RICARDO DE BRITO DAMM

Métodos mono e multiobjetivo para
o problema de escalonamento de técnicos de campo

São Paulo
2016

RICARDO DE BRITO DAMM

Métodos mono e multiobjetivo para
o problema de escalonamento de técnicos de campo

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Doutor em Ciências

Área de Concentração:
Engenharia de Produção

Orientadora:
Profa. Dra. Débora Pretti Ronconi

São Paulo
2016

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 23 de maio de 2016

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

Damm, Ricardo de Brito

Métodos mono e multiobjetivo para o problema de escalonamento de técnicos de campo / R. B. Damm – versão corr. – São Paulo, 2016.
199 p.

Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo.
Departamento de Engenharia de Produção.

1. Pesquisa Operacional. 2. Roteamento de técnicos. 3. Heurística.
4. Algoritmos Genéticos. 5. Janelas de tempo. I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Produção. II.t.

*Ao meu pai,
de quem herdei
a paixão pela pesquisa.*

Agradecimentos

Agradeço inicialmente a Deus Pai, pela beleza do mundo criado e por ter-nos dado a oportunidade de desvendar os seus mistérios através da ciência.

Em segundo lugar, à minha orientadora, professora Débora Pretti Ronconi, pela grande dedicação, amizade e estímulo ao longo de todos esses anos. A busca da perfeição é uma arte que tentei aprender em cada uma de nossas reuniões e que espero levar para o início desta caminhada no mundo científico.

À minha mãe (*in memoriam*), por todos os anos de dedicação aos filhos e por sempre continuar muito presente em todos os momentos de minha vida. Aos meus familiares, que me acompanham com entusiasmo e carinho, ou que nasceram ao longo destes anos: Débora, Irene, Thomas, Aline, Gabriela, Victor, Margarida, Pedro, João, José Luiz, Márcia, Regina... E todos os meus muitos outros familiares e amigos nesta vida, o meu profundo agradecimento!

Aos professores do Departamento, grandes mestres que me acompanharam desde a graduação e com quem tanto aprendi, agradeço de coração. Em especial, o prof. Miguel César Santoro, a profa. Celma de Oliveira Ribeiro, a profa. Linda Lee Ho e o prof. Hugo Yoshizaki. À profa. Vitória Pureza, por todas as sugestões feitas a essa pesquisa. E também aos professores do Instituto de Matemática e Estatística, nossos vizinhos, que com tanto rigor científico ensinam: prof. Paulo Feofiloff, prof. Ernesto Birgin e prof. Luis Carlos de Castro Santos.

Agradeço imensamente também à Lídia, por toda a atenção e ajuda desde os anos de graduação. À Ana, Olívia, Sara, Marlene, Rose, Cristina, ao Osni e todos os que participam da vida acadêmica do Departamento.

Aos meus amigos: Everton Luiz de Melo, João Manguino, Guilherme Barroso Mainieri, Luís Roberto Henriques, Rafael de Freitas Lemos, Hugo Watanuki, Oscar Adorno, Oberlan Romão, Rafael Ribeiro.

E ao CNPq e à USP, pelo financiamento da pesquisa.

Muitas coisas não tentamos fazer
por parecerem difíceis;
entretanto, são difíceis
por não tentarmos fazê-las.
(Sêneca)

RESUMO

Um tema pouco estudado na literatura, mas frequentemente encontrado por empresas prestadoras de serviço, é o Problema de Escalonamento de Técnicos de Campo (*Field Technician Scheduling Problem*). O problema consiste em associar um número de tarefas – em diversos locais, com diferentes prioridades e com janelas de tempo – a uma quantidade de técnicos – com diferentes horários de expediente e com habilidades distintas – que saem no início do horário de trabalho da sede da empresa, para onde devem retornar antes do fim do expediente. Cada tarefa é atendida por um único técnico. Esse problema é estudado neste trabalho.

A primeira parte do trabalho apresenta um modelo de programação linear inteira mista (PLIM) e, dada a complexidade do problema, heurísticas construtivas e meta-heurísticas foram desenvolvidas. Na função objetivo, procura-se principalmente maximizar o número ponderado de tarefas executadas em um dia de trabalho, de acordo com as suas prioridades. Em linhas gerais, as heurísticas construtivas ordenam as tarefas de acordo com um critério pré-estabelecido e, em seguida, designam cada uma a um dos técnicos capazes de realizá-la sem violar as restrições do problema. Tendo em conta o bom desempenho obtido em outros problemas semelhantes, foi adotado um Algoritmo Genético denominado *Biased Random-Key Genetic Algorithms* (BRKGA), que utiliza chaves aleatórias para codificar e decodificar as soluções. Codificadores e decodificadores adaptados ao problema foram desenvolvidos e testes computacionais são apresentados. As soluções obtidas em problemas de pequenas dimensões são comparadas com as soluções ótimas conhecidas e, para aprimorar a avaliação do desempenho nas instâncias médias e grandes, quatro procedimentos para obter limitantes superiores foram propostos. Testes computacionais foram realizados em 1040 instâncias. O BRKGA encontrou 99% das 238 soluções ótimas conhecidas e, nas 720 instâncias de dimensões médias e grandes, ficou em média a 3,8% dos limitantes superiores. As heurísticas construtivas superaram uma heurística construtiva da literatura em 90% das instâncias.

A segunda parte do trabalho apresenta uma nova abordagem para o Problema de Escalonamento de Técnicos de Campo: um modelo biobjetivo, onde uma segunda função objetivo buscará que as tarefas prioritárias sejam realizadas o mais cedo possível. Uma versão multiobjetivo do BRKGA foi desenvolvida, considerando diversas estratégias para classificar a população do algoritmo e escolher as melhores soluções (estratégias de elitismo). Codificadores e decodificadores foram criados para o problema multiobjetivo. Os resultados computacionais obtidos são comparados com os resultados de um Algoritmo Genético conhecido na literatura, o *Nondominated Sorting Genetic Algorithm II* (NSGA II). Para instâncias de pequenas dimensões, os resultados da meta-heurística

proposta também são comparados com a fronteira ótima de Pareto de 234 instâncias, obtidas por enumeração completa. Em média, o BRKGA multiobjetivo encontrou 94% das soluções da fronteira ótima de Pareto e, nas instâncias médias e grandes, superou o desempenho do NSGA-II nas medidas de avaliação adotadas (porcentagem de soluções eficientes, hipervolume, indicador epsilon e cobertura).

Palavras-chave: heurísticas, programação de técnicos, roteirização, agendamento de tarefas, janelas de tempo, otimização multiobjetivo, *Biased Random-Key Genetic Algorithm*.

ABSTRACT

An important topic in service companies, but little studied until now, is the field technician scheduling problem. In this problem, technicians have to execute a set of jobs or service tasks. Technicians have different skills and working hours. Tasks are in different locations within a city, with different time windows, priorities, and processing times. Each task is executed by only one technician. This problem is addressed in this thesis.

The first part of the research presents the mixed integer linear programming model (MILP) and, due to the complexity of this problem, constructive heuristics and meta-heuristics were proposed. The objective function is to maximize the sum of the weighted performed tasks in a day, based on the priority of tasks. In general terms, in the proposed constructive heuristics, jobs are ordered according to a criterion and, after that, tasks are assigned to technicians without violating constraints. A Genetic Algorithm (the Biased Random Key Genetic Algorithm – BRKGA) is applied to the problem, based on its success in similar problems; the BRKGA uses random keys and a decoder transforms each chromosome of the Genetic Algorithm into a feasible solution of the problem. Decoders and encoders adapted to the problem were developed and computational tests are presented. A comparison between the solutions of the heuristic methods and optimal solutions values was also conducted for small instances and, to analyze medium and large instances, four upper bound models were proposed. Computational experiments with 1040 instances were carried out. The BRKGA reached 99% of the 238 optimal solutions and, for 720 medium and large instances, the average upper bound gap was 3.8%. Constructive heuristics overcame a heuristic of the literature in 90% of the instances.

The second part of this research presents a new approach of the Field Technician Scheduling Problem: a multiobjective model, with a second objective function to execute the priority tasks as soon as possible. A multiobjective BRKGA was developed, with different strategies to classify the Genetic Algorithm population and to select the elite solutions (elite strategies). Decoders and encoders were developed for the multiobjective problem too. The results were compared with a known Genetic Algorithm, the Nondominated Sorting Genetic Algorithm II (NSGA II). For 234 small instances, the results were compared with the Pareto optimal solutions, obtained by complete enumeration. On average, the BRKGA found 94% of the Pareto optimal solutions and, for 720 medium and large instances, outperformed the NSGA-II by means of the measures adopted (percentage of efficient solutions, hypervolume, epsilon and coverage).

Keywords: Heuristics, routing and scheduling technicians, scheduling tasks, time windows, multiobjective optimization, Biased Random Key Genetic Algorithm.

LISTA DE FIGURAS

1.1	Exemplo com 6 tarefas e 2 técnicos: a) solução heurística; b) solução ótima.	9
2.1	Solução da heurística XC para o exemplo	26
2.2	Solução ótima do exemplo	27
2.3	Solução da heurística NT para o exemplo	30
2.4	Solução da heurística EST para o exemplo	33
2.5	Solução da heurística TS para o exemplo	36
2.6	Solução da heurística Cluster para o exemplo	39
3.1	Solução do exemplo: a) solução decodificada; b) solução ótima.	46
3.2	BRKGA: Transição da geração K para a geração K+1	48
5.1	Desempenho do BRKGA com a função similaridade e a elite tradicional: a) BRKGA-A; b) BRKGA-STT.	83
6.1	Exemplo do conceito de dominância	104
6.2	Exemplo de conjunto ótimo estrito de Pareto ou soluções eficientes	105
6.3	Exemplo de conjunto ótimo fraco de Pareto	106
6.4	Soluções eficientes de três métodos heurísticos	115
6.5	Hipervolume dos métodos heurísticos com relação ao ponto de referência (4,0; 1,0)	116
6.6	Conjuntos referência e soluções eficientes de cada heurística, com as solu- ções do conjunto referência multiplicadas por ϵ . As soluções que definem o valor de ϵ estão indicadas em cada uma das figuras.	117
7.1	Exemplo com 10 tarefas e 2 técnicos: a) solução do modelo mono-objetivo ($\max f_1$); b) solução do modelo multiobjetivo (método ϵ -restrito: $\max f_2$ s.a. $f_1 \geq 45,0$); c) solução do modelo multiobjetivo (método ϵ -restrito: $\max f_2$ s.a. $f_1 \geq 51,0$)	126
8.1	Distância de <i>crowding</i>	130
8.2	População P e a densidade de vizinhança de uma solução	132

8.3	Espaço das funções objetivo e sua divisão em hipercubos, com o número de soluções em cada célula.	133
9.1	Fronteira ótima de Pareto de uma instância com 16 tarefas e 2 técnicos. .	141
9.2	Soluções de referência e soluções eficientes encontradas pelo BRKGA-A com diferentes estratégias: a) somente multiobjetivo com elite tradicional; b) somente multiobjetivo com elite formada pela função similaridade; c) mono e multiobjetivo com elite tradicional; d) mono e multiobjetivo com elite formada pela função similaridade.	149
9.3	Soluções de referência e soluções eficientes encontradas pelo BRKGA-STT com diferentes estratégias: a) somente multiobjetivo com elite tradicional; b) somente multiobjetivo com elite formada pela função similaridade; c) mono e multiobjetivo com elite tradicional; d) mono e multiobjetivo com elite formada pela função similaridade.	151
9.4	Soluções de referência e soluções eficientes encontradas pelo BRKGA-Balanceado com diferentes estratégias: a) somente multiobjetivo com elite tradicional; b) somente multiobjetivo com elite formada pela função similaridade; c) mono e multiobjetivo com elite tradicional; d) mono e multiobjetivo com elite formada pela função similaridade.	153
9.5	Soluções de referência e soluções eficientes encontradas pelo: a) BRKGA-A; b) BRKGA-STT; c) BRKGA-Balanceado.	157
9.6	Soluções de referência e soluções eficientes encontradas por: a) BRKGA-Balanceado; b) NSGA-II.	161
C.1	Soluções da fronteira de Pareto para o exemplo	184
C.2	Solução 1 do exemplo	186
C.3	Solução 6 do exemplo	187
C.4	Solução 2 do exemplo	188

LISTA DE TABELAS

1.1	Parâmetros do exemplo: a) prioridades, tempos de execução e janelas de tempo das tarefas. b) Tempo de deslocamento c_{ij} (em minutos) entre as tarefas e a sede da empresa.	8
1.2	Informações das tarefas executadas por cada técnico na solução da heurística construtiva: a) Técnico 1. b) Técnico 2.	10
1.3	Informações das tarefas executadas por cada técnico na solução ótima: a) Técnico 1. b) Técnico 2.	10
1.4	Diferentes funções objetivo e parâmetros do problema encontrados na literatura	14
2.1	Parâmetros do exemplo: a) prioridades, tempos de execução e janelas de tempo das tarefas. b) Tempo de deslocamento c_{ij} (em minutos) entre as tarefas e a sede da empresa.	25
2.2	Informações das tarefas executadas por cada técnico na solução da heurística XC: a) Técnico 1. b) Técnico 2.	26
2.3	Informações das tarefas executadas por cada técnico na solução ótima: a) Técnico 1. b) Técnico 2.	28
2.4	Informações das tarefas executadas por cada técnico na solução da heurística NT: a) Técnico 1. b) Técnico 2.	30
2.5	Informações das tarefas executadas por cada técnico na solução da heurística EST: a) Técnico 1. b) Técnico 2.	34
2.6	Informações das tarefas executadas por cada técnico na solução da heurística TS: a) Técnico 1. b) Técnico 2.	36
2.7	Informações das tarefas executadas por cada técnico na solução da heurística <i>Cluster</i> : a) Técnico 1. b) Técnico 2.	40
3.1	Informações das tarefas executadas por cada técnico na solução obtida pelo decodificador: a) Técnico 1. b) Técnico 2.	47
3.2	Chaves aleatórias na décima geração. Os cromossomos estão ordenados e os 9 melhores formam o conjunto elite	51
3.3	Similaridade entre os cromossomos	52
3.4	Conjunto elite selecionado com similaridade de até 55%	53

3.5	Habilidades de 7 técnicos para 10 tarefas	58
3.6	Decodificador para o exemplo de 7 técnicos e 10 tarefas	58
5.1	Dimensão das instâncias geradas	70
5.2	Diferença percentual (<i>Diff%</i>) média em cada caso	71
5.3	Comparação entre os modelos de limitantes superiores e os valores ótimos	73
5.4	Comparação entre os modelos propostos de limitantes superiores e o UB_0 .	74
5.5	RDI médio das heurísticas construtivas	77
5.6	NIMD das heurísticas construtivas	77
5.7	Porcentagem (%) de melhoria em relação a heurística XC	78
5.8	<i>Gap</i> de otimalidade das heurísticas construtivas	79
5.9	Média e Desvio Padrão (σ) do GAP_{UB} das heurísticas construtivas	80
5.10	Parâmetros calibrados para cada versão do BRKGA.	81
5.11	Número de gerações necessárias para atingir a convergência e tempo de execução (critério de parada) para cada caso.	81
5.12	RDI médio das quatro versões do BRKGA	84
5.13	NIMD das quatro versões do BRKGA	84
5.14	Melhoria média (em %) de cada versão do BRKGA em relação à melhor heurística construtiva (<i>Cluster</i>)	85
5.15	Comparação entre os resultados do BRKGA e os valores ótimos conhecidos.	86
5.16	Média, Mínimo (Min.) e Desvio Padrão (σ) do GAP_{UB} dos métodos heu- rísticos e das soluções ótimas conhecidas.	87
5.17	Valor médio do GAP_{UB} de cada versão do BRKGA para as diferentes distribuições geográficas de tarefas. a) instâncias pequenas (casos 1 a 4). b) instâncias médias e grandes (casos 5 a 13).	89
5.18	Valor médio do GAP_{UB} de cada versão do BRKGA para instâncias com diferentes tamanhos de janelas de tempo: a) instâncias pequenas (casos 1 a 4). b) instâncias médias e grandes (casos 5 a 13).	89
5.19	Média do GAP_{UB} de uma execução (1 exec.); média dos valores mínimos (Min.), média geral (Med.) e média dos valores máximos (Max.) do GAP_{UB} de 5 execuções de 8 instâncias para cada caso.	91
5.20	Desempenho do BRKGA-STT no conjunto de instâncias <i>complete</i> pro- postas por Kovacs et al. (2012)	94
5.21	Desempenho do BRKGA-STT no conjunto de instâncias <i>reduced</i> propos- tas por Kovacs et al. (2012)	95
5.22	Resultados do BRKGA-STT em 12 instâncias do PRVJT propostas por Solomon	97
6.1	Soluções eficientes de três métodos heurísticos	115
6.2	Medidas do conjunto de cobertura	118

