CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Diretoria de Pesquisa e Pós-Graduação

Programa de Pós-Graduação em Modelagem Matemática e Computacional

Projeto e análise de métodos computacionais para problemas de sequenciamento de tarefas em ambientes de maquinas paralelas não relacionadas

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG como parte dos requisitos necessários para obtenção do título de Doutor em Modelagem Matemática e Computacional.

Aluno : Rodney Oliveira Marinho Diana

Orientador : Prof. Dr. Sérgio Ricardo de Souza (CEFET-MG)

Coorientadora : Profa. Dra. Elizabeth Fialho Wanner (CEFET-MG)

Belo Horizonte - MG
Agosto de 2021

Diana, Rodney Oliveira Marinho
D538p Projeto e análise de métodos computacionais para problemas de sequenciamento de tarefas em ambientes de máquinas paralelas não relacionadas / Rodney Oliveira Marinho Diana. – 2021.
188 f.

Tese de doutorado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional.
Orientador: Sérgio Ricardo de Souza.
Coorientadora: Elizabeth Fialho Wanner.
Tese (doutorado) – Centro Federal de Educação Tecnológica de Minas Gerais.

1. Máquinas – Modelos matemáticos – Teses. 2. Processamento paralelo (computadores eletrônicos) – Teses. 3. Metaheurísticas – Teses. 4. Controle de qualidade – Teses. 5. Pesquisa operacional – Teses. I. Souza, Sérgio Ricardo de. II. Wanner, Elizabeth Fialho. III. Centro Federal de Educação Tecnológica de Minas Gerais. IV. Título.

CDD 519.6

Elaboração da ficha catalográfica pela bibliotecária Jane Marangon Duarte, CRB 6º 1592 / Cefet/MG

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

"PROJETO E ANÁLISE DE MÉTODOS COMPUTACIONAIS PARA PROBLEMAS DE SEQUENCIAMENTO DE TAREFAS EM AMBIENTES DE MÁQUINAS PARALELAS NÃO RELACIONADAS".

Tese de Doutorado apresentada por **Rodney Oliveira Marinho Diana**, em 27 de agosto de 2021, ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, e aprovada pela banca examinadora constituída pelos professores:

Prof. Dr. Sérgio Ricardo de Souza (Orientador)
Centro Federal de Educação Tecnológica de Minas Gerais

Profª. Drª. Elizabeth Fialho Wanner (Coorientadora)
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Luiz Satoru Ochi
Universidade Federal Fluminense

Prof. Dr. José Elias Claudio Arroyo
Universidade Federal de Viçosa

Prof. Dr. Moacir Felizardo de França Filho
Centro Federal de Educação Tecnológica de Minas Gerais

Profª. Drª. Elisangela Martins de Sá
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Marcone Jamilson Freitas Souza
Centro Federal de Educação Tecnológica de Minas Gerais

Visto e permitida à impressão,

Profª. Drª. Elizabeth Fialho Wanner
Presidenta do Colegiado do Programa de Pós-Graduação em Modelagem Matemática e Computacional

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

DECLARAÇÃO

Declaro que **RODNEY OLIVEIRA MARINHO DIANA** realizou todas as correções exigidas pela banca examinadora em sua Tese de Doutorado, intitulada *"Projeto e análise de métodos computacionais para problemas de sequenciamento de tarefas em ambientes de máquinas paralelas não relacionadas"*, cuja defesa foi realizada em de 27 de agosto de 2021, e que, portanto, o presente exemplar representa a versão final desse trabalho acadêmico, estando, assim, pronto para ser publicado.

Belo Horizonte, 07 / 01 / 2022

Prof. Dr. Sérgio Ricardo de Souza (Orientador)

Profª. Drª. Elizabeth Fialho Wanner (Coorientadora)

Agradecimentos

Primeiramente, agradeço ao Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) e ao Programa de Pós-Graduação em Modelagem Matemática e Computacional, pela oportunidade de realizar o doutoramento.

Agradeço ao meu orientador, Prof. Dr. Sérgio Ricardo de Souza, que sempre me motivou, me respeitou, depositou confiança no meu trabalho, além de ser sempre compreensível nos momentos difíceis e me dar sábios conselhos sempre que o procurei para conversar. Agradeço também a minha co-orientadora Profa. Dra. Elizabeth Fialho Wanner pela atenção, tempo despendido, paciência e, principalmente, pelos ensinamentos compartilhados.

Gostaria de agradecer também ao Prof. Dr. Moacir Felizardo de França Filho, por acreditar no meu potencial e me incentivar a ingressar no doutorado.

Agradeço a minha esposa Catarina Dias de Freitas pelo amor, motivação e suporte neste período.

Gostaria de agradecer a minha mãe pelo seu amor e incentivo aos meus estudos. Com certeza sem os seus sacrifícios eu não teria chegado até aqui.

Também gostaria de agradecer ao restante da minha família começando pelo meu pai Cláudio, irmão Rodrigo, sobrinho Logan e minha madrinha Maria José, que sempre me apoiaram durante este percurso.

A minha sogra Ana, pelo suporte e apoio, além da Mya e Saninho, pelas brincadeiras e distrações.

Ao meu amigo César Soares, pelas conversas e incentivo durante o trajeto.

Aos docentes do Programa de Pós-Graduação em Modelagem Matemática e Computacional, assim como os discentes, principalmente, Alexandre Frias, Eduardo Siqueira e demais alunos do grupo de otimização.

Ainda, gostaria de agradecer a todos os funcionários do CEFET-MG e pessoas que de alguma forma contribuíram para a realização desta tese.

Por fim, gostaria de agradecer à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), uma vez que o presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Esta tese estuda uma importante classe de problemas de *scheduling*, denominada sequenciamento de tarefas em máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência e das máquinas. Para isto, é realizada uma minuciosa revisão bibliográfica, que identifica três lacunas na literatura: (i) pouca importância é dada à construção e, principalmente, à validação de componentes utilizados em métodos de busca local e, assim, não há informações suficientes para se determinar padrões que podem potencializar a construção de operadores de busca local; (ii) as metaheurísticas construídas para resolução dos problemas incorporam muitas características dos critérios de otimização no processo de busca, dificultando a adaptação destas a critérios de otimização com pouca visibilidade teórica; (iii) não são encontrados estudos para problemas que envolvam o sequenciamento guiado por uma política *Just-in-Time* (JIT), considerando janelas de tempo. Esta tese tem como principal objetivo apresentar três estudos a respeito dessas lacunas. No primeiro estudo é proposta uma metodologia para o projeto de operadores de busca local de metaheurísticas. Esta metodologia é avaliada em três metaheurísticas propostas previamente na literatura. O estudo mostra que a utilização da metodologia leva a um incremento significativo nas três metaheurísticas avaliadas, inclusive encontrando resultados superiores às abordagens de estado da arte para o problema avaliado. Além disto, através das análises dos componentes de busca local, foram encontrados padrões que podem ser usados para construção de operadores de busca local em outros cenários. Já o segundo estudo é destinado à construção de uma abordagem metaheurística híbrida para a classe de problemas de sequenciamentos avaliada nesta tese. A abordagem reduz o uso de características do critério de otimização no processo de busca. Ao mesmo tempo, a abordagem não reduz a qualidade dos resultados encontrados, quando comparada a abordagens projetadas especificamente para um critério de otimização. Neste estudo é mostrado, através de quatro estudos de caso, que a abordagem proposta encontra, na maior parte dos cenários avaliados, resultados superiores ou similares às abordagens de estado da arte. Por fim, realizamos um estudo a respeito da importância e dificuldade do projeto de metaheurísticas para a classe de problemas estudados, quando guiados por uma política JIT, permitindo a inserção de tempos ociosos em conjunto com janelas de tempo. Neste estudo é proposto adaptar, para máquinas paralelas não relacionadas, um método de inserção ótima de tempos ociosos para ambientes de máquina única. Este método acarreta em incremento significativo da ordem de complexidade das abordagens. Devido a isto, é proposto um método de busca local com estruturas de vizinhança reduzidas. Os métodos propostos são integrados a quatro metaheurísticas previamente propostas na literatura e à metaheurística híbrida proposta nesta tese. É avaliado como as

metaheurísticas se comportam para resolução do problema. Os resultados mostram que a metaheurística híbrida apresenta resultados superiores às demais abordagens na maior parte dos cenários avaliados. Além disto, é avaliado qual a influência do tamanho das janelas de tempo nos resultados advindos do sequenciamento. Os resultados indicam a existência de uma correlação linear entre o tamanho da janela de tempo com os custo advindos dos atrasos e avanços das tarefas.

Palavras-chaves: Sequenciamento de tarefas, Máquinas paralelas não relacionadas, Metaheurísticas, Sequenciamento *Just-in-Time*, Métodos de busca local.

Abstract

This thesis addresses an important class of scheduling problems named unrelated parallel machines with sequence and machine-dependent setup times scheduling problems. Firstly, a literature review is performed, which raises three gaps: (i) the contribution of components used in local search has received little attention in the metaheuristics convergence validation. Consequently, there is little qualitative information to determine patterns that can enhance the construction of local search operators; (ii) the implemented metaheuristics for solving the scheduling problems incorporate many features of the own optimization criteria in the search process. As a result, these implemented metaheuristics are hard to adapt to the optimization criteria with little theoretical visibility; (iii) the literature review did not found studies for the class of scheduling problems under study guided by a Just-in-Time (JIT) policy considering time windows. This thesis has as main objective to present three contributions regarding these gaps. In the first study, a methodology for the local search operators design for metaheuristics is proposed. This methodology is evaluated in three metaheuristics previously proposed in the literature. The study shows that the use of the proposed methodology leads to a significant increase in the performance of the metaheuristics. The redesigned metaheuristics even outperform the state-of-the-art approaches for the evaluated problem. Furthermore, through the analysis of the local search components, some patterns were found that can be used to build local search operators in other scenarios. The second study is aimed at building a hybrid metaheuristic approach to the evaluated scheduling problem. This approach must reduce the use of optimization criteria features in the search process. At the same time, the approach must found results with the same quality of the state-of-the-art approaches designed for a specific optimization criterion. It is shown, using four case studies, that the proposed approach finds superior or similar results to the state-of-the-art approaches in most of the evaluated scenarios. Finally, we conducted a study about the importance and difficulty of the design of metaheuristics for the studied problems class, when guided by a Just-in-Time policy, allowing the insertion of idle times together with time windows. In this study, it is proposed to adapt, for unrelated parallel machines, a method of optimal insertion of idle times in single-machine environments. This method entails a significant increase in the order of complexity of the metaheuristics approaches. For this reason, a local search method is proposed with reduced neighborhood structures. The proposed methods are integrated into four metaheuristics previously proposed in the literature and to the hybrid metaheuristic proposed in this thesis. The metaheuristics behaviors are evaluated to the problem solution. The results show that the hybrid metaheuristic outperforms the four other approaches in most of the evaluated scenarios. Furthermore, the influence of the size

of the time windows on the scheduling results is evaluated. The results indicate the existence of a linear correlation between the size of the time window and the costs arising from jobs earliness and tardiness.

Keywords: Job scheduling, Unrelated parallel machines, Metaheuristics, Just-in-Time scheduling, Local search methods

Sumário

Lista de Tabelas xv
Lista de Figuras xvii
Lista de Algoritmos xviii

1 Introdução **2**
1.1 Apresentação Geral 2
1.2 Objetivos 6
1.2.1 Objetivos Específicos 6
1.3 Contribuições 7
1.4 Organização da Tese 8

2 Sequenciamento de Tarefas em Máquinas Paralelas **9**
2.1 Problemas de Sequenciamento de Tarefas em Máquinas Paralelas . . . 9
2.1.1 Ambientes de produção: Máquinas Paralelas 10
2.1.2 Tempos de Preparação 11
2.1.3 Critérios de Otimização 12
2.1.4 Tempos ociosos 15
2.2 Ambientes de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência e das máquinas 16
2.3 Revisão Bibliográfica para ambientes UPMSMDST 17
2.4 Instâncias 23
2.4.1 TWT 23
2.4.2 Makespan 25
2.4.3 *Total Tardiness* 25
2.4.4 *Total Weighted Completion Times* 26
2.5 Movimentos clássicos para ambientes de máquinas paralelas não relacionadas 26
2.5.1 Inserção Interna 27
2.5.2 Troca Interna 27
2.5.3 Inserção Externa 27
2.5.4 Troca Externa 28
2.6 Lacunas encontradas na literatura de sequenciamento de tarefas em ambiente UPMSMDST 28

3 Fundamentação Teórica **30**
3.1 Definição de um problema de otimização mono-objetivo 30
3.2 Otimização Combinatória 31
3.3 Heurísticas e Metaheurísticas 31

3.3.1 Estruturas de Vizinhança . . . 32
3.3.2 Heurísticas de Refinamento . . . 33
3.3.3 *Variable Neighborhood Search* - VNS . . . 33
3.3.4 *Variable Neighborhood Descent* - VND . . . 36
3.3.5 *Iterated Greedy Search* - IGS . . . 37
3.4 Sistemas Imunológicos Artificiais . . . 39
3.4.1 Sistema Imunológico Natural . . . 41
3.4.2 Sistemas Imunológicos Artificiais . . . 46
3.4.3 VNS-aiNet . . . 58

4 Projeto de métodos de busca local em ambientes de máquinas não relacionadas 68
4.1 Introdução . . . 69
4.2 Definição do Problema . . . 71
4.3 *Variable Neighborhood Descent* . . . 71
4.3.1 Estruturas de vizinhança . . . 72
4.3.2 Estratégias de exploração das estruturas de vizinhança . . . 76
4.4 Metodologia Experimental . . . 78
4.4.1 Detalhes de implementação das metaheurísticas . . . 79
4.4.2 Ambiente Computacional . . . 79
4.4.3 Metodologia para projeto dos operadores de busca local . . . 80
4.4.4 Análise de estruturas de vizinhança . . . 81
4.4.5 Comparação entre metaheurísticas . . . 83
4.5 Resultados . . . 84
4.5.1 Seleção de componentes do VND . . . 84
4.5.2 Análise de estruturas de vizinhança . . . 86
4.5.3 Comparações entre metaheurísticas . . . 88
4.6 Discussão e Conclusões . . . 95

5 Projeto de um método adaptável a diferentes critérios de otimização em ambientes de sequenciamento de tarefas em máquinas paralelas não relacionadas 97
5.1 Introdução . . . 98
5.2 Descrição do problema . . . 99
5.3 Algoritmo VNS-aiNet . . . 100
5.3.1 Codificação da solução e inicialização da população . . . 100
5.3.2 Função de avaliação célula/antígeno(*fitness*) . . . 101
5.3.3 Clonagem . . . 102
5.3.4 Mutação . . . 102
5.3.5 Busca local *Variable Neighborhood Descent* . . . 104
5.3.6 Atualização do fator de maturação e variáveis autoadaptáveis de mutação . . . 106
5.3.7 Remoção de células maturadas . . . 106
5.3.8 Supressão . . . 106
5.3.9 Inserção de novas soluções . . . 107
5.3.10 Atualização da memória imunológica . . . 108
5.4 Metodologia experimental . . . 108

5.4.1 Implementação computacional . . . 108
5.4.2 Calibração de Parâmetros . . . 108
5.4.3 Análise de Componentes . . . 110
5.4.4 Estudos de caso - VNS-aiNet perante abordagens de estado da arte . . . 111
5.5 Resultados . . . 115
5.5.1 Análise de Componentes . . . 116
5.5.2 Estudos de caso . . . 116
5.6 Discussão e conclusões . . . 129

6 Influência de janelas de tempo no projeto de metaheurísticas em ambientes de máquinas não relacionadas 132
6.1 Introdução . . . 133
6.2 Definição do problema e formulação matemática . . . 135
6.2.1 *Mixed Integer Programming* . . . 135
6.3 Métodos Heurísticos . . . 137
6.3.1 Inserção de tempos ociosos . . . 137
6.3.2 Busca Local . . . 141
6.4 Metodologia Experimental . . . 143
6.4.1 Implementação computacional . . . 143
6.4.2 Instâncias de *benchmark* . . . 144
6.4.3 Critérios dos experimentos computacionais . . . 145
6.4.4 Calibração de parâmetros . . . 146
6.4.5 Comparação entre metaheurísticas . . . 147
6.4.6 Influência do tamanho das janelas de tempo no TWET . . . 148
6.5 Resultados . . . 149
6.5.1 Análise do MIP e Metaheurísticas . . . 149
6.5.2 Influência do tamanho das janelas de tempo no TWET . . . 151
6.6 Conclusões . . . 159

7 Conclusões e considerações finais 160
7.1 Conclusões Gerais . . . 160
7.2 Publicações associadas a esta Tese . . . 164
7.2.1 Artigos publicados em periódicos . . . 164
7.2.2 Artigos em processo de avaliação em periódicos . . . 164
7.2.3 Artigos publicados em anais de congresso . . . 164
7.3 Propostas de Trabalhos Futuros . . . 165

A Elementos base para os métodos computacionais propostos 167
A.1 Representação da Solução . . . 167
A.2 Inicialização aleatória de soluções . . . 167
A.3 Funções Base . . . 167
A.4 Procedimentos de Descida . . . 168

Referências Bibliográficas 172

Lista de Tabelas

2.1 Sumário da literatura destinada a resolução de problemas de *scheduling* em ambientes UPMSMDST. 24

4.1 Lista de fatores controláveis para projeto do VND como operador de busca local. 81
4.2 Sumário da configuração do VND desenhado para cada metaheurística. 87
4.3 $\overline{RPD}$ alcançado por cada metaheurística para o conjunto de instâncias *small*. 90
4.4 $\overline{RPD}$ alcançado por cada metaheurística para o conjunto de instâncias *large*. 91
4.5 Teste pareado de Wilcoxon por grupo de instâncias comparando o IGS-VND em relação ao IGS nativo e as abordagens estado da arte. . . . 91
4.6 Teste pareado de Wilcoxon por grupo de instâncias comparando o ABC-VND em relação ao ABC nativo e as abordagens estado da arte. 93
4.7 Teste pareado de Wilcoxon por grupo de instâncias comparando o GA-VND em relação ao GA nativo e as abordagens estado da arte IHM e ILS e as metaheurísticas IGS e ABC integradas ao VND. 94
4.8 Teste de Friedman para comparação do desempenho geral das metaheurística. 95
4.9 Rank obtido pelo teste de Friedman do desempenho geral das abordagens baseada em metaheurísticas. 95

5.1 Fatores controláveis e seus respectivos níveis utilizados na calibração do VNS-aiNet. 109
5.2 Configuração de *hardware*, STP e fator de compensação utilizados na normalização do tempos computacionais das abordagens estado da arte. 113
5.3 T-test pareado entre o VNS-aiNet e variações geradas a partir de desativação de componentes do algoritmo. 118
5.4 Comparação do VNS-aiNet proposto com IHM, ACO e ILS-VND para instâncias TWT *small*. 119
5.5 Comparação do VNS-aiNet proposto com IHM, ACO e ILS-VND para instâncias TWT *large*. 119
5.6 Resultados do teste *paired Wilcoxon signed rank* por grupo de instâncias comparando o VNS-aiNet contra abordagens de estado da arte para o critério TWT. 120
5.7 Resultados do VNS-aiNet para o critério de minimização do *makepan* em instâncias de *Setup-domain- small*. 121

5.8 $\overline{RPD}$ para o critério de minimização do *makespan* para instâncias *large - Balanced* . . . 123
5.9 $\overline{RPD}$ para o critério de minimização do *makespan* para instâncias *large - Proc-domain.* . . . 124
5.10 $\overline{RPD}$ para o critério de minimização do *makespan* para instâncias *large - Setup-domain.* . . . 125
5.11 Tempo computacional médio (em segundos) para o critério de minimização do *makespan.* . . . 126
5.12 Resultados do teste *Paired Wilcoxon signed rank* por grupo de instâncias comparando o VNS-aiNet contra abordagens de estado da arte para o critério *makespan.* . . . 126
5.13 Teste de Friedman para o desempenho geral dos algoritmos para o critério de minimização do *makespan.* . . . 126
5.14 Classificação de algoritmos para o critério de minimização do *makespan* obtido através do Teste de Friedman. . . . 127
5.15 Resultado para o critério de minimização do *makespan* comparando o VNS-aiNet e abordagens determinísticas. . . . 127
5.16 Comparação para o critério de TT entre VNS-aiNet e abordagens de estado da arte IG e ABC. . . . 128
5.17 Comparação para o critério TWCT entre VNS-aiNet e abordagens de estado da arte RRT1. . . . 129

6.1 Notações para definição do MIP. . . . 137
6.2 Notações complementares a Tabela 6.1 para definição do Algoritmo 22. 138
6.3 Lista de fatores para geração de instâncias. . . . 145
6.4 Parâmetros metaheurística ABC. . . . 146
6.5 Parâmetros metaheurística GA. . . . 146
6.6 Parâmetros metaheurística GRASP. . . . 147
6.7 Parâmetro metaheurística IGS. . . . 147
6.8 Parâmetros metaheurística VNS-aiNet. . . . 147
6.9 $\overline{RPD}$ para as metaheurísticas para instância com 12 tarefas e 3 máquinas. 150
6.10 Tempo em segundos das metaheurísticas para instância com 12 tarefas e 3 máquinas. . . . 150
6.11 $\overline{RPD}$ de metaheurísticas para o conjunto de instâncias com 60 tarefas. 153
6.12 $\overline{RPD}$ de metaheurísticas para o conjunto de instâncias com 80 tarefas. 153
6.13 $\overline{RPD}$ de metaheurísticas para o conjunto de instâncias com 100 tarefas. 154
6.14 Teste pareado de Wilcoxon, comparando o VNS-aiNet contra as outras abordagens metaheurísticas. O teste é realizado para os grupos de instâncias em que o VNS-aiNet apresentou $\overline{RPD}$ de melhor qualidade que as demais abordagens. . . . 155
6.15 Teste pareado de Wilcoxon comparando o GA contra as outras abordagens metaheurísticas. O teste é realizado para os grupos de instâncias em que o GA apresentou $\overline{RPD}$ de melhor qualidade que as demais abordagens. . . . 156

6.16 Teste pareado de Wilcoxon comparando o GRASP contra as outras abordagens metaheurísticas. O teste é realizado para os grupos de instâncias em que o GRASP apresentou $\overline{RPD}$ de melhor qualidade que as demais abordagens. 156
6.17 Teste pareado de Wilcoxon comparando o IGS contra as outras abordagens metaheurísticas. O teste é realizado para os grupos de instâncias em que o IGS apresentou $\overline{RPD}$ de melhor qualidade que as demais abordagens. 156
6.18 Teste de Wilcoxon pareado por grupo ($Algorithm \times \tau \times R$), comparando as instâncias com janelas de tempo originais e reescaladas pelo fator descrito na coluna. 158

Lista de Figuras

2.1 Tempos de preparação dependentes da sequência em máquinas paralelas não relacionadas. 12
2.2 Exemplo na estrutura de vizinhança de Inserção Interna. 27
2.3 Exemplo na estrutura de vizinhança de Troca Interna. 27
2.4 Exemplo na estrutura de vizinhança de Inserção Externa. 27
2.5 Exemplo na estrutura de vizinhança de Troca Externa. 28

3.1 Ilustração do princípio da seleção clonal. Fonte: de Castro (2001). . . 44
3.2 Ilustração dos mecanismos de identificação de *linfócito B* por *linfócito B* através dos *idiotopos*. Fonte: de Castro (2001) 45
3.3 Comportamento do operador de hipermutação somática. Fonte: de Castro (2001). 50
3.4 População de soluções encontradas ao final da execução do opt-aiNet. Fonte: Coelho (2011) 54
3.5 Evolução do número de células em relação ao número de iterações (gerações) do opt-aiNet. Fonte: Coelho (2011) 55
3.6 Exemplo do comportamento do operador de mutação com função de *fitness* proposta. 62

4.1 Exemplo de um movimento na estrutura de vizinhança Iterated Greedy Search. 75
4.2 S/N_{ratio} para os fatores controláveis do VND com operador de busca local do IGS. 85
4.3 S/N_{ratio} para os fatores controláveis do VND com operador de busca local do ABC. 86
4.4 S/N_{ratio} para os fatores controláveis do VND com operador de busca local do GA. 86
4.5 Análise de *ablation* do comportamento das estruturas de vizinhança para cada metaheurística. Cada ponto indica o desempenho da estrutura de vizinhança em cada passo. A linha ligando as estruturas de vizinhança indica os componentes que acarretaram em maior contribuição na convergência da metaheurística em cada passo. *A calibração pela abordagem de Taguchi indicou que as estruturas de vizinhança NE.IS e NE.SIGS deterioraram o desempenho do VND integrado á abordagem ABC; assim, a influência destas duas estruturas não é analisada para a metaheurística ABC. 88

4.6 Gráficos de convergência para cada metaheurística. Cada configuração do VND considera a remoção de uma estrutura de vizinhança do VND afim de verificar a sua influência na convergência da metaheurística. *A calibração pela abordagem de Taguchi indicou que as estruturas de vizinhaça NE.IS e NE.SIGS deterioraram o desempenho do VND integrado à abordagem ABC; assim, a influência destas duas estruturas não é analisada para a metaheurística ABC. 89
4.7 Boxplot comparando o RPD alcançado pelas mataheurísticas para cada um dos 16 cenários avaliados ($\eta \times \tau \times R \times r_\tau$) para os conjunto de instâncias *large*. 92

5.1 Representação de um *linfócito B* para problemas em ambientes UPMSMDST. 101
5.2 Exemplo de mutação do clone cl_p com $n_{mut} = 2$. Dois movimentos são realizados: o primeiro é um movimento de Troca; o segundo, é de Inserção. 104
5.3 Exemplo da evolução da variável de mutação alto-adaptável. 107
5.4 Gráfico de efeitos principais do indicador S/N ratio para o algoritmo VNS-aiNET. 110
5.5 Média acumulada das variações nos componentes do VNS-aiNet. . . 117
5.6 Comparação de *boxplot* para erro do $\overline{RPD}$ médio para o critério do *makespan*. 122

6.1 Convergência por NFE e *boxplot* do *fitness* médio normalizado, por combinação $n \times m$ do conjunto de instâncias *large*. 152
6.2 Regressão linear considerando o quanto variações no tamanho da janela de tempo influenciam os custos de atraso e avanço no sequenciamento, sendo $tw_{ratio} = 1.0$ o tamanho original, $tw_{ratio} = 0.0$ a inexistência da janela de tempo e $tw_{ratio} = 1.8$ considera 1.8 vezes o tamanho original da janela de tempo. 157

Lista de Algoritmos

1 *Best Improvement* (BI) . . . 34
2 *First Improvement* (FI) . . . 34
3 *Variable Neighborhood Search* (VNS) . . . 35
4 *Variable Neighborhood Descent* (VND) . . . 36
5 *Construção Gulosa* . . . 38
6 *Fase Destrutiva IGS* . . . 39
7 *Iterated Greedy Search* (IGS) . . . 40
8 Pseudocódigo do algoritmo CLONALG . . . 49
9 Pseudocódigo do algoritmo opt-aiNet . . . 52
10 Pseudocódigo do algoritmo copt-aiNet . . . 56
11 *Framework* para aplicação da abordagem *Ageing* . . . 57
12 Variable Neighborhood Search Imunne Network(VNS-aiNet) . . . 59
13 Modelo de atualização do fator de maturação das células. . . . 64
14 Modelo de remoção de células totalmente maturadas. . . . 64
15 Modelo de supressão da população. . . . 65
16 Estrutura de vizinhança NE.II. . . . 73
17 Estrutura de vizinhança NE.II. . . . 73
18 Estrutura de vizinhança inserção externa - NE.EI. . . . 74
19 Estrutura de vizinhança NE.IGS. . . . 75
20 Estrutura de vizinhança NE.SIGS. . . . 76
21 Função de mutação para problemas em ambientes UPMSMDST . . . 104
22 Inserção de tempos ociosos . . . 140
23 Algoritmo que descreve a estrutura de vizinhança NE.I. . . . 142
24 Descida Inserção. . . . 168
25 Descida *Best Improvement* Inserção. . . . 169
26 Descida *First Improvement* Inserção. . . . 169
27 Descida Troca. . . . 170
28 Descida *Best Improvement* Troca. . . . 170
29 Descida *First Improvement* Troca. . . . 171

Lista de Siglas e Abreviaturas

ABC *Artificial Bee Colony.*

ACO *Ant Colony Optimization.*

aiNet *Artificial Immune Network.*

ALNS *Adaptive Large Neighborhood Search.*

ASPT *Adaptive Shortest Processing Time.*

ATCS *Apparent Tardiness Cost with separable Setups.*

ATCSR_Rm *Apparent Tardiness Cost with separable Setup and Ready times for unrelated parallel machines.*

B&B *Branch and Bound.*

B&C *Branch and Check.*

BI *Shortest Weighted Processing Time.*

CLONALG *CLONal selection ALGorithm.*

CSA *Clonal Selection Algorithm.*

EDA *Estimation of Distribution Algorithm.*

EDD *Earliest Due Date.*

EMA *ElectroMagnetism Algorithm.*

EWDD *Earliest Weighted Due Date.*

FI *First Improvement.*

GA *Genetic Algorithm.*

GALA *Genetic Algorithm with Local search Algorithm.*

GRASP *Greedy Randomized Adaptive Search Procedure.*

HABC *Hybrid Artificial Bee Colony.*

IGS *Iterated Greedy Search.*

IHM *Iterated Hybrid Metaheuristic.*

ILS *Iterated Local Search.*

JIT *Just-in-Time.*

LAHC *Late Acceptance Hill-Climbing.*

MIP *Mixed-Integer Programming.*

MPA *Mathematical Programming-based Algorithm.*

opt-aiNet *Artificial Immune Network for Optimization.*

PMSDSP *Parallel Machines Sequence Dependent Scheduling Problem.*

PMSP *Parallel Machine Scheduling Problem.*

PR *Path relinking.*

RRT *Record-to-Record Travel algorithm.*

SA *Simulated Annealing.*

SCHC *Step Counting Hill-Climbing.*

SIAs *Sistemas Imunológicos Artificiais.*

SIN *Sistema Imunológico Natural.*

SRT *Shortest Ready Times.*

SWPT *Shortest Weighted Processing Time.*

TS *Tabu Search.*

TT *Total Tardiness.*

TWCT *Total Weighted Completion Time.*

TWT *Total Weighted Tardiness.*

UPMS *Unrelated Parallel Machine Scheduling.*

UPMSMDST *Unrelated Parallel Machine with Sequence and Machine Dependent Setup Times.*

VND *Variable Neighbourhood Descent.*

VNS *Variable Neighbourhood Search.*

WO *Worm Optimization.*

Capítulo 1

Introdução

1.1 Apresentação Geral

A palavra *scheduling*, segundo o *Cambridge Dictionary*[1] significa a atividade de planejar as datas em que determinadas tarefas serão realizadas ou eventos irão ocorrer. Com base nesta definição, pode-se perceber que o termo *scheduling* está diretamente relacionado ao processo de tomada de decisão referente à definição/atribuição de datas (intervalos de tempo) que um conjunto de "tarefas" irá ocupar na dimensão temporal. Na pesquisa operacional, o termo *scheduling* normalmente é associado a problemas destinados a sequenciar/atribuir, na dimensão temporal, um conjunto de "tarefas", de modo a otimizar um determinado objetivo estratégico. Assim, a definição do que é uma "tarefa" depende diretamente das peculiaridades do problema de *scheduling* estudado. Por exemplo, no problema de geração de energia elétrica conhecido como *Unit Commitment Problem*, o *scheduling* consiste na definição, por unidade de tempo, quais turbinas elétricas devem estar ativas de forma que a demanda desejada de eletricidade seja satisfeita (Sen e Kothari, 1998). Já em um problema de planejamento de colheita agrícola em propriedades fragmentadas, uma tarefa consiste em realizar a colheita em um determinado fragmento da propriedade (He et al., 2018).

Apesar da característica comum do planejamento de alocação temporal de tarefas, deve-se ter em mente que o ambiente no qual estas tarefas serão alocadas/sequenciadas tem uma influência direta na elaboração do planejamento. Devido a isto, existe uma grande quantidade de classes de problemas de *scheduling*, cada classe apresentando ambientes com especificidades próprias. Somando estas características à complexidade computacional, em sua maioria de natureza NP-Difícil (Allahverdi et al., 1999, 2008; Allahverdi, 2015), torna-se inviável a construção de modelos e métodos que sejam generalizados para diferentes classes de problema.

Problemas de *scheduling* em ambientes de máquinas paralelas (PMSP) constituem uma importante classe de problemas, bastante estudados na literatura, por serem muito presentes em ambientes produtivos reais, tais como: indústria têxtil (Liao et al., 2016); manufatura de *displays* de cristal líquido (Chen, 2009; Ekici et al., 2019); indústria cerâmica (Ruiz e Andrés-Romano, 2011); aplicação de pesticidas (Afzalirad e Rezaeian, 2016); planejamento de colheita em propriedades agrícolas fragmentadas (He et al., 2018); indústria de alimentos perecíveis (Shirvani et al., 2014); dentre

[1] https://dictionary.cambridge.org/pt/dicionario/ingles/scheduling acessado em 08/06/2021

outros. Em geral, estes ambientes possuem um conjunto de máquinas paralelas que podem ser classificadas em 3 tipos: (i) Máquinas idênticas; (ii) Máquinas uniformes; (iii) Máquinas não relacionadas. Problemas de sequenciamento em máquinas paralelas não relacionadas (UPMS) vêm ganhando muito destaque na literatura nas duas últimas décadas, já que este tipo de configuração é muito presente em manufaturas reais (Chen, 2009; Ruiz e Andrés-Romano, 2011; Liao et al., 2016; Ekici et al., 2019). A principal particularidade de ambientes com máquinas paralelas não relacionadas consiste na variação do tempo de processamento de uma tarefa de acordo com a máquina à qual a tarefa é atribuída. O tempo de preparação (ou configuração) para o processamento de uma tarefa é outra característica muito comum relacionada a ambientes de máquinas paralelas não relacionadas (Al-Salem (2004); Helal e Al-Salem (2006); Armentano e Filho (2007); Diana et al. (2013); Arnaout et al. (2014); Nogueira et al. (2014); Diana et al. (2018)). Características dos ambientes UPMS, como máquinas de tecnologias diferentes para um mesmo fim (Ekici et al., 2019), e/ou máquinas que processam diferentes materiais (Ruiz e Andrés-Romano, 2011) requerem, em geral, que antes do processamento de uma tarefa, seja realizada a reconfiguração da máquina. Este processo é conhecido como tempo de preparação, sendo muito relevante em problemas de sequenciamento. Nas contínuas revisões bibliográficas encontradas na literatura (Allahverdi et al., 1999, 2008; Allahverdi, 2015) percebe-se claramente, cada vez mais, a incorporação dos tempos de preparação nos modelos e métodos estudados para diferentes classes de problemas de sequenciamento. Ambientes UPMS com tempos de preparação dependentes da sequencia e da máquina são conhecidos na literatura como UPMSMDST (*Unrelated Parallel Machine with Sequence and Machine Dependent Setup Times*)

Devido à relevância em aplicações reais, critérios de otimização relacionadas a datas de entrega são comumente encontrados em problemas de *scheduling* em ambientes UPMSMDST (e.g.Chen (2009); Lin e Hsieh (2014); Nogueira et al. (2014); Diana et al. (2018)). Os critérios de otimização com datas de entrega mais estudados neste ambiente consideraram exclusivamente o atraso de tarefas. Apesar da relevância destes objetivos, existem cenários nos quais a adoção apenas do atraso torna, do ponto de vista prático, soluções advindas do sequenciamento ineficientes ou mesmo infactíveis (e.g. produção de alimentos perecíveis (Shirvani et al., 2014)).

O planejamento de produção em ambientes nos quais os insumos e materiais são perecíveis e/ou os custos de armazenamento destes são muito elevados sempre devem considerar, no processo de *scheduling*, os custos, a disponibilidade e a efetividade do processo de estocagem. Por esta razão, idealmente, os insumos/materiais destinados ao processamento de uma tarefa devem ingressar no ambiente produtivo o mais próximo possível do instante inicial de processamento da tarefa. Ao mesmo tempo, o produto resultante deve ser despachado assim que produzido. Deste modo, a definição do *scheduling* nestes ambientes deve ser guiada por uma política conhecida na literatura como *Just-In-Time* (JIT). Segundo Cheng e Podolsky (1996), JIT é uma política de gerenciamento, desenvolvido inicialmente pela Toyota Motor Company e aplicada em manufaturas, a qual envolve ter os itens corretos, na quantidade e qualidade necessárias, no ambiente e tempo requeridos para realização da atividade planejada.

Apesar de grande relevância prática da política JIT na definição de *scheduling* em ambientes produtivos, a literatura que trabalha esta temática em ambientes

UPMSMDST é muito pouco explorada (i.e. Vallada e Ruiz (2012); Nogueira et al. (2014); Kramer e Subramanian (2019)) e, adicionalmente, não há trabalhos que abordem considerações de janelas de tempo, segundo a revisão bibliográfica realizada. Entregas definidas por meio de janelas de tempo são uma importante ferramenta em ambientes JIT. Em ambiente reais, a rigidez de datas de entrega dificilmente leva a situações factíveis, seja pelo constante atraso de insumos por fornecedores, ou pela indisponibilidade do recebimento de produtos pelos clientes. Assim, as cadeias produtivas sempre devem manter uma capacidade mínima de estocagem para tornar flexível o processo produtivo e, consequentemente, o cumprimento de contratos.

Estudos para ambientes de *scheduling* guiados pela política JIT com (ou sem) janelas de tempo (e.g. Hendel e Sourd (2007); Nogueira et al. (2014); Polyakovskiy e M'Hallah (2014); Rosa et al. (2017)), mostram que abordagens computacionais desenvolvidas para estes ambientes apresentam um incremento considerável da complexidade computacional, quando comparado a abordagens que consideram apenas a otimização do atraso. Isto se dá, principalmente, pela necessidade de definição de tempo ociosos entre a execução de duas tarefas, já que, na maior parte das abordagens, esta definição passa pela solução de um subproblema de otimização (Hendel e Sourd, 2007; Nogueira et al., 2014; Rosa et al., 2017). Esta característica é bastante relevante no projeto de abordagens para problemas de *scheduling* e se torna ainda mais relevante em ambientes UPMSMDST.

Devido à complexidade do espaço de busca encontrado em problemas de *scheduling* em ambientes UPMSMDST, métodos/modelos computacionais determinísticos exatos apresentavam-se viáveis apenas para problemas de baixa dimensão (e.g. Lin e Hsieh (2014); Nogueira et al. (2014)). Recentemente, algumas abordagens têm se mostrado efetivas para problemas de grande dimensão, mas ainda despendem elevado custo computacional (e.g. Tran et al. (2016); Fanjul-Peyro et al. (2019)). Por outro lado, métodos estocásticos se mostram muito eficientes tanto do ponto de vista computacional, quanto da qualidade das soluções. Para aumentar a eficácia das abordagens estocásticas, grande parte destas têm adicionado especificidades do problema no processo de busca. A incorporação de características do critério de otimização no processo de busca é uma característica muito utilizada. Esta prática visa reduzir estruturas de vizinhança, construir soluções iniciais mais efetivas e encontrar atalhos no espaço de busca. Apesar de ser altamente eficaz, este tipo de abordagem deixa os métodos de solução extremamente dependentes do objetivo estudado. Isto deixa um vácuo na literatura para problemas com critérios de otimização de pouca visibilidade teórica, mas que podem aparecer em ambientes reais de processos produtivos. Para reduzir este vácuo, algumas iniciativas baseadas em inteligência computacional vêm sendo estudadas para contornar este problema (e.g. Đurasević et al. (2016)).

Técnicas baseadas em inteligência computacional tendem a ser bastante utilizadas em problemas que realizam a pesquisa em espaços de busca grandes e complexos, nos quais, usualmente, métodos exatos não se mostram muito efetivos. As metaheurísticas são os métodos de inteligência computacional mais utilizados para o projeto de soluções de problemas de otimização desta natureza. Nas últimas duas décadas, observa-se um crescimento expressivo na proposta de novos métodos metaheurísticos para a solução de problemas de otimização. Devido a isto, consolidaram-se alguns paradigmas no projeto de métodos metaheurísticos (Martí et al., 2018). Especificamente, para a resolução de problemas de natureza combinatória, classe que engloba

problemas de *scheduling*, duas classes de metaheurísticas têm sido muito utilizadas e alcançado alta eficácia, sendo estas: (i) métodos baseados no paradigma de busca guiada por estruturas de vizinhanças; (ii) métodos guiados por população de soluções. Abordagens guiadas por estruturas de vizinhanças têm o maior enfoque na definição de elementos de exploração local, sendo definidas estruturas de busca local e realizada a intensificação nestas estruturas até se atingir um mínimo local. Após a estagnação, estas metaheurísticas acionam mecanismos de diversificação para gerar "saltos" no espaço de busca e levar as soluções a novas bacias de atração. Métodos populacionais, por sua vez, baseiam-se em processos de busca pela iteração entre múltiplas soluções presentes no espaço de busca. Assim, metaheurísticas populacionais tendem a apresentar estratégias bastante eficientes de otimização global, mas podem ser deficientes na exploração local (Weise et al., 2016), criando, assim, a necessidade de balancear as características de diversificação e intensificação (Lala et al., 2015).

Os Sistemas Imunológicos Artificiais (SIAs) são uma classe importante de metaheurísticas populacionais desenvolvidas a partir de teorias que tentam simular o comportamento do Sistema Imunológico Natural (SIN). Devido a grande quantidade de subsistemas que compõem o SIN, existe uma grande quantidade de (meta)heurísticas baseadas neste, destinadas à resolução de vários tipos de problemas, como aprendizagem de máquina, detecção de anomalias, mineração de dados, dentre outros. Para a resolução de problemas de otimização, grande parte dos SIAs são baseados no comportamento do sistema imunológico adaptativo, em sua maioria são derivados do Principio da Seleção Clonal proposto por Burnet (1957). Dentro deste grupo se destacam o CLONALG (de Castro e Von Zuben, 2000), *B-Cell Algorithm* (Kelsey e Timmis, 2003) e a família de algoritmos baseadas no *Artificial Immune Network for Optimization* (opt-aiNet) (de Castro e Timmis, 2002b).

Algoritmos da família aiNet integram características advindas do Principio da Seleção Clonal com o comportamento da Teoria da Rede Imunológica descrita por Jerne (1974). Metaheurísticas baseadas na Teoria da Rede Imunológica são amplamente reconhecidas na literatura para a resolução de problemas de otimização multimodais. Nestes, um mesmo problema pode apresentar múltiplos mínimos (ou máximos). Assim, o algoritmo utilizado para a resolução do problema explora simultaneamente múltiplas soluções que se encontram em bacias de atração distintas. Esta é uma característica que acarreta em um alto poder de dispersão das soluções no espaço de busca. Em contrapartida, se usada de maneira indiscriminada, pode acarretar em elevado custo computacional, devido ao incremento dinâmico da população permitido por estas abordagens (Diana et al., 2017). Além disto, apesar destes algoritmos terem alto poder de otimização global, possuem estratégias muito limitadas de exploração local (Diana et al., 2017). Em ambientes complexos, para atingir desempenho elevado destas abordagens,é necessário a incorporação de estratégias de busca local altamente efetivas e eficazes (e.g. de Sousa et al. (2004); Coelho et al. (2011); Diana et al. (2015)).

Estudos recentes indicam que a combinação de características de metaheurísticas populacionais e métodos de busca guiados por estruturas de vizinhança vem apresentando resultados mais efetivos que a aplicação dos mesmos métodos em sua forma nativa (Blum et al., 2011). Entretanto, alguns cuidados devem ser considerados na construção destes modelos híbridos para garantir o equilíbrio entre diversificação e intensificação (Lala et al., 2015). Deve-se considerar que métodos de busca local

podem guiar prematuramente toda a população para uma bacia de atração forte, fazendo, assim, com que o método convirja prematuramente. Além disto, algoritmos de busca local tendem a ter um custo computacional elevado e, quando combinados a metaheurísticas populacionais, podem comprometer toda a estratégia de busca, devido ao custo computacional inviável. Portanto, estruturas de vizinhança de busca local devem ser projetadas de forma efetiva/eficiente, considerando características do espaço de busca. Assim, a definição destas estruturas deve ser guiada por estudos que mostrem a verdadeira efetividade das mesmas, tornando o projeto destas abordagens mais próximo à ciência e menos próximo à "arte".

1.2 Objetivos

Considerando o contexto exposto, o objetivo geral desta tese é contribuir para tornar mais eficaz o projeto de abordagens de inteligência computacional aplicadas a problemas de *scheduling* em ambientes UPMSMDST. Para isto, busca-se atingir três objetivos principais:

(i) propor uma metodologia para verificar a eficácia de elementos computacionais propostos, utilizados na construção de algoritmos de busca local integrados à metaheurísticas;

(ii) propor uma abordagem de inteligência computacional baseada na hibridização de um algoritmo da família aiNet com métodos de busca local, que seja adaptável a diferentes critérios de otimização para encontrar soluções eficientes e eficazes, sem incorporar características relevantes do critério de otimização usado;

(iii) por fim, verificar como a incorporação de janelas de tempos influi no projeto da abordagens metaheurísticas para o sequenciamento de tarefas em ambientes UPMSMDST guiados por uma política JIT.

1.2.1 Objetivos Específicos

Para que estes objetivos gerais sejam alcançados, é necessário atingir os seguintes objetivos específicos:

(i) Fazer um levantamento da literatura sobre os principais temas envolvidos nesta tese, os quais são: sequenciamento de tarefas em ambientes UPMSMDST, utilização de janelas de tempo no sequenciamento guiados pela política JIT, métodos de busca local e estruturas de vizinhança, hibridização de metaheurísticas e algoritmos baseados em SIA;

(ii) Projetar uma metodologia para avaliação da efetividade e eficácia dos componentes de métodos de busca local no processo de busca;

(iii) Projetar estruturas de vizinhança baseadas em movimentos comumente encontrados na literatura de ambientes UPMSMDST, que sejam efetivas, eficazes e extensíveis a diferentes critérios de otimização;

(iv) Utilizar a metodologia e estruturas de vizinhança propostas para projetar operadores de busca local que sejam incorporados a metaheurísticas previamente propostas na literatura, afim de melhorar o desempenhos destas abordagens metaheurísticas;

(v) Projetar a hibridização de um algoritmo, baseado na família de algoritmos aiNet, que seja extensível a diferentes critérios de otimização em ambiente UPMSMDST, projetando componentes que tornem a abordagem comparável a abordagens de estado da arte para critérios de otimização específicos;

(vi) Verificar se os componentes propostos realmente contribuem para a adaptação da abordagem híbrida a diferentes critérios de otimização;

(vii) Avaliar, para os quatro critérios de otimização mais estudados em ambientes UPMSMDST, se os resultados encontrados pela metaheurística híbrida proposta são compatíveis com os resultados das abordagens de estado da arte para cada critério de otimização específico;

(viii) Avaliar quais as necessidades adicionais para construir abordagens metaheurísticas para o sequenciamento de tarefas em ambientes UPMSMDST guiados por uma política JIT com janelas de tempo;

(ix) Adaptar um algoritmo de inserção ótima de tempos ociosos para o sequenciamento de tarefas em ambientes de máquina única para ambientes de máquinas paralelas não relacionas;

(x) Adaptar os operadores de metaheurísticas propostas previamente na literatura para diferentes critérios de otimização para a resolução do problemas de sequenciamento de tarefas em ambientes UPMSMDST guiados por uma política JIT com janelas de tempo;

(xi) Propor um conjunto de instâncias para o problema com janelas de tempo e avaliar o comportamento das metaheurísticas adaptadas ao problema, segundo características das instâncias;

(xii) Avaliar a influência do tamanho das janelas de tempo na resposta das abordagens metaheurísticas adaptadas.

1.3 Contribuições

Este trabalho tem como principal contribuição o projeto de métodos baseados em inteligência computacional para resolução de lacunas existentes na literatura de problemas de *scheduling* em ambientes de máquinas paralelas não relacionadas. O trabalho apresenta as seguintes contribuições específicas:

(i) Apresenta, para problemas de *scheduling* em ambiente UPMSMDST, uma metodologia para o projeto de métodos de busca local baseados em estruturas de vizinhança;

(ii) Realiza um estudo que detalha a contribuição de seis estruturas de vizinhança no processo de busca de três metaheurísticas, indicando padrões que podem se repetir no projeto de operadores de busca local em ambiente UPMSMDST;

(iii) Propõe um algoritmo para a resolução de problemas de *scheduling* em ambiente UPMSMDST que minimiza a utilização do critério de otimização no processo de busca, que consegue ser, em grande parte dos cenários avaliados, mais eficiente que abordagens estado da arte para critérios específicos de otimização;

(iv) Realiza o primeiro estudo para problemas de *scheduling* em ambiente UPMSMDST guiados por uma política JIT com presença de janelas de tempo;

(v) Adapta um método de inserção de tempos ociosos para *scheduling* em máquina simples com janelas de tempo para problemas em ambientes UPMSMDST;

(vi) Avalia como se comportam, em ambientes UPMSMDST com janelas de tempo, quatro metaheurísticas previamente propostas na literatura de ambientes UPMSMDST e a metaheurística proposta neste trabalho, que se adapta a diferentes critérios de otimização. Identificamos em quais cenários cada uma melhor se adéqua;

(vii) Identifica qual o tipo de correlação existente entre o tamanho das janelas de tempo e os custos relativos a atrasos e avanços em problemas de *scheduling* em ambiente UPMSMDST guiados por uma política JIT.

1.4 Organização da Tese

O restante desta tese é dividida nos seguintes capítulos: o Capítulo 2 apresenta as definições para o estudo de problemas de sequenciamento de tarefas (*scheduling*) em ambientes UPMSMDST, além de introduzir uma revisão bibliográfica, apresentando três lacunas importantes, exploradas nesta tese, na literatura de ambientes UPMSMDST. O Capítulo 3 apresenta os fundamentos computacionais necessários para os estudos que são propostos nesta tese. Já o Capítulo 4 apresenta um estudo detalhado para criação e avaliação de métodos de busca local em ambientes UPMSMDST. O Capítulo 5 introduz a proposta de uma metaheurística híbrida que tem como principal objetivo reduzir a utilização de características do critério de otimização no processo de busca, mas, ao mesmo tempo, encontrar resultados "robustos"[2]. O Capítulo 6 apresenta, por sua vez, um estudo a respeito da importância e das dificuldades de inserção de janelas de tempo, em conjunto com tempo ociosos, na construção de métodos computacionais para problemas de sequenciamento de tarefas em ambientes UPMSMDST. Por fim, o Capítulo 7 apresenta as conclusões obtidas nesta tese, sendo realizada a discussão referente às lacunas presentes em ambientes UPMSMDST que a tese atacou. Já ao final do capítulo são apresentados os trabalhos, resultantes destas tese, publicados (e em processo de avaliação) em periódicos e conferências, além de uma série de pontos não cobertos pela tese e direcionamentos para trabalhos futuros.

[2] O termo "*robustos*" é usado aqui para denominar uma abordagem cujos resultados devem apresentar alta qualidade, independentemente dos critérios de otimização usados.

Capítulo 2

Sequenciamento de Tarefas em Máquinas Paralelas

Este capítulo apresenta a fundamentação teórica e a revisão bibliográfica de sequenciamento de tarefas em máquinas paralelas, necessárias para esta tese. Primeiramente, na Seção 2.1 é apresentado o ambiente de máquinas paralelas, sendo, na Subseção 2.1.1, mostradas as diferentes características de máquinas que podem compor este ambiente. Na Subseção 2.1.2 discute-se como a inserção de tempos de preparação, antes do processamento de tarefas, impactam no sequenciamento em ambientes compostos por máquinas paralelas não relacionadas. Em seguida, na Subseção 2.1.3, são apresentados os principais critérios de otimização utilizados para problemas de sequenciamento de tarefas. Na Seção 2.2 é apresentada uma definição global do problema de sequenciamento em máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência estudado nesta tese. A Seção 2.3 introduz uma extensa revisão bibliográfica sobre as abordagens computacionais propostas para ambientes de sequenciamento similares ao estudado nesta tese. Por fim, nas seções 2.4 e 2.5, são apresentados, respectivamente, os conjuntos de instâncias de *benchmark* disponíveis na literatura e os movimentos clássicos utilizados para o projeto de estruturas de vizinhança.

2.1 Problemas de Sequenciamento de Tarefas em Máquinas Paralelas

O sequenciamento de tarefas (problemas de *scheduling*) em máquinas paralelas é um dos problemas de otimização combinatória mais estudados na literatura. De maneira generalizada, este problema pode ser definido como: dado um conjunto $N = \{1, \ldots, n\}$ de tarefas e um determinado ambiente composto por um conjunto $M = \{1, \ldots, m\}$ de máquinas, um problema de sequenciamento consiste em organizar e distribuir as tarefas de modo a sequenciá-las no conjunto de máquinas, com o intuito de minimizar ou maximizar um determinado critério de otimização. Como pode ser observado, este é um problema combinatório, sendo demonstrado, por French (1982), que problemas de sequenciamento de tarefas se caracterizam como problemas da classe NP-Difícil, mesmo em um ambiente de única máquina.

O sequenciamento de tarefas possui aplicações nas mais diversas áreas, como ci-

ência da computação, engenharia de produção, engenharia civil, gestão de hospitais, dentre outras. Na ciência da computação, diversos problemas de recuperação da informação, engenharia de software e arquitetura de computadores podem ser modelados como problemas de sequenciamento. O mesmo ocorre na engenharia civil com, por exemplo, o planejamento de recursos para execução de uma obra. Em hospitais que apresentem grande demanda por cirurgias e recursos (salas, equipamentos, cirurgiões) escassos, um planejamento adequado visando a melhor utilização dos recursos promove o aumento do número de cirurgias realizadas ao longo do tempo.

A engenharia de produção é a área de maior aplicação para problemas de sequenciamento, tendo em vista o impacto econômico gerado pelo sequenciamento da produção industrial. Em uma indústria, o planejamento de longo prazo tem características estratégicas, sendo realizado pela alta administração. No planejamento de médio prazo, técnicas de Pesquisa Operacional começam a ser adotadas, visando atender aos prazos acordados com os clientes ou definidos pelo plano estratégico da empresa. Já no âmbito operacional, o uso de técnicas de otimização da produção se torna mais frequente, tanto para minimizar os custos operacionais, quanto para maximizar a utilização dos recursos.

Graham et al. (1979), em seu trabalho pioneiro sobre sequenciamento de tarefas, introduziu a notação com 3 campos, na forma $\alpha/\beta/\gamma$, para descrever, de modo sucinto, as principais características particulares de um dado problema de sequenciamento de tarefas. O campo α descreve as características do ambiente das máquinas. O campo β descreve as características do processamento das tarefas, enquanto que o campo γ especifica a medida de desempenho (critério de otimização) segundo a qual as soluções serão avaliadas. Como exemplo, a notação $Rm/s_{ijk}/C_{max}$ é utilizada para representar um problema de planejamento de produção em máquinas paralelas não relacionadas ($\alpha = Rm$), em que a transição entre duas tarefas j e k, adjacentes na sequência de tarefas atribuídas a uma máquina i, exige um tempo de preparação s_{ijk}, que depende da sequência e da máquina considerada ($\beta = s_{ijk}$) e cujo objetivo é a minimização do máximo instante de conclusão das tarefas ($\gamma = C_{max}$).

Nas próximas subseções serão apresentadas as características de problemas clássicos de planejamento de produção em máquinas paralelas e suas representações, segundo a notação introduzida por Graham et al. (1979).

2.1.1 Ambientes de produção: Máquinas Paralelas

Em ambientes de produção em máquinas paralelas, cada tarefa envolve a execução de uma única operação a ser realizada em uma única máquina, dentre as disponíveis. Ambientes de máquinas paralelas são muito estudados na literatura, devido a ocorrência em ambientes reais de manufatura, como a indústria têxtil (Liao et al., 2016); a produção de monitores de cristal líquido (*LCD display*) (Chen (2009), Eom et al. (2002)); a indústria cerâmica (Ruiz e Andrés-Romano, 2011); a fabricação de tubos de PVC (Lee et al., 2014), dentre outros ambientes industriais. Segundo Pinedo (2008), as máquinas paralelas normalmente são classificadas em três tipos: (i) idênticas; (ii) uniformes; e (iii) não-relacionadas. Em máquinas idênticas, segundo a notação de Graham et al. (1979), $\alpha = Pm$ e o tempo de processamento de uma tarefa k, representado por p_k, é o mesmo em qualquer máquina. Nas máquinas ditas uniformes, $\alpha = Qm$, os tempos de processamento de qualquer tarefa variam de acordo com a

velocidade u_i da máquina i. Assim, o tempo de processamento da tarefa k na máquina i é dado por $p_{ik} = p_k/u_i$, em que p_k representa um tempo de processamento de referência da tarefa k. Quanto maior a velocidade de uma máquina, menores os tempos para processamento de cada tarefa que a ela for atribuída. Por fim, quando as máquinas são ditas não-relacionadas ($\alpha = Rm$), os tempos de processamento p_{ik} variam entre máquinas, sem qualquer padrão bem definido, diferente do que se observa no caso das máquinas uniformes. Segundo French (1982), estes três ambientes são capazes de descrever a maior parte dos problemas de planejamento de produção em máquinas paralelas.

A elegibilidade de máquinas é outra questão a ser observada, podendo estar presente nos mais diversos ambientes de planejamento de produção. Isto implica em dizer que nem todas as tarefas podem ser processadas por todas as máquinas. Esta situação é representada por $\beta = M_{ik}$.

Também é possível que máquinas e/ou tarefas não estejam disponíveis no instante inicial do horizonte de planejamento. A situação em que o instante inicial de disponibilidade (*ready time*) de uma (ou mais) máquina(s) não é igual ao inicio do planejamento é representada por $\beta = r_i$. Já quando o mesmo ocorre com uma (ou mais) tarefa(s) k, é definido que o problema apresenta *job ready time*, sendo representado por $\beta = r_k$.

2.1.2 Tempos de Preparação

Tempo de *setup* (ou preparação) é uma das características mais presentes e importantes em ambientes de máquinas paralelas não relacionadas. Tempo de preparação é o processo de configuração da máquina para executar uma tarefa. O tempo despendido inclui operações como disponibilização, posicionamento e inspeção de materiais, além de limpeza e configurações das máquinas, necessárias antes que o processamento da tarefa possa ser iniciado (Cao e Bedworth, 1992).

Levantamento bibliográfico realizado por Allahverdi et al. (1999) sobre trabalhos de sequenciamento de tarefas revela que, até o inicio dos anos 2000, poucos trabalhos consideravam os tempos de preparação. Já a revisão realizada por Allahverdi et al. (2008) mostra uma mudança de cenário, na qual grande parte dos trabalhos passaram a considerar os tempos de preparação no sequenciamento. A simplificação do modelo com a não-utilização dos tempos de preparação tende a acarretar em significativas distorções nas soluções, principalmente em cenários em que os tempos de preparação dependem não apenas da tarefa k a ser processada, como também da tarefa j concluída imediatamente anterior a estas na sequência da máquina i. Ruiz e Andrés-Romano (2011) apresenta um processo de indústria de cerâmica em que, entre o processamento de lotes de peças cerâmicas (tarefas), é necessário realizar a limpeza da linha de produção. Em linhas de produção (máquinas) mais modernas, o processo é mais rápido; além disto, o tempo de limpeza varia de acordo com a matéria-prima utilizada no lote anterior. Sendo assim, é incorporado, ao custo de processamento de cada tarefa, um custo de preparação, o qual depende da tarefa que imediatamente a precede na sequência, além de depender da máquina para a qual ela foi direcionada para o processamento. Esta situação, denotada por $\beta = s_{ijk}$, de acordo com a notação de Graham et al. (1979), é conhecida por tempos de preparação dependentes da sequência e da máquina e ocorre, segundo Baker (1974), quando máquinas de

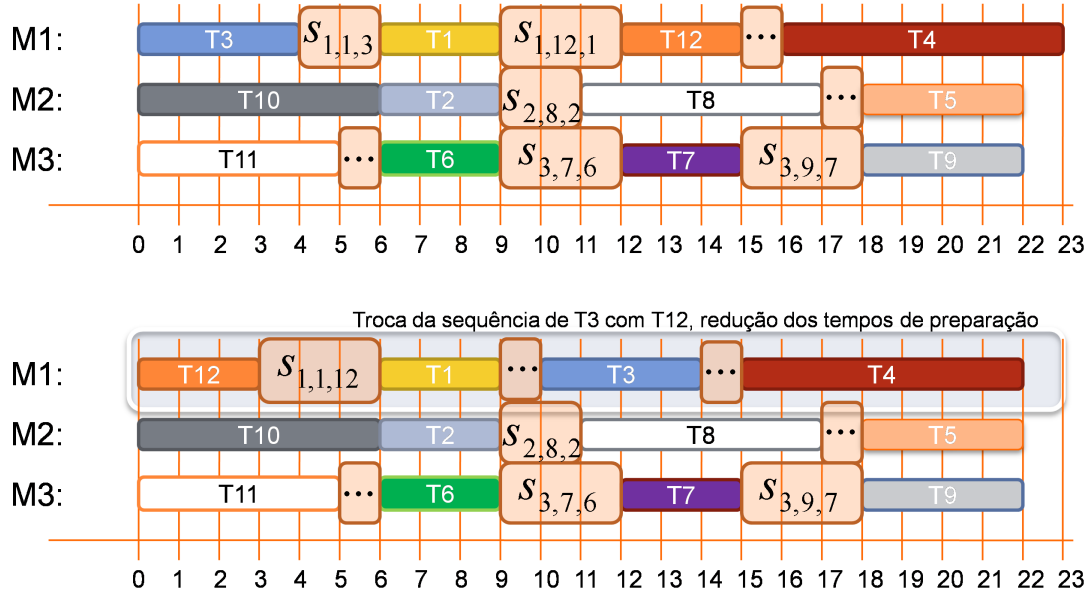


Figura 2.1: Tempos de preparação dependentes da sequência em máquinas paralelas não relacionadas.

múltiplos propósitos atendem a um grande número de tarefas muito distintas entre si. Nestes ambientes, o tempo de conclusão C_k de uma tarefa k é definido como:

$$C_k = C_j + s_{ijk} + p_{ik}, \tag{2.1}$$

sendo C_j o tempo de conclusão da tarefa j que precede imediatamente k, ambas alocadas na máquina i.

A Figura 2.1 apresenta um ambiente de máquinas paralelas não relacionadas, ilustrando a redução no instante de término da última tarefa na máquina $M1$, em consequência de uma diminuição nos tempos de preparação, ocasionada pela troca entre as tarefas $T3$ e $T12$. Neste exemplo, no primeiro sequenciamento da Figura 2.1, na máquina $M1$ estão sequenciadas quatro tarefas, na ordem $T3$, $T1$, $T12$ e $T4$. Nesta configuração a máquina $M1$ leva 23 unidade de tempo para concluir o processamento, sendo 6 unidades destinadas a tempos de preparação. A troca de ordem de processamento das tarefas $T3$ com $T12$ leva a configuração $T12$, $T1$, $T3$ e $T4$, vista no segundo sequenciamento da Figura 2.1. Pode-se observar que, apesar dos tempos de preparação da tarefa $T1$ e $T3$ terem aumentado, devido a estas serem processadas, nesta configuração, respectivamente, após a tarefa $T12$ e $T1$, o tempo de preparação da tarefa $T12$ reduziu a zero e da tarefa $T4$ se manteve o mesmo. Nesta nova configuração, a soma dos tempos de preparação reduziu em uma unidade, assim o tempo total de processamento da máquina $M1$ reduziu para 22 unidades de tempo.

2.1.3 Critérios de Otimização

Os critérios de otimização são os objetivos que devem ser minimizados ou maximizados em um processo de otimização. Em problemas de sequenciamento, estes objetivos normalmente estão associados aos instantes de conclusão das tarefas, seja buscando concluí-las o mais cedo possível ou evitando desvios em relação às datas de

entrega. Alguns critérios de otimização mais comumente usados na literatura serão descritos a seguir.

2.1.3.1 *Makespan*

Segundo Pinedo (2008), o critério de otimização denominado *makespan* é a função objetivo mais estudada em problemas de sequenciamento, sendo definida como a minimização do maior instante de conclusão de uma tarefa. O *makespan* tende a minimizar o tempo de processamento de todas as tarefas, acarretando em uma maior eficiência do ambiente de máquinas. Este critério de otimização é representado por $\gamma = C_{max}$, sendo definido como:

$$C_{max} = \max_{\forall k \in N} \{C_k\}, \tag{2.2}$$

sendo C_k o instante de término de uma tarefa k.

2.1.3.2 *Total Weighted Completion Time* - TWCT

A minimização da soma ponderado do instante de término (TWCT - *Total Weighted Completion Time*) é um critério de otimização muito observado na literatura, relacionado aos instantes de conclusões de tarefas. Este critério tem como objetivo minimizar o tempo que as tarefas permanecem na linha de produção, dando prioridade para tarefas cujo o custo de manutenção no inventário é mais elevado. Este critério de otimização é representado por $\gamma = TWCT$, sendo definido como:

$$TWCT = \sum_{k \ inN} w_k C_k, \tag{2.3}$$

sendo w_k a ponderação relacionada à permanência da tarefa k no inventário de processamento.

2.1.3.3 Soma dos atrasos com e sem ponderação

Critérios de otimização baseados em desvios em relação às datas de entrega (d_k) também são de grande importância prática. No mais simples deles, procura-se minimizar o máximo desvio (*lateness*), sendo o *lateness* de uma tarefa k definido por:

$$L_k = C_k - d_k. \tag{2.4}$$

Observe que L_k pode ter valores positivos ($C_k > d_k$) ou negativos ($C_k < d_k$). A notação empregada para a minimização do maior atraso é representada por $\gamma = L_{max}$, sendo L_{max} definido por:

$$L_{max} = \max_{\forall j \in N} \{L_j\}. \tag{2.5}$$

Apesar de existirem algumas aplicações práticas, os critérios de otimização baseados no *lateness* apresentam algumas deficiências. O *lateness* pode apresentar valores negativos, assim a minimização por custos de atraso não é permitida. Devido a isto,

os critérios de otimização baseados no indicador *tardiness* apresentam maior utilização, relacionada ao atendimento de datas de entrega. O *tardiness* de uma tarefa k é definido como:

$$T_k = \max\{0, (C_k - d_k)\}. \tag{2.6}$$

Observa-se que, na definição do *tardiness*, não existem atrasos negativos. A partir da definição mostrada na expressão (2.6), pode-se definir critérios de otimização como a soma ponderada dos atrasos ($\gamma = TWT$):

$$TWT = \sum_{k \in N} \alpha_k T_k, \tag{2.7}$$

em que α_k representa o custo (penalização) por unidade de atraso na conclusão da tarefa k. Já a soma simples dos atrasos ($\gamma = TT$), muito utilizada para aumentar o nível de satisfação de clientes, pode ser definida como:

$$TT = \sum_{j \in N} T_k. \tag{2.8}$$

2.1.3.4 Soma dos atrasos e avanços

Com a difusão da política *Just in Time* (JIT), os atrasos e avanços em relação às datas de entrega são vistos associados, respectivamente, às multas contratuais impostas por clientes e aos custos financeiros extras para antecipação da produção e/ou manutenção de estoques. Fica então evidente a importância da minimização da soma (simples ou ponderada) destes valores de atrasos e dos avanços. Definindo o avanço da tarefa k como:

$$E_k = \max\{0; (d_k - C_k)\}, \tag{2.9}$$

tem-se o critério de otimização para a soma de atrasos e avanços ($\gamma = TET$) definido como:

$$TET = \sum_{k \in N} E_k + \sum_{k \in N} T_k. \tag{2.10}$$

Já a soma ponderada de atrasos e de avanços ($\gamma = TWET$) pode ser expressa como:

$$TWET = \sum_{k \in N} \beta_k E_k + \sum_{k \in N} \alpha_k T_k, \tag{2.11}$$

em que β_k representa o custo (penalização) por unidade de avanço na conclusão da tarefa k.

2.1.3.5 Janelas de tempo

Em ambientes que adotam a política JIT, datas de entrega d_k rigidamente definidas tendem a inviabilizar o sequenciamento e, muitas vezes, podem ser evitadas na confecções de contratos, sendo substituídas por janelas de tempo de entrega. Neste

modelo, os clientes permitem que cada tarefa k seja entregue dentro do período contido em uma janela de tempo pré-definida por $[e_k, d_k]$, sendo e_k o instante inicial da janela de tempo e d_k a data limite de entrega. Sendo assim, as unidades de avanço (E_k) e atraso (T_k) de uma tarefa k são definidos, respectivamente, como:

$$E_k = \max\{0; (e_k - C_k)\} \tag{2.12}$$

$$T_k = \max\{0, (C_k - d_k)\} \tag{2.13}$$

Com esta nova definição de E_k e T_k, pode-se aplicar a Equação 2.11 para a minimização ponderada de atrasos e avanços em ambientes de sequenciamento que permitem janelas de tempo de entrega. Este contexto é representado, pela notação de Graham et al. (1979), como $\beta = e_k \neq d_k$ e $\gamma = TWET$.

2.1.4 Tempos ociosos

Em geral, na maior parte dos problemas de sequenciamento de máquinas paralelas deseja-se maximizar a utilização do ambiente. Devido a esta característica, a maior parte das formulações vinculadas a estes ambientes não consideram a possibilidade de tempos ociosos (*idle times*), i. e., tempos ociosos entre o processamento de duas tarefas consecutivas. Apesar disto, restrições impostas ao ambiente podem acarretar a necessidade da inserção de tempos ociosos, como, por exemplo, restrições do número de recursos por instante de processamento (Fanjul-Peyro et al., 2017; Yepes-Borrero et al., 2020). Em problemas de máquinas paralelas, restrições desta natureza muitas vezes acarretam na necessidade de adiar o processamento de uma tarefa em uma máquina, já que recursos disponíveis em determinado instante de tempo, podem estar alocados para o processamento das demais tarefas, alocadas em outras máquinas, naquele mesmo instante de tempo.

Ambientes que adotam políticas JIT constituem o caso de inserção de tempos ociosos mais comum na literatura de *scheduling*. Estes ambientes normalmente trabalham com critérios de minimização da soma total dos atrasos e avanços. A não inserção de tempos ociosos na minimização do TWET acarreta na minimização dos custos decorrentes de atrasos, mas, ao mesmo tempo, maximiza os custos relacionados aos avanços. Caso os custos com avanço sejam mais relevantes que os custos de atraso, não considerar os tempos ociosos incorre em perdas consideráveis no sequenciamento.

Os tempos ociosos neste trabalho são definidos como $\beta = I_{ijk}$, sendo I_{ijk} o intervalo de tempo ocioso que a máquina i irá aguardar entre o intervalo de processamento de duas tarefas j e k imediatamente consecutivas. Assim, em ambientes em que se permitem tempos ociosos, o tempo de conclusão de uma tarefa k é definido como:

$$C_k = C_j + I_{ijk} + s_{ijk} + p_{ik}, \tag{2.14}$$

sendo i a máquina para a qual a tarefa k é atribuída; C_j o instante de término da tarefa j que imediatamente precede k na máquina i, e s_{ijk} e p_{ik}, respectivamente, os tempos de processamento e preparação da tarefa k, conforme definido nas seções anteriores.

2.2 Ambientes de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência e das máquinas

Considerando as definições de problemas de sequenciamento apresentadas nas seções anteriores, um ambiente de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência e das máquinas (UPMSMDST - *Unrelated Parallel Machine with Sequence and Machine Dependent Setup Times*) consiste no sequenciamento de um conjunto de tarefas $N = \{1, 2, \ldots, n\}$ em um conjunto $M = \{1, 2, \ldots, m\}$ de máquinas, disponíveis continuamente e não relacionadas. Cada tarefa $k \in N$ possui um tempo de processamento p_{ik}, dependente da máquina i na qual esta será processada. Duas tarefas não podem ser processadas na mesma máquina ao mesmo tempo e, uma vez iniciado o processamento de uma tarefa k, este não pode ser interrompido até a conclusão do processamento. Além dos tempos de processamento, cada tarefa k possui um tempo de preparação s_{ijk}, o qual é dependente da sequência e da máquina, ou seja, depende da tarefa j, que precede imediatamente à tarefa k, na sequência da máquina i. Cada tarefa k pode possuir um *job ready time* r_k, o qual indica o instante de tempo mínimo no qual a tarefa pode iniciar o seu processamento. Pode ser permitido, ainda, a presença de tempos ociosos, I_{ijk}, entre o processamento de duas tarefas j e k atribuídas a uma mesma máquina i. Considerando este ambiente, pode-se definir, de maneira ampla, o instante de término de uma tarefa k como:

$$C_k = \max\{r_k, (C_j + I_{ijk} + s_{ijk})\} + p_{ik},$$

sendo j a tarefa imediatamente anterior a k na sequência da máquina i e C_j o instante de término de processamento da tarefa j na máquina i.

Dada a descrição do ambiente UPMSMDST, suas características são sumarizadas a seguir:

(i) As tarefas do conjunto $N = \{1, 2, \ldots, n\}$ devem ser atribuídas a um conjunto de máquinas paralelas não-relacionadas $M = \{1, 2, \ldots, m\}$;

(ii) O tempo de processamento da tarefa k na máquina i é representado por p_{ik};

(iii) Todas as tarefas estão disponíveis no instante r_k do horizonte de planejamento; caso $r_k = 0$, indica que a tarefa está disponível no instante inicial do planejamento;

(iv) Cada tarefa é composta por apenas um estágio (operação);

(v) Uma vez iniciado, o processamento da tarefa k na máquina i não pode ser interrompido, ficando a máquina k impedida de processar qualquer outra tarefa, antes de concluir a tarefa k;

(vi) Todas as tarefas podem ser processadas em qualquer uma das máquinas;

(vii) Os tempos de preparação dependem da sequência de processamento das tarefas e da máquina e são representados por s_{ijk}. Não existe um tempo de preparação da máquina após o processamento da última tarefa a ela atribuída, mas pode ser

considerado um instante de preparação inicial antes da primeira tarefa, sendo este definido como s_{i0k};

(viii) Entre duas tarefas j e k, pode haver a presença de tempos ociosos, indicado por I_{ijk}; caso $I_{ijk} = 0$, indica que não há intervalo ocioso na máquina i entre os processamentos de j e k.

