticas e metaheurísticas com métodos de programação matemática. Os métodos de programação matemática e as metaheurísticas são estratégias de otimização complementares em termos de tempo computacional e qualidade de solução. De acordo com Talbi (2013), a integração entre os métodos exatos e não exatos pode ocorrer em diferentes níveis, sendo a integração maior ou menor dependendo da configuração utilizada. As principais configurações utilizadas na integração dos dois métodos são:

- O algoritmo exato ou a metaheurística resolvendo um problema de natureza diferente do problema de otimização considerado. Um exemplo dessa configuração seria a utilização da metaheurística para geração dos limites superiores do algoritmo exato.
- A metaheurística incorporando um algoritmo exato ao longo da busca pela solução, ou vice versa.
- As duas famílias de algoritmos sendo usadas de forma independente, em uma etapa de pré-processamento ou pós-processamento. Nesta configuração, informações são trocadas entre as duas famílias de algoritmos de forma sequencial.
- As duas famílias de algoritmos resolvendo o problema de forma independente e

trocando informações ao longo de processo de busca pela solução. Nesta configuração, informações são trocadas entre as duas famílias de algoritmos de forma paralela.

O tempo computacional necessário para solucionar o problema SMCDD através de métodos exatos cresce exponencialmente à medida que o número de tarefas aumenta. Entretanto, existem estratégias para reduzir o tempo computacional do algoritmo exato quando o número de tarefas é grande. Nestas estratégias, utiliza-se a solução incumbente (melhor solução encontrada até o momento) da heurística ou da metaheurística para reduzir o número de variáveis que serão otimizadas pelo software de programação inteira.

As estratégias de fixação são classificadas em dois tipos: fixação forte e fixação fraca. Nas estratégias com fixação forte, um conjunto de variáveis da solução incumbente são fixadas no modelo matemático e o outro conjunto é otimizado. Nas estratégias de fixação fraca, as variáveis que manterão seu valor não são definidas previamente e isso proporciona ao algoritmo uma flexibilidade para procurar as melhores variáveis para serem alteradas (CAMARGO; TOLEDO & ALMADA-LOBO, 2014).

Neste trabalho foi realizado um estudo de heurísticas matemáticas para o problema SMCDD e os dois tipos de estratégias de fixação foram explorados. Os algoritmos com estratégia de fixação forte utilizaram abordagens inspiradas em *Fix-and-Optimize* (FixOpt) e RINS, que estão descritos nas Seções 3.3.1 e 3.3.2. O algoritmo com estratégia de fixação fraca utilizado foi o *Local Branching* descrito na Seção 3.3.3.

Um estudo dos modelos matemáticos descritos na Seção 2.1, MIP 1 e MIP 2, foi realizado e os resultados serão apresentados na Seção 4.3. Entretanto, será indicado no pseudocódigo de cada uma das vizinhanças exatas quais modelos podem ser utilizados na sua implementação.

3.3.1 FIX-AND-OPTIMIZE

Na abordagem *Fix-and-Optimize*, a metaheurística controla o processo de busca local e, enquanto parte das variáveis da solução incumbente são fixadas no modelo MILP, as variáveis restantes são deixadas livres para serem modificadas livremente pelo solver MILP (FONSECA; SANTOS & CARRANO, 2016). Uma das dificuldades ao utilizar esta abordagem é a escolha das variáveis que serão fixadas em cada etapa, pois escolhas erradas podem influenciar negativamente o comportamento do algoritmo.

Algumas vizinhanças exatas foram implementadas neste trabalho utilizando a ideia do algoritmo FixOpt; estas estão descritas nas próximas seções.

3.3.1.1 VIZINHANÇA FIXOPT 1

O objetivo da vizinhança FixOpt 1 é dividir a solução incumbente em blocos e, então, fazer a otimização. Um exemplo do funcionamento desta vizinhança pode ser visto na Figura 6. As tarefas em vermelho formam os blocos que serão otimizados e as linhas 2 à 4 da figura correspondem às configurações das iterações do algoritmo. Um tamanho de janela é definido de acordo com a quantidade de tarefas da sequência que serão deslocadas para se obter a sequência da próxima iteração.

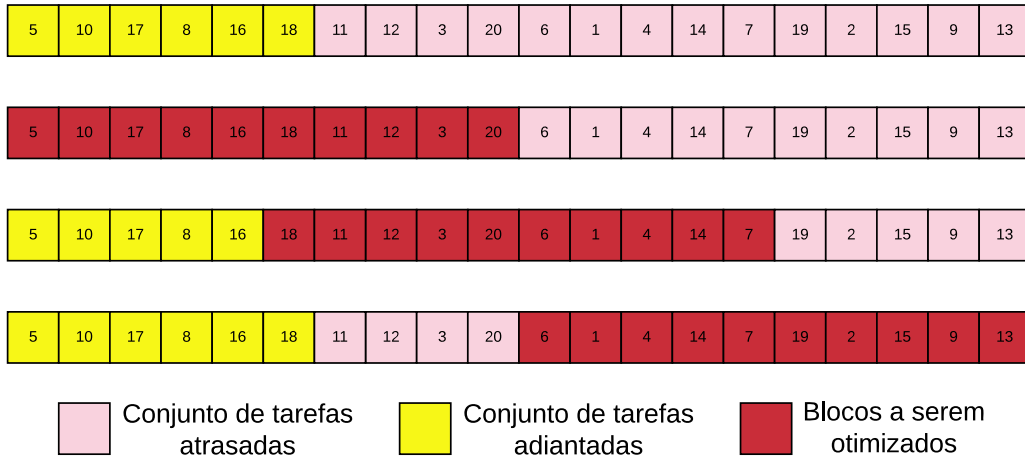


Figura 5: Exemplo da vizinhança FixOpt 1

Nas iterações cujos blocos são formados exclusivamente por tarefas adiadas ou exclusivamente por tarefas atrasadas, a otimização é realizada através da ordenação das tarefas de acordo com a Propriedade 2¹. Somente nos blocos onde existem tarefas atrasadas e adiadas, utiliza-se o solver MILP. O pseudocódigo desta vizinhança pode ser visto no Algoritmo 4. Os parâmetros de entrada do algoritmo são a solução incumbente (s), a instância do problema SMCDD, a data de entrega comum (d), o tamanho da janela (tam_janela), a quantidade de tarefas do bloco (N_bloco), o número de tarefas da instância (N) e uma variável r , que indica qual modelo MILP está sendo utilizado (MIP 1 ou MIP 2).

¹Propriedade 2: na sequência ótima, as tarefas são ordenadas em forma de V, onde as tarefas que estão adiadas são sequenciadas em ordem decrescente de p_i/α_i e as tarefas que estão atrasadas são sequenciadas em ordem crescente de p_i/β_i

Algoritmo 4: Vizinhança FixOpt 1

Entrada: Solução incumbente s , Instância do problema SMCDD, Data de entrega comum d , Tamanho da janela (tam_janela), Número de tarefas da instância (N), Quantidade de tarefas do bloco (N_bloco), r

Saída: Melhor solução s

```

1  início
2      janela  $\leftarrow 0$ ;
3      repita
4           $jobs \leftarrow$  Selecione as tarefas de  $s$  de acordo com  $N\_bloco$ s e a janela atual;
5           $t_0 \leftarrow$  Instante de início da primeira tarefa do bloco (de acordo com  $s$ );
6           $t_f \leftarrow$  Instante do fim da última tarefa do bloco (de acordo com  $s$ );
7          se  $t_f < d$  então
8               $s^* \leftarrow$  Ordene as tarefas de  $jobs$  em ordem decrescente de  $p_i/\alpha_i$ ;
9          senão se  $t_0 > d$  então
10              $s^* \leftarrow$  Ordene as tarefas de  $jobs$  em ordem crescente de  $p_i/\beta_i$ ;
11         senão
12             Carregue as informações das tarefas em  $jobs$  e construa  $\mathcal{M}^r$ ;
13             Fixe  $t_0$  e  $t_f$  em  $\mathcal{M}^r$ ;
14              $s^* \leftarrow$  Resolva  $\mathcal{M}^r$ ;
15         fim
16         Atualize a solução  $s$  de acordo com  $s^*$ ;
17         janela  $\leftarrow$  janela +  $\text{tam\_janela}$ ;
18     até  $janela > N - 10$ ;
19     Retorne  $s$ 
20 fim
  
```

A vizinhança FixOpt1 parte de uma solução inicial e de um valor de janela igual à zero (linha 2). Em seguida, tarefas da solução s são selecionadas de acordo com a janela atual (tam_janela) e com a quantidade de tarefas de cada bloco (N_bloco) formando o bloco $jobs$ e as informações referentes as tarefas do bloco $jobs$ são carregadas para a construção do modelo matemático \mathcal{M} (linhas 4 e 5). O instante de início da primeira tarefa do bloco e o instante de fim da última tarefa do bloco são determinados considerando a posição destes blocos na solução s (linhas 6 e 7). Se todas as tarefas do bloco estiverem adiantadas na solução s ($t_f < d$), as tarefas são ordenadas em ordem decrescente de p_i/α_i (linha 9). Se todas as tarefas do bloco estiverem atrasadas na solução s ($t_0 > d$), as tarefas são ordenadas em ordem crescente de p_i/β_i (linha 11). Caso

existam tarefas adiantadas e atrasadas no bloco, t_f e t_0 são fixados em \mathcal{M} e as tarefas são otimizadas utilizando o solver MILP (linhas 13 e 14). Em seguida, a solução s é atualizada de acordo com s^* e a nova janela é obtida (linhas 16 e 17). O processo se repete até que todos os blocos sejam otimizizados (linha 18) e a solução s é retornada (linha 19).

3.3.1.2 VIZINHANÇA FIXOPT 2

Na vizinhança FixOpt 2, as tarefas da solução incumbente são divididas nos conjuntos S_T e S_E e, com a finalidade de otimizar tarefas próximas da data de entrega comum, tarefas consecutivas dos dois conjuntos são otimizadas pelo solver MILP em cada iteração do algoritmo. Um exemplo do funcionamento desta vizinhança pode ser visto na Figura 6.



Figura 6: Exemplo da vizinhança FixOpt 2

Em todas as iterações, a otimização é realizada com 10 tarefas consecutivas (este valor foi determinado através do estudo dos modelos matemáticos descrito na Seção 4.3). A escolha das tarefas que serão otimizadas pelo solver MILP é realizada da seguinte forma: Inicialmente são selecionadas 9 tarefas de S_E e 1 tarefa de S_T ; Nas iterações seguintes, retira-se uma tarefa de S_E e acrescenta uma tarefa de S_T ; O processo é repetido até que, na última iteração, a otimização é realizada com 1 tarefa de S_E e 9 tarefas de S_T . O pseudocódigo da vizinhança FixOpt 2 pode ser visto no Algoritmo 5. Os parâmetros de entrada do algoritmo são a solução incumbente s , a instância do problema SMCDD e uma variável r , que indica qual modelo MILP está sendo utilizado (MIP 1 ou MIP 2).