6.3	Índices de medida para o exemplo	118
7.1	Parâmetros do exemplo: a) prioridades, tempos de execução e janelas de tempo das tarefas. b) Tempo de deslocamento c_{ij} (em minutos) entre as tarefas e a sede da empresa.	124
7.2	Momento de início da execução das tarefas (t) e respectivas folgas (γ) de três soluções do exemplo	125
9.1	Instâncias com fronteira ótima de Pareto conhecida.	140
9.2	Pareto ótimo de uma instância com 16 tarefas e 2 técnicos.	141
9.3	Parâmetros calibrados para cada uma das versões do BRKGA multiobjetivo	143
9.4	Número de gerações necessárias para a convergência do BRKGA multi-objetivo e critério de parada (<i>limit_time</i>) para cada caso.	143
9.5	Desempenho do BRKGA-A para os diferentes métodos de ordenação: a) instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.	146
9.6	Desempenho do BRKGA-STT para os diferentes métodos de ordenação: a) instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.	146
9.7	Desempenho do BRKGA-Balanceado para os diferentes métodos de ordenação: a) instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.	147
9.8	Desempenho do BRKGA-A com distância de <i>crowding</i> para diferentes modos de construir o conjunto elite: a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.	148
9.9	Desempenho do BRKGA-STT com distância de <i>crowding</i> para diferentes modos de construir o conjunto elite: a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.	150
9.10	Desempenho do BRKGA-Balanceado com densidade de vizinhança para diferentes modos de construir o conjunto elite: a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.	152

9.11	Desempenho das três versões do BRKGA: a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.	155
9.12	Desempenho das três versões do BRKGA em relação ao conjunto de referência: a) Casos 5 a 7. b) Casos 8 a 10. c) Casos 11 a 13.	155
9.13	Análise de sensibilidade do Hipervolume	156
9.14	Desempenho do BRKGA e do NSGA-II: a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.	159
9.15	Desempenho do BRKGA e do NSGA-II, ambos com distância de <i>crowding</i> : a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.	160
C.1	Soluções da fronteira de Pareto para o exemplo	184
C.2	Parâmetros do exemplo: a) prioridades, tempos de execução e janelas de tempo das tarefas. b) Tempo de deslocamento c_{ij} (em minutos) entre as tarefas e a sede da empresa.	185
C.3	Informações das tarefas executadas por cada técnico na solução 1: a) Técnico 1. b) Técnico 2.	186
C.4	Informações das tarefas executadas por cada técnico na solução 6: a) Técnico 1. b) Técnico 2.	187
C.5	Informações das tarefas executadas por cada técnico na solução 2: a) Técnico 1. b) Técnico 2.	188
D.1	Desempenho do BRKGA-Balanceado para determinação do tamanho do conjunto elite: a) instâncias pequenas (casos 1 a 4) em relação ao conjunto de referência. b) instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.	190

LISTA DE ABREVIATURAS E SIGLAS

ABC Artificial Bee Colony

AG Algoritmos Genéticos

ALNS Adaptive Large Neighborhood Search

B&P Branch-and-Price

BRKGA Biased Random-Key Genetic Algorithm

BT Busca Tabu

EST Heurística construtiva menor instante de início (*Earliest start time*)

FLS Fast Local Search

GLS Guided Local Search

GRASP Greedy Randomized Adaptive Search Procedure

NIMD Número de Instâncias com Melhor Desempenho

NT Heurística construtiva técnico mais próximo (*Nearest technician*)

PETC Problema de Escalonamento de Técnicos de Campo

PLIM Programação Linear Inteira Mista

PMM Problema de Múltiplas Mochilas

PMPI Problema de Máquinas Paralelas Idênticas

PRVJT Problema de Roteamento de Veículos com Janelas de Tempo

PSO Particle Swarm Optimization

RDI Relação de Desempenho Individual

RKGA Random-Key Genetic Algorithm

SA Simulated Annealing

STT Heurística construtiva menor tempo de viagem (*Shortest travel time*)

tar. Tarefa

tec. Técnico

TS Heurística construtiva habilidade dos técnicos (*Technicians' skill*)

XC Heurística proposta por Xu e Chiu

SUMÁRIO

Introdução	1
I Otimização Mono-objetivo	5
1 O Problema de Escalonamento de Técnicos	7
1.1 Definição do problema	7
1.2 Exemplo	8
1.3 Revisão bibliográfica	11
1.4 Modelo de Programação Linear Inteira Mista	16
1.5 Complexidade computacional	19
1.5.1 Redução ao problema de múltiplas mochilas	19
2 Heurísticas construtivas	23
2.1 Heurística proposta por Xu e Chiu (XC)	24
2.2 Técnico mais próximo (<i>Nearest technician</i> - NT)	28
2.3 Menor tempo de viagem (<i>Shortest travel time</i> - STT)	31
2.4 Menor instante de início das tarefas prioritárias (<i>Earliest start time of priority tasks</i> - EST)	32
2.5 Habilidade dos técnicos (<i>Technicians' skill</i> - TS)	34
2.6 <i>Cluster</i>	36
2.7 Outros métodos	40
3 Meta-heurísticas	41
3.1 Algoritmos Genéticos	42
3.2 Biased Random-Key Genetic Algorithm	44
3.2.1 Revisão Bibliográfica	44
3.2.2 Descrição do BRKGA	45
3.3 Diversificando a população elite	50
3.4 Decodificadores	55
3.4.1 Decodificando com a heurística Menor Tempo de Viagem (<i>STT</i>) .	55
3.4.2 Decodificando com a heurística <i>Cluster</i>	55

3.4.3	Decodificador Aleatório (A)	56
3.4.4	Decodificador Busca em Vizinhança (BV)	58
3.4.5	Outros decodificadores	60
4	Limitantes Superiores (<i>Upper Bound</i>)	61
4.1	Modelo zero: sem tempos de deslocamentos e janelas de tempo	61
4.2	Modelo 1: reintroduzindo tempos de deslocamento e janelas de tempo	63
4.3	Modelo 2: excluindo combinações infactíveis	65
4.4	Modelo 3: relaxando as variáveis inteiras do modelo de PLIM	67
5	Experimentos Computacionais	69
5.1	Instâncias geradas	69
5.2	Soluções do modelo de PLIM	71
5.3	Limitantes superiores	72
5.4	Heurísticas construtivas	75
5.5	BRKGA	79
5.5.1	Calibragem dos parâmetros do BRKGA	79
5.5.2	Desempenho do BRKGA	83
5.5.3	Desempenho do BRKGA em instâncias da literatura	92
5.6	Análise dos resultados da Parte I	97
II	Otimização Multiobjetivo	99
6	Otimização Multiobjetivo: fundamentos teóricos	101
6.1	Fundamentos de Otimização Multiobjetivo	102
6.1.1	Definição de otimalidade	103
6.1.2	Métodos Tradicionais	106
6.1.3	Meta-heurísticas	110
6.2	Medidas de avaliação das soluções	112
6.2.1	Porcentagem de soluções eficientes	112
6.2.2	Hipervolume	113
6.2.3	Indicador epsilon	113
6.2.4	Medida do conjunto de cobertura	114
6.2.5	Exemplo	114
7	Modelo Multiobjetivo	121
7.1	Proposta de uma nova função objetivo	121
7.2	Modelo PLIM Multiobjetivo	123
7.3	Exemplo	125

8	BRKGA Multiobjetivo	129
8.1	Métodos de classificação das soluções	130
8.1.1	Fronteiras de Pareto	130
8.1.2	Distância de <i>Crowding</i>	130
8.1.3	Densidade de vizinhança	131
8.1.4	Densidade baseada em células	133
8.2	Estratégias para formar o conjunto elite	134
8.3	Decodificadores	135
8.3.1	Aleatório	135
8.3.2	Menor tempo de viagem - <i>STT</i>	136
8.3.3	Balanceado	137
9	Resultados Computacionais	139
9.1	Fronteiras ótimas de Pareto e características das instâncias	139
9.2	Calibragem dos parâmetros do BRKGA	142
9.3	Composição do conjunto elite	144
9.3.1	Classificação das soluções	144
9.3.2	Estratégias adicionais	147
9.4	Comparação dos decodificadores	154
9.5	Comparação com um AG da literatura	158
9.6	Conclusões do capítulo	162
10	Conclusões e futuros trabalhos	163
III	Apêndices	165
A	Relação do PETC com outros problemas NP-difíceis	166
A.1	Extensão do problema de Roteamento de Veículo com Janelas de Tempo	166
A.2	Redução ao problema de máquinas paralelas	171
B	Pseudocódigo da heurística construtiva STT	179
C	Exemplo multi em problemas com janelas de tempo apertadas	183
D	Calibração dos parâmetros do BRKGA multiobjetivo	189
	Referências Bibliográficas	191

Introdução

Janelas de tempo estão cada vez mais presentes no cotidiano das pessoas que vivem em cidades médias ou grandes. O aumento do custo de vida nas metrópoles, o trânsito intenso e o tempo diário dentro dos meios de transportes, a maior presença das mulheres no mercado de trabalho e a redução do número de empregados domésticos são fatores que estão reduzindo a disponibilidade de as pessoas receberem, por exemplo, a entrega de produtos comprados.

Conforme divulgado recentemente pela mídia,

Esperar pela chegada de uma mercadoria comprada na internet ou numa loja comum tem sido motivo de queixas frequentes dos consumidores brasileiros. Por causa da falta de um horário determinado, é como se eles ficassem presos em casa. [...] Segundo um dos maiores sites de reclamações, em 2015 aumentou em 10% o número de queixas por causa do atraso na entrega. [...] Pouca gente sabe e pouca empresa respeita, mas existem leis municipais e estaduais regulando a entrega: em várias regiões do país, é obrigatório marcar a data e determinar um turno específico para a entrega: manhã, tarde, ou noite. A defensora pública Larissa Davidovich diz que o consumidor pode ter o tempo perdido recompensado. “Dependendo do caso, ele tem sim o direito inclusive a ser ressarcido. Existem teorias, já modernas, que dizem que o nosso tempo, o tempo do consumidor, vale dinheiro”, afirmou Larissa Davidovich. (*www.g1.globo.com*)

De fato, a entrega de produtos é um dos principais problemas das empresas varejistas. Segundo levantamento de dados feito no cadastro de reclamações fundamentadas de 2014 da Fundação Procon-SP (Proteção e Defesa do Consumidor), no Estado de São Paulo, entre as empresas varejistas¹ que se encontram na lista das empresas que mais receberam reclamações em 2014, em média 28% das reclamações referiram-se a não entrega ou demora na entrega dos produtos. Em algumas empresas, essa porcentagem chegou a 44% do total das reclamações (*www.procon.sp.gov.br*).

Na cidade de São Paulo, as leis de rodízio de veículos e de restrições de circulação

¹ <http://sistemas.procon.sp.gov.br/reclamacao_fundamentada/lista_grupos.php>

de caminhões dificultam ainda mais o trabalho das empresas (veja, por exemplo, o decreto 49.487 de 12/05/2008 e o decreto 49.800 de 23/07/2008, que regulamenta a Lei 14.751/08 – *www3.prefeitura.sp.gov.br*). No levantamento de dados citado no parágrafo anterior, considerando as reclamações apenas para a capital paulista, para as mesmas empresas, em média 33% das reclamações referiram-se a não entrega ou demora na entrega dos produtos e, em algumas empresas, esse valor foi superior a 50%.

Os serviços de manutenção ou instalação de um equipamento, por terem janelas de tempo, também fazem parte desta realidade, com o agravamento de que o tempo de permanência dos funcionários na residência do cliente é maior, e, por isso muitos clientes desejam recebê-los em um dia e horário específico. Um desejo e também um direito, uma vez que o Governo do Estado de São Paulo aprovou as leis 13.747 (de 7 de outubro de 2009) e 14.951 (de 6 de fevereiro de 2013) que estabelecem que, tanto na entrega de produtos quanto na prestação de serviços, as empresas fixem uma data e um turno (manhã, tarde ou noite) para a realização dos serviços ou entrega dos produtos, sem ônus para o consumidor e que sejam comunicados previamente ao cliente² (*www.procon.sp.gov.br*). Para as empresas, tudo isso faz dos serviços de manutenção um maior desafio: como atender as restrições de prazos, horários, prioridades, especialidades dos técnicos, etc., e ao mesmo tempo construir suas competências: qualidade, confiabilidade, idoneidade, cortesia, credibilidade, comunicação e compreensão da realidade do cliente (DAVIS; CHASE; AQUILANO, 2001).

O problema abordado neste trabalho – conhecido como problema de escalonamento de técnicos de campo – começou a receber maior atenção da literatura nos últimos anos, mas ainda foi pouco estudado.

O problema consiste em fazer um planejamento diário do trabalho dos técnicos de uma empresa, de modo a maximizar o número ponderado de tarefas atendidas, ou seja, procurando que as tarefas mais prioritárias sejam realizadas. Os técnicos têm diferentes horários de trabalho e encontram-se na empresa no início do seu dia, de onde sairão para realizar diversas tarefas e para onde deverão retornar até o fim do expediente. Não são permitidas horas-extras.

Os serviços ou as tarefas estão localizados em diferentes locais de uma região e têm janelas de tempo, ou seja, os clientes determinam um intervalo de horas em que o técnico deverá realizar o serviço. A cada tarefa é atribuída uma prioridade, representando a

² <<http://www.procon.sp.gov.br/texto.asp?id=2924>>

importância do cliente, o grau de urgência do serviço ou o tempo em que a chamada está pendente. Tempos determinísticos são estipulados para a realização do serviço e para os deslocamentos.

Uma última característica do problema é que os técnicos não necessariamente realizam todas as tarefas: cada um tem um conjunto de habilidades que permitem ou o impedem de realizar cada tarefa.

Trata-se de um problema de otimização combinatória, com aspectos semelhantes a outros problemas, como o problema de roteamento de veículos (onde os caminhões representariam os técnicos e o tempo de entrega dos produtos, o tempo de execução dos serviços), o problema de máquinas paralelas não relacionadas (em que cada técnico representaria uma máquina e os tempos de deslocamentos, os *setup*), ou o problema de mochilas múltiplas (onde as mochilas fariam o papel dos técnicos e o tamanho dos itens, o tempo de processamento das tarefas). Assim como todos esses problemas NP-difíceis, para o problema de escalonamento de técnicos de campo não são conhecidos métodos exatos que o resolva em tempo polinomial e, por isso, métodos heurísticos também foram aplicados na sua resolução.

O problema foi estudado em dois contextos: mono e multiobjetivo. Após investigar o modelo de Programação Linear Inteira Mista, heurísticas construtivas foram desenvolvidas para cada um dos contextos. Por serem simples de serem implementadas, as heurísticas construtivas requerem pequeno esforço computacional e podem obter bons resultados em muitos problemas (RONCONI, 2004). Em seguida, meta-heurísticas, que há duas décadas têm apresentado bons resultados em problemas combinatórios (OSMAN; LAPORTE, 1996; BLUM; ROLI, 2003), também foram desenvolvidas para o problema; concretamente, um Algoritmo Genético adaptado ao problema de escalonamento de técnicos de campo.

Desse modo, o trabalho está estruturado em duas partes.

A primeira parte apresenta o enfoque mono-objetivo do problema. O Capítulo 1 descreve o modelo de Programação Linear Inteira Mista, apresenta a revisão bibliográfica e a relação do problema com outros problemas combinatórios conhecidos da literatura, possibilitando discutir sua complexidade computacional. Nos Capítulos 2 e 3 são apresentados, respectivamente, as heurísticas construtivas desenvolvidas e o Algoritmo Genético escolhido (*Biased Random-Key Genetic Algorithms* – BRKGA). Para

avaliar o desempenho dos algoritmos, limitantes superiores são propostos no Capítulo 4. A primeira parte é encerrada com o Capítulo 5, que relata os experimentos computacionais: como as instâncias foram geradas, a análise dos resultados obtidos pelos métodos exatos, pelas heurísticas construtivas e pelas meta-heurísticas. Foram feitas comparações com as soluções ótimas obtidas para os problemas de pequenas dimensões e com os limitantes superiores.

Na segunda parte do trabalho encontra-se um estudo do problema de escalonamento de técnicos de campo no contexto multiobjetivo. Inicialmente são apresentados os fundamentos teóricos de otimização multiobjetivo (Capítulo 6). A seguir, uma nova função objetivo é proposta, através da qual se procura que as tarefas mais urgentes ou os clientes mais importantes sejam priorizados, atendendo-os no início da sua janela de tempo (Capítulo 7). Uma versão multiobjetivo do Algoritmo Genético adotado é proposta (Capítulo 8). Resultados computacionais são apresentados com comparações com um Algoritmo Genético multiobjetivo conhecido na literatura (Capítulo 9).

O último capítulo apresenta as principais conclusões da tese e possíveis futuros trabalhos.

Parte I

Otimização Mono-objetivo

Capítulo 1

O Problema de Escalonamento de Técnicos

A primeira seção deste capítulo descreve o Problema de Escalonamento de Técnicos de Campo (PETC) e, em seguida, são apresentados um exemplo gráfico (seção 1.2), a revisão bibliográfica (seção 1.3) e o modelo de Programação Linear Inteira Mista (seção 1.4). O capítulo termina com a discussão da complexidade computacional do problema e da semelhança com outros problemas clássicos de otimização combinatória (seção 1.5).