Segundo as características apresentadas acima, a notação de Graham et al. (1979) para o ambiente UPMSMDST base estudado nesta tese é representado por $Rm \mid s_{ijk} \mid \gamma$, sendo γ algum dos critérios de otimização descritos na Seção 2.1.3. Em alguns casos específicos, restrições podem ser incorporadas ao problema, como tempos ociosos ($\beta = I_{ijk}$), *job ready times* ($\beta = r_k$) e janelas de tempo ($\beta = e_k \neq d_k$). Assim, especificamente no Capítulo 4, utiliza-se, para o estudo de construção de operadores de busca local em ambientes UPMSMDST, o problema que pode ser representado por $Rm \mid s_{ijk}, r_k \mid TWT$. No Capítulo 5, destinado ao estudo de uma metaheurística híbrida para a solução de diferentes critérios de otimização, o problema é definido como $Rm \mid s_{ijk}, r_k \mid \gamma$, sendo que γ varia de acordo com o estudo de caso realizado. Neste capítulo são avaliados quatro estudos de caso, cada um destinado a um critério de otimização. Assim, $\gamma \in \{TWT, C_{max}, TT, TWCT\}$. Por fim, no Capítulo 6, é avaliada a influência de janelas de tempo na construção de metaheurísticas para ambientes UPMSMDST. Assim, o problema no Capítulo 6 é classificado como $Rm \mid s_{ijk}, I_{ijk}, e_k \neq d_k \mid TWET$.

2.3 Revisão Bibliográfica para ambientes UPMSMDST

A literatura para problemas de sequenciamento de tarefas em máquinas paralelas com tempos de preparação dependentes da sequência (PMSDSP -*Parallel Machines Sequence Dependent Scheduling Problem*) vem crescendo continuamente nas últimas décadas. Parker et al. (1977) foi um dos pioneiros, estudando um problema para minimização da soma dos custos dos tempos de preparação em máquinas paralelas e propondo, para a resolução do mesmo, uma adaptação da heurística de Clarke & Wright, proposta inicialmente para o problema de roteamento de veículos em Clarke e Wright (1964). Apesar deste trabalho precursor, o PMSDSP somente começou a ganhar maior destaque na literatura ao final dos anos 1990. Para máquinas idênticas e/ou uniformes, foram propostas uma infinidade de abordagens envolvendo os mais diversos critérios, como: modelo de otimização inteira para a minimização do avanço/atraso proposto por Balakrishnan et al. (1999); as metaheurísticas *Tabu Search* (TS) e GRASP (*Greedy Randomized Adaptive Search Procedure*), para a minimização da soma ponderada dos atrasos ($\gamma = TWT$), respectivamente propostas por Bilge et al. (2004) e Armentano e Filho (2007); as heurísticas construtivas *Multiple MULTI-FIT*, *Multiple Insertion* (MI) e *Slicing* para a minimização do *makespan*, proposta por Kurz e Askin (2001), além de inúmeras outras abordagens.

O primeiro estudo de um PMSDSP para máquinas não relacionadas (UPMSMDST) foi realizado por Zhu e Heady (2000). Este trabalho apresenta um modelo de formulação matemática para a minimização da soma do avanço/atraso e tempos de preparação dependentes apenas da sequência e não das máquinas. O modelo é capaz

de resolver, em tempo computacional viável, problemas de até 9 tarefas alocadas em 3 máquinas.

Kim et al. (2002) apresentam um algoritmo *Simulated Annealing* (SA) para minimização da soma ponderada dos atrasos com tempos de preparação dependentes da sequencia e da máquina. A solução inicial é construída através da regra de despacho EDD (*Earliest Due Date*) (Pinedo, 2008), e as pertubações aleatórias do SA são realizadas por seis estruturas de vizinhança. O método proposto é comparado a um SA convencional, no qual as estruturas de vizinhança não consideram regras do problema, usando movimentos puramente aleatórios. A versão do SA proposta apresenta resultados superiores à abordagem convencional.

Fowler et al. (2003) propõem um algoritmo genético (GA) integrado com a regra de despacho ATCS (*Apparent Tardiness Cost with separable Setups*) (Lee et al., 1997). Nesta abordagem, um GA é usado para atribuir as tarefas às máquinas; em seguida, para cada solução da população, a regra de despacho ATCS é aplicada separadamente em cada máquina, gerando a sequência na máquina. A abordagem proposta é comparada a um SA desenvolvido pelos autores e a uma abordagem baseada na regra de despacho ATCS para ambientes PMSDSP proposta por Lee e Pinedo (1997). Foram testados instâncias de 100 tarefas sequenciadas em 5 máquinas e os resultados da abordagem GA híbrida apresentou resultados superiores as demais abordagens.

Logendran et al. (2007) desenvolveram seis heurísticas de busca baseadas na metaheurística Busca Tabu (*Tabu Search* – TS) para resolução da minimização da soma ponderada dos atrasos. São considerados os instantes de disponibilização das tarefas ($\beta = r_j$) e das máquinas ($\beta = r_i$). Para o mesmo problema, Lopes e de Carvalho (2007) desenvolveram um algoritmo *Branch and Price* e conseguiram resolver até a otimalidade, em tempo computacional viável, problemas para a razão de $n/m = 5$.

Rocha et al. (2008) propõem um algoritmo *Branch & Bound* (B&B) para a minimização da soma ponderada dos atrasos em UPMSMDSTs. Para geração de *upper bounds*, uma metaheurística GRASP é utilizada, sendo usado como função gulosa a regra de despacho EDD. A busca local é baseada em estruturas de vizinhanças de Troca (Seção 2.5.2 e 2.5.4). Ao fim de cada iteração, é utilizada uma estratégia de *Path relinking* (PR). Para todos os problemas de até 25 tarefas alocadas, o algoritmo conseguiu encontrar soluções ótimas; já para instâncias acima de 70 tarefas, o B&B não foi capaz de incrementar as soluções obtidas pelo GRASP.

Chen (2012), por sua vez, realiza um estudo da minimização ponderada do número de tarefas atrasadas. Para a resolução do problema, os autores propuseram uma heurística híbrida, que integra as metaheurísticas TS e *Variable Neighbourhood Descent* (VND). Os autores testaram quatro métodos de criação de solução inicial: geração randômica, regras de despachos *Earliest Weighted Due Date* (EWDD) baseadas na EDD, *Shortest Ready Times* (SRT) e pela heurística EMBISG (Chen e Chen, 2008). São construídas quatro estruturas de vizinhança baseadas tanto em movimentos de troca (Seção 2.5.2 e 2.5.4), quanto Inserção (Seção 2.5.1 e 2.5.3). O método TS com busca local VND e inicialização da solução pela regra EMBISG apresentou os melhores resultados.

Lin e Hsieh (2014) apresentam um modelo matemático inteiro misto para o UPMSMDST com minimização da soma ponderada dos atrasos, considerando instantes de disponibilização das tarefas ($\beta = r_j$). Este modelo é capaz de resolver, em tempo computacional viável, problemas de até 12 tarefas em 3 máquinas, sendo mais

eficiente que o proposto por Zhu e Heady (2000). Além disto, os autores propuseram uma regra de despacho baseada na ATCS, denominada ATCSR_Rm (*Apparent Tardiness Cost with separable Setup and Ready times for unrelated parallel machines*), propondo uma abordagem que integra esta à metaheurística EMA (*Electro Magnetism Algorithm*) (Birbil e Fang, 2003) e métodos de busca local. Esta abordagem é denominada IHM (*Iterated Hybrid Metaheuristic*) alcançou resultados superiores à metodologia TS, proposta por Logendran et al. (2007), e ao método ACO (*Ant Colony Optimization*), proposto em Lin et al. (2013).

Ainda para minimização do TWT, Li (2018) propõe um nova regra de despacho baseada na ATCS e nomeada ATCSR_UP, a qual apresenta resultados superiores à regra ATCSR_Rm (Lin e Hsieh, 2014). No trabalho, os autores também apresentam operadores de busca local baseados em movimentos de Troca (Seções 2.5.2 e 2.5.4). Diana et al. (2018), por sua vez, propõem um estudo com o uso do VND (*Variable Neighborhood Descent*) como operador de busca local da metaheurística ILS (*Iterated Local Search*). Nos cenários avaliados, a abordagem apresenta resultados superiores aos apresentados pelas metaheurística IHM (Lin e Hsieh, 2014) e ACO (Lin et al., 2013). Já Kramer e Subramanian (2019) propõem um *framework* híbrido baseado na metaheurística ILS usando o VND randômico como operador de busca local. Este *framework* é destinado à resolução de diversas variações de problemas de *scheduling* que consideram diferentes objetivos envolvendo datas de entrega. A abordagem apresenta resultados extremamente eficientes para *scheduling* em ambientes UPMS. Apesar disto, os autores não realizam experimentos nestes ambientes considerando tempos de preparação.

Afim de verificar o sequenciamento em um ambiente UPMSMDST com o objetivo de garantir satisfação dos clientes, Chen (2009) propõe um estudo para a minimização do *Total Tardiness* (TT), incorporando restrições nas datas de entrega. As tarefas destinadas a clientes primários são proibidas de apresentarem atrasos. Esta restrição teria como objetivo a satisfação destes clientes e a segurança no plano estratégico da manufatura. Para resolver este problema, Chen (2009) propõe uma abordagem SA. Com o intuito de apresentar técnicas que melhorassem o plano de sequenciamento deste problema, Lin et al. (2011), Ying e Lin (2012) e Lee et al. (2013) propõem, respectivamente, para resolução do problema, as abordagens IGS (*Iterated Greedy Search*), ABC (*Artificial Bee Colony*) e TS. Zeidi e MohammadHosseini (2015) por sua vez apresentam um Algoritmo Genético (GA) integrado a uma abordagem SA usada com operador de busca local. É proposto um novo operador de cruzamento muito eficiente para minimização do TT e TWT. Os resultados desta abordagem se apresentaram muito eficientes, sendo superiores aos apresentados pelas abordagens IGS (Lin et al., 2011) e ABC (Ying e Lin, 2012).

Apesar da grande relevância de ambientes de máquinas paralelas não relacionadas com tempos de *setup*, poucos trabalhos são encontrados na literatura considerando a filosofia JIT. Zhu e Heady (2000) propõem um *Mixed-Integer Programming* (MIP) para a minimização da soma de atrasos e avanços em máquinas paralelas. Randall e Kurz (2007) realizam um estudo de operadores de *crossover* em algoritmos genéticos, sem considerar tempos ociosos. Şen e Bülbül (2015) utilizam o algoritmo de decomposição de Benders para a resolução de MIPs com relaxações. Nestes modelos não foram considerados tempos de *setup*. Polyakovskiy e M'Hallah (2014) e Cheng e Huang (2017) propõem, respectivamente, um sistema multiagente e um algoritmo genético

híbrido. Ambos os trabalhos também não consideram tempos de *setup*. Já Nogueira et al. (2014) propõem um MIP para o problema, considerando tempos de preparação dependentes da sequência e da máquina. Para a resolução de problemas de grande dimensionalidade, os mesmos autores propõem uma metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) integrada a uma abordagem *Iterated Local Search* (ILS) e *Path Relinking* (PR). Vallada e Ruiz (2012) propõem um MIP e um Algoritmo Genético para o mesmo problema estudado por Nogueira et al. (2014). Vale destacar que todos estes trabalhos utilizam datas de atraso iguais às datas de avanço. Assim, nenhum trabalho encontrado pelos autores considera a utilização de janelas de tempo de minimização de atrasos e avanços em ambientes de máquinas paralelas não relacionadas com tempos de preparação, com ou sem a possibilidade de inserção de tempos ociosos.

Para a minimização do TWCT, Chen (2015) apresenta um estudo de três abordagens, todas baseadas na metaheurística *Record-to-Record Travel* (RRT) integrada a uma heurística de geração de soluções *Shortest Weighted Processing Time* (SWPT). Assim como em Chen (2009), Chen (2015) utiliza, para o critério de TWCT, a restrições nas datas de entregas para clientes prioritários. A variação denominada RRT1 apresentou os melhores resultados para os experimentos realizados.

O instante máximo de término de uma tarefa, denominado *makespan*, é o critério de otimização mais estudado para ambientes UPMSMDST. Al-Salem (2004) propõe uma heurística de três fases, denominada *Partition Heuristic*. Na primeira fase é realizada a construção de uma solução, utilizando-se uma heurística similar à heurística MI de Kurz e Askin (2001), mas variando a forma como o *makespan* é calculado. Na segunda fase é aplicada a heurística de refinamento *Composite Exchange Heuristic*, de Hariri e Potts (1991), alterada para suportar tempos de preparação dependentes da sequência. Na última fase, realiza-se uma pós-otimização em cada máquina, isoladamente, a fim de diminuir os tempos de preparação. Nesta fase, em que a sequência de tarefas atribuídas a cada máquina é vista como um problema do caixeiro viajante, utiliza-se a heurística *Nearest Neighbor Heuristic*. O conjunto de instâncias proposto em Al-Salem (2004) é utilizado em diversos trabalhos publicados na literatura e também no presente trabalho.

Em Rabadi et al. (2006) é apresentada a metaheurística denominada *Meta-RaPS* (*Metaheuristic for Randomized Priority Search*). Trata-se de uma variação na fase construtiva do GRASP, em que parte dos elementos é inserida com base em uma função estritamente gulosa. Os elementos restantes são inseridos de forma aleatória, a partir da lista restrita de candidatos do GRASP padrão. Na fase de busca local são exploradas estruturas de vizinhança segundo a estratégia *Best Improvemnt* (BI) (discutida na Seção 3.3.2). Os resultados alcançados por Rabadi et al. (2006) superaram os apresentados por Al-Salem (2004).

Uma implementação de TS é apresentada por Helal e Al-Salem (2006) e aplicada às instâncias criadas por Al-Salem (2004), superando os resultados apresentados por Al-Salem (2004). Não há uma comparação com os resultados apresentados por Rabadi et al. (2006), uma vez que os trabalhos foram publicados na mesma época. No entanto, Arnaout et al. (2010) mostram que a metaheurística Meta-RaPS supera a TS.

Um algoritmo Colônia de Formigas (ACO) é proposto por Arnaout et al. (2010). O algoritmo opera em duas fases. Na primeira, as tarefas são atribuídas às máquinas. Na segunda, as tarefas atribuídas a cada máquina são sequenciadas. Ao final, é rea-

lizada uma busca local randômica, utilizando-se as estruturas de vizinhança de troca e inserção externas. Os resultados alcançados por Arnaout et al. (2010) mostram-se superiores a todos os resultados anteriormente divulgados para o conjunto de instâncias em questão, isto é, os provindos de Al-Salem (2004), Rabadi et al. (2006) e Helal e Al-Salem (2006).

Ying et al. (2012) propõem a utilização de uma metaheurística *Simulated Annealing*, sendo propostas duas variações do algoritmo. Na primeira variação, as buscas locais são realizadas através da descida completa em todas as estruturas de vizinhança. Na segunda, são avaliados apenas os movimentos que atuem sobre a máquina que define o *makespan*, uma vez que apenas alterações em tal máquina são capazes de promover a redução no *makespan*. As duas variações encontraram resultados muito próximos, mas a segunda requer menor esforço computacional. Os resultados alcançados por Ying et al. (2012) mostraram-se superiores a todos os resultados anteriormente divulgados para o conjunto de instâncias em questão (Al-Salem (2004), Rabadi et al. (2006), Helal e Al-Salem (2006) e Arnaout et al. (2010)).

Também para o *makespan*, Vallada e Ruiz (2011a) propõem um algoritmo genético. Este é aplicado a um novo conjunto de instâncias, diferente do proposto por Al-Salem (2004). Na construção da população inicial, um único indivíduo é construído, segundo a heurística *Multiple Insertion* (MI) de Kurz e Askin (2001). Os demais são construídos de modo aleatório. Com uma certa probabilidade, os indivíduos da população inicial podem ser submetidos a uma busca local na vizinhança de inserções externas. A busca é conduzida entre pares de máquinas. No processo de criação de novos indivíduos, que se dá por meio do operador de *crossover* *One Point Crossover*, utiliza-se uma busca local limitada a cada máquina, com movimentos de inserção interna. Os indivíduos recém-criados também podem ser submetidos a uma busca local (idêntica àquela aplicada aos indivíduos da população inicial). Na busca local, merece destaque a avaliação desnecessária de movimentos que não afetam diretamente a máquina mais carregada. Para avaliar o desempenho do algoritmo proposto, a heurística *Multiple Insertion* e o algoritmo genético *GAK* de Kurz e Askin (2001), bem como a heurística *Meta-RaPS* (Rabadi et al., 2006) foram implementadas. Os resultados indicaram um desempenho consideravelmente superior do algoritmo genético de Vallada e Ruiz (2011a).

Arnaout et al. (2014) realizam uma nova investigação sobre ACO, propondo uma extensão, denominada ACOII, da abordagem proposta em Arnaout et al. (2010). Nesta nova investigação, o problema é dividido em duas fases. Primeiramente, considera-se um problema de atribuição de tarefas em máquinas. Após esta fase, é realizado a fase de sequenciamento das tarefas, atribuídas às máquina na fase anterior. A principal diferença entre as duas abordagens se dá na forma como é atualizado o feromônio. A abordagem ACOII apresentou, na maior parte dos cenários, resultados superiores às abordagens ACO (Arnaout et al., 2010) e SA (Ying et al., 2012).

Já Lin e Ying (2014) propõem um algoritmo *Hybrid Artificial Bee Colony* (HABC). Esta é uma abordagem hibrida que integra o algoritmo ABC com as fases construtivas de destrutivas da metaheurística IG, além de métodos de busca local. Estes são baseados em descida *First Improvement* (FI) com estruturas de vizinhança de Troca. O algoritmo é baseado na abordagem proposta em Ying e Lin (2012) para minimização da soma ponderada dos atrasos. Esta abordagem divide com o CSA (*Clonal Selection Algorithm*) (Diana et al., 2015) e WO (*Worm Optimization*) (Arnaout, 2019),

os resultados de estado-da-arte para o conjunto de instâncias proposto por Al-Salem (2004). Já Eroglu et al. (2014), por sua vez, estuda um Algoritmo Genético integrado a um operador de busca local denominada pelos autores como GALA. Esta abordagem apresentou resultados satisfatórios, comparáveis as abordagens ACO (Arnaout et al., 2010) e SA (Ying et al., 2012).

Diana et al. (2015) apresentam uma proposta de implementação do algoritmo de seleção clonal (CSA) com a criação de soluções pela fase construtiva da metaheurística GRASP. Este trabalho propõe uma regra de despacho denominada *Dynamic Multiple Insertion*, usada como função gulosa da fase construtiva do GRASP. Além disto, é proposto um método de busca local, que considera a característica do objetivo de minimização do *makespan*, em que apenas alterações na máquina mais carregada podem alterar o objetivo. Esta abordagem é comparada com o GA proposto por Vallada e Ruiz (2011a), através do conjunto de instâncias proposto pelos mesmos, e com as abordagens ACOII (Arnaout et al., 2014) e SA (Ying et al., 2012), através do conjunto e instâncias proposto por Al-Salem (2004). Os resultados encontrados foram superiores a todas as abordagens comparadas. Santos et al. (2019) propõem uma análise das buscas locais estocásticas *Simulated Annealing* (SA), *Iterated Local Search* (ILS), *Late Acceptance Hill-Climbing*(LAHC) e *Step Counting Hill-Climbing* (SCHC). São avaliadas seis estruturas de vizinhança baseadas em movimentos de Inserção e Troca. As abordagens propostas são comparadas entre si através do conjunto de instâncias proposto por Vallada e Ruiz (2011a). A abordagem SA apresentou os melhores resultados para a maior parte dos problemas, principalmente para problemas de maior dimensionalidade. A principal vantagem destas abordagens é a simplicidade. Apesar de não haver comparação no trabalho, é possível verificar, através dos resultados, que o desempenho desta abordagem foi inferior às abordagens de estado da arte HABC (Lin e Ying, 2014), CSA (Diana et al., 2015) e WO (Arnaout, 2019).

Já Wang et al. (2016) propõem uma abordagem híbrida, que integra o algoritmo *Estimation of Distribution Algorithm* (EDA) (Larranaga e Lozano, 2002) com a metaheurística IG. Esta abordagem é comparada com o algoritmo genético proposto por Vallada e Ruiz (2011a), encontrando resultados superiores a este na maior parte dos cenários avaliados.

Cota et al. (2017) propõem uma abordagem *Adaptive Large Neighborhood Search* (ALNS). Uma solução inicial é gerada pela fase construtiva da metaheurística GRASP, utilizando, como função gulosa, a regra de despacho *Adaptive Shortest Processing Time* (ASPT) (Pinedo, 2008). É proposta uma busca local VND utilizando sempre a estratégia *First Improvement*. Três estruturas de vizinhança são utilizadas, sendo baseadas em movimentos de Inserção e Troca. Os experimentos realizados foram executados sobre o conjunto de instâncias proposto por Al-Salem (2004). A abordagem é comparada com as abordagens ACO (Arnaout et al., 2010) e ACOII (Arnaout et al., 2014). Apesar de apresentar resultados superiores a ambas, este método não é comparado às abordagens estado-da-arte para este conjunto de instâncias, i.e., HABC (Lin e Ying, 2014) e CSA (Diana et al., 2015).

Arnaout (2019) introduz a implementação da metaheurística *Worm Optimization* (WO), a qual, segundo o autor, apresentou resultados superiores às abordagens Tabu Search (Helal e Al-Salem, 2006), ACO (Arnaout et al., 2010), RSA (Ying et al., 2012), HABC (Lin e Ying, 2014), e GALA (Eroglu et al., 2014). Apesar de não ser comparado às abordagens HABC (Lin e Ying, 2014) e CSA (Diana et al., 2015), é pos-

sível verificar que a metaheurística WO apresentou resultados superiores a ambas em alguns cenários, principalmente para grupos de instâncias de baixa dimensionalidade.

Ainda para o critério de minimização do *makespan* em ambientes UPMSMDST, existem dois estudos que apresentam abordagens determinísticas exatas. O primeiro, realizado por Tran et al. (2016), propõem dois métodos de decomposição, denominados *Logic-Based Benders Decomposition* (LBBD) e *Branch-and-Check* (B&C). Já Fanjul-Peyro et al. (2019) propõem um *Mixed Integer Programming* e um método denominado *Mathematical Programming-based Algorithm* (MPA). Estas abordagens foram capazes de encontrar soluções próximas aos ótimos, mesmo para problemas de grande dimensão. Em contrapartida, demandam um custo computacional muito elevado, sendo os experimentos realizados com três horas de duração para cada problema teste e ainda utilizando recursos de processamento paralelo.

Um problema muito similar ao considerado em Al-Salem (2004) é investigado por de Paula et al. (2007). A diferença fica por conta da função objetivo, que passa a ser a minimização da soma do *makespan* com uma soma ponderada dos quadrados dos atrasos, ou seja, $C_{max} + \sum w\ {T_j}^2$. É utilizada a metaheurística VNS (*Variable Neighbourhood Search*), em que são exploradas as vizinhanças de trocas internas, trocas externas e inserções externas. Nas duas primeiras ocorre a exploração das vizinhanças completas. Na última, são avaliados apenas os movimentos que retiram tarefas da máquina mais carregada e as inserem na máquina menos carregada. Comparações realizadas contra três variações do GRASP indicam a superioridade da metaheurística VNS. Caniyilmaz et al. (2015) realiza um estudo similar ao realizado por de Paula et al. (2007), considerando, como objetivos, a minimização da soma do *makespan* com o atraso total (TT). Para resolução deste problema, Caniyilmaz et al. (2015) propõem uma metaheurística ABC.

Na Tabela 2.1 é apresentado um resumo dos trabalhos encontrados na literatura para ambientes UPMSMDST.

2.4 Instâncias

Nesta seção apresentamos os principais conjuntos de instâncias utilizadas como *benchmark* para as abordagens destinadas a resoluções de problemas de *scheduling* em ambientes UPMSMDST.

2.4.1 TWT

Para ambientes UPMSMDST com o objetivo de minimização o TWT (Equação 2.7), Lin e Hsieh (2014) propôs um conjunto de 640 instâncias, estas sendo divididas em dois grupos. O primeiro grupo, chamado *small*, é composto por 320 instâncias com 12 tarefas a serem processadas em 3 máquinas. Este conjunto apresenta soluções ótimas geradas por um *Mixed-Integer Programming* (MIP). O segundo grupo, denominado *large*, é composto por 320 instâncias de 100 tarefas e 10 máquinas e as melhores soluções conhecidas são obtidas pelas metaheurísticas *Iterated Hybrid Metaheuristic* (IHM) (Lin e Hsieh, 2014), *Ant Colony Optimization* (ACO) (Lin et al., 2013) e Iterated Local Search + VND (Diana et al., 2018). Em ambos os grupos, os tempo de processamento p_{ik} de uma tarefa k e máquina i e o custo de atraso

Tabela 2.1: Sumário da literatura destinada a resolução de problemas de *scheduling* em ambientes UPMSMDST.

Trabalho	Métodos computacionais	Critério
Kim et al. (2002)	SA	TWT
Fowler et al. (2003)	GA + ATCS	TWT
Logendran et al. (2007)	TS	TWT
Lopes e de Carvalho (2007)	B&P	TWT
Rocha et al. (2008)	B&B + GRASP + PR	TWT
Chen (2012)	EMBISG + TS + VND	TWT
Lin et al. (2013)	ACO	TWT
Lin e Hsieh (2014)	ATCSR_Rm + EMA + Busca Local	TWT
Diana et al. (2018)	ILS+VND	TWT
Li (2018)	ATCSR_UP + Busca Local	TWT
Chen (2009)	SA	TT
Lin et al. (2011)	IGS	TT
Ying e Lin (2012)	ABC	TT
Lee et al. (2013)	TS	TT
Zeidi e MohammadHosseini (2015)	GA + SA	TT
Zhu e Heady (2000)	MIP	TTE
Vallada e Ruiz (2012)	GA	$TWET$
Nogueira et al. (2014)	GRASP + ILS + PR + Busca Local	$TWET$
Kramer e Subramanian (2019)	ILS + Random VND	TWT, TT, $TWET$
Chen (2015)	SWPT + RRT + Busca Local	$TWCT$
Al-Salem (2004)	Partition Heuristic	C_{max}
Rabadi et al. (2006)	Meta-RaPS	C_{max}
Helal e Al-Salem (2006)	TS	C_{max}
Arnaout et al. (2010)	ACO + Busca Local	C_{max}
Ying et al. (2012)	SA + Busca Local	C_{max}
Vallada e Ruiz (2011a)	MI + GA + Busca Local	C_{max}
Arnaout et al. (2014)	ACO + Busca Local	C_{max}
Lin e Ying (2014)	ABC + IGS + Busca Local	C_{max}
Eroglu et al. (2014)	GALA	C_{max}
Diana et al. (2015)	CSA + GRASP + Busca Local	C_{max}
Santos et al. (2019)	SA, ILS, LAHC, SCHC	C_{max}
Tran et al. (2016)	B&C	C_{max}
Wang et al. (2016)	EDA + IGS	C_{max}
Cota et al. (2017)	ALNS	C_{max}
(Arnaout, 2019)	WO	C_{max}
Fanjul-Peyro et al. (2019)	MPA	C_{max}
de Paula et al. (2007)	GRASP, VNS	$C_{max} + TWT$
Caniyilmaz et al. (2015)	ABC	$C_{max} + TT$

w_k são gerados aleatoriamente, com distribuição uniforme, sendo $p_{ik} \in [50, 150]$ e $w_k \in [0, 10]$. Os tempos de preparação são gerados com distribuição uniforme entre $[0, 2\bar{s}]$, sendo $\bar{s} = \eta \times \bar{p}$ e $\bar{p} = \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}/mn$. As datas de entrega d_k são geradas aleatoriamente, sendo $d_k \in [(1 - R)\bar{d}, \bar{d}]$, com probabilidade τ; já com probabilidade $1-\tau$, $d_k \in [\bar{d}, \bar{d}+(C_{max}-\bar{d})R]$, sendo a data de entrega estimado $\bar{d} = C_{max}(1-\tau)$, e o *makespan* estimado $C_{max} = ((0.4 + 10/(n/m)^2)\bar{s} + \bar{p}) \times (n/m)$. Os *job ready times* r_k são distribuídos uniformemente entre $[max(d_k-(r_\tau \times \bar{p_k}), d_k]$, sendo $\bar{p_k} = \sum_{i=1}^{m} p_{ik}/m$. As instâncias são divididas em 4 fatores, da seguinte maneira:

(i) Fator de severidade dos tempos de preparação: $\eta = [0.02; 2]$;

(ii) Fator de proximidade entre datas de entrega: $\tau = [0.3; 0.9]$;

(iii) Fator de proximidade das datas de entrega com o instante inicial de planejamento: $R = [0.25; 1]$; and

(iv) Fator de proximidade dos *job ready times*: $r_\tau = [1; 10]$.

Cada combinação de fatores $\eta \times \tau \times R \times r_\tau$ possui 20 instâncias e totalizam 16 grupos.

2.4.2 Makespan

Para o critério de minimização do *makespan* (Equação 2.2), a maior parte dos trabalhos utiliza o conjunto de instâncias proposto por Al-Salem (2004). Entre os trabalhos disponíveis na literatura que utilizam este conjunto como instância de *benchmark*, as seguintes metaheurísticas são aplicadas: *Ant Colony Optimization* (ACOII) (Arnaout et al., 2014); Hybrid Artificial Bee Colony (HABC) (Lin e Ying, 2014), o *Clonal Selection Algorithm* (CSA) (Diana et al., 2015), o algoritmo genético com busca local GALA (Eroglu et al., 2014) e o Worm Optimization (WO) (Arnaout, 2019). Dentre as abordagens determinísticas, podemos destacar, *Branch and Check* (B&C) (Tran et al., 2016) e *Mathematical Programming-based Algorithm* (MPA) (Fanjul-Peyro et al., 2019). Este conjunto é composto por 1620 instâncias sendo estas divididas em 3 subconjuntos. No primeiro, os tempos de processamento são distribuídos uniformemente no intervalo $U[125, 175]$, enquanto os tempos de preparação seguem a distribuição $U[50, 100]$, ou seja, os tempos de processamento são sempre maiores que os tempos de preparação. No segundo subconjunto, as distribuições são invertidas, sendo os tempos de preparação maiores que os tempos de processamento. Já no terceiro subconjunto, há um balanceamento entre tempos de processamento e preparação, ambos distribuídos com probabilidade uniforme entre o intervalo $U[50, 100]$. Os três grupos são respectivamente denominados aqui como *Proc-Domain*, *Setup-Domain* e *Balanced*. Cada subconjunto possui 450 instâncias, configuradas com $N = \{20, 40, 60, 80, 100, 120\}$ e $M = \{2, 4, 6, 8, 10, 12\}$, formando 36 grupos de diferentes dimensões $(n \times m)$, com 15 instâncias em cada grupo.

2.4.3 *Total Tardiness*

Para o problema de minimização do *Total Tardiness* (Equação 2.8) em ambientes UPMSMDST com restrições em datas de entrega, Chen (2009) propõe um conjunto

de 960 instâncias de *benchmark*. Neste problema, clientes prioritários não podem sofrer atrasos nas datas de entrega definidas; caso isto ocorra, a solução se torna infactível. Este conjunto de instâncias é subdivido em 4 grupos de tarefas, na forma $N = \{30, 50, 70, 90\}$, e 3 grupos de máquinas, na forma $M = \{4, 6, 8\}$. Os tempos de preparação são gerados de forma aleatória, com probabilidade uniforme no intervalo $U[10, 100]$. As tarefas são subdivididas em famílias, e o número de tarefas por família é dado pelo fator f do conjunto de instâncias dividido em dois níveis, $\lfloor n/7 \rfloor + 1$ e $\lfloor n/8 \rfloor + 1$. O tempo de processamento p_{ik} de cada tarefa k alocada na máquina i é dado por $1/f_{iq} \times U[10, 40]$, em que $f_{iq} = 1/U[5, 15]$ representa a velocidade de processamento das tarefas pertencentes à família q na máquina i. As datas de entrega são geradas de forma aleatória, com probabilidade uniforme dada por $U[C_{max}(1-\tau-R/2), C_{max}(1-\tau+R/2)]$, sendo τ um fator de proximidade entre datas de entrega e R o fator de proximidade das datas de entrega com o instante inicia de planejamento, ambos apresentam dois níveis, $\tau = \{0.4, 0.8\}$ e $R = \{0.4, 1\}$. Já $C_{\max}$ representa o *makespan* estimado, dado por $C_{\max} = (\sum_{j=1}^{n} \sum_{i=1}^{m} (p_{ij} + (\sum_{k=1}^{n} s_{ijk})/n)/m)/m$. A proporção de clientes prioritários é definida pelo fator p, sendo este dado em dois níveis, na forma $p = \{0.2, 0.3\}$. Para cada combinação dos níveis dos seis fatores, $n \times m \times f \times \tau \times R \times p$, são geradas 5 instâncias. As abordagens que apresentam melhor desempenho neste conjunto de instâncias são: Iterated Greedy Search (IGS) (Lin et al., 2011), Arificial Bee Colony (ABC) (Ying e Lin, 2012) e *Genetic Algorithm* (GA) (Zeidi e MohammadHosseini, 2015).

2.4.4 *Total Weighted Completion Times*

O critério de minimização do TWCT (Equação 2.3) ainda é pouco estudado em ambientes UPMSMDST. O único trabalho encontrado (Chen, 2015) apresenta uma proposta de solução sobre o conjunto de instâncias propostas por Chen (2009) para a minimização do TT, e descrito na Seção 2.4.4. Chen (2015) mantém para o critério de TWCT a restrições nas datas de entregas para clientes prioritários. A única alteração necessária no conjunto de instâncias é o acréscimo, para cada tarefa, do peso por unidade de espera do processamento w_k, sendo este gerado com probabilidade uniforme no intervalo $[20, 90]$. Sobre este conjunto de instâncias é realizado um estudo de 3 abordagens, todas baseadas na metaheurística *Record-to-Record Travel* (RRT) integrada a uma heurística de geração de soluções *Shortest Weighted Processing Time* (SWPT).

2.5 Movimentos clássicos para ambientes de máquinas paralelas não relacionadas

Como mostrado pela revisão bibliográfica descrita na Seção 2.3, os tipos de estruturas de vizinhança mais utilizadas em ambientes UPMSMDST são baseadas em movimentos de Inserção e Troca (e.g. Chen (2009); Lin et al. (2011); Ying e Lin (2012); Fleszar et al. (2012); Diana et al. (2013, 2015, 2018); Chen (2015); Santos et al. (2019); Cota et al. (2017); Kramer e Subramanian (2019)). Estes tipos de movimentos são definidos detalhadamente nas próximas seções.

2.5.1 Inserção Interna

Em um movimento de inserção interna, uma tarefa k, que ocupa a posição p na sequência de tarefas atribuídas à máquina i, é retirada de sua posição original p e inserida em uma outra posição $p' \neq p$, na mesma máquina i. Na Figura 2.2 é mostrado um exemplo de movimento de inserção interna.

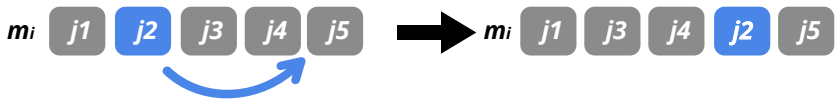


Figura 2.2: Exemplo na estrutura de vizinhança de Inserção Interna.

2.5.2 Troca Interna

Em um movimento de troca interna, duas tarefas, k e j, atribuídas a uma mesma máquina i, têm suas posições trocadas entre si. Na Figura 2.3 é mostrado um exemplo de movimento de troca interna.

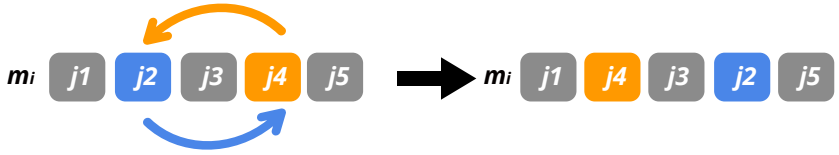


Figura 2.3: Exemplo na estrutura de vizinhança de Troca Interna.

2.5.3 Inserção Externa

Uma tarefa k é retirada da máquina i à qual estava atribuída e é inserida em uma máquina w, em que $i \neq w$. Na Figura 2.4 é mostrado um exemplo de movimento de inserção externa. Movimentos de inserções externas são os únicos, dentre os considerados no presente trabalho, capazes de alterar as cardinalidades dos conjuntos de tarefas atribuídas às máquinas. São, portanto, os únicos que, combinados com os movimentos de Inserção Interna, permitem a varredura de todo o espaço de soluções do problema de sequenciamento de tarefas em máquinas paralelas (de França Filho, 2007).

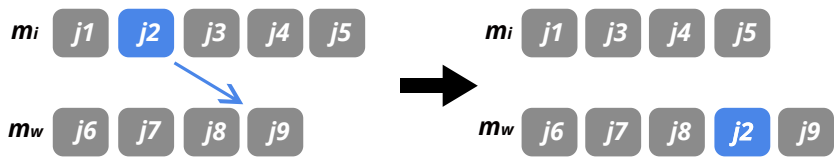


Figura 2.4: Exemplo na estrutura de vizinhança de Inserção Externa.

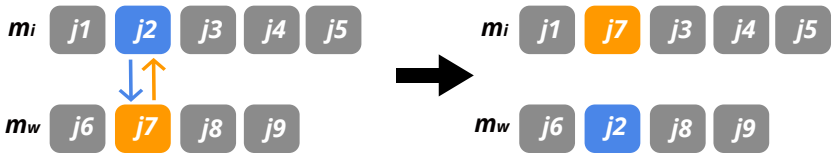


Figura 2.5: Exemplo na estrutura de vizinhança de Troca Externa.

2.5.4 Troca Externa

Seja k a tarefa que ocupa a posição p na máquina i e j a tarefa que ocupa a posição p' na máquina w, sendo $i \neq w$. Em decorrência de um movimento de troca externa, a tarefa k passa a ser processada na posição p' da máquina w e a tarefa j passa a ser processada na posição p da máquina i. Na Figura 2.5 é mostrado um exemplo de movimento de troca externa.

2.6 Lacunas encontradas na literatura de sequenciamento de tarefas em ambiente UPMSMDST

Através da revisão bibliográfica realizada, foram identificadas três grandes lacunas na literatura de sequenciamento de tarefas em ambientes UPMSMDST:

(a) Em geral, as abordagens baseadas em metaheurísticas possuem operadores de busca, mas, em contrapartida, pouca importância é dada para a análise e validação dos mecanismos de busca local no processo de busca. Assim, não é possível identificar, nas abordagens, (i) quais as contribuições das estruturas de busca local no processo de busca; (ii) qual tipo de movimento é mais efetivo para a construção de estruturas de vizinhança; e (ii) quais formas e métodos de exploração de estruturas de vizinhança são mais efetivos. Este tipo de informação é bastante relevante para guiar o projeto de operadores de busca local para abordagens ainda não contempladas na literatura;

(b) Praticamente todas as abordagens metaheurísticas presentes na literatura para resolução de problemas UPMSMDST utilizam características do critério de otimização para a construção de operadores que guiam o processo de busca. Este tipo de abordagem se mostra muito útil para aumentar a eficiência das abordagens metaheurísticas, permitindo que estas encontrem resultados muito próximos aos resultados ótimos. Por outro lado, o projeto de metaheurísticas guiado pelo critério de otimização torna estas abordagens muito específicas às características utilizadas, dificultando sua utilização para critérios de otimização com pouca ou nenhuma visibilidade teórica e que não partilham das características utilizadas no projeto do operador;

(c) Diversos cenários práticos em que se pode aplicar soluções computacionais para otimizar o planejamento de produção requerem que o sequenciamento seja guiado por uma política JIT (*Just in Time*). Considerando este cenário, foram encontradas, na literatura de UPMSMDST, duas abordagens que realizam o sequenciamento guiados por uma política JIT (Vallada e Ruiz, 2012; Nogueira et al.,

2014). Apesar disto, ambas abordagens têm a desvantagem de serem destinadas à solução de problemas com datas de avanço e atraso rigidamente definidas. Em diversos cenários práticos não há a necessidade de utilização de datas rigidamente definidas. Nestes, a utilização de janelas de tempo de entrega permite uma maior flexibilidade e auxilia na redução de custos provenientes dos atrasos e avanços. Apesar da relevância prática, nenhum estudo foi encontrado para a minimização do TWET com janelas de tempo em ambientes UPMSMDST.

Esta tese direciona, respectivamente, os Capítulos 4, 5 e 6 ao preenchimento das lacunas (a), (b) e (c) descritas acima. No Capítulo 4, é apresentada uma metodologia para o projeto de métodos de busca local de metaheurísticas para ambientes UPMSMDST. Neste estudo são mostrados os movimentos, estruturas baseadas nestes métodos, formas de exploração de estruturas de vizinhança e ordem de exploração que acarretam em incremento considerável de desempenho de metaheurísticas previamente propostas na literatura. Já no Capítulo 5 é proposta uma metaheurística híbrida, e operadores para esta, que reduzem a dependência de característica dos critérios de otimização no processo de busca. Por fim, no Capítulo 6, é realizado um estudo sobre as peculiaridades de se projetar metaheurísticas para resolução de sequenciamento de tarefas em ambientes UPMSMDST, guiado por uma política JIT com janelas de tempos.

No próximo capítulo (Capítulo 3) é apresentada a fundamentação teórica computacional necessária para a realização dos estudos apresentados nos Capítulos 4, 5 e 6.

Capítulo 3

Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica e a revisão bibliográfica dos métodos computacionais necessários para a realização desta tese. A Seção 3.1 apresenta uma definição básica e generalista sobre um problema de otimização. A Seção 3.2 apresenta conceitos sobre otimização combinatória. Já a Seção 3.3 é destinada na descrição dos métodos heurísticos de trajetória, e busca local, que são utilizados na hibridização dos algoritmos propostos. Já a Seção 3.4 apresenta uma revisão bibliográfica sobre sistemas imunológicos artificiais, sendo detalhados os conceitos do sistema imunológico natural utilizados como inspiração dos algoritmos de otimização aqui estudados. Em seguida, são traduzidos os conceitos biológicos para a metáfora computacional, detalhando um algoritmo baseado no Princípio da Seleção Clonal e dois algoritmos baseados na Teoria da Rede Imunológica, sendo um para otimização em espaço contínuo e outro para problemas combinatórios. É apresentado também um *framework* de *Ageing*, bastante utilizado na literatura para algoritmos baseados no Princípio da Seleção Clonal. Por fim, na Seção 3.4.3 é apresentada a metaheurística híbrida VNS-aiNet.

3.1 Definição de um problema de otimização mono-objetivo

Dado um espaço n-dimensional A e um critério de otimização definido por uma função $f : A \to \mathbb{R}$, um problema de otimização mono-objetivo consiste em buscar o elemento $x_0 \in A$ que minimize (ou maximize) f. Assim, para todo $x \in A$, em um problema de minimização $f(x_0) \leqslant f(x)$. Em contrapartida, para um problema de maximização, $f(x_0) \geqslant f(x)$.

Nesta tese, por convenção, todo problema de otimização sera tratado com o objetivo de minimizar f. Além disto, apesar dos problemas de sequenciamentos tratados neste trabalho serem classificados com espaço de natureza combinatória (Seção 3.2), o espaço de busca n-dimensional A pode assumir diversas naturezas ao longo do trabalho, dependendo da situação em que ele é mencionado. Por exemplo, na Seção 3.4.2.3 ao explicar a rede imunológica para otimização, o espaço n-dimensional A assume uma natureza vetorial $\mathbb{R}^n$; logo, $f : \mathbb{R}^n \to \mathbb{R}$, uma vez que, originalmente, este algoritmo é proposto para otimização vetorial. Por outro lado, na formulação matemática para problemas com janela de tempo, proposta na Seção 6.2, A é um

espaço binário; logo, $f : \mathbb{Z}_2 \rightarrow \mathbb{R}$ sendo $\mathbb{Z}_2 = \{0, 1\}$. Apesar disto, salvo exceções explicitamente indicadas, como nos casos acima, $x \in A$ sempre será uma variável de natureza combinatória.

3.2 Otimização Combinatória

Otimização combinatória é uma área da matemática aplicada e da ciência da computação, utilizada para auxílio do processo de planejamento e tomada de decisão. Tem como intuito encontrar uma ou mais soluções ótimas (ou sub-ótimas) que minimizem (ou maximizem) algum objetivo estratégico dentro de um espaço finito de soluções.

A maior parte dos problemas de otimização combinatória possui natureza NP (*Non-Deterministic Polynomial time*)[1], sendo muitos classificados como NP-Difícil (Korte e Vygen, 2008).

Existem várias técnicas para resolução de problemas de otimização combinatória. Para problemas de menor complexidade, ou baixa dimensão, resultados ótimos podem ser obtidos por técnicas como Programação Dinâmica (Bellman, reimpressão (2003), *Branch and bound* (B&B) (Land e Doig, 1960), dentre outras. Estas técnicas também podem ser usadas para a obtenção de soluções parciais, sub-ótimas, em problemas de grande dimensionalidade. Já as heurísticas e metaheurísticas não garantem a otimalidade das soluções, mas são muito empregadas para resolução de problemas de otimização combinatória NP-Completo. A maior vantagem das técnicas (meta)heurísticas é que conseguem encontrar, em tempo computacional viável, soluções sub-ótimas, de qualidade, para problemas de grande dimensão. Neste trabalho serão explorados apenas métodos (meta)heurísticos, principalmente métodos de trajetória baseados em busca através de estruturas de vizinhança e algoritmos imuno-inspirados. Assim, as próximas seções detalham os conceitos e métodos metaheurísticos utilizados nesta tese para as soluções propostas.

3.3 Heurísticas e Metaheurísticas

Heurísticas são métodos utilizados para tratar problemas de natureza NP-Difícil, para os quais a aplicação de métodos exatos exige tempo e esforço computacional inaceitáveis para instâncias de grande porte. Métodos heurísticos não garantem a obtenção de uma solução ótima, mas podem conduzir a soluções de elevada qualidade, com esforço computacional incomparavelmente menor àquele requerido por métodos exatos. Para encontrar soluções de elevada qualidade, em tempos computacionais aceitáveis, as heurísticas devem ser guiadas pelas características específicas do problema que se deseja resolver. Assim, empregando critérios racionais, apenas uma pequena porção do espaço de soluções é investigada (Kampke, 2010). Logo, alterações nas características de um problema podem exigir alterações na heurística, para que soluções de elevada qualidade continuem sendo obtidas.

[1] Problemas que podem ser resolvidos em tempo polinomial através de uma máquina de Turing não-determinística.

Metaheurísticas são estruturas gerais que incorporam um conjunto de estratégias que possibilitam uma exploração mais ampla do espaço de busca (Blum e Roli, 2003). Uma importante característica de uma metaheurística é o balanceamento dinâmico entre a diversificação e a intensificação da busca. Enquanto a diversificação tem por fim escapar de ótimos locais, permitindo que diversas regiões do espaço de busca sejam exploradas, a intensificação procura explorar mais detalhadamente uma região específica do espaço de soluções, em busca de um ótimo local.

Dois grupos importantes de metaheurísticas destinadas a otimização combinatória são: (i) métodos de busca em vizinhança (trajetória) e (ii) métodos populacionais. Segundo Melián et al. (2003), metaheurísticas de busca em trajetória, consiste em métodos que, partindo de uma solução inicial, percorrem o espaço de busca de forma iterativa, transformando a solução a cada iteração, guiando a solução para mínimos locais através de métodos de refinamento. Adicionalmente, são acionadas estratégias de diversificação que possibilitam que a solução "salte" no espaço de soluções, afim de escapar de bacias de atração pouco atraentes, ou já exploradas, para que assim busque mínimos locais de melhor qualidade através dos métodos de refinamento. Normalmente, as diferenças entre metaheurísticas deste grupo está na estratégia de diversificação adotada. Uma estratégia muito utilizada para a diversificação são movimentos aleatórios ou pseudo-aleatórios, ou movimentos realizados em diferente estruturas de vizinhança, ou mesmo, o aceite de movimentos que provoquem piora na solução.

Alguns exemplos de metaheurísticas de trajetória são: *Tabu Search* (Glover, 1986); *Simulated Annealing* (SA) (Kirkpatrick et al., 1983); ILS (*Iterated Local Search*) (Stützle, 1998). As metaheurísticas de trajetória *Variable Neighborhood Search* (VNS) (Mladenovic e Hansen, 1997), *Variable Neighborhood Descent* (VND) (Mladenovic e Hansen, 1997) e *Iterated Greedy Search* (IGS) (Ruiz e Stützle, 2007) serão detalhadas nas próximas seções, uma vez que serão utilizadas características destas nos algoritmos propostos neste trabalho.

Enquanto métodos de trajetória normalmente manipulam uma única solução fazendo com que esta caminhe no espaço de busca, metaheurísticas populacionais dispõem de um conjunto de soluções que interagem entre si, sendo que características coletivas determinam como a população irá explorar o espaço de soluções. Segundo Blum e Roli (2003), a forma como a população é manipulada irá definir o desempenho do método, sendo os algoritmos genéticos e a otimização por colônia de formigas, as metaheurísticas populacionais mais estudadas para problemas combinatórios. Os algoritmos baseados na teoria da rede imunológica, utilizadas no presente trabalho, também pertencem ao grupo de metaheurísticas populacionais, sendo abordados na Seção 3.4.

3.3.1 Estruturas de Vizinhança

Estrutura de vizinhança é um conceito muito utilizado por métodos de trajetória de otimização combinatória. Uma vizinhança é definida como o conjunto de soluções s' vizinhas a uma solução $s \in S$, sendo S o espaço de soluções de um determinado problema de otimização combinatória. Assim, a vizinhança de uma solução $s \in S$, constitui-se do subespaço de soluções $V \subseteq S$ de todas as soluções $s' \in S$, as quais podem ser obtidas por movimentos na solução s, produzidos de acordo com uma função $N(\cdot)$, ou seja, $V = \{\forall s' \in S \quad : \quad s' = N(s)\}$. Um movimento em uma

solução consiste na alteração de sua estrutura (de suas variáveis), levando esta de um ponto a outro no espaço de soluções. Assim, $N(\cdot)$ é uma função que define a forma de encaminhar uma solução de um ponto a outro no espaço de busca, definindo o conjunto de pontos que podem ser explorados (vizinhos) pela solução. Dizemos, portanto, que $N(\cdot)$ define uma estrutura de vizinhança.

3.3.2 Heurísticas de Refinamento

As heurísticas de refinamento, também conhecidas na literatura como métodos de busca local, guiam uma solução existente em direção a um ótimo local, segundo uma determinada estrutura de vizinhança (Seção 3.3.1).

Seja s a solução corrente. Por meio de uma heurística de refinamento move-se, a cada iteração, de uma solução para outra, de acordo com a vizinhança N definida, buscando soluções sucessivamente melhores. Os métodos de descida completa (*Best Improvement*), primeira melhora (*First Improvement*) e descida randômica são heurísticas clássicas de refinamento.

Dada uma solução corrente s e uma vizinhança $N(s)$, no método de descida (subida) completa, descrito no Algoritmo 1, avaliam-se todos os vizinhos s' da solução corrente e, em seguida, executa-se o movimento que leva à solução vizinha s^* que maximiza a melhora da solução corrente. A solução corrente é atualizada e o processo é repetido até que não ocorram mais melhoras, ou seja, até que a solução corrente seja um ótimo local segundo a vizinhança considerada.

Os custos computacionais para o emprego do método de descida completa podem ser muito elevados ou até mesmo proibitivos. Assim, uma alternativa pode ser a atualização da solução corrente tão logo seja encontrada uma solução vizinha que a melhore. Esta é a proposta da heurística de refinamento *First Improvement*, descrita no Algoritmo 2, na qual a exploração de toda a vizinhança de s ocorre apenas no pior caso.

Outra alternativa para não realizar a exploração completa da vizinhança $N(s)$ é o método de descida randômica. Neste método, um movimento é escolhido aleatoriamente e a solução s' resultante é avaliada. Se a solução s' for melhor que a solução corrente s, promove-se a atualização da solução corrente. Caso contrário, um outro movimento é escolhido, também aleatoriamente. O método é executado até que um número determinado de iterações sem melhora seja alcançado ou até que um número predefinido de iterações (com ou sem melhora) seja realizado.

3.3.3 *Variable Neighborhood Search* - VNS

Variable Neighborhood Search (VNS) é uma metaheurística, proposta por Mladenovic e Hansen (1997), voltada principalmente para otimização combinatória.O VNS possui uma estrutura extremamente simples, conforme descrito no Algoritmo 3, que recebe inicialmente uma solução s de partida e um conjunto NS de estruturas de vizinhança, que deseja-se explorar. Em seguida, estas vizinhanças serão utilizadas de maneira cíclica e sistêmica, para fazer com que uma solução possa caminhar no espaço de busca, conseguindo escapar de ótimos locais. Para isto, primeiramente, a solução de partida é direcionada para um ótimo local por um método de busca local. Em seguida, para tentar escapar deste mínimo local, é realizada uma "perturbação"

Algoritmo 1 *Best Improvement* (BI)

Parâmetros:
$-s$: Solução que será refinada;
$-N(.)$: Estrutura de vizinhança usada na busca;
$-f(.)$: Função objetivo;

```
1: function BEST_IMPROVEMENT(s, N(.), f(.))
2:     s* ← s;
3:     repeat
4:         s' ← arg min_{∀x∈N(s*)} f(x); #Vizinho de s* em N(.) com melhor avaliação de f(.)
5:         s'' ← s*;
6:         if f(s') < f(s*) then
7:             s* ← s';
8:         end if
9:     until f(s'') = f(s*)
10: return s*;
11: end function
```

Algoritmo 2 *First Improvement* (FI)

Parâmetros:
$-s$: Solução que será refinada;
$-N(.)$: Estrutura de vizinhança usada na busca;
$-f(.)$: Função objetivo;

```
1: function FIRST_IMPROVEMENT(s, N(.), f(.))
2:     s* ← s;
3:     repeat
4:         s' ← primeiro movimento de melhora segundo uma ordem pré-definida
5:               de todos os vizinhos de N(s*)
6:         s'' ← s*;
7:         if f(s') < f(s*) then
8:             s* ← s';
9:         end if
10:    until f(s'') = f(s*)
11: return s*;
12: end function
```

(denominada no VNS como *shake*) na solução, utilizando-se a primeira estrutura de vizinhança de NS, gerando-se, assim, uma nova solução s'. Esta "pertubação" consiste na análise do entorno da solução s, normalmente realizada por um movimento aleatório na estrutura de vizinhança. O objetivo da perturbação é tentar encaminhar a solução s' para uma bacia de atração do espaço de busca diferente daquela em que a solução s se encontra. Desse modo, em seguida, a solução s' é submetida ao processo de busca local, verificando, após este, se s' representa uma solução com melhor avaliação da função objetivo que a solução s. Caso isto ocorra, a solução s é substituída por s' e a primeira estrutura de vizinhança ($k \leftarrow 1$) é utilizada novamente para a realização da perturbação. Caso contrário, s continua sendo a solução a ser explorada, mas é realizada uma mudança na estrutura de vizinhança ($k \leftarrow k + 1$) que realizará a perturbação na solução. O processo é repetido com a perturbação da solução s, na estrutura de vizinhança N_k, seguida de busca local. O processo é finalizado quando a última estrutura de vizinhança ($kmax$) for utilizada para perturbar a solução e esta não conseguir gerar uma solução de melhora após a busca local.

Deve-se perceber que, toda vez que a solução perturbada s' não apresentar me-

Algoritmo 3 *Variable Neighborhood Search* (VNS)

Parâmetros
$-s$: Solução que será refinada pelo VNS;
$-NS$: Vetor de estruturas de vizinhança exploradas $NS \leftarrow \{N_1, N_2, ..., N_{kmax}\}$;
$-kmax$: Número de estruturas de vizinhança presentes no vetor ;

```
1: function VNS(s, NS, kmax)
2:     s* ← s;
3:     while (Critério de parada não satisfeito) do
4:         s' ← s;
5:         s'' ← NULL;
6:         k ← 1;
7:         while (k < kmax) do
8:             s'' ← shake(s', k, NS);
9:             s'' ← local_search(s'');
10:            if f(s'') < f(s') then
11:                s' ← s'';
12:                k ← 1;
13:            else
14:                k ← k + 1;
15:            end if
16:        end while
17:        if f(s') < f(s*) then
18:            s* ← s';
19:        end if
20:    end while
21: return s*;
22: end function
```

lhora, a próxima estrutura de vizinhança ($k \leftarrow k + 1$) do vetor de estruturas NS é selecionada e utilizada na próxima iteração do método. Isto é feito de forma ordenada e sistemática. Além disto, toda vez que uma solução apresente melhora, após a pertubação e busca local, retorna-se à primeira estrutura de vizinhança ($k \leftarrow 1$) e o processo será reiniciado, considerando agora a última solução de melhora. Assim, o VNS trabalha com a premissa que o mínimo local de uma estrutura de vizinhança não é necessariamente o mínimo em outra estrutura, mas o mínimo global é o mínimo local de todas as estruturas de vizinhança. Deste modo, o VNS tenta encaminhar uma solução para o mínimo local em uma estrutura. Uma vez que o método se torna incapaz de evoluir nesta estrutura, tenta-se, através da troca sistemática de estruturas de vizinhança, realizar continuamente o encaminhamento da solução para regiões "mais distantes" (ou diferentes) do espaço de busca, afim de encontrar um mínimo local de melhor qualidade que a solução atual. Caso isto ocorra, retoma-se a exploração ao redor do novo mínimo. Mas, à medida que o método não consiga encontrar novas soluções de melhora, este continuamente tentará novamente encaminhar a solução para regiões cada vez "mais distantes" do ponto atual. Assim, a troca de estruturas de vizinhança pode ser vista como uma espécie de aumento no sinal de perturbação da solução, pois, quanto maior a dificuldade de escapar de um mínimo local, maior o ruído inserido.

3.3.4 *Variable Neighborhood Descent* - VND

Variable Neighborhood Descent (VND) pode ser considerada como uma versão estática e determinística da metaheurística VNS (Mladenovic e Hansen, 1997). É um método de refinamento de soluções, que, assim como o VNS, realiza a busca no espaço de soluções, alternando, de modo cíclico e sistemático, as estruturas de vizinhança a serem consideradas.

Considere um conjunto $NS = \{N_1(s), N_2(s), \ldots, N_{kmax}\}$, composto por diferentes estruturas de vizinhança. A versão padrão do procedimento VND começa pela exploração da vizinhança $N_1(s)$, de acordo com uma estratégia de busca definida para esta estrutura. Finalizando a exploração de $N_1(s)$, passa-se à exploração da vizinhança $N_2(s)$. Se, ao final da exploração de $N_2(s)$ é observada melhoria, a exploração da vizinhança $N_1(s)$ é retomada. Caso contrário, passa-se a explorar a próxima vizinhança. Este processo de alternância cíclica e sistemática de vizinhanças continua até que a k-ésima estrutura de vizinhança seja explorada sem melhora na solução corrente ou até que algum critério de parada seja satisfeito.

Mjirda et al. (2017) identifica que a ordem em que as estruturas de vizinhança são exploradas pode influenciar no desempenho do algoritmo. Testar todas as ordens possíveis apresenta um custo computacional inviável. Na literatura, grande parte dos trabalhos utilizam a ordem crescente de cardinalidade de possíveis permutações a serem realizadas na exploração de uma estrutura de vizinha. Outra característica que pode provocar impacto na resposta do VND é a estratégia de busca local utilizada na exploração da estrutura de vizinhança. Estas estratégias podem ser *Best Improvement* ou *First Improvement*

Algoritmo 4 *Variable Neighborhood Descent* (VND)

Parâmetros
$-s$: Solução que será refinada pelo VND;
$-NS$: Estruturas de vizinhança exploradas $NS \leftarrow \{N_1, N_2, ..., N_{kmax}\}$;

1: **function** VND(s, NS)
2: $s' \leftarrow NULL$;
3: $s^* \leftarrow s$;
4: $NS' \leftarrow NS$;
5: $k \leftarrow 1$;
6: **while** ($NS' \neq \{\}$) **do**
7: $k \leftarrow choose\_neighborhood(NS')$;
8: $s' \leftarrow \arg \min_{\forall x \in NS'[k](s^*)} f(x)$; #Melhor vizinho x presente na vizinhança $NS'[k](s^*)$
9: **if** $f(s') < f(s^*)$ **then**
10: $s^* \leftarrow s'$;
11: $NS' \leftarrow NS$;
12: **else**
13: $remover(NS', k)$;
14: **end if**
15: **end while**
16: **return** s^*;
17: **end function**

3.3.4.1 *Basic-VND*

O *Basic-VND* é a versão padrão do VND, na qual as estruturas de vizinhança são exploradas pela ordem previamente definida. Caso a aplicação de alguma estrutura acarrete em melhora na solução, reinicia-se a busca a partir da primeira estrutura. No Algoritmo 4, a função $choose\_neighborhood(NS')$ (Linha:7) sempre retorna 1, assim sempre ira explorar a estrutura de vizinhança presente na primeira posição do conjunto NS'. O conjunto NS' começa com todas as estruturas que deseja-se explorar, ordenadas pelo critério de ordem adotado, ou seja, $NS' = NS$. Caso a aplicação da primeira estrutura de vizinhança presente em NS' não apresente uma solução de melhora, o algoritmo remove esta estrutura do conjunto NS', e o processo de busca é retomado com a próxima estrutura presente em NS'. Caso contrário, encontrada uma solução de melhora, NS' é reiniciado com todas as estruturas de vizinhança, i.e. $NS' \leftarrow NS$, e volta-se a exploração da primeira estrutura de vizinhança. O algoritmo para quando todas as estruturas tiverem sido exploradas sem melhora, ou seja, $NS' = \{\}$.

3.3.4.2 *Random-VND*

A única diferença do *Random-VND* para o *Basic-VND* está no comportamento da função $choose\_neighborhood(NS')$, presente no Algoritmo 4, responsável pela escolha da estrutura de vizinhança a ser explorada a cada iteração. No *Random-VND*, a função $choose\_neighborhood(NS')$ seleciona em NS', de maneira aleatória com probabilidade uniforme, a estrutura de vizinhança que será utilizada no processo de busca (Kramer e Subramanian, 2019). O restante do Algoritmo 4 se comporta estritamente igual a versão *Basic-VND*. Assim, o *Random-VND* explora as estruturas de vizinhança em ordem randômica, em contrapartida, o *Basic-VND* utiliza ordem determinística de exploração.

3.3.4.3 *Union-VND*

Na versão *Union-VND*, proposta por Lü et al. (2011), todas as estruturas de vizinhança são exploradas individualmente e os seus resultados são comparados entre si. A estrutura efetivada é aquela que apresenta a permutação acarretando na melhor melhora da solução. O procedimento é repetido, até que a aplicação de nenhuma estrutura acarrete em melhora da solução. Este método vem apresentando bom desempenho na literatura, mas requer alto custo computacional, já que explora todas as estruturas para efetivar apenas uma alteração (Mjirda et al., 2017).

3.3.5 *Iterated Greedy Search* - IGS

Iterated Greedy Search (ou *Iterated Greedy*) é uma metaheurística de refinamento muito aderente a problemas de sequenciamento. O IGS foi proposto por Ruiz e Stützle (2007) para um problema de sequenciamento em ambientes *flowshop*. Para ambientes *flowshop*, este algoritmo vem se mostrando muito eficiente. Na revisão bibliográfica proposta por Fernandez-Viagas et al. (2017), verificou-se que metaheurísticas que incorporam características do IGS em seu funcionamento tendem a apresentar desempenho médio superior às que não incorporam. Já para ambientes de máquinas

paralelas, o IGS também tem apresentado bom desempenho, tendo características incorporadas a algumas metaheurísticas destinadas a soluções de problemas com características de máquinas paralelas não relacionadas (Fanjul-Peyro e Ruiz, 2010; Lin et al., 2011; Ying e Lin, 2012; Diana e de Souza, 2020).

O IGS é baseado em heurísticas gulosas para a construção de soluções. Métodos gulosos se baseiam na ideia de construir iterativamente uma solução parcial s'', a partir de uma lista de objetos Lo, não presentes em s''. Para isto, deve existir uma função heurística $H(s'', o)$, a qual avalia o benefício imediato da inserção de cada objeto $o \in Lo$ na solução parcial s''. Assim, um algoritmo guloso consiste em construir uma solução de forma iterativa, de modo que, a cada iteração, é inserida, na solução parcial s'', o objeto em Lo que apresenta melhor benefício imediato para a solução, dado por $H(s'', o)$. Além disto, o objeto o selecionado e inserido em s'' deve ser removido da lista Lo. O processo é repetido até que Lo se torne vazia, indicando que a solução está completamente construída e deve ser retornada. Dois pontos devem ser observados: a solução s'', no instante de entrada no algoritmo, não necessariamente é uma solução vazia, podendo ser uma solução parcialmente construída anteriormente por outro método. Além disto, o ponto de inserção de um objeto o na solução s'' não necessariamente é realizada ao final da solução, como comumente é observado em algoritmos gulosos, pois o pode ser inserido em qualquer ponto, sendo definido pela função heurística $H(s'', o)$. Um pseudo código deste algoritmo de construção gulosa é mostrado no Algoritmo 5.

Algoritmo 5 *Construção Gulosa*

Parâmetros
$-s''$: Solução de partida a ser completada;
$-Lo$: Lista de objetos faltantes em s'';

```
1: function CONSTRUCTION(s'', Lo)
2:     o ← NULL;
3:     while Lo <> {} do
4:         (s'', o) ← arg max H(s'', o_i), sujeito à o_i ∈ Lo;
5:         Lo ← remover objeto o da lista Lo;
6:     end while
7:     return s'';
8: end function
```

Este algoritmo iterativo de construção de soluções é base fundamental para a metaheurística IGS, assim como o algoritmo de desconstrução de uma solução. O método de desconstrução no IGS (Algoritmo 6) tem a finalidade de remover $size_{Lo}$ objetos de uma solução s', retornar a solução modificada s'' e os objetos removidos em uma lista Lo. Assim, como entrada, o algoritmo recebe uma solução s' e o número e objetos ($size_{Lo}$) a serem removidos. O parâmetro $size_{Lo}$ indica o nível de perturbação que uma determinada solução irá sofrer. A maior parte dos estudos encontrados na literatura utilizam o sorteio aleatório com probabilidade uniforme como critério para seleção dos objetos que serão removidos da solução s. Em casos específicos, está seleção pode se dar em algum critério específico. Uma vez selecionados os $size_{Lo}$ objetos a serem removidos, estes são extraídos da solução s', gerando uma solução parcial s'' e uma lista Lo de objetos removidos. Ambos, s'' e Lo, são retornados pelo métodos.

O Algoritmo 7 apresenta um pseudo código do IGS. Deve-se observar que a es-

Algoritmo 6 *Fase Destrutiva IGS*

Parâmetros
$-s'$: Solução parcial a ser completada;
$-size_{Lo}$: Número de objetos a serem removidos de s';

```
1: function DESTRUCTION(s', size_Lo))
2:     s'' ← s';
3:     Lo ← {};
4:     o ← NULL;
5:     while |Lo| < size_Lo do
6:         o ← seleciona objeto em s'';
7:         s'' ← remover objeto o da solução s'';
8:         Lo ← Lo ∪ o;
9:     end while
10:    return (s'', Lo);
11: end function
```

trutura do algoritmo é bastante similar a métodos de trajetória, como ILS e VNS, consistindo em explorar uma solução s de forma iterativa. Inicialmente, a solução de entrada s é atribuída às soluções s' e s^*, sendo, respectivamente, a solução explorada e a melhor solução conhecida pelo processo de busca. A cada iteração, s' é "desconstruída" pelo método de desconstrução (Algoritmo 6), gerando uma solução parcial s''. Posteriormente, s'' é reconstruída pelo método guloso de construção de soluções apresentado no Algoritmo 5. Caso s'' seja mais eficiente que a melhor solução s^*, s'' se torna s^*. Ainda, verifica-se se s'' deve substituir s'. Na versão padrão do IGS, s'' substitui s' quando s'' é mais eficiente que s', mas este operador também pode ter seu comportamento alterado para aceitar soluções de piora. Este mecanismo possibilita o incremento de diversidade no processo de busca, principalmente quando s' for uma solução que represente um mínimo local. Como pode-se observar, a principal diferença do IGS para metaheurísticas de trajetória como VNS e ILS se encontra na forma como a solução é explorada, sendo no IGS realizada pelos métodos de desconstrução e reconstrução gulosa de uma solução.

3.4 Sistemas Imunológicos Artificiais

Sistemas Imunológicos Artificiais (SIAs) são compostos por metodologias inteligentes, inspiradas no Sistema Imunológico Natural (SIN), para soluções de problemas do mundo real (Dasgupta, 1998). Grande parte dos trabalhos pioneiros relacionados a SIAs foram aplicados a área de aprendizagem de máquina, tentando extrair características do SIN para o reconhecimento de padrões. O algoritmo apresentado por Hunt e Cooke (1996) extraiu características do SIN para o reconhecimento de sequências de DNA, sendo que o método proposto alcançou resultados similares aos melhores métodos da época. Em sequencia, Hunt e Fellows (1996) apresentam uma abordagem para *data mining*. Em 1999, a editora *Springer* lançou um livro intitulado *Artificial Immune Systems and Their Applications*, o qual compilava em capítulos diversas abordagens computacionais imunoinspiradas. Dentre estas, Dasgupta e Forrest (1999) propõem um algoritmo para detecção de anomalias baseado no princípio da *Seleção Negativa*. Já Fukuda et al. (1999b) propõem um método de multiagentes baseado na *Teoria da Rede Imunológica* para o controle de sistemas dinâmicos

Algoritmo 7 *Iterated Greedy Search* (IGS)

Parâmetros
$-s$: Solução de partida exploradas pelo IGS;
$size_{Lo}$: Número de objetos removidos de uma solução a cada iteração.

```
1: function IGS(s, size_Lo)
2:     s* ← s;
3:     s' ← s;
4:     s'' ← NULL;
5:     Lo ← NULL;
6:     while (Critério de parada não satisfeito) do
7:         (s'', Lo) ← destruction(s', size_Lo);
8:         s'' ← construction(s'', Lo);
9:         if f(s'') < f(s*) then
10:            s* ← s'';
11:        end if
12:        if aceitaSolucao(s', s'') then
13:            s' ← s'';
14:        end if
15:    end while
16:    return s*;
17: end function
```

utilizados na produção de semicondutores. Watanabe et al. (1999) apresentam um modelo para detecção de fraudes em hipotecas.

Devido aos resultados satisfatórios encontrados pelos primeiros trabalhos direcionados ao reconhecimento de padrões, pesquisadores de outras áreas, como análise de dados, agrupamento de dados e otimização (objetivo de estudo do presente trabalho) passaram a pesquisar como capturar características do SIN para a concepção de um SIA. Os SIAs propostos para problemas de otimização, em sua grande maioria, são algoritmos de otimização multimodal, sendo baseados no *Princípio da Seleção Clonal* e *Teoria da Rede Imunológica*. Algoritmos de *Células Dendríticas* baseadas na *Danger Theory* (Greensmith e Aickelin, 2008) não serão abordados neste trabalho, uma vez que são baseados em comportamentos do SIN não relacionados a esta pesquisa.

A primeira utilização encontrada de um SIA aplicado a um problema de otimização foi proposta por Hart et al. (1998), para a solução de um problema de sequenciamento. Neste trabalho é abordado um problema do tipo *Jobshop* dinâmico, em que tarefas chegam continuamente e datas de entrega podem ser alteradas por razões práticas. O sequenciamento representa o anticorpo e o antígeno contém as restrições que o conjunto de tarefas deve atender. Assim, quando uma solução é apresentada ao antígeno, esta deve se adequar às restrições por meio de operações de hipermutações somáticas. Alterações no antígeno forçam uma evolução do anticorpo (solução) para atender à nova configuração. Fukuda et al. (1999a) propõem uma abordagem para otimização de funções continuas, sendo, nesta proposta, o primeiro desenho geral de um algoritmo baseado no princípio da seleção clonal (Seção 3.4.1.1), mas que ainda incorpora características de algoritmos genéticos em sua estrutura. Utilizando o princípio da seleção clonal, de Castro e Von Zuben (2000) apresentam o CLONALG, proposto para problemas de otimização contínua e combinatória. Este algoritmo apresenta um operador de hipermutação inversamente proporcional à afinidade. Ainda utilizando o princípio da seleção clonal, Kelsey e Timmis (2003) apresentam o al-

goritmo B-Cell, algoritmo derivado do (1+1) EA(*Evolutionary Algorithm*), obtendo desempenho superior a este (Doerr et al., 2016).

Baseado na teoria da rede imunológica, de Castro e Timmis (2002b) propõem um algoritmo denominado opt-aiNet para otimização contínua multimodal. Esta abordagem é uma extensão da rede imunológica artificial para análise de dados aiNet (de Castro e Zuben, 2002). O opt-aiNet consiste basicamente em gerar um número de soluções aleatórias, evoluir estas soluções através dos mecanismos de microevolução do CLONALG, e, em seguida, suprimir a população resultante, retirando da população células que estejam próximas no espaço de busca, mantendo apenas a melhor solução da região (teoria da rede imunológica). Novas células são acrescentadas à população e o processo é repetido. Esta abordagem apresentou inúmeros casos de sucesso para problemas de otimização multimodal e, desde então, diversas extensões foram propostas. Para otimização combinatória, destacam-se o copt-aiNet (de Sousa et al., 2004), adaptação do opt-aiNet para resolução de um problema de alinhamento genético. Já o *Bayesian Artificial Immune System* (de Castro e Zuben, 2009) é uma extensão que incorpora características de modelos gráficos probabilísticos ao opt-aiNet. Coelho et al. (2011) expandem o copt-aiNet, incorporando características de concentração de anticorpos, e esta característica também é incorporada para abordagens para otimização contínua multimodal (Coelho e Von Zuben, 2010) e multiobjetivo (Coelho e Von Zuben, 2011). Já de França et al. (2005) propõem uma extensão do opt-aiNet para otimização em ambientes dinâmicos. Para otimização multiobjetivo, são propostas as abordagens omni-aiNet (Coelho e Von Zuben, 2006), para espaços de busca contínuos, e MOBAIS (de Castro e Von Zuben, 2008), para otimização combinatória. Como pode-se perceber, o algoritmo opt-aiNet apresenta grande importância na aplicação de sistemas imunológicos artificiais a problemas de otimização, este sendo um dos principais focos de estudo deste trabalho.