Inicialmente, todas as possibilidades de escolha das tarefas de S_E e S_T são deter-

minadas (linha 2). A vizinhança FixOp2 parte de uma solução inicial e da iteração 1 (linha 3) e, em seguida, 10 tarefas consecutivas da solução s são selecionadas de acordo com $\mathcal{S}(i)$ formando o bloco $jobs$ (linha 5). As informações referentes as tarefas do bloco $jobs$ são carregadas para a construção do modelo matemático \mathcal{M} (linha 6). O instante de início da primeira tarefa do bloco e o instante de fim da última tarefa do bloco são determinados considerando a posição destes blocos na solução s (linhas 7 e 8). Os valores de t_f e t_0 são fixados em \mathcal{M} e as tarefas são otimizadas utilizando o solver MILP (linhas 9 e 10). Em seguida, a solução s é atualizada de acordo com s^* e ordenada em forma de V conforme a Propriedade 2 (linhas 11 e 12). O processo se repete até que todas as possibilidades de \mathcal{S} sejam otimizadas (linha 14) e a solução s é retornada (linha 15).

Algoritmo 5: Vizinhança FixOpt 2

Entrada: Solução incumbente s , Instância do problema SMCDD, r

Saída: Melhor solução s

```

1  início
2  |  $\mathcal{S} \leftarrow$  Determine todas as possibilidade de escolha das tarefas de  $S_E$  e  $S_T$ ;
3  |  $i \leftarrow 1$ ;
4  | repita
5  |    $jobs \leftarrow$  Selecione 10 tarefas consecutivas de  $s$  de acordo com  $\mathcal{S}(i)$ ;
6  |   Carregue as informações das tarefas em  $jobs$  e construa  $\mathcal{M}^r$ ;
7  |    $t_0 \leftarrow$  Instante de início da primeira tarefa do bloco (de acordo com  $s$ );
8  |    $t_f \leftarrow$  Instante do fim da última tarefa do bloco (de acordo com  $s$ );
9  |   Fixe  $t_0$  e  $t_f$  em  $\mathcal{M}^r$ ;
10 |    $s^* \leftarrow$  Resolva  $\mathcal{M}^r$ ;
11 |   Atualize a solução  $s$  de acordo com  $s^*$ ;
12 |   Ordene  $s$  em forma de V;
13 |    $i \leftarrow i + 1$ ;
14 | até  $i > |\mathcal{S}|$ ;
15 | Retorne  $s$ 
16 fim

```

3.3.1.3 VIZINHANÇA FIXOPT 3

A ideia da vizinhança FixOpt 3 é determinar a melhor posição para cada uma das tarefas de determinada solução s . Para tanto, o impacto no valor de função objetivo de

cada uma das tarefas é calculado. Em seguida, as tarefas são ordenadas de acordo com o valor obtido em ordem decrescente, de forma a otimizar primeiramente as tarefas com maior impacto na função objetivo. O pseudocódigo da vizinhança FixOpt 3 pode ser visto no Algoritmo 6. Os parâmetros de entrada do algoritmo são a solução incumbente s , a instância do problema SMCDD, o número de tarefas da instância (N), o modelo matemático do problema (\mathcal{M}) e uma variável r , que indica qual modelo MILP está sendo utilizado (MIP 1 ou MIP 2).

Algoritmo 6: Vizinhança FixOpt 3

Entrada: Solução incumbente s , Instância do problema SMCDD, Número de tarefas da instância (N), Modelo matemático \mathcal{M} , r

Saída: Melhor solução s

```

1 início
2   Carregue o modelo matemático  $\mathcal{M}^r$ ;
3    $x \leftarrow$  Carregue as variáveis binárias  $x_{ij}$  da solução  $s$ ;
4    $jobs \leftarrow$  Ordene as tarefas da solução  $s$  em ordem decrescente de impacto no
      valor de função objetivo;
5    $i \leftarrow 1$ ;
6   repita
7     Selecione a tarefa  $jobs(i)$ ;
8     Fixe em  $\mathcal{M}^r$  todas as variáveis binárias  $(x_{ij})$  de  $x$ , exceto as referentes à
      tarefa  $jobs(i)$  ;
9      $(s, x) \leftarrow$  Resolva  $\mathcal{M}^r$ ;
10     $i \leftarrow i + 1$ ;
11  até  $i > N$ ;
12  Retorne  $s$ 
13 fim

```

Inicialmente, o modelo matemático \mathcal{M} e as variáveis binárias x_{ij} da solução s são carregados (linhas 2 e 3). O impacto no valor de função objetivo de cada uma das tarefas é calculado e as tarefas são ordenadas em ordem decrescente (linha 4). Em seguida, a tarefa com maior impacto no valor de função objetivo é selecionada (linha 7) e todas as variáveis binárias x_{ij} referentes a esta tarefa são deixadas livres. As variáveis binárias x_{ij} das outras tarefas são fixadas em \mathcal{M} (linha 8). Em seguida, o modelo \mathcal{M} é resolvido utilizando o solver MILP (linha 9). O processo se repete, selecionando a próxima tarefa de $jobs$, até que todas as tarefas tenham tido suas variáveis binárias x_{ij}

livres em algum momento (linha 11). A saída do algoritmo é a solução s obtida.

3.3.2 RELAXATION INDUCED NEIGHBORHOOD SEARCH (RINS)

O algoritmo RINS foi proposto por Danna, Rothberg e Pape (2005) e baseia-se na ideia de que as variáveis da solução incumbente e as variáveis encontradas ao resolver o problema relaxado (resolução do problema sem considerar as restrições de integralidade) podem ser combinadas para alcançar uma solução que atinja a integralidade e um bom valor de função objetivo. Os passos básicos do algoritmo RINS são:

1. Fixar no modelo matemático as variáveis que possuem os mesmos valores na solução incumbente e na solução relaxada;
2. Definir um ponto de corte baseado na solução incumbente;
3. Resolver o sub-problema MILP com as variáveis restantes.

Uma vizinhança exata baseada no algoritmo RINS foi implementada neste trabalho. Os detalhes dessa vizinhança estão descritos a seguir.

3.3.2.1 VIZINHANÇA EXATA BASEADA NO ALGORITMO RINS

A vizinhança descrita a seguir foi inspirada no algoritmo RINS e apresenta duas diferenças básicas com relação ao algoritmo original. Nesta abordagem, o problema SMCDD é resolvido pelo solver MILP na sua forma completa (considerando as restrições de integralidade) com o tempo de execução limitado. Para que esta vizinhança apresente um bom desempenho, o tempo de execução do solver MILP em cada iteração deve ser pequeno o suficiente para que a vizinhança tenha um tempo de execução total relativamente pequeno e grande o suficiente para que a otimização com o solver MILP retorne uma solução razoável.

A segunda diferença consiste no fato de que somente parte das variáveis binárias, x_{ij} , que possuem os mesmos valores na solução de referência e na solução obtida pelo solver MILP são fixadas. O pseudocódigo desta vizinhança pode ser visto no Algoritmo 7. Os parâmetros de entrada do algoritmo são a solução de referência (s^*), o modelo matemático do problema (\mathcal{M}), a porcentagem das variáveis binárias, x_{ij} , que serão fixadas (p) e uma variável r , que indica qual modelo MILP está sendo utilizado (MIP 1 ou MIP 2). Os valores do tempo de execução do solver MILP em cada iteração e da porcentagem variáveis que serão fixadas estão descritos na Seção 4.4.

Inicialmente, o modelo matemático e as variáveis da solução de referência são carregados (linhas 2 e 3). Então, o modelo matemático \mathcal{M} é resolvido utilizando o solver MILP com o tempo de execução limitado (linha 4). Em seguida, parte das variáveis binárias, x_{ij} , são escolhidas aleatoriamente e fixadas no modelo matemático \mathcal{M} (linhas 6 e 7). O modelo matemático é resolvido novamente com o tempo de execução limitado (linha 8) e o processo se repete até que o solver MILP atinja a otimalidade (linha 9). A saída do algoritmo consiste na solução s obtida ordenada em forma de V (linhas 10 e 11).

Algoritmo 7: Heurística matemática baseada no algoritmo RINS

Entrada: Solução de referência s^* , Modelo matemático \mathcal{M} , Porcentagem das variáveis que serão fixadas p, r

Saída: Solução s

```

1  início
2       $\mathcal{M}^r \leftarrow$  Carregue o modelo matemático;
3       $x^* \leftarrow$  Carregue as variáveis binárias  $x_{ij}$  da solução de referência  $s^*$ ;
4       $(s, x) \leftarrow$  Resolva  $\mathcal{M}^r$  com tempo de execução limitado;
5      repita
6           $\mathcal{X} \leftarrow$  Escolha aleatoriamente  $p\%$  variáveis binárias  $x_{ij}$  que são iguais
              em  $x$  e  $x^*$ ;
7          Fixe  $\mathcal{X}$  em  $\mathcal{M}^r$ ;
8           $(s, x) \leftarrow$  Resolva  $\mathcal{M}^r$  com tempo de execução limitado;
9      até  $status = \text{"Optimal"}$ ;
10     Ordene  $s$  em forma de V;
11     Retorne  $s$ 
12 fim

```

3.3.3 LOCAL BRANCHING

O algoritmo *Local Branching* foi proposto por Fischetti e Lodi (2003) e utiliza a estratégia de fixação de variáveis fraca. Neste algoritmo, uma restrição é adicionada ao modelo matemático indicando a porcentagem das variáveis da solução incumbente que devem manter seu valor atual. Entretanto, a restrição não indica quais variáveis manterão o seu valor. A quantidade de variáveis que devem ser modificadas deve ser pequena o suficiente para resolver o problema em um tempo computacional curto e grande o suficiente para conter uma solução melhor que a incumbente.

Suponha que seja dada uma solução \bar{s} binária de um problema MILP com n variáveis, em que pelo menos 90% das variáveis devem manter seu valor atual. A restrição que deverá ser incluída ao modelo matemático é apresentada na Equação 3.2.

$$\sum_{j=1}^n \bar{s}_j \times s_j \geq 0,9 \left[\sum_{j=1}^n \bar{s}_j \right] \quad (3.2)$$

Dada uma solução de referência \bar{s} de um problema P, o suporte binário desta solução é dado por $\bar{\mathcal{S}} := \{j \in \mathcal{B} : \bar{s}_j = 1\}$. Sendo que, o suporte binário $\bar{\mathcal{S}}$ contém os índices de todas as variáveis binárias que possuem o valor 1.

Para um dado parâmetro inteiro k , definimos o k -OPT da vizinhança $\mathcal{N}(\bar{s}, k)$ de \bar{s} como o conjunto de soluções factíveis do problema que satisfazem a restrição de *Local Branching* apresentada na Equação 3.3.

$$\Delta(s, \bar{s}) := \sum_{j \in \bar{\mathcal{S}}} (1 - s_j) + \sum_{j \in \mathcal{B} \setminus \bar{\mathcal{S}}} s_j \leq k \quad (3.3)$$

Para problemas onde a cardinalidade do suporte binário $\bar{\mathcal{S}}$ de qualquer solução factível é constante, a restrição pode ser reescrita na sua forma equivalente assimétrica, conforme pode ser visto na Equação 3.4.

$$\Delta(s, \bar{s}) := \sum_{j \in \bar{\mathcal{S}}} (1 - s_j) \leq k' (= k/2) \quad (3.4)$$

Fischetti e Lodi (2003) também propuseram uma heurística utilizando o *Local Branching*. O Algoritmo 8 apresenta esta heurística. Os parâmetros de entrada do algoritmo são a solução inicial (s), o modelo matemático do problema (\mathcal{M}), o tamanho da vizinhança k , o tempo limite por iteração (*TimeLimitIt*) e uma variável r , que indica qual modelo MILP está sendo utilizado (MIP 1 ou MIP 2).