1.1 Definição do problema

Neste trabalho o PETC é modelado da seguinte maneira: seja $J = \{1, \dots, n\}$ um conjunto de tarefas ou serviços e $K = \{1, \dots, m\}$ um conjunto de técnicos disponíveis para executá-los. Cada tarefa $i \in J$ tem uma prioridade w_i e a sua execução tem duração de tempo estimada em p_i , que deve ser iniciada e concluída dentro de um intervalo de horário específico $[e_i, l_i]$. As tarefas estão distribuídas em locais dentro de uma região e c_{ij} é o tempo de deslocamento do local da tarefa i para o local da tarefa j , com $i, j \in J \cup \{0\}$, onde 0 é uma tarefa falsa que representa a origem ou a sede da empresa.

Cada técnico $k \in K$ tem seu horário de trabalho diário definido por $[a_k, b_k]$ e deve deixar a origem após o início e retornar até o fim do seu horário de trabalho. A habilidade de um técnico k executar uma tarefa i é dada por s_{ik} , que pode ser 0 ou 1, onde 1 indica ser apto e 0, incapaz de executá-la.

Neste trabalho é adotada a hipótese de que a quantidade de tarefas (n) é grande e de que, em princípio, é possível que os técnicos (m) não realizem todas as tarefas.

Desse modo, procura-se principalmente conseguir atender os clientes mais prioritários e, portanto, o principal objetivo é maximizar a soma dos valores das prioridades das tarefas executadas. Um objetivo secundário também é adotado, onde a disponibilidade dos técnicos após o retorno à sede da empresa deve ser maximizado; com esse objetivo secundário procura-se indiretamente reduzir o tempo de deslocamento total dos técnicos e os tempos de espera para o início da execução de uma tarefa.

1.2 Exemplo

Para ilustrar o problema, a seguir encontra-se um exemplo com seis tarefas e dois técnicos. O primeiro técnico trabalha entre 7h e 16h e pode executar todas as tarefas, com exceção da tarefa 6; o segundo técnico trabalha das 9h às 18h e pode executar as tarefas 1, 4, 5 e 6. As tarefas estão dispersas em uma região. A Tabela 1.1 apresenta os parâmetros do problema. As prioridades das tarefas variam de 1 (baixa) a 10 (alta).

Considere agora uma heurística gulosa que define a programação das tarefas de acordo com as prioridades (começando pelas mais prioritárias até as menos prioritárias) e alocando-as ao técnico que estiver mais próximo. Seguindo essa diretriz, a tarefa

Tabela 1.1 – Parâmetros do exemplo: a) prioridades, tempos de execução e janelas de tempo das tarefas. b) Tempo de deslocamento c_{ij} (em minutos) entre as tarefas e a sede da empresa.

(a)						
Tarefa	1	2	3	4	5	6
Prioridade (w)	10	1	5	9	4	8
Tempo de execução (p)	75min	30min	115min	105min	70min	90min
Início da janela de tempo (e)	8h	12h30min	10h	13h	11h30min	13h
Fim da janela de tempo (l)	11h	14h30min	13h	16h	15h	16h

(b)							
c_{ij}	Emp.	1	2	3	4	5	6
Emp.	0	42	35	74	36	78	65
1	42	0	16	50	49	63	82
2	35	16	0	41	33	51	66
3	74	50	41	0	50	18	68
4	36	49	33	50	0	47	34
5	78	63	51	18	47	0	56
6	65	82	66	68	34	56	0

Figura 1.1 – Exemplo com 6 tarefas e 2 técnicos: a) solução heurística; b) solução ótima.

Fonte: elaborado pelo autor.

1 (a mais prioritária) é a primeira tarefa programada pela heurística. Como ambos técnicos têm habilidade para executá-la e estão equidistantes do local dessa tarefa (pois no início do dia estão na sede da empresa), o técnico com menor número de identificação é escolhido. O técnico 1 inicia seu dia de trabalho na sede da empresa às 7h e, portanto, consegue chegar no local da tarefa 1 às 7h42min; a janela de tempo dessa tarefa começa 8h e, assim, o técnico aguardará 18min para iniciar a sua execução. A segunda tarefa prioritária é a tarefa 4, que é associada ao técnico 2, uma vez que o local em que ele está (a sede da empresa) é mais perto da tarefa 4 do que o local em que está o técnico 1 (tarefa 1). O técnico chega ao cliente às 9h36min, mas aguarda até o início da janela de tempo da tarefa (às 13h) para poder iniciar a sua execução. A tarefa 6 é a terceira mais prioritária e pode ser executada apenas pelo técnico 2. No entanto, as tarefas 4 e 6 têm janelas de tempo idênticas e, considerando os tempos de execução e de deslocamento, não é possível o mesmo técnico realizá-las. Portanto, a tarefa 6 não poderá ser executada nesse dia. As demais tarefas serão programadas da mesma forma pela heurística, resultando na solução apresentada na Figura 1.1(a). A Tabela 1.2 apresenta os horários em que os técnicos chegam nos locais das tarefas, os tempos de espera para iniciar a execução de uma tarefa (devido às janelas de tempo), e os horários em que a execução de cada tarefa é iniciada e concluída. Note que o tempo total de espera dos técnicos antes de realizarem as tarefas é alto (3h42min) e a disponibilidade

Tabela 1.2 – Informações das tarefas executadas por cada técnico na solução da heurística construtiva: a) Técnico 1. b) Técnico 2.

(a)				
Tarefa	1	3	5	Empresa
Horário de chegada do técnico	7h42min	10h05min	12h18min	14h46min
Tempo de espera	18min	-	-	-
Início da execução	8h00min	10h05min	12h18min	-
Fim da execução	9h15min	12h00min	13h28min	-

(b)		
Tarefa	4	Empresa
Horário de chegada do técnico	9h36min	15h21min
Tempo de espera	3h24min	-
Início da execução	13h00min	-
Fim da execução	14h55min	-

dos técnicos após o retorno à empresa também é grande (3h43min); como há duas tarefas que não são realizadas, pode-se intuir que há soluções melhores do que a gerada por esta heurística gulosa.

De fato, na solução ótima deste exemplo todas as tarefas são executadas, como pode ser visto na Figura 1.1(b). A Tabela 1.3 apresenta os horários em que os técnicos chegam nos locais das tarefas, os tempos de espera para iniciar a execução de uma tarefa (devido às janelas de tempo), e os horários em que a execução de cada tarefa é iniciada e

Tabela 1.3 – Informações das tarefas executadas por cada técnico na solução ótima: a) Técnico 1. b) Técnico 2.

(a)				
Tarefa	3	2	4	Empresa
Horário de chegada do técnico	8h14min	12h36min	13h39min	16h00min
Tempo de espera	1h46min	-	-	-
Início da execução	10h00min	12h36min	13h39min	-
Fim da execução	11h55min	13h06min	15h24min	-

(b)				
Tarefa	1	5	6	Empresa
Horário de chegada do técnico	9h42min	12h00min	14h06min	16h41min
Tempo de espera	-	-	-	-
Início da execução	9h42min	12h00min	14h06min	-
Fim da execução	10h57min	13h10min	15h36min	-

concluída. Para que a execução de todas as tarefas fosse possível, o tempo de espera dos técnicos antes das tarefas (1h46min) e o tempo disponível total após o retorno à sede da empresa (1h19min) diminuíram, enquanto que o tempo de deslocamento dos técnicos aumentou de 4h20min (solução heurística) para 6h50min (solução ótima). Com isso, o valor do objetivo principal (a soma das prioridades das tarefas executadas) passou de 28 (na solução heurística) para 37 com a solução ótima (ou seja, 32% maior).

Através deste pequeno exemplo, pode-se observar que o número significativo de parâmetros do problema exige abordagens heurísticas mais sofisticadas, onde simultaneamente sejam consideradas as janelas de tempo, as prioridades das tarefas, os tempos de deslocamento e as habilidades dos técnicos, de modo a encontrar soluções de melhor qualidade. Com o objetivo de contribuir para esse estudo, diferentes abordagens heurísticas para o PETC são propostas nos próximos capítulos.

1.3 Revisão bibliográfica

Pela pesquisa realizada, constatou-se que poucos trabalhos até o momento abordaram o PETC. Entre os primeiros autores, estão Tsang e Voudouris (1997) e Xu e Chiu (2001). Depois de 2005, o problema recebeu atenção crescente na literatura, principalmente a partir de 2007, quando a Sociedade de Pesquisa Operacional da França (*French Operational Research Society*) e a *France Telecom* lançaram esse problema como um desafio. Diferentes funções objetivo são encontradas na literatura: maximizar o número de tarefas atendidas, minimizar o instante de término da última tarefa (*makespan*), minimizar custos (deslocamentos, horas extras, terceirização ou tarefas não realizadas), minimizar o número de técnicos necessários, etc. Com relação aos parâmetros, além dos já apresentados na definição do problema (Seção 1.1), o PETC também pode ter algumas outras características, como horas extras, possibilidade de terceirização, formação de um grupo de técnicos para executar as tarefas, etc. A seguir encontra-se uma breve descrição de cada um dos principais trabalhos da literatura.

Hashimoto et al. (2011) e Cordeau et al. (2010) estudaram o PETC minimizando o instante de término do processamento de grupos de tarefas. Ambas as pesquisas foram motivadas por uma proposta da *France Telecom*, que desejava conciliar um crescimento significativo no número de serviços com uma quantidade limitada de técnicos. Os serviços foram divididos em três grupos de prioridades e a função objetivo a ser minimizada

foi a soma ponderada pela prioridade do instante de término do último serviço realizado de cada grupo. Há uma relação de precedência entre os serviços. Os técnicos têm diferentes habilidades e para executar as tarefas é necessário formar equipes. As tarefas não alocadas aos técnicos são terceirizadas. Não há janelas de tempo; custos e tempos de deslocamentos também não são considerados. Hashimoto et al. (2011) aplicaram a meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) para a resolução deste problema; já Cordeau et al. (2010) desenvolveram heurísticas construtivas e adaptaram a meta-heurística *Adaptive Large Neighborhood Search* (ALNS) para o PETC.

Kovacs et al. (2012) estudaram um problema similar, com o objetivo de minimizar o custo total de deslocamento e os custos de terceirização (tarefas não atendidas). As tarefas têm janelas de tempo, mas não prioridades. Os técnicos não têm janelas de tempo e podem ser agrupados em times quando necessário. Os autores também aplicaram a meta-heurística ALNS nesse problema.

Tsang e Voudouris (1997) também procuraram minimizar custos. Concretamente, a função objetivo do trabalho considerou o custo total de deslocamento, o custo de horas extras e o custo (ou penalidade) das tarefas não executadas. Trata-se de um problema real da *British Telecom*. As tarefas não têm prioridades e, ao invés de especificar janelas de tempo em horas, três possibilidades foram adotadas: tarefas que devem ser executadas de manhã, de tarde ou indiferente. Os autores desenvolveram duas heurísticas para o problema: a *Fast Local Search* (FLS) e a *Guided Local Search* (GLS).

Já Tang; Miller-Hooks e Tomastik (2007) estudaram um problema de programação semanal de técnicos para manutenções periódicas em equipamentos localizados em diversos pontos de uma região. Trata-se de um caso real da *United Technologies Corporation* (UTC), com o objetivo de maximizar o retorno gerado pelas tarefas atendidas (nesse contexto, retorno significa o valor associado à urgência de cada tarefa e não o seu lucro financeiro). Não há janelas de tempo, nem as tarefas têm prioridade. Os autores utilizaram a meta-heurística Busca Tabu.

Pillac; Gueret e Medaglia (2012) analisaram a semelhança do problema de escalonamento de técnicos com o problema de roteamento de veículos com janelas de tempo (PRVJT), onde a função objetivo de ambos é minimizar o tempo total de deslocamento. Os autores desenvolveram um algoritmo ALNS paralelo e validaram o algoritmo com as

instâncias de Solomon (1987) para o PRVJT.

Um diferente objetivo foi proposto por Li; Lim e Rodrigues (2005): minimizar o número total de trabalhadores e o tempo total de deslocamentos no porto de Cingapura. Esse trabalho também exigiu a formação de grupos de trabalhadores para realizar os diferentes tipos de tarefas, que permanecem juntos ao longo de todo o dia de trabalho. Tarefas e técnicos têm janelas de tempos, mas não há prioridades para as tarefas. Para resolver esse problema, os autores utilizaram a meta-heurística *Simulated Annealing* (SA) associado a duas heurísticas construtivas.

Outros trabalhos estudaram o PETC, porém maximizando o número de tarefas realizadas. Um deles é o de Overholts II; Bell e Arostegui (2009), que estudaram a manutenção diária em equipamentos militares. As principais restrições do problema são os níveis de segurança exigidos. Não há janelas de tempo. Os autores desenvolveram um modelo de Programação Linear Inteira Mista (PLIM), focando a pesquisa em uma detalhada análise de sensibilidade dos critérios de segurança e de qualidade da manutenção.

Dohn; Kolind e Clausen (2009) estudaram a formação de equipes de técnicos para executar tarefas, procurando maximizar o número de tarefas realizadas. Tanto os times quanto as tarefas têm janelas de tempo e existe a possibilidade de uma cooperação entre os times durante a execução de alguma tarefa. Os autores utilizaram o método *Branch-and-Price* (B&P) em 12 instâncias reais, encontrando 11 soluções ótimas.

Cortés et al. (2014) apresentaram um problema real de manutenção de máquinas copiadoras e impressoras de uma empresa em Santiago (Chile), onde vinte técnicos visitam diariamente cerca de setenta clientes. A empresa estabelece uma janela de tempo flexível (*soft time window* ou janelas de tempo apenas com o fim definido) para cada serviço, de modo que sejam atendidos primeiro os clientes mais importantes e que exista uma meta de tempo máximo para concluir cada serviço (*target response time*). A função objetivo considera o número de vezes que as janelas de tempo são violadas, o número de clientes atendidos e o tempo total de deslocamento dos técnicos. Os autores desenvolveram um *Branch-and-Price* com Programação por Restrições.

Tabela 1.4 – Diferentes funções objetivo e parâmetros do problema encontrados na literatura

Autores	Função objetivo	Prioridade das tarefas	Janelas de tempo	Equipes de trabalho	Terceirização	Principal Método
Tsang e Voudouris (1997)	min custos de deslocamentos + horas extras + tarefas não realizadas	-	tar. e tec.	-	-	FLS / GLS
Xu e Chiu (2001)	max tarefas atendidas + ociosidade	sim	tar. e téc.	-	-	GRASP
Li; Lim e Rodrigues (2005)	min quantidade de téc. + distância	-	tar. e téc.	sim	-	SA
Tang; Miller-Hooks e Tomastik (2007)	max receitas	sim ^a	-	-	-	Busca Tabu
Overholts II; Bell e Arostegui (2009)	max tarefas atendidas	sim	-	sim	-	PLIM
Dohn; Kolind e Clausen (2009)	max tarefas atendidas	-	tar. e téc.	sim	-	B&P
Cordeau et al. (2010)	min ITUSG ^b	sim	-	sim	sim	ALNS
Hashimoto et al. (2011)	min ITUSG ^c	sim	-	sim	sim	GRASP
Kovacs et al. (2012)	min custo de deslocamento + terceirização	-	tar.	sim ^d	sim	ALNS
Pillac; Gueret e Medaglia (2012)	min deslocamentos	-	tar.	-	-	ALNS
Cortés et al. (2014)	min violação das janelas de tempo + deslocamentos + tarefas não programadas	^e	tar. ^f	-	-	B&P

^a Cada tarefa está associada a uma remuneração, que varia ao longo dos dias.

^b As tarefas foram divididas em três grupos de prioridades. A função objetivo minimizou a soma ponderada do instante de término do último serviço realizado de cada grupo (ITUSG).

^c Idem.

^d Os autores também trabalham com instâncias em que as tarefas são executadas por um único técnico (sem times).

^e Para a empresa, os clientes prioritários devem ser atendidos primeiro e, por isso, os autores associam a esses clientes prioritários janelas de tempo mais cedo.

^f É definido apenas o limite superior da janela de tempo (*soft-time window*), ou seja, a empresa define um tempo máximo para atender o cliente (*target response time*).

Xu e Chiu (2001) também adotaram como função objetivo maximizar a soma ponderada de tarefas realizadas, porém com um acréscimo: maximizar a soma dos tempos disponíveis dos técnicos após o retorno à empresa, no fim da jornada de trabalho. As tarefas têm diferentes janelas de tempo, diferentes prioridades e são executadas por um único técnico. Os técnicos também têm janelas de tempo e diferentes habilidades. Os autores desenvolveram uma heurística construtiva para o problema, um algoritmo de busca local e também aplicaram a meta-heurística GRASP para encontrar soluções.

A Tabela 1.4 apresenta uma visão geral da bibliografia citada, com os principais parâmetros do PETC em cada estudo. Observe que enquanto Tang; Miller-Hooks e Tomastik (2007), Overholts II; Bell e Arostegui (2009), Cordeau et al. (2010) e Hashimoto et al. (2011) consideraram as tarefas associadas a prioridades, Tsang e Voudouris (1997), Li; Lim e Rodrigues (2005), Dohn; Kolind e Clausen (2009), Kovacs et al. (2012), Pillac; Gueret e Medaglia (2012) e Cortés et al. (2014) trabalharam com janelas de tempo. Até onde se sabe, somente Xu e Chiu (2001) consideram simultaneamente essas duas importantes características¹ para uma empresa de serviços (como manutenções, instalações, etc.). Motivado também pela relevância da aplicação prática do problema e que janelas de tempo e prioridades podem ser um importante fator de vantagem competitiva nas empresas, esse trabalho pretende explorar essa abordagem do PETC proposta por Xu e Chui (2001), que foi pouco investigada na literatura.