O restante desta seção apresenta uma breve introdução do SIN, detalhando tanto o princípio da seleção clonal quanto a teoria da rede imunológica. Em seguida, será feita uma introdução aos SIAs, apresentando uma definição básica dos componentes utilizados nos algoritmos imunoinspirados para otimização. Será abordado o algoritmo CLONALG, detalhando seus componentes, o modelo de rede imunológica para otimização em espaços contínuos (opt-aiNet), sendo apresentadas suas diferenças para o CLONALG. Também será apresentado a variação do opt-aiNet para otimização combinatória (copt-aiNet). Por fim, será apresentado um operador denominado *ageing*, utilizado comumente em algoritmos imunoinspirados para incremento da diversidade dos mesmos.

3.4.1 Sistema Imunológico Natural

Os vertebrados são expostos continuamente a patógenos, que são organismos com vírus, bactérias, protozoários capazes de desenvolverem doenças infecciosas em seus hospedeiros. Para evitar a contração de enfermidades, os vertebrados possuem um sistema denominado Sistema Imunológico, ou Sistema Imune, composto por um conjunto de estruturas biológicas e químicas, que atuam de maneira auto organizada e colaborativa, com o intuito de eliminar os patógenos, coibindo o seu desenvolvimento no organismo.

O sistema imunológico é composto por múltiplas camadas e subsistemas inde-

pendentes, que interagem e complementam uns aos outros. Dentre tais camadas e subsistemas, podem ser citadas barreiras físicas e bioquímicas, as quais podem ser observadas tanto em vertebrados quanto em alguns seres primitivos como invertebrados multicelulares, plantas e fungos (Janeway et al., 2005), além de dois grandes subsistemas, denominados sistema imune inato e sistema imune-adaptativo (de Castro, 2001).

O sistema imune inato, em geral, é responsável por promover a primeira resposta a um agente patogênico após a transposição deste às barreiras físicas e bioquímicas. As células denominadas *fagócitos* são componentes do sistema imune inato, as quais trabalham de maneira independente e são responsáveis por identificar e eliminar os patógenos através de *fagocitose*. Os *fagócitos* percorrem o corpo através do sistema circulatório e, quando um patógeno entra em contato com um *fagócito*, o primeiro é englobado pela membrana plasmática do *fagócito*, sendo posteriormente digerida e incorporada ao citoplasma da célula. Este processo é denominado *fagocitose*, sendo utilizado por seres celulares (como protozoários) como mecanismo de alimentação. Já no sistema imune dos vertebrados é utilizado para a eliminação de patógenos ou células infectadas pelos mesmos (May e Machesky, 2001). Outra função do sistema imune inato é apresentar, através dos *fagócitos* denominados *células dendríticas*, material antígeno ao sistema imune adaptativo, permitindo a este ativar a resposta imune adaptativa. Material antígeno são as moléculas e substâncias patogênicas *não próprias*, ou seja, estranhas ao organismo, que, ao serem reconhecidos pelo sistema imune, geralmente disparam uma resposta imune. *Células dendríticas*, ao realizar a *fagocitose* de patógenos, metabolizam o seu material antígeno, digerindo este em *peptídeos* que, posteriormente, são apresentados na superfície da membrana celular externa como um complexo protéico, denominado Complexo Principal de Histocompatibilidade (MHC). Estas células são encaminhadas para os gânglios linfáticos, de modo que o MHC, ao se ligar ao receptor de *linfócitos T* auxiliares (reconhecimento antígeno), ativa a resposta imune adaptativa e dispara o processo de expansão clonal. Deve-se destacar que este não é o único mecanismo de ativação do sistema imune adaptativo, podendo ser ativado também pelo contato direto do material antígeno com o receptor dos *linfócitos B*.

Os *fagócitos* que constituem o sistema imune inato são células que já estão presentes no organismo dos vertebrados desde o nascimento dos mesmos. Estas células não atuam de maneira adaptativa, não sendo capazes de armazenar informações de patógenos previamente apresentados ao sistema. Em contrapartida, o sistema imune adaptativo é composto por células denominadas *linfócitos*, que possuem capacidade de identificar material antígeno específico e adaptar seus receptores a estes, possibilitando uma resposta personalizada. Além disto, uma vez que os *linfócitos* sejam adaptados ao reconhecimento de um antígeno específico, são mantidos no sistema, mesmo após a interrupção da resposta imune. Desta forma, o sistema imune adaptativo possui uma espécie de memória imunológica, que possibilita uma resposta mais rápida a um determinado patógeno, desde que este já tenha sido previamente apresentado ao organismo do vertebrado.

Devido a este trabalho se concentrar em metodologias que tentam extrair o poder de auto organização, adaptação e memorização do sistema imune adaptativo através da ativação do mesmo por *linfócitos B*, o restante desta seção será destinada a detalhar dois componentes deste sistema. O primeiro descreve o processo que permite a

adaptação da superfície dos *linfócitos B* à identificação de antígenos específicos, sendo denominado princípio da seleção clonal, e é apresentado na Seção 3.4.1.1. Já a teoria da rede imunológica, exposta na Seção 3.4.1.2, descreve um possível comportamento geral do sistema imune adaptativo. Detalhes referentes ao sistema imune inato não serão detalhados aqui, já que este não é o foco deste trabalho. Assim, para fins de simplificação, a ativação do princípio da seleção clonal será explicada apenas pelo ligamento de material antígeno diretamente ao receptor dos *linfócitos B*, não detalhando a ativação por *linfócitos T* estimulados por células *dendríticas* apresentadoras de antígenos.

3.4.1.1 Princípio da Seleção Clonal

O princípio da seleção clonal é um processo desencadeado pela resposta imune adaptativa, acionada quando patógenos (micro-organismos específico possíveis provocadores de doenças) são reconhecidos pelas células de defesa, denominadas *linfócitos*, presentes no sistema imune adaptativo. A superfície de um patógeno é composta por um conjunto de antígenos, os quais são, em geral, moléculas protéicas estranhas ao organismo. Os antígenos possuem *epítopos* que, ao entrarem em contato com o receptor de *linfócitos*, podem ativar a resposta imune adaptativa.

Os *linfócitos* são divididos em três tipos, com características e comportamentos distintos, denominados células NK (*Natural Killers*), *linfócitos T* (ou "células T"), *linfócitos B* (ou "células B"). Os dois últimos, no entanto, expressam, em suas superfícies, um receptor altamente específico para a identificação de um determinado antígeno (de Castro, 2001). Cada célula B possui apenas um receptor, denominado BCR (*B-Cell Receptor*). Este, ao entrar em contato com um *epítopo* de um antígeno, desencadeia uma série de reações químicas. Caso as forças atrativas superem as repulsivas, o *linfócito B* ativa a expansão clonal da célula. Em decorrência disto, a célula se multiplica continuamente por sucessivas *mitoses*, gerando um grande número de clones, divididos posteriormente em dois tipos: *plasmócitos* e células de memória.

Plasmócitos são os *linfócitos B* responsáveis pela produção e secreção de grande quantidade de anticorpos, os quais possuem receptores que se ligaram aos *epítopos* do *antígeno* identificado. Segundo Ravetch e Bolland (2001), os anticorpos podem contribuir com a resposta imune de três formas: (i) impedindo que os patógenos danifiquem células ao unir-se a elas (neutralização); (ii) estimulando a eliminação do patógeno pelos *macrófagos* e outras células, recobrindo o patógeno (opsonização); e (iii) desencadeando a destruição direta do patógeno, estimulando outras respostas imunes (*lise*). Já as células de memória são clones que irão circular pelos vasos linfáticos e tecidos durante longo período de tempo. Ao serem expostos novamente a um antígeno previamente identificado, reativarão a expansão clonal, gerando novos plasmócitos e células de memória. A Figura 3.1 ilustra o princípio da seleção clonal.

Durante o processo de expansão clonal, os receptores das células B sofrem pequenas mutações aleatórias, as quais podem provocar, ocasionalmente, a melhora na "afinidade" do BCR ao estímulo antígeno que ativou a expansão clonal (de Castro, 2001). Este fenômeno, denominado hipermutação somática, promove a maturação da afinidade célula/antígeno, possibilitando que *linfócitos B* sejam capazes de se adaptar a diferentes *epítopos* antigênicos apresentados ao SIN, permitindo que um individuo crie imunidade a um grande número de patógenos ao longo de sua vida. A hiper-

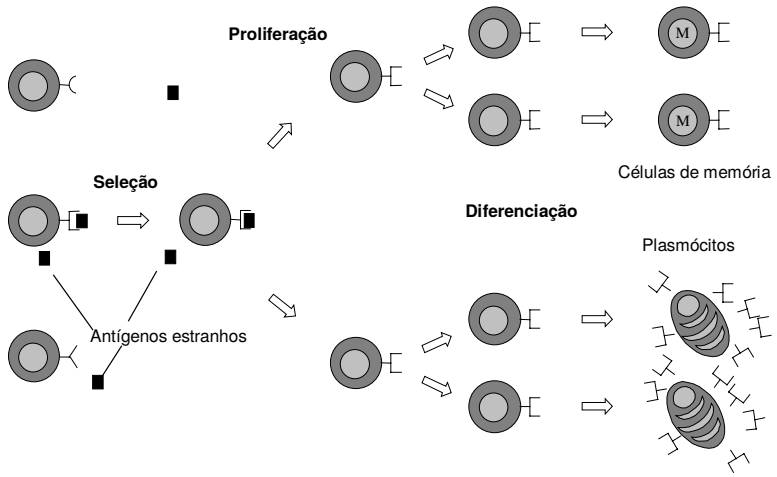


Figura 3.1: Ilustração do princípio da seleção clonal. Fonte: de Castro (2001).

mutação somática também permite que o sistema imune adaptativo possa apresentar uma resposta, eficiente e eficaz, a antígenos ligeiramente diferentes aos previamente identificados, reduzindo, assim, tanto o tempo de ativação quanto de execução da resposta imune. Além disto, células com alta afinidade célula/antígeno se multiplicaram em maior escala. Assim, consequentemente, pelo processo de seleção natural darwiniano, estas terão maior probabilidade de sobreviverem à "contração" clonal e, assim, células adaptadas com "forte" adaptação a antígenos apresentam maior probabilidade de integrar o repertório das células que compõem a memória imunitária.

3.4.1.2 Teoria da Rede Imunológica

Os avanços na área de imunologia acarretados pelo princípio da seleção clonal possibilitaram o desenvolvimento, por Jerne (1974), da teoria da rede imunológica. Esta teoria tenta descrever o comportamento global do sistema imunológico adaptativo dos vertebrados, levantando a hipótese de que *linfócitos B*, além de serem responsáveis durante a resposta imunológica por secretar anticorpos contra os antígenos relacionados aos patógenos, também possuem um mecanismo de supressão celular. Durante a resposta imune, este mecanismo é responsável por conter a expansão clonal, além de promover, devido à supressão celular entre *linfócitos B*, uma contínua redução na resposta imune à medida em que ocorrer a diminuição na identificação de antígenos. Este comportamento possibilita que o sistema retorne a um estado de equilíbrio após a eliminação dos patógenos, até que ocorra um novo distúrbio no sistema, ativando uma nova resposta imune.

Segundo a teoria da rede imunológica, o princípio da seleção clonal se comportaria com uma espécie de seleção positiva, na qual o contato da estrutura dos receptores de *linfócitos* com um antígeno de um patógeno desencadearia uma resposta imune, composta por uma expansão clonal e secreção de anticorpos para identificação do antígeno e posterior eliminação do patógeno. Por outro lado, pela teoria da rede imunológica, receptores dos *linfócitos B* também são capazes de reconhecer outros *linfócitos B* através de estruturas denominadas *idiotopos* (Merelli et al., 2015). Assim,

uma vez que o receptor de um *linfócito B* entrar em contato com um *idiotopo* de outro *linfócito B*, e ambos apresentarem alta afinidade, é realizada a supressão do *linfócito B* cujo *idiotopo* foi identificado. Este processo, denominado seleção negativa, é ilustrado na Figura 3.2.

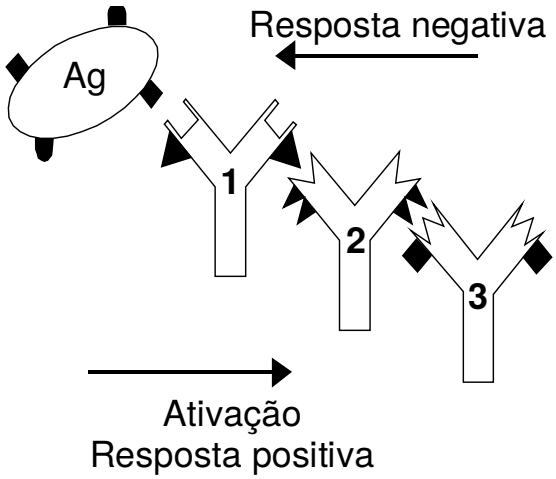


Figura 3.2: Ilustração dos mecanismos de identificação de *linfócito B* por *linfócito B* através dos *idiotopos*. Fonte: de Castro (2001)

Deve-se observar que o processo de supressão será mais presente em uma expansão clonal deflagrado por uma resposta imune. Os *linfócitos B* que estão se expandindo tendem a ter descendentes células de memória comuns, já que *linfócitos B* mais afins ao antígeno tendem a ter uma maior concentração. Assim, os *idiotopos* e as estruturas dos receptores dos *linfócitos B* tendem a ser similares, acarretando na auto regulação das populações de *linfócitos B* do sistema após redução na resposta imune. Ao se estabilizar, o sistema imunológico tende a manter na população células com característica diferente na estrutura dos receptores, não permitindo uma concentração a longo prazo de *linfócitos* com estruturas semelhantes.

A partir de seu lançamento, a teoria da rede imunológica foi amplamente aceita por pesquisadores em Imunologia. Devido a suas contribuições na teoria de seleção clonal e na teoria da rede imunológica, Jerne recebeu o Premio Nobel de Medicina e Psicologia em 1984. No inicio dos anos 90, principalmente devido à falta de evidências empíricas, a teoria da rede imunológica começou a ser questionada (Coelho, 2011). Apesar disto, a teoria da rede imunológica continua apresentando relevância nas áreas relacionadas à Imunologia. Esta teoria tem sido utilizada, por exemplo, para entender o comportamento do patógeno do HIV (*Human Immunodeficiency Virus*) (Douek et al., 2002), sendo, a partir deste entendimento, derivada uma proposta conceitual para uma vacina do HIV baseado na teoria da rede imunológica (Hoffmann et al., 2012).

Apesar de inúmeras abordagens computacionais desenvolvidas sobre características específicas do SIN, a teoria da rede imunológica, em conjunto com o princípio da seleção clonal, tem se destacado ao longo dos últimos anos em aplicações computacionais imunoinspiradas. As abordagens propostas na literatura consideram principalmente os aspectos de auto-organização, autoregulação do tamanho da população,

manutenção da diversidade e os mecanismos de aprendizagem. Os principais detalhes da modelagem de aspectos da teoria da rede imunológica e princípio da seleção clonal em ferramentas computacionais serão apresentas nas próximas seções.

3.4.2 Sistemas Imunológicos Artificiais

A pesquisa em computação natural, dentre outros objetivos, tenta extrair e simular características de processos naturais para resolução de problemas computacionais complexos, como problemas NP-Difíceis ou problemas de aprendizagem, para os quais técnicas convencionais e modelos matemáticos não apresentam soluções satisfatórias, seja pelo elevado custo computacional, seja pela complexidade de modelagem do problema. Como pode ser percebido, a resposta imune adaptativa apresenta diversos mecanismos sistêmicos, os quais permitem ao organismo se adaptar a patógenos desconhecidos previamente e memorizar elementos da resposta, permitindo que, em um contato futuro com patógenos similares, o sistema dispare uma resposta imune mais eficiente. Os SIN apresentam uma série de características e comportamentos, que podem ser modelados em ferramentas computacionais. de Castro (2009) destaca, dentre outras:

- **Reconhecimento de padrões**: os receptores dos *linfócitos* são capazes de reconhecer antígenos, não sendo específicos para um antígeno, se adaptando ao reconhecimento de padrões antígenos;
- **Detecção de anomalia**: o sistema imunológico pode reconhecer e reagir a agentes infecciosos sem que o organismo nunca tenha tido contato com tais agentes;
- **Aprendizado e memória**: como descrito anteriormente, através do princípio da seleção clonal, *linfócitos* se adaptam ao reconhecimento de antígenos, o que caracteriza um processo de aprendizagem. Além disto, este aprendizado é mantido no sistema para reconhecimento futuro de padrões antígenos similares, gerando uma espécie de memória imunológica;
- **Descentralização**: os *linfócitos* responsáveis pela resposta imune atuam em conjunto, mas de forma descentralizada, não tendo um elemento central que controle a resposta;
- **Auto-organização**: uma vez desencadeada a resposta imune, os *linfócitos* que entrarem em contato com as estruturas dos patógenos e apresentarem alta afinidade com estas se multiplicaram, afim da identificação e posterior eliminação do mesmo. Este processo de resposta é descentralizado e com características dinâmicas e adaptáveis.

Diversas ferramentas computacionais, para diversos fins, têm sido propostas baseadas nestas características do SIN. Estas ferramentas são nomeadas como Sistemas Imunológicos Artificiais. No contexto de Otimização, uma série de algoritmos baseados nos SIN são propostos, a maior parte deles inspirados no princípio da seleção clonal. Muitas vezes estas abordagens são consideradas, erroneamente, extensões de algoritmos evolutivos, como Algoritmo Genético e Estratégia Evolutiva (Gomes,

2006). Em geral, os SIAs são destinados a otimização multimodal, apresentando estratégias de busca paralela e distribuída no espaço de busca, estando presente simultaneamente em várias regiões do espaço de busca. Em contrapartida, algoritmos evolutivos tendem a utilizar estratégias elitistas de busca, as quais podem gerar perda de diversidade, o que pode ser compensado por estratégias baseadas em *modelos de ilhas* (Whitley et al., 1998) e *ninchos* (Miller e Shaw, 1996).

Grande parte das abordagens para otimização baseadas no princípio da seleção clonal utilizam populações de soluções candidatas. Cada indivíduo da população é considerado um *linfócito B*, o qual deve apresentar alguma representação e função de avaliação, esta representando a ligação entre o receptor do *linfócito* e o antígeno, sendo chamada de função de afinidade célula/antígeno. Em seguida, é realizado um processo de expansão clonal, no qual soluções que apresentarem maior afinidade célula antígeno criam maiores concentrações de clones. O processo de expansão clonal desencadeia um série de mutações, as quais permitem que as soluções explorem o espaço de busca. Após esta fase, algum método de seleção de células deve ser utilizado para que a população possa ser reduzida e uma nova geração de células possa continuar a exploração do espaço de busca.

A partir desta notação básica de algoritmos baseados no princípio da seleção clonal, pode-se extrair vários elementos básicos para a construção de um algoritmo, sendo estes:

- Célula (c_p): representação computacional de uma solução candidata a resolução do problema;
- População de células (P_u): representa a população de *linfócitos B* (células) presentes no sistema, cada célula representando uma solução candidata;
- Afinidade célula/antígeno (f_{Ag}): o sistema deve apresentar alguma função de avaliação que retorne o nível de ligamento entre uma célula e um antígeno;
- Operador de clonagem: operador que define o número de clones que devem ser gerados para cada célula;
- Operador de hipermutação somática: operador que realiza as alterações nos clones de modo a permitir que estes explorem o espaço de soluções;
- Operador de seleção: operador que seleciona quais células irão compor a população de células da próxima geração.

3.4.2.1 Espaços de formas

A partir da extração de elementos básicos de um SIA baseado no princípio da seleção clonal, descrita na Seção 3.4.1.1, observa-se que, para a transposição da metáfora para um modelo computacional, é necessária a definição da representação matemática das células e antígenos, além da definição de métricas de avaliação da ligação entre os receptores de uma célula e os epítopos dos antígenos. O formalismo proposto para esta representação matemática é denominada na literatura referente a SIAs como espaços de formas (Gomes, 2006).

Pelo princípio da seleção clonal, *linfócitos B* possuem receptores (BCR) que, ao entrarem em contato com os epítopos de antígenos, produzem reações físico-químicas.

Uma resposta imune será ativada caso as forças atrativas intermoleculares, geradas pelo contato do BCR com os epítopos da estruturas dos antígenos, superem as forças repulsivas. Deve-se observar que a conexão entre BCR e epítopos antígenos não precisa ser perfeita, mas quanto maior a "similaridade", maior a probabilidade de ativação da resposta imune.

Traduzindo esta metáfora para uma representação computacional voltada para problemas de otimização, uma célula é comumente representada por uma possível solução do problema. Uma célula pode ser composta por uma série de informações, como o valor da função objetivo, idade (caso de utilização de operadores de *ageing*, descritos na Seção 3.4.2.5), dentre outros. A representação matemática da solução candidata, ou seja, os valores das variáveis da solução, é uma das principais informações contidas em uma célula. Esta representação é formalmente definida com os BCR das células B.

A definição de um antígeno, em SIAs voltados para a otimização, não é trivial como em problemas de reconhecimento de padrões, uma vez que em um problema de otimização não há um objeto a ser identificado, mas uma função objetivo a ser otimizada. Considerando isto, de Castro e Timmis (2002a) destacam que não há uma representação clara de um antígeno, mas é possível interpretar a afinidade célula/antígeno como a qualidade da solução contida em uma determinada célula em relação ao ambiente no qual esta está inserida. Para problemas de otimização, convenciona-se representar a função de afinidade célula/antígeno (f_{Ag}) em termos do valor da função objetivo, de uma determinada célula, em relação às demais células do sistema. Este conceito é muito similar às função de *fitness* utilizadas em algoritmos evolutivos. Entretanto, como será mostrado no Capítulo 5, para algoritmos baseados na teoria da rede imunológica, a função f_{Ag}, baseada apenas na função objetivo, pode acarretar em problemas de convergência do algoritmo.

Como neste trabalho serão explorados algoritmos baseados tanto no princípio da seleção clonal quanto na teoria da rede imunológica, torna-se necessária, assim, a definição formal de métricas da afinidade entre células. Como detalhado anteriormente, pela teoria da rede imunológica, células contém *idiotopos*, os quais também podem ser ligados a receptores de outras células B, acarretando na eliminação da célula cujo o *idiotopo* foi identificado. Assim, para problemas de otimização, métricas de similaridade entre soluções são comumente utilizados para medir o grau de afinidade entre *idiotopo* e receptores de células. O grau de similaridade entre soluções representa afinidade célula/célula (f_{Ab}), sendo avaliado por alguma métrica de distância entre soluções no espaço de variáveis, podendo ser definida, por exemplo, pela *distância euclidiana* e pela *distância de hamming*, respectivamente, para espaços de busca contínuos e binários.

3.4.2.2 *CLONal selection ALGorithm* - CLONALG

O algoritmo CLONALG (*CLONal selection ALGorithm*), proposto por de Castro e Von Zuben (2000), é baseado no princípio da seleção clonal apresentado na Seção 3.4.1.1. São aspectos do algoritmo:

(i) a criação de novos *linfócitos*, no algoritmo denominados como anticorpos;

(ii) a expansão clonal das células;

(iii) os elementos de hipermutação somática inversamente proporcional à afinidade célula/antígeno (*fitness*); e

(iv) a manutenção da memória imunológica, por meio da re-seleção e inserção de indivíduos.

Em de Castro e Von Zuben (2000) é apresentado um algoritmo para o reconhecimento de padrões e uma versão simplificada para otimização. O algoritmo para otimização é dividido em seis fases, conforme apresentado no Algoritmo 8.

Algoritmo 8 Pseudocódigo do algoritmo CLONALG

Parâmetros:
$-nAb$: Número de células presentes na população;
$-\beta$: Fator da ponderação do número de clones criados por célula;
$-d\%$: Porcentagem de novos indivíduos a serem substituídos na população;

function CLONALG(nAb, β, $d\%$)
  $u \leftarrow 0$;
  $N_c \leftarrow$ Definir número de clones por célula;
  $P_u \leftarrow$ gerar nAb células aleatoriamente;
  while Critério de parada não satisfeito **do**
    Avaliar o *fitness* das células $c_p \in P_u$;
    $P_c \leftarrow$ Gerar N_c clones por célula $c_p \in P_u$;
    Aplicar processo de hipermutação em cada clone $cl_k \in P_c$;
    $P_{u+1} \leftarrow$ para cada célula pai $c_p \in P_u$ selecionar o seu melhor clone em $cl_{c_p} \in P_c$;
    Avaliar o *fitness* das células $c_p \in P_{u+1}$;
    Substituir $d\%$ das piores células na população P_{u+1} por novas células;
    $u \leftarrow u + 1$;
  end while
  retornar P_u;
end function

A primeira fase representa a construção da população P_0 de células inicial, com nAb indivíduos (células B). É proposto por de Castro e Von Zuben (2000) que a criação destes indivíduos se dê de forma aleatória ou através de algum método que mantenha diversidade populacional.

Na segunda fase, os indivíduos são avaliados por um função de afinidade célula/antígeno para cada individuo de P_u. Como discutido na Seção 3.4.2.1, em problemas de otimização não existe uma população de antígenos a ser reconhecida, mas, sim, uma função a ser otimizada. Então, a afinidade de um anticorpo deve considerar, de alguma forma, o valor da função objetivo a ser minimizada ou maximizada (de Castro, 2001), comportando-se como uma função de *fitness*.

Na terceira fase (clonagem), um novo conjunto P_c é gerado, sendo os elementos cópias idênticas (clones) das células do conjunto P_u. Diferentemente da versão de reconhecimento de padrões, para otimização, todos os indivíduos devem ser clonados, sendo gerado o mesmo número de clones (N_c) por célula $c_p \in P_u$. Para tanto, pode-se usar:

$$N_c = round(\beta * nAb), \tag{3.1}$$

proposta por de Castro (2001), em que β é um parâmetro ponderador do percentual de clones gerado em função do número de células presentes na população (nAb); $round()$ é o operador que arredonda o valor entre parênteses para o inteiro mais próximo.

O processo de clonagem biológico acarreta em mutações aleatórias nos receptores dos clones (conjunto P_c). Esta característica é simulada pela quarta fase do algoritmo. Segundo de Castro e Von Zuben (2000), as mutações devem ser inversamente proporcionais à afinidade. Este tipo de mutação gera dois tipos de comportamento no algoritmo. Indivíduos menos qualificados em termos da função de *fitness* são submetidos a pertubações mais significativas em sua estrutura, tendo, por objetivo, a diversificação da população. Já indivíduos mais bem avaliados realizam pequenas mutações, com o intuito de realizar a maturação do indivíduo em direção aos ótimos locais. A Figura 3.3 ilustra o comportamento do operador de hipermutação somática, ilustrando os saltos entre regiões do espaço por indivíduos e a caminhada em direção aos ótimos locais.

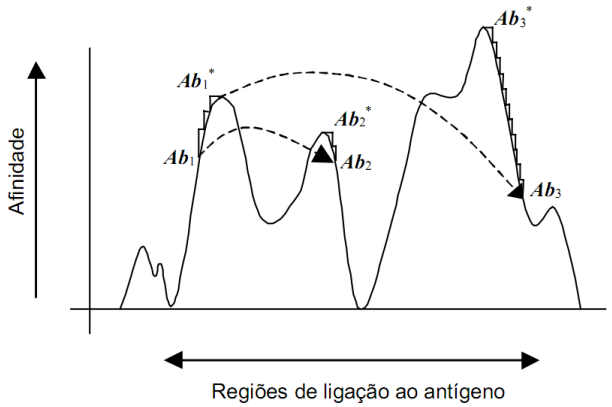


Figura 3.3: Comportamento do operador de hipermutação somática. Fonte: de Castro (2001).

Na quinta fase do algoritmo, o conjunto de células que integram a população P_{u+1} da próxima geração é formado, a partir do conjunto $P_u \cup P_c$, selecionando-se nAb indivíduos. Neste processo, para cada conjunto célula/clones descendentes, é selecionada a célula que contenha a solução com melhor valor da função de *fitness*. Por fim, na sexta e última fase, $d\%$ das células presentes na população P_{u+1}, com pior avaliação da função de *fitness*, serão substituídas por novas células, afim de garantir a manutenção da diversidade de população.

3.4.2.3 Rede imunológica para otimização - opt-aiNet

A *Artificial Immune Network* (aiNet) é um modelo de SIA baseado na teoria de rede imunológica e princípio da seleção clonal. Inicialmente, a aiNet foi proposta por de Castro e Zuben (2002) para reconhecimento de padrões, utilizando internamente o processo de microevolução do CLONALG como parte do processo de treinamento da rede. O algoritmo funciona como um extrator de regras a partir de dados de entrada, gerando *clusters* de células que representam as regras extraídas dos padrões dos dados de treinamento. Estes *clusters* poderiam ser utilizados para a classificação de novos dados de entrada, ou mesmo para o entendimento do comportamento de um conjunto de dados. A rede de células geradas pelo aiNet é uma abordagem auto-organizável sem definição previa do número de regras extraídas, tendo, assim, uma

população dinâmica e autoregulável, caracterizando um método de aprendizagem não supervisionado com características multimodais.

A partir desta abordagem, de Castro e Timmis (2002b) propõem uma extensão da aiNet para a resolução de problemas de otimização multimodal, denominada opt-aiNet (*Artificial Immune Network for Optimization*). Neste sentido, a rede se comporta como uma extensão do CLONALG, mas incorporando características da teoria da rede imunológica. Nesta abordagem, o princípio da seleção clonal contido no CLONALG é utilizado como método de exploração do espaço de soluções. Entretanto, externamente a este, são acrescentadas características da teoria da rede imunológica, como a supressão de células, o que acarreta na dispersão, e, consequentemente, na diversificação das soluções que serão exploradas pelo processo de microevolução do CLONALG. Como será detalhado posteriormente, este tipo de abordagem permite ao método uma maior cobertura do espaço de soluções, a definição dinâmica do tamanho da população e evita a exploração de uma determinada região por múltiplas células.

A versão da aiNet para otimização para espaços contínuos, proposta por de Castro e Timmis (2002b) e expressa no Algoritmo 9, considera a seguinte terminologia:

- Célula - c_p: um vetor de números reais, que representam a codificação de variáveis de uma possível solução do problema;
- *Fitness* - $f_{Ag}(c_p)$: representa a afinidade célula/antígeno, sendo o valor da função objetivo normalizado no intervalo $[0, 1]$, de modo que quanto maior a qualidade do valor da função objetivo de uma célula, maior será o seu *fitness*;
- *Afinidade* - $f_{Ab}(c_1, c_2)$: representa a afinidade célula/célula, sendo calculada pela distância euclidiana entre duas células (c_1 e c_2) ou qualquer outro operador de distância de vetores reais;
- Clones - cl_{c_p}: representa as N_c células descendentes de uma célula c_p;
- População - P_u: representa a população de células presentes na geração u;
- População de clones - P_c: representa a população de clones gerados a partir das células presentes em P_u;

O algoritmo começa com a criação de nAb células de forma aleatória e com probabilidade uniforme, dentro das fronteiras do espaço de variáveis. Estas células são atribuídas a uma população P_u, sendo que u representa a geração. Em seguida, estas células serão exploradas pelo princípio da seleção clonal, com uma estrutura muito similar ao CLONALG. Primeiramente, é avaliado o *fitness* de cada célula $c_p \in P_u$, sendo que quanto maior a qualidade da solução presente na célula c_p em relação às demais células contidas na população Ab, maior será o *fitness* da célula c_p. As Equações 3.2 e 3.3 representam, respectivamente, funções de *fitness* para problemas de maximização e minimização, sendo $f(c_p)$ o valor da função objetivo da célula c_p:

$$f_{Ag}(c_p) = \frac{f(c_p) - \min_{j \in P_u}(f(c_j))}{\max_{j \in P_u}(f(c_j)) - \min_{j \in P_u}(f(c_j))}; \tag{3.2}$$

Algoritmo 9 Pseudocódigo do algoritmo opt-aiNet

Parâmetros
$-nAb$: Número inicial de células presentes na população;
$-N_c$: Número de clones gerados por geração para cada célula presente em nAb;
$-\beta$: Fator de ponderação do operador de mutação;
$-\sigma_s$: Taxa limiar de supressão, distância mínima que células devem permanecer entre si para que não sejam sujeitas a seleção negativa;
$-d\%$: Porcentagem de novos indivíduos a ser inserida após a supressão;
$-limit_{supressao}$: Limiar de variação do *fitness* médio para disparar processo de supressão;

function OPT-AINET(nAb, N_c, β, σ, $d\%$)
  $u \leftarrow 0$;
  $P_u \leftarrow$ gerar nAb células aleatoriamente;
  while Critério de parada não satisfeito **do**
    Avaliar o *fitness* f_{Ag} das células $c_p \in P_u$;
    $avgF_u \leftarrow$ Avaliar $fitness$ médio da população P_u ;
    $P_c \leftarrow$ Gerar N_c para cada célula $c_p \in P_u$;
    Aplicar processo de hipermutação em cada clone $cl_k \in P_c$;
    $P_{u+1} \leftarrow$ para cada célula pai $c_p \in P_u$ selecionar o seu melhor clone em $cl_{c_p} \in P_c$;
    $avgF_{u+1} \leftarrow$ Avaliar $fitness$ médio da população P_{u+1} ;
    if $avgF_u - avgF_{u+1} < limit_{supressao}$ **then**
      $P_{u+1} \leftarrow$ Suprimir população P_{u+1} pelo limiar σ_s e afinidade f_{Ab};
      Inserir $d\%$ novas células em P_{u+1};
    end if
    $u \leftarrow u + 1$;
  end while
  return P_u;
end function

$$f_{Ag}(c_p) = 1 - \frac{f(c_p) - \min_{j \in P_u}(f(c_j))}{\max_{j \in P_u}(f(c_j)) - \min_{j \in P_u}(f(c_j))}. \tag{3.3}$$

Em seguida, é avaliado o valor médio do *fitness* da população e este valor é armazenado em $avgF_u$. Esta informação será utilizada posteriormente para verificar se o processo de supressão de células será disparado na geração corrente. O método prossegue e, para cada célula c_p presente na população Ab, é gerado um conjunto cl_{c_p} contendo N_c clones, sendo os clones cópias idênticas do vetor de números reais que representam a solução contida na célula c_p. Todos os clones gerados são armazenados em uma população provisória P_c.

Um vez obtida a população de clones, cada clone será submetido ao processo de hipermutação somática. Esta, por sua vez, assim como no CLONALG, é inversamente proporcional à afinidade, ou seja, quanto maior a afinidade, menores as alterações em um determinado clone, tentando realizar assim o direcionamento deste para um mínimo local. Já células com pouca afinidade dão saltos maiores no espaço de busca. A Equação 3.4 apresenta a definição de uma operação de mutação em espaços contínuos, sendo cl'_k um clone após a mutação do clone cl_k, β o parâmetro que define a taxa de mutação e $N(0,1)$ um número aleatório seguindo a distribuição de probabilidade gaussiana normal:

$$cl'_k = cl_k + \beta \times e^{-f_{Ag}(cl_k)} \times N(0,1). \tag{3.4}$$

Posteriormente, serão selecionadas, dentre os clones mutantes presentes em P_c e a população ancestral de células P_u, quais células irão compor a próxima geração

de células P_{u+1}. Este método é realizado segundo os mesmos passos do CLONALG. Assim, para cada célula $c_p \in P_u$, são verificados se os clones descendentes desta célula apresentam melhoria na função objetivo devido à mutação aplicada. Caso ocorra, o clone que apresentar melhor valor da função objetivo é selecionado para ingressar em P_{u+1}. Caso contrário, as células c_p permanecem na população durante a próxima geração.

Até este ponto, o algoritmo segue praticamente os mesmos passos do CLONALG, com apenas algumas diferenças sutis. Entretanto, a partir dos próximos passos, o algoritmo acrescenta elementos da teoria da rede imunológica, o que torna a dinâmica do opt-aiNet substancialmente diferente do CLONALG. Primeiramente, o algoritmo avalia o *fitness* médio da população de células da próxima geração P_{u+1}, e verifica se há grande variação do *fitness* médio entre a geração u e $u+1$, ou seja, se a diferença entre os valores de *fitness* médio da geração atual ($avgF_u$) e o da próxima geração ($avgF_{u+1}$) é maior que o parâmetro $limit_{supressao}$ de indicador de estabilidade de variação. Em caso afirmativo, indica que a resposta imune das células pertencentes à população P_{u+1} ainda está em fase de maturação de afinidade e, assim, não é realizada a supressão, já que a resposta imune ainda não se estabilizou. Caso contrário, as células presentes em P_{u+1} já estão estáveis e pode-se realizar o processo de supressão da população.

A supressão é baseada no processo de "seleção negativa" presente na teoria da rede imunológica, no qual o sistema realiza uma autoregulação da população após o processo de expansão clonal, no qual as próprias células do sistema, ao entrarem em contato com outras células complementares, realizam a marcação para eliminação destas, e, assim, ocorre uma redução na população de células. Para simular este comportamento, o opt-aiNet propõe que soluções que estejam em uma mesma região no espaço de variáveis sejam suprimidas, sendo mantida apenas a célula com melhor avaliação do *fitness*. Logo, para verificar a similaridade entre soluções, é utilizado algum operador de distância no espaço de variáveis. Em espaços contínuos, é utilizada a distância euclidiana, sendo chamada de afinidade célula/célula f_{Ab}. Além disto, é utilizado o parâmetro σ_s, o qual representa o limiar de supressão, indicando a distância mínima na qual as células devem permanecer entre si.

Após a supressão de células, o método insere na população P_{u+1} um novo conjunto de células, criando $d\%$ soluções aleatoriamente. Este passo acrescenta diversidade à população, para que um novo ciclo de microevolução seja iniciado.

O comportamento geral do algoritmo opt-aiNet basicamente pode ser descrito como uma método que gera repetidas respostas imune, começando com uma população e a evoluindo através do processo de microevolução, em um processo muito similar ao CLONALG. Uma vez que a resposta imune tenha sido estabilizada, parte da população é suprimida, retirando-se soluções que ocupam uma mesma região do espaço de soluções, sendo, em seguida, inseridas novas soluções e iniciada uma nova resposta imune, para maturação das células que compõem a nova população. Esta ação de retirar da população células que ocupam uma mesma região no espaço de busca evita esforço computacional desnecessário e garante uma maior cobertura do espaço de busca.

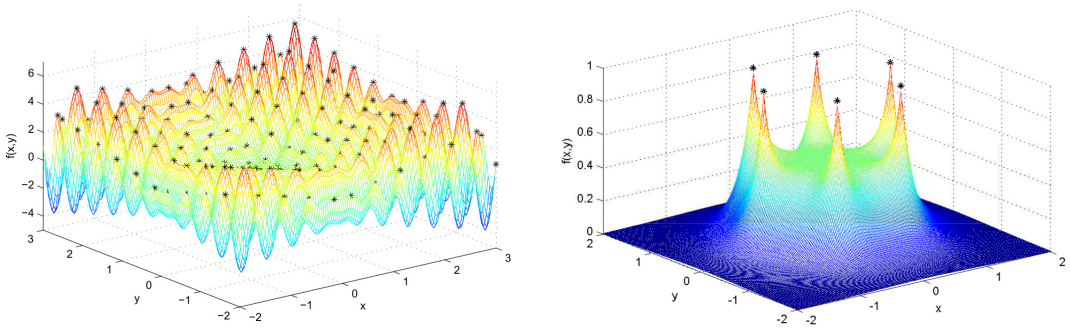
Os experimentos realizados por Coelho (2011) para as funções bidimensionais *Multi* (Equação 3.5) e *Roots* (Equação 3.6) são ilustrados, respectivamente, nas Fi-

guras 3.4(a) e 3.4(b):

$$f(x,y) = x.sin(4\pi x) - y.sin(4\pi y + \pi)x, y \in [-2,3]; \tag{3.5}$$

$$f(z) = \frac{1}{1+|z^6-1|}, z = x + iyx, y \in [-2,2]. \tag{3.6}$$

Percebe-se, a partir destes experimentos, que a cobertura do espaço de busca promovida pelo opt-aiNet encontra boa parte dos mínimos locais presentes nas funções, além de garantir a distribuição das soluções em todo o espaço de soluções.



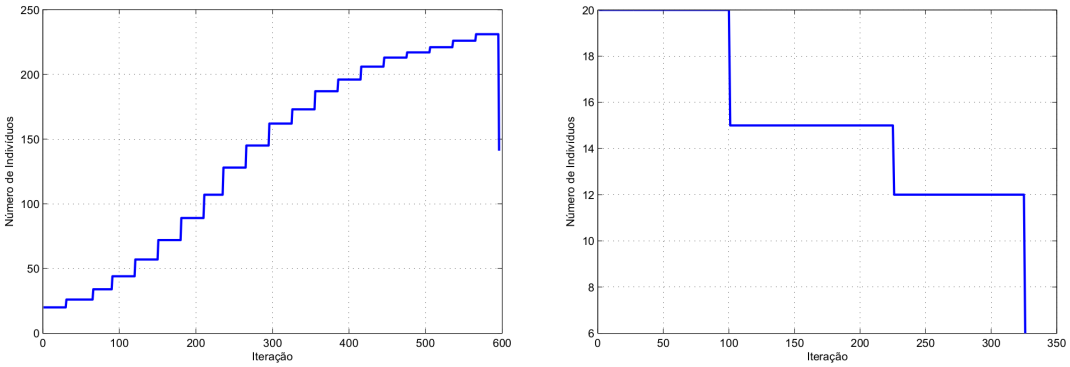
(a) Função *Multi* (b) Função *Roots*

Figura 3.4: População de soluções encontradas ao final da execução do opt-aiNet. Fonte: Coelho (2011)

Ainda observando os experimentos de Coelho (2011), apresentados nas Figuras 3.4(a) e 3.4(b), percebe-se que a população de células não apresenta um tamanho fixo, podendo variar em relação ao tamanho inicial (nAb) tanto positiva quanto negativamente. As Figuras 3.5(a) e 3.5(b) apresentam, respectivamente, para as funções *Multi* e *Roots*, o número de células presentes na população a cada geração. Deve-se observar que, para a função *Multi*, a população aumenta consideravelmente com o passar das gerações. O contrário ocorre com a função *Roots*. O tamanho da população varia de acordo com o limiar de supressão (σ_s). Um valor elevado de σ_s reduz o tamanho da população, mas pode dificultar a busca por soluções, já que uma determinada célula terá que cobrir um espaço grande de soluções. Caso este seja muito "acidentado", como o espaço de busca da função *Multi*, dificultará encontrar o ótimo local da região. Já um valor muito reduzido de σ_s acarretará em uma superpopulação, o que pode acarretar em alto custo computacional, principalmente em abordagens que integram o opt-aiNet a métodos de busca local. Estes problemas serão discutidos com maiores detalhes no Capítulo 5, sendo proposta uma alternativa para contornar os mesmos.

3.4.2.4 Rede imunológica para otimização combinatória - copt-aiNet

Considerando as características dos problemas combinatórios descritas na Seção 3.2, de Sousa et al. (2004) propõem uma extensão do opt-aiNet para otimização combinatória. Esta extensão é mostrada no Algoritmo 10, apresentando duas principais diferenças em relação ao opt-aiNet.



(a) Função *Multi* (b) Função *Roots*

Figura 3.5: Evolução do número de células em relação ao número de iterações (gerações) do opt-aiNet. Fonte: Coelho (2011)

A primeira diferença fundamental entre as abordagens é a inserção de algoritmos de busca local. Problemas de otimização combinatória são muito sensíveis a alterações aleatórias em sua estrutura, e estas dificilmente levam a mínimos locais sem nenhum método de refinamento. No algoritmo copt-aiNet, o operador de busca local não é utilizado em todas as iterações, sendo ativado após ls_{freq} gerações sem melhora das nk melhores células da população. Mesmo não sendo acionada em todas as gerações, a inserção de um método de busca local ajuda o algoritmo a encaminhar para os mínimos locais soluções diversificadas a partir dos clones hipermutados.

O segundo ponto de alteração no algoritmo copt-aiNet em relação ao algoritmo opt-aiNet está relacionado à geração de novas células. No opt-aiNet, sugere-se que, após o processo de supressão, sejam inseridas $d\%$ novas células na população, geradas de forma aleatória. Já no copt-aiNet, de Sousa et al. (2004) sugerem que as novas células a ingressarem na população sejam geradas a partir de recombinação gênica (Oprea, 1999). Nesta, trechos do código genético de células com alta qualidade compõem uma biblioteca gênica. Assim, a partir da combinação de trechos genéticos presentes na biblioteca, são geradas novas soluções. O método tem comportamento similar aos operadores de *crossover* presentes em algoritmos genéticos.

3.4.2.5 Operador *Ageing*

Abordagens evolucionárias tendem a convergir prematuramente para mínimos locais, normalmente devido ao elitismo promovido pelos métodos de seleção. Para contornar este problema, diversas técnicas de incremento de diversidade são propostas na literatura, como *modelos de ilhas* (Whitley et al., 1998) e *nichos* (Miller e Shaw, 1996). Um abordagem de incremento da diversidade denominada *Ageing* (ou *Aging*) foi inicialmente proposta por Schwefel e Rudolph (1995) para Algoritmos Evolucionários (AEs). Apesar disto, Oliveto e Sudholt (2014) destacam que esta abordagem raramente é utilizada por AEs, mas é comumente encontrada integrada a SIAs. Os trabalhos de Horoba et al. (2009) e Jansen e Zarges (2011a) mostraram que, para SIAs baseados apenas no princípio da seleção clonal, como opt-IA Cutello et al. (2007), o operador de *Ageing* incrementa o desempenho do algoritmo para problemas nos quais os mínimos locais apresentam fortes bacias de atração e a bacia de atração

Algoritmo 10 Pseudocódigo do algoritmo copt-aiNet

Parâmetros:
$-nAb$: Número inicial de célula presentes na população;
$-N_c$: Número de clones gerados por geração para cada célula presente em nAb;
$-\beta$: Fator de ponderação do operador de mutação;
$-\sigma_s$: Taxa limiar de supressão, distância mínima que células devem permanecer entre si para que não sejam sujeitas à seleção negativa;
$-limit_{supressao}$: Limiar de variação do *fitness* médio para disparar processo de supressão;
$-d\%$: Porcentagem de novos indivíduos a ser inserida após a supressão;
$-ls_{freq}$: Número de gerações que se admite a não melhoria no *fitness* dos indivíduos;
$-nk$: Número de células a serem avaliadas para verificar se não houve melhoria no *fitness* durante M gerações;

function COPT-AINET(nAb, N_c, β, σ, $limit_{supressao}$, $d\%$, ls_{freq}, nk)
  $u \leftarrow 0$;
  $P_u \leftarrow$ gerar nAb células aleatoriamente;
  while Critério de parada não satisfeito **do**
    Avaliar o *fitness* f_{Ag} das células $c_p \in P_u$;
    $avgF_u \leftarrow$ Avaliar $fitness$ médio da população P_u ;
    $P_c \leftarrow$ Gerar N_c para cada célula $c_p \in P_u$;
    Aplicar processo de hipermutação em cada clone $cl_k \in P_c$;
    $P_{u+1} \leftarrow$ para cada conjunto $c_p \in P_u \cup cl_{c_p} \in P_c$ selecionar a melhor célula;
    $avgF_{u+1} \leftarrow$ Avaliar $fitness$ médio da população P_{u+1} ;
    if $avgF_u - avgF_{u+1} < limit_{supressao}$ **then**
      $P_{u+1} \leftarrow$ Suprimir população P_{u+1} pelo limiar σ_s e afinidade f_{Ab};
      Inserir $d\%$ novas células através de recombinação gênica em P_{u+1};
    end if
    if nk melhores células em P_{u+1} não tiverem o *fitness* aumentado em ls_{freq} gerações **then**
      Aplicar busca local em cada célula $c_p \in P_{u+1}$;
    end if
    $u \leftarrow u + 1$;
  end while
  Aplicar busca local em cada célula $c_p \in P_u$;
  return P_u;
end function

do ótimo global apresenta um caminho pouco atrativo.

O operador de *Ageing* é um operador relativamente simples, consistindo em determinar um ciclo de vida para cada solução. No contexto de SIAs para otimização, células que estão apresentando evolução pelo processo de maturação celular apresentam "idade" nula. À medida em que os descendentes de uma célula não acrescentarem melhoria em relação à célula original, a idade celular é incrementada. Caso uma célula atinja uma "idade máxima", indicando que se encontra presa em uma região do espaço de busca e já não contribui com a resposta imune, deve ser descartada do processo de busca, independente da qualidade da solução. Jansen e Zarges (2010) apresentam um *framework* para a utilização de *Ageing* para algoritmos populacionais. Este *framework* é apresentado no Algoritmo 11, considerando a seguinte terminologia:

- **Célula -** c_p: A célula c_p é composta por uma solução $c_p[s]$ e uma idade $c_p[age]$;
- **Idade -** $c_p[age]$: idade de um solução, mais precisamente, o número de gerações sem apresentar melhora na função objetivo;
- **Descendente -** cl_{c_p}: um descendente direto da célula c_p;

- **População -** P_u: população de soluções presentes na geração u;
- **População de descendentes -** P'_u: população de soluções cl'_{c_p} descendentes de $c_p \in P_u$;
- **Idade máxima -** τ: idade máxima que uma solução pode atingir antes de ser eliminada do processo de busca.

Algoritmo 11 *Framework* para aplicação da abordagem *Ageing*

Parâmetros:
$-\mu$: Tamanho de soluções presentes na população;
$-\tau$: Idade máxima permitida por solução;

1: **function** AGEING-FRAMEWORK(μ, τ)
2: **Inicialização:**
3: $u \leftarrow 0$;
4: $P_u \leftarrow$ gerar μ soluções aleatoriamente;
5: Para todo $c_p \in P_u$ fazer $c_p[age] = 0$;
6: **while** Critério de parada não satisfeito **do**
7: **Ageing: Incremento de idade**
8: Para todo $c_p \in P_u$ fazer $c_p[age] = c_p[age] + 1$;
9: **Evolução: Exploração do espaço de busca**
10: $P'_u \leftarrow$ Executar operadores de exploração do espaço de busca;
11: **Aging: Inferir idade das soluções descendentes,** cl_{c_p}**, presentes em** P'_u
12: **if** $f(cl_{c_p}) < f(c_p)$ **then**
13: $cl_{c_p}[age] = 0$;
14: **else**
15: $cl_{c_p}[age] = c_p[age]$;
16: **end if**
17: $P_u \leftarrow P_u \cup P'_u$;
18: **Ageing: Remoção de soluções:**
19: $P_{u+1} \leftarrow$ Remover todas soluções $c_p \in P_u$, as quais, $c_p[age] > \tau$
20: **Seleção:**
21: $P_{u+1} \leftarrow$ Aplicar operador de seleção de soluções para próxima geração;
22: **Ageing: Preenchimento da população**
23: **if** $|P_{u+1}| < \mu$ **then**
24: $P_{u+1} \leftarrow P_{u+1} \cup \{\mu - |P_{u+1}|$ novas soluções$\}$;
25: **end if**
26: $u \leftarrow u + 1$;
27: **end while**
28: **return** P_u;
29: **end function**

Jansen e Zarges (2010) destacam que a forma como a idade dos descendentes de uma solução é inferida (Linhas 12 à 16 do Algoritmo 11) pode contribuir diretamente no desempenho do operador. Jansen e Zarges (2009) apresentam três operadores de incremento de idade dos descendentes, sendo estes:

(i) *static pure aging*: este operador está descrito na versão padrão do Algoritmo 11, na qual, caso a solução descendente cl_{c_p} não apresente melhora na função objetivo em relação à solução geradora c_p, é atribuída ao descendente a idade do gerador; logo, para um problema de minimização, se $(f(cl_{c_p}) \geq f(c_p))$, então $cl_{c_p}[age] = c_p[age]$; caso contrário, o descendente tem sua idade anulada e, logo, $cl_{c_p}[age] = 0$;

(ii) *evolutionary aging*: na abordagem evolucionária, a idade do descendente segue o processo biológico, sempre sendo anulada; logo, $cl_{c_p}[age] = 0$;

(iii) *genotypic aging*: é uma abordagem híbrida, que apresenta o mesmo mecanismo do *static pure aging*, mas se difere deste na condição para anulação da idade. No *genotypic aging*, além de anular a idade de descendentes que melhorem a função objetivo, também anula-se a idade de soluções com mesmo valor da função objetivo, mas que ocuparem diferentes regiões do espaço de variáveis; logo, se $(f(cl_{c_p}) \leq f(c_p)$ e $cl_{c_p} \neq c_p)$, então $cl_{c_p}[age] = 0$; caso contrário, $cl_{c_p}[age] = c_p[age]$.

Nos trabalho teóricos realizados por Jansen e Zarges (2009) e Jansen e Zarges (2011b), é demonstrado que o operador *static pure aging* permite encaminhar para ótimos locais soluções em que o *evolutionary aging* falha. Os autores atribuem este comportamento ao fato do *evolutionary aging* inviabilizar a remoção de descendentes com alto desempenho, inviabilizando assim a inserção de novos indivíduos e a diversificação da população. Em contrapartida, em ambientes com muitas regiões planas, o *evolutionary aging* apresenta melhor desempenho em relação ao *static pure aging*. Este último não consegue identificar melhoria nas soluções descendentes, uma vez que observa apenas o valor da função objetivo, não conseguindo observar que a solução descendente caminhou no espaço de busca. Devido a isto, Jansen e Zarges (2009) introduzem o *genotypic aging*, que considera, para o reinício da idade, tanto a melhora no valor da função objetivo quanto a não alteração do valor da função objetivo, além de alterações na estrutura da solução no espaço de variáveis. Isto permite que as soluções descendentes que estão caminhando em um espaço de busca plano tenham o tempo de vida prolongado.

3.4.3 VNS-aiNet

O trabalho proposto por Diana et al. (2017) levanta uma série de restrições que o copt-aiNet apresenta quando aplicado em problemas combinatórios que exigem a incorporação de heurísticas e/ou estruturas de vizinhança para percorrer o espaço de busca. Para contornar os problemas levantados, Diana et al. (2017) propõem uma metaheurística híbrida denominada VNS-aiNet. Esta abordagem é definida pela incorporação, ao opt-aiNet, de características dos métodos de trajetória, como ILS e VNS (i.e., a perturbação de soluções seguida por um processo de busca local), além da inclusão de uma estratégia baseada em operadores de *Ageing* para remover e desprivilegiar, do processo de busca, soluções presas em ótimos locais. O Algoritmo 12 apresenta um pseudo código da extensão da rede imunológica para otimização combinatória, denominada VNS-aiNet, sendo seus componentes detalhados nas próximas seções.

3.4.3.1 Representação da célula B e populações de soluções

Uma célula c_p no algoritmo VNS-aiNet é composta por duas variáveis. A variável $c_p[s]$ representa a solução codificada de acordo as características do problemas estudado. Já a variável $c_p[mat]$ indica o nível de maturação da célula. Esta última é o

Algoritmo 12 Variable Neighborhood Search Imunne Network(VNS-aiNet)

Parâmetros:
$-nAb$: Tamanho da população de células ativas;
$-nC_{max}$: Número máximo de clones por célula;
$-\beta$: Taxa máxima de mutação ;
$-mat_{+}$: Nível de incremento do fator de maturação;
$-\sigma_s$: Raio limite supressão;

function VNS-aiNet(nAb, nC_{max}, β, mat_{+}, σ_s)
  $u \leftarrow 0$;
  $M_u \leftarrow \{\}$;
  $P_u \leftarrow$ gerar nAb células aleatoriamente;
  while Critério de parada não satisfeito **do**
    Avaliar o *fitness* das células $c_p \in P_u$;
    $P_c \leftarrow$ Clonar células $c_p \in P_u$;
    Aplicar mutação em cada clone $cl_k \in P_c$;
    Aplicar busca local em cada clone $cl_k \in P_c$;
    Atualizar o fator $c_p[mat]$ das células $c_p \in P_u$ e $cl_k \in P_c$;
    $P_u \leftarrow P_u \cup P_c$;
    $R_u \leftarrow$ Remover células de P_u em que $c_p[mat] \geq 1$;
    Atualizar o *fitness* das células $c_p \in P_u$;
    $P_{u+1} \leftarrow$ Suprimir pelo *fitness* P_u;
    Adicionar $nAb - |P_{u+1}|$ novas células a P_{u+1};
    $M_{u+1} \leftarrow$ Suprimir pela função objetivo $M_u \cup R_u \cup (P_u - P_{u+1})$;
    $u \leftarrow u + 1$;
  end while
  $M_u \leftarrow$ Suprimir pela função objetivo $M_u \cup P_u$;
  return M_u;
end function

artifício usado para identificar que as soluções estão estagnadas em mínimos locais, que não estão contribuindo com a resposta imune.

O VNS-aiNet contém duas populações de células, denominadas P_u (Células ativas) e M_u (Memória Imunológica). P_u é a população de células ativas na geração u, responsáveis pela resposta imune na atual geração, ou seja, a população de células que estão explorando o espaço de busca. Portanto, as células presentes em P_u apresentam o nível de maturação $c_p[mat] < 1$. Já a população M_u representa a memória imunológica do sistema, ou seja, as soluções que foram exploradas e atingiram o nível de maturação máximo $c_p[mat] = 1$, ou foram suprimidas pelo operador de supressão (e.i. Seção 3.4.3.8). Para garantir a multimodalidade do algoritmo, em M_u são armazenadas as soluções factíveis de melhor avaliação da função objetivo, desde que elas respeitem a distância σ_s entre si no espaço de variáveis. O modelo de seleção de células que permanecem na memória será detalhado na Seção 3.4.3.10.

Já que M_u é composta por células de memória, M_0 deve ser inicialmente vazia. A população de células ativas da primeira geração, denominada P_0, deve ser iniciada de maneira aleatória, ou através de um método de construção de soluções, como a fase construtiva da metaheurística GRASP. Deve-se apenas garantir que as células não se encontrem em uma mesma região do espaço de busca, já que a principal característica do método é trabalhar paralelamente em diferentes regiões do espaço de busca, evitando que as soluções se concentrem em uma única bacia de atração.

3.4.3.2 Afinidade célula/antígeno

O segundo passo do algoritmo consiste na avaliação da afinidade célula/antígeno. Como, em problemas de otimização, não existe um antígeno a ser reconhecido e sim uma função objetivo a ser otimizada, como descrito na Seção 3.4.2.1, a afinidade célula/antígeno normalmente é interpretada como uma função de *fitness* exclusivamente ligada à função objetivo. Diana et al. (2017) argumentam que esta abordagem acarreta na convergência prematura do algoritmo, propondo a função de afinidade célula/antígeno:

$$f_{Ag}(c_p) = (1 - \alpha) f_{like}(c_p) + \alpha f_{mat}(c_p), \tag{3.7}$$

sendo $f_{like}(c_p) \in [0, 1]$ uma função de *fitness-like*, que considera a qualidade da solução em relação às demais células da população. Já o termo $f_{mat}(c_p) \in [0, 1]$ é uma função inversamente proporcional ao número de gerações em que a solução c_p não apresenta melhoria, devido a estar presa em uma região pouco promissora do espaço de busca ou ter encontrado um mínimo local. Este termo reduz a afinidade, desestimulando a exploração da solução pelos demais operadores, exceto pelo operador de mutação, o qual é estimulado. O valor α é um parâmetro do algoritmo que regula a relação de compromisso entre $f_{like}(c_p)$ e $f_{mat}(c_p)$.

Um exemplo de função de *fitness-like* para problemas de minimização é dada por:

$$f_{like}(c_p) = 1 - \frac{f(c_p) - \min_{j \in P_u} (f(c_j))}{\max_{j \in P_u} (f(c_j)) - \min_{j \in P_u} (f(c_j))}, \tag{3.8}$$

sendo $f(c_p)$ o valor da função objetivo a ser minimizada pela solução presente na célula c_p. Já $\min_{j \in P_u}(f(c_j))$ e $\max_{j \in P_u}(f(c_j))$ são, respectivamente, os valores mínimo e máximo da função objetivo das soluções presentes na população ativa P_u.

A função $f_{mat}(c_p)$ é apresentada como:

$$f_{mat}(c_p) = 1 - c_p[mat], \tag{3.9}$$

sendo o complemento do fator de maturação $c_p[mat]$, criado para cada célula $c_p \in P_u$. Como já discutido, este fator é inicialmente nulo e, a cada geração sem melhora, é incrementado pelo índice mat_+, sendo este um parâmetro do algoritmo. Caso o fator atinja o valor máximo, a célula é retirada da população ativa. Tanto o incremento do fator como a retirada de células da população são discutidos na Seção 3.4.3.6.

As consequências desta função no comportamento de cada componente do algoritmo são apresentados nas próximas seções. Para fins de simplificação na notação, a função de afinidade célula/antígeno será chamada, nas próximas seções, de função de *fitness*.

3.4.3.3 Clonagem

O processo de clonagem é a primeira etapa da expansão clonal. Neste processo, as células com maior afinidade com o antígeno são estimuladas a se multiplicarem. No VNS-aiNet é proposto evitar esforço computacional em soluções que se encontrem em regiões não promissoras do espaço de busca ou que já se encontrem nos mínimos locais de sua região. Assim, diferentemente do algoritmo copt-aiNet, que gera um número

fixo de clones por célula, é proposto que cada célula c_p gere um número distinto de clones, dado por nC_p, diretamente proporcional à função de *fitness*. O valor nC_p é expresso como:

$$nC_p = \max\left\{1, f_{Ag}(c_p) \times nC_{\max}\right\}, \tag{3.10}$$

sendo $nC_{\max}$ um parâmetro do algoritmo, definido como o número máximo de clones por célula.

A partir da Equação 3.10, observa-se que, à medida que uma solução deixe de apresentar melhoria, a afinidade desta com o antígeno, representada por $f_{Ag}(c_p)$, será reduzida, desestimulando, assim, continuamente, a expansão clonal e a exploração desta solução, mesmo que esta tenha uma boa avaliação da função objetivo.

É importante ressaltar que cada clone gerado é uma cópia idêntica da célula origem, inclusive o fator de maturação $c_p[mat]$.

3.4.3.4 Hipermutação

A hipermutação é o processo no qual se realiza a diversificação e a maturação das células clonadas. Para garantir esta duas características, o algoritmo copt-aiNet utiliza a taxa de mutação inversamente proporcional ao valor da função *fitness*. Assim, células com baixo desempenho são submetidas a elevados níveis de mutação (diversificação). Já soluções com elevado *fitness* são submetidas a pequenas mutações, para garantir a maturação das células.

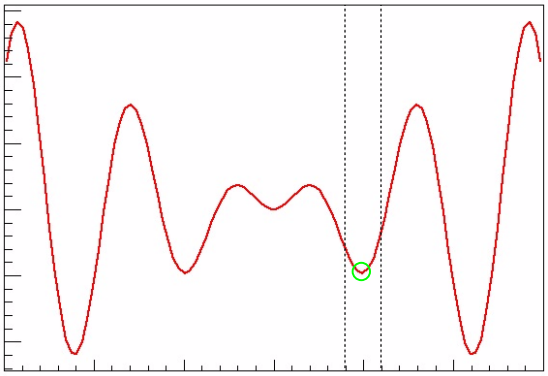
Devido às características dos problemas combinatórios que permitem busca por estruturas de vizinhança, o VNS-aiNet propõe alterar este processo, acrescentando características da metaheurística VNS. No VNS é feita uma perturbação na solução, denominada *shake*, e, em seguida, é realizada uma busca local. O nível do ruído acrescentado à solução é aumentado sistematicamente, variando as estruturas de vizinhança. A perturbação deve começar como um pequeno ruído e, à medida que este não consiga produzir soluções que escapem dos mínimos locais no qual a solução se encontra, o nível do ruído deve ser elevado, com o intuito de levar a solução para regiões mais distantes do espaço de busca. Neste sentido, a perturbação pode ser interpretada como o operador de hipermutação somática proposto originalmente para o algoritmo opt-aiNet, mas considerando o nível de mutação inversamente proporcional ao termo $f_{mat}(c_p)$ de maturação da função de *fitness*, como pode ser visto em:

$$c_p = T_{mut}\left(c_p, \max\left\{1, round\left(\beta \times e^{-f_{mat}(c_p)}\right)\right\}\right). \tag{3.11}$$

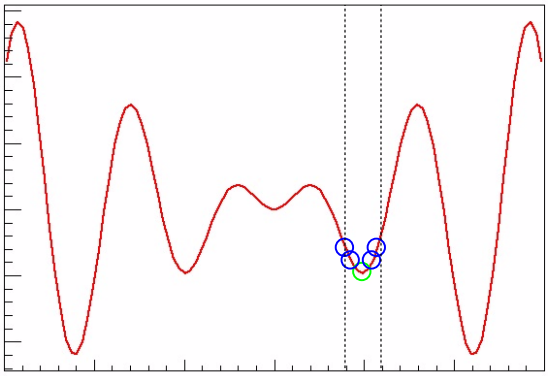
Esta abordagem permite que a perturbação em uma solução presa em uma região ou em um mínimo local seja sistematicamente ampliada, permitindo que o processo de busca local possa maturá-la e encaminhá-la a um mínimo de melhor qualidade. Nesta expressão, β é um parâmetro do algoritmo, que representa o número máximo de pertubações em um clone; $T_{mut}(c_p, n_{mut})$ é uma função que realiza n_{mut} perturbações na solução presente na célula c_p. Note que n_{mut} é um número inteiro. Assim, a parte fracionária da função $\beta \times e^{-f_{mat}(c_p)}$ deve ser arredondada para o inteiro mais próximo. Já que as pertubações são utilizadas para diversificação da solução, pelo menos uma pertubação deve ser realizada por clone. Assim, caso $round(\beta \times e^{-f_{mat}(c_p)})$ seja menor que 1, o valor de n_{mut} é uma unidade. A função $T_{mut}(c_p, n_{mut})$ deve ser específica para

o problema; assim, pode ser uma função que realiza tanto pertubações em apenas uma estrutura de vizinhança ou pode considerar mais estruturas, podendo variar com a modelagem que deseja-se obter para resolução do problema.

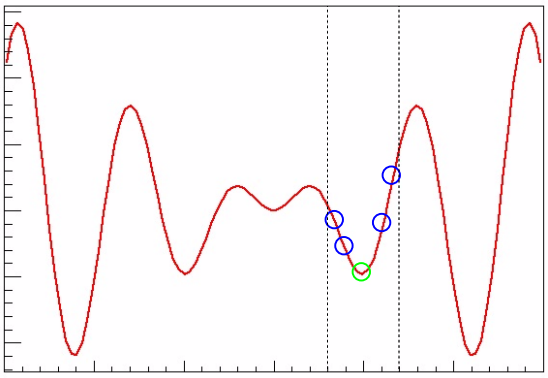
A Figura 3.6 apresenta um exemplo do comportamento do operador, considerando o nível de mutação como inversamente proporcional à função $f_{mat}(c_p)$. A Figura 3.6(a) apresenta o raio de mutação nos instantes em que uma solução acabou de encontrar o ótimo local, ou seja, o valor de maturação da célula é nulo. Já a Figura 3.6(b) apresenta os clones gerados e submetidos à mutação dentro da faixa inicial de mutação. As Figuras 3.6(c) e 3.6(d) apresentam o aumento na faixa de mutação, resultante do incremento do fator de maturação da célula. É possível visualizar que, com o passar das gerações, a faixa de mutação irá aumentar devido ao incremento do fator de maturação, possibilitando que a mutação possa encaminhar os clones para regiões mais distantes do espaço de soluções. Assim, o operador de busca local poderá encaminhar estas soluções para novos mínimos locais.



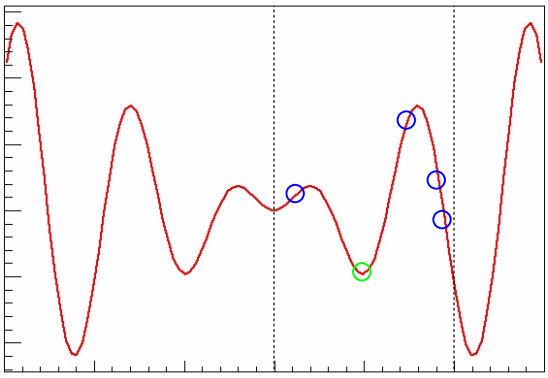
(a) Limite de mutação com nível de maturação nulo



(b) Clones mutados com nível de maturação nulo



(c) Clones mutados com nível de maturação intermediario



(d) Clones mutados com nível de maturação alto

Figura 3.6: Exemplo do comportamento do operador de mutação com função de *fitness* proposta.

3.4.3.5 Busca Local

Como descrito na seção anterior, o operador de mutação proposto tem como único intuito a diversificação de uma solução, sendo necessário, portanto, o uso de mecanismos para o refinamento dos clones. de Sousa et al. (2004) e Coelho et al. (2011) já levantaram a necessidade de operadores de busca local para as versões do algoritmo opt-aiNet para otimização combinatória, mas as abordagens propostas por estes utilizam a busca local após a estabilização da busca aleatória, realizada pelo processo de microevolução advindo das contínuas hipermutações. Ao contrário destas abordagens, no VNS-aiNet, logo após a diversificação, é realizado um processo de busca local. Em decorrência disto, após cada geração, os clones serão encaminhados para os mínimos locais da região na qual se encontram.

Ao utilizar um operador de busca local a cada geração, o custo do processo de busca local pode inviabilizar o método. Está é a razão para o VNS-aiNet separar a população nas populações, P_u e M_u descritas anteriormente. Assim, P_u é uma população de tamanho fixo e deve ter o número de células devidamente controlado. Além disto, o processo de busca local tende a consumir o maior custo computacional do algoritmo, e, desse modo, deve-se tentar incorporar características do problema ao processo de busca local, para tentar reduzir o esforço computacional despendido por esta fase.

3.4.3.6 Atualização do fator de maturação

Tanto a clonagem e a mutação, descritas nas seções anteriores, quanto a supressão, descrita na próxima seção, dependem diretamente do fator de maturação da célula $c_p[mat]$. Este fator indica o tempo no qual uma determinada célula c_p não apresenta melhora no valor da função objetivo. Como pode ser observado, o fator de maturação é baseado nos operadores de *ageing* descritos na Seção 3.4.2.5, e, desse modo, a cada geração devem ser atualizados. A atualização é realizada através do operador *static pure aging*, uma vez que, por mais que o operador *genotypic aging* permita uma melhor exploração de regiões planas, pode permitir também que as células fiquem transitando de um mínimo para outro dentro de uma região plana, sempre anulando o fator de maturação e não permitindo que a célula seja eliminada pelo operador.

O processo de atualização do fator de maturação das células é descrito no Algoritmo 13. Neste algoritmo, após o processo de mutação e maturação (busca local) das células clonadas, espera-se que estas tenham escapado dos mínimos locais nos quais as células genitoras se encontram. Assim, todas as soluções $cl_k \in P_c$ clonadas têm sua função objetivo avaliada. Caso apresentem melhora em relação a sua célula genitora $cl_k[parent]$, seu fator de maturação é anulado. Caso contrário, o fator de maturação é incrementado em $mat_+ \in [0, 1]$, sendo este valor um parâmetro do algoritmo. Todas as células genitoras têm o fator $c_p[mat]$ incrementado em mat_+, já que não sofrem alteração em relação à geração anterior $u - 1$.

3.4.3.7 Remoção de células completamente maturadas

Após a atualização do fator de maturação das células, é necessário verificar quais células atingiram o nível máximo de maturação para serem retiradas do processo

Algoritmo 13 Modelo de atualização do fator de maturação das células.

Parâmetros:
$-P_u$: População de células geradoras;
$-P_c$: População de células clonadas;
$-mat_+$: Nível de incremento do fator de maturação;

```
1: procedure ATUALIZAR_FATOR_DE_MATURAÇÃO(P_u, P_c, mat_+)
2:     for cl_k ∈ P_c do
3:         if f(cl_k) < f(cl_k[parent]) then
4:             cl_k[mat] ← 0;
5:         else
6:             cl_k[mat] ← cl_k[mat] + mat_+;
7:         end if
8:     end for
9:     for c_p ∈ P_u do
10:        c_p[mat] ← c_p[mat] + mat_+;
11:    end for
12: end procedure
```

de busca. Neste ponto não se faz mais necessário distinguir clones e células genitoras e, assim, os clones são incorporados à população de células ativas P_u (Linha 11, Algoritmo 12). Em seguida, esta nova população P_u é submetida ao processo de remoção de células descrito no Algoritmo 14. As células cujo o fator de maturação satisfizerem a condição $c_p[mat] >= 1$ considera-se que atingiram o nível máximo de maturação. Neste caso, devem ser removidas da população ativa P_u, já que atingiram sua máxima contribuição à resposta imunológica. Como deseja-se manter a multimodalidade, criou-se uma população de memória $M_u \neq P_u$. Assim, as soluções que atingirem o limiar de maturação na geração u são retiradas de P_u e inseridas em R_u, sendo retornado pelo método à população de células removidas R_u. As células presentes em R_u serão utilizadas pelo processo que irá atualizar a memória M_u, descrito na Seção 3.4.3.10.

Algoritmo 14 Modelo de remoção de células totalmente maturadas.

Parâmetros:
$-P_u$: População de células;

```
1: function REMOÇÃO_DE_CÉLULAS_MATURADAS(P_u)
2:     R_u ← {};
3:     for c_p ∈ P_u do
4:         if (c_p[mat] ≥ 1) then
5:             P_u ← P_u − {c_p};
6:             R_u ← R_u ∪ {c_p};
7:         end if
8:     end for
9:     return R_u;
10: end function
```

3.4.3.8 Supressão

Após o processo do expansão clonal, a resposta imune se encontra estável. Neste ponto ocorre uma interação entre as células da população P_u, com o intuito de se regular a quantidade de células na população. Pela teoria da rede imunológica, célu-

las com alto grau de afinidade entre si identificam umas às outras através da ligação do receptor BCR com os *idiotopos*, marcando-se para a remoção pelo sistema imunológico. A similaridade entre a estrutura de duas células define a afinidade entre as mesmas. Observa-se assim que soluções em um mesmo subespaço de busca concorrem entre si pela permanência na população.