Inicialmente, a restrição de *Local Branching* é adicionada ao modelo matemático (linha 2). Em seguida, o modelo matemático \mathcal{M} é resolvido utilizando o solver MILP com o tempo de execução limitado (linha 4). Após a otimização, o solver pode apresentar dois status: *Optimal* (linha 5) ou *TimeLimit* (linha 15).

Se o status da otimização for “*Optimal*” e a solução obtida for melhor que a melhor encontrada até o momento (linha 6), a melhor solução encontrada é atualizada (linhas 7 e 8), caso contrário, o valor do parâmetro k é aumentado (linha 10). Em seguida, a

última restrição de *Local Branching* e \bar{s} são atualizadas (linhas 12 e 13) e uma nova restrição de *Local Branching* é adicionada ao modelo matemático (linha 14).

Caso o status da otimização seja “*TimeLimit*” e a solução obtida for melhor que a melhor encontrada até o momento (linha 16), a melhor solução encontrada é atualizada (linhas 17 e 18) e a última restrição adicionada é removida (linha 19). Se a solução obtida pelo solver MILP não for melhor, o valor do parâmetro k é diminuído (linha 21). Em seguida, \bar{s} é atualizada (linha 23) e uma nova restrição de *Local Branching* é adicionada ao modelo matemático (linha 24). O processo se repete até que o tempo limite da otimização seja atingido (linha 26) e o valor de s^* é retornado (linha 27).

Algoritmo 8: Local Branching

Entrada: Solução inicial s , Modelo matemático \mathcal{M} , Tamanho da vizinhança k ,
Tempo limite por iteração ($TimeLimitIt$), r

Saída: Melhor solução s^*

```

1  início
2  |  $\mathcal{M}^r \leftarrow$  Carregue o modelo matemático;
3  | Adicione a restrição de Local Branching  $\Delta(s, \bar{s}) \leq k$  em  $\mathcal{M}^r$ ;
4  | repita
5  |    $(s, \text{status}) \leftarrow$  Resolva  $\mathcal{M}^r$  com o solver limitado por  $TimeLimitIt$ ;
6  |   se  $\text{status} = \text{"Optimal"}$  então
7  |     se  $f(s) < f(s^*)$  então
8  |       |  $best \leftarrow f(s)$ ;
9  |       |  $s^* \leftarrow s$ ;
10 |     senão
11 |       |  $k \leftarrow k + \frac{k}{2}$ ;
12 |     fim
13 |     Reverta a última restrição de local branching para  $\Delta(s, \bar{s}) \leq k + 1$ ;
14 |      $\bar{s} \leftarrow s$ ;
15 |     Adicione a restrição de local branching  $\Delta(s, \bar{s})$  em  $\mathcal{M}^r$ ;
16 |   senão se  $\text{status} = \text{"TimeLimit"}$  então
17 |     se  $f(s) < f(s^*)$  então
18 |       |  $best \leftarrow f(s)$ ;
19 |       |  $s^* \leftarrow s$ ;
20 |       | Remova a última restrição de local branching de  $\mathcal{M}^r$ ;
21 |     senão
22 |       |  $k \leftarrow k - \frac{k}{2}$ ;
23 |     fim
24 |      $\bar{s} \leftarrow s$ ;
25 |     Adicione a restrição de local branching  $\Delta(s, \bar{s}) \leq k$  em  $\mathcal{M}^r$ ;
26 |   fim
27 | até  $\text{elapsedTime} > TimeLimit$ ;
28 | Retorne  $s^*$ 
29 fim

```

3.4 ABORDAGEM PROPOSTA

As vizinhanças apresentadas nos Algoritmos 4 à 8 foram implementadas com algoritmos exatos, sendo que os dois modelos matemáticos estudados, MIP 1 e MIP 2, foram testados em cada uma delas. Para as vizinhanças FixOpt1, FixOp2, RINS e *Local Branching*, o modelo MIP 2 foi mais eficiente. Já a vizinhança FixOpt3 só pôde ser implementada com o modelo MIP 1. Isto porque, no modelo MIP 2, existem duas restrições que garantem que uma tarefa só pode ser atribuída à uma posição e cada posição só pode receber uma tarefa. Desta forma, ao fixar todas as variáveis binárias de uma solução no modelo matemático exceto a de uma única tarefa, o solver MILP retorna a solução de referência.

A Figura 7 mostra a abordagem proposta neste trabalho. A vizinhança exata baseada em RINS foi utilizada para refinar o resultado obtido pela heurística implementada para a construção da solução inicial descrita na Seção 3.1. A justificativa para o seu uso consiste no fato que a vizinhança exata baseada em RINS apresentou uma melhoria no valor de função objetivo relevante em um tempo computacional razoável. Os valores obtidos através da vizinhança exata baseada em RINS foram utilizados como solução inicial do algoritmo VNS descrito na Seção 3.2.2. A busca local no VNS foi realizada utilizando o algoritmo VND descrito na Seção 3.2.3 e as quatro estruturas de vizinhança implementadas $V^{(k)} = \{V^{(1)}, V^{(2)}, V^{(3)}, V^{(4)}\}$ foram utilizadas no VNS e no VND.

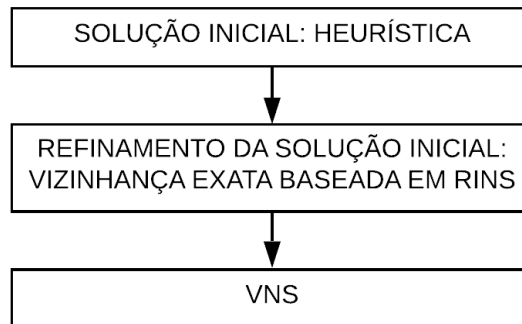


Figura 7: Abordagem proposta

Com relação ao desempenho dos outros algoritmos implementados, as vizinhanças FixOpt1, FixOpt2 são eficientes do ponto de vista de tempo computacional, mesmo em instâncias com um grande número de tarefas. Entretanto, a partir de testes preliminares, observou-se que estas vizinhanças não forneceram uma melhoria significativa no valor de função objetivo quando acrescentadas ao longo da busca pela solução da metaheurística, ou quando utilizadas para refinar o valor obtido pela solução inicial.

Por esta razão, não foram utilizadas na abordagem proposta. Porém testes mais completos são necessários para avaliar a eficiência ou não destas vizinhanças no problema SMCDD.

As vizinhanças FixOpt 3 e o *Local Branching* são usadas para determinar a melhor posição para determinada tarefa. Porém, o tempo computacional da vizinhança FixOpt 3 depende do número de tarefas da instância e cresce muito à medida que a quantidade de tarefas aumenta, o que não é interessante para a elaboração de heurísticas matemáticas. O mesmo ocorre com o algoritmo *Local Branching*, que apresentou um tempo computacional elevado mesmo em instâncias com poucas tarefas, inviabilizando uma abordagem com este algoritmo.

4 RESULTADOS COMPUTACIONAIS

Este capítulo apresenta os resultados obtidos pelos algoritmos desenvolvidos neste trabalho. A Seção 4.1 apresenta o ambiente computacional em que os testes foram realizados. A Seção 4.2 apresenta as características das instâncias utilizadas para avaliação do desempenho do algoritmo. A Seção 4.3 apresenta a validação dos modelos matemáticos, MIP 1 e MIP 2, descritos no trabalho. A Seção 4.4 apresenta os resultados da vizinhança exata baseada em RINS. A Seção 4.5 apresenta os resultados obtidos pelo VNS. A Seção 4.6 apresenta uma comparação dos resultados obtidos no trabalho com diferentes métodos da literatura.

4.1 AMBIENTE COMPUTACIONAL

Os algoritmos foram implementados utilizando o software Matlab 2016a e o software de otimização Gurobi 8.1.0. Os testes foram feitos em um computador portátil Intel Core i5, com 4GB de memória RAM, em um ambiente Windows 10.

4.2 CARACTERIZAÇÃO DAS INSTÂNCIAS

Os testes foram realizados utilizando as instâncias *benchmark* do problema SMCDD desenvolvidas por Biskup e Feldmann (2001). O conjunto de instâncias apresenta quantidades de tarefas distintas, que variam em 10, 20, 50, 100, 200, 500 e 1000 tarefas. Cada conjunto contém 10 problemas e os valores de função objetivo são analisados para 4 valores de data de entrega comum (utilizando os valores de $h = 0,2, 0,4, 0,6$ e $0,8$ na Equação 2.1).

4.3 VALIDAÇÃO DOS MODELOS MATEMÁTICOS

A validação dos modelos matemáticos descritos na Seção 2.1 foi realizada utilizando as instâncias *benchmark* do problema SMCDD. Os testes foram realizados com as

instâncias de 10 e 20 tarefas e os valores de data de entrega comum foram calculados com $h=0,2$, $h=0,4$, $h=0,6$ e $h=0,8$.

Os resultados dos testes realizados podem ser vistos nas Tabelas 1, 2 e 3. As colunas das tabelas estão dispostas da seguinte forma: na primeira coluna, estão os nomes das instâncias; nas colunas 2 e 3 estão os valores do parâmetro representado pela tabela (tempo, função objetivo ou GAP) dos modelos MIP 1 e MIP 2, respectivamente, para os valores de data de entrega comum calculada com $h=0,2$; nas colunas 4 e 5 estão os valores obtidos quando a data de entrega comum é calculada com $h=0,4$; nas colunas 6 e 7 estão os valores obtidos quando a data de entrega comum é calculada com $h=0,6$; nas colunas 8 e 9 estão os valores obtidos quando a data de entrega comum é calculada com $h=0,8$. Em cada uma das tabelas foi destacado o modelo (MIP 1 ou MIP 2) que apresentou o melhor desempenho, de acordo com o parâmetro analisado, para cada um dos valores de h .

As instâncias com 10 tarefas convergiram em poucos segundos. O tempo necessário para a convergência variou nas execuções do software Gurobi, o que justifica a realização de 30 execuções do algoritmo para cada um dos problemas. O tempo médio que cada um dos modelos demora para realizar a otimização pode ser visto na Tabela 1. As instâncias com 20 tarefas não convergem rapidamente e, por esta razão, o software de otimização Gurobi foi limitado a 5 minutos. Em todas as instâncias, a otimização terminou quando o tempo limite foi atingido. Os valores de função objetivo obtidos pelos dois modelos podem ser observados na Tabela 2.

Além do valor de função objetivo, o valor do GAP (que consiste na diferença entre o valor de função objetivo da melhor solução encontrada até o momento e o valor de função objetivo do melhor limitante identificado) é um parâmetro importante quando se limita o tempo no teste. Logo, por meio deste parâmetro é possível saber se a otimização está próxima do fim. Os valores dos GAPs dos testes das instâncias com 20 tarefas podem ser observados na Tabela 3.

Análise dos resultados

A Tabela 1 mostra os tempos médios para se obter a solução ótima nas instâncias de 10 tarefas. É possível observar que os tempos médios são relativamente baixos. O maior valor de tempo médio obtido pelo MIP 1 foi 25,51s e o maior valor de tempo médio obtido pelo MIP 2 foi 13,11s.