¹Note que neste trabalho, diferentemente do problema abordado por Cortés et al. (2014), esses dois parâmetros são independentes (pois as janelas de tempo são definidas pelo cliente e as prioridades, pela empresa) e ambos são considerados na resolução do problema.

1.4 Modelo de Programação Linear Inteira Mista

O problema abordado contém um conjunto de n tarefas ($J = \{1, \dots, n\}$) e um conjunto de m técnicos ($K = \{1, \dots, m\}$), podendo ser definido como um grafo. Seja o grafo $G = (V, A)$ formado pelos nós $V = J \cup \{0\}$, onde 0 é uma tarefa falsa que representa a origem ou a sede da empresa, e pelas arestas $A = V \times V$. Para cada aresta $(i, j) \in A$ é associado um tempo de deslocamento c_{ij} , que é igual à distância euclidiana entre as coordenadas dos pontos i e j . A cada nó $i \in J$ é associada uma prioridade w_i , um tempo de execução p_i e uma janela de tempo $[e_i, l_i]$. Cada técnico $k \in K$ tem seu horário de trabalho diário definido por $[a_k, b_k]$ e deve deixar a origem após o início e retornar até o fim do seu horário de trabalho. É permitido que um técnico chegue a um nó i antes do início da janela de tempo deste nó, mas ele deverá aguardar o horário e_i para iniciar a execução deste serviço. A habilidade de um técnico k executar uma tarefa i é dada por s_{ik} , que pode ser 0 ou 1, onde 1 indica ser apto e 0, incapaz de executá-la.

A seguir será apresentado o modelo de Programação Linear Inteira Mista (PLIM) para o PETC, que está baseado na proposta de Xu e Chiu (2001). Quatro tipos de variáveis de decisão são utilizados no modelo: y_{ik} , $i \in V$ e $k \in K$ é uma variável binária que é igual a 1 quando uma tarefa i é designada ao técnico k e 0, caso contrário; x_{ijk} , $i, j \in V$ e $k \in K$, também é uma variável binária que é igual a 1 quando uma tarefa i precede outra tarefa j na rota do técnico k e 0, caso contrário; t_i , $i \in J$ é o momento do início da execução de uma tarefa i e z_k , $k \in K$ é o tempo disponível do técnico k após retornar para a sede da empresa. O modelo também adota os parâmetros M , MZ e MW , que são definidos como:

$$\begin{aligned}
 MW &= \min_{i \in J} w_i \\
 MZ &= \max_{k \in K} (b_k - a_k) + \varepsilon \\
 \varepsilon &\text{ é um número pequeno} \\
 M &= \max \left(\max_{i \in J} l_i, \max_{k \in K} a_k \right) + \max_{i, j \in J \cup \{0\}} c_{ij}
 \end{aligned}$$

Formulação:

$$\max \sum_{i \in J} \sum_{\substack{k \in K \\ |s_{ik}=1}} \frac{w_i y_{ik}}{MW} + \sum_{k \in K} \frac{z_k}{MZ} \quad (1.1)$$

Sujeito a:

$$\sum_{\substack{k \in K \\ |s_{ik}=1}} y_{ik} \leq 1 \quad i \in J \quad (1.2)$$

$$\sum_{\substack{j \in J \cup \{0\} \setminus i \\ |s_{jk}=1}} x_{ijk} = \sum_{\substack{j \in J \cup \{0\} \setminus i \\ |s_{jk}=1}} x_{jik} = y_{ik} \quad k \in K, i \in J, s_{ik} = 1 \quad (1.3)$$

$$\sum_{\substack{i \in J \\ |s_{ik}=1}} x_{0ik} \leq 1 \quad k \in K \quad (1.4)$$

$$e_i \leq t_i \leq l_i - p_i \sum_{\substack{k \in K \\ |s_{ik}=1}} y_{ik} \quad i \in J \quad (1.5)$$

$$t_i + p_i + c_{ij} \leq t_j + M(1 - x_{ijk}) \quad k \in K, i \neq j \in J, s_{ik} = s_{jk} = 1 \quad (1.6)$$

$$a_k + c_{0j} \leq t_j + M(1 - x_{0jk}) \quad k \in K, j \in J, s_{jk} = 1 \quad (1.7)$$

$$t_i + p_i + c_{i0} \leq b_k - z_k + M(1 - x_{i0k}) \quad k \in K, i \in J, s_{ik} = 1 \quad (1.8)$$

$$z_k \leq b_k - a_k \quad k \in K \quad (1.9)$$

$$x_{ijk}, y_{ik} \in \{0, 1\} \quad i, j \in J, k \in K, s_{ik} = s_{jk} = 1 \quad (1.10)$$

$$t_i, z_k \in \mathbf{R}_+ \quad i \in J, k \in K \quad (1.11)$$

Na função objetivo (1.1), com o primeiro termo procura-se maximizar a soma dos valores das prioridades das tarefas realizadas e, com o segundo, a soma dos tempos de disponibilidade dos técnicos após o retorno à base. Os denominadores têm duas funções: tornar adimensionais os valores somados e priorizar a execução de uma tarefa, reduzindo a ociosidade de um técnico, e não o contrário. Concretamente, quando uma tarefa i é associada a um técnico k ($y_{ik} = 1$), a fração $\frac{w_i y_{ik}}{MW}$ sempre será maior ou igual a um, enquanto que o termo ε garante que a expressão $\frac{z_k}{MZ}$ sempre será estritamente menor que um.

Em outras palavras, o primeiro objetivo procura executar as tarefas mais prioritárias e o objetivo secundário busca indiretamente reduzir o tempo de deslocamento total e os tempos de espera para o início da execução de uma tarefa. Ao mesmo tempo, esse segundo objetivo é uma estratégia que pode permitir a inclusão de uma eventual nova

tarefa externa ou que os técnicos executem tarefas administrativas da empresa (como a elaboração de relatórios, manutenção de equipamentos, etc.). Pode-se notar também que esse segundo objetivo é similar ao dos problemas de *scheduling* em máquinas paralelas que minimizam o somatório do instante de término das últimas tarefas de cada máquina (*the total completion time*).

As restrições (1.2) garantem que cada tarefa será alocada a, no máximo, um técnico. O conjunto de restrições (1.3) garante que, quando uma tarefa i é designada a um técnico k ($y_{ik} = 1$), há uma única antecessora e uma única sucessora de i .

Cada técnico deve deixar a origem, no máximo, uma vez, como assegura (1.4). Essa condição tem um impacto direto nos conjuntos de restrições (1.7) e (1.8), que garantem que os horários de trabalho dos técnicos são respeitados. Cada tarefa deve ser executada dentro de sua janela de tempo e, portanto, o conjunto de restrições (1.5) determina que o tempo de início de sua execução deve ser maior ou igual ao início de sua janela e menor ou igual ao seu término menos o seu tempo de execução.

Quando duas tarefas i e j são alocadas a um mesmo técnico e são executadas uma após a outra, (1.6) exige que o início do processamento de uma tarefa j aconteça depois do instante de término da tarefa antecessora i somado ao tempo de deslocamento entre os locais das dessas duas tarefas. O termo $M(1 - x_{ijk})$ serve para ativar a restrição quando i precede j no técnico k (ou seja, $x_{ijk} = 1$) e para desativá-la, caso contrário (quando $x_{ijk} = 0$). As restrições (1.7) garantem que o início da primeira tarefa de um técnico k ocorra somente depois do início de seu turno de trabalho somado ao deslocamento da base ao local de execução dessa tarefa; para todas as demais tarefas, $x_{0jk} = 0$ e essas restrições serão desativadas. Para que um técnico k retorne à origem antes do término da sua jornada de trabalho, (1.8) determina que o fim da execução da última tarefa de um técnico somado ao tempo de deslocamento até a origem é menor ou igual ao término da sua janela de tempo; analogamente a (1.7), as restrições (1.8) são desativadas para as demais tarefas. Pode-se ressaltar que como a função objetivo procura maximizar a disponibilidade dos técnicos no fim do dia (z_k), essa restrição determinará o menor valor possível para t_i e o maior valor possível para z_k . As restrições (1.6), (1.7) e (1.8), ao garantir que as janelas de tempo são respeitadas e os outros aspectos mencionados anteriormente, evitam *subtour*, ou seja, subrotas que não contêm a base. O tempo disponível (ou ocioso) de um técnico k após o retorno à origem é menor ou igual ao

tamanho de sua janela de tempo, conforme assegura (1.9).

As variáveis x e y são binárias e as variáveis t e z , reais não negativos, como pode-se observar em (1.10) e (1.11). O parâmetro M será um número maior que o máximo horário de término da janela de tempo de qualquer tarefa ou o máximo horário de início de turno de qualquer técnico, somado ao maior tempo de deslocamento entre os locais de duas tarefas ou entre a origem e uma tarefa. Desse modo, M será um número grande o suficiente para desativar as restrições (1.6), (1.7) e (1.8) quando necessário.

Note que, tanto na função objetivo quanto nas restrições, as variáveis y_{ik} são geradas somente quando $s_{ik} = 1$ e as variáveis x_{ijk} , quando $s_{ik} = s_{jk} = 1$, ou seja, somente quando um técnico k pode executá-las. Com isso, procurou-se reduzir o número de variáveis do modelo e torná-lo mais eficiente.

Pode-se finalmente ressaltar que alguns aspectos não foram considerados no modelo PLIM e que poderiam ser abordados em futuros trabalhos. Por exemplo, o horário de almoço de cada técnico poderia ser considerado como uma tarefa fictícia com duração de 60 min., janela de tempo entre 11h30min e 14h30min e obrigatoriamente incluída no roteiro de cada técnico.

1.5 Complexidade computacional

De acordo com Kovacs et al. (2012), Pillac; Gueret e Medaglia (2012) e Xu e Chiu (2001), o PETC é uma extensão do Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT), que é NP-difícil. Além do PRVJT, pode-se comparar o PETC ao Problema de Máquinas Paralelas (PMP) ou o Problema de Múltiplas Mochilas (PMM). A relação entre o PETC e os dois primeiros são apresentadas no Apêndice A. A seguir, é apresentada uma redução do PMM – um problema conhecidamente NP-difícil (GAREY; JOHNSON, 1979) – ao PETC.

1.5.1 Redução ao problema de múltiplas mochilas

Deseja-se mostrar que uma instância do PMM pode ser convertida em tempo polinomial a uma instância do PETC e, portanto, o PMM é reduzível ao PETC.

Inicialmente, considere um caso particular do PETC, em que o tempo de deslocamento entre as tarefas é zero ($c_{ij} = 0, \forall i, j \in J \cup \{0\}$), as janelas de tempo das tarefas iniciam antes do início do dia de todos os técnicos ($e_i \leq a_k, \forall i \in J, k \in K$) e terminam

após o fim do dia de todos os tecnicos ($b_k \leq l_i, \forall i \in J, k \in K$). Considere também que todos os técnicos podem processar qualquer tarefa ($s_{ik} = 1, \forall i \in J, k \in K$), que $MW = 1$ e que $MZ \rightarrow \infty$ ou $\frac{1}{MZ} = 0$. O modelo PLIM do PETC, apresentado na Seção 1.4, pode ser re-escrito da seguinte maneira:

$$\max \sum_{i \in J} \sum_{k \in K} w_i y_{ik} \quad (1.12)$$

Sujeito a:

$$\sum_{k \in K} y_{ik} \leq 1 \quad i \in J \quad (1.13)$$

$$\sum_{j \in J \cup \{0\} \setminus i} x_{ijk} = \sum_{j \in J \cup \{0\} \setminus i} x_{jik} = y_{ik} \quad k \in K, i \in J \quad (1.14)$$

$$\sum_{i \in J} x_{0ik} \leq 1 \quad k \in K \quad (1.15)$$

$$t_i + p_i \leq t_j + M(1 - x_{ijk}) \quad k \in K, i \neq j \in J \quad (1.16)$$

$$a_k \leq t_j + M(1 - x_{0jk}) \quad k \in K, j \in J \quad (1.17)$$

$$t_i + p_i \leq b_k + M(1 - x_{i0k}) \quad k \in K, i \in J \quad (1.18)$$

$$x_{ijk}, y_{ik} \in \{0, 1\} \quad i, j \in J, k \in K \quad (1.19)$$

$$t_i \in \mathbf{R}_+ \quad i \in J \quad (1.20)$$

Onde:

$$M > \max \left(\max_{i \in J} l_i, \max_{k \in K} a_k \right) \quad (1.21)$$

Repare que as restrições (1.5) e (1.9) deixaram de ser necessárias no modelo, uma vez que as janelas de tempo dos técnicos estão contidas nas janelas de tempo das tarefas e que a ociosidade deixou de influenciar na função objetivo, pois $\frac{1}{MZ} = 0$. Em outras palavras, essas restrições deixaram de restringir a região factível do problema e, portanto, ao retirá-las a região factível não será alterada.

Com relação à variável z_k , uma vez que ela deixa de fazer parte da função objetivo, a sua presença passa a acontecer unicamente na restrição (1.8). Ao maximizar o número de tarefas atendidas, convém que a parte direita dessa restrição seja a maior possível e, portanto, z_k deverá assumir valor zero. Por isso, não há inconveniente em retirá-la do modelo e re-escrever a restrição (1.8) como (1.18).

Como as janelas de tempo das tarefas são suficientemente grandes e, portanto, as

tarefas podem ser feitas em qualquer ordem, as variáveis x_{ijk} perdem a importância e outra simplificação que se pode fazer é eliminá-las. Pelo mesmo motivo, as variáveis t_i também podem ser eliminadas, como será detalhado nos próximos parágrafos.

A seguir as restrições (1.16) a (1.18) serão re-escritas em sua versão linear:

$$t_i + p_i \leq t_j + M(1 - x_{ijk}) \iff x_{ijk}(t_i + p_i - t_j) \leq 0, k \in K, i \neq j \in J \quad (1.22)$$

$$a_k \leq t_j + M(1 - x_{0jk}) \iff x_{0jk}(a_k - t_j) \leq 0, k \in K, j \in J \quad (1.23)$$

$$t_i + p_i \leq b_k + M(1 - x_{i0k}) \iff x_{i0k}(t_i + p_i - b_k) \leq 0, k \in K, i \in J \quad (1.24)$$

Somando essas restrições re-escritas, para cada técnico tem-se que:

$$\sum_{j \in J} x_{0jk}(a_k - t_j) + \sum_{i \in J} \sum_{j \in J \setminus i} x_{ijk}(t_i + p_i - t_j) + \sum_{i \in J} x_{i0k}(t_i + p_i - b_k) \leq 0, k \in K \quad (1.25)$$

Na equação (1.25), a primeira parte está relacionada ao início da janela de tempo do técnico k e à sua primeira tarefa; a segunda parte consiste no somatório que envolve a primeira e a segunda tarefas, a segunda e a terceira tarefas, e assim sucessivamente, até a penúltima e a última tarefa; a terceira parte associa a última tarefa do técnico ao fim da sua janela de tempo. Ou seja, há uma soma e uma subtração de todas as variáveis t e, então, elas podem ser retiradas da expressão, ficando a restrição (1.25) equivalente a:

$$\sum_{j \in J} x_{0jk}a_k + \sum_{i \in J} \sum_{j \in J \setminus i} x_{ijk}p_i + \sum_{i \in J} x_{i0k}(p_i - b_k) \leq 0, k \in K \quad (1.26)$$

Sem perda da generalidade, pode-se considerar que todos os técnicos sairão e voltarão para a base e que $\sum_{j \in J} x_{0jk} = 1$ e $\sum_{i \in J} x_{i0k} = 1, \forall k \in K$. Portanto:

$$a_k + \sum_{i \in J} \sum_{j \in J \cup \{0\} \setminus i} x_{ijk}p_i - b_k \leq 0, k \in K \quad (1.27)$$

Pela restrição 1.14, pode-se substituir $\sum_{j \in J \cup \{0\} \setminus i} x_{ijk}$ por y_{ik} na 1.27:

$$\sum_{i \in J} y_{ik}p_i \leq b_k - a_k, k \in K \quad (1.28)$$

Desse modo – e também desconsiderando as variáveis x_{ijk} , como já foi explicado

anteriormente – pode-se relaxar o modelo, que ficará da seguinte maneira:

$$\max \sum_{i \in J} \sum_{k \in K} w_i y_{ik} \quad (1.29)$$

Sujeito a:

$$\sum_{k \in K} y_{ik} \leq 1 \quad i \in J \quad (1.30)$$

$$\sum_{i \in J} y_{ik} p_i \leq b_k - a_k \quad k \in K \quad (1.31)$$

$$y_{ik} \in \{0, 1\} \quad i \in J, k \in K \quad (1.32)$$

O modelo de PLIM acima é equivalente ao modelo de PLIM do PMM, onde:

- A função objetivo (1.29) procura maximizar o número ponderado de tarefas realizadas pelos técnicos ou de itens colocados nas mochilas.
- Cada tarefa pode ser associada a no máximo um técnico ou cada item pode ser colocado no máximo em uma mochila, como assegura as restrições (1.30).
- Pelas restrições (1.31), a soma dos tempos de serviços realizados de cada técnico não poderá superar o seu número de horas de trabalho ou o volume dos itens colocados em cada mochila não poderá superar a sua capacidade.
- A variável binária y_{ik} indica se uma tarefa foi associada a um técnico ou se um item foi colocado em uma mochila, conforme (1.32) .