Em todas as abordagens da família opt-aiNet, quando duas células se encontram em uma mesma região do espaço de busca, a célula com maior valor da função objetivo é selecionada para permanecer na população. Diana et al. (2017) argumentam que esta abordagem em populações de tamanho restrito pode trazer o problema de levar, para a próxima geração, soluções estagnadas em mínimos locais, desperdiçando recursos computacionais e impossibilitando uma maior cobertura do espaço de busca. Para contornar este problema, o VNS-aiNet incorpora a competição entre células pertencentes a um mesmo subespaço, utilizando a função de *fitness* (Equação 3.7). Este método de supressão tenta garantir uma resposta imune contínua, priorizando soluções de boa qualidade, mas, ao mesmo tempo, evitando a seleção de soluções estagnadas em mínimos locais. Esta abordagem prejudica diretamente na capacidade de multimodalidade do algoritmo. Assim, as células descartadas por este processo devem se tornar candidatas a comporem a população de células de memória M_u. A atualização de M_u é discutida na Seção 3.4.3.10. O pseudo-código apresentado no Algoritmo 15 define o operador de supressão.

Algoritmo 15 Modelo de supressão da população.

Parâmetros:
$-P_u$: População a ser suprimida;
$-\sigma_s$: Raio limite supressão;
$-nAb$: Número máximo de células presentes na população gerada P_{u+1};

```
1: function SUPRESSÃO(P_u, σ_s, nAb)
2:     i ← 0;
3:     P_{u+1} ← [ ];
4:     Ordenar P_u pela função f_Ag(c_p);
5:     Inserir c_p ∈ P_u com maior fitness em P_{u+1};
6:     while |P_{u+1}| < nAb and i < |P_u| do
7:         c_p ← P_u[i];
8:         if Af_cell(c_p, P_{u+1}) < (1 − σ_s) then
9:             P_{u+1} ← add(P_{u+1}, c_p);
10:        end if
11:        i ← i + 1;
12:    end while
13:    return P_{u+1};
14: end function
```

No Algoritmo 15, deve-se considerar a afinidade célula/célula dada por:

$$Af_{cell}(c_p, P_{u+1}) \;=\; 1 - \min_{\forall c_j \in P_{u+1}} d(c_p, c_j), \tag{3.12}$$

sendo $d(c_p, c_j) \in [0, 1]$ a função de distância entre as células c_p e $c_j \in P_{u+1}$, que deve ser definida de acordo com a estrutura do problema. Esta expressão representa o quanto a estrutura de uma solução $c_p[s]$ se assemelha, no espaço de variáveis, às células que integram a população P_{u+1} passada como parâmetro. Desse modo, para construir a próxima população de células ativas P_{u+1}, o Algoritmo 15 ordena a população P_u

em ordem decrescente pelo valor da função de *fitness* $f_{Ag}(c_p)$. A solução com maior *fitness* é inserida na população da próxima geração P_{u+1}. Em seguida, percorre-se a lista de forma ordenada pelo valor de *fitness*, verificando se cada célula apresenta a afinidade inferior a $1 - \sigma_s$, em relação às demais células previamente selecionadas para P_{u+1}. Portanto, avalia-se se a célula $c_p \in P_u$, candidata a entrar na população P_{u+1}, apresenta-se a uma distância superior a σ_s no espaço de variáveis em relação a todas as células selecionadas para P_{u+1}. Caso isto ocorra, a célula candidata passa a ingressar a população de células ativas da próxima geração. Caso contrário, ela será descartada pelo processo de supressão. O método seleciona no máximo nAb células, dadas pelas células com maiores avaliações de *fitness* que respeitarem as distâncias entre si no espaço de variáveis superior ao valor σ_s. As demais células presentes em P_u também serão descartadas pelo processo de supressão.

3.4.3.9 Inserção de novas soluções

A supressão pode gerar a população da próxima geração com número de células inferior a nAb. Em consequência disto, após a supressão são geradas $\max(0, nAb - |P_{u+1}|)$ novas células pelo método utilizado na geração da população inicial. Estas nova células geradas são inseridas na população ativa da próxima geração P_{u+1}. Este processo contribui na diversificação das solução presentes na população.

3.4.3.10 Atualização da memória imunológica

Como discutido na Seção 3.4.3.1, a memória imunológica M_u tem como objetivo garantir a multimodalidade do algoritmo, armazenando soluções históricas que se encontrem em mínimos locais a algumas gerações e não contribuem com o processo de busca. Para garantir estas características, a atualização da memória M_u deve ser disparada em dois pontos, conforme pode ser observado no Algoritmo 12.

No primeiro ponto (Linha:16, Algoritmo 12), logo após a supressão da população de células ativas e clones P_u, as células que não ingressaram em P_{u+1}, ou seja, $P_u - P_{u+1}$, em união com as células que atingiram o nível máximo de maturidade, mostradas em R_u, e as células já presentes na memória imunológica M_u, se tornam candidatas a comporem a memória imunológica M_{u+1} da próxima geração. Deste conjunto de soluções candidatas, são selecionadas as melhores células separadas entre si pelo menos à distância σ_s no espaço de variáveis. Caso seja necessário, pode-se usar outro parâmetro para selecionar a distância entre células permitidas na memória, já que σ_s deve ser calibrado para ajudar na cobertura do espaço de busca pelo algoritmo. Não há necessidade de se definir o tamanho da memória M_u, mas a não definição pode tornar o processo oneroso, já que o tamanho da memória pode crescer exponencialmente. A atualização da memória pode ser realizada pelo operador de supressão descrito no Algoritmo 15, com uma pequena alteração. Para ordenar as soluções (Linha:4, Algoritmo 15), é utilizado o critério de otimização $f_{like}(c_p)$ e não a função de fitness $f_{Ag}(c_p)$, utilizada pelo operador de supressão. Esta alteração se faz necessária, uma vez que as células que irão ingressar em M_{u+1} não irão participar do processo de busca, sendo utilizada apenas para armazenar um banco das melhores soluções encontradas separadas pela distâncias σ_s, garantindo, assim, a multimodalidade do método.

Já o segundo ponto (Linha:19, Algoritmo 12) ocorre quando o algoritmo atinge o critério de parada. Neste ponto, a memória M_u deve ser retornada, sendo u a última geração. Deve-se perceber que as células que compõem a última população ativa, ou seja, P_u, em nenhum momento se tornaram candidatas a ingressarem na memória imunológica. Logo, antes de retornar à memória M_u, esta deve ser submetida a um processo de união com a última população de células ativas P_u, usando o operador de supressão pela função objetivo, conforme descrito no parágrafo anterior.

Capítulo 4

Projeto de métodos de busca local em ambientes de máquinas não relacionadas

Este Capítulo discute a importância de métodos de busca local para o desempenho de metaheurísticas que visam solucionar problemas de sequenciamento de tarefas em ambientes de máquinas paralelas não relacionadas. Para isto, é proposta uma metodologia para o projeto e validação de métodos de busca local para um ambiente de máquinas paralelas não relacionadas, com o objetivo de minimizar o atraso total ponderado. É realizado o reprojeto do operador de busca local de três metaheurísticas previamente propostas na literatura. Primeiramente, na Seção 4.1 é apresentada uma contextualização do Capítulo, apresentando os objetivos, justificativas e hipóteses avaliadas. Em seguida, na Seção 4.2, é apresentada a definição do problema de sequenciamento em ambientes UPMSMDST usados neste capítulo para o projeto de métodos de busca local. A Seção 4.3, por sua vez, apresenta as três variantes da metaheurística VND avaliadas como operadores de busca local, as seis estruturas de vizinhança propostas, além de apresentar as formas e ordens de exploração destas estruturas de vizinhança avaliadas. Posteriormente, na Seção 4.4, é apresentada a metodologia experimental utilizada no capítulo. Nesta seção é apresentada uma metodologia para o reprojeto dos operadores de busca local das três metaheurísticas avaliadas. Além disto, são apresentadas as condições computacionais usadas nos experimentos, além da metodologia utilizada para comparar os resultados obtidos pelas abordagens resultantes do reprojeto dos operadores de busca local, com as abordagens de estado arte disponíveis na literatura. Já na Seção 4.5 são apresentados os resultados dos experimentos computacionais, que são avaliados em três estágios. Primeiro, avalia-se como os componentes utilizados para construção dos operadores de busca local se comportam para cada metaheurística, sendo selecionados os componentes que acarretam em melhor incremento do desempenho das mesmas. Em seguida, são avaliadas as contribuições das estruturas de vizinhança no processo de busca. Por fim, é avaliado o desempenho das metaheurísticas com o operador de busca local reprojetado contra as metaheurísticas originais e as abordagens estados da arte para o problema. A última seção do Capítulo (Seção 4.6) apresenta discussões e conclusões obtidas através dos resultados encontrados. Destacamos que este Capítulo é fortemente baseado no artigo Diana e de Souza (2020), publicado como

parte dos requisitos obrigatórios para conclusão desta tese.

4.1 Introdução

Nas ultimas duas décadas diversos métodos metaheurísticos foram propostos para a solução de problemas complexos de otimização. Metaheurísticas se consolidaram como uma das principais ferramentas computacionais usadas para a solução de problemas combinatórios de natureza NP-Difícil. Estes métodos têm, como principal vantagem, baixo custo computacional quando comparado a métodos exatos. Apesar de metaheurísticas não garantirem soluções ótimas, elas possuem estratégias inteligentes de busca global que permitem navegar por regiões promissoras do espaço de busca. Para problemas de otimização combinatória, muitas metaheurísticas incorporam algum operador de busca local atrelado às estratégias de varredura do espaço de busca. Em geral, estes operadores tendem a consumir a maior parte do custo computacional de uma abordagem metaheurística. Em contrapartida, muitas vezes, componentes de otimização global de uma metaheurística recebem maior atenção e mais detalhes nos estudos sobre a sua real efetividade sobre o processo de otimização de um problema.

Como visto na Seção 3.3.4, o método *Variable Neighborhood Descent* (VND) é uma metaheurística normalmente usada como operador de busca local da metaheurística VNS. A integração do VND como operador de busca local pode ser estendida para além do VNS, tendo vários casos de sucesso quando incorporado a outras metaheurísticas (e.g. Avci e Topaloglu (2017); Diana et al. (2015); Zeng et al. (2018); Duarte et al. (2018)). A principal diferença do VND para outros operadores de busca local é considerar que o mínimo local em um estrutura de vizinhança não necessariamente é o mínimo local em outra estrutura. Devido a isto, o VND considera a busca de forma sistêmica de duas ou mais estruturas de vizinhança. Esta característica exige que um operador de busca local baseado em VND seja cuidadosamente desenhando, considerando a efetividade de cada estrutura de vizinhança proposta. Adicionalmente a isto, já que as estruturas de vizinhança trabalham em conjunto, o desempenho total do operador se dá pela interação entre os processos de busca destas estruturas de vizinhança (Mjirda et al., 2017). Considerando isto, o estudo realizado por Mjirda et al. (2017) indicou que, para o problema do caixeiro-viajante, a efetividade do VND varia de acordo com a ordem, método (variações VND, Seção 3.3.4) e forma (estratégia de busca local, e.g. *Best Improvement, First Improvement*, Seção 3.3.4) de exploração das estruturas de vizinhança.

A literatura recente de problemas de sequenciamento vem apresentando grande aderência do VND como operador de busca local de metaheurísticas, como, por exemplo, para problemas de sequenciamento de máquina única (Subramanian et al., 2014; González e Vela, 2015; Subramanian e Farias, 2017; Chen, 2019) e *flow shop* híbrido (Peng et al., 2019). Já para problemas em ambientes UPMSMDST como os estudados nesta tese, Fleszar et al. (2012) encontraram, para a minimização do *makespan*, soluções de alta qualidade utilizando uma estratégia *Multistart*, gerando múltiplas soluções randômicas, refinando-as com um operador de busca local VND. Esta abordagem proposta por Fleszar et al. (2012) mostrou-se mais eficiente que abordagens complexas baseadas em *Tabu Search* (Helal e Al-Salem, 2006) e *Ant Colony Opti-*

mization (Arnaout et al., 2010). Ainda para a minimização do *makespan*, Diana et al. (2015) apresentaram uma integração muito eficiente do VND, como operador de busca local, a uma abordagem populacional imuno inspirada. Já para minimização do TWT em ambientes UPMSMDST, Diana et al. (2018) propuseram a integração do VND como operador de busca local da metaheurística *Iterated Local Search* - ILS (Lourenço et al., 2003). Esta abordagem, superou, na maior parte dos cenários testados, abordagens de estado da arte para o problema em estudo.

Considerando a sinergia observada na aplicação do VND como operador de busca local de problemas de sequenciamento, este capítulo analisa o comportamento do VND como operador de busca local de metaheurísticas para ambientes UPMSMDST com o objetivo de minimização do TWT. Como o principal objetivo do capítulo é estudar a influência de características do operador de busca local na convergência de metaheurísticas, foca-se aqui apenas em aspectos do domínio do operador de busca local. Em consequência, componentes de otimização global são negligenciados. Devido a isto, integramos o VND como operador de três metaheurísticas estado da arte para o ambiente e objetivo estudado (i.e., *Iterated Greedy Search* - IGS (Lin et al., 2011), *Artificial Bee Colony* - ABC (Ying e Lin, 2012) e *Genetic Algorithm* - GA (Zeidi e MohammadHosseini, 2015)), substituindo o operador proposto nos respectivos artigos pelas variações do VND propostas em nosso estudo. Assim, neste capítulo são avaliadas as seguintes hipóteses:

(H1) A ordem, o método (variações VND) e a forma que as estruturas de vizinhança são exploradas podem incrementar/decrementar o desempenho do operador de busca local baseado em VND;

(H2) As metaheurísticas têm diferentes demandas do operador de busca local. Assim, a importância de cada estrutura de vizinhança varia de acordo com a metaheurística avaliada;

(H3) O desempenho de uma metaheurística, já consolidada, pode ser incrementado apenas com a substituição do operador de busca local.

A maior contribuição deste capítulo está em um rigoroso estudo do VND como operador de busca local de metaheurísticas endereçadas para a solução de problemas de minimização do TWT em ambientes UPMSMDST. Para isto, baseados nos principais tipos de movimentos para ambientes UPMSMDST, apresentados na Seção 2.5, propomos seis estruturas de vizinhança, utilizando características do problema para reduzir o espaço de busca. Através destas estruturas, propomos uma metodologia baseada em projeto de experimentos, para avaliar como a integração destas, em conjunto com o VND, impacta no desempenho das metaheurísticas avaliadas. Através desta metodologia identificamos a importância de cada estrutura de vizinhança para cada metaheurística. A metodologia também ajuda a identificar a ordem, método e forma de exploração das estruturas de vizinhança que se mostrem mais aderentes às características de cada metaheurística. Os resultados gerados por cada metaheurística, integrada ao operador de busca local gerado pela metodologia proposta, é comparado aos resultados gerados pela metaheurística com operador de busca local original. Além disto, comparamos as "novas" metaheurísticas aos resultados de estado da arte para o problema estudado. Como contribuição adicional, a metodologia proposta para o projeto do VND pode ser facilmente adaptada para projeto de

VND em outros contextos, apesar de ficarmos restritos neste trabalho ao problema de minimização do TWT em ambientes UPMSMDST.

4.2 Definição do Problema

Como apresentado na seção anterior, o problema estudado neste capítulo consiste na minimização do TWT em ambientes UPMSMDST. Seguindo as definições apresentadas no Capítulo 2, este problema pode ser definido resumidamente como: dado um conjunto $N = \{1, 2, \ldots, n\}$ de tarefas, que devem ser sequenciadas em um conjunto $M = \{1, 2, \ldots, m\}$ de máquinas paralelas não relacionadas ($\alpha = Rm$). Cada tarefa $k \in N$ tem um tempo de processamento p_{ik}, que depende da máquina i na qual a tarefa k é sequenciada. Uma vez iniciado o processamento de uma tarefa k na máquina i, este não pode ser interrompido, e a máquina i fica impedida de receber o processamento de qualquer outra tarefa. Antes do processamento de cada tarefa $k \in N$, deve ser realizada a configuração da máquina i, ou seja, o processamento de cada tarefa k depende de um tempo de preparação s_{ijk}, dependente da tarefa k, da máquina i na qual esta é atribuída, e da tarefa j que precede imediatamente k no sequenciamento da máquina i. Além disto, no problema estudado neste capítulo, é permitida a presença de *job ready times*. Assim, cada tarefa k apresenta o instante mínimo r_k no qual k estará disponível para o sequenciamento. Considerando este ambiente, o instante de término C_k de uma tarefa k é definido como:

$$C_k = \max\{r_k, C_j + s_{ijk}\} + p_{ik}, \tag{4.1}$$

sendo C_j o instante de término da tarefa j que precede imediatamente k na sequência da máquina i. Como dito anteriormente, utilizamos como objetivo a minimização do TWT. Assim, dado que d_k e α_k são, respectivamente, a data desejada de entrega de k e a penalização por atraso de k, o TWT pode ser definido conforme apresentado na Equação 2.7.

Pela notação de três campos apresentada na Seção 2.1, o problema estudado neste capítulo pode ser resumidamente definido como $Rm \mid r_k, s_{ijk} \mid TWT$. De acordo com Lin e Hsieh (2014), este problema é NP-Difícil, já que a minimização do TWT em máquina simples também é NP-Difícil (Lenstra et al., 1977).

4.3 *Variable Neighborhood Descent*

Como pode-se observar na Seção 3.3.4, o projeto de um operador de busca local baseado na metaheurística VND passa primeiramente pela definição do conjunto de estruturas de vizinhança NS. Este capítulo propõe seis estruturas de vizinhança para o problema de minimização do TWT em ambientes UPMSMDST. Estas seis estruturas são apresentadas na Seção 4.3.1. Além disto, a Hipótese (H1), avaliada neste trabalho é baseada no estudo sobre a aplicação do VND ao problema do caixeiro viajante proposto em Mjirda et al. (2017), o qual indica que a ordem e método (variações do VND, Seção 3.3.4) e forma de exploração das estruturas de vizinhança influenciam o desempenho do VND como operador de busca local. Para avaliar a Hipótese (H1), apresentamos, na Seção 4.3.2.1, duas ordens de exploração da estrutura

de vizinhança. A Seção 4.3.2.2 apresenta as três variações do VND utilizadas no projeto experimental do operador de busca local. Também relacionado à Hipótese (H1), na Seção 4.3.2.3 apresentamos os critérios de aceite de soluções avaliados para cada uma das seis estruturas de vizinhança avaliadas.

4.3.1 Estruturas de vizinhança

As estruturas de vizinhança propostas neste capítulo são baseados nos movimentos clássicos de sequenciamento apresentados na Seção 2.5. Nesta seção propomos a utilização de seis estruturas de vizinhança, quatro destas são denominadas como:

(i) NE.II (Seção 4.3.1.1): baseada em movimento de Inserção Interna (Seção 2.5.1);

(ii) NE.IS (Seção 4.3.1.2): baseada em movimento de Troca Interna (Seção 2.5.2);

(iii) NE.EI (Seção 4.3.1.3): baseada em movimento de Inserção Externa (Seção 2.5.3); e

(iv) NE.ES (Seção 4.3.1.4): baseada em movimento de Troca Externa (Seção 2.5.4).

Estruturas de vizinhança baseadas nestes movimentos são comumente encontrados na literatura (e.g. Fleszar et al. (2012); Diana et al. (2013); Kramer e Subramanian (2019); Pei et al. (2019)). Também são propostas duas estruturas de vizinhança baseadas na fase construtiva/destrutiva da metaheurística IGS (Seção 3.3.5), denominadas:

(v) NE.IGS (Seção 4.3.1.5); e

(vi) NE.SIGS (Seção 4.3.1.6).

Estas duas estruturas são baseadas no método de busca local proposto por Lin e Hsieh (2014).

4.3.1.1 Vizinhança de Inserção Interna - NE.II

A *Internal Insertion Neighborhood* (NE.II), baseada nos movimentos de inserção interna ilustrado na Figura 2.2, consiste em remover uma tarefa k, atribuída a uma máquina i, e inserir esta em uma posição p da mesma máquina. A estratégia de exploração desta estrutura é descrita no Algoritmo 16. Uma máquina $i \in MT$ é selecionada aleatoriamente com probabilidade uniforme do conjunto MT (Linha:3), sendo MT o conjunto de máquinas que apresentem tarefas com atraso, ou seja, $MT = \{i \in M : T_i > 0\}$, $T_i = \sum_{k \in N_i} T_k$, em que N_i é o conjunto de tarefas atribuídas à máquina i. Em seguida, é aplicado um método de descida entre as tarefas presentes na máquina i, considerando o movimento de inserção interna (Algoritmo 24, Apêndice A), passando o argumento $w = i$. Nesta estrutura de vizinhança avaliamos estratégias de descida por BI (*Best improvement*) e FI (*First Improvement*).

Algoritmo 16 Estrutura de vizinhança NE.II.

1: **function** NE.II(s, *tipo_descida*)
2: $MT \leftarrow \{\forall\ i \in M : T_i > 0\}$;
3: $i \leftarrow U[1, |MT|]$;
4: **return** *descida_insercao*$(s, TWT, i, i,$ *tipo_descida*$)$;
5: **end function**

4.3.1.2 Vizinhança de Troca Interna - NE.IS

A *Internal Swap Neighborhood* (NE.IS) é uma estrutura de vizinhança baseada em movimentos de troca interna (Seção 2.5.2), que consistem na troca de posição do sequenciamento de duas tarefas k e j, $j \neq k$, ambas atribuídas à mesma máquina i. Como pode ser observado no Algoritmo 17, esta estrutura de vizinhança é definida similarmente à vizinhança NE.II, alterando apenas o procedimento de exploração de vizinhança (Linha: 4), considerando a descida por troca (Algoritmo 27, Apêndice A), sendo o argumento $w = i$.

Algoritmo 17 Estrutura de vizinhança NE.II.

1: **function** NE.IS(s, *tipo_descida*)
2: $MT \leftarrow \{\forall\ i \in M : T_i > 0\}$;
3: $i \leftarrow U[1, |MT|]$;
4: **return** *descida_troca*$(s, TWT, i, i,$ *tipo_descida*$)$;
5: **end function**

4.3.1.3 Vizinhança Inserção Externa - NE.EI

A estrutura de vizinhança NE.EI (*External Insertion Neighborhood*) é baseada no movimento de inserção externa (Seção 2.5.3), no qual uma tarefa k, atribuída à máquina i, é removida de i e inserida na posição p da máquina w, $i \neq w$. A exploração completa (i.e., envolvendo todas as máquinas e tarefas) desta estrutura de vizinhança apresenta um custo computacional elevado. Devido a isto, propomos uma redução na estrutura descrita no Algoritmo 18. Primeiramente, uma máquina i, que contém custos relativos a atraso, ou seja, $i \in MT = \{i \in M : T_i > 0\}$, deve ser selecionada aleatoriamente, com probabilidade uniforme (Linha 5). Em seguida, outra máquina w é selecionada aleatoriamente da lista $L = \{w \in M : \forall\ w \neq i\}$. É permitido que a máquina w não apresente custos de atraso (Linha 8). Posteriormente, é aplicado uma estratégia de descida por inserção, removendo tarefas da máquina i e inserindo na máquina w (Algoritmo 24, Apêndice A). A estratégia de busca pode ser por BI ou FI, definida pelo parâmetro *tipo_descida*. Se a estratégia de descida acarretar em melhora da solução, a nova solução é retornada (Linha 15). Caso contrário, outra máquina é aleatoriamente escolhida e removida de L, sendo atribuída a w e o procedimento de busca é retomado. O processo de busca é finalizado assim que uma solução de melhora for encontrada, ou todas as tarefas alocadas na máquina i tenham sido avaliadas em todas as demais máquinas (Linha 7).

Algoritmo 18 Estrutura de vizinhança inserção externa - NE.EI.

```
1:  function NE.EI(s, tipo_descida)
2:      s* ← s;
3:      s' ← NULL;
4:      MT ← {∀ i ∈ M : T_i > 0};
5:      i ← rand(MT);
6:      L ← {∀ w ∈ M : w ≠ i};
7:      while (f(s*) == f(s) and |L| > 0) do
8:          w ← rand(L);
9:          s' ← descida_insercao(s, TWT, i, w, tipo_descida);
10:         if f(s') < f(s*) then
11:             s* ← s';
12:         end if
13:         remover(L, w);
14:     end while
15:      return s*;
16: end function
```

4.3.1.4 Vizinhança de Troca Externa - NE.ES

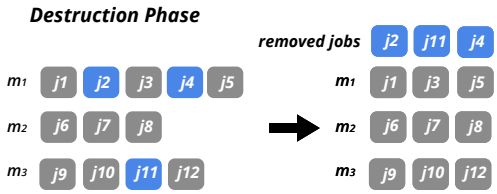
A Figura 2.5 apresenta o movimento na estrutura de vizinhança aqui denominada como NE.ES (*External Swap Neighborhood*). Conforme descrito na Seção 2.5.4, os movimentos nesta estrutura consistem na troca de uma tarefa k, atribuída em uma máquina i, com uma tarefa j atribuída em uma máquina w, sendo $i \neq w$. A estrutura NE.ES é definida e explorada de forma similar à vizinhança NE.EI, sendo o Algoritmo 18 utilizado quase em sua totalidade, substituindo apenas o método de descida utilizado pela estrutura (Linha 9). Na vizinhança NE.ES utiliza-se a descida por troca. Assim, a Linha 9 do Algoritmo 18 é substituída pela função $descida\_troca(s, TWT, i, w, tipo\_descida)$, definida no Algoritmo 27 (Apêndice A).

4.3.1.5 Vizinhança *Iterated Greedy Search* - NE.IGS

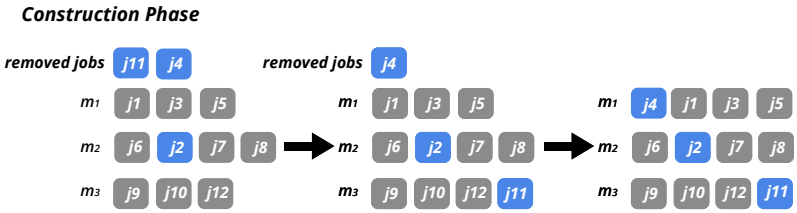
A estrutura de vizinhança *Iterated Greedy Search Neighborhood* (NE.IGS) é uma função de vizinhança que difere consideravelmente das quatro estruturas previamente apresentadas. A $NE.IGS$ é baseada na estratégia gulosa das fases de destruição e construção da metaheurística IGS (Seção 3.3.5). Esta estrutura é ilustrada no Algoritmo 19 e consiste em remover aleatoriamente, com probabilidade uniforme, rj tarefas de uma solução s, gerando uma solução s', como mostrado na Figura 4.1(a). Estas tarefas podem estar presentes em qualquer máquina. Em seguida, cada tarefa $k \in L_{rj}$ é removida de forma ordenada de L_{rj}, sendo inserida, na máquina/posição de s', que introduza o menor custo do TWT. Esta ação é ilustrada na Figura 4.1(b). O processo de remoção de tarefas em L_{rj} e reinserção em s' é repetido até que L_{rj} se encontre vazia. Nesta estruturas não utilizamos as estratégias BI e FI, mas o parâmetro rj deve ser definido e este pode influenciar diretamente no desempenho da estrutura de vizinhança.

Algoritmo 19 Estrutura de vizinhança NE.IGS.

```
1:  function NE.IGS(s, rj)
2:      s' ← s;
3:      L_rj ← remove aleatoriamente rj tarefas de s';
4:      while (|L_rj| > 0) do
5:          k ← pop(L_rj);
6:          s' ← melhor inserção de k em s';
7:      end while
8:      if f(s') < f(s) then
9:          s ← s';
10:     end if
11:      return s;
12: end function
```



(a) Fase de destruição.



(b) Fase de construção.

Figura 4.1: Exemplo de um movimento na estrutura de vizinhança Iterated Greedy Search.

4.3.1.6 Vizinhança *Swap Iterated Greedy Search* - NE.SIGS

Assim como a NE.IGS, a NE.SIGS (*Swap Iterated Greedy Search Neighborhood*) também é baseada nas fases de destruição e construção de soluções da metaheurística IGS. O primeiro passo da estrutura NE.SIGS consiste em selecionar aleatoriamente, com probabilidade uniforme, r_j tarefas, que não são removidas da solução, mas são inseridas em uma lista L_{rj}. Como a estrutura NE.SIGS é baseada em movimentos de troca, de forma ordenada e sequencial, cada tarefa k presente em L_{rj} tem seu movimento de troca avaliado com todas as demais tarefas do sequenciamento. Ao final da avaliação da troca da primeira tarefa $k \in L_{rj}$ com todas as demais tarefas, caso exista um movimento de troca que acarrete no decremento do TWT da solução, este movimento é efetivado. O processo é repetido para a segunda tarefa de L_{rj}, e assim sucessivamente, até que todas as tarefas em L_{rj} sejam exploradas. Um pseudo código da exploração da vizinhança NE.SIGS é apresentado no Algoritmo 20.

Algoritmo 20 Estrutura de vizinhança NE.SIGS.

```
1: function NE.SIGS(s, rj)
2:     s' ← s;
3:     j* ← NULL;
4:     L_rj ← seleciona aleatoriamente rj tarefas de s';
5:     while (|L_rj| > 0) do
6:         k ← pop(L_rj);
7:         i ← M(s', k);
8:         f* ← f(s')
9:         for each j ∈ s' do:
10:            f' ← avaliar_troca(s, TWT, i, k, M(s', j), j);
11:            if f' < f* then
12:                f*, j* ← f', j;
13:            end if
14:        end for each
15:        if f* < f(s') then
16:            s' ← troca(s', i, k, M(s', j*), j*);
17:        end if
18:    end while
19:    return s';
20: end function
```

4.3.2 Estratégias de exploração das estruturas de vizinhança

Como indicado por Mjirda et al. (2017), as estratégias de exploração de estruturas de vizinhança tendem a influenciar diretamente no desempenho do VND. A Hipótese (H1), proposta neste trabalho, indica que três características da estratégia de exploração de estruturas de vizinhança apresentadas influenciam o desempenho do VND para minimização do TWT no ambiente UPMSMDST estudado. Assim, nesta seção iremos detalhar:

(i) as ordens de exploração das estruturas de vizinhança propostas;

(ii) as variações do VND avaliadas para o projeto dos operadores de busca local; e

(iii) as formas de exploração utilizadas pelas estruturas de vizinhança.

4.3.2.1 Ordem de exploração das estruturas de vizinhança

Como já indicado por diversos estudos propostos na literatura, a ordem de exploração das estruturas de vizinhança influencia diretamente o desempenho e a estratégia de busca VND. Apesar disto, o custo computacional necessário para a avaliar todas as possíveis ordens de exploração das estruturas de vizinhança inviabiliza experimentos para determinar a ordem ótima. Devido a isto, neste trabalho queremos avaliar duas ordens de exploração.

A primeira ordem é definida pelo incremento da cardinalidade de permutações necessárias para determinar um movimento na estrutura de vizinhança. Está ordem é denominada aqui como $NE.O_{card}$. Deve-se observar que a definição desta ordem de exploração está diretamente relacionada ao custo computacional de exploração (tamanho) da estrutura de vizinhança. Considerando as seis estruturas de vizinhança propostas na Seção 4.3.1, a $NE.O_{card}$ é definida como:

$$NE.O_{card} = \{NE.IGS, NE.SIGS, NE.IS, NE.II, NE.ES, NE.EI\}. \quad (4.2)$$

Já a segunda ordem avaliada neste trabalho é definida pelo incremento do impacto esperado na solução devido à aplicação da estrutura de vizinhança. Esta ordem foi definida, uma vez que movimentos que causam maior mudança na estrutura de uma solução tendem a levar esta solução para regiões mais distantes do espaço de busca. Esta ordem é denominada aqui como $NE.O_{imp}$, sendo definida como:

$$NE.O_{imp} = \{NE.II, NE.IS, NE.EI, NE.ES, NE.IGS, NE.SIGS\}. \quad (4.3)$$

4.3.2.2 Métodos de exploração das estruturas de vizinhança

Outro ponto relevante no desempenho do VND são as variações na forma como as estruturas de vizinhança serão exploradas. Apesar da versão padrão do VND (Seção 3.3.4.1) ser a versão mais difundida na literatura, existem variações do mesmo, que, em cenários específicos, muitas vezes apresentam desempenho superior à versão padrão (e.g. Lü et al. (2011); Mjirda et al. (2017); Kramer e Subramanian (2019)).

Como já destacado, cada metaheurística avaliada neste capítulo apresenta características específicas de exploração do espaço de busca. Devido a isto, consideramos avaliar, para o projeto do operador de busca local de cada metaheurística, três variações do VND, sendo estas:

(i) *Basic*-VND: versão padrão do VND descrita na Seção 3.3.4.1, na qual as estruturas de vizinhança são percorridas em ordem predeterminada;

(ii) *Random*-VND: versão do VND descrita na Seção 3.3.4.2, na qual as estruturas de vizinhança são percorridas em ordem aleatória; e

(iii) *Union*-VND: descrita na Seção 3.3.4.3. Nesta variação, as estruturas são exploradas de forma "paralela" e a escolha do movimento a ser promovido a cada iteração é feita pela avaliação de todas as estruturas.

Vale destacar aqui que a forma de exploração das estruturas de vizinhança influencia na ordem de exploração das mesmas. Como deve ser observado, das três variações analisadas, apenas a versão padrão irá usar as duas ordens de exploração definidas na Seção 4.3.2.1. A variação *Random*-VND utiliza ordem aleatória. Já o método *Union*-VND não predefine uma ordem, pois necessita explorar todas as estruturas de vizinhanças para a realização de um movimento.

4.3.2.3 Forma de exploração das estruturas de vizinhança

A estratégia de busca local adotada por uma estrutura de vizinhança é outro ponto que avaliamos como determinantes para o desempenho da estratégia VND projetada. Para as estruturas baseadas em movimentos clássicos, i.e., $NE.II$, $NE.IS$, $NE.EI$, $NE.ES$, a forma de exploração da estrutura de vizinhança é representada pelo parâmetro *tipo_descida*, podendo assumir três estratégias, listadas a seguir:

(i) *Best Improvement* (BI): analisa todas as permutações possíveis na estrutura de vizinhança e efetiva o movimento que apresentar melhor resultado. Para as estruturas NE.II e NE.EI, esta estratégia é detalhada no Algoritmo 25 presente no Apêndice A.4. Já para as estruturas NE.IS e NE.ES, são detalhadas no Algoritmo 28;

(ii) *First Improvement* (FI): as permutações são analisadas em ordem aleatória, de modo que, ao encontrar qualquer permutação que acarrete em melhora efetiva, a mesma paralisa o processo de busca. Esta estratégia é detalhada no Algoritmo 26 para as estruturas NE.II e NE.EI, já o Algoritmo 29 detalha as estruturas NE.IS e NE.ES. Ambos os algoritmos são presentes no Apêndice A.4;

(iii) Não Aplicada (NA): este caso, indica que a estrutura de vizinhança não serão aplicadas.

As estruturas de vizinhança $NE.IGS$ e $NE.SIGS$, baseadas na fase construtiva e destrutiva da metaheurística IGS, não apresentam variações na forma de exploração. Em contrapartida, o parâmetro rj influencia diretamente na forma de exploração da mesma. Devido a isto, para as estruturas de vizinhança $NE.IGS$ e $NE.SIGS$, analisamos diferentes níveis do parâmetro rj no projeto das buscas locais baseadas em VND.

4.4 Metodologia Experimental

Nesta seção descrevemos toda a metodologia experimental utilizada neste capítulo sobre o projeto de operadores de busca local para problemas de minimização do TWT em ambientes UPMSMDST. Todos os experimentos realizados neste capítulo são baseados no conjunto de instâncias descritas na Seção 2.4.1. Desta forma, os resultados obtidos pelas abordagens IHM (Lin e Hsieh, 2014) e ILS (Diana et al., 2018) para o conjunto de instâncias são considerados estados da arte para o problema estudado.

4.4.1 Detalhes de implementação das metaheurísticas

Para realização dos experimentos propostos neste trabalho, foram implementas as metaheurísticas de estado da arte IGS (Lin et al., 2011), ABC (Ying e Lin, 2012) e GA (Zeidi e MohammadHosseini, 2015). Os algoritmos foram implementados na linguagem Java, com JDK 1.8. As implementações seguiram criteriosamente as diretrizes indicadas pelos autores nos respectivos manuscritos. Para garantir as boas práticas de pesquisa operacional propostas por Kendall et al. (2016), o código fonte e os resultados analíticos gerados pelos experimentos aqui descritos estão hospedados no domínio público `https://github.com/rodsxe/VND-UPMSSD`.

Uma vez que os experimentos realizados neste capítulo não realizam alterações substanciais nos operadores das metaheurísticas avaliadas, iremos descrever aqui apenas os pontos nos quais as mesmas foram alteradas. Em todas as metaheurísticas foram alterados a construção de soluções e os métodos de busca local, objeto de estudo deste trabalho. Neste trabalho optamos pela construção de soluções utilizadas pelas metaheurísticas IHM (Lin e Hsieh, 2014) e ILS (Diana et al., 2018), sendo estas as duas abordagens de estado da arte para o conjunto de instâncias utilizados nos experimentos. Assim, para as duas metaheurísticas populacionais, ABC e GA, o método consiste na construção de uma solução pela heurística gulosa ATCSR_Rm (Lin e Hsieh, 2014), e as demais soluções sendo geradas de forma aleatória. Já para a metaheurística de trajetória IGS, geramos a primeira solução pela abordagem ATCSR_Rm, em seguida a solução é explorada pelo IGS até não apresentar melhora. Os demais reinícios são explorados com a geração de soluções aleatórias.

Os métodos de busca local são substituídos pelas abordagens VND estudadas. O ponto de substituição é facilmente identificado, pois todas as metaheurísticas têm claramente uma fase em sua estrutura geral que indica o local exato em que os novos operadores são acoplados.

Apesar dos parâmetros das metaheurísticas poderem influenciar no desempenho das mesmas, optamos neste trabalho por não recalibrar os mesmos, mantendo os parâmetros definidos nos manuscritos nos quais as abordagens foram propostas. Definimos, assim, que a nossa metodologia deve fazer com que o operador de busca local proposto adapte-se aos parâmetros e operadores previamente calibrados.

Por fim, o problema estudado nos trabalhos em que as metaheurísticas ABC, GA e IGS são propostas não utiliza *job ready times*. Não foi necessária nenhuma alteração nas abordagens ABC, GA e IGS para considerar esta característica nos experimentos aqui realizados, sendo apenas necessário acrescentar esta característica no algoritmo que avalia a função objetivo de uma solução.

4.4.2 Ambiente Computacional

Os experimentos realizados neste capítulo foram executados em uma máquina equipada com um processado Intel Pentium E5200 @ 2.50GHz com 2GB de memória RAM e sistema operacional Linux Ubuntu 16.04. De acordo com o *Single Thread Performance* (STP) dado pela CPU *benchmarks* da PassMark Software[1], a configuração desta máquina é, aproximadamente, o mesmo *hardware* utilizado no trabalho

[1]Disponível em https://www.cpubenchmark.net/singleThread.html. Acessado em 25 de agosto de 2018.

que obteve os resultados das abordagens IHM (Lin e Hsieh, 2014) e ILS (Diana et al., 2018). Assim, em todos os experimentos realizados neste capítulo, utilizados o mesmo critério de parada utilizado pela abordagem IHM, sendo este 1 e 60 segundos, respectivamente, para as instâncias do tipo *small* e *large* (Seção 2.4.1). Para tornar ainda mais próximo o critério de parada com o da abordagem IHM, reescalamos o tempo de processamento pelo fator STP. Este é o mesmo critério de parada utilizado pela abordagem ILS (Diana et al., 2018). Tempos computacionais reescalados pelo STP são bastante utilizados na literatura de pesquisa operacional. Para maiores informações sobre este reescalonamento ver os trabalhos de Santos et al. (2019); Silvestrin e Ritt (2017); Kramer e Subramanian (2019) e Corstjens et al. (2019).

4.4.3 Metodologia para projeto dos operadores de busca local

A Hipótese (H1) avaliada neste trabalho indica que a variação do VND, ordem e forma que as estruturas de vizinhança são exploradas, incrementam ou decrementam o desempenho do operador de busca local de uma metaheurística. A avaliação desta hipótese permite analisar detalhadamente o operador de busca local, e escolher, dentre diversos fatores, as estratégias que potencializem a função do operador de busca local no desempenho uma metaheurística. Para a utilização de um projeto de experimentos (DoE) do tipo fatorial completo, são necessários $2 \times 3^7 r = 4374r$ experimentos, sendo r o número do replicações para cada combinação. Este projeto experimental se mostra inviável para o problema estudado. Neste trabalho utilizou-se a metodologia de calibração robusta de parâmetros proposta por Taguchi (Taguchi, 1986, 1987). O método de Taguchi utiliza um DoE do tipo fatorial fracionado, que utiliza arranjos ortogonais. Estes arranjos tendem a reduzir o número de experimentos realizados. Em contrapartida, nesta metodologia não é possível observar as interações entre os fatores. Esta metodologia é utilizada para a definição de fatores controláveis de um processo, para que tornem a operação menos sensível a fatores não controláveis, denominados *Noise Factors*. Este projeto de experimentos é uma ferramenta reconhecidamente eficaz na literatura de controle de qualidade de processos, e recentemente vem sendo utilizada para a calibração de metaheurísticas (Zandieh et al. (2009); Thepphakorn et al. (2014); Rambod e Rezaeian (2014); Zhao et al. (2019)).

Para a definição do VND para cada uma das metaheurísticas, avaliou-se oito fatores controláveis, conforme mostrado na Tabela 4.1. O fator Ordem é composto por dois níveis ($NE.O_{card}$ e $NE.O_{imp}$), definidos na Seção 4.3.2.1. Os demais fatores são compostos por três níveis. O fator Variação VND apresenta as três variações do VND mostradas na Seção 4.3.2.2, a saber, *Basic-VND*, *Random-VND*, e *Union-VND*. Os seis fatores restantes representam as formas de exploração das estruturas de vizinhança. Para os fatores NE.II, NE.IS, NE.EI e NE.ES, o nível NA indica que a estrutura de vizinhança não será utilizada. Já os níveis FI e BI indicam, respectivamente, as estratégias de busca *First Improvement* e *Best Improvement*, conforme descrito na Seção 4.3.2.3. Já para os fatores NE.IGS e NE.SIGS, os quais não utilizam as estratégias FI e BI, o segundo e terceiro nível indicam o parâmetro rj usado na estrutura de vizinhança.

Como os experimentos propostos nesta seção pretendem calibrar oito fatores controláveis, sete fatores com três níveis, e um valor com dois níveis, é utilizado o arranjo ortogonal $L_{18}(2^1 \times 3^7)$. Assim, são necessários $18r$ experimentos para calibração dos

Tabela 4.1: Lista de fatores controláveis para projeto do VND como operador de busca local.

			Fatores Controláveis					
Níveis	Ordem	Variação VND	NE.II	NE.IS	NE.EI	NE.ES	NE.IGS	NE.SIGS
1	$NE.O_{card}$	*Basic-VND*	NA	NA	NA	NA	NA	NA
2	$NE.O_{imp}$	*Random-VND*	FI	FI	FI	FI	2	1
3	–	*Union-VND*	BI	BI	BI	BI	3	2

fatores controláveis. Para definir o número de réplicas r para cada experimento, consideramos que os experimentos propostos neste capítulo são utilizados para resolver dezesseis cenários distintos (*noise factors* $\eta \times \tau \times R \times r_\tau$, Seção 2.4.1) e que cada cenário ainda é compostos por ruídos aleatórios. Assim, foram gerados, para cada cenário, duas instâncias, seguindo as diretrizes definidas na Seção 2.4.1, totalizando assim trinta e duas réplicas para cada experimento ($r = 32$). O projeto experimental foi gerado no *software* `R` pelo pacote *Quality Tools* [2].

A avaliação do desempenho dos fatores controláveis na metodologia de Taguchi é dado pelo indicador *signal-to-noise ratio* (S/N_{ratio}). O S/N_{ratio} é derivado das respostas resultantes de cada uma das combinações dos parâmetros. Este indicador é uma medida de qualidade dos fatores de controle, de modo que a combinação ótima dos fatores deve minimizar a resposta média e a variabilidade dos resultados. Neste trabalho, como se deseja atingir o menor valor possível do TWT, utilizou-se, para cálculo do S/N_{ratio}, a função de perda *"smaller the better"*, expressa por:

$$S/N_{ratio} = -10 \log \left(\frac{1}{r} \sum_{i=1}^{r} y_i^2 \right), \tag{4.4}$$

sendo y_i a resposta de uma execução do algoritmo proposto para uma determinada replicação. Como há variação na escala dos valores de resposta devido aos fatores não controláveis, como o número de tarefas ou máquinas, optou-se por normalizar as respostas pelo desvio percentual relativo (RDP), dado por:

$$y_i = \frac{1}{10} \sum_{j=1}^{10} \frac{f(i,j) - best_{f(i)}}{best_{f(i)}}, \tag{4.5}$$

sendo $f(i,j)$ a resposta da instância i para a execução independente j; $best_{f(i)}$ a resposta mínima para a instância i encontrada após todos os experimentos desta seção. Além disto, y_i é calculado usando 10 execuções independentes para cada réplica (instância), e isto minimiza, na resposta, o impacto estocástico das metaheurísticas.

Uma vez calculado o S/N_{ratio} para cada nível de cada fator, deve-se considerar que, quanto maior o indicador S/N_{ratio}, mais o nível contribui no desempenho do VND. Assim, para cada metaheurística avaliada, a análise da influência dos níveis de cada fator é realizada por gráficos de efeitos principais do indicador S/N_{ratio}. Os resultados desta análise são apresentados na Seção 4.5.1.

4.4.4 Análise de estruturas de vizinhança

O estudo realizado por Fawcett e Hoos (2016) enfatiza que um grande número de trabalhos destinados ao desempenho de algoritmos de alta performance, para proble-

[2] Disponível em http://www.r-qualitytools.org/

mas computacionais complexos, costumam adotar *frameworks* de calibração de parâmetros. Em contrapartida, poucos destes trabalhos realizam estudos para analisar a real contribuição destes parâmetros/componentes no desempenho destes algoritmos. Para não cair neste cenário, nesta seção detalhamos a metodologia usada para avaliar a segunda hipótese estudada neste capítulo, buscando entender a influência de cada estrutura de vizinhança utilizada no operador de busca local para a convergência de cada metaheurística. A configuração do VND definida pela metodologia descrita na Seção 4.4.3 e sumarizada na Seção 4.5 pela Tabela 4.2 é considerada na metodologia proposta nesta seção como o controle experimental. Assim, os experimentos propostos nesta seção são conduzidos através de variações na versão controle do VND de cada metaheurística.

Os experimentos para a análise das estruturas de vizinhança são realizadas com o conjunto de trinta e duas instâncias geradas para o projeto do VND (Seção 4.4.3). Para cada variação do VND avaliada, são executadas dez execuções independentes para cada instância. Os resultados do TWT são analisados sobre um *fitness* médio normalizado, dado por:

$$Avg.Norm.Fitness = \frac{1}{10 \times r} \sum_{i=1}^{r} \sum_{j=1}^{10} \frac{f(i,j) - best_{f(i)}}{worst_{f(i)} - best_{f(i)}}, \tag{4.6}$$

sendo r o número de instâncias avaliadas ($r = 32$); 10 é o número de execuções independentes para cada instância i; $f(i,j)$ representa o menor TWT encontrado para a instância i e execução independente j; $best_{f(i)}$ e $worst_{f(i)}$ representam, respectivamente, o menor e maior TWT encontrado para a instância i após todos os experimentos realizados nesta seção.

A partir destas definições, para avaliar a influência das estruturas de vizinhança no desempenho do VND como operador de busca local de cada metaheurística avaliada, propusemos duas análises, denominadas análise de *ablation* e remoção de componentes.

A análise de *ablation* (Fawcett e Hoos, 2016) consiste na transformação de um algoritmo θ_{source} em um algoritmo alvo θ_{target}, através da inserção de todos os componentes presentes no conjunto S_{comp}. A transformação é realizada através de $|S_{comp}|$ passos. A cada passo p, são avaliados a inserção de todos os componentes presentes em S_{comp} no algoritmo θ_{st_p}, sendo $\theta_{st_0} = \theta_{source}$. O componente que acarretar no maior ganho de desempenho de θ_{st_p} é selecionada para ingressar neste, gerando um algoritmo intermediário, $\theta_{st_{p+1}}$. O componente selecionado é removido de S_{comp} e repete-se o processo até que S_{comp} se torne um conjunto vazio. Considerando que θ_{source} apresenta pior desempenho que θ_{target}, em geral, os passos iniciais tendem a selecionar os componentes de transformação que acarretam nas maiores contribuições na convergência do algoritmo θ_{target}.

Para a realização da análise de *ablation* pretendida nesta seção, para cada metaheurística, IGS, ABC, and GA, consideramos como algoritmo alvo (θ_{target}) a versão com o VND de controle experimental. Já como algoritmo de partida θ_{source}, consideramos a metaheurística integrada a um VND sem estruturas de vizinhança, ou seja, sem busca local. Assim, o conjunto de componentes S_{comp} de transformação de $\theta_{source} \rightarrow \theta_{target}$ é definido pelo conjunto de estruturas de vizinhança presentes no VND de controle experimental da metaheurística avaliada. O primeiro passo (st_1) da

análise consiste em adicionar, isoladamente, cada estrutura de vizinhança, no algoritmo θ_{source}. A configuração que adiciona maior eficiência à metaheurística (reduza o $Avg.Norm.Fitness$, Equação 4.6) é incorporada ao algoritmo θ_{source}, gerando uma nova versão, θ_{st_1}. A estrutura de vizinhança incorporada em θ_{st_1} é removida de S_{comp}. Em seguida, no segundo passo (st_2), é repetida a lógica do primeiro passo, considerando, como algoritmo de entrada, θ_{st_1}, e gerando o algoritmo θ_{st_2}. Este processo é repetido até que S_{comp} se torne vazio, ou seja, todas as estruturas de vizinhança sejam inseridas no VND, gerando o algoritmo alvo θ_{target}. Para uma melhor visualização desta análise, os resultados gerados pelos experimentos são plotados para cada metaheurística em um gráfico de convergência do $Avg.Norm.Fitness$ por passo da análise. Os resultados destes experimentos são apresentados na Seção 4.5.2.

Adicionalmente à análise de *ablation*, foi conduzido um conjunto de experimentos para comparar a convergência de cada uma das metaheurísticas com o operador de busca local de controle (VND + ALL), contra o mesmo operador de busca local, mas retirando cada uma das estruturas de vizinhança e gerando uma nova variação do operador. Estes experimentos contribuem como uma análise complementar para entender o impacto das estruturas de vizinhança na convergência das metaheurísticas. Esta análise é ilustrada na Seção 4.5.2 através de gráficos de convergência no tempo do $Avg.Norm.Fitness$.

4.4.5 Comparação entre metaheurísticas

Para verificar a eficiência do VND como operador de busca local das metaheurísticas IGS, ABC e GA, são realizados experimentos para cada metaheurística, considerando duas configurações das mesmas, sendo estas:

(i) metaheurística + operador nativo de busca local;

(ii) metaheuristica + VND resultante da metodologia descrita na Seção 4.4.3.

Esta comparação é realizada através do conjunto de instâncias proposto por Lin e Hsieh (2014) e descrito na Seção 2.4.1. Todas as comparações são realizadas por agrupamento de instâncias $\eta \times \tau \times R \times r_\tau$. Para cada grupo de instâncias, é calculado o desvio percentual relativo médio ($\overline{RPD}$), dado por:

$$\overline{RPD} = \frac{1}{N_{gi} \times 10} \sum_{i=1}^{N_{gi}} \sum_{j=1}^{10} \frac{f(i,j) - best_{f(i)}}{best_{f(i)}}, \tag{4.7}$$

sendo N_{gi} o número de instâncias do grupo $\eta \times \tau \times R \times r_\tau$; 10 o número de execuções independentes de cada configuração de metaheurística para cada instância i; $f(i,j)$ representa o TWT encontrado para a instância i e execução independente j; e $best_{f(i)}$ representa o menor TWT previamente conhecido na literatura para a instância i.

Para o subconjunto de instâncias denominado *small*, os resultados são comparados diretamente pelo $\overline{RPD}$, já que este conjunto apresenta os resultados ótimos encontrados por um MIP. Já para as instâncias *large*, as abordagens são comparados pelo $\overline{RPD}$, por gráficos *boxplots* e pelo teste de inferência estatística *Wilcoxon signed-rank test* pareado (Wilcoxon, 1945). O teste de Wilcoxon é realizado para verificar se os resultados obtidos pela metaheurística com operador de busca local VND apresentam

diferença estatística significante quando comparados à versão da metaheurística com a busca local nativa. Este teste é executado considerando as seguintes hipóteses:

$$H_0 : \quad md_1 - md_2 = 0 \tag{4.8}$$
$$H_1 : \quad md_1 - md_2 < 0, \tag{4.9}$$

sendo md_1 e md_2, respectivamente, a mediana obtida pela metaheurística integrada ao VND e a mediana da abordagem com busca local nativa (IGS, ABC, GA). É utilizado um teste pareado para minimizar os efeitos aleatórios não controláveis presentes nas instâncias.

Para comparar o desempenho das novas metaheurísticas geradas com os operadores de busca local VND contra abordagens de estado da arte (i.e., IHM - (Lin e Hsieh, 2014) e ILS - (Diana et al., 2018)), são realizados dois *Wilcoxon signed-rank test* pareados. O primeiro considera a hipótese que a abordagem proposta com VND apresenta resultados inferiores ao das abordagens de estado da arte. Neste caso, $H_0 : md_1 - md_2 = 0$ e $H_1 : md_1 - md_2 < 0$. Já no segundo cenário, verifica-se a hipótese que as metaheurísticas integradas ao VND apresenta resultados inferiores às abordagens de estado da arte. Neste caso, $H_0 : md_1 - md_2 = 0$ e $H_1 : md_1 - md_2 > 0$. Em ambos os cenários, md_1 representa a mediana da metaheurística integrada ao VND e md_2 os resultados obtidos pela abordagem de estado da arte. Por fim, foi realizado um teste de inferência estatística de Friedman, usado para comparar o desempenho geral dos algoritmos.

Todos os resultados obtidos pelos testes de inferência estatística são sumarizados em tabelas. Apenas o *p-valor* dos respectivos testes são apresentados. Todos os testes foram conduzidos considerando um intervalo de confiança de 95% ($\alpha = 0.05$). Todos os testes estatísticos são conduzidos na versão 3.2.3 do R-project para computação estatística, e são apresentados na Seção 4.5.3.

4.5 Resultados

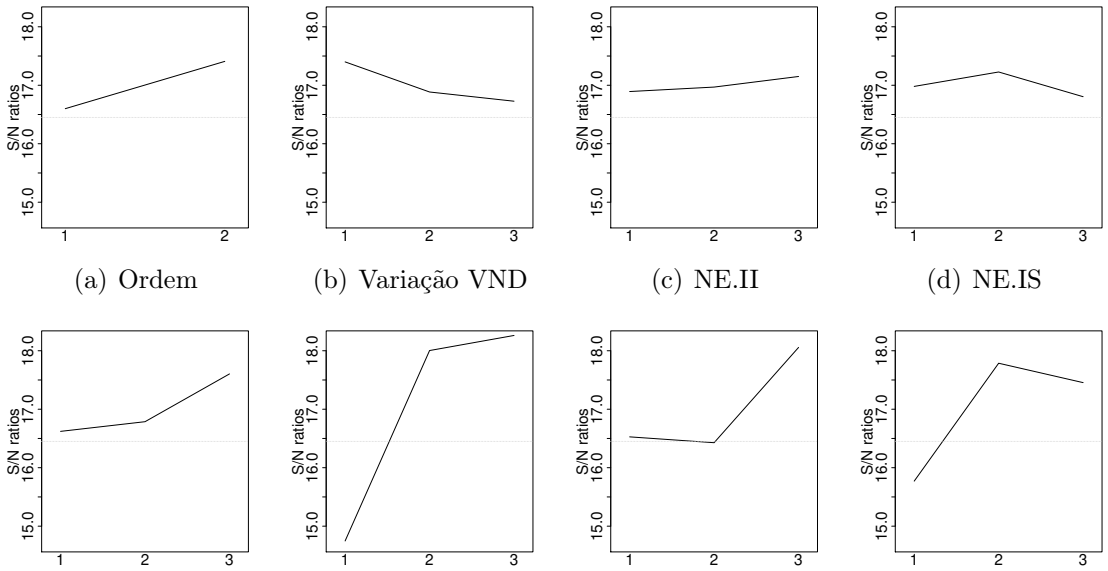
Nesta seção são apresentados os resultados dos experimentos do VND como operador de busca local das metaheurísticas IGS, ABC e GA. A Seção 4.5.1 apresenta, para cada metaheurística, os resultados para seleção da variação do VND, ordem e forma de exploração das estruturas de vizinhança, que acarretam em melhor desempenho do VND como operador de busca local. A Seção 4.5.2 apresenta os resultados que verificam o impacto das estruturas de vizinhança na resposta das abordagens heurísticas. Por fim, a Seção 4.5.3 verifica a hipótese de que a substituição da busca local nativa pela busca local VND acarreta em incremento do desempenho das abordagens IGS, ABC e GA. Nesta seção ainda são comparados os resultados gerados pelas abordagens que incorporam o VND em relação às abordagens de estado da arte para o problema.

4.5.1 Seleção de componentes do VND

Os gráficos de efeitos principais expressos nas Figuras 4.2, 4.3 e 4.4 apresentam, respectivamente, os efeitos dos fatores controláveis do VND no desempenho das me-

taheurísticas IGS, ABC e GA. Um sumário da configuração selecionada do VND para cada metaheurística é apresentado no Tabela 4.2.

Para a metaheurística IGS (Figura 4.2), a estrutura de vizinhança NE.ES é o fator que acarreta maior impacto (i.e. variação do efeito entre os níveis) na resposta da abordagem. A forma de busca *Best Improvement* é selecionada como a melhor estratégia de exploração da NE.ES. A heurística BI também é selecionada para as estruturas NE.II e NE.EI. A estrutura NE.IGS também acarreta em grande influência na resposta quando selecionado o nível $rj = 3$. A ordem de exploração se mostrou importante nesta abordagem, uma vez que, dentre as variações do VND, o *Basic-VND* é a abordagem com maior S/N_{ratio} em conjunto com a ordem $NE.O_{imp}$.



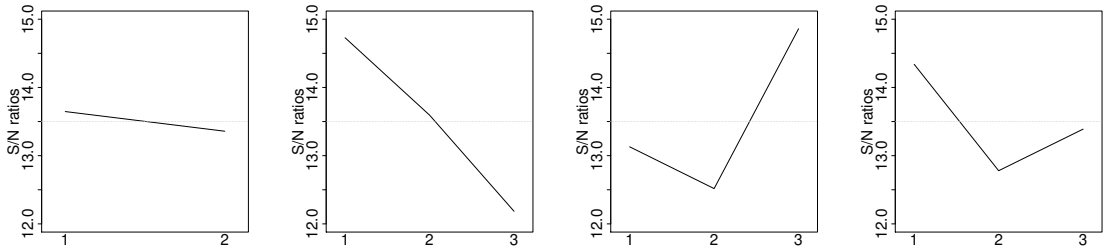
(a) Ordem (b) Variação VND (c) NE.II (d) NE.IS

(e) NE.EI (f) NE.ES (g) NE.IGS (h) NE.SIGS

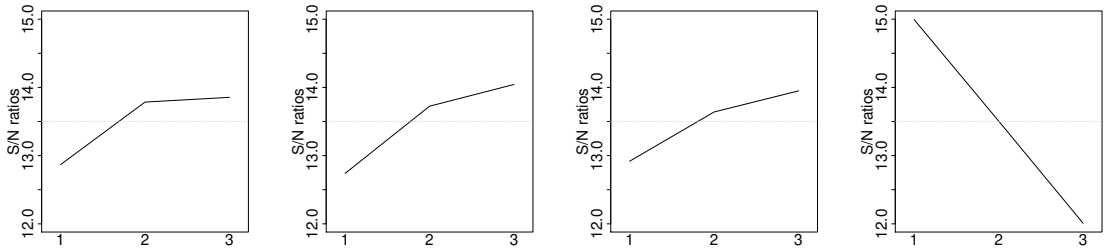
Figura 4.2: S/N_{ratio} para os fatores controláveis do VND com operador de busca local do IGS.

A variação do VND é o fator que apresenta maior impacto no desempenho da abordagem ABC (Figura 4.3). Para esta metaheurística, a abordagem *Basic-VND* é selecionada em conjunto com a ordem $NE.O_{card}$. Os experimentos indicam que as estruturas de vizinhança NE.IS e NE.SIGS deterioram o desempenho do VND como operador de busca local. A forma de busca *Best Improvement* é selecionada para as estruturas NE.II, NE.EI e NE.ES, tendo a estrutura NE.II acarretando em alto impacto no desempenho da metaheurística ABC. Já para estrutura NE.IGS, o nível $rj = 3$ é selecionado.

As estruturas de vizinhança NE.EI e NE.IGS são os fatores do VND que acarretam maior impacto na resposta da metaheurística GA (Figura 4.4). Assim como nas abordagem IGS e ABC, o nível $rj = 3$ é selecionado para a estrutura NE.IGS. Já para NE.EI, a forma *Best Improvement* é selecionada, esta também sendo o nível selecionado para estrutura NE.II. A forma *First Improvement* é selecionada para as estruturas de troca NE.IS e NE.ES . A ordem de exploração das estruturas também se mostrou importante para a abordagem GA, sendo selecionada a ordem $NE.O_{imp}$ em conjunto com a variação *Basic-VND*.

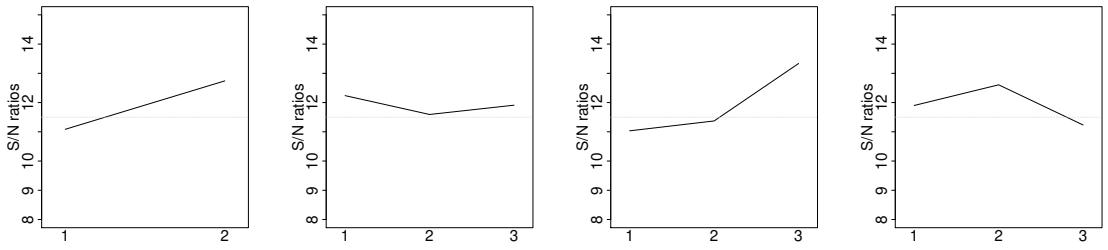


(a) Ordem (b) Variação VND (c) NE.II (d) NE.IS

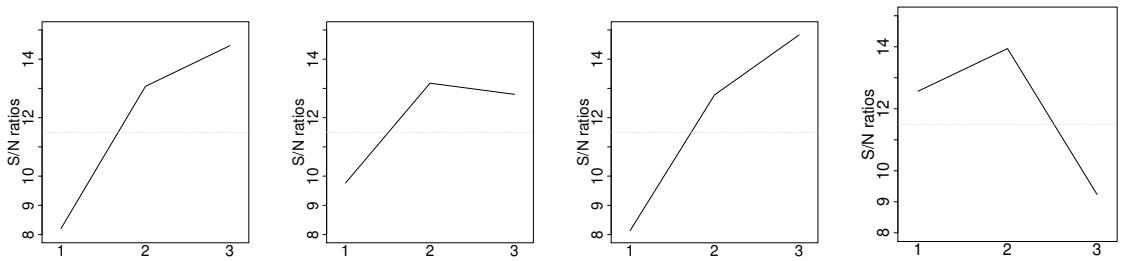


(e) NE.EI (f) NE.ES (g) NE.IGS (h) NE.SIGS

Figura 4.3: S/N_{ratio} para os fatores controláveis do VND com operador de busca local do ABC.



(a) Ordem (b) Variação VND (c) NE.II (d) NE.IS



(e) NE.EI (f) NE.ES (g) NE.IGS (h) NE.SIGS

Figura 4.4: S/N_{ratio} para os fatores controláveis do VND com operador de busca local do GA.

4.5.2 Análise de estruturas de vizinhança

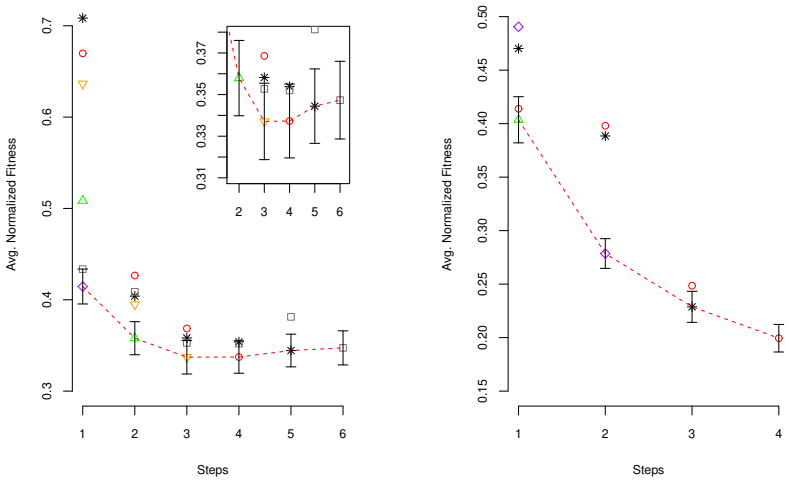
As Figuras 4.5(a), 4.5(b) e 4.5(c) indicam que as estruturas de vizinhança são divididas em dois grupos. O primeiro grupo, composto por NE.IGS, NE.ES e NE.EI, é responsável pela maior taxa de convergência dos algoritmos. Deste grupo, as estruturas NE.IGS e NE.ES devem ser destacadas. Os experimentos indicaram um grau de

Tabela 4.2: Sumário da configuração do VND desenhado para cada metaheurística.

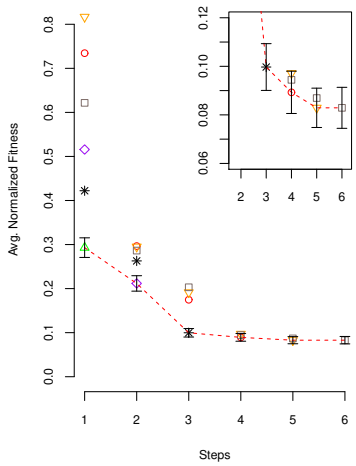
Fator	IGS	ABC	GA
Ordem	$NE.O_{imp}$	$NE.O_{card}$	$NE.O_{imp}$
Variação VND	*Basic-VND*	*Basic-VND*	*Basic-VND*
NE.II	BI	BI	BI
NE.IS	FI	NA	FI
NE.EI	BI	BI	BI
NE.ES	BI	BI	FI
NE.IGS	3	3	3
NE.SIGS	1	NA	1

complementariedade entre elas. Observando os experimentos das abordagens ABC (Figura 4.5(b)) e GA (Figura 4.5(c)), em ambos, no primeiro passo, a vizinhança NE.IGS é selecionada para ingressar no VND. Deve-se observar que, em ambos os experimentos, a vizinhança NE.ES não é a segunda opção de seleção no primeiro passo, mas, devido à incorporação da NE.IGS ao VND, se transforma na primeira opção no segundo passo. O mesmo ocorre, no sentido inverso, nos experimentos destinados à metaheurística IGS (Figura 4.5(a)). Nesta, no primeiro passo, a NE.ES é selecionada para ingressar no VND. Já no passo seguinte, a vizinhança NE.IGS é selecionada, mesmo esta não sendo a segunda opção no passo anterior. Já a estrutura NE.EI parece agir de forma independente, mas incorporando significativa contribuição na convergência das abordagens em todos os cenários avaliados. Já o segundo grupo de estruturas de vizinhança é formado pelas estruturas NE.II, NE.IS e NE.SIGS, sendo este responsável pelo ajuste fino. Em geral, a inserção das estruturas de vizinhança deste grupo resultou no incremento discreto do desempenho das abordagens. Na maioria dos casos, a vizinhança NE.SIGS tende a não afetar a busca. Em contrapartida, a estrutura NE.II, em geral é a responsável por adicionar a maior contribuição dentre as vizinhanças deste grupo.

Por sua vez, as análises apresentadas nas Figuras 4.6(a), 4.6(b), e 4.6(c) permitem verificar o impacto na convergência das metaheurísticas, em função do tempo, acarretado pela remoção de estruturas de vizinhança. Podemos observar que a vizinhança NE.ES desempenha papel fundamental no processo de busca, uma vez que a remoção desta estrutura acarreta em alto impacto no desempenho de todas as abordagens. Por outro lado, observamos que, para as metaheurísticas ABC e GA, a remoção da estrutura NE.IGS acarretou em baixo impacto no resultado final. Deve-se lembrar que, pela análise de *ablation*, a NE.IGS em conjunto com a NE.ES acarretaram em maior incremento no desempenho de ambas as abordagens. Este resultado pode indicar que, na ausência da NE.IGS, outras estruturas de ajuste fino podem estar compensando a ausência da mesma. Mesmo isto ocorrendo para as abordagens ABC e GA, para a abordagem IGS, a estrutura NE.IGS continua como peça fundamental da estratégia de busca, e a sua remoção acarretando em grande impacto na convergência da abordagem. Já a remoção da estrutura NE.EI tem maior impacto nas metaheurísticas ABC e GA, e, em contrapartida, pouca influência na convergência da abordagem IGS. Já para as estruturas consideradas de ajuste fino (NE.II, NE.IS, NE.SIGS), os experimentos indicaram que a remoção de NE.II tende a acarretar em significativa degeneração da convergência. Estes resultados coincidem, em parte, com as conclusões obtidas pelas análises de *ablation* (Figure 4.5).



(a) IGS (b) ABC*

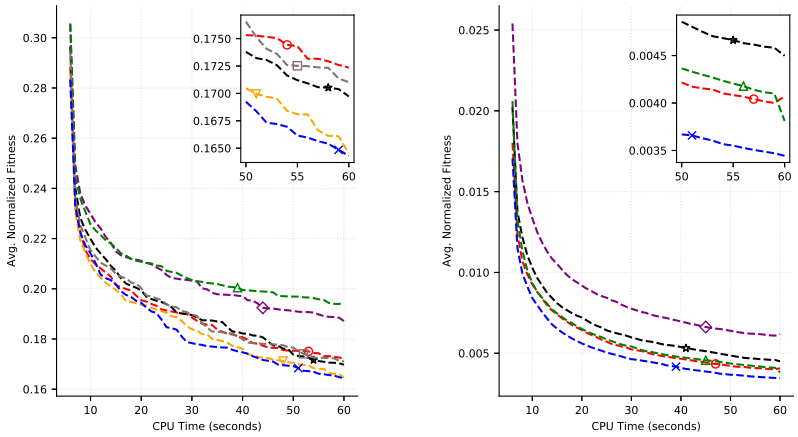


(c) GA

Figura 4.5: Análise de *ablation* do comportamento das estruturas de vizinhança para cada metaheurística. Cada ponto indica o desempenho da estrutura de vizinhança em cada passo. A linha ligando as estruturas de vizinhança indica os componentes que acarretaram em maior contribuição na convergência da metaheurística em cada passo. *A calibração pela abordagem de Taguchi indicou que as estruturas de vizinhança NE.IS e NE.SIGS deterioraram o desempenho do VND integrado á abordagem ABC; assim, a influência destas duas estruturas não é analisada para a metaheurística ABC.

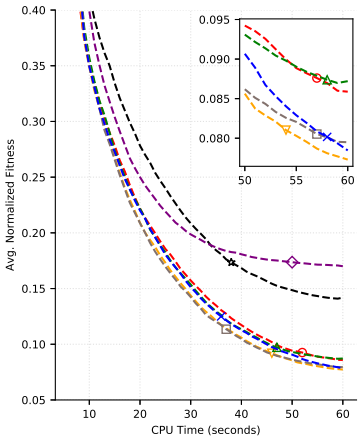
4.5.3 Comparações entre metaheurísticas

Os resultados para o grupo de instâncias *small* são apresentados na Tabela 4.3. Para este conjunto de instâncias, a substituição da busca local nativa pelos operadores de busca local VND, sumarizados na Tabela 4.2, incrementaram o desempenho das metaheurísticas. Para todas as abordagens analisadas (IGS, ABC, and GA), a incorporação do VND encontrou os resultados ótimos, para todas as instâncias do



(a) IGS

(b) ABC*



(c) GA

Figura 4.6: Gráficos de convergência para cada metaheurística. Cada configuração do VND considera a remoção de uma estrutura de vizinhança do VND afim de verificar a sua influência na convergência da metaheurística. *A calibração pela abordagem de Taguchi indicou que as estruturas de vizinhaça NE.IS e NE.SIGS deterioraram o desempenho do VND integrado à abordagem ABC; assim, a influência destas duas estruturas não é analisada para a metaheurística ABC.

conjunto *small*. Deve-se observar que as abordagens nativas ABC e GA apresentaram desempenho inferior à abordagem IHM (i.e., metaheurística estado da arte para o conjunto) em diversos grupos de instâncias. Já as mesmas abordagens (ABC e GA), quando integradas ao VND, têm desempenho igual a IHM para todos as instâncias com tempos de *setup* irrelevantes ($\eta = 0.02$), e superior para todos os grupos em que os tempos de setup são relevantes ($\eta = 2.00$). Não comparamos para este conjunto

de instâncias a abordagem estado da arte ILS (Diana et al., 2018), já que a mesma não apresenta resultados para este conjunto.

O tempo de convergência é outro aspecto importante a ser observado no conjunto *small*. A incorporação do VND fez com que as abordagens baseadas em metaheurísticas atinjam todos os resultados ótimos em 1 segundo (i.e. critério de parada Seção 4.4.2). A definição destes pontos ótimos pelo MIP demandaram em média 2191,72 segundos (Lin e Hsieh, 2014).

Tabela 4.3: $\overline{RPD}$ alcançado por cada metaheurística para o conjunto de instâncias *small*.