Tabela 1: Tempo médio (em segundos) dos modelos MIP 1 e MIP 2 para as instâncias com 10 tarefas

Instâncias	h=0,2		h=0,4		h=0,6		h=0,8	
	MIP 1	MIP 2	MIP 1	MIP 2	MIP 1	MIP 2	MIP 1	MIP 2
sch10k1	3,50s	2,85s	2,71s	1,54s	1,99s	2,32s	1,63s	2,47s
sch10k2	1,08s	0,84s	0,68s	1,72s	0,63s	2,29s	0,70s	1,84s
sch10k3	4,78s	1,58s	3,23s	1,66s	2,53s	1,85s	2,99s	1,80s
sch10k4	8,66s	2,81s	5,09s	13,11s	1,83s	6,59s	1,94s	10,44s
sch10k5	4,36s	5,58s	2,32s	2,21s	0,81s	4,84s	1,02s	5,97s
sch10k6	11,21s	2,73s	25,51s	2,06s	6,23s	2,23s	5,24s	2,12s
sch10k7	2,49s	2,81s	3,13s	2,51s	2,05s	2,50s	1,42s	3,04s
sch10k8	3,68s	5,26s	2,69s	2,60s	1,89s	1,40s	1,00s	2,64s
sch10k9	2,96s	1,95s	2,27s	1,24s	1,34s	1,30s	1,25s	1,20s
sch10k10	2,75s	2,89s	3,24s	2,33s	1,44s	1,96s	1,17s	2,30s

Tabela 2: Valores de função objetivo obtidos pelos modelos MIP 1 e MIP 2 para as instâncias com 20 tarefas limitadas em 5 minutos no software Gurobi

Instâncias	h=0,2		h=0,4		h=0,6		h=0,8	
	MIP 1	MIP 2	MIP 1	MIP 2	MIP 1	MIP 2	MIP 1	MIP 2
sch20k1	4545	4558	3070	3186	3005	3081	3117	2998
sch20k2	8781	8524	5229	4973	3319	3241	3037	2981
sch20k3	6257	6197	3868	3838	3655	3604	3613	3608
sch20k4	9356	9347	5183	5297	3334	3330	3051	3052
sch20k5	4269	4219	2522	2589	2234	2266	2188	2201
sch20k6	6610	6597	3588	3555	3093	3045	3060	3022
sch20k7	10581	10468	6416	6208	4143	4204	3895	3892
sch20k8	3968	4096	2362	2107	1757	1698	1638	1656
sch20k9	3478	3413	2080	2081	1973	2017	1984	1996
sch20k10	5188	5023	3097	2949	2117	2110	1995	2002

Tabela 3: Valores dos GAPs obtidos pelos modelos MIP 1 e MIP 2 para as instâncias com 20 tarefas limitadas em 5 minutos no software Gurobi

Instâncias	h=0,2		h=0,4		h=0,6		h=0,8	
	MIP 1	MIP 2	MIP 1	MIP 2	MIP 1	MIP 2	MIP 1	MIP 2
sch20k1	76,65%	54,40%	59,08%	76,72%	56,82%	80,24%	56,57%	73,20%
sch20k2	89,84%	46,22%	82,93%	60,70%	72,78%	50,90%	69,99%	45,26%
sch20k3	82,60%	63,93%	73,07%	72,30%	71,87%	68,20%	69,51%	69,98%
sch20k4	88,80%	66,25%	80,84%	64,80%	70,84%	53,48%	66,27%	56,90%
sch20k5	75,65%	52,50%	64,86%	54,92%	57,70%	68,65%	55,38%	67,84%
sch20k6	83,92%	74,19%	71,58%	71,99%	63,72%	59,01%	68,87%	70,12%
sch20k7	88,23%	68,24%	83,20%	73,85%	72,14%	59,33%	71,21%	55,04%
sch20k8	82,10%	55,05%	73,09%	69,21%	65,25%	59,56%	62,58%	56,76%
sch20k9	80,33%	43,24%	68,61%	58,57%	67,67%	74,95%	68,19%	68,08%
sch20k10	81,06%	53,15%	74,98%	65,85%	61,43%	52,65%	57,92%	55,75%

O MIP 1 apresentou tempos médios melhores em 19 instâncias, sendo que em 13 instâncias o problema foi solucionado para um valor de data de entrega comum com $h=0,6$ e $h=0,8$. O MIP 2 apresentou tempos médios melhores em 21 instâncias, sendo que em 14 instâncias o problema foi solucionado para um valor de data de entrega comum com $h=0,2$ e $h=0,4$. Além disto, também é interessante destacar que dependendo do modelo matemático utilizado, o tempo de execução pode variar bastante. Um exemplo é a instância *sch10k6* para um valor de data de entrega comum com $h=0,4$, que com o MIP 1 obteve tempos médios elevados, enquanto que com o MIP 2 obteve tempos médios relativamente pequenos.

Na Tabela 2, é possível observar que o MIP 1 apresentou valores de função objetivo melhores em 15 instâncias, sendo que em 9 instâncias o problema foi resolvido para um valor de data de entrega comum com $h=0,6$ e $h=0,8$, enquanto o MIP 2 apresentou valores de função objetivo melhores em 25 instâncias, sendo que em 14 instâncias o problema foi resolvido para um valor de data de entrega comum com $h=0,2$ e $h=0,4$.

Analizando os valores dos GAPs da Tabela 3, é possível observar que, de uma forma geral, os GAPs são elevados. Isto indica que a otimização estava longe de terminar nos dois modelos. Os valores dos GAPs obtidos pelo MIP 1 foram melhores em 9 instâncias, enquanto os valores dos GAPs obtidos pelo MIP 2 foram melhores em 31 instâncias. Também é importante ressaltar que o valor do GAP e o valor de função objetivo não apresentam uma relação linear, ou seja, o modelo que apresenta o melhor GAP nem sempre é o que apresenta o melhor valor de função objetivo. Isto ocorre por causa das características do modelo, como o GAP se baseia nos valores de função objetivo da melhor solução encontrada e do melhor limitante identificado, pode ocorrer de um modelo obter um valor de função objetivo melhor, mesmo com um valor de GAP maior.

A partir da análise dos resultados, pode-se verificar que, de uma forma geral, o MIP 2 apresentou um desempenho melhor que o MIP 1 na maioria dos testes realizados. Contudo, para valores de data de entrega comum com $h=0,6$ e $h=0,8$, os resultados obtidos pelo MIP 1 são relevantes. Sendo assim, ambos modelos são interessantes para elaboração de vizinhanças com algoritmos exatos.

Também foi observado que o tempo de execução do algoritmo exato nas instâncias com 10 tarefas é relativamente baixo, viabilizando a implementação de vizinhanças exatas. Entretanto, os testes com as instâncias de 20 tarefas mostram que à medida que o número de tarefas aumenta, a complexidade do problema aumenta muito. A otimização nestas instâncias terminou quando o tempo limite de 5 minutos foi atingido

e com valores de GAP elevados. Isto mostra que para a construção de vizinhanças exatas com um tempo computacional baixo, é necessário que o algoritmo exato realize a otimização com poucas tarefas. Problemas com 20 tarefas apresentariam um tempo computacional impeditivo.

4.4 VIZINHANCA EXATA BASEADA EM RINS

Nesta seção são apresentados os resultados obtidos pela vizinhança exata baseada em RINS descrita na Seção 3.3.2.1. Os testes foram realizados com as instâncias de 20, 50, 100, 200 e 500 tarefas e os valores de data de entrega comum foram calculados com $h = 0,2$, $h = 0,4$, $h = 0,6$ e $h=0,8$. Para cada problema, foram realizadas 30 execuções. O critério de parada foi o status "*Optimal*" no software de otimização Gurobi.

Os testes não foram realizados com as instâncias de 10 e 1000 tarefas, isto porque as instâncias de 10 tarefas convergiram em poucos segundos, não sendo necessário utilizar a abordagem baseada em RINS. Os resultados das instâncias de 10 tarefas foram apresentados na Seção 4.3. As instâncias com 1000 tarefas não convergiram com os parâmetros utilizados. Sendo assim, para estas instâncias, foi utilizado o valor obtido pela heurística da Seção 3.1 como solução inicial do VNS.

Os parâmetros que necessitam de ajustes na vizinhança exata baseada em RINS são a porcentagem das variáveis a serem fixadas no modelo matemático em cada iteração do algoritmo e o tempo de execução do software de otimização. A porcentagem das variáveis foi determinada empiricamente em 40%, sendo que somente as variáveis binárias cujo valor é 0 são fixadas. O software de otimização foi limitado em 60 segundos.

Os valores de função objetivo médios obtidos pela vizinhança exata baseada em RINS podem ser observados na Tabela 4. Na primeira coluna da tabela são descritas as instâncias. Nas quatro colunas seguintes são apresentados os valores médios de função objetivo para $h=0,2$, $h=0,4$, $h=0,6$ e $h=0,8$, respectivamente.

As Tabelas 5 e 6 apresentam os limitantes superiores obtidos pela vizinhança exata baseada em RINS. Na primeira coluna da tabela são descritas as instâncias e três colunas da tabela são utilizadas para apresentar os resultados, sendo que a coluna F_{SolIn} contém os valores obtidos pela heurística utilizada para a construção da solução inicial, a coluna $F_{VizRINS}$ apresenta os melhores valores de função objetivo obtidos através da vizinhança exata baseada em RINS e a coluna *Diferença* apresenta a melhoria da vizinhança exata baseada em RINS com relação à heurística. A Tabela 5 apresenta os

resultados obtidos para datas de entregas comum calculadas com $h=0,2$ e $h=0,4$ e a Tabela 6 apresenta os resultados quando as datas de entrega comum foram calculadas com $h=0,6$ e $h=0,8$. Os tempos médios de execução do algoritmo podem ser vistos na Tabela 7.

Análise dos resultados

A partir das Tabelas 5 e 6, é possível observar que a abordagem com a vizinhança exata baseada em RINS melhorou os valores de função objetivo obtidos pela heurística utilizada para a construção da solução inicial em 75% das instâncias testadas. Conforme pode ser visto na Tabela 7, o tempo computacional necessário para a convergência, por se tratar de uma abordagem com algoritmos exatos em um problema complexo, foi relativamente baixo.

Para um valor de data de entrega comum calculada com $h=0,2$, todas as instâncias obtiveram melhoras significativas no valor de função objetivo. Nos testes realizados com $h=0,4$, em 35 das 50 instâncias testadas, o valor de função objetivo melhorou com a vizinhança exata baseada em RINS, sendo que na maioria das instâncias em que não houve melhora o número de tarefas é elevado. Para um valor de data de entrega calculada com $h=0,6$, em 32 das 50 instâncias, o valor de função objetivo diminuiu. Nos testes realizados com $h=0,8$ o valor de função objetivo melhorou em 33 das 50 instâncias.

De uma forma geral, a diminuição no valor de função objetivo é maior quando o valor da data de entrega comum é calculado com $h=0,2$. Nos problemas com $h=0,6$ e $h=0,8$, a melhoria não é muito significativa. Entretanto, isto também acontece nos resultados do VNS, o que pode indicar que a heurística utilizada para a construção da solução inicial é mais eficiente para estas abordagens. Uma das razões pela qual a solução inicial não é tão eficiente para as abordagens com $h=0,2$ consiste no fato de que parte da Propriedade 3¹ do problema SMCDD não é levada em consideração na construção da solução.

¹Propriedade 3: Existe uma sequência ótima em que o processamento da primeira tarefa inicia no instante de tempo zero ou uma tarefa termina o seu processamento exatamente na data de entrega comum.