Como a função objetivo e as restrições deste caso particular do PETC e do PMM são equivalente, pode-se concluir que o PMM é reduzível ao PETC e que, portanto, o PETC é realmente NP-difícil. **(c.q.d.)**

Capítulo 2

Heurísticas construtivas

As heurísticas construtivas geram soluções factíveis para problemas de otimização através de regras (*dispatching rules*) ou estratégias *gulosas*, que requerem um pequeno esforço computacional. Apesar de não garantirem a qualidade das soluções, quando são bem exploradas as características específicas de um problema, essas heurísticas podem ter um bom desempenho com um pequeno tempo computacional e, por isso, foram utilizadas em diversas pesquisas de *scheduling* (RONCONI e HENRIQUES, 2009; BRANCO et al., 2009; PAN e WANG, 2012; SOBREIRO e NAGANO, 2013) e também em outros problemas combinatórios (SILVA JUNIOR et al., 2008; QU et al., 2009). Outra motivação para desenvolver heurísticas construtivas para o problema de escalonamento de técnicos de campo, é que as suas eventuais boas soluções poderão ser utilizadas como soluções iniciais para as metas-heurísticas, podendo influenciar positivamente o seu desempenho.

Nesse contexto, diversas heurísticas construtivas são apresentadas a seguir. A primeira foi proposta por Xu e Chiu (2001) e as demais, desenvolvidas neste trabalho. Resumidamente, essas heurísticas inicialmente ordenarão as tarefas de acordo com algum critério e depois irão alocá-las nas rotas dos técnicos. Para ordenar as tarefas foi utilizado o algoritmo *Merge-Sort*, de complexidade computacional $O(n \lg n)$, onde n é o número de elementos a serem ordenados (CORMEN et al., 2001).

2.1 Heurística proposta por Xu e Chiu (XC)

Xu e Chiu (2001) propuseram uma heurística com 4 passos:

1. *Ordenar tarefas:* ordene as tarefas em ordem decrescente da proporção $\frac{w_i}{p_i}$. Em caso de empate, o segundo critério de decisão é a ordem decrescente do valor de $l_i - p_i$. Selecione a tarefa i melhor classificada.
2. *Ordenar os técnicos candidatos:* ordene os técnicos que são capazes de executar a tarefa i (ou seja, $s_{ik} = 1$) em ordem crescente do valor da distância entre o local da última tarefa da rota do técnico (ou a base) e a tarefa i .
3. *Escolha um técnico:* associe a tarefa i ao primeiro técnico da lista do Passo 2, que possa realizá-la sem violar as restrições do problema. Avalie todas as posições possíveis da tarefa i na rota do técnico, escolha a posição que resulta na solução factível com maior ociosidade após o retorno à base (z_k) e implemente a atribuição.
4. *Critério de parada:* se a tarefa i for a última classificada no Passo 1, fim do algoritmo. Caso contrário, selecione a próxima tarefa i e volte para o Passo 2.

A complexidade computacional desta heurística é $O(n^2m)$.

Para ilustrar o desempenho da heurística construtiva XC e das demais heurísticas apresentadas neste capítulo, a seguir encontra-se um exemplo com dezesseis tarefas e dois técnicos. O primeiro técnico trabalha entre 8h e 17h e pode executar todas as tarefas, com exceção das tarefas 2, 4 e 6; o segundo técnico trabalha das 11h às 20h e pode executar as tarefas, exceto 7, 8, 10, 12, 14 e 15. A Tabela 2.1 apresenta os demais parâmetros do problema. As prioridades das tarefas variam entre 1 (baixa) e 10 (alta).

Tabela 2.1 – Parâmetros do exemplo: a) prioridades, tempos de execução e janelas de tempo das tarefas. b) Tempo de deslocamento c_{ij} (em minutos) entre as tarefas e a sede da empresa.

(a)

Tarefa	Prioridade	Tempo de execução	Início da janela	Fim da janela
	w_i	p_i	e_i	l_i
1	1	55min	14h15min	19h00min
2	2	1h15min	15h30min	17h45min
3	9	55min	14h00min	18h15min
4	4	1h40min	10h55min	17h25min
5	8	1h55min	14h20min	17h30min
6	9	1h20min	14h25min	18h35min
7	2	1h35min	9h35min	12h45min
8	2	1h00min	8h50min	11h00min
9	4	40min	9h25min	17h15min
10	3	1h10min	9h15min	16h00min
11	10	55min	15h00min	17h20min
12	9	50min	9h25min	15h50min
13	10	2h00min	11h40min	18h30min
14	4	40min	14h25min	18h40min
15	7	50min	9h50min	12h50min
16	8	1h50min	9h10min	17h10min

(b)

c_{ij}	base	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
base	0	15	18	22	25	20	11	21	26	31	25	33	15	11	31	30	29
1	15	0	32	14	32	32	24	21	31	17	15	26	16	26	46	44	42
2	18	32	0	34	20	23	16	36	35	46	42	49	23	9	21	13	25
3	22	14	34	0	25	42	33	35	44	15	29	40	11	31	52	46	50
4	25	32	20	25	0	40	31	45	50	39	46	56	15	25	40	29	44
5	20	32	23	42	40	0	10	20	14	49	33	35	35	15	20	29	11
6	11	24	16	33	31	10	0	20	19	42	30	35	25	7	22	25	18
7	21	21	36	35	45	20	20	0	12	36	14	15	33	26	40	45	31
8	26	31	35	44	50	14	19	12	0	47	26	24	40	26	33	42	23
9	31	17	46	15	39	49	42	36	47	0	25	35	25	42	63	59	59
10	25	15	42	29	46	33	30	14	26	25	0	11	31	34	51	54	44
11	33	26	49	40	56	35	35	15	24	35	11	0	42	40	54	60	45
12	15	16	23	11	15	35	25	33	40	25	31	42	0	22	42	35	42
13	11	26	9	31	25	15	7	26	26	42	34	40	22	0	21	20	20
14	31	46	21	52	40	20	22	40	33	63	51	54	42	21	0	15	11
15	30	44	13	46	29	29	25	45	42	59	54	60	35	20	15	0	25
16	29	42	25	50	44	11	18	31	23	59	44	45	42	20	11	25	0

Figura 2.1 – Solução da heurística XC para o exemplo

Fonte: elaborado pelo autor.

O Passo 1 da heurística XC ordena as tarefas de acordo com a ordem decrescente da razão prioridade por tempo de processamento. Assim, tem-se a seguinte ordem de tarefas: 11, 12, 3, 15, 6, 14, 9, 13, 16, 5, 10, 4, 8, 2, 7 e 1. Note que as tarefas 9 e 14, por terem a mesma razão, foram ordenadas pelo segundo critério, que é a ordem decrescente de $l_i - p_i$. A programação começa com a tarefa 11, que pode ser executada por ambos técnicos e que estão equidistantes do local da tarefa; desse modo, o técnico 1 é selecionado, por ter o menor número de identificação. A tarefa 12 pode ser feita apenas pelo técnico 1 e é adicionada na sua rota na primeira posição (antes da tarefa

Tabela 2.2 – Informações das tarefas executadas por cada técnico na solução da heurística XC: a) Técnico 1. b) Técnico 2.

Tarefa	12	15	9	10	11	Empresa
Horário de chegada do técnico	8h15min	10h50min	12h40min	13h44min	15h05min	16h33min
Tempo de espera	1h10min	-	-	-	-	-
Início da execução	9h25min	10h50min	12h40min	13h44min	15h05min	-
Fim da execução	10h15min	11h40min	13h20min	14h54min	15h60min	-

(a)

Tarefa	13	3	2	6	Empresa
Horário de chegada do técnico	11h11min	14h11min	15h40min	17h11min	18h42min
Tempo de espera	29min	-	-	-	-
Início da execução	11h40min	14h11min	15h40min	17h11min	-
Fim da execução	13h40min	15h06min	16h55min	18h31min	-

(b)

11), pois realizá-la após a tarefa 12 não é factível. A tarefa 3 é adicionada ao técnico 2, pois ele está na sede da empresa e mais perto do local desta tarefa, uma vez que a última tarefa do técnico 1 é a tarefa 11. E assim por diante, as tarefas seguintes são programadas (caso haja solução factível). A solução da heurística encontra-se na Figura 2.1. A rota do técnico 1 é formada pelas tarefas 12, 15, 9, 10 e 11; a rota do técnico 2, pelas tarefas 13, 3, 2 e 6. A Tabela 2.2 apresenta o detalhamento da solução. Note que 9 das 16 tarefas são programadas e que a soma das prioridades dessas tarefas é 63. O tempo de disponibilidade total dos técnicos após o retorno à base é 1h45min. O valor da função objetivo¹ é, portanto, 63,2.

Figura 2.2 – Solução ótima do exemplo

Fonte: elaborado pelo autor.

Para medir a qualidade da solução gerada pela heurística XC, o seu respectivo valor da função objetivo foi comparado com o valor da função objetivo da solução ótima. A Figura 2.2 apresenta a solução ótima, que é detalhada na Tabela 2.3. Das 16 tarefas, 8 são programadas, sendo que todas as tarefas programadas têm prioridades acima de 7 e as não programadas, prioridades inferiores a 4. A soma das prioridades das tarefas programadas é igual a 70 e a soma das disponibilidades dos técnicos no fim do dia, igual a 1h41min, resultando no valor de 70,2 para a função objetivo. A diferença entre o valor da solução ótima e a solução da heurística (GAP_{Opt}) é $GAP_{Opt} = 9,9\%$, que é definido por:

¹Adotou-se que $MW = 1$ e $MZ = 550$, sendo *minutos* adotado como unidade da variável z_k .

Tabela 2.3 – Informações das tarefas executadas por cada técnico na solução ótima: a) Técnico 1. b) Técnico 2.

(a)					
Tarefa	12	15	13	5	Empresa
Horário de chegada do técnico	8h15min	10h50min	12h00min	14h15min	16h35min
Tempo de espera	1h10min	-	-	5min	-
Início da execução	9h25min	10h50min	12h00min	14h20min	-
Fim da execução	10h15min	11h40min	14h00min	16h15min	-

(b)					
Tarefa	16	3	11	6	Empresa
Horário de chegada do técnico	11h29min	14h09min	15h44min	17h13min	18h44min
Tempo de espera	-	-	-	-	-
Início da execução	11h29min	14h09min	15h44min	17h13min	-
Fim da execução	13h19min	15h04min	16h39min	18h33min	-

$$GAP_{Opt} = 100 \cdot \frac{Opt - SV}{Opt} \quad (2.1)$$

onde Opt é o valor da função objetivo da solução ótima e SV , o valor da função objetivo do método heurístico.

2.2 Técnico mais próximo (*Nearest technician* - NT)

Esse método é uma tentativa de melhorar a heurística XC e tem quatro passos:

1. *Ordenar tarefas*: ordene as tarefas em ordem decrescente de ρ , dado por:

$$\rho_i = \begin{cases} w_i - \frac{NST_i}{m+1}, & \text{se } \frac{n}{m} < 10 \\ w_i + \frac{w_i}{p_i} + \frac{p_i}{l_i - e_i}, & \text{caso contrário} \end{cases} \quad (2.2)$$

onde NST_i é o número de técnicos com habilidade para realizar a tarefa i . Note que, se a instância tem poucas tarefas por técnico (menos de 10), então as tarefas são ordenadas por ordem de prioridade; a segunda parte da expressão servirá como critério de desempate quando duas ou mais tarefas têm a mesma prioridade. Caso contrário (10 ou mais tarefas por técnico), é muito provável que algumas tarefas não são realizadas e, dado o tempo limitado de trabalho diário dos técnicos, é

importante também levar em conta o tempo de processamento. Dessa forma, as tarefas são ordenadas em ordem decrescente do valor da prioridade somada à prioridade por tempo de processamento ($w_i + \frac{w_i}{p_i}$); se duas ou mais tarefas têm o mesmo valor, o desempate é feito pelo valor de $\frac{p_i}{l_i - e_i}$, procurando programar antes as tarefas que têm maior tempo de processamento e menores janelas de tempo. Caso duas tarefas tenham o mesmo valor de ρ_i , elas são ordenadas pela ordem crescente do número de identificação da tarefa.

Selecione a tarefa i melhor classificada.

2. *Selecionar um técnico:* selecione o técnico capaz k (não examinado ainda) com o local da última tarefa da sua rota (ou da base) mais perto do local da tarefa i . Se o técnico não puder executá-la, escolha o próximo técnico mais próximo; no entanto, se a tarefa não pode ser introduzida na rota de nenhum técnico com habilidade, siga para o Passo 4. Quando mais de um técnico tem a mesma distância mínima da tarefa i (o que ocorre, por exemplo, quando todos os técnicos estão na base), selecione o técnico k com a menor soma ponderada de habilidades ($\sum_{j \in D} w_j s_{jk}$, onde D é o conjunto de tarefas que ainda não se tentou introduzir nas rotas dos técnicos), procurando preservar os técnicos mais capazes de realizar tarefas ainda não avaliadas. O último critério de decisão é o número de identificação do técnico.
3. *Programar as tarefas:* avalie todas as posições possíveis da tarefa i na rota do técnico k . Se há soluções factíveis, escolha a posição que resulta na solução com maior z_k , implemente a atribuição e siga para o próximo passo.
4. *Critério de parada:* se a tarefa i é a última classificada no Passo 1, fim. Caso contrário, selecione a próxima tarefa i e volte para o Passo 2.

A complexidade computacional desta heurística é $O(n^2m)$.

Para exemplificar o desempenho dessa heurística, a seguir é apresentada a solução que ela gera para o exemplo apresentado na Seção 2.1. O Passo 1 da heurística NT ordena as tarefas de acordo com a seguinte ordem: 11, 13, 6, 12, 3, 5, 16, 15, 4, 14, 9, 10, 2, 7, 8 e 1. A primeira tarefa a ser programada (tarefa 11) é destinada para a

Figura 2.3 – Solução da heurística NT para o exemplo

Fonte: elaborado pelo autor.

rota do técnico 2: uma vez que ambos técnicos têm habilidade para executá-la e estão equidistantes, o critério de desempate é que o técnico 2 tem habilidade para executar apenas 9 tarefas, enquanto que o técnico 1 pode realizar 13 tarefas. A segunda tarefa a ser programada é destinada para o técnico 1, que está na sede da empresa e mais perto do local da tarefa 13. E assim por diante, as tarefas são programadas e a seguinte solução é gerada: o técnico 1 executa as tarefas 12, 15, 13, 3 e 9; e o técnico 2, as tarefas 16, 6, 11 e 1. A Figura 2.3 apresenta essa solução e a Tabela 2.4, os seus detalhes numéricos. Note que, assim como na solução da heurística XC, 9 das 16 tarefas são

Tabela 2.4 – Informações das tarefas executadas por cada técnico na solução da heurística NT: a) Técnico 1. b) Técnico 2.

(a)

Tarefa	12	15	13	3	9	Empresa
Horário de chegada do técnico	8h15min	10h50min	11h60min	14h31min	15h41min	16h52min
Tempo de espera	1h10min	-	-	-	-	-
Início da execução	9h25min	10h50min	11h60min	14h31min	15h41min	-
Fim da execução	10h15min	11h40min	13h60min	15h26min	16h21min	-

(b)

Tarefa	16	6	11	1	Empresa
Horário de chegada do técnico	11h29min	13h36min	16h20min	17h41min	18h50min
Tempo de espera	-	49min	-	-	-
Início da execução	11h29min	14h25min	16h20min	17h41min	-
Fim da execução	13h19min	15h45min	17h15min	18h36min	-

programadas, mas que agora a soma das prioridades dessas tarefas aumenta para 67 e o tempo de disponibilidade total dos técnicos após o retorno à base diminui para 1h18min. O valor da função objetivo é, portanto, 67,1 (ou seja, 6% superior ao valor da função objetivo da solução gerada pela heurística XC). A diferença em relação à solução ótima, definida pela equação (2.1), é $GAP_{Opt} = 4,4\%$.

2.3 Menor tempo de viagem (*Shortest travel time* - STT)

Esta heurística construtiva também está formada por quatro etapas:

1. *Inicialização*: ordene as tarefas em ordem decrescente de ρ , calculado pela equação (2.2). Selecione a tarefa i melhor classificada.
2. *Análise dos técnicos candidatos*: para cada técnico k que seja capaz de executar a tarefa i (ou seja, $s_{ik} = 1$), avalie todas as posições possíveis para a nova tarefa na sua rota. Escolha a posição que resulta na solução factível com maior ociosidade após o retorno à base (z_k).
3. *Escolha um técnico*: se não houver soluções factíveis no Passo 2, siga para o 4. Caso contrário, selecione o técnico k que possa executar essa a tarefa i com menor tempo de deslocamento total. Em caso de empate, o segundo e o terceiro critério de decisão são o técnico com a maior ociosidade z_k e o número de identificação do técnico. Implemente a atribuição.
4. *Critério de parada*: se a tarefa i for a última classificada no Passo 1, fim do algoritmo. Caso contrário, selecione a próxima tarefa i e volte para o Passo 2.