$12n3m$				MIP	State of art		IGS		ABC		GA	
η	τ	R	r_τ	TWT	IHM	ILS	LS	VND	LS	VND	LS	VND
0.02	0.30	0.25	1.00	4.284.35	**0.00%**	NA	**0.00%**	**0.00%**	**0.00%**	**0.00%**	1.99%	**0.00%**
			10.00	1.234.10	**0.00%**	NA	**0.00%**	**0.00%**	**0.00%**	**0.00%**	14.87%	**0.00%**
		1.00	1.00	2.825.10	**0.00%**	NA	**0.00%**	**0.00%**	**0.00%**	**0.00%**	5.00%	**0.00%**
			10.00	902.60	**0.00%**	NA	**0.00%**	**0.00%**	**0.00%**	**0.00%**	13.14%	**0.00%**
	0.90	0.25	1.00	6.664.25	**0.00%**	NA	**0.00%**	**0.00%**	**0.00%**	**0.00%**	0.66%	**0.00%**
			10.00	6.575.70	**0.00%**	NA	**0.00%**	**0.00%**	**0.00%**	**0.00%**	0.51%	**0.00%**
		1.00	1.00	6.749.55	**0.00%**	NA	**0.00%**	**0.00%**	**0.00%**	**0.00%**	1.08%	**0.00%**
			10.00	6.877.55	**0.00%**	NA	**0.00%**	**0.00%**	**0.00%**	**0.00%**	0.94%	**0.00%**
2.00	0.30	0.25	1.00	7.452.80	0.57%	NA	**0.00%**	**0.00%**	0.20%	**0.00%**	8.44%	**0.00%**
			10.00	781.35	1.05%	NA	**0.00%**	**0.00%**	1.67%	**0.00%**	102.11%	**0.00%**
		1.00	1.00	4.433.35	**0.00%**	NA	**0.00%**	**0.00%**	0.13%	**0.00%**	10.68%	**0.00%**
			10.00	439.65	0.40%	NA	**0.00%**	**0.00%**	0.56%	**0.00%**	874.88%	**0.00%**
	0.90	0.25	1.00	9.091.75	0.19%	NA	**0.00%**	**0.00%**	0.01%	**0.00%**	4.86%	**0.00%**
			10.00	8.904.35	0.08%	NA	**0.00%**	**0.00%**	0.05%	**0.00%**	5.55%	**0.00%**
		1.00	1.00	9.532.05	**0.00%**	NA	**0.00%**	**0.00%**	0.06%	**0.00%**	5.56%	**0.00%**
			10.00	9.322.50	0.02%	NA	**0.00%**	**0.00%**	0.05%	**0.00%**	6.74%	**0.00%**

Ao analisar os resultados da metaheurística IGS para o conjunto de instâncias *large*, expressos nas Tabelas 4.4, 4.5 e Figura 4.7, conclui-se que o IGS que incorpora a busca local VND (IGS-VND) apresenta resultados superiores à versão nativa (coluna IGS-LS) em 15 dos 16 cenários avaliados. Apenas no cenário $0.02 \times 0.3 \times 1.00 \times 10$ não há diferença estatística significativa ($p-value < 0.05$). Comparando o IGS-VND em relação às abordagens de estado da arte, este apresenta-se significativamente superior a IHM para 9 cenários. Já a hipótese do IGS-VND apresentar resultados inferiores ao IHM é satisfeita em 4 cenários, 3 para grupos de instâncias em que os tempos de *setup* são relevantes ($\eta = 2.00$) e datas de entrega são próximas entre si ($\tau = 0.9$). Já quando compara-se o IGS-VND ao ILS, o primeiro apresenta-se superior para 7 grupos de instâncias. O IGS-VND ainda apresenta resultados inferiores ao ILS em 8 cenários, sendo 7 em situações nas quais os tempos de *setup* são relevantes ($\eta = 2.00$).

Os resultados alcançados pela a abordagem ABC indicam que a incorporação do VND como operador de busca local acarretou no incremento do desempenho da metaheurística para todos os grupos do conjunto de instâncias *large* (Tabelas 4.4 e 4.6, Figura 4.7). O ABC-VND (ABC com operador de busca local VND) também apresentou resultados superiores às abordagens estado da arte IHM e ILS em todos os cenários avaliados. Analisando a Tabela 4.4 e a Figura 4.7, pode-se observar que o maior incremento do desempenho do ABC-VND contra as abordagens de estado da arte ocorre nos cenários em que os tempos de *setup* são relevantes ($\eta = 2.00$), as datas de entrega são mais distantes entre si ($\tau = 0.3$) e os *job ready times* são mais distantes das datas de entrega ($r_\tau = 10$).

A metaheurística GA, tal como proposta por Zeidi e MohammadHosseini (2015),

Tabela 4.4: $\overline{RPD}$ alcançado por cada metaheurística para o conjunto de instâncias *large*.

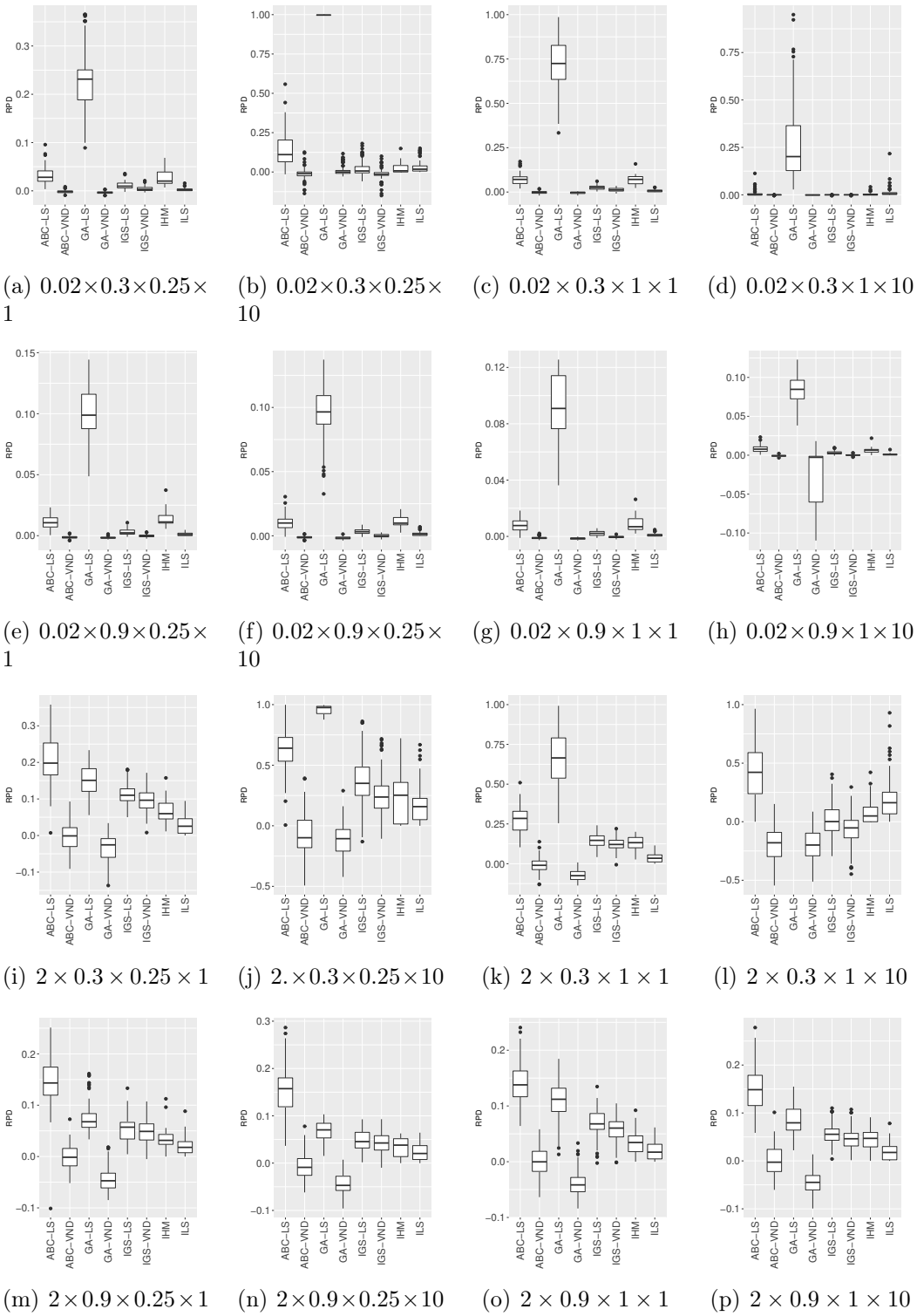
$100n \times 10m$				State of art		IGS		ABC		GA	
η	τ	R	r_τ	IHM	ILS	LS	VND	LS	VND	LS	VND
0.02	0.3	0.25	1	2.59%	0.26%	1.09%	0.42%	3.15%	-0.17%	22.77%	**-0.35%**
			10	2.88%	3.07%	2.17%	**-1.47%**	14.32%	-1.03%	281.60%	0.32%
		1.00	1	7.15%	0.74%	2.58%	1.28%	7.03%	-0.20%	75.09%	**-0.74%**
			10	0.60%	1.19%	**-0.03%**	**-0.03%**	0.65%	**-0.03%**	26.93%	**-0.03%**
	0.9	0.25	1	1.43%	0.12%	0.30%	-0.02%	1.08%	-0.13%	10.17%	**-0.19%**
			10	1.09%	0.15%	0.33%	0.00%	1.03%	-0.12%	9.68%	**-0.16%**
		1.00	1	0.89%	0.10%	0.23%	-0.04%	0.78%	-0.11%	9.34%	**-0.18%**
			10	0.64%	0.10%	0.31%	0.02%	0.83%	-0.08%	8.52%	**-0.13%**
2.00	0.3	0.25	1	6.86%	2.84%	11.17%	9.47%	20.63%	-0.17%	15.27%	**-5.59%**
			10	23.06%	16.91%	37.10%	25.40%	77.46%	-7.73%	188.05%	**-11.46%**
		1.00	1	12.72%	3.58%	14.48%	12.30%	27.24%	-0.89%	68.32%	**-7.17%**
			10	9.40%	22.16%	2.24%	-6.42%	52.03%	-19.23%	351.82%	**-20.10%**
	0.9	0.25	1	3.64%	2.02%	5.38%	4.83%	14.79%	-0.30%	7.26%	**-4.40%**
			10	3.16%	2.30%	4.69%	4.12%	15.22%	-0.68%	6.76%	**-4.51%**
		1.00	1	3.65%	2.00%	7.01%	5.74%	14.13%	-0.06%	11.33%	**-3.87%**
			10	4.39%	1.90%	5.66%	4.63%	15.08%	0.07%	8.84%	**-4.33%**

Tabela 4.5: Teste pareado de Wilcoxon por grupo de instâncias comparando o IGS-VND em relação ao IGS nativo e as abordagens estado da arte.

$100n10m$				$H_1 : md_1 < md_2$*			$H_1 : md_1 > md_2$*	
η	τ	R	r_τ	IGS-LS	IHM	ILS	IHM	ILS
0.02	0.3	0.25	1	**1.15E-14**	**9.54E-07**	9.34E-01	1.00E+00	7.15E-02
			10	**2.24E-14**	**2.50E-04**	**9.54E-07**	1.00E+00	1.00E+00
		1.00	1	**2.78E-14**	**9.54E-07**	9.98E-01	1.00E+00	**1.83E-03**
			10	5.00E-01	**1.80E-02**	**1.91E-06**	9.89E-01	1.00E+00
	0.9	0.25	1	**6.66E-18**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00
			10	**1.06E-17**	**9.54E-07**	**9.54E-06**	1.00E+00	1.00E+00
		1.00	1	**3.95E-18**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00
			10	**4.87E-18**	**9.54E-07**	**9.54E-06**	1.00E+00	1.00E+00
2.00	0.3	0.25	1	**1.27E-10**	1.00E+00	1.00E+00	**4.25E-04**	**9.54E-07**
			10	**1.38E-09**	5.80E-01	1.00E+00	4.35E-01	**4.25E-04**
		1.00	1	**1.35E-05**	2.61E-01	1.00E+00	7.51E-01	**9.54E-07**
			10	**1.69E-08**	**8.39E-05**	**1.91E-06**	1.00E+00	1.00E+00
	0.9	0.25	1	**1.79E-05**	9.96E-01	1.00E+00	**4.15E-03**	**9.54E-06**
			10	**5.72E-07**	9.95E-01	9.98E-01	**6.04E-03**	**2.43E-03**
		1.00	1	**1.11E-10**	9.99E-01	1.00E+00	**9.93E-04**	**1.91E-06**
			10	**7.04E-08**	6.76E-01	1.00E+00	3.37E-01	**9.54E-06**

*No teste de hipótese, md_1 sempre corresponde à mediana obtida pelo IGS-VND; já md_2 indica a mediana da abordagem descrita no cabeçalho de coluna. Valores em negrito indicam os casos em que a hipótese nula (H_0) é rejeitada, aceitando assim a hipótese alternativa H_1 ($\alpha = 0.05$).

não apresentou resultados satisfatórios para o problema aqui tratado, como pode ser observado na Tabela 4.4 e na Figura 4.7. Isto pode estar relacionado ao fato do GA, na sua forma nativa, usar o SA (*Simulated Annealing*) como operador de busca local, e o desempenho deste ter sido prejudicado pelo critério de parada utilizado, ou ainda, à omissão do autor de algum detalhe de implementação do operador de busca local. Entretanto, os resultados obtidos na Tabela 4.7 mostram que, quando o VND é incorporado pelo GA como operador de busca local, esta nova estrutura apresenta desempenho superior em todos os cenários avaliados frente às abordagens de estado da arte IHM e ILS. É importante observar que, no cenário $2.00 \times 0.3 \times 1.00 \times 10$, os resultados obtidos pelo GA-VND reduzem em média 20.10% os melhores resultados conhecidos na literatura. Vale destacar que, em cenários em que as datas de entrega são mais distantes entre si ($\tau = 0.3$), um sequenciamento eficiente tende a ter



(a) $0.02\times0.3\times0.25\times1$ (b) $0.02\times0.3\times0.25\times10$ (c) $0.02\times0.3\times1\times1$ (d) $0.02\times0.3\times1\times10$

(e) $0.02\times0.9\times0.25\times1$ (f) $0.02\times0.9\times0.25\times10$ (g) $0.02\times0.9\times1\times1$ (h) $0.02\times0.9\times1\times10$

(i) $2\times0.3\times0.25\times1$ (j) $2.\times0.3\times0.25\times10$ (k) $2\times0.3\times1\times1$ (l) $2\times0.3\times1\times10$

(m) $2\times0.9\times0.25\times1$ (n) $2\times0.9\times0.25\times10$ (o) $2\times0.9\times1\times1$ (p) $2\times0.9\times1\times10$

Figura 4.7: Boxplot comparando o RPD alcançado pelas mataheurísticas para cada um dos 16 cenários avaliados ($\eta \times \tau \times R \times r_\tau$) para os conjunto de instâncias *large*.

mais impacto na função objetivo, já que as datas de entrega estão mais espaçadas. Deve-se observar que o GA-VND, quando comparado às outras duas metaheurísticas que também incorporam o VND como operador de busca local, apresenta resultados

Tabela 4.6: Teste pareado de Wilcoxon por grupo de instâncias comparando o ABC-VND em relação ao ABC nativo e as abordagens estado da arte.

$100n10m$				$H_1 : md_1 < md_2{}^*$			$H_1 : md_1 > md_2{}^*$	
η	τ	R	r_τ	ABC-LS	IHM	ILS	IHM	ILS
0.02	0.3	0.25	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00
			10	**9.54E-07**	**2.90E-04**	**9.54E-07**	1.00E+00	1.00E+00
		1.00	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00
			10	**5.49E-04**	**1.80E-02**	**1.91E-06**	9.89E-01	1.00E+00
	0.9	0.25	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00
			10	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00
		1.00	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00
			10	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00
2.00	0.3	0.25	1	**9.54E-07**	**1.91E-06**	**1.91E-06**	1.00E+00	1.00E+00
			10	**9.54E-07**	**4.77E-06**	**9.54E-07**	1.00E+00	1.00E+00
		1.00	1	**9.54E-07**	**9.54E-07**	**4.77E-06**	1.00E+00	1.00E+00
			10	**9.54E-07**	**1.91E-06**	**9.54E-07**	1.00E+00	1.00E+00
	0.9	0.25	1	**9.54E-07**	**3.15E-05**	**9.54E-07**	1.00E+00	1.00E+00
			10	**9.54E-07**	**9.54E-07**	**1.91E-06**	1.00E+00	1.00E+00
		1.00	1	**9.54E-07**	**4.77E-06**	**6.68E-06**	1.00E+00	1.00E+00
			10	**9.54E-07**	**9.54E-07**	**6.68E-05**	1.00E+00	1.00E+00

No teste de hipótese, md_1 sempre corresponde à mediana obtida pelo ABC-VND; já md_2 indica a mediana da abordagem descrita no cabeçalho de coluna. Valores em negrito indicam os casos em que a hipótese nula (H_0) é rejeitada, aceitando assim a hipótese alternativa H_1 ($\alpha = 0.05$).

superiores ao IGS-VND e ABC-VND, respectivamente, em 14 e 13 cenários de 16 possíveis. Em contrapartida, apenas para o cenário $0.02 \times 0.3 \times 0.25 \times 10$ há evidência estatística significante que indique que as abordagens IGS-VND e ABC-VND apresentam desempenho superior ao GA-VND.

Tabela 4.7: Teste pareado de Wilcoxon por grupo de instâncias comparando o GA-VND em relação ao GA nativo e as abordagens estado da arte IHM e ILS e as metaheurísticas IGS e ABC integradas ao VND.

$100n10m$				$H_1 : md_1 < md_2{}^*$					$H_1 : md_1 > md_2{}^*$			
η	τ	R	r_τ	GA-LS	IHM	ILS	IGS-VND	ABC-VND	IHM	ILS	IGS-VND	ABC-VND
0.02	0.3	0.25	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**4.10E-05**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
			10	**9.54E-07**	**5.30E-03**	**1.91E-06**	1.00E+00	1.00E+00	9.95E-01	1.00E+00	**1.61E-04**	**3.41E-04**
		1.00	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
			10	**1.91E-06**	**1.80E-02**	**1.91E-06**	1.00E+00	1.00E+00	9.89E-01	1.00E+00	1.00E+00	1.00E+00
	0.9	0.25	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**1.58E-04**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
			10	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**4.77E-06**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
		1.00	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
			10	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**8.39E-05**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
2.00	0.3	0.25	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
			10	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**2.20E-02**	1.00E+00	1.00E+00	1.00E+00	9.80E-01
		1.00	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
			10	**9.54E-07**	**1.91E-06**	**9.54E-07**	**2.86E-06**	2.26E-01	1.00E+00	1.00E+00	1.00E+00	7.85E-01
	0.9	0.25	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**1.91E-06**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
			10	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
		1.00	1	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00	1.00E+00	1.00E+00
			10	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	**9.54E-07**	1.00E+00	1.00E+00	1.00E+00	1.00E+00

No teste de hipótese, md_1 sempre corresponde à mediana obtida pelo GA-VND; já md_2 indica a mediana da abordagem descrita no cabeçalho de coluna. Valores em negrito indicam os casos em que a hipótese nula (H_0) é rejeitada, aceitando assim a hipótese alternativa H_1 ($\alpha = 0.05$).

Por fim, analisando o desempenho geral entre as abordagens pelo teste de Friedman, apresentado nas Tabelas 4.8 e 4.9, conclui-se que as três abordagens que substituíram o operador de busca local pelas variações do VND apresentaram resultados superiores às abordagens de estado da arte IHM e ILS. Em contrapartida, as mesmas metaheurísticas, em sua forma nativa, apresentaram desempenho inferior (ABC-LS, GA-LS) ou igual (IGS-LS) comparado às abordagens IHM and ILS.

Tabela 4.8: Teste de Friedman para comparação do desempenho geral das metaheurística.

Chi-squared	df	p-value
1835.2	7	**<2.2E-16**

Tabela 4.9: Rank obtido pelo teste de Friedman do desempenho geral das abordagens baseada em metaheurísticas.

Metaheurística	Soma dos ranks	Posição*
GA-VND	422.5	a
ABC-VND	666.0	b
IGS-VND	1185.5	c
ILS	1269.0	d
IGS-LS	1607.5	e
IHM	1647.0	e
ABC-LS	2227.5	f
GA-LS	2459.0	g

*Posições com mesma letra indicam que não há diferença estatística significativa entre as metaheurísticas.

4.6 Discussão e Conclusões

Este capítulo é direcionado para o projeto e análise do procedimento *Variable Neighborhood Descent* (VND) como operador de busca local de metaheurísticas para minimização do TWT em ambientes UPMSMDST. Foram exploradas três metaheurísticas (ABD, IGS, GA) previamente propostas na literatura, sendo apresentados novos operadores de busca local para estas abordagens. O VND é projetado através da metodologia de planejamento de experimentos proposta por Taguchi para parametrização robusta de parâmetros. No estudo, foram propostas seis estruturas de vizinhança, sendo avaliadas três variações do VND.

As estruturas de vizinhança foram exploradas considerando duas ordens de exploração: (i) pelo incremento na cardinalidade de permutações da vizinhança ($NE.O_{card}$); e (ii) pelo incremento do impacto de um movimento da estrutura em uma solução ($NE.O_{imp}$). Para as metaheurísticas IGS e GA, a exploração pela ordem $NE.O_{imp}$ apresentou melhor desempenho. Em contrapartida, para a metaheurística ABC, a análise indicou que a ordem pela cardinalidade ($NE.O_{card}$) resultou em melhor desempenho. Considerando as variações do VND, a variação *Basic*-VND apresenta-se como a melhor opção para todas as metaheurísticas. Estes resultados indicam que a ordem em que as estruturas de vizinhança são exploradas influenciam, diretamente,

no desempenho do VND como operador de busca local de metaheurísticas para o problema estudado. O *Basic*-VND é a única variação que considera a ordem de exploração das estruturas de vizinhança. Já ao se analisar as formas de exploração das estruturas de vizinhança, a estratégia *Best Improvement* é selecionada para ser utilizada na maior parte das estruturas de vizinhança. A estratégia *First Improvement* é selecionada apenas três vezes, sendo duas em conjunto com a vizinhança NE.IS. A NE.IS, entretanto, apresenta apenas uma pequena contribuição na convergência das abordagens em que é utilizada (IGS e GA). Em contrapartida, as estruturas de vizinhança NE.ES, NE.EI e NE.IGS apresentam as maiores contribuições na convergência do VND para todas as metaheurísticas avaliadas. A análise das estruturas de vizinhança indicou que a NE.ES é a estrutura responsável pela maior influência na convergência das abordagens avaliadas. Por outro lado, a NE.IS e NE.SIGS são as estruturas que têm apenas um pequeno (às vezes nenhum) impacto na convergência. Já a estrutura NE.II contribui para todas as abordagens, como ajuste fino do resultado.

Já através dos experimentos que comparam os resultados entre metaheurísticas, é observado que o redesenho dos operadores de busca local acarreta em significativa melhora de desempenho das abordagens avaliadas. Para as abordagens populacionais (ABS e GA), a incorporação do VND acarretou em incremento do desempenho em todos os cenários avaliados. Quando comparados às abordagens de estado da arte (IHM e ILS), observou-se que as metaheurísticas que incorporaram o VND apresentaram resultados superiores em quase todos os cenários avaliados. O mesmo comportamento não é observado para as mesmas metaheurísticas com o operador de busca local nativo.

Nos próximos dois capítulos, as estruturas de vizinhança propostas neste capítulo e algumas das conclusões obtidas para guiar o projeto do operadores de busca local serão utilizadas. No Capítulo 5, propomos um operador de busca local para uma abordagem baseada na metaheurística híbrida VND-aiNET para a resolução de problemas em ambiente UPMSMDST. A abordagem é projetada para se adaptar a diversos objetivos. Assim, o operador de busca local deve se adaptar a características do critério de otimização utilizado. O Capítulo 6 utiliza as estruturas de vizinhança e as conclusões obtidas no presente Capítulo para o projeto de um operador de busca local para a minimização do TWET, com janelas de tempo, em ambientes UPMSMDST. Neste operador de busca local, optou-se por minimizar o número de avaliações da função objetivo, já que a ordem de complexidade da mesma é consideravelmente elevada quando comparada ao TWT.

Capítulo 5

Projeto de um método adaptável a diferentes critérios de otimização em ambientes de sequenciamento de tarefas em máquinas paralelas não relacionadas

Este capítulo apresenta a importância de se estudar métodos computacionais para ambientes UPMSMDST que sejam menos dependentes dos critérios de otimização no processo de busca. No capítulo é proposta uma solução, baseada na metaheurística VNS-aiNet, que reduz o uso de características dos critérios de otimização para guiar o processo de busca. São realizados quatro estudos de caso, cada um destinado a um critério de otimização. Na primeira seção são apresentados as justificativas e objetivos do capítulo. A Seção 5.1 também apresenta as razões pelas quais é escolhida a metaheurística híbrida VNS-aiNet para a construção do método que se adapte a diferentes critérios de otimização. Posteriormente, na Seção 5.2, é apresentada a definição do problema de sequenciamento em ambientes UPMSMDST usado no capítulo, bem como são apresentados os quatro critérios de otimização usados nos estudos de caso. A Seção 5.3 detalha os operadores propostos para a construção do VNS-aiNet. Em seguida, na Seção 5.4, é apresentada a metodologia experimental do capítulo, sendo introduzidos dois conjuntos de experimentos: o primeiro é realizado para avaliar se os operadores propostos contribuem para a adaptação da abordagem proposta a diferentes critérios de otimização; o segundo é para a verificar qual o desempenho da abordagem em relação a abordagens de estado da arte destinadas a critérios de otimização específicos. A Seção 5.5 apresenta os resultados dos dois conjuntos de experimentos propostos. Por fim, na Seção 5.6 mostra-se uma discussão e as conclusões obtidas com o estudo, além de apresentar pontos não contemplados na abordagem proposta. Destacamos que este capítulo é fortemente baseado no artigo Diana et al. (2021a), publicado como parte dos requisitos obrigatórios para conclusão desta tese.

5.1 Introdução

Nos últimos anos, para unir problemas teóricos e ambientes de manufatura do mundo real, muitos pesquisadores abordaram problemas de sequenciamento considerando ambientes de máquinas paralelas não relacionadas. Como pode ser observado na revisão bibliográfica apresentada na Seção 2.3 para ambientes UPMSMDST, são publicados continuamente na literatura diversos trabalhos com o intuito de dar resposta aos principais critérios de otimização e restrições presentes nestes ambientes. Para garantir um desempenho elevado, em sua maioria, abordagens computacionais presentes na literatura tendem a ser propostas incorporando características específicas do problema estudado, como o critério de otimização, e restrições específicas ao problema. Devido a isto, são construídos operadores extremamente dependentes do critério de otimização, o que torna as abordagens de difícil extensão para aplicações no mesmo ambiente mas com critérios de otimização e restrições diferentes do cenário inicialmente previsto. Assim, alterações nos critérios de otimização e incorporação de restrições tendem a acarretar grandes transformações nos métodos computacionais utilizadas para a resoluções dos problemas. Esta dependência impõe uma grande limitação destas abordagens, uma vez que critérios de otimização com pouca visibilidade teórica, mas de interesse em aplicações de problemas do mundo real, tendem a não ser contemplados na literatura de sequenciamento.

Considerando o cenário exposto, este capítulo tem com objetivo principal a construção de uma abordagem computacional para solução de problemas de sequenciamento em ambientes UPMSMDST, que seja extensível a diferentes critérios de otimização, de forma a garantir resultados competitivos com abordagens propostas para critérios específicos. Para atingir este objetivo, a abordagem aqui proposta deve atender a dois objetivos específicos:

(i) OE1: a abordagem deve possuir componentes que atuem de maneira integrada e cooperativa, de forma que consiga generalizar o processo de busca, produzindo uma resposta "*robusta*" a diferentes critérios de otimização;

(ii) OE2: a abordagem proposta deverá produzir resultados compatíveis às abordagens estado-da-arte relativas a ambientes UPMSMDST para critérios de otimização específicos.

Na literatura de sequenciamento, o termo "*robusta*" pode ser usado em diferentes contextos, como problemas que consideram tempos de processamento incertos (Yang e Yu, 2002; Wu et al., 2020b), ambientes que consideram eventos de quebra de máquina (Liu et al., 2007; Al-Hinai e ElMekkawy, 2011), incorporação dinâmica de tarefas (Skutella e Verschae, 2016), dentre outros. No entanto, usamos aqui o termo resposta "*robusta*" para denominar uma abordagem cujos resultados devem apresentar alta qualidade, independentemente dos critérios de otimização usados.

Algumas hiper-heurísticas, como a *Cloud Theory-based Simulated Annealing* (Wu et al., 2020b,c), têm sido utilizadas na literatura para a resolução de problemas de sequenciamento robusto. No entanto, para ambientes UPMSMDST com minimização de *makespan*, o Algoritmo de Seleção Clonal (CSA), proposto por Diana et al. (2015), tem apresentado bom desempenho para diferentes variações do ambiente UPMSMDST padrão, se adaptando bem a diferentes cenários (veja, Perez-Gonzalez

et al. (2019); Yepes-Borrero et al. (2020)). Além disso, este mesmo CSA é utilizado como base da metaheurística híbrida chamada VNS-aiNet (Diana et al., 2017). Para a resolução de problemas de otimização combinatória, Diana et al. (2017) propõem a hibridização de algoritmos baseados na teoria da rede imunológica, incorporando características de diversificação e busca local de métodos de trajetória como VNS e ILS. Considerando o poder de adaptação de algoritmos baseados na teoria da rede imunológica, e os resultados obtidos por Diana et al. (2017, 2015), consideramos que os mecanismos de diversificação do VNS-aiNet, caso bem projetados, são adequados para explorar o espaço de busca em ambientes UPMSMDST. Portanto, para a investigação dos objetivos levantados acima, este capítulo se propõe a explorar uma abordagem baseada na metaheurística VNS-aiNet.

Para verificar se a abordagem proposta é capaz de atender de forma eficiente aos objetivos definidos, propusemos, como metodologia, realizar um conjunto de experimentos para analisar o comportamento dos componentes da abordagem proposta, sujeita à diferentes critérios de otimização. Além disto, são realizados quatro estudos de caso, cada um destinado a um critério de otimização específico (i.e., TWT, *Makespan*, Atraso Total (TT) e TWCT). Os estudos de caso se destinam aos critérios de otimização mais relevantes para o ambiente UPMSMDST e o algoritmo proposto é comparado com as abordagens do estado da arte de cada critério específico. Através dos experimentos queremos verificar se o algoritmo consegue atingir uma resposta "*robusta*" para critérios de otimização tão distintos.

5.2 Descrição do problema

Considerando o problema de sequenciamento de tarefas em ambientes UPMSMDST apresentado na Seção 2.2 e os critérios de otimização discutidos na Seção 2.1.3, pode-se descrever o ambiente de sequenciamento estudado neste capítulo como: dado um conjunto de tarefas $N = \{1, 2, \ldots, n\}$, cada tarefa $k \in N$ deve ser atribuída e sequenciada em uma máquina do conjunto $M = \{1, 2, \ldots, m\}$ de máquinas paralelas, não relacionadas e continuamente disponíveis. Cada tarefa $k \in N$ possui um tempo de processamento p_{ik}, dependente da máquina i na qual está será processada. Duas tarefas não podem ser processadas na mesma máquina ao mesmo tempo e, uma vez iniciado o processamento de uma tarefa k, este não pode ser interrompido até a conclusão desta tarefa. O processamento de cada tarefa k é necessariamente precedido por um tempo de preparação s_{ijk}, o qual é dependente da sequência e da máquina, ou seja, depende da tarefa j, a qual precede imediatamente a tarefa k, na sequência da máquina i. Cada tarefa k pode possuir um *job ready time* r_k e um *due date* d_k, os quais indicam, respectivamente, o instante mínimo de tempo no horizonte de planejamento no qual a tarefa k pode iniciar o seu processamento e a data de entrega desejada da tarefa. Considerando este ambiente, pode-se definir o instante de término de uma tarefa k como:

$$C_k = \max(r_k, C_j + s_{ijk}) + p_{ik}.$$

Dado este ambiente, a abordagem proposta neste capítulo deve respeitar as restrições básicas descritas acima, e gerar respostas "*robustas*" para diferentes critérios de otimização aplicáveis a este ambiente. Como já descrito na seção anterior, este

trabalho irá realizar quatro estudos de caso, cada um destinado a um critério de otimização específico, sendo estes: TWT (Seção 2.1.3.3); *Makespan* (Seção 2.1.3.1); TT (Seção 2.1.3.3); e $TWCT$ (Seção 2.1.3.2). Assim, pela notação de Graham et al. (1979), os quatro estudos de caso realizados neste trabalho podem ser definidos como:

(i) Minimização do atraso total ponderado: $Rm \mid s_{ijk},\ r_j \mid TWT$ (Lin e Hsieh, 2014);

(ii) Minimização do *makespan*: $Rm \mid s_{ijk} \mid C_{max}$ (Al-Salem, 2004);

(iii) Minimização do atraso total: $Rm \mid s_{ijk} \mid TT$ (Chen, 2009);

(iv) Minimização ponderada dos instantes de término: $Rm \mid s_{ijk} \mid TWCT$ (Chen, 2015).

Os critérios de otimização dados por TT e $TWCT$ são baseados, respectivamente, nos trabalhos de Chen (2009) e Chen (2015). Como ambos os critérios permitem a existência de clientes primários que não podem ter suas tarefas atrasadas, é considerada a restrição na qual o subconjunto de tarefas destinadas a cliente primários, dado por N_p, não pode apresentar atrasos em suas datas de entrega. Portanto:

$$\sum_{k \in N_p} T_k = 0. \tag{5.1}$$

5.3 Algoritmo VNS-aiNet

Como pode ser observado na Seção 3.4.3, a construção de uma abordagem VNS-aiNet para a resolução de problemas em ambientes UPMSMDST requer a definição de seis passos principais. O primeiro passo consiste na definição da representação computacional de uma solução do problema, codificação de uma célula e inicialização da população inicial P_0. Este passo é descrito na Seção 5.3.1. Em seguida, deve ser realizada a definição de *fitness-like* para composição da função de afinidade célula/antígeno (f_{Ag}). Este passo é descrito na Seção 5.3.2. Já o passo que define a função de perturbação de soluções (T_{mut}) utilizada pelo operador de mutação é descrito na Seção 5.3.4. Um passo fundamental para a abordagem é a definição do método de busca local, sendo este apresentado na Seção 5.3.5. A Seção 5.3.8 detalha o método para cálculo da afinidade célula/célula (f_{Ab}) utilizado pelo operador de supressão. Por fim, a Seção 5.3.10, apresenta condições para que as soluções se tornem candidatas a entrar na memória imunológica.

5.3.1 Codificação da solução e inicialização da população

Uma célula no algoritmo VNS-aiNet, proposto aqui, é composta por quatro variáveis, conforme mostrado na Figura 5.1. As variáveis $c_p[mat]$, que indicam o nível de maturação da célula, e $c_p[s]$, que representam a solução codificada, são presentes na versão básica do VNS-aiNet, conforme descrito na Seção 3.4.3.1. A variável $c_p[s]$ para o problema estudado segue a representação de soluções descrita no Apêndice A.1. Já as variáveis $c_p[\beta_{mut}]$ e $c_p[IM_{prob}]$ são, respectivamente, variáveis auto-adaptativas de taxa de mutação e probabilidade de mutação por inserção, detalhadas na Seção 5.3.4.

Estas duas variáveis auto-adaptativas não estão presentes na versão original do VNS-aiNet, mas, para o problema estudado, auxilia na convergência do algoritmo para solução de diversos critérios de otimização, ajudando a garantir a "*robustez*" das soluções, como mostrado na Seção 5.4.3.

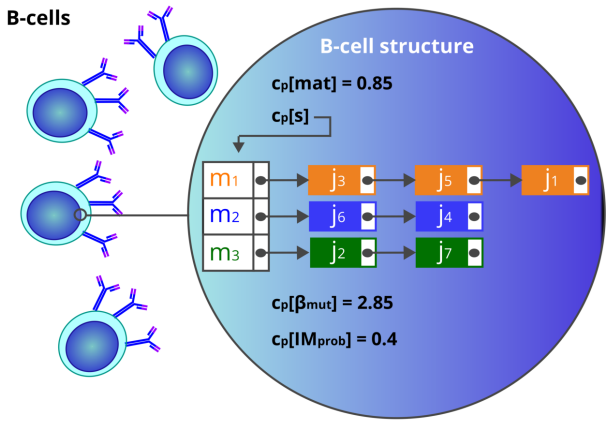


Figura 5.1: Representação de um *linfócito B* para problemas em ambientes UPMSMDST.

Como descrito na Seção 3.4.3.1, o algoritmo VNS-aiNet contém duas populações, denominadas população de células ativas e população de memória, representadas, respectivamente, por P_u e M_u, sendo u a geração em que o método se encontra. As soluções que irão compor a população P_0 de células ativas iniciais ($u = 0$) são construídas de maneira aleatória. Assim, cada solução $c_p[s]$ da célula $c_p \in P_0$ é gerada aleatoriamente, conforme descrito no Apêndice A.2. Para os problemas com restrições, soluções infactíveis são permitidas na população P_0, mas estas células serão penalizadas pela função de *fitness-like*, de acordo com o nível de violação das restrições. O nível de maturação de cada célula $c_p \in P_0$ é definido como nulo, ou seja, $c_p[mat] = 0 \ \forall \ c_p \in P_0$. Além disto, os fatores auto-adaptativos adicionais de probabilidade de mutação por inserção ($c_p[IMprob]$) e taxa de mutação ($c_p[\beta_{mut}]$) são gerados aleatoriamente, com probabilidade uniforme, sendo os intervalos respectivamente definidos como $[0, 1]$ e $[1, \beta]$, no qual β é o parâmetro da taxa máxima de mutação.

5.3.2 Função de avaliação célula/antígeno(*fitness*)

A função de *fitness* (afinidade célula/antígeno f_{Ag}) do VNS-aiNet, descrita na Equação 3.7, representa um *trade-off* entre uma função relacionada à minimização do critério de otimização (f_{like}, dada pela Equação 5.2), e uma função (f_{mat}, dada pela Equação 3.9), inversamente proporcional ao fator de maturação $c_p[mat]$ da célula.

Como é proposto um estudo do algoritmo VNS-aiNet em um ambiente UPMSMDST que deve ser robusto a múltiplos critérios de otimização, com ou sem restrições, o termo da função de *fitness* relacionado à minimização de um critério deve considerar (caso exista) restrições impostas pelo problema. Como descrito na Seção 5.3.1, optou-se por permitir que soluções infactíveis estejam presentes na população P_u. Assim, a penalização de soluções infactíveis pela função objetivo é utilizada, já que

este é um método bastante utilizado na literatura para o controle de restrições. Para garantir esta característica nos mais diversos cenários, optou-se por penalizar o grau de infactibilidade com uma constante de penalização $C_{penalty}$. Esta constante deve ser calibrada afim de garantir que soluções que não satisfaçam às restrições sempre devem apresentar *fitness* inferior ao das soluções factíveis. Assim, para evitar que soluções com alto grau de infactibilidade sejam demasiadamente penalizadas, optou-se por utilizar uma função calculada pela ordem da solução c_p em relação às demais soluções pertencentes à população P_u. A Equação 5.2 representa este termo da função de *fitness*, sendo definida como:

$$f_{like}(c_p) = 1 - \frac{ordem(c_p, P_u)}{nAb}, \tag{5.2}$$

em que nAb é o tamanho da população P_u; $order(c_p, P_u)$ é uma função que retorna o índice em ordem crescente da solução $c_p[s]$ em relação às demais soluções pertencentes a P_u. A ordem é definida pelo valor da função de avaliação $f(c_p)$, dada por:

$$f(c_p) = criterio(c_p[s]) + p(c_p[s]), \tag{5.3}$$

a qual considera o valor de um dos critérios descritos na Seção 5.2. Para uma solução $c_p[s]$, é adicionado o valor de penalização por violações de restrições pela solução $c_p[s]$, definido pela função $p(c_p[s])$.

Como descrito na Seção 5.2, os critérios de otimização TT e $TWCT$ apresentam uma restrição envolvendo o atraso de tarefas para clientes prioritários. Assim, para estes problemas, a função de penalização $p(c_p[s])$ será expressa como:

$$p(c_p[s]) = \sum_{k \in N_p} T_k C_{penalty}, \tag{5.4}$$

na qual o fator de penalização, dado por $C_{penalty}$, penaliza, de acordo com o nível da violação, todas as tarefas destinadas a clientes primários ($jp \in N_p$) que apresentarem atraso. Para problemas com restrições não estudadas neste trabalho, esta função de penalização deve ser redefinida, de acordo com as restrições presentes no problema.

Para o problema estudado, o termo f_{mat} da função de *fitness* do VNS-aiNet, utilizado para rebaixar a avaliação de soluções que estão presas em mínimos locais, é utilizado na forma originalmente proposta pela abordagem, descrita na Equação 3.9.

5.3.3 Clonagem

Não há necessidade de nenhuma adaptação na fase de clonagem em relação à abordagem originalmente proposta. Assim, para a implementação do VNS-aiNet para ambientes UPMSMDST, basta seguir os passos descritos ne Seção 3.4.3.3. Deve-se apenas reforçar que um clone é uma cópia exata da solução clonada. Desta forma, as variáveis $c_p[\beta_{mut}]$ e $c_p[IM_{prob}]$, presentes apenas neste trabalho, também têm seu conteúdo clonado neste processo.

5.3.4 Mutação

No algoritmo VNS-aiNet, a fase de mutação é baseada no método de trajetória VNS, sendo utilizada apenas para diversificação da solução. O encaminhamento

para os mínimos locais é realizada pelos métodos de busca local. A mutação para o UPMSPSD é definida como a execução de permutações aleatórias com probabilidade uniforme na estrutura dos clones gerados. O VNS-aiNet proposto neste capítulo utiliza dois tipos de permutações:

(i) Inserção: dada a solução $cl_p[s]$, presente no clone cl_p, escolhe-se aleatoriamente, com probabilidade uniforme, uma máquina i; remove-se da máquina i uma tarefa k escolhida aleatoriamente; insere-se a tarefa k em uma posição p' de uma máquina w, sendo que p' e w também são escolhidas aleatoriamente, de modo tal que a máquina w não é necessariamente diferente da máquina i. Este tipo de perturbação consiste em movimentos aleatórios nas estruturas de vizinhança Inserção Interna (Seção 2.5.1) ou Inserção Externa (Seção 2.5.3), ilustrados, respectivamente, nas Figuras 2.2 e 2.4.

(ii) Troca: dada a solução $cl_p[s]$, presente no clone cl_p, escolhe-se aleatoriamente, com probabilidade uniforme, uma máquina i e uma tarefa k sequenciada na máquina i; aleatoriamente, seleciona-se também uma máquina w e uma tarefa j sequenciada na máquina w. Posteriormente, troca-se as tarefas k e j nas respectivas posições em que ambas ocupavam nas máquinas i e w. As máquinas i e w não necessariamente são diferentes, mas, caso $i = w$, necessariamente as tarefas k e j devem ser diferentes; logo, $j \neq k$ caso $i = w$. Em caso de $i \neq w$, a perturbação consiste em um movimento aleatório na estrutura de vizinhança Troca Externa (Seção 2.5.4), ilustrado na Figura 2.5; caso $i = w$, a perturbação representa um movimento aleatório de Troca Interna (Seção 2.5.2), ilustrado na Figura 2.3.

Considerando os dois tipos de perturbação descritos acima, a função T_{mut}, utilizada pelo operador de mutação do VNS-aiNet descrito na Equação 3.11, pode ser definida conforme observado no Algoritmo 5.3.4. Neste algoritmo, são realizadas n_{mut} mutações no clone cl_p passado como parâmetro, sendo que cada mutação consiste na aplicação de uma pertubação selecionada aleatoriamente com probabilidade para Inserção e Troca dada, respectivamente, por $cl_p[IM_{prob}]$ e $1 - cl_p[IM_{prob}]$. A variável $cl_p[IM_{prob}]$ é um fator auto-adaptável, que se adequa ao ambiente em que a célula se encontra com o passar das gerações. Já o número de mutações n_{mut} sofre influência do fator auto-adaptável $c_p[\beta_{mut}]$. Portanto, a Equação 3.11 é reescrita, substituindo-se o parâmetro β pela variável $cl_p[\beta_{mut}]$.

A Figura 5.2 mostra um exemplo do processo de mutação sobre um clone cl_p, considerando o número de mutações $n_{mut} = 2$ e a probabilidade de mutação por Inserção $cl_p[IM_{prob}] = 0.32$. Como descrito acima, dado o clone cl_p, para cada mutação n_{mut}, um tipo de permutação é aleatoriamente selecionada, definida por $cl_p[IM_{prob}]$. Neste exemplo, a primeira mutação é realizada através de uma permutação por Troca, selecionando-se aleatoriamente um número com probabilidade $U[0, 1] = 0.74$ e $cl_p[IM_{prob}] = 0.32$. Observe que $0.74 > 0.32$, assim, a perturbação por troca é selecionada. Realiza-se a troca da tarefa j_2, presente na máquina m_1, e a tarefa j_7, sequenciada na máquina m_2 ($i = 1, w = 2, k = 2$ *and* $j = 7$). A solução resultante é sujeita a uma nova mutação, pelo tipo de permutação por Inserção, já que $U[0, 1] = 0.02$; logo, $0.02 < 0.32$. Nesta mutação, a tarefa j_6, alocada na máquina m_2, é realocada na mesma máquina na posição subsequente à tarefa j_8

Algoritmo 21 Função de mutação para problemas em ambientes UPMSMDST

Parâmetros:
$-cl_p$: Clone que será submetido ao processo de mutação;
$-n_{mut}$: Número de mutações realizadas na célula;

```
1: procedure T_mut(cl_p, n_mut)
2:     for 1 até n_mut do
3:         op ← U[0, 1];
4:         if op < cl_p[IM_prob] then
5:             Aplicar perturbação do tipo inserção em cl_p;
6:         else
7:             Aplicar perturbação do tipo troca em cl_p;
8:         end if
9:     end for
10: end procedure
```

$(i = w = 2, k = 6\ and\ p = 3)$. A solução resultante é retornada pelo operador de mutação.

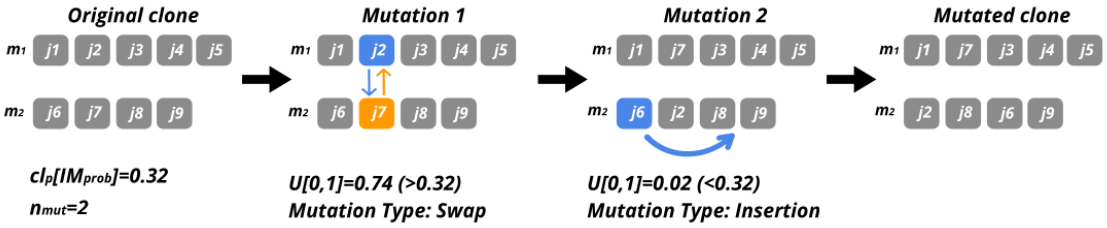


Figura 5.2: Exemplo de mutação do clone cl_p com $n_{mut} = 2$. Dois movimentos são realizados: o primeiro é um movimento de Troca; o segundo, é de Inserção.

5.3.5 Busca local *Variable Neighborhood Descent*

O operador de mutação utilizado na abordagem proposta tem, como único intuito, a diversificação da população de clones, sendo necessária uma estratégia de refinamento das soluções para convergência destas para os mínimos locais. No capítulo anterior, realizamos um estudo sobre métodos de busca local para a minimização do TWT em um ambiente UPMSMDST com características iguais às estudadas no presente capítulo. Devido aos resultados alcançados pelas abordagens propostas no capítulo anterior, utilizamos aqui uma busca local baseada no método *Variable Neighborhood Descent* (VND), utilizando quatro estruturas propostas nas Seção 4.3.1: $NE.II$ (Seção 4.3.1.1); $NE.IS$ (Seção 4.3.1.2); $NE.EI$ (Seção 4.3.1.3) e $NE.ES$ (Seção 4.3.1.4). No capítulo anterior, deve-se observar que a seleção da máquina i, que guia o processo de busca, é um ponto fundamental para o desempenho destas estruturas de vizinhança. Para generalizar estas estruturas para diversos critérios de otimização, propomos especializar esta seleção por critério de otimização, definindo, assim, a função $ls\_machine(f(\cdot), s)$, descrita na Seção 5.3.5.1. Esta função assume o papel de selecionar, para cada critério de otimização, uma máquina que, naquela iteração, possa contribuir na minimização da função objetivo $f(\cdot)$ para a

solução s. A incorporação de $ls\_machine(f(\cdot), s)$ nas estruturas $NE.II$, $NE.TI$, $NE.EI$ e $NE.ES$ requer a substituição por $i \leftarrow ls\_machine(f(\cdot), s)$; da Linha 5 do Algoritmo 15, Linhas 2–3 do Algoritmo 17 e Linhas 2–3 do Algoritmo 16 . Além disto, os algoritmos que definem estas estruturas de vizinhança recebem a solução $cl_p[s]$ presente no clone cl_p, além de receber o tipo de descida utilizado. Para as quatro estruturas, utilizamos a descida por *Best Improvement*. Assim, o parâmetro *tipo_descida* sempre recebe BI. Apesar da estrutura NE.IGS (Seção 4.3.1.5) ter tido alto grau de acoplamento para o critério de TWT, nossos experimentos preliminares mostram que a generalização da mesma para diversos critérios de otimização não apresentou impacto significante ao VNS-aiNet aqui proposto. Para a exploração das estruturas de vizinhança, utilizamos a versão *Basic*-VND (Seção 3.3.4.1) e a ordem de explorações $NE.O_{imp}$ (Seção 4.3.2.1) definida pelo incremento do impacto esperado de aplicação da estrutura na solução. Ambas as escolhas se baseiam nos resultado apresentados no capítulo anterior.

5.3.5.1 Seleção de máquina para busca local

Devido ao alto custo computacional dos operadores de busca local e com o objetivo de descartar permutações que não influem diretamente no valor da função objetivo, para as quatro estruturas de vizinhança usadas, i.e., $NE.II$, $NE.TI$, $NE.EI$ e $NE.ES$, a máquina resultante da Equação (5.5) é utilizada como referência do processo de busca:

$$ls\_machine(f(\cdot), s) = \begin{cases} U[1,\ |MT|], & f(\cdot) = TWT; \\ U[1,\ |MT|], & f(\cdot) = TT; \\ i \mid C_i = C_{\max}, & f(\cdot) = C_{\max}; \\ U[1,\ |M|], & f(\cdot) = TWCT. \end{cases} \tag{5.5}$$

sendo $f(\cdot)$ a função que define o critério de otimização o qual o algoritmo está otimizando e s a solução a ser otimizada na iteração.

Para o objetivo de minimização na forma dos critérios TWT e TT, apenas permutações em máquinas que apresentarem atraso podem acarretar alterações no objetivo. Assim, é selecionada, de forma aleatória e com probabilidade uniforme, uma máquina i, no conjunto de máquinas com tarefas que apresentem atraso MT; logo, $MT = \{i \in M : \sum_{k \in N_i} T_k > 0\}$, sendo N_i o conjunto de tarefas escalonadas na máquina i. Já na minimização do *makespan* ($C_{\max}$), apenas alterações na máquina i que define o *makespan*, ou seja, $i \mid C_i = C_{\max}$, surtem efeito na função objetivo. Portanto, esta máquina é selecionada para busca local. Por fim, na minimização do critério TWCT, uma alteração em qualquer máquina pode surtir efeito na função objetivo. Desta forma, é selecionada uma máquina i do conjunto de máquinas M, de forma completamente aleatória e com probabilidade uniforme.

Deve-se observar que a função $ls\_machine(f(\cdot), s)$ é o único ponto do VNS-aiNet proposto que utiliza características do critério de otimização no processo de busca. Por esta razão, a função $ls\_machine(f(\cdot), s)$ é isolada do restante do algoritmo e pode facilmente ser adaptada para incorporar critérios de otimização específicos.

5.3.6 Atualização do fator de maturação e variáveis autoadaptáveis de mutação

Após a mutação e busca local, as células clonadas presentes em P_c são incorporadas à população de células ativas P_u e submetidas à atualização do fator de maturação (veja Algoritmo 12, linhas 10–11 e Seção 3.4.3.6). Todas as células genitoras pertencentes a P_u têm automaticamente seu fator $c_p[mat]$ incrementado em mat_+, já que estas células não sofreram quaisquer alterações pela mutação/busca local. Já para cada célula clone $cl_p \in P_c$, é verificado se o processo de mutação e busca local acarretou em melhora na resposta da função objetivo $f(\cdot)$. Caso afirmativo, $c_p[mat]$ é reiniciado, ou seja, $c_p[mat] \leftarrow 0$. Caso contrário, este vai ser incrementado em mat_+.

Na versão aqui proposta, além de atualizar o fator de maturação das células, esta etapa também atualiza os fatores alto-adaptáveis $c_p[\beta_{mut}]$ e $c_p[IM_{prob}]$, segundo:

$$c_p[\beta_{mut}] = f_{mat}(c_p) \times c_p[\beta_{mut}] + (1 - f_{mat}(c_p)) \times U[0,1] \times \beta, \tag{5.6}$$

$$c_p[IM_{prob}] = f_{mat}(c_p) \times c_p[IM_{prob}] + (1 - f_{mat}(c_p)) \times U[0,1]. \tag{5.7}$$

Observa-se que ambos os fatores são ponderados pela função $f_{mat}(c_p)$, a qual é inversamente proporcional ao fator de maturação $c_p[mat]$ (Seção 3.4.3.2). Assim, quando a solução da célula c_p é incrementada na geração corrente, ambas as variáveis $c_p[\beta_{mut}]$ e $c_p[IM_{prob}]$ se mantém inalteradas. Adicionalmente, à medida que o nível do fator de maturação $c_p[mat]$ é incrementado, ruídos aleatórios são introduzidos nestas variáveis.

A Figura 5.3 apresenta um exemplo do comportamento da taxa de mutação $c_p[\beta_{mut}]$ para duas células (c_1 e c_2) com o passar das gerações, ambas com a mesma taxa na primeira geração. É observado que, nas primeiras dez gerações, a taxa de mutação das duas soluções se mantém estável para ambas as soluções. Ao ficar presa em um mínimo local, ao redor da décima geração, a célula c_1 tem o seu fator de maturação incrementado, decrementando $f_{mat}(c_p)$. Assim, em cada geração subsequente, ruídos randômicos são imputados à variável $c_1[\beta_{mut}]$. Uma vez que a solução de c_1 consegue escapar de mínimo local ($\approx 14^a$ geração), $c_1[\beta_{mut}]$ se estabiliza na nova taxa que levou a solução à melhora e se mantém assim até ficar presa novamente em um mínimo local ($\approx 22^o$ geração).

5.3.7 Remoção de células maturadas

Assim como a fase de clonagem, a remoção de células com maturação completa não necessita da inclusão de características do problema. Desse modo, esta fase é executada pelo algoritmo como em sua forma original, descrita na Seção 3.4.3.7.

5.3.8 Supressão

A supressão no VNS-aiNet é realizada para selecionar as células que irão participar da resposta imune na próxima geração. A população P_{u+1} é composta pelas células que apresentarem melhor desempenho da função de *fitness*, conforme Equação (3.7), mas que estejam distantes entre si a um raio σ_s, sendo este um parâmetro do algoritmo. Este método garante que as células não se aglomerem em uma região

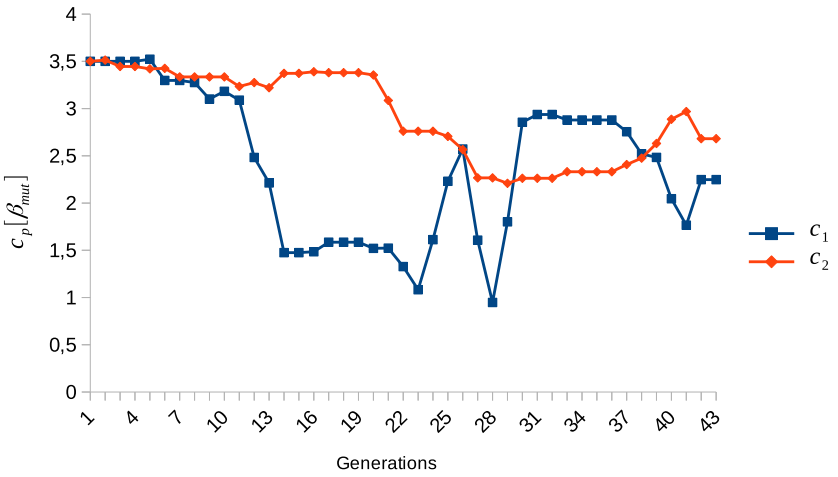


Figura 5.3: Exemplo da evolução da variável de mutação alto-adaptável.

do espaço de busca, permitindo uma melhor cobertura do mesmo, bem como garante que células estagnadas em mínimos locais sejam continuamente desprivilegiadas, já que o valor do *fitness* destas irá decair à medida em que nível de maturação for incrementado.

A definição da função de afinidade célula-célula $Af_{cell}(c_i, P_{u+1})$ é a única adaptação necessária para a utilização do operador de supressão padrão do VNS-aiNet (Seção 3.4.3.8) a qualquer problema. Esta função de afinidade é representada pela Equação 3.12, para a qual deve-se definir uma métrica de distância $d(c_1, c_2)$ entre duas células. Neste trabalho é utilizada a métrica de distância entre soluções de ambientes UPMSMDST proposta por Diana et al. (2015). Esta medida de distância considera cada máquina como um grafo, sendo as tarefas alocadas em cada máquina os vértices deste grafo e a sequência de tarefas as arestas. Assim, o operador de distância entre duas soluções pode ser definido como o percentual de arestas diferentes entre as soluções contidas nas células c_1 e c_2. Este operador de distância é dado por:

$$d(c_1, c_2) \quad = \quad \frac{1}{N} \sum_{k=1}^{N} \varphi(k, c_1[s], c_2[s]), \tag{5.8}$$

sendo:

$$\varphi(k, s_1, s_2) = \begin{cases} 0, & \text{se a tarefa } k \text{ é imediatamente precedida em } s_1 \text{ e } s_2 \\ & \text{pela mesma tarefa } j \text{ e ambas são sequenciadas na} \\ & \text{mesma máquina } i; \\ 1, & \text{caso contrário.} \end{cases}$$

5.3.9 Inserção de novas soluções

Como descrito na Seção 3.4.3.9, após a supressão pode haver a necessidade de inserção de $\max\{0, nAb - |P_{u+1}|\}$ novas células na população. A criação de novas células é realizada pelo método descrito na Seção 5.3.1.

5.3.10 Atualização da memória imunológica

A memória imunológica M_u tem o objetivo de armazenar as soluções históricas de boa qualidade e que se encontrem em seus mínimos locais, mas não contribuem com a resposta imune. O operador de atualização da memória descrito na Seção 3.4.3.10 é utilizado na abordagem aqui proposta para os problemas em ambientes UPMSMDST. Deve-se considerar, entretanto, a situação em que células cujas soluções são infactíveis não podem ingressar na memória imunológica da próxima geração M_{u+1}. Para problemas com restrições, envolvendo os critérios TT e TWCT, estudados neste trabalho, deve-se verificar se as restrições do modelo são atendidas pelas soluções presentes nas populações R_u e $P_u - P_{u+1}$ antes destas se tornarem candidatas a ingressar na memória M_{u+1}. Células em que as soluções não atendam às restrições impostas pelo modelo são descartadas pelo método de atualização da memória.

5.4 Metodologia experimental

Nesta seção descrevemos a metodologia experimental utilizada para verificar se a abordagem proposta atinge o objetivo proposto neste capítulo da construção de uma abordagem que gere resultados "*robustos*" e extensíveis a diferentes critérios de otimização. Com este intuito, a Seção 5.4.1 apresenta detalhes da implementação computacional e ambiente utilizado nos experimentos. Já a Seção 5.4.2 apresenta a metodologia para a sintonia de parâmetros da abordagem. Por sua vez, a Seção 5.4.3 descreve o conjunto de experimentos para verificar se o primeiro objetivo específico (OE1) é atingido. Por fim, a Seção 5.4.4 apresenta os estudos de casos e as metodologias utilizadas para verificar se o VNS-aiNet apresenta resultados compatíveis com abordagens de estado da arte para critérios de otimização específicos (OE2).

5.4.1 Implementação computacional

Todos os experimentos realizados neste capítulo são baseados em uma implementação do algoritmo VNS-aiNet desenvolvida em linguagem Java sobre uma Oracle JDK 1.8. Os experimentos são realizados em um ambiente Linux Ubuntu 16.04 com processador Intel Pentium E5200 com clock de 2.5 GHz, 2MB de cache e 4GB de memória RAM. Apesar deste processador apresentar dois núcleos, não é utilizado processamento paralelo neste trabalho. Com o intuito de seguir as *guidelines* de boas práticas de laboratório para experimentos em pesquisa operacional (Kendall et al., 2016), o código fonte e os dados analíticos apresentados neste capítulo se encontram disponíveis em `https://github.com/rodsxe/vns-ainet-UPMSMDST`.

5.4.2 Calibração de Parâmetros

Neste capítulo é projetado um algoritmo para oferecer soluções para diversos critérios do problema de sequenciamento em ambientes de máquinas paralelas não relacionadas. Devido às características específicas de cada problema, fatores como função objetivo, número de máquinas e tarefas, velocidade de processamento das máquinas não relacionadas, tempos de setup, *job ready times*, dentre outros, tendem a influenciar na resposta do algoritmo. Para reduzir a influência destes fatores não

controláveis na resposta do algoritmo, é utilizada a metodologia *Robust Parameter design*, proposta por Taguchi (Taguchi, 1986, 1987). Como já discutido no capítulo anterior, o projeto experimental proposto por Taguchi é uma metodologia reconhecida no processo de controle de qualidade e vem sendo utilizado para calibração de parâmetros de metaheurísticas destinadas a múltiplos cenários (vide (Zandieh et al., 2009), (Thepphakorn et al., 2014), (Rambod e Rezaeian, 2014)).

Os parâmetros tamanho da população de células ativas (nAB), fator de ponderação da função de afinidade célula antígeno (α), número máximo de clones por célula (nC_{max}), limiar de supressão (σ_s) e fator de maturação (mat_+) são utilizados como os fatores controláveis do algoritmo VNS-aiNet. A Tabela 5.1 apresenta os 4 níveis testados para cada fator, sendo estes definidos empiricamente através de experimentos de pré-calibração. Na abordagem aqui proposta, a taxa de mutação ($c_p[\beta_{mut}]$) de cada célula é uma variável auto-adaptável. Assim, a taxa máxima de mutação (β) foi definida empiricamente com o valor 4, não sendo recalibrada. O mesmo ocorre com o fator de penalização de solução infactíveis $C_{penalty}$, utilizado nos estudos de caso destinados aos critérios de otimização TT (Seção 5.4.4.3) e TWCT (Seção 5.4.4.4). Definiu-se nos experimentos de pré-calibração que $C_{penalty} = 1000$ é um valor suficientemente grande para ambos os critérios.

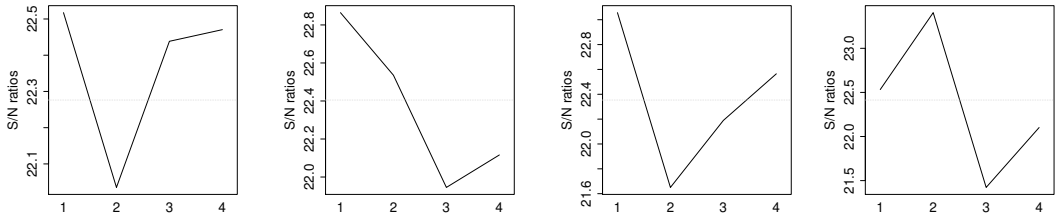
Tabela 5.1: Fatores controláveis e seus respectivos níveis utilizados na calibração do VNS-aiNet.

Parâmetro	Nível			
	1	2	3	4
Tamanho da população de células ativas (nAB)	**10**	15	20	25
Fator de ponderação da função de afinidade célula antígeno (α)	**0.4**	0.45	0.5	0.55
Número máximo de clones por célula (nC_{max})	**4**	6	8	10
Taxa máxima de mutação (β)	–	–	–	**4**
Limiar de supressão (σ_s)	0.07	**0.09**	0.11	0.13
fator de maturação (mat_+)	0.01	**0.03**	0.05	0.07

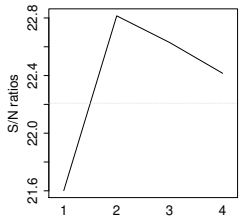
Considerando os cinco fatores controláveis e os quatro níveis destinados a cada fator, a utilização de um projeto de experimentos do tipo fatorial completo implicaria em $4^5 \times r = 1024r$ experimentos, sendo r o número do replicações para cada combinação, o que torna inviável sua utilização. Assim, utilizamos um projeto experimental do tipo fatorial fracionado com o arranjo ortogonal de Taguchi $L_{16}(4^5)$. São executados $16r$ experimentos para calibração dos fatores controláveis, sendo r o número de replicações para cada combinação dos parâmetros. Como a abordagem proposta é utilizada para resolver uma grande quantidade de cenários, e em cada cenário existem fatores de ruídos distintos, com níveis distintos, optou-se por selecionar 100 instâncias de forma aleatória no conjunto união de todas as instâncias estudadas neste trabalho. Devido à natureza estocástica do VNS-aiNet, para cada instância foram realizadas 10 execuções independentes e o resultado de cada replicação r representa o resultado médio destas execuções. Para cada repetição r, considerou-se o critério de parada definido por $2 \times (n+m-1)/(m-1) \times 10^7$ avaliações de função objetivo. Este critério é utilizado considerando que o incremento dos parâmetros nAB e $nC_{\max}$ sem redução do número de gerações acarreta indiscriminadamente no incremento do custo computacional. O projeto experimental foi gerado no *software* `R` pelo pacote *Quality Tools*

[1].

As respostas resultantes de cada uma das combinações dos parâmetros são transformadas em um *signal-to-noise ratio* (S/N_{ratio}) pela Equação 4.4, considerando exatamente os mesmos critérios descritos na Seção 4.4.3 do capítulo anterior. A Figura 5.4 apresenta o indicador S/N_{ratio} para cada nível de cada fator aqui avaliado. Os níveis selecionados para cada fator (maiores valores de S/N_{ratio}) são destacados em negrito na Tabela 5.1, sendo utilizados nos demais experimentos realizados neste capítulo.



(a) nAB (b) $nC_{\max}$ (c) α (d) σ_s



(e) mat_+

Figura 5.4: Gráfico de efeitos principais do indicador S/N ratio para o algoritmo VNS-aiNET.

5.4.3 Análise de Componentes

Como destacado na introdução, este capítulo tem como objetivo construir um algoritmo que seja extensível a diversos critérios de otimização. Para cumprir este objetivo, destacamos que a abordagem proposta deve atender a dois objetivos específicos. Assim, esta seção descreve a metodologia utilizada para verificar a aderência da abordagem ao primeiro objetivo específico (OE1), que define que a abordagem proposta deve possuir operadores auto-adaptáveis que propiciem adaptação a distintos espaços de busca afim de gerar soluções "*robustas*" para diferentes critérios de otimização. Nos experimentos descritos nesta seção, a versão do VNS-aiNet calibrada na Seção 5.4.2, com os parâmetros selecionados apresentados na Tabela 5.1, são consideradas como a versão de controle experimental, denominada como *control*. Variações derivadas de *control* através da desativação/ativação de componentes são avaliadas e comparadas entre si, para verificar se os componentes presentes em *control* contribuem na adaptação da abordagem. As variações avaliadas nestes experimentos são descritas como:

[1]Disponível em http://www.r-qualitytools.org/

(i) *control*: versão padrão do VNS-aiNet proposta;

(ii) $w/ls-machine$: versão *control* que não usa a função $ls\_machine(f(\cdot), cl_p)$. Assim, a máquina i da busca local é selecionada aleatoriamente com probabilidade uniforme;

(iii) $mat_+ = 0.0$: versão *control* que usa $mat_+ = 0.0$. O parâmetro mat_+ influência diretamente na remoção de células maturadas, e, indiretamente, através da função de afinidade célula-antígeno $f_{Ag}(c_p)$, na taxa de mutação, número do clones, e operador de supressão;

(iv) $\sigma_s = 0.0$: versão *control* que usa $\sigma_s = 0.0$. Isto faz o operador de supressão ter um comportamento elitista;

(v) w/sa: versão *control* que não usa as variáveis auto-adaptáveis $c_p[\beta_{mut}]$ e $c_p[IM_{prob}]$;

(vi) $mat+ = 0.0$ & $\sigma_s = 0.0$: versão *control* que acumula os fatores $mat+ = 0.0$ e $\sigma_s = 0.0$;

(vii) $mat+ = 0.0$ & $\sigma_s = 0.0$ & w/sa: versão *control* que acumula os fatores $mat+ = 0.0$, $\sigma_s = 0.0$ e w/sa;

(viii) GVNS: implementação da metaheurística *Multistart General VNS* metaheuristic. O operadores *Shake* são implementados usando os movimentos de Troca e Inserção descritos na Seção 5.3.4. Já a busca local utilizada é a mesma utilizada pelo VNS-aiNet descrita na Seção 5.3.5. GVNS é implementado para verificar se o clássico VNS, usando operadores derivados deste estudo, apresenta "*robustez*" semelhante ao VNS-aiNet para os critérios de otimização avaliados.

Para a realização deste estudo são selecionadas as duas instâncias de treinamento de cada critério de otimização que apresentaram o maior coeficiente de variação durante a fase de calibração dos parâmetros (Seção 5.4.2), gerando oito cenários de avaliação. Para cada cenário são realizadas 100 execuções independentes, e os resultados são analisados sobre a média acumulada por execução independente. Nestes experimentos é utilizado o critério de parada por número de avaliações de função objetivo descrito na Seção 5.4.2.

Para verificar quais variações apresentam diferenças estatísticas significativas em relação à versão de controle do VNS-aiNet, é realizado um *t-test* pareado, considerando a seguinte hipótese: $H_0 : \mu_1 - \mu_2 = 0$ e $H_1 : \mu_1 - \mu_2 \neq 0$, na qual μ_1 é a média dos resultados obtidos pela versão *control* e μ_2 o resultado médio por cada variação da versão *control* descrita acima. Os testes são realizados considerando um nível de confiança de 95% ($\alpha = 0.05$). Os resultados são sumarizados na Seção 5.5.1, sendo apresentado apenas o *p-valor*, e são apresentados em negrito os casos nos quais a hipótese nula (H_0) é rejeitada, aceitando-se, assim, a hipótese alternativa (H_1).

5.4.4 Estudos de caso - VNS-aiNet perante abordagens de estado da arte

A metodologia descrita nesta seção tem como intuito avaliar se a abordagem VNS-aiNet proposta atingiu o segundo objetivo específico (OE2), definido na introdução

deste capítulo. Este objetivo específico define que o algoritmo proposto deve gerar resultados competitivos contra abordagens propostas para a solução de critérios de otimização específicos. Para isto, são realizados quatro estudos de caso, sendo que cada estudo de caso é direcionado a um critério específico e, em cada caso, o algoritmo proposto é comparado às abordagens de estado da arte, usando instâncias de *benchmark* e resultados disponíveis na literatura. O primeiro estudo de caso, mostrado na Seção 5.4.4.1, é realizado para o critério de minimização do TWT (Equação 2.7), através do conjunto de instâncias de *benchmark* propostas por Lin e Hsieh (2014). Já o segundo, mostrado na Seção 5.4.4.2, é considerado para o *Makespan* (Equação 2.2), que é o objetivo mais estudado na literatura. Os experimentos são conduzidos pelo conjunto de instâncias propostas por Al-Salem (2004). Em seguida, na Seção 5.4.4.3, é avaliado o critério de minimização do atraso total (TT) (Equação 2.8). Este estudo de caso apresenta uma restrição nas datas de entrega, as quais não podem sofrer atraso para algumas tarefas prioritárias. Os experimentos são realizados usando instâncias de *benchmark* propostas por Chen (2009). Por fim, na Seção 5.4.4.4, é feito um estudo para o critério de *Total Weighted Completion Time* (Equação 2.3), conduzido com o conjunto de instâncias propostas por Chen (2015) e também apresentando restrições nas datas de entrega.

Para garantir a convergência do algoritmo e ao mesmo tempo um baixo custo computacional, para todos os estudos de caso utilizou-se o critério de parada de 50 gerações sem melhora. Cada instância é processada 10 vezes, uma vez que os métodos são estocásticos. Não é feita comparação direta por instância, pois, de maneira geral, as comparações são feitas pelo valor médio de grupos de instâncias. Uma vez que as abordagens computacionais de estado da arte, utilizadas aqui como controle experimental da nossa metodologia, apresentam configurações de *hardware* diferentes da utilizada neste trabalho, para verificar a factibilidade da abordagem proposta sob o ponto de vista do tempo computacional escalou-se os tempos de respostas das abordagens propostas por um fator de compensação apresentado na Tabela 5.2, obtido pelo indicador *Single Thread Performance* (STP), apresentado no *CPU Benchmarks do PassMark Software*[2].

5.4.4.1 TWT

Para o critério de otimização TWT, o estudo de caso é baseado no conjunto de instâncias descritos na Seção 2.4.1. As análises sobre este conjunto de instâncias são divididas e realizadas separadamente para os subgrupos *small* e *large*. Para ambos os subgrupos, os resultados são subdivididos e apresentados pela combinações dos fatores $\eta \times \tau \times R \times r_\tau$. Os resultados apresentados para cada combinação de fatores são as respostas médias para as instâncias que compõem o grupo. Estes resultados são sumarizados em tabelas presentes na Seção 5.5.2.1. As colunas "RPD" representam o desvio percentual relativo médio, calculado de acordo com a Equação (4.7), sendo o $best_{f(i)}$ para o subconjunto *small* o valor ótimo para a instância i obtido pelo MIP apresentado em Lin e Hsieh (2014). Já para o subconjunto *large*, $best_{f(i)}$ representa o melhor valor conhecido na literatura para a instância i. Para ambos os conjuntos, N_{gi} representa o número de instâncias do grupo $\eta \times \tau \times R \times r_\tau$. A coluna "CPU Time"

[2]Disponível em https://www.cpubenchmark.net/singleThread.html, acessado em 25 de agosto de 2019.

Tabela 5.2: Configuração de *hardware*, STP e fator de compensação utilizados na normalização do tempos computacionais das abordagens estado da arte.