Tabela 4: Valores médios de função objetivo obtidos pela vizinhança exata baseada em RINS

Instâncias	h=0,2	h=0,4	h=0,6	h=0,8
sch20k1	4424,47	3067	3045,1	3034,17
sch20k2	8462,37	4847	3229,53	2980
sch20k3	6262,4	3857,07	3583	3583
sch20k4	9247,8	5118	3326	3048,1
sch20k5	4259,6	2533,53	2173	2173
sch20k6	6588,6	3582	3011,47	3010,27
sch20k7	10500,67	6289,57	4133	3884,13
sch20k8	3941,33	2145	1638	1638
sch20k9	3492,6	2097	1970,53	1970,13
sch20k10	5019,17	2958,4	2112,6	1995
sch50k1	41182,37	24586,03	17989,53	18019,9
sch50k2	31278,6	18273,03	14229,3	14080,43
sch50k3	35199,2	21337,76	16497	16497
sch50k4	28646,27	17243,53	14088	14088
sch50k5	32599,43	18418,3	14647,6	14650
sch50k6	36141,67	21826,17	14527	14066
sch50k7	43667,9	23659,1	17715	17715
sch50k8	44663,27	25376,4	21356,07	21355
sch50k9	35194,17	20698,47	14298	14050
sch50k10	33815,87	19991,8	14377	14367,93
sch100k1	150454,17	89547,7	72019	72019
sch100k2	130211,93	75584,77	59351	59351
sch100k3	132726,47	85323,6	68664	68664
sch100k4	133093,07	86759,17	69134,37	69127,7
sch100k5	130384,13	76179,97	55291	55277
sch100k6	143558,27	85021,03	62513,2	62516
sch100k7	139709,67	79854	62213	62208
sch100k8	164411,63	95361	80722	80722
sch100k9	120478,7	73248,67	58743	58743
sch100k10	123286,27	72434	61398	61398
sch200k1	520227,87	301603	254273	254273
sch200k2	562821,77	335478,33	266028	266028
sch200k3	515310,67	307771,97	254647	254647
sch200k4	602392	360852	297229	297229
sch200k5	543150,83	322159,67	260415	260410
sch200k6	495584,43	292453	235816	235816
sch200k7	478002,3	279674,33	247169	247169
sch200k8	522585,03	290719,27	225555	225555
sch200k9	567861,57	330466,53	254985	254985
sch200k10	563578,13	332808	268701	268701
sch500k1	3107103	1839902	1581563	1581563
sch500k2	3569903,6	2064923,5	1714485	1714485
sch500k3	3297446,87	1909304	1643752	1643752
sch500k4	3406338,87	1932396	1640856	1640856
sch500k5	3379874,5	1878023,87	1468321	1468321
sch500k6	3024440,27	1661308,4	1412611	1412611
sch500k7	3384329,3	1971221	1634502	1634502
sch500k8	3372243,27	1924152	1541587	1541587
sch500k9	3608223,17	2065647	1684055	1684055
sch500k10	3311688,4	1938870	1519539	1519539

Tabela 5: Limitantes superiores obtidos pela vizinhança exata baseada em RINS com $h=0,2$ e $h=0,4$

Instâncias	h=0,2			h=0,4		
	F_{SolIn}	$F_{VizRINS}$	Diferença	F_{SolIn}	$F_{VizRINS}$	Diferença
sch20k1	4431	4398	-33	3067	3067	0
sch20k2	8777	8430	-347	4897	4847	-50
sch20k3	6331	6210	-121	3883	3841	-42
sch20k4	9504	9188	-316	5122	5118	-4
sch20k5	4340	4215	-125	2571	2501	-70
sch20k6	6766	6527	-239	3631	3582	-49
sch20k7	11510	10455	-1055	6375	6264	-111
sch20k8	4203	3920	-283	2151	2145	-6
sch20k9	3530	3465	-65	2097	2097	0
sch20k10	5545	4979	-566	3632	2925	-707
sch50k1	42363	40805	-1558	24868	24219	-649
sch50k2	33663	30817	-2846	19387	18020	-1367
sch50k3	38196	34495	-3701	21353	20943	-410
sch50k4	30166	28091	-2075	17607	16875	-732
sch50k5	32604	32467	-137	18441	18043	-398
sch50k6	39611	35458	-4153	23966	20711	-3255
sch50k7	44277	43287	-990	23883	23206	-677
sch50k8	46065	43995	-2070	25402	25069	-333
sch50k9	37527	34391	-3136	22171	20082	-2089
sch50k10	36279	33157	-3122	20319	19375	-944
sch100k1	156496	147445	-9051	89588	88594	-994
sch100k2	132605	126604	-6001	75734	74614	-1120
sch100k3	137463	131195	-6268	85562	83174	-2388
sch100k4	137265	130385	-6880	88.107	83788	-4319
sch100k5	136761	127954	-8807	76424	74367	-2057
sch100k6	156828	141224	-15604	87080	81990	-5090
sch100k7	141613	137296	-4317	79854	79854	0
sch100k8	169445	161675	-7770	95361	95361	0
sch100k9	125153	118301	-6852	73605	71573	-2032
sch100k10	124446	121508	-2938	72434	72434	0
sch200k1	526792	507929	-18863	301603	301603	0
sch200k2	567848	554638	-13210	335714	332357	-3357
sch200k3	530145	504074	-26071	308278	302672	-5606
sch200k4	603709	597665	-6044	360852	360852	0
sch200k5	552280	533760	-18520	322268	319018	-3250
sch200k6	502276	486404	-15872	292453	292453	0
sch200k7	485829	469333	-16496	279686	279336	-350
sch200k8	530896	508704	-22192	290875	288519	-2356
sch200k9	585564	552530	-33034	331107	325636	-5471
sch200k10	572866	553446	-19420	332808	332808	0
sch500k1	3108242	3096243	-11999	1839902	1839902	0
sch500k2	3573361	3546227	-27134	2064998	2062821	-2177
sch500k3	3300744	3260044	-40700	1909304	1909304	0
sch500k4	3408867	3386159	-22708	1932396	1932396	0
sch500k5	3385927	3345998	-39929	1878157	1874453	-3704
sch500k6	3030468	2980116	-50352	1661437	1658343	-3094
sch500k7	3388581	3346887	-41694	1971221	1971221	0
sch500k8	3376678	3331247	-45431	1924158	1924152	-6
sch500k9	3617807	3579867	-37940	2065647	2065647	0
sch500k10	3316082	3287851	-28231	1938870	1938870	0

Tabela 6: Limitantes superiores obtidos pela vizinhança exata baseada em RINS com $h=0,6$ e $h=0,8$

Instâncias	h=0,6			h=0,8		
	F_{SolIn}	$F_{VizRINS}$	Diferença	F_{SolIn}	$F_{VizRINS}$	Diferença
sch20k1	3073	2987	-86	3073	2986	-87
sch20k2	3357	3206	-151	2980	2980	0
sch20k3	3583	3583	0	3583	3583	0
sch20k4	3336	3326	-10	3053	3040	-13
sch20k5	2173	2173	0	2173	2173	0
sch20k6	3012	3010	-2	3012	3010	-2
sch20k7	4175	4133	-42	3900	3878	-22
sch20k8	1638	1638	0	1638	1638	0
sch20k9	1980	1965	-15	1980	1965	-15
sch20k10	2116	2110	-6	1996	1995	-1
sch50k1	17990	17976	-14	18020	18017	-3
sch50k2	14231	14214	-17	14132	14050	-82
sch50k3	16497	16497	0	16497	16497	0
sch50k4	14105	14088	-17	14105	14088	-17
sch50k5	14650	14621	-29	14650	14650	0
sch50k6	14527	14527	0	14071	14066	-5
sch50k7	17715	17715	0	17715	17715	0
sch50k8	21427	21341	-86	21427	21355	-72
sch50k9	14298	14298	0	14050	14050	0
sch50k10	14377	14377	0	14368	14367	-1
sch100k1	72019	72019	0	72019	72019	0
sch100k2	59351	59351	0	59351	59351	0
sch100k3	68664	68664	0	68664	68664	0
sch100k4	69218	68971	-247	69218	69005	-213
sch100k5	55291	55291	0	55277	55277	0
sch100k6	62519	62432	-87	62519	62516	-3
sch100k7	62213	62213	0	62213	62208	-5
sch100k8	80751	80722	-29	80751	80722	-29
sch100k9	58771	58743	-28	58771	58743	-28
sch100k10	61403	61398	-5	61403	61398	-5
sch200k1	254274	254273	-1	254274	254273	-1
sch200k2	266028	266028	0	266028	266028	0
sch200k3	254647	254647	0	254647	254647	0
sch200k4	297269	297229	-40	297269	297229	-40
sch200k5	260428	260415	-13	260428	260410	-18
sch200k6	235816	235816	0	235816	235816	0
sch200k7	247176	247169	-7	247176	247169	-7
sch200k8	225572	225555	-17	225572	225555	-17
sch200k9	255029	254985	-44	255029	254985	-44
sch200k10	268782	268701	-81	268782	268701	-81
sch500k1	1581616	1581563	-53	1581616	1581563	-53
sch500k2	1714665	1714485	-180	1714665	1714485	-180
sch500k3	1643767	1643752	-15	1643767	1643752	-15
sch500k4	1640873	1640856	-17	1640873	1640856	-17
sch500k5	1468321	1468321	0	1468321	1468321	0
sch500k6	1412614	1412611	-3	1412614	1412611	-3
sch500k7	1634520	1634502	-18	1634520	1634502	-18
sch500k8	1541621	1541587	-34	1541621	1541587	-34
sch500k9	1684055	1684055	0	1684055	1684055	0
sch500k10	1519580	1519539	-41	1519580	1519539	-41

Tabela 7: Tempo médio (em segundos) da vizinhança exata baseada em RINS

h	N=20	N=50	N=100	N=200	N=500
0.2	209,27	424,22	557,12	730,36	1416,15
0.4	201,67	426,80	560,76	714,11	1426,76
0.6	197,23	422,50	564,27	744,05	1459,48
0.8	199,91	423,82	564,30	746,82	1477,32
Média	202,02	424,34	561,61	733,84	1444,93

4.5 VNS

Nesta seção são apresentados os resultados obtidos pelo algoritmo VNS descrito na Seção 3.2. Os testes foram realizados com as instâncias de 20, 50, 100, 200, 500 e 1000 tarefas e os valores de data de entrega comum foram calculados com $h=0,2$, $h=0,4$, $h=0,6$ e $h=0,8$. Para cada problema, foram realizadas 30 execuções. O critério de parada foi o número de iterações.

O tamanho da busca realizada nas vizinhanças do VNS pode ser ajustado. Isto é feito pois, ao explorar todos os vizinhos de todas as vizinhanças, a busca pode se tornar exaustiva. Neste trabalho, foram usados valores próximos do trabalho de Coelho (2016) para limitar o tamanho do espaço de busca. Os valores usados, para cada vizinhança, foram:

Vizinhança $V^{(1)}$: N

Vizinhança $V^{(2)}$: $3 \times N/2$

Vizinhança $V^{(3)}$: $9 \times N/2$

Vizinhança $V^{(4)}$: $27 \times N/2$

Os valores de função objetivo médios obtidos pelo VNS podem ser observados na Tabela 8. Na primeira coluna da tabela são descritas as instâncias. Nas quatro colunas seguintes são apresentados os valores médios de função objetivo para $h=0,2$, $h=0,4$, $h=0,6$ e $h=0,8$, respectivamente.