A complexidade computacional desta heurística é $O(n^2m)$.

Para o exemplo apresentado na Seção 2.1, a heurística STT gera a mesma solução encontrada pela heurística NT. O primeiro passo de ambas heurísticas é idêntico. No segundo passo, é verificado que ambos técnicos podem executar a tarefa e que as disponibilidades no fim do dia para a solução do técnico 1 é de 32min e, para a solução do técnico 2, 3h32min. No terceiro passo, o segundo técnico é escolhido, uma vez que ambos

têm o mesmo tempo de deslocamento total para executar essa tarefa (neste momento, apenas o deslocamento para ir e voltar da sede da empresa para o local da tarefa 11), mas o técnico 2 tem maior disponibilidade no fim do dia de trabalho. E assim sucessivamente, tenta-se programar as demais tarefas e, coincidentemente, a solução obtida tem as mesmas tarefas e as mesmas rotas da solução gerada pela heurística NT. Como já foi descrito, essa solução tem valor da função objetivo 6% superior ao valor da função objetivo da solução gerada pela heurística XC, com $GAP_{Opt} = 4,4\%$.

2.4 Menor instante de início das tarefas prioritárias (*Earliest start time of priority tasks* - EST)

Essa heurística construtiva tem o objetivo de que as tarefas mais prioritárias sejam atendidas mais cedo. Concretamente, seja uma variável *folga* (γ) a fração de tempo disponível entre o fim da execução de uma tarefa e o término da sua janela de tempo:

$$\gamma_i = \frac{(l_i - t_i - p_i + \varepsilon)}{(l_i - e_i - p_i + \varepsilon)} \quad (2.3)$$

onde:

ε é um número muito pequeno

Desse modo, a heurística seleciona o técnico e a posição da tarefa na sua rota que resultem no maior aumento da soma ponderada das *folgas* ($\sum_{i \in J} w_i \gamma_i y_{ik}$). A seguir encontra-se a descrição detalhada desta heurística construtiva.

1. *Inicialização*: ordene as tarefas em ordem decrescente de ρ , calculado pela equação (2.2). Selecione a tarefa i melhor classificada.
2. *Análise dos técnicos candidatos*: para cada técnico k que seja capaz de executar a tarefa i (ou seja, $s_{ik} = 1$), avalie todas as posições possíveis para a nova tarefa na sua rota. Escolha a posição que resultar na solução factível com maior aumento da soma ponderada das *folgas* ($\sum_{i \in J} w_i \gamma_i y_{ik}$). Serão descartadas as soluções factíveis com valor da soma ponderada das *folgas* menor do que o valor da soma ponderada sem a tarefa i .

3. *Escolha um técnico:* se não houver soluções no Passo 2, siga para o Passo 4. Caso contrário, selecione o técnico k que pode executar essa a tarefa i com o maior aumento da soma ponderada das *folgas*. Em caso de empate, o segundo e o terceiro critério de decisão são o técnico com o menor tempo total de viagem e o número de identificação do técnico. Implemente a atribuição.
4. *Critério de parada:* se a tarefa i for a última classificada no Passo 1, fim do algoritmo. Caso contrário, selecione a próxima tarefa i e volte para o Passo 2.

A complexidade computacional desta heurística é $O(n^2m)$.

No exemplo apresentado na Seção 2.1, a heurística EST encontra a solução apresentada na Figura 2.4. O Passo 1 da heurística ordena as tarefas do mesmo modo que as heurísticas NT e STT. Assim, a tarefa 11 – que pode ser igualmente realizada pelos técnicos 1 e 2, com a mesma folga e mesmo tempo de viagem total – é incluída na rota do técnico com o menor número de identificação (técnico 1). A segunda tarefa programada (tarefa 13) é incluída na rota do técnico 2, uma vez que ambos podem realizá-la com igual folga e que o técnico 1 terá um maior tempo de deslocamento total caso a execute. A terceira tarefa a ser programada é a tarefa 6, que pode ser executada apenas pelo técnico 2. Como esse técnico já tem em sua rota a tarefa 13, o Passo 2 analisa todas

Figura 2.4 – Solução da heurística EST para o exemplo

Fonte: elaborado pelo autor.

Tabela 2.5 – Informações das tarefas executadas por cada técnico na solução da heurística EST: a) Técnico 1. b) Técnico 2.

(a)					
Tarefa	12	15	16	11	Empresa
Horário de chegada do técnico	8h15min	10h50min	12h05min	14h40min	16h28min
Tempo de espera	1h10min	-	-	20min	-
Início da execução	9h25min	10h50min	12h05min	15h00min	-
Fim da execução	10h15min	11h40min	13h55min	15h55min	-

(b)					
Tarefa	13	3	6	1	Empresa
Horário de chegada do técnico	11h11min	14h11min	15h39min	17h24min	18h34min
Tempo de espera	29min	-	-	-	-
Início da execução	11h40min	14h11min	15h39min	17h24min	-
Fim da execução	13h40min	15h06min	16h59min	18h19min	-

as posições possíveis da nova tarefa na rota. Se a tarefa 3 for a primeira tarefa da rota, ela poderá ser executada no início de sua janela de tempo (14h25min), mas o técnico chegará no local da tarefa 13 às 15h52min, reduzindo a folga dessa tarefa para 0,13 e o somatório ponderado das folgas para 10,3. Se a nova tarefa for realizada após a tarefa 13, o técnico poderá chegar nos seus locais no início das respectivas janelas de tempo e, portanto, as folgas são iguais a 1 e soma ponderada das folgas é igual a 19,0. Portanto, a tarefa 6 é programada na segunda posição da rota do técnico 2. E assim por diante, as tarefas são programadas, encontrando a solução apresentada na Figura 2.4 e detalhada na Tabela 2.5. Apenas 8 das 16 tarefas são programadas; a soma das prioridades das tarefas executadas é 63 e o tempo de disponibilidade total dos técnicos após o retorno à base, 1h58min, resultando no valor da função objetivo igual a 63,2 (valor semelhante ao da solução gerada pela heurística XC). A diferença em relação à solução ótima, definida pela equação (2.1), é $GAP_{Opt} = 9,9\%$.

2.5 Habilidade dos técnicos (*Technicians' skill* - TS)

Tendo em conta a relevância das habilidades dos técnicos (s_{ik}), este método os ordena em ordem crescente do número total de tarefas potenciais, ou seja, pela quantidade de habilidades ($\sum_{i \in J} s_{ik}$). Primeiramente, procurar-se-á designar as tarefas mais prioritárias para os técnicos com menor número de habilidades; depois, para o segundo técnico com menor número de habilidades e assim por diante.

O método é composto por 4 passos:

1. *Inicializar*: ordene as tarefas em ordem decrescente de ρ , calculado pela equação (2.2). Selecione a tarefa i melhor classificada.
2. *Ordenar os técnicos*: classifique os técnicos em ordem crescente do número de habilidades ($\sum_{i \in J} s_{ik}$) e selecione o técnico k com o menor número de habilidades.
3. *Designar tarefas a um técnico*:
 - (a) Se i já foi designada a algum técnico ou se k não for capaz de executá-la ($s_{ik} = 0$), siga para o Passo 3b. Caso contrário, avalie todas as posições possíveis para a tarefa i na rota do técnico k , escolha a que resulte em uma solução factível com maior z_k e implemente a atribuição.
 - (b) Se a tarefa i for a última classificada no Passo 1, siga para o próximo passo. Caso contrário, selecione a próxima tarefa i e volte ao Passo 3a.
4. *Critério de parada*: Se o técnico k for o último classificado no Passo 2, fim do algoritmo. Caso contrário, selecione o próximo técnico k e a tarefa i melhor classificada no Passo 1 e volte para o Passo 3.

A complexidade computacional desta heurística é $O(n^2m)$.

No exemplo apresentado na Seção 2.1, a heurística TS encontra a solução apresentada na Figura 2.5. O Passo 1 da heurística ordena as tarefas do mesmo modo que as heurísticas NT, STT e EST. O segundo passo, ordena os técnicos pelo número de habilidades: no exemplo, como o técnico 2 tem habilidades para 10 tarefas, a sua rota é programada antes da do técnico 1, que tem habilidades para 13 tarefas. Assim, as tarefas 13, 6, 11 e 1 formam a rota deste técnico. Em seguida, a rota do técnico 2 é programada com as tarefas 12, 15, 16 e 3. A Tabela 2.6 apresenta os detalhes da solução. Note que essa heurística programou as mesmas tarefas que a heurística EST, mas com uma rota distinta. O valor da função objetivo é idêntico (63,2) e a diferença em relação à solução ótima, definida pela equação (2.1), é $GAP_{Opt} = 9,9\%$.

Figura 2.5 – Solução da heurística TS para o exemplo

Fonte: elaborado pelo autor.

Tabela 2.6 – Informações das tarefas executadas por cada técnico na solução da heurística TS: a) Técnico 1. b) Técnico 2.

(a)

Tarefa	12	15	16	3	Empresa
Horário de chegada do técnico	8h15min	10h50min	12h05min	14h45min	16h02min
Tempo de espera	1h10min	-	-	-	-
Início da execução	9h25min	10h50min	12h05min	14h45min	-
Fim da execução	10h15min	11h40min	13h55min	15h40min	-

(b)

Tarefa	13	6	11	1	Empresa
Horário de chegada do técnico	11h11min	13h47min	16h20min	17h41min	18h50min
Tempo de espera	29min	38min	-	-	-
Início da execução	11h40min	14h25min	16h20min	17h41min	-
Fim da execução	13h40min	15h45min	17h15min	18h36min	-

2.6 Cluster

A principal ideia desta heurística é fazer um equilíbrio entre a estratégia de explorar as altas concentrações de tarefas prioritárias (programando-as em técnicos com grande aptidão para essas áreas) e os métodos que consideram as tarefas em todas as regiões simultaneamente. Inicialmente a heurística identifica áreas com alta concentração de tarefas e quantifica um valor P para essas áreas. Um certo número de técnicos n_{tec} são designados aos *clusters*, começando pelo de maior valor P . Estes técnicos são escolhidos de acordo com a habilidade para executar as tarefas muito vantajosas de

um *cluster*, isto é, tarefas com grande valor de prioridade por tempo ($\frac{w_i}{p_i}$). Após isso, as outras tarefas são programadas através da heurística construtiva NT; a escolha da heurística construtiva NT foi baseada em testes computacionais que identificaram que a sua combinação com a estratégia *Cluster* resulta em melhores resultados do que a estratégia *Cluster* associada às outras heurísticas construtivas. Essa segunda parte da heurística permite que técnicos destinados a um *cluster* possam realizar algumas tarefas em outros locais, de modo a aumentar o número de tarefas realizadas. Os principais passos da heurística construtiva *Cluster* são descritos a seguir.

1. *Identificar a quantidade e locais dos clusters:* o número de *clusters* selecionados por esta heurística construtiva dependerá do número de tarefas da instância. Se o número total de tarefas (J) é superior a 100, então haverá 10 *clusters*; caso contrário, o número de *clusters* é $\lceil \frac{J}{10} \rceil$. As tarefas são ordenadas em ordem decrescente da razão da prioridade por tempo de processamento ($\mu_i = \frac{w_i}{p_i}$) e a primeira tarefa será o centro do primeiro *cluster*. O centro do segundo *cluster* será a próxima tarefa com mais de $t_{cluster}$ minutos de deslocamento do centro do primeiro *cluster*. O centro do terceiro *cluster* será a próxima tarefa com mais de $t_{cluster}$ minutos de deslocamento do centro do primeiro e do segundo *cluster*, e assim por diante. Caso, após verificar todas as tarefas, não seja possível selecionar um novo centro, então haverá menos que 10 ou $\lceil \frac{J}{10} \rceil$ *clusters*.
2. *Construir os clusters:* para cada tarefa, identifique qual centro é o mais próximo e, se o tempo de deslocamento for inferior a $t_{cluster}$ minutos, então ela pertence a esse *cluster*; caso contrário, a tarefa não pertence a nenhum *cluster*. Calcule o valor P de cada *cluster*, determinado por $P = \sum_{i \in A} \mu_i$, onde A é o conjunto de tarefas do *cluster* considerado que têm proporção de prioridade por tempo acima de um valor pré-estabelecido, ou seja, com $\mu_i \geq \mu_{ideal}$.
3. *Designar n_{tec} técnicos aos clusters:*
 - (a) selecione o *cluster* C com maior valor P e siga para o próximo passo. Se todos os *clusters* têm valor zero, siga para o Passo 4.

- (b) calcule a aptidão ao *cluster* C de cada técnico k ainda não associado a nenhum *cluster*, dado por $\sum_{i \in B} s_{ik} w_i$, onde B é o conjunto de tarefas não programadas do *cluster* C . No entanto, se um técnico não é capaz de executar nenhuma tarefa com $\mu_i \geq \mu_{ideal}$, então a sua aptidão é igual a zero.
- (c) se a aptidão de todos os técnicos é igual a zero, então mude o valor do *cluster* selecionado P para zero e volte ao Passo 3a. Caso contrário, escolha o técnico k com maior aptidão.
- (d) selecione as tarefas do *cluster* com $\mu_i \geq \mu_{ideal}$, ordene essas tarefas em ordem decrescente de μ_i e selecione a tarefa i melhor classificada.
- (e) avalie todas as posições possíveis para a nova tarefa na rota do técnico k . Se não houver solução factível, siga para o próximo passo. Caso contrário, escolha a posição que resulta na solução factível com maior ociosidade após o retorno à base (z_k) e implemente a atribuição.
- (f) se a tarefa i for a última classificada, siga para o próximo passo. Caso contrário, selecione a próxima tarefa i e volte para o Passo 3e.
- (g) se já foram designados aos *clusters* n_{tec} técnicos siga para o Passo 4. Caso contrário, atualize o valor do *cluster* C ($P = \sum_{i \in A} \frac{w_i}{p_i}$, onde A é o conjunto de tarefas não programadas do *cluster* C com $\frac{w_i}{p_i} \geq \mu_{ideal}$) e volte ao Passo 3a.

4. *Programando as demais tarefas e técnicos*: programe as tarefas restantes através da heurística construtiva NT.

Esta heurística construtiva é a que apresenta maior complexidade computacional: $O(n^2 m^2)$.

A seguir encontra-se uma descrição de como a heurística *Cluster* constroi uma solução para o exemplo apresentado na Seção 2.1. No Passo 1, a heurística determina a quantidade de *clusters* igual a 2 ($\lceil \frac{16}{10} \rceil = 2$); ordena as tarefas pela razão $\mu_i = \frac{w_i}{p_i}$ e seleciona as tarefas 11 e 12, respectivamente, como centro do primeiro e do segundo *cluster*. É adotado que as tarefas que estiverem a menos de $t_{cluster} = 30$ minutos do centro de

um *cluster*, pertencem a este *cluster*. Caso uma tarefa esteja a menos de 30 minutos de dois centros, como é o caso da tarefa 1, então ela pertence ao *cluster* que tem o centro mais próximo. Desse modo, no Passo 2, os *clusters* são construídos e o *cluster* 1 terá as tarefas 7, 8, 10 e 11 e o *cluster* 2, as tarefas 1, 2, 3, 4, 6, 9, 12 e 13. O valor do *cluster* 1 é $P_1 = 10,9$ e o valor do *cluster* 2, $P_2 = 38,4$, adotando $\mu_{ideal} = 5$. No Passo 3(a) o *cluster* 2 é selecionado, por ter maior valor P, e no Passo 3(b) o técnico 2 é associado a este *cluster*, pois ele pode realizar até 39 das prioridades do *cluster* ($\sum_{i \in B} s_{i2}w_i = 39$), enquanto que o técnico 1 pode realizar 33 ($\sum_{i \in B} s_{i1}w_i = 33$). O Passo 3(d) determina que, no Passo 3, apenas as tarefas 3, 6, 9, 12 e 13 podem ser programadas no técnico 2, pois essas tarefas têm $\mu_i > \mu_{ideal} = 5$. As tarefas são ordenadas por μ_i resultando na seguinte ordem: 12, 3, 6, 9 e 13. A tarefa 12, por ser a primeira, é selecionada. O Passo 3(e) tenta programar a tarefa, mas como o técnico 2 não tem habilidade para executá-la, não há solução factível. O Passo 3(f) seleciona então a próxima tarefa e o algoritmo volta para o Passo 3(e); a tarefa 3 e as demais são programadas na rota do técnico 2, gerando a sequência 13, 9, 6 e 3. Em seguida, o Passo 3(g) atualiza o valor do *cluster* 2 ($P_2 = 10,8$) e volta para o Passo 3(a), que escolhe agora o *cluster* 1 (que tem maior valor P). Seguindo os mesmos passos, o Técnico 1 é destinado para esse *cluster* ($\sum_{i \in B} s_{i1}w_i = 17$) e a tarefa 11 é programada na rota deste técnico (as demais tarefas têm $\mu_i < \mu_{ideal}$ e não são programadas pelo Passo 3). Após ter destinado os dois técnicos aos *clusters*, o Passo 3 termina. No Passo 4, as tarefas ainda

Figura 2.6 – Solução da heurística Cluster para o exemplo

Fonte: elaborado pelo autor.