CPU	Critério	Abordagem	STP*	Fator
Intel Pentium E5200 @ 2.50GHz	*	VNS-aiNet	1,016	1.000
Intel Core2 Quad Q6700 @ 2.66GHz	TWT	ACO (Lin et al., 2013)	1,031	1.010
Intel Core2 Quad Q6700 @ 2.66GHz	TWT	IHM (Lin e Hsieh, 2014)	1,031	1.010
Intel Pentium E5200 @ 2.50GHz	TWT	ILS-VND (Diana et al., 2018)	1,016	1.000
Intel Core2 Duo E6700 @ 2.66GHz	$C_{\max}$	HABC (Lin e Ying, 2014)	996	0.980
Intel Core i5 2450M @ 2.50GHz	$C_{\max}$	CSA (Diana et al., 2015)	1,403	1.381
Intel Pentium 4 @ 1.80GHz	$C_{\max}$	ACO-II (Arnaout et al., 2014)	427	0.420
Intel Core 2 Duo @ 2.26 GHz	$C_{\max}$	GALA (Eroglu et al., 2014)	NA	NA
Intel Core i3-370M @ 2.4 GHz	$C_{\max}$	WO (Arnaout, 2019)	918	0.904
Intel Core i7 @ 3.00GHz	$C_{\max}$	B&C (Tran et al., 2016)	NA	NA
AMD Opteron Abu Dhabi 6344 @ 2.6 GHz	$C_{\max}$	MPA (Fanjul-Peyro et al., 2019)	NA	NA
Intel Pentium 4 @ 3.40GHz	TT	IG (Lin et al., 2011)	744	0.732
Intel Pentium 4 @ 3.40GHz	TT	ABC (Ying e Lin, 2012)	744	0.732
Intel Core2 Duo E6400 @ 2.13GHz	TWCT	RRT1 (Chen, 2015)	801	0.788

*NA indica as abordagens que o STP não foi encontrado no PassMark Software na data de acesso.

apresenta o tempo computacional médio em segundos usado por cada abordagem. Como já dito na Seção 5.4.4, os tempos computacionais são reescalados pelo STP, sendo os fatores de escala apresentados na Tabela 5.2. Os resultados do VNS-aiNet para o subgrupo de instâncias *large* são comparados às abordagens de estado da arte ACO (Lin et al., 2013), IHM (Lin e Hsieh, 2014) e ILS-VND (Diana et al., 2018), sendo validados estatisticamente por um teste *Paired Wilcoxon signed-rank* (Wilcoxon, 1945). O teste pareado considera a hipótese $H_0 : md_1 - md_2 = 0$ e $H_1 : md_1 - md_2 < 0$, no qual md_1 e md_2 são, respectivamente, a mediana obtida pelo VNS-aiNet e a mediana das abordagens de estado da arte (ACO, IHM, ILS-VND). Os testes estatísticos são conduzidos considerando um nível de significância de 5% ($\alpha = 0.05$).

5.4.4.2 Makespan

O estudo de caso para a critério de otimização do *makespan* é baseado no conjunto de instâncias proposto por Al-Salem (2004) e descrito na Seção 2.4.2. Este conjunto é subdividido em dois subconjuntos, chamados *small* e *large*. Os resultados para os experimentos sobre este conjunto de instâncias são apresentados na Seção 5.5.2.2, sendo sempre agrupados por conjunto de instâncias $m \times n$, sendo m e n, respectivamente, o número de máquinas e tarefas de cada instância. Para o conjunto *small*, os resultados são apresentados na forma *makespan* médio (coluna C_{max}), do desvio percentual relativo médio (coluna, $\overline{RPD}$) e tempo computacional médio (coluna *CPU Time*). O $\overline{RPD}$ é calculado usando a Equação 4.7, sendo o termo $best_{f(i)}$ o resultado ótimo para a instância i, e N_{gi} o número de instâncias do grupo $m \times n$. Os resultado para este conjunto de instâncias são apenas usados para validar a factibilidade dos resultados do VNS-aiNet, não sendo usado para comparar o VNS-aiNet a abordagens de estado da arte.

Já para o conjunto *large*, os resultados são apresentados em termos do $\overline{RPD}$ (Equação 4.7) em relação a um *lowerbound*, proposto por Al-Salem (2004) e definido

como:

$$\begin{aligned} LB1 &= \frac{1}{m}\sum_{j=1}^{n} \min_{i\in M \ \& \ k\in N} [p_{ij} + s_{ijk}]\,, \\ LB2 &= \max_{j\in N}\left\{ \min_{i\in M \ \& \ k\in N} [p_{ij} + s_{ijk}] \right\}, \\ LB &= \max\{LB1,\ LB2\}\,. \end{aligned} \tag{5.9}$$

Os resultados do conjunto large são apresentados subdivididos para os subgrupos de instâncias *Proc-Domain*, *Setup-Domain* e *Balanced*, sendo estes respectivamente os conjuntos nos quais os tempos de processamento são dominantes, os tempos de setup são dominantes e o grupo no qual os tempos de processamento e setup são balanceados. Os resultados do VNS-aiNet são comparados pelo $\overline{RPD}$ (Equação (4.7)) em relação ao *lowerbound* ($best_{f(i)} = LB$) contra as abordagens metaheurísticas de estado da arte *Ant Colony Optimization* – ACOII (Arnaout et al., 2014); *Restrictive Simulated Annealing* (RSA) (Ying et al., 2012); *Hybrid Artificial Bee Colony* – HABC (Lin e Ying, 2014); *Clonal Selection Algorithm* – CSA (Diana et al., 2015); *Genetic Algorithm with Local Search* – GALA (Eroglu et al., 2014); *Worm Optimization* – WO (Arnaout, 2019). O simbolo "–" é apresentado sempre que uma determinada abordagem não apresentar o $\overline{RPD}$ para o subconjunto $m \times n$. O tempos computacionais médios, por subconjunto $m \times n$, das abordagens estado da arte ACOII, RSA, HABC, CSA e WO, são apresentados de forma reescalada, conforme descrito na Seção 5.4.4. Para cada subgrupo *Proc-Domain*, *Setup-Domain* e *Balanced*, é gerado um *boxplot* comparando o erro de cada abordagem. Para o grupo de instância $m \times n$ é calculado o erro de cada abordagem, sendo o erro dado pela diferença entre o menor $\overline{RPD}$ conhecido para o grupo $m \times n$ e o $\overline{RPD}$ da abordagem. Como o valor por instância não é conhecido para todas as abordagens de estado da arte, optou-se por utilizar o $\overline{RPD}$ das abordagens por grupos $m \times n$. Considerando estas restrições, é realizado um teste estatístico *Paired Wilcoxon signed-rank* (Wilcoxon, 1945) para verificar se há diferença estatística significante entre os resultados da abordagem VNS-aiNet proposta e as abordagens de estado da arte. A análise estatística é realizada por subgrupo (i.e., *Proc-Domain*, *Setup-Domain* e *Balanced*) e pareada por grupo $n \times m$. Todos os testes pareados consideram a hipótese $H_0 : md_1 - md_2 = 0$ e $H_1 : md_1 - md_2 < 0$, sendo md_1 a mediana obtida pelo VNS-aiNet entre os 36 grupos $m \times n$ e md_2 a mediana conhecida das abordagens de estado da arte. O teste estatístico considera um nível de confiança de 95% ($\alpha = 0,05$). Por fim, um teste estatístico de *Friedman* é realizado para gerar um *rank* do desempenho geral das metaheurísticas.

Para o conjunto *large*, o VNS-aiNet ainda é comparado a dois métodos exatos (i.e., B&C (Tran et al., 2016) e MPA (Fanjul-Peyro et al., 2019)). São usados 12 grupos $n \times m$ do conjunto de instâncias *Balanced*, uma vez que estes são as únicas informações disponíveis pelos autores das abordagens B&C e MPA para o conjunto *large*. As comparações são feitas pelo $\overline{RPD}$ e tempo computacional médio.

5.4.4.3 TT - *Total Tardiness*

O conjunto de instâncias proposto por Chen (2009) e descrito na Seção 2.4.3 é utilizado para o estudo de caso para o critério de otimização do atraso total (TT).

Este estudo de caso se difere dos anteriores (i.e., TWT e $C_{\max}$), uma vez que este apresenta restrições que impõem que um conjunto de tarefas prioritárias que não podem sofrer atraso. Se alguma das restrições de datas não forem atendidas, a solução se torna infactível. Conforme descrito na Seções 5.3.2 e 5.3.10, o VNS-aiNet utiliza a penalização da função de *fitness-like* e o não aceite de soluções infactíveis na memória imunológica para contornar este tipo de restrição.

As abordagens IG (Lin et al., 2011) e ABC (Ying e Lin, 2012) apresentam-se como métodos de estado da arte para este problema. As abordagens destinadas a este conjunto de instâncias disponibilizam apenas dados sumarizados. Assim, nenhum teste estatístico é realizado para este conjunto e os resultados são comparados por estatística descritiva. A comparação entre as saídas do VNS-aiNet em relação às abordagens de estado da arte é feita em seis dimensões: (i) número de máquinas (m); (ii) número de tarefas(n); (iii) número de família de tarefas (f); (iv) proximidade entre as datas de entrega (τ); (v) proximidade do instante inicial de planejamento e as datas de entrega (R); e (vi) percentual de tarefas prioritárias (p). Cada uma destas dimensões é composta por dois ou mais níveis. Para cada nível, são agrupados os resultados obtidos para as instâncias que possuem as características que o fator/nível representam. Para este subgrupo de instâncias são calculadas as seguintes métricas: (i) valor médio; (ii) média mínima e máxima; e (iii) tempo computacional médio, sendo este reescalado de acordo com o STP descrito na Tabela 5.2. A média mínima e máxima representa, respectivamente, a média da melhor e pior execução, entre as 10 possíveis, para cada instância por uma abordagem. Os resultados destes indicadores são sumarizados na Seção 5.5.2.3.

5.4.4.4 TWCT

O último estudo de casos é realizado para o critério de otimização da minimização do *Total Weighted Completion Time* (TWCT). Neste estudo de caso, o VNS-aiNet é comparado à abordagem *Record-to-Record Travel* (RRT) (Chen, 2015). Os experimentos deste estudo de caso são realizados sobre um conjunto de instâncias descrito na Seção 2.4.4, baseado nas instâncias de minimização do TT estudada na seção anterior. Assim como na minimização do TT, este estudo de caso também considera restrições na data de entrega de tarefas prioritárias, havendo a possibilidade de soluções infactíveis. Os resultados são comparados através de estatística descritiva, considerando as mesmas dimensões (fatores) descritas na seção anterior. Os resultados do VNS-aiNet são sumarizados seguindo as métricas: (i) valor médio; (ii) média mínima e máxima; e (iii) tempo computacional médio. Já para a abordagem RRT apenas o valor médio e o tempo computacional médio é sumarizado, uma vez que esta é a única informação disponibilizada pelos autores. Os resultados deste estudo de caso são apresentados na Seção 5.5.2.4.

5.5 Resultados

Nesta seção são apresentados os resultados que verificam que o algoritmo VNS-aiNet proposto atinge o objetivo de construir uma abordagem que garante resultados "*robustos*" para diversos critérios de otimização. A Seção 5.5.1 apresenta experimentos que mostram como os componentes do VNS-aiNet contribuem para a abordagem

se adaptar à solução de diferentes critérios de otimização. Já os resultados apresentados na Seção 5.5.2 são destinados à aplicação do VNS-aiNet a critérios de otimização previamente estudados na literatura. Esta seção apresenta os resultados de quatro estudos de casos, mostrando que o VNS-aiNet apresenta resultados compatíveis e/ou superiores a abordagens estado da arte construídas especificamente para determinados critérios de otimização.

5.5.1 Análise de Componentes

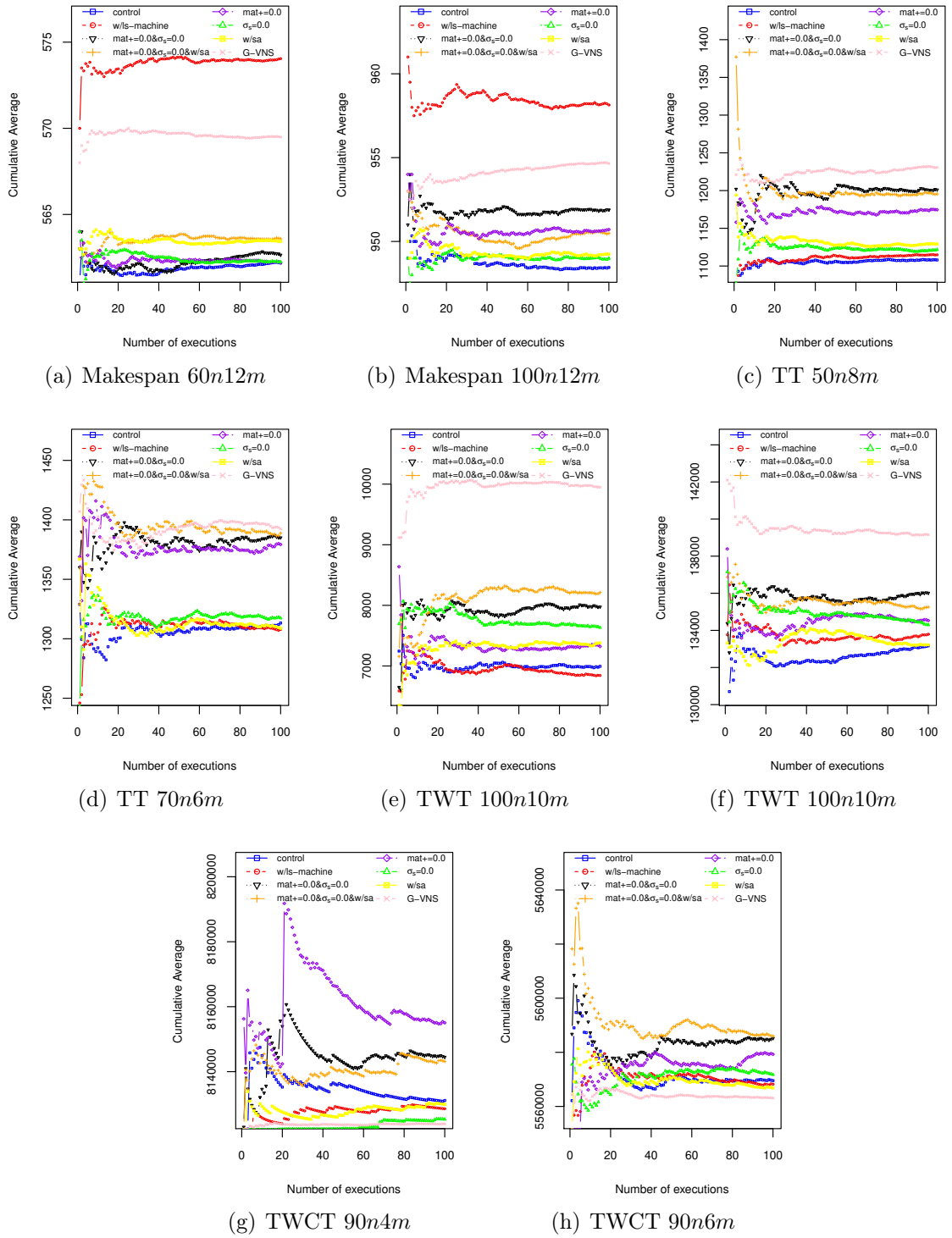
Analisando a Figura 5.5 e a Tabela 5.3, observamos que, devido aos critérios e características do espaço de busca (instâncias), diferentes componentes são acionados para garantir a convergência do algoritmo em diferentes situações. O componente de seleção de máquina que fará parte da busca local (*control* - $w/ls - machine$) tem forte influência na convergência do critério de minimização do *makespan*. Para os demais critérios, não há diferença estatisticamente significativa. Os fatores de mutação auto-adaptativos (*control* - w/sa) influenciam a minimização dos critérios *makespan*, TT e TWT. O operador de supressão (*control* - $\sigma_s = 0.0$) auxilia na convergência de critérios com datas de entrega (TT e TWT). A remoção do fator mat_+ ($mat_+ = 0.0$) apresenta influência em todos os critérios analisados. Esse fator influencia diretamente vários componentes que agregam diversidade ao processo de busca, como a remoção de células maduras (envelhecimento), o aumento da taxa de mutação e a redução da expansão clonal. Além disso, deve-se observar que o acúmulo de remoção de componentes (*control* - $mat_+ = 0.0$ & $\sigma_s = 0.0$ e *control* - $mat_+ = 0.0$ & $\sigma_s = 0.0$ & w/sa) aumentam a deterioração dos resultados alcançados pela versão controle. Em contrapartida, quando analisa-se a hipótese de aumento de desempenho do algoritmo devido à remoção dos mesmos componentes, deve-se observar que, em nenhum cenário analisado, a remoção desses componentes resulta em melhora significativa do desempenho atingido pela versão de controle (*control*). Por fim, quando analisa-se os resultados da versão *control* comparado aos resultados alcançados pela metaheurística clássica GVNS, pode-se observar que a GVNS não generaliza o desempenho para todos os critérios de otimização considerados. Deve-se notar que GNVS utiliza o mesmo operador busca local e as mesmas estruturas de vizinhança de mutação (*Shake*) que o VNS-aiNet.

5.5.2 Estudos de caso

As seções subsequentes analisam os resultados dos estudos de caso comparando o VNS-ainet a abordagens de estado da arte para os seguintes critérios de otimização: TWT (Seção 5.5.2.1), *Makespan* (Seção 5.5.2.2), TT (Seção 5.5.2.3) e TWCT (Seção 5.5.2.4).

5.5.2.1 TWT

De acordo com a Tabela 5.4, o VNS-aiNet é capaz de encontrar, para o critério TWT, os valores ótimos (i.e., definidos por um MIP) para todas as instâncias do grupo de instâncias *small*. Em apenas uma execução (uma instância do conjunto $2.00 \times 0.30 \times 1.00 \times 10.00$), a abordagem proposta não encontrou o resultado ótimo.



(a) Makespan $60n12m$ (b) Makespan $100n12m$ (c) TT $50n8m$

(d) TT $70n6m$ (e) TWT $100n10m$ (f) TWT $100n10m$

(g) TWCT $90n4m$ (h) TWCT $90n6m$

Figura 5.5: Média acumulada das variações nos componentes do VNS-aiNet.

Analisando o tempo computacional, observa-se que o VNS-aiNet consegue convergir para as soluções mínimas em menos de um segundo em todas as configurações, enquanto o MIP, no melhor dos cenários, converge em 9.45 segundos e, no pior caso, tem um custo de 6672 segundos ($\sim$ 1.85 horas). Ao comparar os resultados obtidos pelas metaheurísticas ACO e IHM com os resultados reportados para a abordagem

Tabela 5.3: T-test pareado entre o VNS-aiNet e variações geradas a partir de desativação de componentes do algoritmo.

Instance	$H_0 : \mu_1 - \mu_2 = 0$ and $H_1 : \mu_1 - \mu_2 \neq 0$					
	w/sa	w/ls-machine	$\sigma_s = 0.0$	$mat+ = 0.0$	$mat_+ = 0.0$ & $\sigma_s = 0.0$	$mat+ = 0.0$ & $\sigma_s = 0.0$ & w/sa
Makespan: $100n12m$	**5.47E-02**	**4.73E-37**	1.43E-01	**1.35E-07**	**7.98E-13**	**1.21E-05**
Makespan: $60n12m$	**7.95E-06**	**1.04E-61**	7.95E-01	9.46E-01	1.72E-01	**7.83E-06**
TT: $50n8m$	**1.24E-04**	9.03E-02	**6.58E-03**	**2.34E-12**	**7.48E-15**	**7.95E-16**
TT: $70n6m$	5.94E-01	5.59E-01	6.30E-01	**1.16E-06**	**2.58E-08**	**1.48E-09**
TWT: $100n10m$	**2.01E-03**	1.78E-01	**1.44E-07**	**1.44E-02**	**3.24E-09**	**1.53E-16**
TWT: $100n10m$	9.12E-01	1.31E-01	**7.25E-03**	**3.49E-03**	**8.96E-09**	**1.93E-05**
TWCT: $90n4m$	6.41E-01	6.02E-01	7.09E-02	**8.44E-03**	**2.55E-03**	**2.17E-02**
TWCT: $90n6m$	4.14E-01	7.14E-01	5.45E-01	**2.50E-02**	**2.60E-04**	**6.40E-05**

No teste de hipótese, md_1 sempre corresponde à mediana obtida pelo VNS-aiNet; já md_2 indica a mediana da variação descrita no cabeçalho de coluna. Valores em negrito indicam os casos no qual a hipótese nula (H_0) é rejeitada, aceitando assim a hipótese alternativa H_1 ($\alpha = 0.05$).

proposta, pode-se perceber que, embora apresente um esforço computacional maior do que o alcançado pela VNS-aiNet, essas abordagens não são capazes de atingir os mínimos locais em todas as execuções. A abordagem de estado da arte ILS-VND não apresenta resultados para este conjunto de instâncias.

Para o conjunto de instâncias *large*, a abordagem MIP não é capaz de encontrar as soluções ótimas em tempo computacional viável. Devido a este fato, a Tabela 5.5 apresenta apenas as soluções obtidas pelas metaheurísticas ACO, IHM e ILS-VND. Conforme mostrado na Tabela 5.5, o VNS-aiNet apresenta, em geral, resultados superiores e menor tempo computacional quando comparado às três metaheurísticas de estado da arte. Observa-se que, para as instâncias em que os tempos de *setup* são menos expressivos ($\eta = 0.02$), existe uma distância sutil entre a resposta média do VNS-aiNet e os melhores resultados da literatura. Porém, nos casos que os tempos de *setup* são mais expressivos ($\eta = 2.00$), o VNS-aiNet apresenta uma melhora notável dos resultados. Particularmente, quando as datas de entrega são menos restritas (mais distantes entre si) ($\tau = 0.3$), e os *job ready times* são mais próximas das datas de entrega ($r_\tau = 10.00$), a abordagem apresenta uma distância relativa média maior para os melhores resultados da literatura. Nestes cenários ($\eta = 2.00$ & $\tau = 0.30$ & $r_\tau = 10.00$), as mudanças de sequência permitem maiores atualizações do critério TWT, uma vez que as datas de entrega são mais espaçadas e os tempos de *setup* apresentam alta variabilidade, que permitem uma grande redução nos atrasos por ajustes no sequenciamento. Além disso, observando a Tabela 5.6, é possível concluir que, para todos os grupos de instâncias, o desempenho obtido pelo algoritmo VNS-aiNet corrobora a hipótese de que os resultados obtidos pelo VNS-aiNet superam as abordagens de estado da arte.

5.5.2.2 Makespan

Para o critério de otimização do *makespan*, os resultados para os grupos de instâncias *small* apresentados na Tabela 5.7 indicam que a abordagem analisada encontra as soluções ótimas para todas as instâncias avaliadas. Por sua vez, analisando os resultados do grupo de instâncias *large* mostradas nas Tabelas 5.8, 5.9 e 5.10, observamos que VNS-aiNet apresenta resultados superiores ou iguais aos das abordagens ACOII,

Tabela 5.4: Comparação do VNS-aiNet proposto com IHM, ACO e ILS-VND para instâncias TWT *small*.

$12n3m$				RPD				CPU Time (segundos)*				
η	τ	R	r_τ	ACO	IHM	ILS	VNS-aiNet	MIP	ACO	IHM	ILS	VNS-aiNet
0.02	0,3	0.25	1	0.03	**0.00**	–	**0.00**	4,124.43	1.03	1.01	–	0.14
			10	0.67	**0.00**	–	**0.00**	1,273.31	1.03	1.01	–	0.14
		1	1	0.15	**0.00**	–	**0.00**	650.16	1.03	1.01	–	0.14
			10	0.04	**0.00**	–	**0.00**	260.83	1.03	1.01	–	0.13
	0.9	0.25	1.00	**0.00**	**0.00**	–	**0.00**	6,031.51	1.03	1.01	–	0.13
			10	**0.00**	**0.00**	–	**0.00**	6,672.02	1.03	1.01	–	0.13
		1	1	**0.00**	**0.00**	–	**0.00**	4,143.11	1.03	1.01	–	0.13
			10	**0.00**	**0.00**	–	**0.00**	6,265.96	1.03	1.01	–	0.13
2.00	0.3	0.25	1	1.89	0.57	–	**0.00**	483.58	1.03	1.01	–	0.13
			10	33.26	1.05	–	**0.00**	348.73	1.03	1.01	–	0.12
		1	1	1.69	**0.00**	–	**0.00**	9.45	1.03	1.01	–	0.14
			10	4.63	0.40	–	0.04	29.08	1.03	1.01	–	0.11
	0.9	0.25	1.00	1.08	0.19	–	**0.00**	1,496.18	1.03	1.01	–	0.13
			10	1.08	0.08	–	**0.00**	1,428.24	1.03	1.01	–	0.12
		1	1	0.26	**0.00**	–	**0.00**	809.31	1.03	1.01	–	0.13
			10	0.74	0.02	–	**0.00**	1,041.59	1.03	1.01	–	0.13

*Tempo em segundos reescalado pelo fator STP mostrado na Tabela 5.2 e fornecido pelo *CPU benchmark PassMark Software*.

Tabela 5.5: Comparação do VNS-aiNet proposto com IHM, ACO e ILS-VND para instâncias TWT *large*.

$100n10m$				RPD				CPU Time (in seconds)*			
η	τ	R	r_τ	ACO	IHM	ILS-VND	VNS-aiNet	ACO	IHM	ILS-VND	VNS-aiNet
0.02	0.3	0.25	1	9.75	2.59	0.26	**-0.08**	1.668.54	68.66	44.42	25.30
			10	15.54	2.88	3.07	**-3.40**	1.513.80	64.18	31.07	19.21
		1.00	1	13.85	7.15	0.74	**-0.38**	1.676.12	72.07	45.09	36.39
			10	0.38	0.60	1.19	**-0.03**	1.508.82	66.06	18.95	9.59
	0.9	0.25	1	5.22	1.43	0.12	**-0.13**	1.480.09	59.40	44.92	22.40
			10	5.39	1.09	0.15	**-0.09**	1.702.04	67.24	34.32	21.26
		1.00	1	4.17	0.89	0.10	**-0.12**	1.665.75	68.02	47.09	23.25
			10	4.11	0.64	0.10	**-0.09**	1.666.21	67.97	50.06	22.56
2.00	0.3	0.25	1	5.89	6.86	2.84	**-1.35**	1.665.87	68.18	48.44	44.45
			10	27.53	23.06	16.91	**-25.72**	1.501.38	63.38	43.68	43.82
		1.00	1	10.27	12.72	3.58	**-1.83**	1.674.20	71.00	47.89	47.06
			10	40.01	9.40	22.16	**-30.22**	1.505.26	64.15	39.83	36.85
	0.9	0.25	1	3.06	3.64	2.02	**-1.35**	1.480.76	59.72	47.78	44.90
			10	2.54	3.16	2.30	**-1.50**	1.703.37	67.42	47.38	44.04
		1.00	1	2.95	3.65	2.00	**-0.96**	1.665.55	67.99	49.10	45.78
			10	4.10	4.39	1.90	**-1.24**	1.665.84	68.03	47.92	45.89

*Tempo em segundos reescalado pelo fator STP mostrado na Tabela 5.2 e fornecido pelo *CPU benchmark PassMark Software*.

RSA, GALA e HABC na maioria dos cenários avaliados. Ao comparar a abordagem proposta com o método CSA, o VNS-aiNet apresenta resultados superiores para a maioria dos conjuntos com menor número de máquinas $M = \{4, 6, 8\}$. Entretanto, para os conjuntos com $M = \{10, 12\}$, CSA apresenta um desempenho ligeiramente superior ao que VNS-aiNet na maioria dos cenários. Além disso, comparando o VNS-aiNet com a abordagem WO, observamos que o WO é competitivo em relação ao VNS-aiNet nos cenários, nos quais o número de máquinas é igual a 2, mas, na maior parte dos outros cenários, o VNS-aiNet supera o WO. Ressalta-se que o VNS-aiNet apresenta o menor valor de RPD, em 59 subgrupos $m \times n$ dos 105 avaliados.

O tempo computacional pode ser analisado na Tabela 5.11. Em geral, o VNS-aiNet apresenta maior custo computacional para as instâncias com maior número de

Tabela 5.6: Resultados do teste *paired Wilcoxon signed rank* por grupo de instâncias comparando o VNS-aiNet contra abordagens de estado da arte para o critério TWT.

$100n10m$				$H_0 : md_1 - md_2 = 0$ $H_1 : md_1 - md_2 < 0$*		
η	τ	R	r_τ	ACO	IHM	ILS-VND
0.02	0.3	0.25	1	**1,91E-06**	**1,91E-06**	**3,81E-06**
			10	**1,91E-06**	**1,68E-04**	**1,91E-06**
		1.00	1	**1,91E-06**	**1,91E-06**	**1,91E-06**
			10	**1,43E-02**	**3,60E-02**	**3,81E-06**
	0.9	0.25	1	**1,91E-06**	**1,91E-06**	**1,91E-06**
			10	**1,91E-06**	**1,91E-06**	**1,91E-06**
		1.00	1	**1,91E-06**	**1,91E-06**	**1,91E-06**
			10	**1,91E-06**	**3,81E-06**	**1,91E-06**
2.00	0.3	0.25	1	**1,91E-06**	**3,81E-06**	**1,91E-06**
			10	**1,91E-06**	**1,91E-06**	**1,91E-06**
		1.00	1	**1,91E-06**	**1,91E-06**	**1,91E-06**
			10	**1,91E-06**	**1,91E-06**	**1,91E-06**
	0.9	0.25	1	**1,91E-05**	**3,81E-06**	**1,91E-06**
			10	**1,91E-06**	**1,91E-06**	**5,72E-06**
		1.00	1	**5,72E-06**	**1,91E-06**	**1,91E-06**
			10	**1,91E-06**	**1,91E-06**	**1,91E-06**

*No teste de hipótese, md_1 sempre corresponde à mediana obtida pelo VNS-aiNet; já md_2 indica a mediana da abordagem descrita no cabeçalho de coluna.
◇ Valores em negrito indicam os casos em que a hipótese nula (H_0) é rejeitada, aceitando-se, assim, a hipótese alternativa H_1 ($\alpha = 0.05$).

tarefas e menor número de máquinas. Isso se deve à característica de busca local VND utilizada, que possui um custo computacional diretamente relacionado à razão n/m. Ao comparar os tempos computacionais alcançados pelo VNS-aiNet com abordagens de estado da arte, os tempos computacionais do VNS-aiNet são compatíveis, ou menores, para problemas cuja proporção (n/m) seja inferior a 14. No entanto, na maioria dos cenários avaliados, a proporção n/m é inferior a 14. Dessa forma, o tempo computacional despendido pelo VNS-aiNet mostra-se compatível com as abordagens estado da arte na maior parte dos cenários avaliados. Além disso, o VNS-aiNet não usa heurísticas para construir soluções iniciais. Se necessário, heurísticas construtivas podem ser incorporadas e, consequentemente, podem diminuir o tempo computacional. Essa estratégia é utilizada pelas abordagens HABC e CSA, mas deve ser analisada com cautela, pois pode resultar em diminuição da diversidade e convergência prematura, além de deixar a abordagem específica para determinado critério de otimização.

Analisando a variabilidade dos resultados observados na Figura 5.6, o VNS-aiNet apresenta menor variabilidade nas respostas para os três subconjuntos (*Balanced*, *Proc-Domain* e *Setup-Domain*). Por outro lado, analisando o desempenho geral das abordagens pelo teste *Paired Wilcoxon* (Tabela 5.12) e o teste de Friedman, apresentado nas Tabelas 5.13 e 5.14, concluímos que existem diferenças estatisticamente significativas no desempenho obtido pelo algoritmo VNS-aiNet em comparação com ACOII, RSA, HABC, CSA e WO. Observe que apenas na comparação de VNS-aiNet com CSA no subconjunto *Balanced* a hipótese H_0 não é rejeitada.

A comparação entre os resultados obtidos pelo VNS-aiNET em relação a abordagens determinísticas, para o critério do *makespan*, é mostrada na Tabela 5.15. Duas questões são evidentes nesta tabela: a alta qualidade dos resultados alcançados pelo procedimento proposto, uma vez que, embora os resultados das duas abordagens determinísticas sejam ligeiramente superiores, o $\overline{RPD}$ entre os resultados do

VNS-aiNET e as abordagens determinísticas é inferior a 0.7% em todos os cenários avaliados; além disto, as abordagens determinísticas requerem um alto custo computacional, uma vez que, na maioria dos casos, o tempo de execução excede, em média, 10.800 segundos (3 horas), enquanto o custo computacional do VNS-aiNET é, na pior das hipóteses, inferior a 411 segundos.

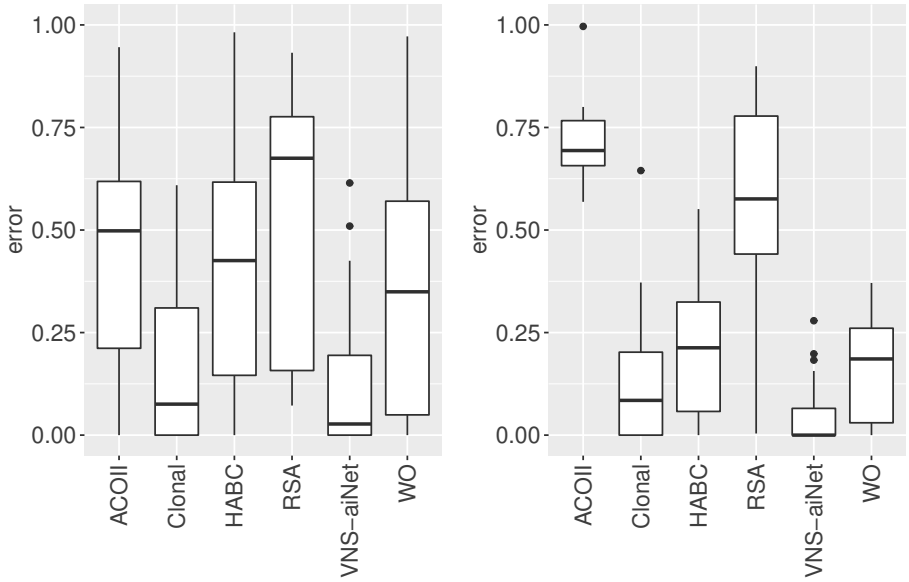
Tabela 5.7: Resultados do VNS-aiNet para o critério de minimização do *makepan* em instâncias de *Setup-domain- small*.

m	n	MIP	VNS-aiNet		
		$C_{\max}$	$C_{\max}$	RPD	CPU Time
2	6	628.47	628.47	**0.00**	0.06
	7	784.40	784.40	**0.00**	0.07
	8	820.27	820.27	**0.00**	0.08
	9	977.13	977.13	**0.00**	0.10
4	6	396.67	396.67	**0.00**	0.06
	7	399.87	399.87	**0.00**	0.07
	8	415.47	415.47	**0.00**	0.08

Os valores em negrito indicam que os resultados obtidos pelo VNS-aiNet são os valores ótimos.

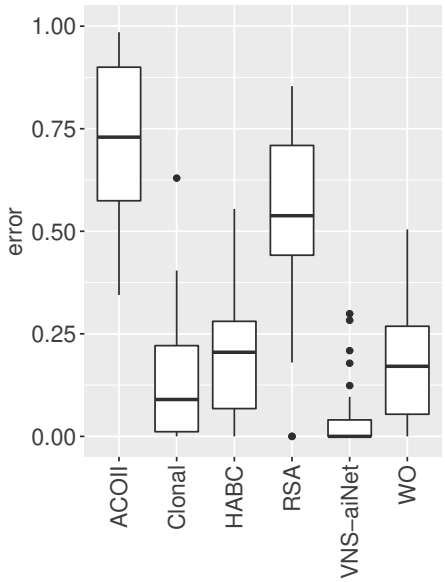
5.5.2.3 TT

Comparando as abordagens para o critério TT, o algoritmo VNS-aiNet apresenta uma resposta média superior às abordagens IG e ABC em todos os níveis de todos os fatores. Além disso, conforme aumenta o número de tarefas e/ou diminui o número de máquinas, o valor de atraso total (TT) obtido pelo VNS-aiNet melhora em relação aos outros dois procedimentos citados. Portanto, esta abordagem proposta mantém o melhor desempenho, mesmo para conjuntos de instâncias maiores. Deve-se notar que o número de famílias, dado pelo fator f, quase não influencia a resposta do algoritmo. Por outro lado, o fator de prioridade das datas de entrega (τ) tem grande impacto na resposta, pois, como esperado, as instâncias com datas de entrega mais distantes entre si ($\tau = 0.4$) reduzem o atraso total. Quando agrupamos fortemente as datas de entrega ($\tau = 0,8$), o valor TT aumenta consideravelmente. Esse comportamento é compartilhado por todas as abordagens. Se o desempenho do método for analisado pelo fator que define o percentual de clientes primários (p), percebe-se que há um aumento no valor de TT à medida que o percentual de clientes primários aumenta. Considerando o tempo computacional despendido, observamos que o VNS-aiNet é viável e pode ser uma técnica computacionalmente atrativa por apresentar uma resposta de melhor qualidade em um tempo computacional relativamente menor.



(a) Balanced

(b) Proc-Domain



(c) Setup-Domain

Figura 5.6: Comparação de *boxplot* para erro do $\overline{RPD}$ médio para o critério do *makespan*.

Tabela 5.8: $\overline{RPD}$ para o critério de minimização do *makespan* para instâncias *large - Balanced*

m	n	$\overline{RPD}$						
		ACOII	RSA	CSA	HABC	GALA	WO	VNS-aiNet
2	20	4.38	4.45	–	**4.14**	4.25	**4.14**	4.16
	40	2.27	2.84	–	2.37	3.46	2.23	**2.21**
	60	1.84	2.53	–	2.1	3.74	1.82	**1.81**
	80	1.41	2.39	–	1.94	3.63	**1.41**	1.61
	100	1.35	2.03	–	1.8	3.45	**1.35**	1.50
	120	1.06	1.95	–	1.69	3.21	**1.06**	1.38
4	20	8.68	8.74	8.84	8.48	8.73	**8.47**	8.51
	40	5.91	5.74	5.57	5.13	7.34	5.07	**4.96**
	60	4.54	4.75	4.27	4.08	6.44	4.05	**3.82**
	80	3.97	4.59	3.75	3.88	6.20	3.83	**3.35**
	100	3.54	4.08	3.41	3.45	5.54	3.40	**3.04**
	120	3.00	3.62	3.10	3.33	5.29	3.10	**2.79**
6	20	**23.05**	23.12	23.10	**23.05**	23.07	**23.05**	**23.05**
	40	10.52	9.37	9.19	8.77	10.37	8.61	**8.61**
	60	7.12	6.51	5.54	5.79	8.49	5.74	**5.40**
	80	7.09	6.91	5.82	5.95	8.03	5.89	**5.35**
	100	5.58	5.63	4.53	4.84	7.13	4.86	**4.39**
	120	4.52	5.37	4.04	4.57	6.95	4.52	**3.90**
8	20	27.34	27.14	27.10	**27.04**	27.29	**27.04**	27.06
	40	9.63	9.03	8.03	8.10	11.24	8.07	**8.01**
	60	11.21	10.49	9.74	9.62	11.48	9.60	**9.44**
	80	7.41	6.90	**5.52**	6.00	9.22	6.00	5.70
	100	8.11	7.53	6.20	6.72	8.78	6.67	**6.11**
	120	5.70	6.66	**4.65**	5.33	8.41	5.24	4.71
10	20	16.46	15.79	15.26	**15.13**	16.55	15.13	15.17
	40	13.40	10.55	**9.12**	9.35	13.02	9.33	9.37
	60	10.09	9.07	**6.99**	7.62	11.82	7.59	7.50
	80	8.15	7.95	6.20	6.96	10.95	6.96	6.59
	100	8.15	7.07	**5.62**	6.35	10.32	6.35	5.88
	120	6.99	6.66	5.25	6.23	10.37	6.13	5.45
12	20	34.15	32.43	32.41	**32.31**	–	**32.31**	32.36
	40	25.94	24.94	24.31	24.27	–	**24.25**	24.30
	60	10.45	9.85	**7.67**	8.22	–	8.20	8.29
	80	11.97	10.98	**9.42**	10.40	–	10.39	9.68
	100	12.18	11.67	**10.16**	11.33	–	11.18	10.25
	120	7.60	7.29	**5.67**	6.87	–	6.82	6.10

Os valores em negrito indicam a melhor média de $\overline{RPD}$ por grupo de instâncias $m \times n$.

Tabela 5.9: $\overline{RPD}$ para o critério de minimização do *makespan* para instâncias *large - Proc-domain*.

m	n	$\overline{RPD}$						
		ACOII	RSA	CSA	HABC	GALA	WO	VNS-aiNet
2	20	2.59	2.61	–	**2.47**	–	**2.47**	2.48
	40	1.53	1.68	–	1.45	–	1.40	**1.38**
	60	1.08	1.61	–	1.30	–	**1.08**	1.09
	80	**0.90**	1.41	–	1.20	–	**0.90**	1.00
	100	2.72	1.31	–	1.19	–	1.16	**0.96**
	120	1.56	1.24	–	1.05	–	1.01	**0.85**
4	20	5.89	5.24	5.24	**5.09**	–	**5.09**	5.11
	40	3.95	3.35	3.16	3.00	–	2.98	**2.95**
	60	2.98	2.73	2.49	2.42	–	2.38	**2.25**
	80	2.67	2.48	2.25	2.30	–	2.25	**2.03**
	100	2.33	2.35	1.98	2.00	–	1.98	**1.76**
	120	2.39	2.51	1.95	2.01	–	1.98	**1.72**
6	20	22.30	21.66	**21.61**	**21.61**	–	**21.61**	**21.61**
	40	9.06	7.35	7.43	7.02	–	**6.79**	6.92
	60	5.10	3.74	3.24	3.23	–	3.23	**3.19**
	80	7.03	5.53	5.33	5.34	–	5.24	**4.96**
	100	4.40	3.78	3.17	3.40	–	3.35	**3.03**
	120	3.30	3.13	2.44	2.53	–	2.52	**2.27**
8	20	25.80	**24.39**	24.41	**24.39**	–	**24.39**	**24.39**
	40	7.56	5.57	**4.77**	4.82	–	4.82	4.84
	60	10.70	9.13	8.60	8.57	–	8.49	**8.23**
	80	5.71	4.10	**3.34**	3.50	–	3.50	3.38
	100	7.09	5.84	5.21	5.51	–	5.40	**5.03**
	120	4.01	3.35	2.88	2.89	–	2.91	**2.77**
10	20	12.30	9.59	9.23	**9.16**	–	**9.16**	9.20
	40	8.76	6.34	**5.30**	5.53	–	5.53	5.50
	60	7.76	5.26	**4.16**	4.36	–	4.36	4.34
	80	6.44	4.54	**3.66**	3.79	–	3.79	3.77
	100	5.41	4.06	**3.39**	3.56	–	3.56	3.49
	120	5.10	3.74	**3.18**	3.47	–	3.47	3.21
12	20	30.60	27.42	27.33	27.28	–	27.28	**27.27**
	40	25.70	22.87	22.61	22.60	–	**22.52**	22.57
	60	8.67	5.75	**4.49**	4.83	–	4.83	4.77
	80	10.70	8.23	7.57	8.10	–	7.92	**7.55**
	100	12.10	9.86	9.34	9.72	–	9.50	**9.24**
	120	6.14	4.18	**3.38**	3.72	–	3.71	3.53

Os valores em negrito indicam a melhor média de $\overline{RPD}$ por grupo de instâncias $m \times n$.

Tabela 5.10: $\overline{RPD}$ para o critério de minimização do *makespan* para instâncias *large - Setup-domain*.

m	n	$\overline{RPD}$						
		ACOII	RSA	CSA	HABC	GALA	WO	VNS-aiNet
2	20	2.57	2.51	–	2.37	–	**2.37**	2.39
	40	1.67	1.71	–	1.58	–	1.53	**1.45**
	60	1.19	1.61	–	1.33	–	1.18	**1.15**
	80	0.87	1.24	–	1.17	–	**0.87**	0.97
	100	1.57	1.28	–	1.09	–	1.07	**0.91**
	120	1.47	1.17	–	1.06	–	1.04	**0.92**
4	20	6.04	5.15	**4.71**	4.99	–	4.99	5.01
	40	4.23	3.52	3.68	3.11	–	3.10	**3.05**
	60	3.39	2.89	2.47	2.60	–	2.58	**2.40**
	80	2.70	2.41	2.17	2.22	–	2.22	**1.97**
	100	2.42	2.34	2.22	2.05	–	2.04	**1.85**
	120	2.00	2.19	1.82	1.90	–	1.82	**1.66**
6	20	22.60	**21.70**	21.93	**21.70**	–	**21.70**	**21.70**
	40	8.99	7.30	7.08	7.04	–	**6.95**	6.96
	60	4.99	3.82	3.29	3.42	–	3.40	**3.23**
	80	6.71	5.40	5.26	5.12	–	5.01	**4.86**
	100	4.61	3.72	3.22	3.33	–	3.31	**3.01**
	120	3.61	2.90	2.40	2.52	–	2.52	**2.32**
8	20	26.20	24.26	24.31	**24.08**	–	**24.08**	24.10
	40	7.87	5.65	4.94	4.90	–	4.89	**4.88**
	60	10.70	8.59	8.47	8.50	–	8.24	**8.14**
	80	5.27	4.00	3.36	3.50	–	3.49	**3.27**
	100	7.12	5.65	5.23	5.35	–	5.32	**4.93**
	120	4.08	3.35	2.83	2.96	–	2.96	**2.75**
10	20	11.90	9.22	8.96	**8.81**	–	**8.81**	**8.81**
	40	8.90	6.15	**5.33**	5.41	–	5.41	5.40
	60	7.60	5.08	**4.27**	4.48	–	4.46	4.37
	80	6.73	4.78	**3.73**	4.00	–	4.00	3.93
	100	5.53	4.07	3.39	3.43	–	3.43	**3.36**
	120	4.86	3.82	**3.08**	3.63	–	3.58	3.26
12	20	30.40	**27.14**	27.50	**27.14**	–	27.14	**27.14**
	40	25.50	22.79	22.53	**22.39**	–	**22.39**	22.44
	60	8.89	5.71	**4.48**	4.87	–	4.86	4.76
	80	10.50	8.14	7.54	7.96	–	7.85	**7.53**
	100	11.80	9.84	9.35	9.86	–	9.81	**9.31**
	120	6.09	4.30	**3.45**	3.58	–	3.58	3.57

Os valores em negrito indicam a melhor média de $\overline{RPD}$ por grupo de instâncias $m \times n$.

Tabela 5.11: Tempo computacional médio (em segundos) para o critério de minimização do *makespan*.

		CPU-Time (in seconds)						ratio
		ACOII	RSA	CSA	HABC	WO	VNS-aiNet	n/m
2	20	6.4	0.1	–	1.3	1.6	0.6	10.0
2	40	12.7	0.7	–	2.5	12.5	6.8	20.0
2	60	25.4	2.2	–	3.8	48.3	28.6	30.0
2	80	41.4	4.2	–	5.1	74.9	79.0	40.0
2	100	62.3	7.3	–	6.3	160.0	171.2	50.0
2	120	86.9	12.6	–	7.6	180.9	325.1	60.0
4	20	6.1	0.1	2.8	2.5	0.4	0.4	5.0
4	40	9.6	0.8	5.5	5.1	5.5	3.0	10.0
4	60	19.2	2.3	8.3	7.6	16.2	11.1	15.0
4	80	30.9	4.7	11.0	10.1	40.5	29.8	20.0
4	100	55.4	8.3	13.8	12.7	112.1	61.4	25.0
4	120	72.3	15.8	16.6	15.2	137.4	111.3	30.0
6	20	6.0	0.2	4.1	20.3	1.0	0.2	3.3
6	40	9.4	0.9	8.3	7.6	4.8	1.7	6.6
6	60	18.7	2.2	12.4	11.4	8.5	7.6	10.0
6	80	30.2	4.6	16.6	15.2	16.0	10.7	13.3
6	100	54.1	9.9	20.7	19.0	48.7	34.9	16.6
6	120	74.9	16.0	24.9	22.8	103.1	66.4	20.0
8	20	6.6	0.2	5.5	5.1	1.3	0.2	2.5
8	40	10.2	0.9	11.0	10.1	1.9	1.6	5.0
8	60	20.5	2.3	16.6	15.2	8.4	3.4	7.5
8	80	33.1	6.0	22.1	20.2	8.3	13.7	10.0
8	100	59.3	12.1	27.6	25.3	34.7	18.0	12.5
8	120	82.1	18.2	33.1	30.4	50.4	45.1	15.0
10	20	6.7	0.2	6.9	6.3	3.5	0.3	2.0
10	40	10.6	1.0	13.8	12.7	4.6	1.5	4.0
10	60	21.2	3.1	20.7	19.0	10.9	4.3	6.0
10	80	34.1	7.0	27.6	25.3	9.5	10.2	8.0
10	100	61.1	15.2	34.5	31.6	54.7	20.3	10.0
10	120	84.5	27.3	41.4	38.0	83.2	34.1	12.0
12	20	–	0.3	8.3	7.6	5.5	0.2	1.6
12	40	–	0.9	16.6	15.2	2.5	0.8	3.3
12	60	–	3.5	24.9	22.8	7.1	4.2	5.0
12	80	–	8.1	33.1	30.4	11.7	6.0	6.6
12	100	–	11.9	41.4	38.0	41.9	7.6	8.3
12	120	–	28.7	49.7	45.6	49.5	26.8	10.0

*Tempo reescalado pelo fator presente na Tabela 5.2 e dado por *CPU Benchmarks of PassMark Software*.

Tabela 5.12: Resultados do teste *Paired Wilcoxon signed rank* por grupo de instâncias comparando o VNS-aiNet contra abordagens de estado da arte para o critério *makespan*.

	Balanced		Proc-domain		Setup-domain	
VNS-aiNet vs.	W	p-value	W	p-value	W	p-value
ACOII	0.0	**1.35E-06**	0.0	**9.31E-10**	0.0	**9.12E-07**
RSA	0.0	**9.10E-07**	0.0	**1.35E-06**	0.0	**2.00E-06**
CSA	192.0	2.05E-01	122.0	**2.00E-02**	114.5	**7.83E-03**
HABC	28.0	**2.19E-05**	14.5	**9.29E-06**	14.0	**1.38E-05**
WO	29.5	**4.07E-05**	39.0	**9.82E-05**	21.0	**2.85E-05**

Valores em negrito indicam os casos em que a hipótese nula (H_0) é rejeitada, aceitando-se, assim, a hipótese alternativa H_1 ($\alpha = 0.05$).

Tabela 5.13: Teste de Friedman para o desempenho geral dos algoritmos para o critério de minimização do *makespan*.

	Chi-squared	df	p-value
Friedman rank sum test	324.15	5	**<2.2E-16**

Tabela 5.14: Classificação de algoritmos para o critério de minimização do *makespan* obtido através do Teste de Friedman.

Algorithm	Sum of ranks	Position*
VNS-aiNet	155.0	a
CSA	229.0	b
WO	237.5	b
HABC	299.5	c
RSA	449.0	d
ACOII	520.0	e

*Algoritmos com a mesma letra não apresentam diferença estatística significativa ($\alpha = 0.05$).

Tabela 5.15: Resultado para o critério de minimização do *makespan* comparando o VNS-aiNet e abordagens determinísticas.

m	n	Avg. $C_{\max}$			$\overline{RPD}$ – VNS-aiNet		CPU Time (in seconds)		
		MPA	Ttran	VNS-aiNet	MPA	Ttran	MPA	Ttran	VNS-aiNet
2	40	**2390.33**	**2390.33**	2392.33	0.08%	0.08%	14.92	10800.00	**6.80**
	60	**3558.33**	**3558.33**	3566.80	0.24%	0.24%	43.25	10800.00	**29.69**
	80	**4712.93**	**4712.93**	4732.20	0.41%	0.41%	185.13	10800.00	**77.16**
	100	**5866.93**	5867.00	5898.27	0.53%	0.53%	301.06	10800.00	**190.47**
	120	**7051.47**	7051.67	7094.67	0.61%	0.61%	583.91	10800.00	**410.42**
4	60	**1704.27**	1705.73	1709.00	0.28%	0.19%	8133.05	10800.00	**11.21**
	80	**2261.67**	2262.20	2269.60	0.35%	0.33%	9855.50	10800.00	**29.24**
	100	**2804.67**	2806.07	2817.00	0.44%	0.39%	10817.59	10800.00	**58.84**
	120	**3353.80**	3356.40	3376.07	0.66%	0.59%	10562.86	10800.00	**110.56**
6	100	1857.73	1863.07	**1855.27**	-0.13%	-0.42%	10807.25	10800.00	**36.58**
	120	**2204.60**	2208.33	2215.07	0.47%	0.30%	10800.87	10800.00	**112.32**
8	120	**1639.67**	1643.60	1649.53	0.60%	0.36%	10838.66	10800.00	**44.33**

*Tempo computacional não é redimensionado pelo fator STP. As abordagens MPA e Ttran não relatam detalhes (ou o STP não está disponível no *software PassMark*) as configurações de *hardware* relatada nos manuscritos.

Tabela 5.16: Comparação para o critério de TT entre VNS-aiNet e abordagens de estado da arte IG e ABC.

		Average			Average Min.			Average Max.			CPU Time (in seconds)		
		IG	ABC	VNS-aiNet	IG	ABC	VNS-aiNet	IG	ABC	VNS-aiNet	IG	ABC	VNS-aiNet
n	30	1396.00	1393.50	**1390.61**	1389.70	1389.20	**1388.82**	1404.80	1402.80	**1394.72**	0.75	0.92	**0.69**
	50	2955.80	2939.70	**2912.79**	2912.30	2903.70	**2896.33**	3010.70	3001.00	**2947.88**	4.55	4.41	**3.56**
	70	6061.10	5992.30	**5913.80**	5942.30	5891.20	**5862.01**	6199.20	6129.20	**6013.39**	16.13	16.94	**12.14**
	90	9943.80	9838.30	**9666.89**	9731.50	9649.30	**9558.46**	10169.00	10054.10	**9849.46**	47.25	46.16	**31.70**
m	4	7767.00	7693.70	**7586.02**	7644.80	7590.20	**7540.23**	7894.90	7819.70	**7669.36**	**17.50**	18.31	19.16
	6	4760.40	4715.60	**4651.82**	4666.60	4634.80	**4604.80**	4871.20	4826.50	**4738.18**	17.54	18.25	**10.52**
	8	2740.10	2713.60	**2675.23**	2670.40	2650.10	**2634.18**	2821.60	2794.00	**2746.54**	16.47	14.77	**6.39**
f	7	5092.10	5042.10	**4970.61**	4994.20	4956.80	**4922.90**	5198.30	5149.30	**5055.81**	17.13	17.26	**11.93**
	8	5086.20	5038.80	**4971.44**	4993.70	4959.90	**4929.91**	5193.50	5144.20	**5046.91**	17.20	16.96	**12.12**
τ	0,4	1.40	1.40	**1.30**	1.30	1.30	**1.29**	1.50	1.60	**1.36**	**0.05**	0.12	0.19
	0,8	10177.00	10080.50	**9940.75**	9986.60	9915.40	**9851.52**	10390.40	10291.90	**10101.36**	34.29	34.10	**23.86**
R	0,4	8924.20	8828.30	**8754.10**	8800.90	8733.40	**8704.05**	9060.00	8948.90	**8857.58**	14.95	20.73	**11.29**
	1	1254.10	1253.60	**1187.95**	1187.00	1183.30	**1148.76**	1331.90	1344.60	**1245.15**	19.39	13.49	**12.76**
p	0,2	4922.70	4869.90	**4810.28**	4837.80	4799.20	**4768.53**	5016.20	4960.30	**4881.64**	16.69	18.00	**11.51**
	0,3	5255.70	5212.00	**5131.77**	5150.10	5117.50	**5084.29**	5375.60	5333.20	**5221.09**	**0.75**	16.22	12.54
Total		5089.20	5041.00	**4971.02**	4993.90	4958.35	**4926.41**	5195.90	5146.70	**5051.36**	17.17	17.11	**12.02**

*Tempo reescalado pelo fator presente na Tabela 5.2 e dado por *CPU Benchmarks PassMark Software*.

5.5.2.4 TWCT

De acordo com a Tabela 5.17, o VNS-aiNet atinge resultados médios superiores à metaheurística RRT1 em todos os níveis dos fatores avaliados do critério TWCT. Por sua vez, para todos os fatores analisados, os resultados médios máximos (coluna Max) do VNS-aiNet são inferiores aos resultados médios do RRT1. Adicionalmente, em todos os níveis dos fatores referentes, as datas de entrega (τ, R, p), bem como o fator família (f), não apresentam alteração significativa na resposta do algoritmo. Quando se compara o tempo computacional gasto, a abordagem proposta atinge os resultados com um custo computacional inferior ao RRT1, indicando que o melhor desempenho obtido não se deve a um maior esforço computacional.

Tabela 5.17: Comparação para o critério TWCT entre VNS-aiNet e abordagens de estado da arte RRT1.

		VNS-aiNet				RRT1	
		Min.	Avg.	Max.	Time	Avg.	Time*
n	30	767725.55	**767781.18**	768035.77	1.26	771071.22	0.52
	50	1646870.00	**1647623.23**	1649386.16	5.36	1664479.80	4.73
	70	3037373.83	**3040015.34**	3046189.48	16.82	3083568.00	26.09
	90	5191244.67	**5197351.41**	5210983.44	57.47	5296431.51	106.08
m	4	3722828.42	**3725014.57**	3730273.93	30.32	3769160.42	55.56
	6	2444533.35	**2446936.99**	2452558.14	15.14	2486637.19	28.96
	8	1815048.77	**1817626.80**	1823114.07	15.23	1855865.29	18.53
f	7	2652529.80	**2655057.40**	2660826.89	18.53	2697133.62	34.02
	8	2669077.23	**2671328.17**	2676470.54	21.93	2710641.65	34.69
τ	0.4	2577575.07	**2578900.17**	2582325.27	17.91	2607391.23	33.94
	0.8	2744031.95	**2747485.41**	2754972.16	22.55	2800384.04	34.77
R	0.4	2674314.80	**2676766.57**	2682485.50	19.16	2719464.84	33.31
	1	2647292.22	**2649619.00**	2654811.92	21.29	2688310.43	35.40
p	0.2	2639566.73	**2641697.39**	2646660.61	19.43	2678247.36	33.98
	0.3	2682040.30	**2684688.19**	2690636.82	21.03	2729527.91	34.74
Total		2660803.51	**2663192.79**	2668648.71	20.23	2703887.63	34.36

*Tempo reescalado pelo fator presente na Tabela 5.2 e dado por *CPU Benchmarks PassMark Software*.

5.6 Discussão e conclusões

Neste capítulo propomos o projeto uma abordagem híbrida, baseada na metaheurística VNS-aiNet, destinada à resolução de diversos critérios de otimização em ambientes UPMSMDST. Para atingir este objetivo, determinamos que dois objetivos específicos deveriam ser atingidos: OE1 indica que abordagem deve possuir operadores que atuam em conjunto, para permitir que a mesma seja guiada por diferentes critérios de otimização de forma a encontrar resultados "*robustos*"; OE2 define que, para considerar os resultados alcançados como "*robustos*", a abordagem proposta deve apresentar desempenho comparável às abordagens do estado da arte para critérios de otimização específico, sem a adição de considerável esforço computacional.

Para avaliar se o primeiro objetivo específico é atendido, realizamos uma análise de componentes do algoritmo proposto. Neste é verificado que diversos operadores do VNS-aiNet agem em conjunto para guiar a abordagem na busca por soluções. É observado que determinados operadores, quando inativados, não influem na respostas de determinados critérios de otimização, mas têm papel fundamental para outros.

Ainda é verificado que, mesmo nos cenários que os operadores não acarretam em incremento de desempenho, eles também não implicam na degeneração dos resultados atingidos pela abordagem. Nesta análise também é comparado o desempenho do VNS-aiNet proposto a uma abordagem GVNS construída a partir dos operadores propostos neste trabalho. Através destes experimentos, observou-se que o GVNS não conseguiu generalizar o seu desempenho para todos os critérios de otimização avaliados. Através destes indicadores, concluímos que a abordagem proposta atingiu de forma satisfatória o primeiro objetivo específico.

Além destes experimentos, para verificar a viabilidade do segundo objetivo específico, quatro estudos de caso são realizados. Cada estudo de caso está relacionado a um critério de otimização específico, possuindo um conjunto próprio de instâncias e resultados de referência definidos por abordagens de estado da arte. Nos estudos de caso destinados aos critérios de otimização de minimização do TWT, TT e TWCT, o procedimento proposto apresentou desempenho superior às abordagens da literatura. Para o primeiro critério (TWT), análises estatísticas inferenciais confirmaram que os resultados apresentados pelo VNS-aiNet apresentam diferença estatística significativa em relação às abordagens de estado da arte. Para os dois últimos critérios (TT e TWCT), os resultados analíticos das abordagens do estado da arte não são disponíveis pelos autores; em consequência, a validação por testes estatísticos não é possível. Em relação ao critério de minimização do *makespan*, o procedimento proposto apresentou desempenho superior às metaheurísticas de estado da arte em mais da metade dos cenários avaliados. Estes resultados apenas reforçam o poder de convergência do VNS-aiNet, uma vez que o *makespan* é o critério de otimização mais estudado para ambientes UPMSMDST e as abordagens do estado da arte incorporam, em suas estruturas, um conjunto de características que auxiliam na convergência para minimizar o *makespan*, mas fazem estas abordagens se restringirem ao tratamento deste critério. Em relação ao tempo computacional gasto, a abordagem convergiu, na maioria dos casos, em um tempo computacional compatível ou inferior aos apresentados pelas metaheurísticas do estado da arte. No entanto, esse fato não indica que a abordagem proposta supera as abordagens metaheurísticas existentes na literatura, mas indica que não utilizou um esforço computacional excessivo. Quando consideramos as abordagens determinísticas, embora o VNS-aiNet tenha apresentado resultados ligeiramente inferiores, teve, por outro lado, um custo computacional muito inferior ao apresentado por essas abordagens.

Embora a abordagem aqui proposta seja facilmente adaptável a outros critérios e restrições de otimização, algumas advertências devem ser identificadas. Este estudo considera apenas ambientes UPMSMDST com tempos de processamento fixos. No entanto, a literatura de sequenciamento recente tem incorporado ao tempo de processamento fatores como efeito de aprendizagem (Wu et al., 2020a; Azzouz et al., 2020) e efeito de deterioração (Wu et al., 2019; Salehi et al., 2020). Em pesquisas futuras, pode-se avaliar como a presença desses fatores influencia a resposta do algoritmo proposto. Além disso, uma vez que um grande número de permutações é usado na fase de busca local, a avaliação da função objetivo pode tornar a abordagem proposta, na forma como está posta, proibitiva para problemas com um alto custo de avaliação desta função. É o caso, por exemplo, do critério de otimização estudado no próximo capítulo, da minimização da soma total ponderada de atraso e atrasos com janelas de tempo e possibilidade de tempos ociosos, em que a ordem de complexidade da função

objetivo é $O(n^2)$ (Rosa et al., 2017).

Capítulo 6

Influência de janelas de tempo no projeto de metaheurísticas em ambientes de máquinas não relacionadas

Este capítulo apresenta uma discussão a respeito da importância e dificuldade da inserção de janelas de tempo em conjunto com tempo ociosos no projeto de metaheurísticas destinadas a problemas de sequenciamento de tarefas em ambientes UPMSMDST. A primeira seção do capítulo mostra uma contextualização da importância de se estudar problemas com janelas de tempo em ambientes UPMSMDST, sendo apresentadas três questões em aberto que o capítulo busca investigar. A Seção 6.2 apresenta uma definição formal do problema e a formulação matemática proposta através de uma formulação inteira mista. Em seguida, a Seção 6.3 é dedicada a apresentar um método de inserção de tempos ociosos para o problema de sequenciamento em ambientes UPMSMDST com janelas de tempo, sendo este método adaptado a partir de uma método proposto para ambientes de máquinas únicas. Além disto, é projetado um algoritmo de busca local para ser utilizado em conjunto com metaheurísticas, uma vez que a inserção de tempos ociosos acarreta em considerável incremento na ordem de complexidade. Após, a Seção 6.4 apresenta a metodologia utilizada para integrar os algoritmos de busca local e inserção de tempos ociosos a cinco metaheurísticas, previamente propostas na literatura, utilizadas aqui para investigar as questões definidas na introdução do capítulo. A seção também apresenta a metodologia utilizada para os experimentos realizados neste capítulo, descrevendo-se: (i) a geração de um conjunto de instâncias para o problema; (ii) os artefatos e condições computacionais utilizados para a condução dos experimentos; (iii) os métodos de calibração de parâmetros utilizados juntos às metaheurísticas; (iv) a metodologia utilizada para a comparação dos resultados entre as abordagens avaliadas; e (v) a metodologia utilizada para verificar a influência da janela de tempo nos resultados obtidos pelo sequenciamento. Posteriormente, a Seção 6.5 apresenta os resultados computacionais alcançados e uma discussão dos mesmos. Por fim, a Seção 6.6 apresenta as conclusões obtidas pelo capítulo. Destacamos que este capítulo é fortemente baseado no artigo Diana et al. (2021b), submetido para publicação à *Computers & Operations Research*.

6.1 Introdução

Problemas que envolvem datas de entrega têm sido discutidos desde da publicação dos primeiros modelos de sequenciamento de tarefas em máquinas paralelas (e.g. McNaughton (1959); Barnes e Brennan (1977)). Não por acaso, portanto, a minimização do atraso total ponderado (*Total Weighted Tardiness* – TWT) é um dos critérios de otimização mais estudados na literatura de sequenciamento. Apesar da minimização do TWT ser um problema bastante relevante para diversos ambientes de manufatura, cenários nos quais os custos de armazenamento de materiais/insumos são proibitivos ou inviáveis, a minimização do TWT tende a se mostrar ineficiente ou até mesmo gerar soluções infactíveis. Como destacado por Shirvani et al. (2014), a indústria de alimentos perecíveis é um cenário no qual o sequenciamento por minimização do TWT se torna inviável devido a perecibilidade dos insumos e produtos gerados neste tipo de cadeia produtiva. Assim como na indústria de alimentos perecíveis, diversas outras cadeias produtivas apresentam restrições que necessariamente obrigam o sequenciamento ser guiado por uma filosofia *Just-in-Time* (JIT).

O principal objetivo do JIT é garantir que uma determinada tarefa seja executada no instante de tempo em que não acarrete atraso nem seja processada com antecedência. Além disto, o fornecimento de materiais/insumos deve ser integrado ao sequenciamento, para evitar custos/restrições relativos ao processo de estocagem (Janiak et al., 2015). Deve-se observar que existe uma série de fatores envolvidos em uma cadeia produtiva que dificultam este tipo de filosofia. Contratos normalmente são negociados considerando-se as necessidades dos clientes e não as restrições do JIT; materiais/insumos, que deveriam estar disponíveis apenas no exato instante requerido para evitar custo com armazenamento, muitas vezes esbarram em restrições de fornecedores e contratos para compra em lote para redução de custos logísticos. Além destes fatores, as políticas de JIT não se adequam bem a ambientes que requerem a utilização máxima dos equipamentos. Estas e outras restrições impostas pelos ambientes reais de manufatura dificultam a utilização de datas de entrega rigidamente impostas. Estas dificuldades podem ser minimizadas pela adoção de janelas de tempo e ambientes menos sobrecarregados, que permitam a utilização de tempos ociosos pelo ambiente de máquinas.

Adoção de datas de entrega rigidamente definidas, muitas vezes, podem ser evitadas na confecções de contratos, sendo substituídas por uma janela de tempo de entrega. Neste modelo, clientes permitem que tarefas sejam entregues dentro do período contido em uma janela de tempo pré-definida. Isto tende a provocar redução nos custos relativos a avanço, já que: (i) permite maior flexibilização das datas de entrega; (ii) toda tarefa que terminar dentro da janela de tempo pode ser imediatamente entregue; (iii) não precisa incorrer em custos exorbitantes de estocagem. Uma consequência direta das janelas de tempo é uma maior flexibilização do sequenciamento.

Ao mesmo tempo que problemas de sequenciamento com janela de tempo flexibilizam a utilização de JIT, levam ao aumento da complexidade para a criação de heurísticas computacionais, principalmente quando é permitida a inserção de tempos ociosos entre tarefas. Heurísticas computacionais para o sequenciamento de tarefas, guiadas por uma política JIT baseada na minimização total dos atrasos e avanços ponderados, que aceite tempos ociosos, apresentam uma característica peculiar que

as fazem comumente ser divididas em duas etapas. A primeira etapa, compartilhada por diversos critérios de otimização, consiste na definição da sequência e atribuição de tarefas a serem processadas nas máquinas. A segunda etapa consiste na resolução de um subproblema de otimização, que define a inserção dos tempos ociosos (e.g. (Nogueira et al., 2014; Rosa et al., 2017)). Esta característica implica, consequentemente, no incremento da complexidade computacional destas abordagens heurísticas, elevando sua ordem de complexidade (Rosa et al., 2017). Apesar da complexidade computacional, considerar a possibilidade de tempos ociosos no sequenciamento JIT é de extrema importância, um vez que a não inserção destes acarreta na minimização dos atrasos, mas, ao mesmo tempo, maximiza os custos relacionados aos avanços. Caso os custos com avanço forem tão relevantes quanto os custos de atraso, não considerar os tempos ociosos incorre em custos consideráveis para a manufatura.

Apesar da grande relevância de janelas de tempo em conjunto com tempos ociosos, não foi encontrado na literatura nenhum trabalho considerando estas características em ambientes UPMSMDST, objeto de estudo desta tese. Assim, considerando o contexto exposto, neste capítulo é proposto o estudo de minimização da soma total ponderada dos atrasos e avanços (TWET) com janelas de tempo em um ambiente UPMSMDST, considerando a possibilidade de tempos ociosos entre a execução de duas tarefas. Este estudo tem por objetivo responder três questões principais:

- (**RQ1**:) Quais intervenções são necessárias para adaptar um modelo *Mixed Integer Programming* (MIP) e metaheurísticas previamente estudadas para outros critérios de otimização, para resolver a minimização do TWET em ambientes UPMSMDST, considerando a existência de janelas de tempo e tempos ociosos?
- (**RQ2**:) Como o MIP e as metaheurísticas avaliadas se comportam frente a diferentes características deste ambiente?
- (**RQ3**:) O tamanho da janela de tempo influi na redução/incremento dos custos relativos ao avanço e atraso em ambientes UPMSMDST?

Para responder a estas questões, neste capítulo é proposto um MIP para resolução de problemas de pequena dimensão e que permita, consequentemente, a validação dos algoritmos metaheurísticos. Adicionalmente a isto, é proposta uma extensão, para máquinas paralelas não relacionadas, do algoritmo de inserção ótima de tempos ociosos em máquina única proposto por Rosa et al. (2017). Este algoritmo é integrado a cinco metaheurísticas destinadas à resolução de problemas de grande dimensão. Três destas foram previamente propostas na literatura para minimização do TWT: *Iterated Greedy Search* - IGS (Lin et al., 2011), *Artificial Bee Colony* - ABC (Ying e Lin, 2012) e *Genetic Algorithm* - GA (Zeidi e MohammadHosseini, 2015)). A quarta é metaheurística GRASP-ILS-*Pathrelink*, proposta por (Nogueira et al., 2014) para a minimização de atrasos e avanços sem a consideração de janelas de tempo. A quinta e última é a metaheurística híbrida e imuno inspirada VNS-aiNet, proposta no Capítulo 5 desta Tese para a otimização de diversos critérios de otimização. Para que estas metaheurísticas ficassem viáveis computacionalmente, devido ao incremento da ordem de complexidade introduzido pelo algoritmo de inserção de tempos ociosos, propusemos um novo operador de busca local baseado nos operadores de busca local projetados no Capítulo 4. Para avaliar o desempenho destas abordagens, é proposto

um conjunto de instâncias, sendo este deixado público para o desenvolvimento de trabalhos futuros por outros autores. Por fim, sobre o conjunto de instâncias proposto, realizamos dois conjunto de experimentos para verificar: (i) o desempenho do MIP e da metaheurística para a resolução do problema; (ii) se a ampliação das janelas de tempo reduz os custos relativos aos avanços e atrasos.

6.2 Definição do problema e formulação matemática

O problema de sequenciamento estudado neste capítulo consiste em atribuir e sequenciar cada tarefa $k \in N = \{1, \ldots, n\}$ ao conjunto $M = \{1, \ldots, m\}$ de máquinas paralelas não relacionadas. Cada tarefa $k \in N$ consiste em apenas uma operação e está sempre disponível no instante inicial de planejamento. Além disto, toda tarefa k possui uma janela de tempo $[e_k, d_k]$, na qual e_k e d_k são, respectivamente, as datas de avanço e atraso da tarefa k. Idealmente, todas as tarefas devem ser concluídas dentro de sua respectiva janela de tempo. Assim, considerando C_k como o instante de término da tarefa k, o avanço e atraso de uma tarefa k é definido, respectivamente, como $E_k = \max\{0, e_k - C_k\}$ e $T_k = \max\{0, C_k - d_k\}$ (Seção 2.1.3.5). O critério de otimização utilizado neste capítulo é a soma ponderada de atrasos e de avanços ($\alpha = TWET$), definido na Equação 2.11, considerando β_k e α_k, respectivamente, os custos/pesos de avanço e atraso de uma tarefa k.

Diferentemente dos demais capítulos desta tese, neste capítulo o instante de término da tarefa k, C_k considera a possibilidade de tempos ociosos. Assim,

$$C_k = C_j + I_{ijk} + s_{ijk} + p_{ik},$$

sendo p_{ik} o tempo de processamento da tarefa k na máquina i, à qual k está atribuída; C_j é o instante de término da tarefa j que precede k na sequência da máquina i; s_{ijk} é o tempo de preparação para o processamento de k após j; e I_{ijk} o tempo ocioso entre o processamento das tarefas j e k (para maiores detalhes, veja Seção 2.1.4).

De acordo com a notação de três campos proposta por Graham et al. (1979), o problema estudado neste capítulo pode ser sumarizado como $Rm|s_{ijk}, I_{ijk}, e_k \neq d_k \mid \sum_{k=1}^{n} \alpha_k T_k + \beta_k E_k$. A próxima seção apresenta um MIP para o problema descrito.