As Tabelas 9 e 10 apresentam os limitantes superiores obtidos pelo VNS. Na primeira coluna da tabela são descritas as instâncias e três colunas da tabela são utilizadas para apresentar os resultados, sendo que a coluna F_{BF} contém os valores definidos como limitantes superiores no trabalho de Biskup e Feldmann (2001), a coluna F_{VNS} apresenta os melhores valores de função objetivo obtidos através do VNS e a coluna DPR apresenta o desvio percentual relativo da melhor solução encontrada, calculado através

da Equação 4.1. A Tabela 9 apresenta os resultados obtidos para datas de entregas comum calculadas com $h=0,2$ e $h=0,4$ e a Tabela 10 apresenta os resultados quando as datas de entrega comum foram calculadas com $h=0,6$ e $h=0,8$. Os tempos médios de execução do algoritmo podem ser vistos na Tabela 11.

$$DPR = 100 \times (F_{VNS} - F_{BF}) / F_{BF} \quad (4.1)$$

Análise dos resultados

A partir das Tabelas 9 e 10, é possível observar que a abordagem do VNS melhorou os valores de função objetivo obtidos pelas heurísticas desenvolvidas por Biskup e Feldmann (2001) em 95% das instâncias. Conforme pode ser visto na Tabela 11, o tempo computacional requerido para a otimização foi baixo para as instâncias com até 200 tarefas. Para as instâncias com 500 e 1000 tarefas, o tempo computacional é maior.

Através do parâmetro DPR, é possível observar que, de uma forma geral, a redução no valor de função objetivo é maior quando o valor da data de entrega comum é calculado com $h=0,2$. Nos problemas com $h=0,6$ e $h=0,8$, a melhoria é menos significativa. Como parte da análise dos resultados, uma comparação com os resultados da literatura será realizada na próxima seção.

4.6 COMPARAÇÃO COM A LITERATURA

Com a finalidade de demonstrar o desempenho do algoritmo desenvolvido neste trabalho, foi realizada uma comparação dos valores de função objetivo obtidos pelo VNS com 15 heurísticas e metaheurísticas que foram utilizadas para resolver o problema SMCDD: GBA (YUCE et al., 2017), PHVNS (LIU & ZHOU, 2013), SS (TALEBI et al., 2009), MSE (WENG & FUJIMURA, 2008a), SE (WENG & FUJIMURA, 2008b), DACO (LEE; LIN & YING, 2008), DE (NEARCHOU, 2008), DDE (TASGETIREN et al., 2007), VNS/TS (LIAO & CHENG, 2007), BEES (PHAM et al., 2007), SEA (LIN; CHOU & YING, 2007), GA/ SA (LIN; CHOU & CHEN, 2007), DPSO (PAN; TASGETIREN & LIANG, 2006) e HTG e HGT (HINO; RONCONI & MENDES, 2005).

Nos trabalhos descritos anteriormente, foi disponibilizado os valores de Desvio Percentual Relativo (DPR) calculados pelos autores utilizando como limitante superior os valores de função objetivo obtidos pelas heurísticas desenvolvidas por Biskup e Feldmann (2001). Os resultados obtidos por cada um dos algoritmos podem ser observados

na Tabela 12. A primeira coluna contém a quantidade de tarefas da instância, a segunda apresenta os valores de h e a partir da terceira coluna são mostrados os valores de DPR de cada um dos algoritmos.

A partir da análise da Tabela 12, é possível observar que os melhores valores de DPR foram obtidos pelo VNS proposto, PHVNS, MSE, SE, DACO e GA/SA. Estes trabalhos não apresentaram os melhores resultados nas instâncias de 20, 50 e 100 tarefas com $h=0,6$ e $h=0,8$, estes foram reportados no algoritmo SS. Entretanto, apesar do SS apresentar um bom desempenho nestas instâncias, em outras teve um desempenho inferior que a maioria dos outros algoritmos.

Sendo assim, os resultados mostram que os valores de função objetivo obtidos pela abordagem proposta foram similares aos melhores resultados reportados na literatura. Esta comparação levou em consideração apenas os valores de função objetivo reportados pelos autores, entretanto, outros parâmetros como tempo computacional e número de avaliações da função objetivo podem ser empregados para avaliar a eficiência dos algoritmos.

Tabela 8: Valores médios de função objetivo obtidos pelo VNS

Instâncias	h=0,2	h=0,4	h=0,6	h=0,8
sch20k1	4394	3066,8	2986	2986
sch20k2	8430	4847	3206	2980
sch20k3	6210	3838	3583	3583
sch20k4	9188	5118	3317	3040
sch20k5	4215	2495	2173	2173
sch20k6	6527	3582	3010	3010
sch20k7	10455	6238	4126	3878
sch20k8	3920	2145	1638	1638
sch20k9	3465	2096	1965	1965
sch20k10	4979	2925	2110	1995
sch50k1	40697	23792	17969	17934,4
sch50k2	30613	17907	14050	14040
sch50k3	34425	20500	16497	16497
sch50k4	27755	16657	14080	14080
sch50k5	32307,6	18007	14605,4	14612
sch50k6	34969	20385	14251	14066
sch50k7	43134,1	23038	17616	17616
sch50k8	43839	24888	21329	21329
sch50k9	34228	19984	14202	13943,4
sch50k10	32958	19167	14366	14363,1
sch100k1	145516	85884,4	72017	72017
sch100k2	124916	72983,3	59230	59230
sch100k3	129801	79598	68537	68537
sch100k4	129584	79405	68760	68760
sch100k5	124351	71275	55286	55103
sch100k6	139189,6	77778	62398,2	62398,4
sch100k7	135026,6	78245,1	62197	62197
sch100k8	160147	94365	80708,7	80708,1
sch100k9	116536,4	69457,4	58727	58728,5
sch100k10	118911	71851,2	61361	61361
sch200k1	498653	295703,1	254259	254260
sch200k2	541181,7	319205,6	266002	266006,2
sch200k3	488665	293886	254477,1	254476
sch200k4	586259,5	353034	297109	297111
sch200k5	513224,6	304663,2	260282	260279,4
sch200k6	478027,4	279924,9	235702	235702,8
sch200k7	454758,3	275038,4	246307	246307
sch200k8	494288,9	279176,3	225215	225215
sch200k9	529280,4	310402,1	254639,1	254637
sch200k10	538332,6	323095,5	268354,7	268358,2
sch500k1	2954986,4	1787693,4	1579031	1579036,1
sch500k2	3365951,9	1994796	1712204	1712208,3
sch500k3	3102639,8	1864388,3	1641440,2	1641438
sch500k4	3221024,6	1887343	1640790,4	1640783,7
sch500k5	3114784,7	1806998,9	1468231	1468246
sch500k6	2792264,4	1610046,8	1411831	1411830
sch500k7	3172425,9	1902705,2	1634330	1634330,1
sch500k8	3122304,3	1819217,9	1540378,9	1540378,4
sch500k9	3364310,1	1973668,5	1680204,5	1680196,2
sch500k10	3120442,4	1837420,6	1519181	1519182
sch1000k1	14055330,8	8111601	6410878	6410879,5
sch1000k2	12296600,9	7271953,2	6110135,1	6110105,1
sch1000k3	11967829,6	6987341,5	5983352,7	5983332,2
sch1000k4	11797133,9	7024547,7	6085867,1	6085896,6
sch1000k5	12449845,6	7365369,5	6341524,9	6341508,7
sch1000k6	11644393,4	6928075,9	6078410,6	6078379,2
sch1000k7	13277301,8	7861834,4	6574311,1	6574308,4
sch1000k8	12275188,1	7222809,4	6067324,9	6067325,6
sch1000k9	11757450,9	7059317,5	6185353,3	6185350,1
sch1000k10	12428048	7276251,4	6145794	6145802,1

Tabela 9: Limitantes superiores obtidos pelo VNS com $h=0,2$ e $h=0,4$

Instâncias	$h=0,2$			$h=0,4$		
	F_{BF}	F_{VNS}	DPR	F_{BF}	F_{VNS}	DPR
sch20k1	4431	4394	-0,835	3066	3066	0
sch20k2	8567	8430	-1,599	4897	4847	-1,021
sch20k3	6331	6210	-1,911	3883	3838	-1,1589
sch20k4	9478	9188	-3,06	5122	5118	-0,0781
sch20k5	4340	4215	-2,88	2571	2495	-2,956
sch20k6	6766	6527	-3,532	3601	3582	-0,5276
sch20k7	11101	10455	-5,819	6357	6238	-1,872
sch20k8	4203	3920	-6,733	2151	2145	-0,2789
sch20k9	3530	3465	-1,841	2097	2096	-0,0477
sch20k10	5545	4979	-10,21	3192	2925	-8,3647
DPR Médio			-3,84			-1,63
sch50k1	42363	40697	-3,933	24868	23792	-4,3268
sch50k2	33637	30613	-8,99	19279	17907	-7,1166
sch50k3	37641	34425	-8,544	21353	20500	-3,9948
sch50k4	30166	27755	-7,992	17495	16657	-4,7899
sch50k5	32604	32307	-0,911	18441	18007	-2,3535
sch50k6	36920	34969	-5,284	21497	20385	-5,1728
sch50k7	44277	43134	-2,581	23883	23038	-3,5381
sch50k8	46065	43839	-4,832	25402	24888	-2,0235
sch50k9	36397	34228	-5,959	21929	19984	-8,8695
sch50k10	35797	32958	-7,931	20048	19167	-4,3945
DPR Médio			-5,70			-4,66
sch100k1	156103	145516	-6,782	89588	85884	-4,1345
sch100k2	132605	124916	-5,798	74854	72981	-2,5022
sch100k3	137463	129800	-5,575	85363	79598	-6,7535
sch100k4	137265	129584	-5,596	87.730	79405	-9,4893
sch100k5	136761	124351	-9,074	76424	71275	-6,7374
sch100k6	151938	139188	-8,392	86724	77778	-10,315
sch100k7	141613	135026	-4,651	79854	78244	-2,0162
sch100k8	168086	160147	-4,723	95361	94365	-1,0445
sch100k9	125153	116522	-6,896	73605	69457	-5,6355
sch100k10	124446	118911	-4,448	72399	71850	-0,7583
DPR Médio			-6,19			-4,94
sch200k1	526666	498653	-5,319	301449	295686	-1,9118
sch200k2	566643	541180	-4,494	335714	319199	-4,9194
sch200k3	529919	488665	-7,785	308278	293886	-4,6685
sch200k4	603709	586257	-2,891	360852	353034	-2,1665
sch200k5	547953	513217	-6,339	322268	304662	-5,4632
sch200k6	502276	478022	-4,829	292453	279920	-4,2855
sch200k7	479651	454757	-5,19	279576	275030	-1,626
sch200k8	530896	494287	-6,896	288746	279172	-3,3157
sch200k9	575353	529275	-8,009	331107	310400	-6,2539
sch200k10	572866	538332	-6,028	332808	323094	-2,9188
DPR Médio			-5,78			-3,75
sch500k1	3113088	2954852	-5,083	1839902	1787693	-2,8376
sch500k2	3569058	3365933	-5,691	2064998	1994796	-3,3996
sch500k3	3300744	3102612	-6,003	1909304	1864365	-2,3537
sch500k4	3408867	3221011	-5,511	1930829	1887321	-2,2533
sch500k5	3377547	3114780	-7,78	1881221	1806979	-3,9465
sch500k6	3024082	2792231	-7,667	1658411	1610015	-2,9182
sch500k7	3381166	3172398	-6,174	1971176	1902680	-3,4749
sch500k8	3376678	3122282	-7,534	1924191	1819186	-5,4571
sch500k9	3617807	3364310	-7,007	2065647	1973635	-4,4544
sch500k10	3315019	3120383	-5,871	1928579	1837378	-4,7289
DPR Médio			-6,43			-3,58
sch1000k1	15190371	14054945	-7,475	8570154	8111169	-5,3556
sch1000k2	13356727	12296071	-7,941	7592040	7271497	-4,2221
sch1000k3	12919259	11967381	-7,368	7313736	6986985	-4,4676
sch1000k4	12705290	11796810	-7,15	7300217	7024153	-3,7816
sch1000k5	13276868	12449608	-6,231	7738367	7365057	-4,8241
sch1000k6	12236080	11644148	-4,838	7144491	6927810	-3,0328
sch1000k7	14160773	13277124	-6,24	8426024	7861510	-6,6996
sch1000k8	13314723	12274978	-7,809	7508507	7222502	-3,8091
sch1000k9	12433821	11757144	-5,442	7299271	7058934	-3,2926
sch1000k10	13395234	12427451	-7,225	7617658	7276020	-4,4848
DPR Médio			-6,77			-4,40