Tabela 2.7 – Informações das tarefas executadas por cada técnico na solução da heurística *Cluster*: a) Técnico 1. b) Técnico 2.

(a)					
Tarefa	12	15	16	11	Empresa
Horário de chegada do técnico	8h15min	10h50min	12h05min	14h40min	16h28min
Tempo de espera	1h10min	-	-	20min	-
Início da execução	9h25min	10h50min	12h05min	15h00min	-
Fim da execução	10h15min	11h40min	13h55min	15h55min	-

(b)					
Tarefa	13	9	3	6	Empresa
Horário de chegada do técnico	11h11min	14h22min	15h17min	16h45min	18h16min
Tempo de espera	29min	-	-	-	-
Início da execução	11h40min	14h22min	15h17min	16h45min	-
Fim da execução	13h40min	15h02min	16h12min	18h05min	-

não programadas podem ser incluídas na rota de algum técnico pela heurística NT. A Figura 2.6 apresenta a solução encontrada, que é detalhada na Tabela 2.7. Apenas 8 das 16 tarefas são programadas; a soma das prioridades das tarefas executadas é 66 e o tempo de disponibilidade total dos técnicos após o retorno à base, 2h16min, resultando no valor da função objetivo igual a 66,2 (valor 4,8% superior ao da solução gerada pela heurística XC). A diferença em relação à solução ótima, definida pela equação (2.1), é $GAP_{Opt} = 5,7\%$.

2.7 Outros métodos

Outros métodos foram desenvolvidos, como ordenar as tarefas por outros parâmetros do problema (como o tempo de processamento, início da janela de tempo, número de técnicos capazes de executar cada tarefa, etc.) ou procurar alocar as tarefas por outras estratégias (por exemplo, minimizando os tempos de espera entre as tarefas, devido às janelas de tempo). O desempenho dessas outras heurísticas não foram superiores às seis heurísticas apresentadas nesse capítulo e, por isso, os seus detalhes foram omitidos.

Para exemplificar a estrutura geral das heurísticas construtivas implementadas, o pseudocódigo da heurística STT é apresentado no Apêndice B.

Capítulo 3

Meta-heurísticas

Dada a complexidade computacional (Seção 1.5), uma meta-heurística é proposta para o Problema de Escalonamento de Técnicos de Campo (PETC). Entre as meta-heurísticas mais populares na literatura para resolver problemas combinatórios, estão os Algoritmos Genéticos (AG). Desde 1975, quando surgiram os AG, muitas versões e diferentes operadores (seleção, cruzamento e mutação) foram propostos na literatura. Em 1994, Bean propôs um AG específico para problemas combinatórios – como o sequenciamento de tarefas em máquinas, roteamento de veículos, caixeiro viajante, alocação de recursos, problema da mochila, etc. – que utiliza chaves aleatórias e evita que as novas soluções geradas sejam infactíveis (BEAN, 1994). Denominado *Random-Key Genetic Algorithm* (RKGA), esse AG obteve bons resultados em um número significativo de pesquisas. Recentemente, novos conceitos foram introduzidos, melhorando o desempenho do RKGA. Essa última versão do algoritmo, denominada *Biased Random-Key Genetic Algorithm* (BRKGA), é utilizada nesse trabalho (GONÇALVES; RESENDE, 2011).

A escolha do BRKGA foi baseada nos seus bons resultados em diversos problemas de combinatórios de otimização, como o problema de máquina única (VALENTE; GONÇALVES, 2009), o problema de cobertura (RESENDE et al., 2012), programação de cargas fracionadas (BRANDÃO et al., 2015), dimensionamento de lotes (GONÇALVES; SOUSA, 2011), telecomunicações (DUARTE et al., 2014), determinação de vencedores em leilões (ANDRADE et al., 2015), problema de empacotamento (GONÇALVES; RESENDE, 2013), atracação de navios (LALLA-RUIZ et al., 2014), leiaute (GONÇALVES; RESENDE, 2015) e transportes (BURIOL et al., 2010).

Mais especificamente, o BRKGA obteve bom desempenho ao ser aplicado a problemas similares ao de roteamento. No problema de Atribuição de Rotas e Comprimento

de Ondas (*routing and wavelength assignment* - RWA), os resultados do BRKGA superaram as melhores heurísticas da literatura (*Best Fit Decreasing Heuristic* e *Partition Coloring Problem*), tanto em qualidade quanto em custo computacional (NORONHA; RESENDE; RIBEIRO, 2011). No problema do caixeiro viajante, o BRKGA superou outra meta-heurística em problemas com 48 a 127 nós (MORÁN-MIRABAL; GONZÁLEZ-VELARDE; RESENDE, 2014). Já no problema de coleta de amostras de sangue, Grasas et al. (2014) aplicaram o BRKGA e reduziram de 20% a 30% os custos de logística de dois grandes laboratórios da Espanha, encontrando soluções ótimas na maior parte das instâncias.

Outra motivação para utilizar o BRKGA é que, até onde se sabe, não há na literatura nenhuma pesquisa que aplicou uma meta-heurística populacional ao PETC.

A primeira seção do capítulo contém uma breve descrição dos Algoritmos Genéticos e em seguida é apresentado o BRKGA. A terceira seção apresenta uma estratégia de diversificação da população elite do algoritmo, de modo a prevenir convergências prematuras. O capítulo é encerrado com a descrição dos codificadores e decodificadores propostos para o BRKGA.

3.1 Algoritmos Genéticos

Os algoritmos genéticos (AG) são algoritmos de busca que utilizam o mecanismo de seleção natural – segundo a teoria da origem das espécies, de Charles Darwin (DARWIN, 2003) – e os conceitos da genética. A partir de uma população artificial inicial, novas gerações são formadas e tendem a evoluir, por meio de uma competição no momento da reprodução. Cada indivíduo da população representa uma solução para um problema de otimização e, quanto melhor é a sua qualidade ou maior é a sua aptidão (*fitness*) ao meio ambiente, maior a probabilidade de ser escolhido para gerar descendentes. Após o sorteio dos futuros pais (onde os melhores tendem a ser mais escolhidos), os filhos são gerados mediante os operadores de recombinação cromossômica (*crossover* ou mistura do material genético de dois progenitores) e de mutação (alteração aleatória de um gene). Depois de um número fixo de gerações, ou de algum outro critério de parada, espera-se que a população final tenha convergido para os pontos de ótimo local e global.

A partir de 1950, foram propostos os primeiros algoritmos evolutivos, que tentaram imitar os fenômenos naturais em simulações. Os criadores do AG foram John Holland e

seu grupo de pesquisa na Universidade de Michigan. Em 1962, buscando projetar os sistemas artificiais com os mecanismos evolutivos, Holland criou o conceito de operadores genéticos nesses sistemas. Em 1968, desenvolveu o teorema fundamental do Algoritmo Genético (*theory of schemata*). Alguns anos depois, publica uma monografia – *Adaptation in Natural and Artificial Systems* (1975) – abordando a eficácia do algoritmo em espaços de busca complexos (GOLDBERG; HOLLAND, 1989).

Também em 1975, De Jong, pesquisador da Universidade de Michigan, completou sua dissertação: *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. De Jong analisa o Teorema Fundamental do Algoritmo Genético em vários experimentos, resultando em um importante trabalho sobre o funcionamento dos operadores do algoritmo.

Muitos trabalhos surgiram após as publicações de Holland e De Jong. Uma grande contribuição foi a do pesquisador da Universidade do Alabama, David E. Goldberg. Em seu livro *Genetic algorithms in search, optimization and machine learning* (1989) apresenta uma visão detalhada dos fundamentos do Algoritmo Genético e da sua implementação. Goldberg também aborda algumas técnicas específicas, frutos de seus trabalhos e outros estudos, como Algoritmos Genéticos com processos paralelos de busca, formação de nichos e medidas de similaridade, inserção do conceito de gene dominante e recessivo, inversão de um cromossomo, taxa de natalidade e idade de um cromossomo, diferenciação sexual, etc. (GOLDBERG; HOLLAND, 1989).

Apesar da quantidade de parâmetros, os AG tornaram-se populares pela sua robustez e eficiência. Desse modo, na década de 90 surgiram vários métodos para adaptar o Algoritmo Genético a problemas com restrições, de modo a preservar a factibilidade das soluções (MICHALEWICZ, 2013; COELLO COELLO, 2002).

Com relação aos métodos multi-objetivos – que serão abordados com mais profundidade na segunda parte deste trabalho – a partir do ano 1992 surgiram diversas propostas para adaptar o AG a esses problemas (ZITZLER; DEB; THIELE, 2000; DEB, 1999).

3.2 Biased Random-Key Genetic Algorithm

3.2.1 Revisão Bibliográfica

O Algoritmo Genético com chaves aleatórias, denominado *Random-Key Genetic Algorithm* (RKGA) foi proposto por Bean (1994). Trata-se de um AG específico para problemas de otimização combinatória, como o sequenciamento de tarefas em máquinas (*scheduling*), roteamento de veículos, caixeiro viajante, alocação de recursos, problema da mochila, etc. O RKGA utiliza vetores com chaves aleatórias (números reais entre 0 e 1) para representar os cromossomos. Dessa forma, o espaço de soluções do problema é indiretamente explorado pelas chaves aleatórias (problema independente) através de um decodificador (dependente do problema) (MORÁN-MIRABAL; GONZÁLEZ-VELARDE; RESENDE, 2014). O decodificador é construído de modo a sempre transformar as chaves aleatórias em soluções factíveis, evitando o problema da infactibilidade de soluções. O RKGA foi adotado em diversas pesquisas. Por exemplo, ao aplicar o RKGA ao problema de *job shop*, Gonçalves; Mendes e Resende (2005) encontraram em 72% dos problemas o melhor resultado conhecido na literatura até aquele momento. Esses resultados superaram os obtidos por outros Algoritmos Genéticos, pelo *Simulated Annealing* e pelo *GRASP*, desenvolvidos por diversos autores, conforme cita o artigo.

Alguns anos depois, foram introduzidos novos conceitos a esse AG, aumentando o protagonismo das melhores soluções (as mais aptas) de cada geração. Denominada por Gonçalves e Resende (2011) como *Biased Random-Key Genetic Algorithm* (BRKGA), essa nova versão do RKGA apresentou um desempenho superior. No mesmo artigo, os autores compararam os resultados das duas versões em um problema de cobertura: em 200 processamentos independentes, em até 325 segundos, o BRKGA encontrou a solução ótima 184 vezes (ou 92%), enquanto que o RKGA não encontrou a solução ótima nenhuma vez.

Em outra comparação, Gonçalves; Resende e Toso (2014) utilizaram 12 instâncias de 4 diferentes problemas de cobertura. Cada instância tinha uma *solução alvo* e em 11 das 12 instâncias o BRKGA foi mais eficiente: a probabilidade de o BRKGA encontrar a *solução alvo* antes do RKGA foi de 73%.

Dos artigos da literatura já citados na introdução deste capítulo, pode-se ressaltar dois. Valente e Gonçalves (2009) desenvolveram versões do BRKGA para o problema de máquina única. Os melhores resultados – principalmente em problemas de grandes

dimensões – foram obtidos associando o BRKGA a uma busca local. Para problemas de dimensões pequenas, a maioria dos resultados ótimos foi atingida e em média os resultados ficaram a 0,9% do ótimo. Já Resende et al. (2012) pesquisaram o problema de recobrimento de triplas de Steiner usando o BRKGA. O algoritmo alcançou as soluções ótimas de todas as instâncias em que essas soluções eram conhecidas e, para duas instâncias de grandes dimensões, obteve melhores resultados dos que os conhecidos na literatura.

A seguir encontra-se uma breve descrição do BRKGA.

3.2.2 Descrição do BRKGA

O BRKGA trabalha com cromossomos formados por chaves aleatórias, ou seja, cada gene do cromossomo é um número aleatório entre 0 e 1. Esses cromossomos são então transformados em soluções que sempre são factíveis, evitando assim um problema comum em alguns AG, que é a geração de soluções infactíveis e a necessidade de corrigi-las e torná-las factíveis. O operador que faz a transformação das chaves aleatórias em uma solução factível é denominado decodificador e é um dos elementos mais importantes do BRKGA, uma vez que o desempenho da meta-heurística está diretamente relacionado com o decodificador. Portanto, quanto melhor o decodificador explorar as características do problema e o espaço de busca, melhor poderá ser o resultado obtido.

Diversos decodificadores são propostos para o PETC neste trabalho. A título de exemplo, considere um cromossomo com uma chave aleatória para cada tarefa e um decodificador que constroi uma solução em dois passos: o primeiro associa – a partir das chaves aleatórias – cada tarefa a um técnico que tenha habilidade para executá-la e o segundo, programa a rota de cada técnico a partir das prioridades das tarefas. Concretamente, no primeiro passo, o decodificador utilizará a chave aleatória para fazer um sorteio de um técnico com habilidade: por exemplo, para uma tarefa que tenha dois técnicos possíveis, se a chave aleatória for um número entre 0 e 0,5 então o primeiro técnico é escolhido; caso contrário, se a chave for um número entre 0,5 e 1,0, então o segundo é selecionado.

Para o exemplo da Seção 1.2, considere um cromossomo com as seguintes chaves aleatórias: (0,56; 0,44; 0,97; 0,30; 0,06; 0,23). As tarefas 1, 4 e 5 podem ser realizadas pelos dois técnicos. Como a primeira chave aleatória é maior que 0,5, a tarefa 1 é

associada ao técnico 2 e, como a quarta e a quinta chaves aleatórias são menores do que 0,5, as tarefas 4 e 5 são destinadas para o técnico 1. As demais tarefas são associadas ao único técnico que tem habilidade para executá-las. Desse modo, ao técnico 1 são associadas as tarefas 2, 3, 4 e 5; e ao técnico 2, são designadas as tarefas 1 e 6, como pode ser observado no esquema a seguir.

tarefas	→	1	2	3	4	5	6
técnicos com habilidade	→	1 e 2	1	1	1 e 2	1 e 2	2
cromossomo	→	(0,56	0,44	0,97	0,30	0,06	0,23)
		↓	↓	↓	↓	↓	↓
técnico selecionado	→	2	1	1	1	1	2

O segundo passo irá construir as rotas de cada técnico ordenando as tarefas a ele associadas pela respectiva prioridade e, em seguida, inserindo essas tarefas na posição que resulte na solução factível com maior disponibilidade após retorno à base. Portanto, as tarefas do técnico 1 (tarefas 2, 3, 4 e 5, com prioridades 1, 5, 9 e 4, respectivamente) terão a seguinte ordem: 4, 3, 5 e 2. Após a tarefa 4 ser incluída na rota, é analisada a tarefa 3 e inserida antes da tarefa 4, por não ser factível a sua realização após essa tarefa. Depois a tarefa 5 é inserida entre as tarefas 3 e 4, por ser a única solução factível. A última tarefa não poderá ser realizada, por não haver solução factível e, portanto, o

Figura 3.1 – Solução do exemplo: a) solução decodificada; b) solução ótima.

Tabela 3.1 – Informações das tarefas executadas por cada técnico na solução obtida pelo decodificador: a) Técnico 1. b) Técnico 2.

(a)				
Tarefa	3	5	4	Empresa
Horário de chegada do técnico	8h14min	12h13min	14h10min	16h31min
Tempo de espera	1h46min	-	-	-
Início da execução	10h00min	12h13min	14h10min	-
Fim da execução	11h55min	13h23min	15h55min	-

(b)			
Tarefa	1	6	Empresa
Horário de chegada do técnico	9h42min	12h19min	15h35min
Tempo de espera	-	41min	-
Início da execução	9h42min	13h00min	-
Fim da execução	10h57min	14h30min	-

técnico 1 terá a rota 3, 5 e 4. Analogamente, a rota do técnico 2 é construída, resultando na sequência 1 e 6.

A Figura 3.1 apresenta a solução obtida pelo decodificador e a solução ótima, cujos valores da função objetivo são 36,2 e 37,1, respectivamente. A Tabela 3.1 apresenta os detalhes da solução obtida pelo decodificador. Outros decodificadores mais elaborados foram desenvolvidos para o PETC, que exploram as características do problema e podem obter um melhor desempenho (ver Seção 3.4). A seguir encontra-se a descrição dos outros componentes do BRKGA (formação de uma nova população, cruzamento, elitismo, etc.).

A população de uma geração K é dividida em dois grupos: um primeiro grupo menor com as melhores soluções (elite) e um segundo grupo maior com as demais soluções (não elite). A geração seguinte ($k+1$) é formada por três grupos: todas as soluções de elite são copiadas, algumas novas soluções são geradas aleatoriamente (como na população inicial e com o objetivo de aumentar a diversidade da população) e o restante da população é formado por cruzamentos dos indivíduos da geração K , como pode ser observado na Figura 3.2.