6.2.1 *Mixed Integer Programming*

O *Mixed-Integer Programming* (MIP) proposto neste capítulo para a resolução do problema objeto de interesse é baseado em MIPs comumente descritos na literatura, destinados à minimização do *makespan* para ambientes UPMS (Guinet, 1991; Rabadi et al., 2006; Vallada e Ruiz, 2011b; Fanjul-Peyro et al., 2019), e pelo MIP proposto por Nogueira et al. (2014) para a minimização de TWET sem janelas de tempo. A Tabela 6.1 introduz a notação adotada para descrever o MIP. No modelo é assumido que, na primeira posição de cada máquina i, existe uma tarefa fictícia, representada por $j = 0$, na qual os tempos de processamento e preparação são iguais a 0. O modelo matemático proposto é dado como:

$$\min \sum_{k=1}^{n} (\alpha_k T_k + \beta_k E_k) \tag{6.1}$$

Sujeito à:

$$\sum_{i=1}^{m}\sum_{\substack{j=0\\j\neq k}}^{n} x_{ijk} = 1 \qquad \forall k \in N \tag{6.2}$$

$$\sum_{i=1}^{m}\sum_{\substack{k=1\\k\neq j}}^{n} x_{ijk} \leq 1 \qquad \forall j \in N \tag{6.3}$$

$$\sum_{k=1}^{n} x_{i0k} \leq 1 \qquad \forall i \in M \tag{6.4}$$

$$\sum_{\substack{h=0\\h\neq j,h\neq k}}^{n} x_{ihj} \geq x_{ijk} \qquad \forall i \in M,\ \forall j,k \in N, j \neq k \tag{6.5}$$

$$C_{ik} + CONST(1 - x_{ijk}) \geq C_{ij} + s_{ijk} + p_{ik} \qquad \begin{array}{r}\forall j \in \{0\} \cup N,\\ \forall k \in N, j \neq k\ \ \forall i \in M\end{array} \tag{6.6}$$

$$C_{i0} = 0 \qquad \forall i \in M \tag{6.7}$$

$$E_k \geq e_k - C_{ik} \qquad \forall i \in M,\ \forall k \in N \tag{6.8}$$

$$T_k \geq C_{ik} - d_k \qquad \forall i \in M,\ \forall k \in N \tag{6.9}$$

$$E_k \geq 0,\quad T_k \geq 0 \qquad \forall k \in N \tag{6.10}$$

$$x_{ijk} \in \{0,1\} \qquad \forall j \in \{0\} \cup N,\ \forall k \in N, j \neq k\ \forall i \in M \tag{6.11}$$

O objetivo de minimização da soma ponderada dos avanços e atrasos é estabelecido na Expressão 6.1. A restrição imposta pela Equação 6.2 estabelece que toda tarefa k tem restritamente um predecessor, sendo executada apenas em uma máquina i. A restrição 6.3 garante que há, no máximo, um sucessor para cada tarefa. Já a restrição 6.4 garante que a tarefa fictícia inicial de cada máquina tenha no máximo uma tarefa sucessora imediata. Por meio das restrições impostas pela Equação 6.5 se uma tarefa j for sequenciada na máquina i precedendo uma tarefa k, existirá necessariamente em tal máquina uma tarefa h que precede a tarefa j, mesmo que h seja uma tarefa fictícia. A restrição 6.6 é utilizada para definir que o instante de término da tarefa k seja superior ao instante de término da tarefa j, acrescido dos tempos de processamento e preparação de k, sendo k a tarefa que sucede imediatamente a tarefa j na sequência da máquina i. No caso de duas tarefas que não ocupam posições adjacentes na sequência de uma dada máquina, ou que estão alocadas em máquinas distintas (casos em que a variável x_{ijk} é nula), as restrições impostas pela restrição 6.6 são automaticamente satisfeitas, graças à constante $CONST$ (positiva e suficientemente grande). A Equação 6.7 define como 0 (zero) o instante de conclusão da tarefa inicial fictícia de cada máquina. Já a restrição representada pelas Expressões 6.8 e 6.9 definem, respectivamente, o atraso e avanço de cada tarefa k, considerando a janela de tempo de cada tarefa definidas pelas constantes e_k e d_k. A restrição 6.10 define que o avanço e atraso de uma tarefa k não podem ser negativos. Finalmente, a restrição representada na Expressão 6.11 define as variáveis de decisão x_{ijk} como binárias.

Tabela 6.1: Notações para definição do MIP.

Notação	Descrição	Tipo
M	Conjunto de máquinas paralelas não relacionadas.	Constante
N	Conjunto de tarefas a serem sequenciadas.	Constante
p_{ik}	Tempo de processamento de uma tarefa k quando atribuída a máquina i	Constante
s_{ijk}	Tempo de preparação de uma tarefa k precedida pela tarefa j na sequência da máquina i	Constante
α_k	Custo por unidade de atraso da tarefa k.	Constante
β_k	Custo por unidade de avanço da tarefa k.	Constante
d_k	Data atraso da tarefa k.	Constante
e_k	Data avanço da tarefa k.	Constante
$CONST$	Uma constante suficiente para garantir a validade do modelo.	Constante
x_{ijk}	Variável de decisão binária, sendo: 1 se a tarefa k é precedida imediatamente pela tarefa j ; na máquina i 0 caso contrario.	Variável de decisão
C_{ik}	Instante de término da tarefa $k \in N$ atribuída a máquina i.	Variável de decisão
C_{i0}	Instante de término da tarefa inicial fictícia da máquina $i \in M$.	Variável de decisão
T_k	Atraso da tarefa k.	Variável de decisão
E_k	Avanço da tarefa k.	Variável de decisão

6.3 Métodos Heurísticos

O foco deste capítulo é avaliar a influência da janela de tempo em soluções de sequenciamento, avaliando o comportamento de cinco metaheurísticas, estabelecidas na literatura, para ambientes UPMSMDST que envolvem objetivos com dadas de entrega na resolução de problemas com janelas de tempo e tempos ociosos. Dentre as metaheurísticas implementadas, apenas a GRASP-ILS-PathRelinking Nogueira et al. (2014) dispõe de um algoritmo para inserção de tempos ociosos, mas sem considerar janelas de tempo. Sendo assim, para adaptarmos as metaheurísticas ao problema, adaptamos, para máquinas paralelas não relacionadas, um algoritmo de inserção de tempos ociosos, proposto por Rosa et al. (2017) para máquina única, que considera janelas de tempo. Este algoritmo é integrado a cada metaheurística em todo ponto que a mesma necessitar da avaliação da função objetivo. Este método de inserção de tempos ociosos acarreta em considerável incremento do custo computacional de avaliação da função objetivo. Assim, para tornar as metaheurísticas computacionalmente viáveis, propusemos um operador de busca local baseado nas estruturas de vizinhança e nas conclusões advindas do Capítulo 4. Esta seção irá detalhar o método de inserção de tempos ociosos e busca local propostos. Não entramos em detalhes na implementação das metaheurísticas, já que estes detalhes podem ser obtidos nos manuscritos nos quais as mesmas foram originalmente propostas.

6.3.1 Inserção de tempos ociosos

Estendemos para máquinas paralelas o algoritmo de inserção de tempos ociosos proposto por Rosa et al. (2017) para máquina única. Em ambientes UPMSMDST, a

inserção de tempos ociosos entre tarefas é um processo independente por máquina. Assim, a adaptação do algoritmo proposto por Rosa et al. (2017) para máquina paralelas não relacionada implica em separar o sequenciamento em m subsequenciamentos. Assim, para o sequenciamento de cada máquina, aplica-se o algoritmo proposto por Rosa et al. (2017) considerando as configurações características de máquina. Esta adaptação do método proposto por Rosa et al. (2017) é descrita no Algoritmo 22. A Tabela 6.2 apresenta um glossário das notações utilizadas no Algoritmo 22. Estas notações são complementares às definições já apresentadas para descrição do modelo, mostradas na Seção 6.2.

Tabela 6.2: Notações complementares a Tabela 6.1 para definição do Algoritmo 22.

Notação	Descrição
s	Solução previamente sequenciada que deseja-se inserir os tempos ociosos.
S_{ik}	Instante inicial de execução da tarefa k na máquina i.
N_i	Lista, devidamente ordenada pelo sequenciamento, das tarefas sequenciadas na máquina i.
B	Bloco de tarefas sem tempos ociosos entre elas presentes em uma máquina i.
$MC(B)$	Função que define o beneficio/penalização de mover o bloco B na dimensão tempo.
k'	Um tarefa pertencente ao bloco B.
z	Primeira tarefa que sucede o bloco B na sequência da máquina i.
φ	Tamanho do passo no tempo realizado pelo bloco B.
$m_1(B, i, z)$	Função que retorna a diferença entre o instante de término do bloco B e o instante inicial (S_{iz}) da tarefa z
$m_2(B, i)$	Função que retorna a menor quantidade de tempo necessária para alguma tarefa k' do bloco B incorrer em atraso, dado que k' tenha o instante de término dentro da janela de tempo.
$m_3(B, i)$	Função que retorna o menor avanço dentre as tarefas que compõem o bloco B.
$next(i, z)$	Função que retorna a tarefa sucessora de z no sequenciamento da máquina i. Retorna -1 caso z seja a última tarefa na sequência da máquina i.

O Algoritmo 22 tem, como entrada, uma solução s representada como m sequências de tarefas. Este algoritmo primeiramente percorre, de forma independente, a sequência de tarefas atribuídas a cada máquina $i \in M$ que compõe s. Cada tarefa k, atribuída a uma máquina i, tem seu instante inicial de processamento S_{ik} e instante de término C_{ik} avaliados sem a inserção de tempos ociosos, ou seja, $S_{ik} \leftarrow C_{ij}$ e $C_{ik} \leftarrow C_{ij} + s_{ijk} + p_{ik}$, sendo j a tarefa que precede k na máquina i (Linhas 6 à 10). Em seguida, para cada máquina i, percorre-se, em ordem reversa, as tarefas atribuídas e sequenciadas nesta máquina (Linha 12). Assim, para cada intervalo de processamento de duas tarefas j e k, é verificado se a inserção de tempos ociosos acarretará em benefício para a redução do TWET. Assim, primeiramente é atribuído a k a última tarefa sequenciada na máquina i e verificado se a mesma apresenta avanço; caso não apresente, não será necessária a inserção de tempo ocioso entre k e j; caso contrário, é criado um bloco de tarefas B, inicialmente apenas com a tarefa k. Deve-se definir que todo bloco B sempre será composto por tarefas que não apresente tempos ociosos entre as mesmas. Um vez definido o primeiro bloco B, é verificado se a redução no custo de avanço pelo deslocamento à direita das tarefas presentes no bloco B compensa o incremento no atraso das mesmas tarefas. Para isto, a variável

de decisão $MC(B)$ é calculada como:

$$MC(B) = \sum_{k' \in B : C_{ik'} \geq d_{k'}} \alpha_{k'} - \sum_{k' \in B : C_{ik'} < e_{k'}} \beta_{k'}. \tag{6.12}$$

Dado $MC(B)$, segundo (Rosa et al., 2017), duas situações podem ocorrer:

(i) $MC(B) \geq 0$: A redução dos custos de avanço, por um possível deslocamento à direita do bloco de tarefas B, não compensará o atraso acarretado pelo mesmo deslocamento;

(ii) $MC(B) < 0$: Neste caso, o deslocamento do início de processamento do bloco B, à direita, irá acarretar em decremento da soma ponderado dos avanços e atrasos da máquina i; assim, um intervalo de tempo ocioso deve ser inserido entre o bloco B e a tarefa j.

Caso $MC(B) \geq 0$, não há nada a ser feito no bloco B, mantendo inalteradas as configurações de $C_{ik'}$ e $S_{ik'}$ para todas as tarefas $k' \in B$. Caso contrário, enquanto $MC(B) < 0$, o algoritmo repete a sequência:

(i) Calcula-se o tamanho do passo φ que as k' tarefas presentes em B irão caminhar à direita, sendo:

$$\varphi = \min\{m_1(B, i, z), m_2(B, i), m_3(B, i)\}, \tag{6.13}$$

para:

$$\begin{aligned} m_1(B, i, z) &= \max_{k' \in B}\{C_{ik'}\} - S_{iz} \\ m_2(B, i) &= \min_{k' \in B : e_{k'} \leq C_{ik'} < d_{k'}}\{d_{k'} - C_{ik'}\} \\ m_3(B, i) &= \min_{k' \in B : C_{ik'} < e_{k'}}\{e_{k'} - C_{ik'}.\} \end{aligned} \tag{6.14}$$

(ii) Após o cálculo de φ, adiciona-se este aos instantes de início $S_{ik'}$ e término $C_{ik'}$ das tarefas $k' \in B$ (Linhas 17 à 20);

(iii) Em seguida, é verificado se o maior instante de término das tarefas em B é igual ao instante inicial de processamento da primeira tarefa z que sucede B na sequência da máquina i. Caso a condição seja aceita, incorpora-se z ao bloco B (Linhas 21 à 24);

(iv) Por fim, recalcula-se $MC(B)$ e retorna-se ao passo inicial da malha de repetição para verificar as condições $MC(B) \geq 0$ e $MC(B) < 0$ (Linha 15).

Quando a condição de parada da repetição acima for atingida, ou seja, $MC(B) \geq 0$, cria-se um novo bloco B, com a tarefa que precede k na sequência da máquina i (Linha 14). Inicia-se, assim, um novo processo de inserção de tempos ociosos.

Na primeira iteração, apenas a última tarefa sequenciada na máquina i estará em B; na segunda iteração, o novo bloco B deve ser criado com a penúltima tarefa sequenciada na máquina i e o procedimento de deslocamento à direita (inserção do tempos ociosos) das tarefas presentes no bloco B se repete; na terceira iteração, o

bloco inicia com a tarefa que precede a penúltima tarefa escalonada na máquina i, e assim por diante. O algoritmo termina após todas as tarefas presentes na sequência da máquinas i tenham sido exploradas. Deve-se observar que todo bloco inicialmente apresenta apenas uma tarefa. Este bloco pode incorporar novas tarefas, à medida que o deslocamento à direita do bloco acarretar no encontro do instante final do bloco B com o instante inicial da tarefa z que suceder este.

A ordem de complexidade do Algoritmo 22 pode ser derivada de Rosa et al. (2017). Considerando que o algoritmo para inserção de tempos ociosos em máquina única apresenta ordem de complexidade de $O(n^2)$, sendo n o número de tarefas no sequenciamento (Rosa et al., 2017), ao estendemos para máquinas paralelas, o número de tarefas esperado por máquina passa a ser n/m; assim, a ordem de complexidade é derivada em $O((n/m)^2)$.

Algoritmo 22 Inserção de tempos ociosos

```
1: procedure INSERÇÃO_TEMPOS_OCIOSOS(s)
2:  with s do
3:     for i = 1, ..., m do
4:         C_i0 ← 0;
5:         j ← 0;
6:         for k ∈ N_i do
7:             S_ik ← C_ij;
8:             C_ik ← C_ij + s_ijk + p_ik;
9:             j ← k;
10:        end for
11:        z ← −1;
12:        for k ← N_i[|N_i|], N_i[|N_i| − 1], N_i[|N_i| − 2], ..., 1 do
13:            if C_ik < e_k then
14:                B ← {k}
15:                while (MC(B) < 0) do
16:                    φ ← min{m_1(B, i, z), m_2(B, i), m_3(B, i)};
17:                    for k' ∈ B do
18:                        C_ik' ← C_ik' + φ;
19:                        S_ik' ← S_ik' + φ;
20:                    end for
21:                    if z ≠ −1 and max_{k'∈B}{C_ik'} = S_iz then
22:                        B ← B ∪ {z};
23:                        z ← next(i, z);
24:                    end if
25:                end while
26:            end if
27:            z ← k;
28:        end for
29:    end for
30: endwith
31: end procedure
```

6.3.2 Busca Local

Devido ao incremento no custo computacional decorrente do procedimento de inserção de tempos ociosos, os operadores de busca local "nativos" das cinco metaheurísticas avaliadas neste capítulo tornam as mesmas impraticáveis e/ou ineficientes. Assim, neste capítulo, optamos por utilizar um mesmo operador de busca local para todas as metaheurísticas avaliadas. Como busca local, utilizamos uma estratégia baseada na Seção 4.3 – *Variable Neighborhood Descent* (VND) – apresentada no Capítulo 4. Nesta seção, é mostrado que as seis estruturas de vizinhança propostas aumentaram o desempenho das metaheurísticas IGS, ABC e GA. Além disto, são apresentadas três estruturas de vizinhança que dominaram a contribuição para convergência do problema e as três demais são utilizadas para ajuste fino das soluções. Considerando que o algoritmo proposto para cálculo da função objetivo incrementa a ordem de complexidade em relação à avaliação do atraso total de $O(n)$ para $O((n/m)^2)$, é necessária a redução do algoritmo de busca local proposto no Capítulo 4.

6.3.2.1 Estruturas de vizinhança

Para o problema com janela de tempo, propusemos utilizar três estruturas de vizinhança. A estrutura de vizinhança NE.IGS (Seção 4.3.1.5) é mantida em sua forma original, já que a mesma se mostrou muito eficiente no processo de convergência, e, entre as seis estruturas de vizinhança propostas na Seção 4.3, a NE.IGS apresenta o menor custo computacional ($O(n)$ avaliações de função objetivo). Já as estruturas NE.EI (Seção 4.3.1.3) e NE.ES (Seção 4.3.1.4) são reformuladas, incorporando, respectivamente, características das estruturas de ajuste fino NE.II (Seção 4.3.1.1) e NE.IS (Seção 4.3.1.2). As novas estruturas de vizinhança geradas são denominadas aqui como *Insertion Neighborhood* - $NE.I$ e *Swap Neighborhood* - $NE.S$ e são apresentadas a seguir.

6.3.2.2 *Iterated Greedy Search Neighborhood*

A única diferença entre a NE.IGS usada neste capítulo e a apresentada no Capítulo 4 - Seção 4.3.1.5) está na definição do parâmetro rj. No Capítulo 4 é utilizado o parâmetro rj como um valor fixo. Assim, o número de tarefas removidas a cada iteração da estrutura de vizinhança não varia de acordo com o tamanho do espaço amostral. Experimentos empíricos realizados neste capítulo indicaram que, quando rj é diretamente proporcional ao número de tarefas, a utilização da estrutura acarretava em melhor desempenho da metaheurística. Assim, definimos $rj = \max(1, igs_\alpha * n)$, sendo $igs_\alpha \in [0, 1]$ um parâmetro a ser calibrado pela metaheurística que utiliza esta estrutura de vizinhança e n o número de tarefas presentes no problema a ser solucionado.

6.3.2.3 *Insertion Neighborhood* - $NE.I$

A estrutura de vizinhança NE.I é derivada da composição das estrutura de vizinhança NE.II (Seção 4.3.1.1) e NE.EI (Seção 4.3.1.3). Nesta estrutura, uma tarefa k atribuída a máquina i é retirada da máquina e, posteriormente, inserida em uma

da máquina w. Como incorporamos NE.II em NE.EI, permitimos tanto movimentos quando $i \neq w$ (e.g. Figura 2.4), quanto $i = w$ (e.g. Figura 2.2). Esta estrutura é explorada de forma reduzida, conforme descrito no Algoritmo 23, sendo primeiramente selecionada aleatoriamente uma máquina i de M_t, sendo $M_t = \{i \in M \ : \ \forall \ E_i + T_i > 0\}$, em que E_i e T_i são, respectivamente, o avanço e atraso total presentes na máquina i; posteriormente, a máquina w é selecionada aleatoriamente com probabilidade uniforme na lista L, sendo $L = \{w \in M\}$; então, é aplicado um método de descida *Best Improvement* por inserção (Algoritmo 25, Apêndice A.4) nas máquinas i e w, retirando-se tarefas da máquina i e inserindo-se nas posições da máquina w. Caso a busca resultar em algum movimento de melhora, a solução é retornada; caso contrário, a busca é reiniciada, selecionando outra máquina w em L. A busca termina ao encontrar algum movimento de melhora em alguma máquina w selecionada de L, ou caso todas as máquinas pertencentes a L tenham sido exploradas.

Algoritmo 23 Algoritmo que descreve a estrutura de vizinhança NE.I.

```
1:  function NE.I(s)
2:      s* ← s;
3:      s' ← NULL;
4:      improvement ← FALSE;
5:      M_t ← {∀ i ∈ M : E_i + T_i > 0};
6:      i ← rand(M_t);
7:      L ← M;
8:      while (f(s*) == f(s) and |L| > 0) do
9:          w ← rand(L);
10:         s' ← descida_insercao(s, TWET, i, w, B.I);
11:         if f(s') < f(s*) then
12:             s* ← s';
13:         end if
14:         remover(L, w);
15:     end while
16:     return s*;
17: end function
```

6.3.2.4 *Swap Neighborhood* - $NE.S$

A estrutura $NE.S$ é gerada a partir da incorporação da estrutura $NE.IS$ (Seção 4.3.1.2) em $NE.ES$ (Seção 4.3.1.4). Assim, sejam as tarefas k e j que ocupam respectivamente as posições k' e j' das máquinas i e w. Um movimento em $NE.S$ consiste na tarefa k ser processada na posição j' da máquina w; já a tarefa j passa a ser processada na posição k' da máquina i. Como esta estrutura é a composição de $NE.IS$ e $NE.ES$, são aceitos tanto movimentos em que $i \neq w$ quanto $i = w$. $NE.S$ é processada de forma reduzida, considerando os mesmos passos descritos no Algoritmo 23, substituindo na linha 10, considerando a descida por troca através do método *Best Improvement* descrito no Algoritmo 28 presente no Apêndice A.4.

6.3.2.5 Métodos, ordem e forma das estruturas de vizinhança.

Utilizamos, para o projeto do operador de busca local usado pelas metaheurística, o método VND na sua forma *Basic*-VND (Seção 3.3.4.1). Esta escolha é baseada nos resultados obtidos no Capítulo 4, no qual é mostrado que esta versão do VND apresentou desempenho superior às demais variações do VND para o espaço de busca UPMSMDST com o objetivo de minimizar o TWT. A escolha pela forma de busca por *Best Improvement* pelas estruturas de vizinhança NE.I e NE.S segue o mesmo critério. Esta forma de busca se mostrou preponderante para as estruturas de vizinhança das quais NE.I e NE.S são derivadas. Já para a ordem de exploração das estruturas de vizinhança, optamos pela ordem definida pela cardinalidade, sendo esta definida como $NE.O_{card} = \{NE.IGS, NE.I, NE.S\}$. No Capítulo 4 não foi encontrado um padrão definitivo para a ordem de exploração das estruturas. Tendo em vista essa observação, realizamos experimentos preliminares que indicaram $NE.O_{card}$ como uma melhor opção para o problema estudado.

6.4 Metodologia Experimental

Como descrito na Introdução, este Capítulo visa responder às questões (RQ1), (RQ2) e (RQ3). A questão (RQ1) é tratada nas seções anteriores. Nesta seção é proposta a metodologia experimental para analisar as questões (RQ2) e (RQ3), sendo organizada como: a Seção 6.4.1 sumariza questões gerais a respeito das implementações computacionais utilizadas nos experimentos; a Seção 6.4.2 descreve os critérios utilizados na geração do conjunto de instâncias; a Seção 6.4.3 detalha os critérios dos experimentos computacionais realizados; a Seção 6.4.4 apresenta a calibração de parâmetros utilizada pelas metaheurísticas; por fim, as Seções 6.4.5 e 6.4.6 detalham, respectivamente, os experimentos conduzidos para responder às questões (RQ2) e (RQ3).

6.4.1 Implementação computacional

O MIP descrito na Seção 6.2.1 é implementado no *solver* Gurobi versão 8.1.1 e descrito em linguagem Python 2.7. As metaheurísticas ABC (Ying e Lin, 2012), GA (Zeidi e MohammadHosseini, 2015), GRASP (Nogueira et al., 2014), IGS (Lin et al., 2011) e VNS-aiNet (Capítulo 5) são implementadas em linguagem Java, usando JDK 1.8, seguindo rigorosamente as *guidelines* descritas nos respectivos manuscritos nos quais as abordagens são propostas. São realizados três adequações para adaptar as abordagens metaheurísticas ao problema estudado:

(i) as soluções iniciais de todas as abordagens são geradas de forma aleatória (Apêndice A.2);

(ii) em todas as metaheurísticas avaliadas neste Capítulo, existe explicitamente um operador de busca local. Este operador é substituído pela abordagem VND descrita na Seção 6.3.2;

(iii) toda avaliação de função objetivo é sempre precedida do algoritmo de definição dos tempos ociosos descrito na Seção 6.3.1.

Para seguir as boas práticas de pesquisa operacional (Kendall et al., 2016), disponibilizamos em `https://github.com/rodsxe/timewindow-upms` um repositório público contendo os *scripts*, código fonte, conjunto de instâncias e resultados dos experimentos computacionais realizados neste Capítulo.

6.4.2 Instâncias de *benchmark*

As instâncias geradas para os experimentos computacionais deste Capítulo são baseadas nas metodologias utilizadas em Chen e Wu (2006); Chen (2009), para máquinas paralelas não relacionadas com datas de entrega, e Potts e Wassenhove (1982); Yano e Kim (1991); Rosa et al. (2017), para a geração de janelas de tempo.

O conjunto de instâncias é subdivido em dois grupos, chamados aqui como *small* e *large*. O conjunto *small* apresenta um conjunto com 12 tarefas a serem sequenciadas em 3 máquinas. Já o conjunto *large* apresenta três níveis de tarefas e máquinas, respectivamente nas formas $N = \{60, 80, 100\}$ e $M = \{4, 6, 8\}$. Os tempos de preparação em ambos os grupos são gerados conforme proposto por Chen (2009), ou seja, de forma aleatória com probabilidade uniforme no intervalo $U[10, 100]$. As tarefas são distribuídas aleatoriamente em $\lfloor n/7 \rfloor + 1$ famílias. Conforme apresentado em Chen e Wu (2006); Chen (2009), o tempo de processamento p_{ik} de cada tarefa k, alocada na máquina i, é dado por $1/f_{iq} \times U[10, 40]$, sendo $f_{iq} = 1/U[5, 15]$ a velocidade de processamento das tarefas pertencentes à família q na máquina i.

Já as janelas de tempo são geradas conforme a metodologia proposta por Yano e Kim (1991) para máquina única sem tempos de preparação. Esta metodologia define um *centroid* CE_k de uma tarefa k como o ponto central da janela de tempo $[e_k, d_k]$. A definição do ponto no espaço temporal ocupada pelo *centroid* é baseada na metodologia proposta por Potts e Wassenhove (1982) para a definição de datas de entrega. Para isto, Yano e Kim (1991) utiliza dois fatores, sendo τ o fator de proximidade entre os *centroids* de diferentes tarefas e R o fator de proximidade entre os *centroids* e o instante inicial de planejamento. Assim, CE_k de cada tarefa k é gerado de forma aleatória, com probabilidade uniforme dada por $U[\bar{C_{max}} \times |1 - \tau - R/2|, \bar{C_{max}} \times |1 - \tau + R/2|]$, sendo $\bar{C_{max}}$ o valor esperado do *makespan* para o conjunto de instâncias. A definição do $\bar{C_{max}}$ em ambientes UPMS com tempos de preparação pode ser extraída de Chen (2009), que utiliza o mesmo conceito para a geração de datas de entrega. Assim, $\bar{C_{max}}$ pode ser definido como:

$$\bar{C_{\max}} = (\sum_{k=1}^{n} \sum_{i=1}^{m} (p_{ik} + (\sum_{j=1}^{n} s_{ijk})/n)/m)/m.$$

Uma vez definidos os *centroids* CE_k de cada tarefa k, o tamanho da janela de tempo ($size_t$) é definido por Yano e Kim (1991) como um valor aleatório com probabilidade uniforme dentro do intervalo $U[0, \bar{C_{max}}/n]$. A janela de tempo $[e_k, t_k]$ de uma tarefa k é definida, portanto, como $e_k = CE_k - size_t/2$ e $t_k = CE_k + size_t/2$.

Como pode-se observar, o esquema de geração de instâncias utilizado neste Capítulo apresenta quatro fatores controláveis: (i) o conjunto N de tarefas; (ii) o conjunto M de máquinas; (iii) o fator τ de proximidade entre as janelas de tempo; e (iv) o fator R de proximidade das janelas de tempo com o instante inicial de planejamento. Os níveis dos fatores N e M foram definidos anteriormente, conforme a definição dos

conjuntos *small* e *large*. Para os fatores τ e R, avaliamos, respectivamente, 2 e 3 níveis, sendo $\tau = \{0.4, 0.8\}$ e $R = \{0.4, 0.8, 1.2\}$. Para cada combinação destes fatores $N \times M \times \tau \times R$, foram geradas 20 instâncias. Desse modo, foram geradas 120 e 1080 instâncias, respectivamente, para os conjuntos *small* e *large*. A Tabela 6.3 apresenta um sumário com todos os fatores, controláveis e calculados, e suas definições.

Tabela 6.3: Lista de fatores para geração de instâncias.

Fator	Definição
N	12*, 60, 80, 100
M	3*, 4, 6, 8
τ	0.4, 0.8
R	0.4, 0.8, 1.2
Número de famílias de tarefas (q)	$\lfloor n/7 \rfloor + 1$
Fator de velocidade de processamento por família q e máquina i (f_{iq})	$1/U[5, 15]$
Tempo de processamento da tarefa k na máquina i (p_{ik})	$1/f_{iq} \times U[10, 40]$
Tempos de preparação da tarefa k (s_{ijk})	$U[10, 100]$
Makespan esperado ($C^{-}_{\max}$)	$(\sum_{k=1}^{n} \sum_{i=1}^{m} (p_{ik} + (\sum_{j=1}^{n} s_{ijk})/n)/m)/m$
Centroid da tarefa k (CE_k)	$U[C^{-}_{max}(\lvert 1 - \tau - R/2 \rvert), C^{-}_{max}(\lvert 1 - \tau + R/2 \rvert)]$
Tamanho da janela de tempo ($size_t$)	$U[0, C^{-}_{max}/n]$
Data de atraso de um tarefa k (e_k)	$e_k = CE_k - size_t/2$
Data de atraso de um tarefa k, (d_k)	$d_k = CE_k j + size_t/2$

*Nível do fator utilizado apenas no conjuntos de instâncias *small*.

6.4.3 Critérios dos experimentos computacionais

Os experimentos computacionais com o MIP foram realizados sobre o conjunto de instâncias *small*. Nestes experimentos, as instâncias são solucionadas até atingirem a otimalidade ou ultrapassassem o tempo de execução limite, definido como 10800 segundos (3 horas). Já para as metaheurísticas, definimos o critério de parada a partir da definição proposta em de França Filho (2007), que determina que o número de soluções distintas em ambientes de UPMSMDST é dado por $(n+m-1)!/(m-1)!$. Assim, o critério de parada utilizado aqui é estabelecido em termos do número de avaliações de função objetivo (NFE):

$$NFE_{ins} = \frac{n + m - 1}{2(m - 1)} \times 10^7, \tag{6.15}$$

sendo n e m o número de máquinas definidos na instância ins. Utilizamos, ainda, um subcritério de parada, dado por 50 gerações/iterações sem melhora. Este é adotado tendo em vista que algumas abordagens podem atingir a convergência antes do número total definido de avaliações da função objetivo. Para minimizar a influência do processo de busca estocástico das metaheurísticas, a resposta de cada metaheurística para cada instância é definida como a média de 5 execuções independentes da mesma. A factibilidade das soluções geradas pelas metaheurísticas foram verificadas utilizando o MIP. Usamos cada solução gerada pelas metaheurísticas como solução

inicial do MIP, verificamos se as restrições não são violadas e se o valor da função objetivo calculado usando o algoritmo de inserção de tempos ociosos e o calculado pelo MIP são compatíveis.

6.4.4 Calibração de parâmetros

Como já mencionado nos capítulos anteriores, a definição dos parâmetros de metaheurísticas é um dos fatores de grande influência na qualidade dos resultados gerados por estas. Com o intuito de evitar que uma calibração ineficiente afete os resultados dos algoritmos, as metaheurísticas ABC, GA, GRASP, IGS e VNS-aiNet tiveram seus parâmetros calibrados pelo *framework iRace* (Ibáñez et al., 2016). A calibração pelo *IRace* é utilizada para evitar possíveis viés cometidos pelos autores, já que define de maneira autônoma os parâmetros. Um conjunto de instâncias de calibração foi gerado, utilizando a metodologia descrita na Seção 6.4.2 e considerando duas instâncias para cada combinação dos fatores $n \times m \times \tau \times R$. Para cada algoritmo, é considerado um *budget* de 10.000 simulações, sendo que cada simulação considera o número máximo de NFE definido pela Equação 6.15. Os níveis que cada parâmetro pode assumir são definidos considerando os valores originais dos parâmetros apresentados no manuscrito em que cada metaheurística é proposta, sendo este considerando o nível central do parâmetro, a partir do qual definimos limites à esquerda e à direita igualmente proporcionais. Para todas as metaheurísticas, a faixa que pode assumir o parâmetro igs_α, utilizado na busca local pela estrutura de vizinhança proposta na Seção 6.3.2.2, é definido entre os limites $[0.01, 0.12]$. Os parâmetros, níveis avaliados e o valor do parâmetro escolhido pelo *IRace* para cada metaheurística são apresentados nas Tabelas 6.4, 6.5, 6.6, 6.7 e 6.8 para as abordagens ABC, GA, GRASP, IGS e VNS-aiNet, respectivamente.

Tabela 6.4: Parâmetros metaheurística ABC.

Parâmetro	Nível	Selecionado *Irace*
SN	(5, 6, 7, 8, 9)	**9**
$limit$	(100, 150, 200, 250)	**100**
$threshold$	(2, 3, 4, 5, 6, 7)	**3**
α_{max}	(3, 4, 5, 6)	**4**
igs_α	[0.01, 0.12]	**0.11**

Tabela 6.5: Parâmetros metaheurística GA.

Parâmetro	Nível	Selecionado *Irace*
Pop_{size}	(60, 80, 100)	**60**
P_c	[0.4, 0.8]	**0.51**
P_m	[0.08, 0.2]	**0.123**
P_s	[0.15, 0.4]	**0.398**
igs_α	[0.01, 0.12]	**0.083**

Tabela 6.6: Parâmetros metaheurística GRASP.

Parâmetro	Nível	Selecionado *Irace*
e_{size}	(5, 10, 15, 20, 25, 30)	**30**
α	[0.05, 0.3]	**0.295**
I_{ILS}	(50, 100, 150, 200, 250, 300)	**300**
d	[0.01, 0.12] $\times \frac{n}{m}$	**0.1**
igs_{α}	[0.01, 0.12]	**0.106**

Tabela 6.7: Parâmetro metaheurística IGS.

Parâmetro	Nível	Selecionado *Irace*
β	[0.5, 1.0]	**0.541**
D	[0.01, 0.12] $\times \frac{n}{m}$	**0.116**
igs_{α}	[0.01, 0.12]	**0.109**

6.4.5 Comparação entre metaheurísticas

A verificação da eficiência das metaheurísticas é realizada através da aplicação destas ao conjunto de instâncias definido na Seção 6.4.2, considerando as condições computacionais definidas na Seção 6.4.3. Os resultados obtidos são sumarizados em tabelas e gráficos presentes na Seção 6.5.1. As tabelas apresentam os resultados agrupados por grupos de instâncias $n \times m \times \tau \times R$. Para cada grupo é calculado o *Relative Percentage Deviation* médio ($\overline{RPD}$), dado por:

$$\overline{RPD} = \frac{1}{N_{gi} \times 5} \sum_{i=1}^{N_{gi}} \sum_{j=1}^{5} \frac{f(i,j) - best_{f(i)}}{best_{f(i)}}, \tag{6.16}$$

sendo N_{gi} o número de instâncias do grupo $n \times m \times \tau \times R$; 5 é o número de execuções independentes para cada instância i; $f(i,j)$ representa o TWET encontrado para a instância i e execução independente j; para o conjunto *small* e *large*, $best_{f(i)}$ representa, respectivamente, a solução retornada pelo MIP e a melhor solução encontrada depois de todos os experimentos sobre a instância i.

Para as instâncias do grupo *small*, os resultados são comparados diretamente pelo $\overline{RPD}$, já que a maior parte das instâncias apresenta um valor ótimo gerado pelo *solver* para o modelo *Mixed-Integer Programming* (MIP) proposto. Já para o grupo de instâncias *large*, além da comparação pelo $\overline{RPD}$, é verificado se os resultados obtidos por uma metaheurística apresentam diferença estatística significante em relação as demais através de análise estatística pelo Teste Pareado de Wilcoxon (Wilcoxon,

Tabela 6.8: Parâmetros metaheurística VNS-aiNet.

Parâmetro	Nível	Selecionado *Irace*
nAB	(5, 10, 15, 20, 25, 30)	**30**
nC_{max}	(4, 6, 8, 10)	**6**
β	(1, 2, 3, 4, 5, 6)	**4**
α	[0.3, 0.7]	**0.635**
σ_s	[0,0001, 0,2]	**0.159**
mat_+	[0.00, 0.06]	**0.019**
igs_{α}	[0.01, 0.12]	**0.076**

1945). O teste estatístico é realizado considerando as hipóteses $H_0 : md_1 - md_2 = 0$ e $H_1 : md_1 - md_2 < 0$, sendo md_1 a mediana obtida pela metaheurística que obteve o menor $\overline{RPD}$ e md_2 a mediana obtida pela metaheurística a ser comparada. Os testes são realizados por grupos $n \times m \times \tau \times R$ e apenas o *p-valor* do teste é apresentado. Todos os testes consideram um intervalo de confiança de 95% ($\alpha = 0.05$). Todos os testes são conduzidos utilizando a versão 3.6.3 do *software* `R`.

Por fim, é realizada uma análise de convergência das abordagens. A análise de convergência é proposta para analisar o comportamento do processo de busca de cada metaheurísticas com o incremento do número de avaliações de função objetivo, sendo realizada em função dos fatores $n \times m$, uma vez que estes definem o tamanho do espaço amostral (de França Filho, 2007), sendo utilizados para definir o número máximo de NFE para cada experimento (Seção 6.4.3). Concomitantemente a esta análise, é apresentado uma análise por gráficos de *boxplot*, mostrando a variabilidade dos resultados na última iteração do algoritmo. O gráfico *boxplot* é utilizado em conjunto com as análises estatísticas descritas acima para avaliar os resultados finais das metaheurísticas.

As análises de convergência são realizadas através do *fitness* normalizado médio, dado por:

$$Avg.Norm.Fitness = \frac{1}{5 \times r} \sum_{i=1}^{r} \sum_{j=1}^{5} \frac{f(i,j) - best_{f(i)}}{worst_{f(i)} - best_{f(i)}}, \tag{6.17}$$

sendo r o número de instâncias presentes no subconjunto $n \times m$; 5 é o número de execuções independentes para cada instância i; $f(i,j)$ representa o TWET encontrado para a instância i e execução independente j; $best_{f(i)}$ e $worst_{f(i)}$ representam, respectivamente, o menor e maior TWET encontrado para a instância i depois de todos os experimentos realizados.

6.4.6 Influência do tamanho das janelas de tempo no TWET

Para avaliar a hipótese que as janelas de tempo influem para reduzirem os atrasos e avanços, é proposto fazer uma análise da influência do tamanho da janela de tempo no TWET. Para isto, definiu-se um fator de reescala da janela de tempo como $tw_{ratio} = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8\}$. As instâncias de *benchmark* geradas na Seção 6.4.2 são divididas aleatoriamente em $|tw_{ratio}|$ grupos, respeitando as proporções de cada fator $n \times m \times \tau \times R$. Para cada subgrupo gerado, reescalou-se a janela de tempo da instância pelo nível do fator correspondente ao grupo. Assim, quando o nível é igual 1.0, a instância não tem a sua janela de tempo reescalada; já para o nível igual a 0.0, indica que não existe janela de tempo, sendo uma data de entrega fixa definida pelo centro da janela de tempo CE_k.

Dado este "novo conjunto de instâncias", considerando as janela de tempo reescaladas, e que md_1 e md_2 são, respectivamente, as medianas de um determinado grupo de instâncias originais e do conjunto de janelas de tempo reescaladas, é proposto realizar um experimento pareado para analisar três hipóteses:

(i) Para $tw_{ratio} = 1.0$, uma vez que o tamanho da janela não é alterado, não há diferença estatística significante entre os resultados obtidos pelo conjunto original e as instâncias reescalonadas. Ou seja, caso $H_0 : m_1 - m_2 = 0$ e $H_1 = md_1 - md_2 \neq 0$, H_0 não será rejeitada.

(ii) Já para $tw_{ratio} < 1.0$, à medida que se reduz a janela de tempo, se aumenta o TWET. Ou seja, caso $H_0 : md_1 - md_2 = 0$ e $H_1 : md_1 - md_2 < 0$, H_0 será rejeitada.

(iii) Já para $tw_{ratio} > 1.0$, à medida que se aumenta a janela de tempo, automaticamente se reduz o TWET. Ou seja, caso $H_0 : md_1 - md_2 = 0$ e $H_1 : md_1 - md_2 > 0$, H_0 será rejeitada.

Para avaliar estas hipóteses, é utilizado o Teste de Wilcoxon pareado por grupo de instâncias $\tau \times R$, já que estes fatores definem o comportamento das janelas de tempo. Os resultados deste teste são apresentados na Seção 6.5.2, sendo mostrado apenas o *p-valor* do respectivo teste. Para todos os testes são considerados o nível de confiança de 95% ($\alpha = 0.05$). Para verificar as três hipóteses levantadas e entender o comportamento da janela de tempo sobre a resposta dos algoritmos, é sugerido verificar se os dados se adequam a um modelo de regressão linear, considerando o erro quadrático médio para ajuste dos coeficientes. As análises de regressão são realizadas por grupos $\tau \times R$ para os algoritmos VNS-aiNet, GA e IGS. A análise de regressão avalia como o *ratio WET* esperado por tarefa se comporta com a variação da janela de tempo (i.e., *ratio WET* é diferença entre a resposta para instância original e a resposta para a instância com janela de tempo reescalada).

6.5 Resultados

Nesta seção são apresentados e discutidos os resultados computacionais dos experimentos sugeridos na Seção 6.4. Na Seção 6.5.1 são apresentados os resultados que ajudam a responder a questão (RQ2), sendo mostrado o comportamento das metaheurísticas nos ambientes UPMSMDST. O comportamento das metaheurísticas, de acordo com as características dos problemas, são discutidos, tentando descrever quais cenários cada uma melhor se adequa. Já a análise do tamanho da janela de tempo no TWET esperado, conforme questão (RQ3), é discutida na Seção 6.5.2.

6.5.1 Análise do MIP e Metaheurísticas

O primeiro conjunto de experimentos tem o objetivo de comparar entre si as soluções geradas pelas metaheurísticas e o MIP. Os resultados do conjunto *small* são apresentados nas Tabelas 6.9 e 6.10. Primeiramente, analisando o comportamento do MIP, observa-se que a distribuição da janela de tempo no horizonte de planejamento (fator R) influi diretamente na convergência do modelo. Quando os *centroids* das janelas de tempo se encontram mais próximos do instante inicial de planejamento ($R = 0.40$), o MIP apresenta dificuldade de encontrar os valores ótimos para o modelo. Isto pode ser observado, já que em 8 instâncias deste cenário, o MIP não consegue convergir até os resultado ótimo dentro do intervalo de 10800 segundos. Já, ao observar os resultados alcançados pelas metaheurísticas, verifica-se que estas encontraram valores de função objetivo inferiores ao MIP em algumas das instâncias em que o MIP não convergiu para as soluções ótimas, indicando que o MIP não convergiu nestes cenários até o resultado ótimo. Vale destacar que as soluções alcançadas pelas

metaheurísticas foram validados pelo MIP, verificando tanto se as restrições são válidas, quanto se o resultado calculado pela função objetivo é válido (ver Seção 6.4.3). Ainda analisando o MIP e o fator R, observa-se que, quando a janela de tempo se afasta do horizonte inicial de planejamento ($R = 1.20$), o MIP converge em todas as instâncias até o valor ótimo, reduzindo também o tempo computacional necessário para convergência. Já quando analisamos a distribuição entre as janelas de tempo ao longo do horizonte de planejamento (fator τ), observa-se que, quando as janelas de tempo são mais distribuídos pelo horizonte de planejamento ($\tau = 0.40$), ocorre uma redução nos valores de TWET, quando comparado às instâncias nas quais as janelas de tempo das tarefas se sobrepõem com maior frequência ($\tau = 0.80$).

Analisando as metaheurísticas sobre o conjunto *small*, observa-se que estas convergem em tempo computacional consideravelmente inferiores comparados ao MIP (todas abaixo de 1 segundo). Além disto, observa-se que as metaheurísticas GA e VNS-aiNet alcançaram os melhores resultados para todos os grupos de instância do conjunto *small*. Ambas alcançam, em todos os cenários, valores de melhor qualidade ou iguais ao MIP. Já a metaheurística ABC apenas alcançou resultados compatíveis ou de melhor qualidade frente ao MIP no conjunto $\tau = 0.80$ e $R = 0.40$. As metaheurísticas GRASP e IGS atingiram valores de melhor qualidade ou iguais ao MIP em 5 subconjuntos de instâncias.

Tabela 6.9: $\overline{RPD}$ para as metaheurísticas para instância com 12 tarefas e 3 máquinas.

		MIP	$\overline{RPD}$				
τ	R	TWET	ABC	GA	GRASP	IGS	VNS-aiNet
0.40	0.40	1967.50	0.15%	**-0.14%**	-0.11%	-0.11%	**-0.14%**
	0.80	669.80	1.93%	**0.00%**	**0.00%**	**0.00%**	**0.00%**
	1.20	1045.55	0.39%	**0.00%**	**0.00%**	**0.00%**	**0.00%**
0.80	0.40	14180.05	**-0.89%**	**-0.89%**	**-0.89%**	**-0.89%**	**-0.89%**
	0.80	3363.95	0.02%	**-0.10%**	**-0.10%**	-0.05%	**-0.10%**
	1.20	1799.55	0.40%	**0.00%**	0.09%	0.12%	**0.00%**

Tabela 6.10: Tempo em segundos das metaheurísticas para instância com 12 tarefas e 3 máquinas.

		Tempo					
τ	R	MIP	ABC	GA	GRASP	IGS	VNS-aiNet
0.40	0.40	5103,85	0,03	0,15	0,75	0,06	0,55
	0.80	208,45	0,03	0,15	0,74	0,06	0,55
	1.20	8,05	0,03	0,14	0,68	0,06	0,54
0.80	0.40	11777,65	0,02	0,09	0,44	0,04	0,34
	0.80	6234,16	0,02	0,12	0,59	0,05	0,42
	1.20	2973,20	0,03	0,15	0,73	0,06	0,54

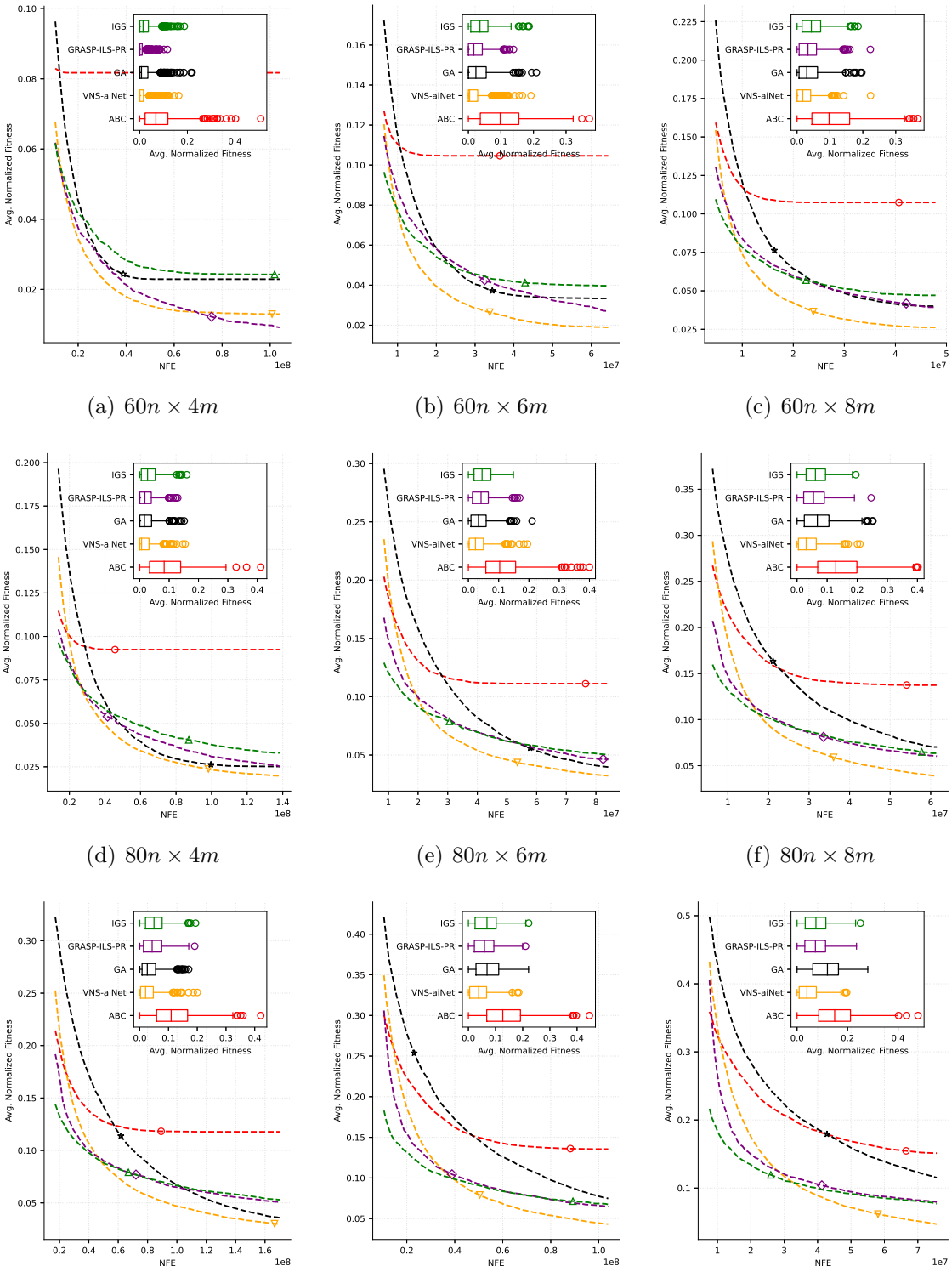
Para o conjunto de instâncias *large*, a Figura 6.1 mostra uma análise da convergência das metaheurísticas em relação ao tamanho dimensional do problema ($n \times m$). Pode-se perceber que a metaheurística VNS-aiNet apresenta, dentro do tempo de convergência predeterminado pela dimensão do problema (Seção 6.4.3), resultados de melhor qualidade frente às demais metaheurísticas em 8 dos 9 cenários avaliados. No

cenário $n = 60$ e $m = 4$, a metaheurística GRASP apresentou convergência superior ao VNS-aiNet. Pode-se observar, ainda, que a metaheurística ABC apresenta convergência muito prematura, apresentando resultados de pior qualidade frente às demais metaheurísticas em todos os cenários. Percebe-se que este padrão se repete quando detalhamos os resultados nas Tabelas 6.11, 6.12 e 6.13, considerando também os fatores $\tau \times R$ e validamos estatisticamente estes resultados, conforme as Tabelas 6.14, 6.15, 6.16 e 6.17. O VNS-aiNet apresenta $\overline{RPD}$ de melhor qualidade frente às demais metaheurísticas em 39 dos 54 grupos de instâncias avaliados, sendo que em 36 a análise apresenta significância estatística. Observa-se também que, para todos os 6 subconjuntos de instâncias do conjunto de $n = 60$ e $m = 4$, a metaheurística GRASP apresenta $\overline{RPD}$ de melhor qualidade frente às demais abordagens, sendo que em apenas 2 casos estes não apresentam diferença estatística em relação ao VNS-aiNet. Deve-se observar também que o GA apresenta-se mais competitivo nos conjuntos em que as janelas de tempo estão muito próximas entre si ($\tau = 0.8$) e estão definidas ao redor do horizonte inicial de planejamento ($R = 0.4$). Nestes cenários, a inserção de tempos ociosos tende a ser menos relevante, com o sequenciamento priorizando as tarefas que têm menor custo de avanço e/ou elevado custo de atraso no início do sequenciamento e as tarefas com baixo custo de atraso e/ou elevado custo de avanço no final do sequenciamento.

Ainda analisando as Tabelas 6.11, 6.12 e 6.13, pode-se perceber que, se dividirmos os grupos de instâncias pela dimensão $n \times m$, os subgrupos em que $\tau = 0.4$ e $R = 0.8$ apresentam a maior variação entre o melhor resultado encontrado para as instâncias e os resultados médios alcançados pelos algoritmos. Além disto, pode-se observar que esta variação aumenta na medida em que se incrementa o número de máquinas. Apesar de não analisadas, esta mesma variação pode ser observada no trabalho de Lin e Hsieh (2014) e no Capítulo 4 para a minimização do TWT com *job ready times*. Como o problema apresenta a característica de máquinas não relacionadas, o incremento no número de máquinas tende a aumentar as opções de sequenciamento que efetivamente afetam a função objetivo, devido às variações nos tempos de processamento e preparação, advindas das características do ambiente. Os experimentos indicam que este comportamento tende a ser potencializado pela maior distribuição das janelas de tempo ao longo do horizonte de sequenciamento ($\tau = 0.4$) e pela definição das datas de entrega mais distantes do instante inicial de planejamento ($R = 0.8$). Apesar disto, a influência de R tende a reduzir quando afastamos as janela de tempo para além do *makespan* esperado ($R = 1.2$).

6.5.2 Influência do tamanho das janelas de tempo no TWET

Analisando a Figura 6.2, percebe-se grande influência da dimensão das janelas de tempo nos custos relacionados ao atraso e avanço. Primeiramente, deve-se perceber que não há variação no comportamento do TWET entre os três algoritmos. Isto indica que o padrão de variação do TWET devido à variação da janela de tempo é decorrente de uma característica do problema. Os experimentos ainda indicam que, quando a janela de tempo é preservada ($tw_{ratio} = 1.0$), não há variação entre os resultados alcançados originalmente pela metaheurística e os encontrados nos experimentos de janela de tempo. Em contrapartida, percebe-se que os custo de atraso/avanço aumentam à medida em que se reduz o tamanho das janelas de tempo ($tw_{ratio} < 1.0$ implica



(a) $60n \times 4m$ (b) $60n \times 6m$ (c) $60n \times 8m$

(d) $80n \times 4m$ (e) $80n \times 6m$ (f) $80n \times 8m$

(g) $100n \times 4m$ (h) $100n \times 6$ (i) $100n \times 8m$

Figura 6.1: Convergência por NFE e *boxplot* do *fitness* médio normalizado, por combinação $n \times m$ do conjunto de instâncias *large*.

Tabela 6.11: $\overline{RPD}$ de metaheurísticas para o conjunto de instâncias com 60 tarefas.

				$\overline{RPD}$				
n	m	τ	R	ABC	VNS-aiNet	GA	GRASP	IGS
60	4	0.40	0.40	1.19%	0.13%	0.23%	**0.10%**	0.27%
			0.80	5.17%	0.66%	2.18%	**0.39%**	2.12%
			1.20	2.87%	0.72%	1.15%	**0.45%**	1.39%
		0.80	0.40	1.51%	0.20%	0.24%	**0.10%**	0.27%
			0.80	3.76%	0.68%	1.19%	**0.50%**	1.25%
			1.20	1.29%	0.24%	0.37%	**0.16%**	0.39%
	6	0.40	0.40	3.03%	**0.50%**	0.61%	0.72%	1.02%
			0.80	22.16%	**3.48%**	12.88%	7.24%	11.40%
			1.20	3.07%	**0.30%**	0.85%	0.46%	0.65%
		0.80	0.40	2.60%	**0.51%**	0.55%	0.54%	0.86%
			0.80	6.11%	**1.14%**	2.11%	1.49%	2.10%
			1.20	2.91%	**0.66%**	0.99%	0.91%	1.29%
	8	0.40	0.40	4.41%	**1.22%**	1.84%	2.07%	2.37%
			0.80	123.66%	**28.45%**	69.58%	41.34%	54.17%
			1.20	7.58%	**0.46%**	2.05%	1.11%	1.32%
		0.80	0.40	3.16%	0.80%	**0.60%**	1.08%	1.28%
			0.80	10.71%	**2.29%**	4.37%	3.14%	3.94%
			1.20	4.11%	**1.14%**	1.64%	1.99%	2.44%

Tabela 6.12: $\overline{RPD}$ de metaheurísticas para o conjunto de instâncias com 80 tarefas.

				$\overline{RPD}$				
n	m	τ	R	ABC	VNS-aiNet	GA	GRASP	IGS
80	4	0.40	0.40	1.63%	**0.38%**	0.41%	0.49%	0.66%
			0.80	5.33%	**0.91%**	2.52%	1.22%	1.74%
			1.20	2.24%	**0.09%**	0.59%	0.24%	0.23%
		0.80	0.40	2.76%	0.69%	**0.58%**	0.84%	1.13%
			0.80	3.02%	**0.67%**	0.85%	0.81%	1.13%
			1.20	1.58%	**0.32%**	0.38%	0.42%	0.48%
	6	0.40	0.40	3.23%	**0.96%**	1.24%	1.45%	1.55%
			0.80	18.24%	**4.22%**	7.83%	6.02%	6.16%
			1.20	5.62%	1.90%	3.75%	**1.09%**	1.12%
		0.80	0.40	3.58%	1.08%	**0.61%**	1.42%	1.65%
			0.80	4.50%	**1.26%**	1.98%	1.50%	2.00%
			1.20	2.46%	**0.81%**	0.94%	1.50%	1.55%
	8	0.40	0.40	4.95%	**1.73%**	3.05%	2.94%	2.78%
			0.80	50.64%	**11.46%**	26.91%	16.41%	17.85%
			1.20	2.91%	**0.37%**	0.95%	0.85%	0.60%
		0.80	0.40	4.41%	1.31%	**0.98%**	1.85%	2.22%
			0.80	8.10%	**2.01%**	4.45%	2.80%	3.05%
			1.20	5.24%	**1.70%**	3.60%	2.63%	2.87%

Tabela 6.13: $\overline{RPD}$ de metaheurísticas para o conjunto de instâncias com 100 tarefas.

				$\overline{RPD}$				
n	m	τ	R	ABC	VNS-aiNet	GA	GRASP	IGS
100	4	0.40	0.40	2.14%	**0.71%**	0.80%	1.17%	1.19%
			0.80	7.99%	**1.93%**	4.10%	2.21%	2.48%
			1.20	2.85%	**0.22%**	0.71%	0.44%	0.38%
		0.80	0.40	3.66%	1.00%	**0.54%**	1.67%	1.78%
			0.80	2.96%	**0.72%**	0.86%	1.15%	1.47%
			1.20	2.16%	**0.65%**	0.74%	1.11%	1.22%
	6	0.40	0.40	3.40%	**1.45%**	2.61%	2.05%	2.30%
			0.80	19.68%	**5.50%**	11.69%	6.34%	6.68%
			1.20	4.98%	0.87%	1.30%	1.52%	**0.43%**
		0.80	0.40	5.31%	1.55%	**1.08%**	2.41%	2.74%
			0.80	5.07%	**1.38%**	3.23%	1.99%	2.33%
			1.20	3.31%	**1.30%**	2.37%	1.97%	2.00%
	8	0.40	0.40	4.55%	**1.67%**	4.35%	3.05%	2.82%
			0.80	40.91%	**12.98%**	31.54%	14.95%	14.90%
			1.20	9.85%	2.85%	4.19%	**2.61%**	2.78%
		0.80	0.40	5.76%	**1.55%**	2.47%	2.89%	3.24%
			0.80	8.13%	**2.29%**	6.20%	3.35%	3.62%
			1.20	5.23%	**2.06%**	5.43%	3.11%	3.39%

em *ratio WET* por tarefa < 0.0). Por outro lado, os mesmos custos se reduzem com o aumento do janela de tempo ($tw_{ratio} > 1.0$ implica em *ratio WET* por tarefa > 0.0). Pode-se perceber pelos experimentos que este comportamento é bem representado por um modelo linear, apresentando boa generalização para a maior parte dos cenários avaliados ($R^2 > 0.85$). Apenas nos cenários em que $\tau = 0.4$ e $R = 1.2$ não é obtida uma boa generalização ($R^2 = 0.53$). Ainda pela Tabela 6.18, pode-se constatar que todos os níveis de $tw_{ratio} < 1.0$ apresentam a mediana de TWET estatisticamente maior a $tw_{ratio} = 1.0$ (rejeita-se H_0 e aceita-se $H_1 : m_1 - m_2 < 0$, sendo $\alpha = 0.05$). Já para $tw_{ratio} > 1.0$, em todos os cenários a mediana de TWET é inferior a $tw_{ratio} = 1.0$ (rejeita-se H_0 e aceita-se $H_1 : m_1 - m_2 > 0$ sendo $\alpha = 0.05$).

Tabela 6.14: Teste pareado de Wilcoxon, comparando o VNS-aiNet contra as outras abordagens metaheurísticas. O teste é realizado para os grupos de instâncias em que o VNS-aiNet apresentou $\overline{RPD}$ de melhor qualidade que as demais abordagens.

				$H_1 : md_1 < md_2$*			
n	m	τ	R	ABC	GA	GRASP	IGS
60	6	0.40	0.40	**2.80e-135**	**1.17e-05**	**1.74e-09**	**4.25e-26**
			0.80	**2.46e-157**	**5.85e-58**	**3.76e-27**	**3.74e-59**
			1.20	**1.19e-66**	**2.34e-07**	5.80e-02	**1.60e-09**
		0.80	0.40	**6.00e-153**	**2.04e-02**	1.53e-01	**7.94e-25**
			0.80	**7.23e-158**	**2.91e-29**	**1.26e-07**	**2.34e-33**
			1.20	**7.06e-129**	**1.72e-11**	**9.41e-07**	**5.48e-26**
	8	0.40	0.40	**8.41e-132**	**1.10e-17**	**2.02e-26**	**1.52e-42**
			0.80	**9.29e-152**	**4.52e-37**	**5.98e-19**	**1.97e-30**
			1.20	**1.24e-35**	9.95e-01	1.00e+00	9.95e-01
		0.80	0.80	**2.80e-161**	**5.15e-36**	**3.86e-15**	**3.25e-38**
			1.20	**6.33e-147**	**2.49e-12**	**8.31e-24**	**3.00e-39**
80	4	0.40	0.40	**1.37e-135**	**5.04e-03**	**1.11e-07**	**1.90e-27**
			0.80	**2.51e-137**	**2.47e-32**	**4.62e-05**	**1.27e-18**
			1.20	**8.77e-101**	**1.24e-21**	**9.66e-07**	**3.20e-14**
		0.80	0.80	**8.18e-160**	**4.97e-05**	**1.70e-04**	**2.19e-28**
			1.20	**4.33e-137**	**6.49e-04**	**4.13e-08**	**2.47e-12**
	6	0.40	0.40	**4.03e-139**	**1.48e-10**	**1.47e-14**	**8.67e-24**
			0.80	**2.99e-157**	**8.37e-32**	**3.28e-15**	**9.05e-17**
		0.80	0.80	**4.63e-152**	**8.04e-20**	**3.80e-06**	**5.26e-29**
			1.20	**1.19e-117**	**1.22e-03**	**2.98e-31**	**4.92e-43**
	8	0.40	0.40	**1.01e-128**	**1.06e-35**	**1.83e-38**	**1.15e-27**
			0.80	**1.51e-157**	**3.97e-66**	**2.54e-15**	**6.59e-26**
			1.20	**1.63e-69**	**7.90e-10**	**5.98e-06**	**1.50e-02**
		0.80	0.80	**1.15e-159**	**2.83e-56**	**2.59e-16**	**4.72e-25**
			1.20	**5.96e-133**	**4.10e-46**	**1.08e-18**	**1.29e-33**
100	4	0.40	0.40	**2.20e-138**	**1.04e-02**	**1.03e-28**	**3.13e-34**
			0.80	**5.68e-155**	**6.80e-44**	**2.47e-08**	**3.04e-13**
			1.20	**1.27e-126**	**2.45e-20**	**4.41e-06**	**7.54e-10**
		0.80	0.80	**5.68e-149**	**1.47e-06**	**4.93e-23**	**1.12e-48**
			1.20	**7.91e-139**	**1.71e-03**	**8.70e-27**	**5.89e-43**
	6	0.40	0.40	**1.30e-97**	**3.48e-41**	**7.20e-19**	**8.29e-28**
			0.80	**1.11e-155**	**6.84e-68**	**2.68e-07**	**2.54e-09**
		0.80	0.80	**3.44e-156**	**3.94e-62**	**8.70e-17**	**7.24e-34**
			1.20	**2.08e-105**	**4.85e-47**	**1.21e-20**	**3.39e-25**
	8	0.40	0.40	**5.22e-126**	**1.84e-72**	**7.83e-40**	**4.12e-34**
			0.80	**4.59e-145**	**7.20e-75**	**2.83e-10**	**1.70e-08**
		0.80	0.40	**5.77e-162**	**3.37e-28**	**2.25e-43**	**1.26e-60**
			0.80	**2.37e-162**	**2.00e-82**	**1.82e-21**	**1.01e-34**
			1.20	**8.71e-130**	**2.88e-77**	**1.29e-21**	**1.77e-27**

*No teste de hipótese, md_1 sempre corresponde à mediana obtida pelo VNS-aiNet; já md_2 indica a mediana da abordagem descrita no cabeçalho de coluna. Valores em negrito indicam os casos em que a hipótese nula (H_0) é rejeitada, aceitando assim a hipótese alternativa H_1 ($\alpha = 0.05$).

Tabela 6.15: Teste pareado de Wilcoxon comparando o GA contra as outras abordagens metaheurísticas. O teste é realizado para os grupos de instâncias em que o GA apresentou $\overline{RPD}$ de melhor qualidade que as demais abordagens.

				$H_1 : md_1 < md_2$*			
n	m	τ	R	ABC	VNS-aiNet	GRASP	IGS
60	8	0.80	0.40	**6.03e-157**	**2.73e-03**	**5.01e-18**	**1.06e-38**
80	4	0.80	0.40	**7.40e-160**	**2.97e-05**	**1.03e-15**	**5.71e-49**
	6	0.80	0.40	**1.06e-162**	**1.93e-19**	**8.31e-55**	**9.09e-67**
	8	0.80	0.40	**2.93e-162**	**6.45e-09**	**1.32e-43**	**3.39e-64**
100	4	0.80	0.40	**4.79e-165**	**6.32e-31**	**1.58e-75**	**2.26e-78**
	6	0.80	0.40	**6.08e-165**	**1.61e-14**	**2.60e-58**	**1.80e-72**

*No teste de hipótese, md_1 sempre corresponde a mediana obtida pelo GA; já md_2 indica a mediana da abordagem descrita no cabeçalho de coluna. Valores em negrito indicam os casos em que a hipótese nula (H_0) é rejeitada, aceitando assim a hipótese alternativa H_1 ($\alpha = 0.05$).

Tabela 6.16: Teste pareado de Wilcoxon comparando o GRASP contra as outras abordagens metaheurísticas. O teste é realizado para os grupos de instâncias em que o GRASP apresentou $\overline{RPD}$ de melhor qualidade que as demais abordagens.

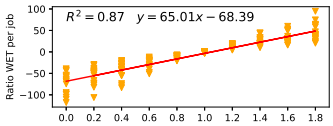
				$H_1 : md_1 < md_2$*			
n	m	τ	R	ABC	VNS-aiNet	GA	IGS
60	4	0.40	0.40	**1.79e-147**	**5.34e-02**	**7.95e-20**	**1.71e-25**
			0.80	**5.88e-155**	1.09e-01	**2.19e-48**	**6.08e-48**
			1.20	**9.27e-75**	1.04e-01	**4.36e-11**	**3.65e-16**
		0.80	0.40	**5.18e-154**	**2.23e-06**	**1.12e-14**	**2.51e-38**
			0.80	**3.12e-158**	**4.41e-05**	**1.65e-27**	**1.35e-38**
			1.20	**3.27e-132**	**7.26e-04**	**1.25e-14**	**1.09e-27**
80	6	0.40	1.20	**5.12e-95**	**3.29e-02**	**2.20e-14**	9.35e-01
100	8	0.40	1.20	**2.44e-94**	5.02e-02	**1.37e-16**	9.88e-01

*No teste de hipótese, md_1 sempre corresponde a mediana obtida pelo GRASP; já md_2 indica a mediana da abordagem descrita no cabeçalho de coluna. Valores em negrito indicam os casos em que a hipótese nula (H_0) é rejeitada, aceitando assim a hipótese alternativa H_1 ($\alpha = 0.05$).

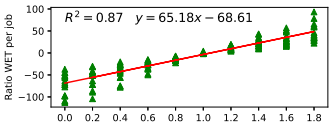
Tabela 6.17: Teste pareado de Wilcoxon comparando o IGS contra as outras abordagens metaheurísticas. O teste é realizado para os grupos de instâncias em que o IGS apresentou $\overline{RPD}$ de melhor qualidade que as demais abordagens.

				$H_1 : md_1 < md_2$*			
n	m	τ	R	ABC	VNS-aiNet	GA	GRASP
100	6	0.40	1.20	**3.01e-107**	8.76e-02	**3.34e-12**	**9.95e-09**

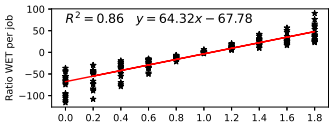
*No teste de hipótese, md_1 sempre corresponde a mediana obtida pelo IGS; já md_2 indica a mediana da abordagem descrita no cabeçalho de coluna. Valores em negrito indicam os casos em que a hipótese nula (H_0) é rejeitada, aceitando assim a hipótese alternativa H_1 ($\alpha = 0.05$).



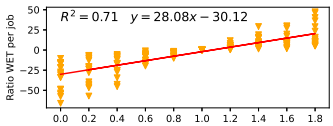
(a) VNS-aiNet - $\tau = 0.4, R = 0.4$



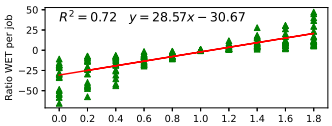
(b) IGS - $\tau = 0.4, R = 0.4$



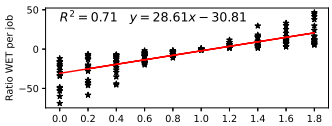
(c) GA - $\tau = 0.4, R = 0.4$



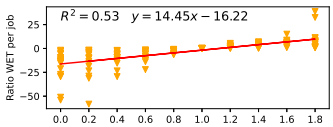
(d) VNS-aiNet - $\tau = 0.4, R = 0.8$



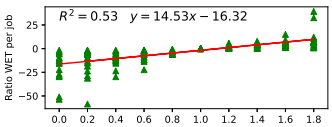
(e) IGS - $\tau = 0.4, R = 0.8$



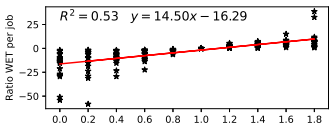
(f) GA - $\tau = 0.4, R = 0.8$



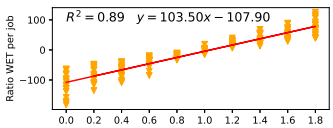
(g) VNS-aiNet - $\tau = 0.4, R = 1.2$



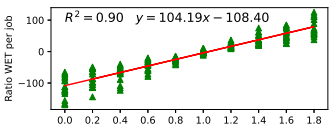
(h) IGS - $\tau = 0.4, R = 1.2$



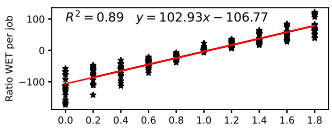
(i) GA - $\tau = 0.4, R = 1.2$



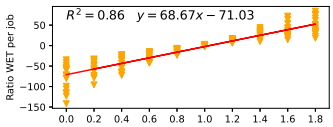
(j) VNS-aiNet - $\tau = 0.8, R = 0.4$



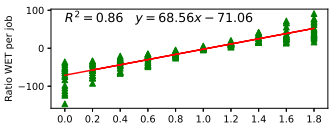
(k) IGS - $\tau = 0.8, R = 0.4$



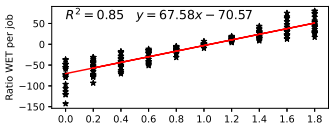
(l) GA - $\tau = 0.8, R = 0.4$



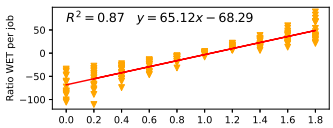
(m) VNS-aiNet - $\tau = 0.8, R = 0.8$



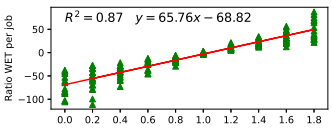
(n) IGS - $\tau = 0.8, R = 0.8$



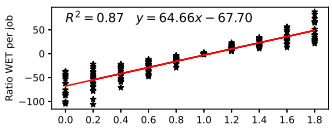
(o) GA - $\tau = 0.8, R = 0.8$



(p) VNS-aiNet -$\tau = 0.8, R = 1.2$



(q) IGS - $\tau = 0.8, R = 1.2$



(r) GA - $\tau = 0.8, R = 1.2$

Figura 6.2: Regressão linear considerando o quanto variações no tamanho da janela de tempo influenciam os custos de atraso e avanço no sequenciamento, sendo $tw_{ratio} = 1.0$ o tamanho original, $tw_{ratio} = 0.0$ a inexistência da janela de tempo e $tw_{ratio} = 1.8$ considera 1.8 vezes o tamanho original da janela de tempo.

Tabela 6.18: Teste de Wilcoxon pareado por grupo ($Algorithm \times \tau \times R$), comparando as instâncias com janelas de tempo originais e reescaladas pelo fator descrito na coluna.

			$H_1 : md_1 < md_2$*					$H_1 : md_1 \neq md_2$*	$H_1 : md_1 > md_2$*			
Algoritmo	τ	R	0.0	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8
VNS-aiNet	0.40	0.40	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	1.76e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		0.80	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	1.24e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		1.20	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	1.16e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
	0.80	0.40	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	7.11e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		0.80	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	9.48e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		1.20	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	2.27e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
GA	0.40	0.40	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	5.00e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		0.80	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	9.48e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		1.20	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	7.90e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
	0.80	0.40	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	5.28e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		0.80	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	1.70e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		1.20	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	1.99e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
IGS	0.40	0.40	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	7.44e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		0.80	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	1.22e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		1.20	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	8.78e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
	0.80	0.40	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	3.96e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		0.80	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	5.86e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**
		1.20	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**	7.95e-01	**1.96e-04**	**1.96e-04**	**1.96e-04**	**1.96e-04**

*No teste de hipótese, md_1 sempre corresponde à mediana obtida pelos experimentos realizados sobre as instâncias sem reescalar as janelas de tempo; já md_2 indica a mediana da abordagem sobre as instâncias com janelas de tempo reescaladas pelo fator descrito no cabeçalho da coluna. Valores em negrito indicam os casos em que a hipótese nula (H_0) é rejeitada, aceitando, assim, a hipótese alternativa H_1 ($\alpha = 0.05$).