Tabela 10: Limitantes superiores obtidos pelo VNS com $h=0,6$ e $h=0,8$

Instâncias	F_{BF}	h=0,6		F_{BF}	h=0,8	
		F_{VNS}	DPR		F_{VNS}	DPR
sch20k1	2986	2986	0	2986	2986	0
sch20k2	3260	3206	-1,6564	2980	2980	0
sch20k3	3600	3583	-0,4722	3600	3583	-0,472
sch20k4	3336	3317	-0,5695	3040	3040	0
sch20k5	2206	2173	-1,4959	2206	2173	-1,496
sch20k6	3016	3010	-0,1989	3016	3010	-0,199
sch20k7	4175	4126	-1,1737	3900	3878	-0,564
sch20k8	1638	1638	0	1638	1638	0
sch20k9	1992	1965	-1,3554	1992	1965	-1,355
sch20k10	2116	2110	-0,2836	1995	1995	0
DPR Médio			-0,72			-0,41
sch50k1	17990	17969	-0,1167	17990	17934	-0,311
sch50k2	14231	14050	-1,2719	14132	14040	-0,651
sch50k3	16497	16497	0	16497	16497	0
sch50k4	14105	14080	-0,1772	14105	14080	-0,177
sch50k5	14650	14605	-0,3072	14650	14612	-0,259
sch50k6	14251	14251	0	14075	14066	-0,064
sch50k7	17715	17616	-0,5588	17715	17616	-0,559
sch50k8	21367	21329	-0,1778	21367	21329	-0,178
sch50k9	14298	14202	-0,6714	13952	13942	-0,072
sch50k10	14377	14366	-0,0765	14377	14363	-0,097
DPR Médio			-0,34			-0,24
sch100k1	72019	72017	-0,0028	72019	72017	-0,003
sch100k2	59351	59230	-0,2039	59351	59230	-0,204
sch100k3	68537	68537	0	68537	68537	0
sch100k4	69231	68760	-0,6803	69231	68760	-0,68
sch100k5	55291	55286	-0,009	55277	55103	-0,315
sch100k6	62519	62398	-0,1935	62519	62398	-0,194
sch100k7	62213	62197	-0,0257	62213	62197	-0,026
sch100k8	80844	80708	-0,1682	80844	80708	-0,168
sch100k9	58771	58727	-0,0749	58771	58727	-0,075
sch100k10	61419	61361	-0,0944	61419	61361	-0,094
DPR Médio			-0,15			-0,18
sch200k1	254268	254259	-0,0035	254268	254260	-0,003
sch200k2	266028	266002	-0,0098	266028	266006	-0,008
sch200k3	254647	254476	-0,0672	254647	254476	-0,067
sch200k4	297269	297109	-0,0538	297269	297111	-0,053
sch200k5	260455	260278	-0,068	260455	260278	-0,068
sch200k6	236160	235702	-0,1939	236160	235702	-0,194
sch200k7	247555	246307	-0,5041	247555	246307	-0,504
sch200k8	225572	225215	-0,1583	225572	225215	-0,158
sch200k9	255029	254637	-0,1537	255029	254637	-0,154
sch200k10	269236	268353	-0,328	269236	268353	-0,328
DPR Médio			-0,15			-0,15
sch500k1	1581233	1579031	-0,1393	1581233	1579031	-0,139
sch500k2	1715332	1712195	-0,1829	1715322	1712195	-0,182
sch500k3	1644947	1641438	-0,2133	1644947	1641438	-0,213
sch500k4	1640942	1640783	-0,0097	1640942	1640783	-0,01
sch500k5	1468325	1468231	-0,0064	1468325	1468231	-0,006
sch500k6	1413345	1411831	-0,1071	1413345	1411830	-0,107
sch500k7	1634912	1634330	-0,0356	1634912	1634330	-0,036
sch500k8	1542090	1540377	-0,1111	1542090	1540377	-0,111
sch500k9	1684055	1680199	-0,229	1684055	1680187	-0,23
sch500k10	1520515	1519181	-0,0877	1520515	1519182	-0,088
DPR Médio			-0,11			-0,11
sch1000k1	6411581	6410875	-0,011	6411581	6410875	-0,011
sch1000k2	6112598	6110091	-0,041	6112598	6110091	-0,041
sch1000k3	5985538	5983303	-0,0373	5985538	5983307	-0,037
sch1000k4	6096729	6085850	-0,1784	6096729	6085849	-0,178
sch1000k5	6348242	6341478	-0,1065	6348242	6341478	-0,107
sch1000k6	6082142	6078373	-0,062	6082142	6078373	-0,062
sch1000k7	6575879	6574297	-0,0241	6575879	6574299	-0,024
sch1000k8	6069658	6067312	-0,0387	6069658	6067312	-0,039
sch1000k9	6188416	6185321	-0,05	6188416	6185321	-0,05
sch1000k10	6147295	6145737	-0,0253	6147295	6145737	-0,025
DPR Médio			-0,06			-0,06

Tabela 11: Tempo médio (em segundos) do VNS

h	N=20	N=50	N=100	N=200	N=500	N=1000
0,2	0,281	0,760	1,917	5,644	50,956	936,816
0,4	0,287	0,887	1,989	6,426	58,634	1236,997
0,6	0,502	1,716	4,406	12,025	87,28	1072,042
0,8	0,519	1,895	4,782	12,277	81,759	820,230
Média	0,397	1,314	3,274	9,093	69,657	1016,521

Tabela 12: Valores de DPR obtidos por algoritmos reportados na literatura

N	h	VNS Proposto	GBA	PHVNS	SS	MSE	SE	DACO	DE	DDE	VNS/TS	BEES	SEA	GA /SA	DPSO	HTG	HGT
10	0,2	0	0	0	0,33	0	0	-	0	0	0	0	-	0	0	0,12	0,12
	0,4	0	0	0	0,19	0	0	-	0	0	0	0	-	0	0	0,19	0,19
	0,6	0	0	0	1,54	0	0	-	0	0	0	0	-	0	0	0,03	0,01
	0,8	0	0	0	0,7	0	0	-	0	0	0	0	-	0	0	0	0
20	0,2	-3,84	-3,84	-3,84	-3,57	-3,84	-3,84	-	-3,84	-3,84	-3,84	-3,84	-3,78	-3,84	-3,84	-3,84	-3,84
	0,4	-1,63	-1,63	-1,63	-0,85	-1,63	-1,63	-	-1,63	-1,63	-1,63	-1,63	-1,58	-1,63	-1,63	-1,62	-1,62
	0,6	-0,72	-0,72	-0,72	-2,9	-0,72	-0,72	-	-0,72	-0,72	-0,72	-0,72	-0,64	-0,72	-0,72	-0,71	-0,71
	0,8	-0,41	-0,41	-0,41	-6,82	-0,41	-0,41	-	-0,41	-0,41	-0,41	-0,41	-0,38	-0,41	-0,41	-0,41	-0,41
50	0,2	-5,7	-5,7	-5,7	-5,23	-5,7	-5,7	-5,7	-5,69	-5,69	-5,7	-5,7	-5,58	-5,7	-5,7	-5,7	-5,7
	0,4	-4,66	-4,66	-4,66	-4,05	-4,66	-4,66	-4,66	-4,66	-4,66	-4,66	-4,66	-4,42	-4,66	-4,66	-4,66	-4,66
	0,6	-0,34	-0,34	-0,34	-1,63	-0,34	-0,34	-0,34	-0,32	-0,34	-0,34	-0,34	-0,31	-0,34	-0,34	-0,27	-0,31
	0,8	-0,24	-0,24	-0,24	-3,13	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,23	-0,23
100	0,2	-6,19	-6,19	-6,19	-5,82	-6,19	-6,19	-6,19	-6,17	-6,19	-6,19	-6,19	-6,12	-6,19	-6,19	-6,19	-6,19
	0,4	-4,94	-4,94	-4,94	-4,28	-4,94	-4,94	-4,94	-4,89	-4,94	-4,94	-4,94	-4,85	-4,94	-4,94	-4,93	-4,93
	0,6	-0,15	-0,15	-0,15	-0,27	-0,15	-0,15	-0,15	-0,13	-0,15	-0,15	-0,15	-0,15	-0,15	-0,15	-0,08	0,04
	0,8	-0,18	-0,18	-0,18	0,37	-0,18	-0,18	-0,18	-0,17	-0,18	-0,18	-0,18	-0,18	-0,18	-0,18	-0,08	-0,11
200	0,2	-5,78	-5,78	-5,78	-5,37	-5,78	-5,78	-5,78	-5,77	-5,78	-5,78	-5,78	-5,78	-5,78	-5,78	-5,76	-5,76
	0,4	-3,75	-3,75	-3,75	-3,12	-3,75	-3,75	-3,75	-3,72	-3,75	-3,75	-3,75	-3,73	-3,75	-3,75	-3,75	-3,75
	0,6	-0,15	-0,15	-0,15	0,19	-0,15	-0,15	-0,15	0,23	-0,15	-0,15	-0,15	-0,15	-0,15	-0,15	0,37	0,07
	0,8	-0,15	-0,15	-0,15	0,43	-0,15	-0,15	-0,15	0,2	-0,15	-0,15	-0,15	-0,15	-0,15	-0,15	0,26	0,07
500	0,2	-6,43	-6,43	-6,43	-5,93	-6,43	-6,43	-6,43	-6,43	-6,43	-6,42	-6,43	-6,43	-6,43	-6,42	-6,41	-6,41
	0,4	-3,58	-3,57	-3,58	-3,06	-3,58	-3,58	-3,58	-3,57	-3,56	-3,56	-3,57	-3,57	-3,58	-3,56	-3,58	-3,58
	0,6	-0,11	-0,11	-0,11	0,31	-0,11	-0,11	-0,11	1,72	-0,11	-0,11	-0,11	-0,11	-0,11	-0,11	0,73	0,15
	0,8	-0,11	-0,11	-0,11	0,38	-0,11	-0,11	-0,11	1,01	-0,11	-0,11	-0,11	-0,11	-0,11	-0,11	0,73	0,13
1000	0,2	-6,77	-6,76	-6,77	-6,18	-6,77	-6,77	-6,77	-6,72	-6,76	-6,75	-6,76	-6,77	-6,77	-6,76	-6,74	-6,74
	0,4	-4,4	-4,35	-4,4	-3,76	-4,4	-4,39	-4,4	-4,38	-4,38	-4,37	-4,35	-4,4	-4,4	-4,37	-4,39	-4,39
	0,6	-0,06	-0,05	-0,06	0,71	-0,06	-0,06	-0,06	1,29	-0,06	-0,05	-0,05	-0,06	-0,06	-0,06	1,28	0,42
	0,8	-0,06	-0,05	-0,06	0,71	-0,06	-0,06	-0,06	2,79	-0,06	-0,05	-0,05	-0,06	-0,06	-0,06	1,28	0,4
Média		-2,155	-2,152	-2,155	-2,147	-2,155	-2,155	-	-1,865	-2,153	-2,152	-2,152	-2,127	-2,155	-2,153	-1,941	-2,062

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo estudar heurísticas matemáticas para o problema de sequenciamento de tarefas em uma única máquina com data de entrega comum, onde o objetivo é minimizar penalidades decorrentes de atrasos e adiantamentos nas entregas das tarefas. Os testes computacionais foram realizados utilizando instâncias *benchmark* do problema disponíveis na literatura.

Como parte deste estudo, dois modelos matemáticos para o problema SMCDD foram definidos e validados. A validação dos resultados foi realizada utilizando as instâncias de 10 e 20 tarefas e os resultados mostraram que para a construção de vizinhanças exatas cuja convergência para a otimalidade deve ocorrer em um tempo computacional baixo, a otimização deve ocorrer com poucas tarefas. As instâncias com 10 tarefas convergiram em poucos segundos, porém as instâncias com 20 tarefas terminaram a otimização quando o tempo limite de 5 min foi atingido, com valores de GAPs elevados.

Após realizado o estudo dos modelos matemáticos, cinco vizinhanças com métodos de programação matemática foram implementadas utilizando estratégias de fixação de variáveis forte e fraca. As vizinhanças com estratégia de fixação forte foram inspiradas no algoritmo FixOpt e no algoritmo RINS. A vizinhança com estratégia de fixação fraca implementada foi baseada no algoritmo *Local Branching*. O desempenho das vizinhanças implementadas foi avaliado com os dois modelos matemáticos.

Dentre as vizinhanças implementadas, a inspirada no algoritmo RINS foi a que apresentou os melhores resultados, sendo a utilizada na abordagem proposta no trabalho para resolução do problema SMCDD. Esta vizinhança foi utilizada para refinar o resultado obtido pela heurística implementada para a construção da solução inicial. Os resultados mostraram que houve uma redução no valor de função objetivo através desta vizinhança em 75% das instâncias testadas, sendo que a melhoria foi mais significativa quando a data de entrega comum foi calculada com $h=0,2$.

As melhores soluções obtidas através da vizinhança exata baseada em RINS foram utilizadas como solução inicial do algoritmo VNS, que foi utilizado para a obtenção dos resultados finais do trabalho. As estruturas de vizinhança implementadas no VNS levaram em consideração as propriedades do problema, o que as faz tornar mais eficientes do ponto de vista de qualidade. Os valores de função objetivo obtidos pelo algoritmo VNS são similares aos melhores resultados reportados na literatura, o que mostra a eficácia da abordagem proposta. Como contribuições principais deste trabalho estão o estudo dos modelos matemáticos e a proposição de vizinhanças exatas para o problema SMCDD.

5.1 TRABALHOS FUTUROS

Algumas sugestões de trabalhos futuros são:

- Determinar parâmetros para a vizinhança exata baseada em RINS para as instâncias com 1000 tarefas ou mais;
- Implementar outras vizinhanças exatas com os modelos matemáticos apresentados, explorando outras topologias de heurísticas matemáticas;
- Implementar vizinhanças exatas utilizando os dois modelos matemáticos simultaneamente;
- Aplicação do método desenvolvido em problemas reais.

5.2 PUBLICAÇÕES

ANTUNES, N.; CARRANO, E. G. Estudo de modelos matemáticos de um problema de sequenciamento de tarefas com foco na construção de heurísticas matemáticas. Em: *51º Simpósio Brasileiro de Pesquisa Operacional*. 2019. Limeira-SP.

REFERÊNCIAS

- BAGCHI, U.; CHANG, Y.; SULLIVAN, R. Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalties and a common due date. *Naval Research Logistics*, v. 34, p. 739–751, 1987.
- BAGCHI, U.; SULLIVAN, R. S.; CHANG, Y. L. Minimizing mean absolute deviation of completion times about a common due date. *Naval Research Logistics Quarterly*, v. 33, p. 227–240, 1986.
- BAKER, K. R.; SCUDDER, G. D. Sequencing with earliness and tardiness penalties: A review. *Operations Research*, v. 38, p. 22–36, 1990.
- BISKUP, D.; FELDMANN, M. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers & Operations Research*, v. 28, p. 787–801, 2001.
- CAMARGO, V.; TOLEDO, F. M.; ALMADA-LOBO, B. Hops-hamming-oriented partition search for production planning in the spinning industry. *European Journal of Operational Research*, v. 234, p. 266–277, 2014.
- CHENG, T.; GUPTA, M. C. Survey of scheduling research involving due date determination decision. *European Journal of Operational Research*, v. 38, p. 156–166, 1989.
- CHENG, T. C. E.; KAHLBACHER, H. G. A proof for the longest-job-first policy in one-machine scheduling. *Naval Research Logistics*, v. 38, p. 715–720, 1991.
- COELHO, D. G. *Estruturas de Memórias e Operadores Adaptativos em Meta-Heurísticas*. Tese (Doutorado em Engenharia Elétrica) — Universidade Federal de Minas Gerais, 2016.
- CROCE, F. D.; SALASSA, F.; T’KINDT, V. A hybrid heuristic approach for single machine scheduling with release times. *Computers & Operations Research*, v. 45, p. 7–11, 2014.
- DANNA, E.; ROTHBERG, E.; PAPE, C. L. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, v. 102, p. 71–90, 2005.
- DE, P.; GHOSH, J. B.; WELLS, C. E. Solving a generalized model for CON due date assignment and sequencing. *International Journal of Production Economics*, v. 34, p. 179–185, 1994.
- FELDMANN, M.; BISKUP, D. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers & Industrial Engineering*, v. 44, p. 307–323, 2003.

- FISCHETTI, M.; LODI, A. Local branching. *Mathematical Programming*, v. 98, p. 23–47, 2003.
- FONSECA, G. H.; SANTOS, H. G.; CARRANO, E. G. Integrating matheuristics and metaheuristics for timetabling. *Computers & Operations Research*, v. 74, p. 108–117, 2016.
- GORDON, V.; PROTH, J.-M.; CHU, C. A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, v. 139, p. 1–25, 2002.
- HALL, N. Single and multi-processor models for minimizing completion time variance. *Naval Research Logistics Quarterly*, v. 33, p. 49–54, 1986.
- HALL, N.; KUBIAK, W.; SETHI, S. Earliness-tardiness scheduling problems, ii: Deviation of completion times about a restrictive common due date. *Operations Research*, v. 39, p. 847–856, 1991.
- HALL, N. G.; POSNER, M. E. Earliness-Tardiness Scheduling Problems, I: Weighted Deviation of Completion Times About a Common Due Date. *Operations Research*, v. 39, p. 836–846, 1991.
- HINO, C.; RONCONI, D.; MENDES, A. Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research*, v. 160, p. 190–201, 2005.
- HOOGEVEEN, H.; VELDE, S. V. D. Earliness-tardiness scheduling around almost equal due dates. *INFORMS Journal on Computing*, v. 9, p. 92–99, 1997.
- HOOGEVEEN, J. A.; VELDE, S. L. Van de. Scheduling around a small common due date. *Journal of Operational Research*, v. 55, p. 237–242, 1991.
- KANET, J. J. Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly*, v. 28, p. 643–651, 1981.
- KEHA, A. B.; KHOWALA, K.; FOWLER, J. W. Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering*, v. 56, p. 357–367, 2009.
- LEE, C.-Y.; DANUSAPUTRO, S. L.; LIN, C.-S. Minimizing weighted number of tardy jobs and weighted earliness-tardiness penalties about a common due date. *Computers & Operations Research*, v. 18, p. 379–389, 1991.
- LEE, Z.-J.; LIN, S.-W.; YING, K.-C. A dynamical ant colony optimization with heuristics for scheduling jobs on a single machine with a common due date. In: SPRINGER. *Metaheuristics for scheduling: in industrial and manufacturing applications*. 2008. (Studies in Computational Intelligence, v. 128), p. 91–103.
- LIAO, C.-J.; CHENG, C.-C. A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. *Computers & Industrial Engineering*, v. 52, p. 404–413, 2007.

- LIN, S.; YING, K. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. *Omega*, v. 64, p. 115–125, 2016.
- LIN, S.-W.; CHOU, S.-Y.; CHEN, S.-C. Meta-heuristic approaches for minimizing total earliness and tardiness penalties of single-machine scheduling with a common due date. *Journal of Heuristics*, v. 13, p. 151–165, 2007.
- LIN, S.-W.; CHOU, S.-Y.; YING, K.-C. A sequential exchange approach for minimizing earliness-tardiness penalties of single-machine scheduling with a common due date. *European Journal of Operational Research*, v. 177, p. 1294–1301, 2007.
- LIU, L.; ZHOU, H. Hybridization of harmony search with variable neighborhood search for restrictive single-machine earliness/tardiness problem. *Information Sciences*, v. 226, p. 68–92, 2013.
- MLADENović, N.; HANSEN, P. Variable neighborhood search. *Computers & Operations Research*, v. 24, p. 1097–1100, 1997.
- NEARCHOU, A. A differential evolution approach for the common due date early/tardy job scheduling problem. *Computers & Operations Research*, v. 35, p. 1329–1343, 2008.
- PAN, Q.-K.; TASGETIREN, M.; LIANG, Y.-C. A discrete particle swarm optimization algorithm for single machine total earliness and tardiness problem with a common due date. In: *IEEE Congress on Evolutionary Computation*. 2006. p. 3281–3288.
- PANWALKAR, S. S.; SMITH, M. L.; SEIDMANN, A. Common due date assignment to minimize total penalty for the one machine scheduling problem. *Operational Research*, v. 30, p. 391–399, 1982.
- PHAM, D. et al. Using the bees algorithm to schedule jobs for a machine. In: *Eighth International Conference on Laser Metrology, CMM and Machine Tool Performance, LAMMAP*. 2007.
- TALBI, E.-G. Combining metaheuristics with mathematical programming, constraint programming and machine learning. *4OR*, v. 11, p. 101–150, 2013.
- TALEBI, J. et al. An efficient scatter search algorithm for minimizing earliness and tardiness penalties in a single-machine scheduling problem with a common due date. In: *IEEE Congress on Evolutionary Computation*. 2009. p. 1012–1018.
- TASGETIREN, M. F. et al. A discrete differential evolution algorithm for the total earliness and tardiness penalties with a common due date on a single-machine. In: *IEEE Symposium on Computational Intelligence in Scheduling*. 2007. p. 271–278.
- TOFFOLO, T. A. et al. An integer programming approach to the multimode resource-constrained multiproject scheduling problem. *Journal of Scheduling*, v. 19, p. 295–307, 2016.
- WENG, W.; FUJIMURA, S. Memorial self evolution algorithm to solve jit machine scheduling problem. In: *IEEE International Conference on Industrial Informatics*. 2008. p. 1706–1711.

WENG, W.; FUJIMURA, S. Self evolution algorithm for common due date scheduling problem. In: *4th IEEE Conference on Automation Science and Engineering*. 2008. p. 790–795.

YUCE, B. et al. Hybrid genetic bees algorithm applied to single machine scheduling with earliness and tardiness penalties. *Computers & Industrial Engineering*, v. 113, p. 842–858, 2017.