6.6 Conclusões

Neste capítulo propomos um estudo sobre a influência de janelas de tempos na minimização do TWET sobre problemas de sequenciamento em ambientes de máquinas paralelas não relacionadas com tempos de preparação dependentes de sequência e das máquinas, considerando a possibilidade de inserção de tempos ociosos entre o processamento de tarefas. Segundo o nosso conhecimento, trata-se do primeiro estudo apresentado na literatura abordando esta questão para esta classe de problemas. Analisamos um MIP proposto e cinco metaheurísticas previamente propostas na literatura para o ambiente de máquinas estudado. Para a avaliação de métodos metaheurísticos, adaptamos, para máquinas paralelas, um método de inserção de tempos ociosos proposto para máquinas única. Além disto, devido ao aumento no custo computacional da função objetivo acarretado pelo método de inserção de tempos ociosos, propusemos um método de busca local reduzindo estruturas de vizinhança previamente estudadas no Capítulo 4.

Na análise do MIP observou-se que, mesmo para instâncias com baixa dimensão, este não é capaz de convergir para soluções ótimas em todos os cenários avaliados, dentro do tempo definido como critério de parada. Observamos que, para algumas instâncias do conjunto de baixa dimensão, algumas metaheurísticas encontram soluções com valores de TWET de melhor qualidade frente ao MIP, sendo estas soluções validadas posteriormente pelo MIP. Comparando as metaheurísticas para os conjuntos de alta dimensão, de modo geral, a metaheurística VNS-aiNet apresentou desempenho superior às demais. Apesar disto, em alguns cenários específicos, outras metaheurísticas apresentam desempenho superior. Nos grupos de instância de menor dimensão ($n = 60$) e com menor número de máquinas ($m = 4$), a metaheurística GRASP apresenta desempenho superior às demais. Já o GA apresentou-se mais relevante nos cenários em quem as janelas de tempo se sobrepõem e se concentram próximo ao instante inicial de planejamento.

Já ao analisar as características do problema, percebeu-se que os fatores de número de máquinas e distribuição das janelas de tempo ao longo do horizonte de planejamento, podem influenciar na variação dos resultados alcançados pelas metaheurísticas. Além disto, ao analisar como o tamanho das janelas de tempos influenciam no TWET, concluiu-se que, independente do algoritmo, existe uma variação da resposta definida pelo tamanho da janela. Observou-se que, à medida que o tamanho das janelas de tempo diminui, os custos relativos ao TWET aumentam linearmente, sendo o oposto observado quando se aumenta o tamanho da janela de tempo. Estes resultados mostram-se bastante relevantes, já que permitem concluir que, quando a filosofia JIT é aplicada em ambientes UPMSMDST, com datas de entrega estritamente restritivas, estas tendem a incrementar os custos referentes a atrasos e avanços; em contrapartida, a negociação de janelas de tempo com clientes e fornecedores pode reduzir consideravelmente estes custos.

Capítulo 7

Conclusões e considerações finais

7.1 Conclusões Gerais

Nesta tese foram estudadas metodologias para o projeto de soluções computacionais, baseadas em metaheurísticas, direcionadas a problemas de sequenciamento de tarefas, em ambientes de máquinas paralelas não relacionadas, com tempos de preparação dependentes da sequência e das máquinas (UPMSMDST). Para isto, primeiramente, realizamos uma revisão da literatura e o levantamento do estado da arte de abordagens computacionais destinadas à solução de problemas de sequenciamento no ambiente UPMSMDST. Neste levantamento, foram identificadas três lacunas importantes na literatura apresentadas na Seção 2.6 e resumidas como:

(a) Pouca importância é dada na literatura de sequenciamento de tarefas em ambientes UPMSMDST para verificar a efetividade de componentes de busca local no processo de busca de metaheurísticas. Isto dificulta a construção de operadores de busca local eficientes, já que se possui pouca informação a respeito dos movimentos que são mais efetivos para a construção de estruturas de vizinhança, quais métodos são mais eficazes na exploração das estruturas, se a ordem de exploração das estruturas influencia no desempenho dos operadores de busca local, dentre outras informações pertinentes a construção de operadores de busca local.

(b) Características do critério de otimização são comumente usadas na construção dos operadores de metaheurística. Estas características são usadas para guiar o processo de busca e tornar as abordagens mais eficientes. O projeto de metaheurísticas guiado por características do critério de otimização tem tornado as abordagens muito dependentes dos critérios. Isto dificulta a utilização das abordagens propostas na literatura em problemas com critérios de otimização com pouca ou nenhuma visibilidade teórica.

(c) Apesar de serem encontrados na literatura estudos que realizam o sequenciamento de tarefas em ambientes UPMSMDST guiados por uma política JIT, nenhum trabalho foi encontrado considerando a utilização de janelas de tempo de entrega. Entretanto, sabe-se que a utilização de janelas de tempo de entrega permite uma maior flexibilidade no sequenciamento auxiliando na redução de custos provenientes dos atrasos e avanços.

Visando solucionar estas lacunas encontradas na literatura, esta tese realizou três estudos principais. O primeiro, encontrado no Capítulo 4, é um estudo para o projeto de operadores de busca local. Neste estudo é proposta uma metodologia detalhada para análise dos componentes presentes nos operadores de busca local para a minimização do TWT em ambiente UPMSMDST. Para isto, são propostas seis estruturas de vizinhança baseadas em movimentos e estruturas de vizinhança previamente propostas na literatura. Além disto, foram avaliadas três variações da metaheurística VND como operador de busca local, além de diferentes formas e ordens de exploração das estruturas de vizinhança. Para avaliar a metodologia de projeto dos operadores de busca local, utilizamos três metaheurísticas previamente propostas na literatura para o critério de otimização do TWT.

Através deste estudo, concluímos que, para ambientes com o critério de otimização do TWT, uma das estruturas de vizinhança baseada em movimentos de troca, denominada NE.ES, dominou a convergência em todas as metaheurísticas. Os experimentos indicaram ainda que a NE.ES encontrava complementariedade em um estrutura chamada NE.IGS, esta baseada nos operadores de construção e desconstrução de soluções da metaheurística IGS. Uma terceira estrutura, denominada NE.EI, também mostrou-se bastante importante na convergência dos operadores de busca local. As demais estruturas de vizinhança se mostram apenas como peças de ajuste fino em alguns cenários; já em outros cenários, não acarretam incremento significativo ao desempenho das abordagens. Estes resultados são importantes, já que mostram que muitas vezes operadores (elementos computacionais) podem ser incorporados ao projeto de métodos metaheurísticos, mas sem a devida validação dos mesmos, podem tornar estes métodos muito complexos e, ainda sim, não acarretaram em ganho de desempenho.

Também é mostrado que a ordem de exploração das estruturas de vizinhança influi diretamente nos resultados alcançados pelos operadores de busca local. Neste Capítulo, todos os experimentos são realizados sobre três variações do VND, sendo que apenas a variação *Basic-VND* considera ordem de exploração das estruturas. Para as três metaheurísticas avaliadas, a variação *Basic-VND* apresentou melhor desempenho frente às demais. Apesar de concluir que a ordem de exploração das estruturas de vizinhança influi no desempenho das abordagens, não foi possível definir um padrão de preferência entre as duas ordens de exploração propostas.

No Capítulo 4 ainda é mostrado que um projeto adequado dos operadores de busca local pode incrementar consideravelmente o desempenho de metaheurísticas já estabelecidas na literatura para ambiente UPMSMDST. Para as três metaheurísticas avaliadas, é mostrado que estas têm seu desempenho incrementado devido à troca dos operadores de busca local. Além disto, esta substituição de operadores de busca local acarreta resultados superiores aos de abordagens de estado da arte para o problema avaliado. A metaheurística GA, integrada ao VND projetado no Capítulo, apresentou desempenho superior a todas as abordagens de estado da arte em todos os cenários avaliados.

Como pode ser visto, o estudo de projeto de operadores de busca local, para metaheurísticas destinadas a ambientes UPMSMDST, apresenta várias contribuições. É definida uma metodologia para o estudo de componentes de operadores de busca local integrados à metaheurísticas. Identificam-se padrões, que devem ser melhor analisados na construções de métodos de busca local em ambientes UPMSMDST.

Mostram-se quais as reais contribuições de estruturas de vizinhança, método de exploração e ordem de exploração das mesmas. Estas informações são utilizadas para o projeto de novos operadores de busca local nos capítulos subsequentes desta tese. Através dos métodos de busca local propostos, definimos novas abordagens de estado da arte para minimização do TWT em ambientes UPMSMDST.

Já referente à segunda lacuna encontrada, no Capítulo 5 é proposta uma abordagem metaheurística para a solução do sequenciamento em ambientes UPMSMDST com o intuito de reduzir a dependência dos critérios de otimização no processo de busca. A metaheurística proposta é denominada VNS-aiNet, sendo uma abordagem híbrida, baseada em algoritmos imuno inspirados da família aiNet e métodos guiados por operadores de busca local VNS e ILS. As conclusões obtidas no Capítulo 4 são utilizadas para o projeto do operador de busca local, sendo usadas quatro das seis estruturas de vizinhança propostas, assim como a versão *Basic-VND* e a ordem de exploração das estruturas de vizinhança. A escolha da máquina que guia o processo de busca local é o único ponto que utiliza características do critério de otimização no processo de busca. Assim, no projeto do operador de busca local, este ponto é isolado em uma função que deve ser definida para cada critério de otimização. Todos os demais operadores e componentes propostos não utilizam características do critério de otimização para guiar a busca.

Para mostrar que a abordagem reduz a dependência dos critérios de otimização no processo de busca, é avaliado se a abordagem atinge os dois objetivos definidos no capítulo. Através de uma análise dos componentes propostos para a abordagem VNS-aiNet, conclui-se que os operadores projetados atuam de maneira integrada e cooperativa, o que possibilita à abordagem encontrar resultados de alta qualidade para diferentes critérios de otimização. É mostrado, por exemplo, que, para o critério de otimização do *makespan* os parâmetros de mutação autoadaptáveis e fator de maturação de células influem positivamente para incrementar o desempenho da abordagens. Já para a minimização do TWT, o fator de supressão atua em conjunto com o fator de maturação para incrementar o projeto da abordagem, enquanto os parâmetros de mutação autoadaptáveis não alteraram o comportamento do processo de otimização. É observado, assim, que diferentes critérios demandam diferentes componentes da metaheurística. Poderíamos supor, assim, que o uso de determinados operadores decrementaria os resultados em determinados cenários. Entretanto, os experimentos indicam que os operadores que não incrementam o desempenho do VNS-aiNet para um determinado cenário/critério não decrementam os resultados para o mesmo cenário/critério.

Além destes resultados, são realizados quatro estudos de caso, cada um destinado a um critério de otimização. Em cada estudo de caso, a abordagem proposta é comparada com abordagens estado da arte projetadas especificamente para o critério de otimização. Para os critérios TWT, TT e TWCT é mostrado que a abordagem supera, em todos os cenários avaliados, as abordagens estado da arte em tempo computacional compatível. Já para o *makespan*, critério mais estudado na literatura, é mostrado que o VNS-aiNet apresenta resultados superiores às demais abordagens metaheurísticas, em pouco mais de 50% dos grupos de instâncias avaliados, sem despender esforço computacional desproporcional. Além disto, na análise estatística realizada para o desempenho geral, é mostrado que os resultados do VNS-aiNet são superiores às demais abordagens metaheurísticas, sendo igual apenas a um algoritmo

de seleção clonal (CSA) no conjunto de instâncias que os tempos de processamento e preparação são balanceados.

Apesar do desempenho elevado para critérios de otimização específicos, os experimentos são realizados com o objetivo principal de validar a adaptação/eficiência da abordagem quando aplicada a diferentes critérios de otimização. Assim, a principal contribuição do Capítulo 5 está na possibilidade de utilizar a abordagem para encontrar soluções eficientes para critérios de otimização de pouca visibilidade teórica. Adicionalmente a isto, a abordagem pode servir como *benchmark* para validação de projeto de soluções computacionais específicas a um critério de otimização. Entretanto, vale destacar que, para qualquer critério de otimização não estudado nesta tese, é aconselhável a validação da eficiência dos resultados, gerados pela abordagem VNS-aiNet, através de resultados gerados por um MIP para problemas de baixa dimensionalidade, ou mesmo, pela validação empírica dos mesmos.

A terceira lacuna explorada nesta tese refere-se ao projeto de metaheurísticas para a solução de problemas guiados por uma política JIT, mas com a possibilidade de utilização de janelas de tempo em ambientes UPMSMDST. Esta lacuna é estudada no Capítulo 6, no qual é realizado um estudo que indica os requisitos necessários para a construção de abordagens destinadas à minimização dos atrasos e avanços ponderados, com janelas de tempo. A construção se dá a partir de metaheurísticas previamente propostas na literatura para critérios de otimização que envolvem atraso e/ou avanço.

É apresentado neste capítulo que, para cada avaliação de função objetivo, é necessária a resolução de um subproblema de otimização para a inserção de tempos ociosos. Para isto, é adaptado, para máquinas paralelas não relacionadas, um algoritmo de inserção de tempos ociosos previamente proposto na literatura para máquina única. Este algoritmo acarreta em elevado incremento da ordem de complexidade das avaliações da função objetivo, o que acarreta na necessidade de projeto de um novo operador de busca local. Sendo assim, as conclusões do Capítulo 4 são utilizadas para o projeto de um operador de busca local, reduzindo e condensando estruturas vizinhança.

As adaptações são avaliadas em cinco metaheurísticas previamente estudadas na literatura. Três são destinadas à minimização do TWT (i.e. ABC, IGS, GA), também estudadas no Capítulo 4; uma destinada à minimização dos atrasos e avanços sem a utilização de janelas de tempo (i.e. GRASP-ILS-PathRelinking) e, por fim, a abordagem VNS-aiNet proposta no Capítulo 5. Além disto, é proposto um MIP. Através dos experimentos, é possível concluir que o MIP apresenta dificuldades de encontrar resultados ótimos, mesmo para instâncias de baixa dimensão. Já quando as metaheurísticas são avaliadas, na maior parte dos cenários o VNS-aiNet apresenta desempenho superior às demais abordagens estudadas. Em um cenário de dimensão média, o GRASP apresenta desempenho superior ao VNS-aiNet. Já o GA apresenta resultados superiores às demais abordagens quando as janelas de tempo se localizavam condensadas, próximas ao instante inicial de planejamento.

No Capítulo 6 ainda é avaliado como, em ambientes UPMSMDST, a utilização de janelas de tempo possibilita a redução dos custos referentes a atrasos e avanços. Através destes experimentos, conclui-se que existe uma correlação linear entre o tamanho das janelas de tempo e os custos decorrentes de atraso e avanço. É mostrado que, a medida que se aumenta as janelas de tempo, os custos são linearmente re-

duzidos. Esta é uma conclusão relevante, uma vez que indica que, em ambientes UPMSMDST, a negociação de janelas de tempo com clientes e fornecedores, além de promover uma maior flexibilidade das entregas, pode acarretar em um redução significativa dos custos acarretados por atrasos e avanços de produção.

Considerando as contribuições listadas acima, pode-se concluir que o objetivo central desta tese é atingido, já que claramente a tese contribui para tornar mais eficaz o projeto de abordagens de inteligência computacional aplicadas a problemas de sequenciamento em ambiente UPMSMDST. Além disto, todos os objetivos específicos listados na Seção 1.2.1 do Capítulo 1 também são alcançados.

7.2 Publicações associadas a esta Tese

Nesta seção estão apresentados os artigos publicados ou em processo de avaliação em periódicos e os artigos aceitos e publicados em anais de congresso durante este período de doutoramento.

7.2.1 Artigos publicados em periódicos

1. Diana, R. O. M.; de Souza, S. R.; Wanner, E. F. (2021) *A robust multi-response VNS-aiNet approach for solving scheduling problems under unrelated parallel machines environments. Expert Systems with Applications*, v.182 DOI:10.1016/j.eswa.2021.115140.

2. Diana, R. O. M.; de Souza, S. R. (2020) *Analysis of variable neighborhood descent as a local search operator for total weighted tardiness problem on unrelated parallel machines. Computers & Operations Research*, v.117 DOI:10.1016/j.cor.2020.104886.

3. Diana, R. O. M.; de Souza, S. R.; Filho, M. F. F. (2018) *A Variable Neighborhood Descent as ILS local search to the minimization of the total weighted tardiness on unrelated parallel machines and sequence dependent setup times. Electronic Notes in Discrete Mathematics*, v.66, p.191-198 DOI:10.1016/j.endm.2018.03.025.

7.2.2 Artigos em processo de avaliação em periódicos

1. Diana, R. O. M.; de Souza, S. R.; Wanner, E. F. (2021) *Weighted earliness and tardiness minimization with idle times: An analysis of time window influence on scheduling over unrelated parallel machines environments.* Submetido ao *Computers & Operations Research.*

7.2.3 Artigos publicados em anais de congresso

1. Diana, R. O. M.; de Souza, S. R.; Wanner, E. F.; Filho, M. F. F. (2017) *Hybrid metaheuristic for combinatorial optimization based on immune network for optimization and VNS. GECCO '17: proceedings of the Genetic and Evolutionary Computation Conference* , 2017, p. 251-258, Berlim, Germany DOI:10.1145/3071178.3071269.

2. Diana, R. O. M.; de Souza, S. R.; Filho, M. F. F. (2017) *A Variable Neighborhood Descent as ILS local search to the minimization of the total weighted*

tardiness on unrelated parallel machines and sequence dependent setup times. VNS '17: Proceedings of 5th International Conference on Variable Neighborhood Search, 2017, p. 191-198, Ouro Preto (MG), Brazil.

3. Diana, R. O. M.; de Souza, S. R. (2018) Minimização do atraso e avanço em máquinas paralelas não relacionadas: Análise de uma abordagem VND integrada à metaheurística GRASP. *Anais do 50Âº Simpósio Brasileiro de Pesquisa Operacional*, 2018, Rio de Janeiro (RJ), Brazil.

4. Siqueira, E. C.; Diana, R. O. M.; Souza, M. J. F.; de Souza, S. R. (2016) *A Study concerning the application of Genetic Algorithms for solving the Multi-Objective Hybrid Flowshop Scheduling Problem. Anais do XIII Encontro Nacional de Inteligência Artificial e Computacional.* 2016, Recife (PE), Brazil.

7.3 Propostas de Trabalhos Futuros

Como propostas de trabalhos futuros, sugere-se dar continuidade a esta pesquisa nas seguintes direções:

(i) A metodologia proposta para o projeto e estudo de operadores de busca local apresentou resultados relevantes para os ambientes UPMSMDST. Recomenda-se aplicar a mesma metodologia para o estudo de outros problemas de otimização combinatória, avaliando as principais estruturas de vizinhança já propostas na literatura para o problema. Assim, pode-se gerar indicadores para a construção de métodos de busca local cada vez mais eficazes, removendo componentes desnecessários;

(ii) O algoritmo VNS-aiNet adaptou-se bem a diferentes critérios de otimização. Isto indica que o mesmo tem mecanismos que podem ser incorporados em algoritmos de otimização multiobjetivo para solução de problemas em ambientes UPMSMDST. Assim, pode-se avaliar quais mecanismos se adaptam bem a uma abordagem multiobjetivo;

(iii) Para problemas destinados à minimização dos atrasos e avanços com janelas de tempo, apesar de termos considerado a inserção de tempos ociosos, não avaliamos quais fatores favorecem às soluções, devido a inserção dos mesmo. Uma avaliação destes fatores ajudaria a avaliar os cenários nos quais a inserção de tempos ociosos é benéfica ao sequenciamento.

(iv) Como discutido, a inserção de tempos ociosos incrementa consideravelmente a ordem de complexidade da avaliação da função objetivo. Isto acarreta um impacto significativo dos operadores de busca local. Devido a esta característica do problema, se tornam necessários estudos para redução de dimensão dos operadores de busca local, desde que estes não acarretem em perda de eficiência das metaheurísticas.

(v) Métodos de construção de soluções também não foram estudados para problemas de minimização dos atrasos e avanços com janelas de tempo. Estes podem

ajudar e reduzir consideravelmente tempos de convergência das abordagens metaheurísticas, desde que considerem tanto a inserção de diversidade além de soluções de alto desempenho. Além disto, operadores específicos para o problema podem ser propostos para reduzirem o tempo computacional e aumentarem o desempenho das abordagens metaheurísticas.

Apêndice A

Elementos base para os métodos computacionais propostos

Neste capítulo são apresentados algumas definições base para as metodologias propostas.

A.1 Representação da Solução

Para todos os algoritmos propostos neste trabalho, uma solução s é representada como um vetor de m posições, em que cada posição representa uma máquina. Cada posição do vetor de máquinas contém um apontador para uma lista encadeada de tarefas, em que a posição da tarefa na lista representa a ordem na qual ela é processada. Este tipo de estrutura se torna muito eficiente devido ao baixo consumo de memória para problemas de grandes dimensões e facilita a realização dos movimentos utilizados no trabalho.

A.2 Inicialização aleatória de soluções

Inicialmente, é gerada uma solução s vazia, ou seja, a lista de máquinas é iniciada, sendo que cada máquina aponta para uma lista encadeada de tarefas vazia. Em seguida, considerando que $\bar{N}$ é a lista de tarefas a serem alocadas, são realizadas $|\bar{N}|$ iterações, sendo a cada iteração uma tarefa $k \in \bar{N}$ selecionada aleatoriamente, com probabilidade uniforme, e retirada de $\bar{N}$; uma máquina $i \in M$ também selecionada aleatoriamente com probabilidade uniforme. Em seguida, a tarefa k é inserida na solução s, ao final da lista encadeada que representa a sequência de uma máquina i. Este processo é repetido até que todas as tarefas sejam sequenciadas, ou seja, $\bar{N}$ se torne vazia.

A.3 Funções Base

Dada a referida estrutura e considerando s como uma solução, os operadores utilizados neste trabalho para os métodos propostos são apresentados a seguir:

- $M(s, k)$**:** Retorna qual a máquina i em que a tarefa k está atribuída na solução s.
- **insercao(**s**,** i**,** k**,** w**,** p**):** Este operador realiza na solução s a remoção da tarefa k da máquina i e a inserção de k na posição p da máquina w.
- **avaliar_insercao(**s**,** $f(s)$**,** i**,** k**,** w**,** p**):** Avalia o critério de otimização $f(s)$ associado à possível inserção da tarefa k, presente na máquina i, na posição p da máquina w.
- **troca(**s**,** i**,** k**,** w**,** j**):** Realiza na solução s a troca de tarefas k e j, assim, sendo k uma tarefa presente na máquina i e j uma tarefa sequenciada na máquina w.
- **avaliar_troca(**s**,** $f(s)$ i**,** k**,** w**,** j**):** Avalia o critério de otimização $f(s)$ associado à possível troca na solução s entre a tarefa k presente na máquina i e a tarefa j da máquina w.
- **jobs(**s**,** i**):** Retorna a lista de tarefas atribuídas a máquina i na solução s.
- U[a, b]: Retorna um número aleatório com probabilidade uniforme entre dentro do intervalo a, b.
- **rand(**L**):** Retorna um elemento aleatório da L utilizando probabilidade uniforme.
- **randomiza(**L**):** Retorna uma nova lista com todos os elementos presentes em L, mas nova lista construída de forma randomizada.
- **remover(**L**,** p**):** Este operador realiza a retirada do elemento presente na posição p da lista L.
- **range(**a**,** b**):** Retorna uma lista sequencial com os pontos inteiros entre os elementos a e b, ou seja retorna $[a, a+1, a+2, ..., b]$.

A.4 Procedimentos de Descida

Nesta seção é apresentado os algoritmos base de descida utilizados neste trabalho.

Algoritmo 24 Descida Inserção.

```
1: function DESCIDA_INSERCAO(s, f(.), i, w, tipo)
2:     s* ← s;
3:     if tipo == BI then
4:         s* ← BI.I(s, f(.), i, w);
5:     else if tipo == FI then
6:         s* ← FI.I(s, f(.), i, w);
7:     end if
8:     return s*;
9: end function
```

Algoritmo 25 Descida *Best Improvement* Inserção.

```
1: function BI.I(s, f(.), i, w)
2:     f* ← f(s);
3:     k* ← NULL;
4:     p* ← NULL;
5:     jobs_i ← jobs(s, i);
6:     jobs_w ← jobs(s, w);
7:     P_w ← range(1, |jobs_w| + 1);
8:     for each k ∈ jobs_i do:
9:         for each p ∈ P_w do:
10:            f' ← avaliar_insercao(s, f(.), i, k, w, p);
11:            if f' < f* then
12:                f*, k*, p* ← f', k, p;
13:            end if
14:        end for each
15:    end for each
16:    if f* < f(s) then
17:        s ← insercao(s, i, k*, w, p*);
18:    end if
19:    return s;
20: end function
```

Algoritmo 26 Descida *First Improvement* Inserção.

```
1: function FI.I(s, f(.), i, w)
2:     f* ← f(s);
3:     k* ← NULL;
4:     p* ← NULL;
5:     jobs_i ← randomiza(jobs(s, i));
6:     jobs_w ← jobs(s, w);
7:     while |jobs_i| > 0 and f* == f(s) do
8:         k ← remover(jobs_i, 1);
9:         p ← 1;
10:        while p <= |jobs_w| + 1 and f* == f(s) do
11:            f' ← avaliar_insercao(s, f(.), i, k, w, p);
12:            if f' < f* then
13:                f*, k*, p* ← f', k, p;
14:            end if
15:            p ← p + 1;
16:        end while
17:    end while
18:    if f* < f(s) then
19:        s ← insercao(s, i, k*, w, p*);
20:    end if
21:    return s;
22: end function
```

Algoritmo 27 Descida Troca.

```
1: function DESCIDA_TROCA(s, f(.), i, w, tipo)
2:     s* ← s;
3:     if tipo == BI then
4:         s* ← BI.T(s, f(.), i, w);
5:     else if tipo == FI then
6:         s* ← FI.T(s, f(.), i, w);
7:     end if
8:     return s*;
9: end function
```

Algoritmo 28 Descida *Best Improvement* Troca.

```
1: function BI.T(s, f(.), i, w)
2:     f* ← f(s);
3:     k* ← NULL;
4:     j* ← NULL;
5:     jobs_i ← jobs(s, i);
6:     jobs_w ← jobs(s, w);
7:     for each k ∈ jobs_i do:
8:         for each j ∈ jobs_w do:
9:             f' ← avaliar_troca(s, f(.), i, k, w, j);
10:            if f' < f* then
11:                f*, k*, j* ← f', k, j;
12:            end if
13:        end for each
14:    end for each
15:    if f* < f(s) then
16:        s ← troca(s, i, k*, w, j*);
17:    end if
18:    return s;
19: end function
```

Algoritmo 29 Descida *First Improvement* Troca.

```
1: function FI.T(s, f(.), i, w)
2:     f* ← f(s);
3:     k* ← NULL;
4:     j* ← NULL;
5:     jobs_i ← randomiza(jobs(s, i));
6:     jobs_w ← randomiza(jobs(s, w));
7:     while |jobs_i| > 0 and f* == f(s) do
8:         k ← remover(jobs_i, 1);
9:         p ← 1;
10:        while p <= |jobs_w| and f* == f(s) do
11:            j ← jobs_w[p];
12:            f' ← avaliar_troca(s, f(.), i, k, w, j);
13:            if f' < f* then
14:                f*, k*, j* ← f', k, j;
15:            end if
16:            p ← p + 1;
17:        end while
18:    end while
19:    if f* < f(s) then
20:        s ← troca(s, i, k*, w, j*);
21:    end if
22:    return s;
23: end function
```

Referências Bibliográficas

Afzalirad, Mojtaba e Rezaeian, Javad. (2016). Design of high-performing hybrid meta-heuristics for unrelated parallel machine scheduling with machine eligibility and precedence constraints. *Engineering Optimization*, v. 48, n. 4, p. 706–726. doi: 10.1080/0305215X.2015.1042475.

Al-Hinai, Nasr e ElMekkawy, T.Y. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics*, v. 132, n. 2, p. 279 – 291. ISSN 0925-5273. doi: https://doi.org/10.1016/j.ijpe.2011.04.020.

Al-Salem, A. (2004). Scheduling to minimize makespan on unequal parallel machines with sequence dependent setup times. *Engineering Journal of the University of Qatar*, v. 17, n. 1, p. 177–187.

Allahverdi, Ali. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, v. 246, n. 2, p. 345 – 378. doi: https://doi.org/10.1016/j.ejor.2015.04.004.

Allahverdi, Ali; Gupta, Jatinder N.D. e Aldowaisan, Tariq. (1999). A review of scheduling research involving setup considerations. *Omega*, v. 27, p. 219–239.

Allahverdi, Ali; Ng, C.T.; Cheng, T. C. E. e Kovalyov, Mikhail Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, v. 187, p. 985–1032.

Armentano, V. A. e Filho, M. F. França. (2007). Minimizing total tardiness in parallel machine scheduling with setup times: An adaptive memory- based grasp approach. *European Journal of Operational Research*, v. 183, p. 100–114.

Arnaout, J.-P.; Musa, R. e Rabadi, G. (2014). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines—part II: enhancements and experimentations. *Journal of Intelligent Manufacturing*, v. 25, n. 1, p. 43–53.

Arnaout, J.-P.; Rabadi, G. e Musa, R. (2010). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, v. 21, n. 6, p. 693–701.

Arnaout, Jean-Paul. Feb(2019). A worm optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Annals of Operations Research*, v. 285, p. 273–293. doi: 10.1007/s10479-019-03138-w.

Avci, Mustafa e Topaloglu, Seyda. (2017). A multi-start iterated local search algorithm for the generalized quadratic multiple knapsack problem. *Computers & Operations Research*, v. 83, p. 54 – 65. ISSN 0305-0548. doi: 10.1016/j.cor.2017.02.004.

Azzouz, Ameni; Pan, Po-An; Hsu, Peng-Hsiang; Lin, Win-Chin; Liu, Shangchia; Said, Lamjed Ben e Wu, Chin-Chia. (2020). A two-stage three-machine assembly scheduling problem with a truncation position-based learning effect. *Soft Computing*, v. 24, n. 14, p. 10515–10533. doi: https://doi.org/10.1007/s00500-019-04561-8.

Baker, K.R. (1974). *Introduction to sequencing and scheduling.* John Wiley & Sons, Inc.

Balakrishnan, Nagraj; Kanet, John J. e Sridharan, V. (1999). Early/tardy scheduling with sequence dependent setups on uniform parallel machines. *Computers and Operations Research*, v. 26, n. 2, p. 127 – 141. doi: http://dx.doi.org/10.1016/S0305-0548(98)00051-3.

Barnes, John e Brennan, J. 03(1977). An improved algorithm for scheduling jobs on identical machines. *AIIE Transactions*, v. 9, p. 25–31. doi: 10.1080/05695557708975117.

Bellman, Richard Ernest. (reimpressão (2003)). *Dynamic Programming.* Dover Publications, Inc., New York, NY, USA, sixth (1972) edição. ISBN 0486428095.

Bilge, Ümit; Kıraç, Furkan; Kurtulan, Müjde e Pekgün, Pelin. (2004). A tabu search algorithm for parallel machine total tardiness problem. *Computers and Operations Research*, v. 31, n. 3, p. 397 – 414. doi: http://doi.org/10.1016/S0305-0548(02)00198-3.

Birbil, Ş İlker e Fang, Shu-Chering. (2003). An electromagnetism-like mechanism for global optimization. *Journal of Global Optimization*, v. 25, n. 3, p. 263–282. doi: 10.1023/A:1022452626305.

Blum, Christian; Puchinger, Jakob; Raidl, Gunther R. e Roli, Andrea. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, v. 11, n. 6, p. 4135–4151.

Blum, Christian e Roli, Andrea. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *Journal ACM Computing Surveys*, v. 35, p. 268–308.

Burnet, F. M. (1957). A modification of jerne's theory of antibody production using the concept of clonal selection. *Australian Journal of Science*, v. 20, p. 67–69.

Caniyilmaz, Erdal; Benli, Betül e Ilkay, Mehmet S. Apr(2015). An artificial bee colony algorithm approach for unrelated parallel machine scheduling with processing set restrictions, job sequence-dependent setup times, and due date. *The International Journal of Advanced Manufacturing Technology*, v. 77, n. 9, p. 2105–2115. doi: 10.1007/s00170-014-6614-9.

Cao, Jing e Bedworth, David D. (1992). Flow shop scheduling in serial multi product process with transfer and set-up times. *International Journal of Production Research*, v. 30, n. 8, p. 1819–1830.

Chen, Chun-Lung. (2012). Iterated hybrid metaheuristic algorithms for unrelated parallel machines problem with unequal ready times and sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, v. 60, n. 5, p. 693 – 705. doi: 10.1007/s00170-011-3623-9.

Chen, Chun-Lung. Jun(2019). Iterated population-based vnd algorithms for single-machine scheduling with sequence-dependent setup times. *Soft Computing*, v. 23, n. 11, p. 3627–3641. doi: 10.1007/s00500-018-3014-3.

Chen, Chun-Lung e Chen, Chuen-Lung. Aug(2008). Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, v. 43, n. 1, p. 161.

Chen, Jeng-Fung. (2009). Scheduling on unrelated parallel machines with sequence- and machine-dependent setup times and due-date constraints. *The International Journal of Advanced Manufacturing Technology*, v. 44, n. 11, p. 1204 – 1212. doi: 10.1007/s00170-008-1917-3.

Chen, Jeng-Fung. Dec(2015). Unrelated parallel-machine scheduling to minimize total weighted completion time. *Journal of Intelligent Manufacturing*, v. 26, n. 6, p. 1099–1112. doi: https://doi.org/10.1007/s10845-013-0842-y.

Chen, Jeng-Fung e Wu, Tai-Hsi. (2006). Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. *Omega*, v. 34, n. 1, p. 81 – 89. doi: http://doi.org/10.1016/j.omega.2004.07.023.

Cheng, Chen-Yang e Huang, Lu-Wei. (2017). Minimizing total earliness and tardiness through unrelated parallel machine scheduling using distributed release time control. *Journal of Manufacturing Systems*, v. 42, p. 1 – 10. ISSN 0278-6125. doi: https://doi.org/10.1016/j.jmsy.2016.10.005. URL `http://www.sciencedirect.com/science/article/pii/S027861251630070X`.

Cheng, T.C. e Podolsky, S. (1996). *Just-in-Time Manufacturing:An introduction.* Springer Netherlands, 1st edição.

Clarke, G. e Wright, J. R. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, v. 12, p. 568–581.

Coelho, G. P.; de França, F. O. e Von Zuben, F. J. June(2011). A concentration-based artificial immune network for combinatorial optimization. *Proc. of the 2011 IEEE Congress of Evolutionary Computation (CEC)*, p. 1242–1249, June(2011).

Coelho, G. P. e Von Zuben, F. J. (2010). A concentration-based artificial immune network for continuous optimization. *Proc. of the 2010 IEEE Congress on Evolutionary Computation (CEC)*, volume 1, p. 108–115, (2010).

Coelho, Guilherme Palermo. *Redes Imunológicas Artificiais para Otimização em Espaços Contínuos: uma Proposta Baseada em Concentração de Anticorpos*. Tese de Doutorado, Universidade Estadual de Campinas, Campinas - SP, (2011).

Coelho, Guilherme Palermo e Von Zuben, Fernando J. (2006). omni-ainet: An immune-inspired approach for omni optimization. Bersini, Hugues e Carneiro, Jorge, editors, *Proceedings of the 5th International Conference on Artificial Immune Systems (ICARIS 2006)*, volume 4163 of *Lecture Notes in Computer Science*, p. 294–308. Springer-Verlag, (2006).

Coelho, Guilherme Palermo e Von Zuben, Fernando J. (2011). A concentration-based artificial immune network for multi-objective optimization. Takahashi, Ricardo H. C.; Deb, Kalyanmoy; Wanner, Elizabeth F. e Greco, Salvatore, editors, *Proceedings of the 6th International Conference on Evolutionary Multi-criterion Optimization*, EMO'11, p. 343–357, Berlin, Heidelberg. Springer-Verlag.

Corstjens, Jeroen; Dang, Nguyen; Depaire, Benoît; Caris, An e De Causmaecker, Patrick. Oct(2019). A combined approach for analysing heuristic algorithms. *Journal of Heuristics*, v. 25, n. 4, p. 591–628. doi: 10.1007/s10732-018-9388-7.

Cota, L. P.; aes, F. G. Guimar de Oliveira, F. B. e Souza, M. J. F. June(2017). An adaptive large neighborhood search with learning automata for the unrelated parallel machine scheduling problem. *Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC'17)*, p. 185–192, June(2017).

Şen, H. e Bülbül, K. (2015). A strong preemptive relaxation for weighted tardiness and earliness/tardiness problems on unrelated parallel machines. *INFORMS Journal on Computing*, v. 27, p. 135–150.

Cutello, V.; Nicosia, G.; Pavone, M. e Timmis, J. Feb(2007). An immune algorithm for protein structure prediction on lattice models. *IEEE Transactions on Evolutionary Computation*, v. 11, n. 1, p. 101–117. ISSN 1089-778X.

Dasgupta, D. (1998). *Artificial Immune Systems and Their Applications*. Springer-Verlag, Berlin, Heidelberg.

Dasgupta, Dipankar e Forrest, Stephanie. (1999). *An Anomaly Entection Algorithm Inspired by the Immune Syste*, p. 262–277. Springer-Verlag, Berlin, Heidelberg.

de Castro, L. N. e Timmis, J. (2002)a. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer Verlag.

de Castro, L. N. e Von Zuben, F. J. July(2000). The clonal selection algorithm with engineering applications. *Workshop Proceedings of the 2000 GECCO Workshop on Artificial Immune Systems and Their Applications*, p. 36–37, Las Vegas, USA.

de Castro, Leandro Nunes. *Engenharia imunológica: desenvolvimento e aplicação de ferramentas computacionais inspiradas em sistemas imunológicos artificiais*. Tese de Doutorado, Universidade Estadual de Campinas, Campinas - SP, (2001).

de Castro, Leandro Nunes e Timmis, J. (2002)b. An artificial immune network for multimodal function optimization. *Proc. of the 2002 Congress on Evolutionary Computation (CEC'02)*, volume 1, p. 699–704, (2002)b.

de Castro, Leandro Nunes e Zuben, Fernando J. Von. (2002). ainet: An artificial immune network for data analysis. *Data Mining: A Heuristic Approach*, p. 231–260. IGI Global.

de Castro, Pablo A. D. e Von Zuben, Fernando J. (2008). MOBAIS: A bayesian artificial immune system for multi-objective optimization. Bentley, Peter J.; Lee, Doheon e Jung, Sungwon, editors, *Proceedings of the 7th International Conference on Artificial Immune Systems (ICARIS'2008)*, volume 5132 of *Lecture Notes in Computer Science*, p. 48–59. Springer, (2008).

de Castro, Pablo Alberto Dalbem. *Sinergia entre Sistemas Imunológicos Artificiais e Modelos Gráficos Probabilísticos*. Tese de Doutorado, Universidade Estadual de Campinas, Campinas - SP, (2009).

de Castro, Pablo Alberto Dalbem e Zuben, Fernando J. Von. (2009). Bais: A bayesian artificial immune system for the effective handling of building blocks. *Information Sciences*, v. 179, n. 10, p. 1426 – 1440.

de França, Fabricio Olivetti; Von Zuben, Fernando J. e de Castro, Leandro Nunes. (2005). An artificial immune network for multimodal function optimization on dynamic environments. *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO'05)*, p. 289–296. ACM, (2005).

de França Filho, M. F. *GRASP e Busca Tabu aplicados a problemas de programacão de tarefas em máquinas paralelas (in portuguese)*. Tese de Doutorado, State University of Campinas, Campinas - SP, (2007).

de Paula, Mateus Rocha; Ravetti, Martín Gómez; Mateus, Geraldo Robson e Pardalos, Panos M. (2007). Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search. *IMA Journal of Management Mathematics*, v. 18, p. 101–115.

de Sousa, Janaína S.; de C. T. Gomes, Lalinka; Bezerra, George B.; de Castro, Leandro N. e Von Zuben, Fernando J. (2004). An immune-evolutionary algorithm for multiple rearrangements of gene expression data. *Genetic Programming and Evolvable Machines*, v. 5, n. 2, p. 157–179.

Diana, R. O. M.; de França Filho, M. F.; de Souza, S. R. e Amelia L. Silva, Maria. 10(2013). A clonal selection algorithm for makespan minimization on unrelated parallel machines with sequence dependent setup times. *Proceedings - 2013 Brazilian Conference on Intelligent Systems, BRACIS 2013*, p. 57–63, 10(2013). doi: 10.1109/BRACIS.2013.18.

Diana, R. O. M.; de França Filho, M. F.; de Souza, S. R. e de Almeida Vitor, J. F. (2015). An immune-inspired algorithm for an unrelated parallel machines' scheduling problem with sequence and machine dependent setup-times for makespan minimisation. *Neurocomputing*, v. 163, n. C, p. 94–105.

Diana, Rodney O. M.; de Souza, Sérgio R. e França Filho, Moacir F. (2018). A variable neighborhood descent as ILS local search to the minimization of the total weighted tardiness on unrelated parallel machines and sequence dependent setup times. *Electronic Notes in Discrete Mathematics*, v. 66, p. 191–198. ISSN 1571-0653. doi: https://doi.org/10.1016/j.endm.2018.03.025. 5th International Conference on Variable Neighborhood Search.

Diana, Rodney Oliveira Marinho; de Souza, Sérgio Ricardo; Wanner, Elizabeth F. e de França Filho, Moacir F. (2017). Hybrid metaheuristic for combinatorial optimization based on immune network for optimization and vns. *Proceedings of the Genetic and Evolutionary Computation Conference 2017*, GECCO '17, p. 251 – 258. ACM, (2017). ISBN 978-1-4503-4206-3. doi: 10.1145/3071178.3071269.

Diana, Rodney Oliveira Marinho e deSouza, Sérgio Ricardo. (2020). Analysis of variable neighborhood descent as a local search operator for total weighted tardiness problem on unrelated parallel machines. *Computers & Operations Research*, v. 117, p. 104886. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2020.104886.

Diana, Rodney Oliveira Marinho; de Souza, Sérgio Ricardo e Wanner, Elizabeth Fialho. (2021)a. A robust multi-response VNS-ainet approach for solving scheduling problems under unrelated parallel machines environments. *Expert Systems with Applications*, v. 182. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.115140.

Diana, Rodney Oliveira Marinho; de Souza, Sérgio Ricardo e Wanner, Elizabeth Fialho. (2021)b. Weighted earliness and tardiness minimization with idle times: Ananalysis of time window influence on scheduling over unrelated parallel machines environments. *preprint*, v. .

Doerr, Benjamin; Jansen, Thomas e Zarges, Christine. (2016). Artificial immune systems can beat evolutionary algorithms in combinatorial optimisation. *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference (GECCO'16)*, (2016). doi: 10.1145/2908812.2908892.

Douek, Daniel C.; Brenchley, Jason M.; Betts, Michael R.; Ambrozak, David R.; Hill, Brenna J.; Okamoto, Yukari; Casazza, Joseph P.; Kuruppu, Janaki; Kunstman, Kevin; Wolinsky, Steven; Grossman, Zvi; Dybul, Mark; Oxenius, Annette; Price, David A.; Connors, Mark e Koup, Richard A. 06(2002). HIV preferentially infects HIV-specific CD4+ T cells. *Nature*, v. 417, p. 95–98.

Duarte, Abraham; Sánchez-Oro, Jesús; Mladenović, Nenad e Todosijević, Raca. (2018). *Variable Neighborhood Descent*, p. 341–367. Springer International Publishing, Cham. doi: 10.1007/978-3-319-07124-4_9.

Ekici, Ali; Elyasi, Milad; ÖrsanÖzener, Okan e Sarıkaya, Merve Burcu. (2019). An application of unrelated parallel machine scheduling with sequence-dependent setups at vestel electronics. *Computers & Operations Research*, v. 111, p. 130–140. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2019.06.007.

Eom, D.-H.; Shin, H.-J.; Kwun, I.-H.; Shim, J.-K. e Kim, S.-S. (2002). Scheduling jobs on parallel machines with sequence-dependent family setup times. *The International*

Journal of Advanced Manufacturing Technology, v. 19, n. 12, p. 926 – 932. ISSN 1433-3015. doi: 10.1007/s001700200105.

Eroglu, Duygu Yilmaz; Ozmutlu, H. Cenk e Ozmutlu, Seda. (2014). Genetic algorithm with local search for the unrelated parallel machine scheduling problem with sequence-dependent set-up times. *International Journal of Production Research*, v. 52, n. 19, p. 5841–5856. doi: 10.1080/00207543.2014.920966.

Fanjul-Peyro, Luis; Perea, Federico e Ruiz, Rubén. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *European Journal of Operational Research*, v. 260, n. 2, p. 482–493. doi: https://doi.org/10.1016/j.ejor.2017.01.002.

Fanjul-Peyro, Luis; Ruiz, Ruben e Perea, Federico. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers and Operations Research*, v. 101, p. 173 – 182. ISSN 0305-0548. doi: 10.1016/j.cor.2018.07.007.

Fanjul-Peyro, Luis e Ruiz, Rubén. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, v. 207, n. 1, p. 55–69. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2010.03.030.

Fawcett, Chris e Hoos, Holger H. Aug(2016). Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, v. 22, n. 4, p. 431–458. doi: 10.1007/s10732-014-9275-9.

Fernandez-Viagas, Victor; Ruiz, Rubén e Framinan, Jose M. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European Journal of Operational Research*, v. 257, n. 3, p. 707–721. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2016.09.055.

Fleszar, Krzysztof; Charalambous, Christoforos e Hindi, Khalil S. Oct(2012). A variable neighborhood descent heuristic for the problem of makespan minimisation on unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing*, v. 23, n. 5, p. 1949–1958. ISSN 1572-8145. doi: 10.1007/s10845-011-0522-8. URL `https://doi.org/10.1007/s10845-011-0522-8`.

Fowler, John W.; Horng, Shwu Min e Cochran, Jeffery K. 9(2003). A hybridized genetic algorithm to solve parallel machine scheduling problems with sequence dependent setups. *International Journal of Industrial Engineering : Theory Applications and Practice*, v. 10, n. 3, p. 232–243.

French, Simon. (1982). *Sequencing and scheduling: an introduction to the mathematics of the job-shop*. Ellis Horwood, Chichester, USA.

Fukuda, T.; Mori, K. e Tsukiyama, M. (1999)a. *Parallel Search for Multi-Modal Function Optimization with Diversity and Learning of Immune Algorithm*, p. 210–220. Springer-Verlag, Berlin, Heidelberg.

Fukuda, Toyoo; Mori, Kazuyuki e Tsukiyama, Makoto. (1999)b. *Immunity-Based Management System for a Semiconductor Production Line**, p. 278–288. Springer-Verlag, Berlin, Heidelberg.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, v. 5, p. 553–549.

Gomes, Lalinka de Campos Teixeira. *Inteligência Computacional na Síntese de Metaheurísticas para Otimização Combinatória e Multimodal.* Tese de Doutorado, Universidade Estadual de Campinas, Campinas - SP, (2006).

González, Miguel A. e Vela, Camino R. (2015). An efficient memetic algorithm for total weighted tardiness minimization in a single machine with setups. *Applied Soft Computing*, v. 37, p. 506 – 518. ISSN 1568-4946. doi: 10.1016/j.asoc.2015.07.050.

Graham, R.L.; Lawler, E.L.; Lenstra, J.K. e Kan, A.H.G.Rinnooy. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, v. 5, p. 287 – 326. doi: http://dx.doi.org/10.1016/S0167-5060(08)70356-X.

Greensmith, Julie e Aickelin, Uwe. (2008). The deterministic dendritic cell algorithm. Bentley, Peter J.; Lee, Doheon e Jung, Sungwon, editors, *Proceedings of the 7th International Conference on Artificial Immune Systems*, ICARIS '08, p. 291–302, Berlin, Heidelberg. Springer-Verlag.

Guinet, A. (1991). Textile production systems: a succession of non-identical parallel processor shops. *Journal of the Operational Research Society*, v. 42, p. 655–671. doi: https://doi.org/10.1057/jors.1991.132.

Hariri, A. M. A. e Potts, C. N. (1991). Heuristics for scheduling unrelated parallel machines. *Computers and Operations Research*, v. 18, p. 323–331.

Hart, Emma; Ross, Peter e Nelson, Jeremy. May(1998). Producing robust schedules via an artificial immune system. *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation (CEC'98)*, p. 464–469, May(1998).

He, Pengfei; Li, Jing e Wang, Xin. (2018). Wheat harvest schedule model for agricultural machinery cooperatives considering fragmental farmlands. *Computers and Electronics in Agriculture*, v. 145, p. 226–234. ISSN 0168-1699. doi: https://doi.org/10.1016/j.compag.2017.12.042.

Helal, R. M. e Al-Salem, A. (2006). A tabu search algorithm to minimize the makespan for unrelated parallel machines scheduling problem with setup times. *International Journal of Operations Research*, v. 3, p. 182–192.

Hendel, Yann e Sourd, Francis. (2007). An improved earliness–tardiness timing algorithm. *Computers & Operations Research*, v. 34, n. 10, p. 2931 – 2938. doi: https://doi.org/10.1016/j.cor.2005.11.004.

Hoffmann, G.W.; Müller, Sybille e Kohler, Heinz. (2012). Towards an hiv vaccine based on immune network theory. *Current Trends in Immunology*, v. 13, p. 69–79.

Horoba, Christian; Jansen, Thomas e Zarges, Christine. (2009). Maximal age in randomized search heuristics with aging. *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO'09)*, p. 803–810, New York, NY, USA. ACM.

Hunt, J. E. e Cooke, D. E. (1996). Learning using an artificial immune system. *Journal of Network and Computer Applications*, v. 19, p. 189–212.

Hunt, J. E. e Fellows, A. (1996). Introducing an immune response into a cbr system for data mining. *Proc. of the 1996 BCS ESG-96 Conference*, volume 1, p. 35–42, (1996).

Ibáñez, M. L.; Lacoste, J. D.; Cáceres, L. P.; Birattari, M. e Stützle, T. (2016). The iRACE package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, v. 3, p. 43–58.

Janeway, Charles A.; Travers, Paul; Walport, Mark e Shlomchik, Mark J. (2005). *Immunobiology: The Immune System in Health and Disease*. Garland Science, 6th edição.

Janiak, Adam; Janiak, Władysław A.; Krysiak, Tomasz e Kwiatkowski, Tomasz. (2015). A survey on scheduling problems with due windows. *European Journal of Operational Research*, v. 242, n. 2, p. 347 – 357. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2014.09.043. URL `http://www.sciencedirect.com/science/article/pii/S0377221714007826`.

Jansen, Thomas e Zarges, Christine. (2009). Comparing different aging operators. Andrews, Paul S.; Timmis, Jon; Owens, Nick D. L.; Aickelin, Uwe; Hart, Emma; Hone, Andrew e Tyrrell, Andy M., editors, *Proceedings of the 8th International Conference of Artificial Immune Systems*, p. 95–108, Berlin, Heidelberg. Springer-Verlag.

Jansen, Thomas e Zarges, Christine. (2010). On the benefits of aging and the importance of details. Hart, Emma; McEwan, Chris; Timmis, Jon e Hone, Andy, editors, *Proceedings of the 9th International Conference of Artificial Immune Systems*, p. 61–74, Berlin, Heidelberg. Springer-Verlag.

Jansen, Thomas e Zarges, Christine. (2011)a. Analyzing different variants of immune inspired somatic contiguous hypermutations. *Theoretical Computer Science*, v. 412, n. 6, p. 517–533.

Jansen, Thomas e Zarges, Christine. (2011)b. On benefits and drawbacks of aging strategies for randomized search heuristics. *Theoretical Computer Science*, v. 412, n. 6, p. 543–559.

Jerne, Niels Kaj. (1974). Towards a network theory of the immune system. *Annales d'immunologie*, v. 125, n. 1-2, p. 373–389.

Kampke, E. H. Metaheurísticas para problema de programação de tarefas em máquinas paralelas com tempos de preparação dependentes da sequência de recursos. Dissertação de Mestrado, Universidade Federal de Viçosa, Viçosa - MG, (2010).

Kelsey, J. e Timmis, J. (2003). Immune inspired somatic contiguous hypermutation for function optimisation. *Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation: PartI*, volume 1 of *GECCO'03*, p. 207–218, (2003).

Kendall, Graham; Bai, Ruibin; Błazewicz, Jacek; De Causmaecker, Patrick; Gendreau, Michel; John, Robert; Li, Jiawei; McCollum, Barry; Pesch, Erwin; Qu, Rong; Sabar, Nasser; Berghe, Greet Vanden e Yee, Angelina. Apr(2016). Good laboratory practice for optimization research. *Journal of the Operational Research Society*, v. 67, n. 4, p. 676–689. doi: https://doi.org/10.1057/jors.2015.77.

Kim, Dong-Won; Kim, Kyong-Hee; Jang, Wooseung e Chen, F Frank. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, v. 18, n. 3, p. 223 – 231. doi: http://doi.org/10.1016/S0736-5845(02)00013-3.

Kirkpatrick, S.; Gellat, D. C. e Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, v. 220, p. 671–680.

Korte, Bernhard e Vygen, Jens. (2008). *NP-Completeness. In:Combinatorial Optimization: Theory and Algorithms*, Capítulo 15, p. 359–392. Springer Berlin Heidelberg, Berlin, Heidelberg, 2th edição.

Kramer, Arthur e Subramanian, Anand. Feb(2019). A unified heuristic and an annotated bibliography for a large class of earliness–tardiness scheduling problems. *Journal of Scheduling*, v. 22, n. 1, p. 21–57. doi: 10.1007/s10951-017-0549-6.

Kurz, M. E. e Askin, R. G. (2001). Heuristic scheduling of parallel machines with sequence-dependent setup times. *International Journal of Production Research*, v. 39, p. 3747–3769.

Lala, Algenti; Kolici, Vladi; Xhafa, Fatos; Herrero, Xavier e Barolli, Admir. (2015). On local vs. population-based heuristics for ground station scheduling. *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, p. 267–275, (2015). doi: 10.1109/CISIS.2015.40.

Land, A. H. e Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, v. 28, n. 3, p. 497–520.

Larranaga, Pedro e Lozano, Jose A. (2002). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer US., New York, NY, USA.

Lee, Cheng-Hsiung; Liao, Ching-Jong e Chao, Chien-Wen. (2014). Unrelated parallel machine scheduling with dedicated machines and common deadline. *Computers and Industrial Engineering*, v. 74, p. 161 – 168. doi: http://doi.org/10.1016/j.cie.2014.05.012.

Lee, Jae-Ho; Yu, Jae-Min e Lee, Dong-Ho. (2013). A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: minimizing total tardiness. *The International Journal of Advanced Manufacturing Technology*, v. 69, n. 9, p. 2081 – 2089. doi: 10.1007/s00170-013-5192-6.

Lee, Young Hoon e Pinedo, Michael. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, v. 100, n. 3, p. 464 – 474. doi: http://dx.doi.org/10.1016/S0377-2217(95)00376-2.

Lee, Youngy Hoonh; Bhaskaran, Kumar e Pinedo, Michael. (1997). A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions*, v. 29, n. 1, p. 45–52. doi: 10.1080/07408179708966311.

Lenstra, J.K.; Kan, A.H.G. Rinnooy e Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, v. 1, p. 343 – 362. ISSN 0167-5060. doi: http://dx.doi.org/10.1016/S0167-5060(08)70743-X.

Li, Tzu Yi. *Heuristic algorithm to minimize total weighted tardiness on the unrelated parallel machine with sequence dependent setup and future ready time*. PhD thesis, The University of Wisconsin-Milwaukee, (2018).

Liao, Ching-Jong; Lee, Cheng-Hsiung e Tsai, Hsing-Tzu. (2016). Scheduling with multi-attribute set-up times on unrelated parallel machines. *International Journal of Production Research*, v. 54, n. 16, p. 4839 – 4853. doi: 10.1080/00207543.2015.1118574.

Lin, Chi-Wei; Lin, Yang-Kuei e Hsieh, Han-Ting. (2013). Ant colony optimization for unrelated parallel machine scheduling. *The International Journal of Advanced Manufacturing Technology*, v. 67, n. 1, p. 35–45. doi: 10.1007/s00170-013-4766-7.

Lin, S.-W. e Ying, K.-C. (2014). Abc-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times. *Computers and Operations Research*, v. 51, p. 172–181.

Lin, Shih-Wei; Lu, Chung-Cheng e Ying, Kuo-Ching. (2011). Minimization of total tardiness on unrelated parallel machines with sequence- and machine-dependent setup times under due date constraints. *The International Journal of Advanced Manufacturing Technology*, v. 53, n. 1, p. 353 – 361. doi: 10.1007/s00170-010-2824-y.

Lin, Yang-Kuei e Hsieh, Feng-Yu. (2014). Unrelated parallel machine scheduling with setup times and ready times. *International Journal of Production Research*, v. 52, n. 4, p. 1200 – 1214. doi: 10.1080/00207543.2013.848305.

Liu, Lin; yuGu, Han e gengXi, Yu. (2007). Robust and stable scheduling of a single machine with random machine breakdowns. *The International Journal of Advanced Manufacturing Technology*, v. 31, n. 7, p. 645–654. doi: https://doi.org/10.1007/s00170-005-0237-0.

Logendran, Rasaratnam; McDonell, Brent e Smucker, Byran. (2007). Scheduling unrelated parallel machines with sequence-dependent setups. *Computers and Operations Research*, v. 34, n. 11, p. 3420 – 3438. doi: http://doi.org/10.1016/j.cor.2006.02.006.

Lopes, Manuel J. Pereira e deCarvalho, J.M. Valério. (2007). A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times. *European Journal of Operational Research*, v. 176, n. 3, p. 1508 – 1527. doi: http://doi.org/10.1016/j.ejor.2005.11.001.

Lourenço, Helena R.; Martin, Olivier C. e Stützle, Thomas. (2003). *Iterated Local Search*, p. 320–353. Springer US, Boston, MA. ISBN 978-0-306-48056-0. doi: 10.1007/0-306-48056-5_11. URL `https://doi.org/10.1007/0-306-48056-5_11`.

Lü, Zhipeng; Hao, Jin-Kao e Glover, Fred. Apr(2011). Neighborhood analysis: a case study on a curriculum-based course timetabling. *Journal of Heuristics*, v. 17, n. 2, p. 97–118. ISSN 1572-9397. doi: 10.1007/s10732-010-9128-0.

Martí, Rafael; Pardalos, Panos M. e Resende, Mauricio G. (2018). *Handbook of Heuristics*. Springer International Publishing, 1th edition edição.

May, R.C. e Machesky, L.M. (2001). Phagocytosis and the actin cytoskeleton. *Journal of Cell Science*, v. 114, n. 6, p. 1061–1077.

McNaughton, Robert. October(1959). Scheduling with deadlines and loss functions. *Manage. Sci.*, v. 6, n. 1, p. 1–12. doi: 10.1287/mnsc.6.1.1.

Melián, B.; Moreno-Pérez, J. A. e Moreno-Vega, J. M. (2003). Metaheuristics: A global view. *Revista Iberoamericana de Inteligencia Artificial*, v. 19, p. 7–28.

Merelli, Emanuela; Pettini, Marco e Rasetti, Mario. (2015). Topology driven modeling: the IS metaphor. *Natural Computing*, v. 14, n. 3, p. 421–430.

Miller, B. L. e Shaw, M. J. May(1996). Genetic algorithms with dynamic niche sharing for multimodal function optimization. *Proceedings of IEEE International Conference on Evolutionary Computation (ICEC'96)*, p. 786–791, May(1996).

Mjirda, Anis; Todosijević, Raca; Hanafi, Saïd; Hansen, Pierre e Mladenović, Nenad. (2017). Sequential variable neighborhood descent variants: an empirical study on the traveling salesman problem. *International Transactions in Operational Research*, v. 24, n. 3, p. 615–633. ISSN 1475-3995. doi: 10.1111/itor.12282. URL `http://dx.doi.org/10.1111/itor.12282`.

Mladenovic, N. e Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, v. 24, p. 1097–1100.

Nogueira, J. P. C. M.; Arroyo, J. E. C.; Villadiego, H. M. M. e Gonçalves, L. B. (2014). Hybrid grasp heuristics to solve an unrelated parallel machine scheduling problem with earliness and tardiness penalties. *Electronic Notes in Theoretical Computer Science*, v. 302, p. 53 – 72.

Oliveto, Pietro S. e Sudholt, Dirk. (2014). On the runtime analysis of stochastic ageing mechanisms. *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO'14)*, p. 113–120, New York, NY, USA. ACM.

Oprea, Mihaela L. *Antibody repertoires and pathogen recognition: The role of germline diversity and somatic hypermutation*. PhD thesis, The University of New Mexico, Albuquerque - New Mexico, (1999).

Parker, R. G.; Deane, R. H. e Holmes, R. A. (1977). On the use of a vehicle routing algorithm for the parallel processor problem with sequence dependent changeover costs. *A I I E Transactions*, v. 9, n. 2, p. 155–160. doi: 10.1080/05695557708975137.

Pei, Jun; Wei, Jinling; Liao, Baoyu; Liu, Xinbao e Pardalos, Panos M. Feb(2019). Two-agent scheduling on bounded parallel-batching machines with an aging effect of job-position-dependent. *Annals of Operations Research*, v. 294, p. 191–223. doi: 10.1007/s10479-019-03160-y.

Peng, Kunkun; Pan, Quan-Ke; Gao, Liang; Li, Xinyu; Das, Swagatam e Zhang, Biao. (2019). A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling. *Swarm and Evolutionary Computation*, v. 45, p. 92 – 112. ISSN 2210-6502. doi: 10.1016/j.swevo.2019.01.002.

Perez-Gonzalez, Paz; Fernandez-Viagas, Victor; Zamora García, Miguel e Framinan, Jose M. (2019). Constructive heuristics for the unrelated parallel machines scheduling problem with machine eligibility and setup times. *Computers & Industrial Engineering*, v. 131, p. 131 – 145. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2019.03.034. URL `http://www.sciencedirect.com/science/article/pii/S0360835219301706`.

Pinedo, M. (2008). *Scheduling: Theory, Algorithms and Systems*. Prentice Hall, 3th edition edição.

Polyakovskiy, S. e M'Hallah, R. (2014). A multi-agent system for the weighted earliness tardiness parallel machine problem. *Computers & Operations Research*, v. 44, p. 115 – 136. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2013.10.013. URL `http://www.sciencedirect.com/science/article/pii/S0305054813003080`.

Potts, C. N e Wassenhove, L. N Van. (1982). A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, v. 1, n. 5, p. 177 – 181. ISSN 0167-6377. doi: https://doi.org/10.1016/0167-6377(82)90035-9.

Rabadi, G.; Moraga, R. e Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, v. 17, p. 85–97.

Rambod, Mahdi e Rezaeian, Javad. (2014). Robust meta-heuristics implementation for unrelated parallel machines scheduling problem with rework processes and machine eligibility restrictions. *Computers and Industrial Engineering*, v. 77, p. 15 – 28. ISSN 0360-8352. doi: http://dx.doi.org/10.1016/j.cie.2014.09.006.

Randall, P. A. e Kurz, M. E. (2007). Effectiveness of adaptive crossover procedures for a genetic algorithm to schedule unrelated parallel machines with setups. *International Journal of Operations Research*, v. 4, p. 1–10.

Ravetch, Jeffrey V. e Bolland, Silvia. (2001). Igg fc receptors. *Annual Review of Immunology*, v. 19, n. 1, p. 275–290.

Rocha, Pedro Leite; Ravetti, Martín Gómez; Mateus, Geraldo Robson e Pardalos, Panos M. (2008). Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Computers and Operations Research*, v. 35, p. 1250–1264.

Rosa, Bruno Ferreira; Souza, Marcone Jamilson Freitas; de Souza, Sérgio Ricardo; de França Filho, Moacir Felizardo; Ales, Zacharie e Michelon, Philippe Yves Paul. (2017). Algorithms for job scheduling problems with distinct time windows and general earliness/tardiness penalties. *Computers and Operations Research*, v. 81, n. Supplement C, p. 203 – 215. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2016.12.024.

Ruiz, Rubén e Andrés-Romano, Carlos. (2011). Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, v. 57, n. 5, p. 777 – 794. doi: 10.1007/s00170-011-3318-2.

Ruiz, Rubén e Stützle, Thomas. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, v. 177, n. 3, p. 2033–2049. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2005.12.009.

Salehi, Mir Saber; Rezaeian, Javad e Mohamadian, Hossein. (2020). Scheduling parallel machine problem under general effects of deterioration and learning with past-sequence-dependent setup time: heuristic and meta-heuristic approaches. *Soft Computing*, v. 24, n. 2, p. 1335–1355. doi: https://doi.org/10.1007/s00500-019-03970-z.

Santos, Haroldo G.; Toffolo, Túlio A.M.; Silva, Cristiano L.T.F. e Vanden Berghe, Greet. (2019). Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem. *International Transactions in Operational Research*, v. 26, p. 707–724. ISSN 1475-3995. doi: 10.1111/itor.12316.

Schwefel, Hans-Paul e Rudolph, Günter. (1995). Contemporary evolution strategies. Morán, Federico; Moreno, Alvaro; Merelo, Juan Julián e Chacón, Pablo, editors, *Proceedings of the Third European Conference on Artificial Life*, volume 929 of *Lecture Notes in Computer Science*, p. 893–907. Springer, (1995).

Sen, Subir e Kothari, D.P. (1998). Optimal thermal generating unit commitment: a review. *International Journal of Electrical Power & Energy Systems*, v. 20, n. 7, p. 443–451. doi: https://doi.org/10.1016/S0142-0615(98)00013-1.

Shirvani, Nargess; Ruiz, Rubén e Shadrokh, Shahram. (2014). Cyclic scheduling of perishable products in parallel machine with release dates, due dates and deadlines. *International Journal of Production Economics*, v. 156, p. 1–12. ISSN 0925-5273. doi: https://doi.org/10.1016/j.ijpe.2014.04.013. URL `http://www.sciencedirect.com/science/article/pii/S0925527314001285`.

Silvestrin, Paulo Vitor e Ritt, Marcus. (2017). An iterated tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, v. 81, p. 192 – 202. doi: 10.1016/j.cor.2016.12.023.

Skutella, Martin e Verschae, José. (2016). Robust polynomial-time approximation schemes for parallel machine scheduling with job arrivals and departures. *Mathematics of Operations Research*, v. 41, n. 3, p. 991–1021. doi: https://doi.org/10.1287/moor.2015.0765.

Stützle, T. G. *Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications*. PhD thesis, Darmstadt University of Technology, Department of Computer Science, (1998).

Subramanian, Anand; Battarra, Maria e Potts, Chris N. (2014). An iterated local search heuristic for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, v. 52, n. 9, p. 2729–2742. doi: 10.1080/00207543.2014.883472.

Subramanian, Anand e Farias, Katyanne. (2017). Efficient local search limitation strategy for single machine total weighted tardiness scheduling with sequence-dependent setup times. *Computers & Operations Research*, v. 79, p. 190 – 206. doi: 10.1016/j.cor.2016.10.008.

Taguchi, G. (1986). *Introduction to Quality Engineering*. UNIPUB/Kraus International. White Plains.

Taguchi, G. (1987). *System of Experimental Design: Engineering Methods to Optimize Quality and Minimize Cost*. UNIPUB/Kraus International. White Plains.

Thepphakorn, Thatchai; Pongcharoen, Pupong e Hicks, Chris. (2014). An ant colony based timetabling tool. *International Journal of Production Economics*, v. 149, p. 131 – 144. ISSN 0925-5273. doi: http://dx.doi.org/10.1016/j.ijpe.2013.04.026. The Economics of Industrial Production.

Tran, Tony T.; Araujo, Arthur e Beck, J. Christopher. (2016). Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS Journal on Computing*, v. 28, n. 1, p. 83–95. doi: 10.1287/ijoc.2015.0666.

Đurasević, Marko; Jakobović, Domagoj e Knežević, Karlo. (2016). Adaptive scheduling on unrelated machines with genetic programming. *Applied Soft Computing*, v. 48, p. 419–430. doi: https://doi.org/10.1016/j.asoc.2016.07.025.

Vallada, E. e Ruiz, R. (2011)a. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, v. 211, p. 612–622.

Vallada, Eva e Ruiz, Rubén. (2011)b. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, v. 211, n. 3, p. 612 – 622. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2011.01.011. URL `http://www.sciencedirect.com/science/article/pii/S0377221711000142`.

Vallada, Eva e Ruiz, Rubén. (2012). *Scheduling Unrelated Parallel Machines with Sequence Dependent Setup Times and Weighted Earliness–Tardiness Minimization*, p. 67–90. Springer New York, New York, NY.

Wang, L.; Wang, S. e Zheng, X. July(2016). A hybrid estimation of distribution algorithm for unrelated parallel machine scheduling with sequence-dependent setup times. *IEEE/CAA Journal of Automatica Sinica*, v. 3, n. 3, p. 235–246. ISSN 2329-9266. doi: 10.1109/JAS.2016.7508797.

Watanabe, Yuji; Ishiguro, Akio e Uchikawa, Yoshiki. (1999). *Decentralized Behavior Arbitration Mechanism for Autonomous Mobile Robot Using Immune Network*, p. 187–209. Springer-Verlag, Berlin, Heidelberg.

Weise, Thomas; Wu, Yuezhong; Chiong, Raymond; Tang, Ke e Lässig, Jörg. (2016). Global versus local search: the impact of population sizes on evolutionary algorithm performance. *Journal of Global Optimization*, v. 66, p. 511–534.

Whitley, Darrell; Rana, Soraya e Heckendorn, Robert. (1998). The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, v. 7, n. 1, p. 33–47.

Wilcoxon, Frank. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, v. 1, n. 6, p. 80–83. doi: 10.2307/3001968.

Wu, Chin-Chia; Azzouz, Ameni; Chung, I-Hong; Lin, Win-Chin e Said, Lamjed Ben. (2019). A two-stage three-machine assembly scheduling problem with deterioration effect. *International Journal of Production Research*, v. 57, n. 21, p. 6634–6647. doi: https://doi.org/10.1080/00207543.2019.1570378.

Wu, Chin-Chia; Bai, Danyu; Azzouz, Ameni; Chung, I.-Hong; Cheng, Shuenn-Ren; Jhwueng, Dwueng-Chwuan; Lin, Win-Chin e Said, Lamjed Ben. (2020)a. A branch-and-bound algorithm and four metaheuristics for minimizing total completion time for a two-stage assembly flow-shop scheduling problem with learning consideration. *Engineering Optimization*, v. 52, n. 6, p. 1009–1036. doi: ttps://doi.org/10.1080/0305215X.2019.1632303.

Wu, Chin-Chia; Gupta, Jatinder N. D.; Cheng, Shuenn-Ren; Lin, Bertrand M. T.; Yip, Siu-Hung e Lin, Win-Chin. (2020)b. Robust scheduling for a two-stage assembly shop with scenario-dependent processing times. *International Journal of Production Research*, v. 0, n. 0, p. 1–16. doi: https://doi.org/10.1080/00207543.2020.1778208.

Wu, Chin-Chia; Lin, Win-Chin; Xin-Gong, Zhang; Bai, Dan-Yu; Tsai, Yung-Wei; Ren, Tao e Cheng, Shuenn-Ren. (2020)c. Cloud theory-based simulated annealing for a single-machine past sequence setup scheduling with scenario-dependent processing times. *Complex & Intelligent Systems*, v. . doi: https://doi.org/10.1007/s40747-020-00196-7.

Yang, Jian e Yu, Gang. (2002). On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, v. 6, p. 17?33. doi: https://doi.org/10.1023/A:1013333232691.

Yano, Candace Arai e Kim, Yeong-Dae. (1991). Algorithms for a class of single-machine weighted tardiness and earliness problems. *European Journal of Operational Research*, v. 52, n. 2, p. 167 – 178. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(91)90078-A.

Yepes-Borrero, Juan C.; Villa, Fulgencia; Perea, Federico e Caballero-Villalobos, Juan Pablo. (2020). Grasp algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. *Expert Systems with Applications*, v. 141, p. 112959. doi: https://doi.org/10.1016/j.eswa.2019.112959.

Ying, Kuo-Ching; Lee, Zne-Jung e Lin, Shih-Wei. (2012). Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing*, v. 23, p. 1795–1803. doi: https://doi.org/10.1007/s10845-010-0483-3.

Ying, Kuo-Ching e Lin, Shih Wei. (2012). Unrelated parallel machines scheduling with sequence and machine dependent setups times and due dates constraints. *International Journal of Innovative Computing, Information and Control*, v. 8, n. 5, p. 3279 – 3297.

Zandieh, M.; Amiri, M.; Vahdani, B. e Soltani, R. (2009). A robust parameter design for multi-response problems. *Journal of Computational and Applied Mathematics*, v. 230, n. 2, p. 463 – 476. ISSN 0377-0427. doi: http://dx.doi.org/10.1016/j.cam.2008.12.019.

Zeidi, Javad Rezaeian e MohammadHosseini, Samir. Dec(2015). Scheduling unrelated parallel machines with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, v. 81, n. 9, p. 1487–1496. ISSN 1433-3015. doi: 10.1007/s00170-015-7215-y. URL `https://doi.org/10.1007/s00170-015-7215-y`.

Zeng, Zhi-zhong; Yu, Xin-guo; He, Kun e Fu, Zhang-hua. (2018). Adaptive tabu search and variable neighborhood descent for packing unequal circles into a square. *Applied Soft Computing*, v. 65, p. 196 – 213. doi: 10.1016/j.asoc.2017.11.051.

Zhao, Fuqing; Qin, Shuo; Zhang, Yi; Ma, Weimin; Zhang, Chuck e Song, Houbin. (2019). A two-stage differential biogeography-based optimization algorithm and its performance analysis. *Expert Systems with Applications*, v. 115, p. 329 – 345. doi: 10.1016/j.eswa.2018.08.012.

Zhu, Zhiwei e Heady, Ronald B. (2000). Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach. *Computers and Industrial Engineering*, v. 38, n. 2, p. 297 – 305. doi: http://doi.org/10.1016/S0360-8352(00)00048-6.