O cruzamento é feito por um *crossover* parametrizado e não pelo tradicional *crossover* em um ou dois pontos (GOLDBERG; HOLLAND, 1989). Dois cromossomos são selecionados: um do grupo de elite ($\mathbf{x}_e$) e outro do grupo não elite ($\mathbf{x}_n$). Um número real aleatório entre 0 e 1 é gerado para selecionar cada gene do novo cromossomo, de

Figura 3.2 – BRKGA: Transição da geração K para a geração K+1. Entre parênteses encontram-se as faixas de porcentagens que cada parte da população pode ter.

Fonte: elaborado pelo autor.

modo que a probabilidade de a solução elite transmitir o seu gene é igual ou maior do que a probabilidade da soluções não elite e, conseqüentemente, a maior parte dos genes de uma nova população venham das soluções elites. A seguir, encontra-se um exemplo de um *crossover* parametrizado. Considerou-se que a solução elite tem 70% de probabilidade de transmitir cada gene e a solução não elite, 30%. Portanto, números aleatórios são gerados no intervalo $[0; 1)$ e se forem menores ou igual a 0,70 o gene da solução de elite é selecionado; caso contrário, a solução não elite transmitirá o seu gene.

número aleatório	→	0,38	0,19	0,88	0,62	0,73
gene escolhido	→	x_e	x_e	x_n	x_e	x_n
x_e	=	(0,14	0,92	0,05	0,31	0,43)
x_n	=	(0,67	0,34	0,02	0,48	0,97)
		↓	↓	↓	↓	↓
nova solução	=	(0,14	0,92	0,02	0,31	0,97)

Gonçalves e Resende (2011) sugerem as seguintes faixas de valores para os parâmetros do BRKGA:

- Tamanho da população: um múltiplo de N , onde N é número de genes do cromossomo.
- Tamanho do conjunto elite: 15 a 25% da população.
- Tamanho do conjunto das novas soluções aleatórias: 5 a 20% da população.
- Probabilidade de a solução de elite transmitir cada gene durante o cruzamento: 60% a 90%.
- Critério de parada: tempo, número de gerações ou número de gerações sem melhorar a melhor solução.

A seguir, encontra-se uma descrição formal do BRKGA, baseado no trabalho de Morán-Mirabal; González-Velarde e Resende (2014). A população inicial P_0 é composta de n_p vetores (ou cromossomos) e cada vetor, $\lambda_j (j = 1, \dots, n_p)$, tem n_c números aleatórios (ou genes) no intervalo $(0, 1]$, representado por $\lambda_j[i] (i = 1, \dots, n_c)$. Um decodificador transforma cada cromossomo λ_j em uma solução factível do problema de otimização e a aptidão $f(\lambda_j)$ é calculada. A população inicial P_0 (e cada uma das populações seguintes P_g) é dividida em dois grupos; as n_e melhores soluções (elite) formam o primeiro grupo (P_g^e) e as restantes (não-elite) são incluídas no segundo conjunto $P_g^{\bar{e}}$, onde $|P_g^e| < |P_g^{\bar{e}}|$. Para formar a geração seguinte ($g + 1$), todos os cromossomos do conjunto elite são copiados para a população P_{g+1} , um conjunto de n_m novas soluções aleatórias é gerado (chamado de soluções mutantes) e o restante dos cromossomos ($n_0 = n_p - n_e - n_m$) são criados através do *crossover* parametrizado entre dois cromossomos de P_g . No *crossover* parametrizado, dois cromossomos são sorteados, um de conjunto P_g^e e outro do conjunto $P_g^{\bar{e}}$. Um número real aleatório entre 0 e 1 é gerado para selecionar cada gene do novo cromossomo: a probabilidade de o cromossomo elite transmitir o seu gene é maior ou igual que a probabilidade do outro cromossomo. Não há mutações após o *crossover*; a diversificação da busca é obtida por meio das soluções mutantes. Uma vez formada a nova população P_{g+1} , todas as operações são repetidas até que o critério de parada seja atingido. O melhor cromossomo da última população é a solução do problema de otimização. O Algoritmo 1 apresenta o pseudocódigo do BRKGA para problemas de maximização.

Algoritmo 1 Pseudocódigo do BRKGA**Entrada:** n_p, n_c, n_e, n_m, p_e

-
- 1: Criar P_0 com n_p cromossomos: incluir soluções provenientes de heurísticas construtivas (se houver) e gerar os vetores restantes com n_c chaves aleatórias $\in (0; 1]$
 - 2: $g = 0$
 - 3: **Enquanto** critério de parada não é satisfeito **faça**
 - 4: Calcule a aptidão (valor da função objetivo) de cada cromossomo de P_g
 - 5: Divida P_g em P_g^e e $P_g^{\bar{e}}$
 - 6: Inicialize a nova população: $P_{g+1} \leftarrow P_g^e$
 - 7: Gere n_m cromossomos mutantes, P_m , com n_c chaves aleatórias $\in (0; 1]$
 - 8: $P_{g+1} \leftarrow P_{g+1} \cup P_m$
 - 9: **Para** $k \leftarrow 1$ a $n_p - n_e - n_m$ **faça**
 - 10: Sorteie um cromossomo a de P_g^e
 - 11: Sorteie um cromossomo b de $P_g^{\bar{e}}$
 - 12: **Para** $i \leftarrow 1$ to n_c **faça**
 - 13: Gere um número aleatório p_{random} entre 0 e 1
 - 14: **Se** $p_{random} \leq p_e$ **então** $c[i] \leftarrow a[i]$
 - 15: **Senão** $c[i] \leftarrow b[i]$
 - 16: **fim Se**
 - 17: **fim Para**
 - 18: $P_{g+1} \leftarrow P_{g+1} \cup \{c\}$
 - 19: **fim Para**
 - 20: $g \leftarrow g + 1$
 - 21: **fim Enquanto**
 - 22: Retorne $\lambda^* \leftarrow \text{argmax}\{f(\lambda_g) | \lambda_g \in P_g\}$
-

3.3 Diversificando a população elite

Nos experimentos realizados e apresentados no Capítulo 5, notou-se que alguns algoritmos convergiam rápida e prematuramente, comprometendo a qualidade dos resultados. De fato, como as soluções das heurísticas construtivas ou as melhores soluções de cada população tendem a gerar filhos com boa aptidão, esses filhos são fortes candidatos a formarem parte da solução elite e, após algumas gerações, é possível que a população elite tenha cromossomos com alto grau de semelhança, levando a busca a talvez não conseguir explorar suficientemente bem outras regiões do espaço.

A seguir encontra-se um exemplo do PETC com 10 tarefas (10 chaves aleatórias, de acordo com o decodificador apresentado na Seção 3.2.2). A Tabela 3.2 apresenta os 15 melhores cromossomos da décima geração do BRKGA, que estão ordenados pelo valor da função objetivo (cada linha corresponde a um cromossomo e cada coluna, os valores das chaves aleatórias correspondentes a uma mesma tarefa); o conjunto elite tradicional é formado pelos 9 primeiros cromossomos, como indica a linha central da tabela.

Tabela 3.2 – Chaves aleatórias na décima geração. Os cromossomos estão ordenados e os 9 melhores formam o conjunto elite

Cromos- somo	Chaves aleatórias									
	1	2	3	4	5	6	7	8	9	10
1	0,633	0,550	0,883	0,650	0,451	0,969	0,324	0,533	0,950	0,284
2	0,633	0,550	0,883	0,997	0,582	0,969	0,324	0,698	0,950	0,284
3	0,633	0,550	0,883	0,650	0,683	0,367	0,017	0,533	0,950	0,433
4	0,633	0,550	0,883	0,650	0,683	0,367	0,017	0,520	0,950	0,433
5	0,633	0,285	0,883	0,191	0,683	0,969	0,017	0,533	0,950	0,476
6	0,633	0,550	0,883	0,650	0,451	0,338	0,324	0,533	0,950	0,284
7	0,460	0,550	0,859	0,650	0,683	0,767	0,837	0,014	0,617	0,318
8	0,783	0,550	0,883	0,650	0,396	0,302	0,017	0,533	0,950	0,476
9	0,460	0,550	0,883	0,650	0,582	0,367	0,017	0,014	0,950	0,433
10	0,460	0,550	0,883	0,650	0,683	0,767	0,017	0,738	0,675	0,433
11	0,633	0,159	0,883	0,243	0,380	0,367	0,017	0,533	0,950	0,786
12	0,633	0,394	0,952	0,668	0,130	0,765	0,528	0,533	0,950	0,476
13	0,312	0,550	0,883	0,650	0,683	0,338	0,528	0,533	0,035	0,476
14	0,633	0,550	0,883	0,650	0,683	0,367	0,324	0,014	0,950	0,746
15	0,312	0,550	0,883	0,271	0,683	0,767	0,017	0,533	0,617	0,318

Note que para algumas tarefas quase não há diversidade nos valores das chaves aleatórias entre os cromossomos. Por exemplo, a tarefa 2 (terceira coluna da Tabela 3.2) tem o mesmo valor (0,550) para 12 dos 15 cromossomos. De um modo geral, para 6 das 10 tarefas ao menos 9 das 15 chaves aleatórias (ou 60%) são iguais (denominamos essas chaves aleatórias de genes dominantes, que estão destacadas em negrito na Tabela 3.2). A situação extrema acontece com as chaves aleatórias da tarefa 3 (coluna 4 da tabela), onde 13 (87%) dos 15 cromossomos têm chaves aleatórias iguais aos outros cromossomos. Se considerarmos apenas o conjunto elite, essas porcentagens de genes dominantes aumentarão. Nas gerações seguintes, com a disseminação dessas chaves aleatórias predominantes, acabarão surgindo indivíduos idênticos (ou *clones*), o que enfraquecerá a capacidade de diversificação do algoritmo e talvez comprometa os resultados finais da busca.

Para evitar que essa predominância de algumas chaves aleatórias leve à convergência prematura, uma medida de similaridade (ou porcentagem de chaves aleatórias iguais) foi criada com o objetivo de diversificar a população elite, de modo que apenas soluções com certo número de chaves aleatórias diferentes façam parte do conjunto elite.

Voltando ao exemplo anterior, pode-se calcular a porcentagem de chaves aleatórias que coincidem em cada combinação de dois cromossomos. A Tabela 3.3 apresenta essas

Tabela 3.3 – Similaridade entre os cromossomos

Cromos- somo	Cromossomo														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	70	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	60	40	-	-	-	-	-	-	-	-	-	-	-	-	-
4	<i>50</i>	40	90	-	-	-	-	-	-	-	-	-	-	-	-
5	<i>50</i>	40	60	<i>50</i>	-	-	-	-	-	-	-	-	-	-	-
6	90	60	60	50	40	-	-	-	-	-	-	-	-	-	-
7	<i>20</i>	10	30	<i>30</i>	<i>10</i>	20	-	-	-	-	-	-	-	-	-
8	<i>50</i>	30	60	<i>50</i>	<i>50</i>	50	<i>20</i>	-	-	-	-	-	-	-	-
9	<i>40</i>	40	60	60	20	40	40	40	-	-	-	-	-	-	-
10	<i>30</i>	20	60	60	30	30	50	40	50	-	-	-	-	-	-
11	<i>40</i>	30	60	<i>50</i>	<i>50</i>	40	<i>0</i>	<i>40</i>	30	20	-	-	-	-	-
12	<i>30</i>	20	30	<i>20</i>	<i>40</i>	30	<i>0</i>	<i>30</i>	10	0	<i>30</i>	-	-	-	-
13	<i>40</i>	20	50	<i>40</i>	<i>40</i>	50	<i>30</i>	<i>50</i>	30	40	<i>20</i>	<i>30</i>	-	-	-
14	60	50	70	70	40	60	40	40	60	40	40	20	40	-	-
15	<i>30</i>	20	50	<i>40</i>	<i>40</i>	30	<i>50</i>	<i>40</i>	20	50	<i>30</i>	<i>10</i>	<i>50</i>	30	-

porcentagens; por exemplo, os cromossomos 1 e 2 têm 7 das 10 chaves aleatórias idênticas e, portanto, têm similaridade de 70% (ver coluna com índice 1 e linha de índice 2 da tabela). Assumindo, por exemplo, que serão aceitos no conjunto elite apenas cromossomos com grau de similaridade de até 55%, os valores superiores a essa porcentagem foram destacados em negrito. As porcentagens que realmente seriam necessárias calcular pelo algoritmo (o que será explicado nos próximos parágrafos) estão em cor azul. Em itálico foram destacados os cromossomos que seriam selecionados para a população elite.

Como os cromossomos já estão ordenados pelo valor da função objetivo, o primeiro passo do algoritmo que selecionará o conjunto elite (chamado de função similaridade) será colocar a melhor solução no conjunto. Em seguida, será selecionado o segundo elemento do conjunto: como os cromossomos 2 e o 3 têm respectivamente 70% e 60% de chaves aleatórias iguais ao cromossomo 1 (mais de 55% de similaridade), eles não serão incluídos na elite e será escolhido o cromossomo 4, que tem 50% de similaridade com o primeiro cromossomo. O passo seguinte será selecionar o terceiro elemento do conjunto: como o cromossomo 5 tem similaridade de 50% com os cromossomos 1 e 4, ele poderá ser incluído no conjunto. E assim por diante. Note que os cromossomos selecionados têm similaridades acima de 55% apenas com os cromossomos não incluídos do conjunto

Tabela 3.4 – Conjunto elite selecionado com similaridade de até 55%

Cromos- somo	Chaves aleatórias									
	1	2	3	4	5	6	7	8	9	10
1	0,633	0,550	0,883	0,650	0,451	0,969	0,324	0,533	0,950	0,284
4	0,633	0,550	0,883	0,650	0,683	0,367	0,017	0,520	0,950	0,433
5	0,633	0,285	0,883	0,191	0,683	0,969	0,017	0,533	0,950	0,476
7	0,460	0,550	0,859	0,650	0,683	0,767	0,837	0,014	0,617	0,318
8	0,783	0,550	0,883	0,650	0,396	0,302	0,017	0,533	0,950	0,476
11	0,633	0,159	0,883	0,243	0,380	0,367	0,017	0,533	0,950	0,786
12	0,633	0,394	0,952	0,668	0,130	0,765	0,528	0,533	0,950	0,476
13	0,312	0,550	0,883	0,650	0,683	0,338	0,528	0,533	0,035	0,476
15	0,312	0,550	0,883	0,271	0,683	0,767	0,017	0,533	0,617	0,318

elite.

A Tabela 3.4 mostra o novo conjunto elite e as suas chaves aleatórias. Com a função similaridade, a tarefa 2 passou a ter três chaves diferentes de 0,550 no conjunto elite, enquanto que as tarefas 3 e 9 passaram a ter duas chaves diferentes de 0,883 e 0,950, respectivamente (no conjunto elite da Tabela 3.2 todas essas tarefas tinham apenas uma chave aleatória distinta). Em termos gerais, entre os 9 primeiros cromossomos da Tabela 3.2 há 33 chaves aleatórias não dominantes (que não estão em negrito) e esse número cresce para 41 na Tabela 3.4: um aumento de 24%.

O Algoritmo 2 apresenta o pseudocódigo da função de *grau de similaridade*. Onde *limit_time* é um tempo predeterminado (critério de parada) suficiente para que o BRKGA atinja a convergência; os outros parâmetros e variáveis são autoexplicativos. O conjunto elite é inicialmente formado pela melhor solução (linha 2) e os outros elementos são selecionados dentro do laço principal (linhas 4 a 29). Dentro deste laço, é necessário decidir se um cromossomo candidato pode ou não ser incluído no conjunto elite (linhas 7 a 21). Com esse propósito, o número de chaves aleatórias iguais entre dois cromossomos é calculado (linhas 13 a 16) e o algoritmo verifica se a porcentagem de chaves aleatórias iguais é superior a *DS* (linhas 17 e 18). Com o objetivo de reduzir o custo computacional do algoritmo, duas condições são verificadas antes de avaliar um cromossomo candidato: primeiro, se o valor da função objetivo do candidato é igual ao valor da do cromossomo anterior, nesse caso o candidato não será incluído no conjunto elite (linhas 7 e 8); segundo, se dois cromossomos pertenciam ao conjunto elite na população anterior, então a similaridade entre eles é menor do que *DS* (linha 11) e não é

Algoritmo 2 Formação do conjunto elite usando a medida de similaridade

Entrada: população atual (P_g), conjunto elite da população anterior (P_{g-1}^e), função objetivo (f), n_p , n_e , n_c , $limit_time$, DS^0

Saída: conjunto elite da população atual (P_g^e)

```

1: Ordene  $P_g$  em ordem decrescente do valor da função objetivo
2: Inclua a melhor solução  $\lambda_1$  em  $P_g^e$  ( $P_g^e \leftarrow \{\lambda_1\}$ )
3:  $j \leftarrow 2$  e  $DS \leftarrow DS^0 + (1 - DS^0) \cdot \frac{current\_time}{limit\_time}$ 
4: Para  $i \leftarrow 2$  a  $n_e$  faça
5:   Enquanto  $\lambda_j \notin P_g^e$  e  $j < n_p$  faça
6:      $decision \leftarrow 0$ 
7:     Se  $f(\lambda_j) = f(\lambda_{j-1})$  então
8:        $decision \leftarrow -1$ 
9:     Senão
10:      Para  $k \leftarrow 1$  to  $j - 1$  faça
11:        Se  $\lambda_k \in P_g^e$  e  $\lambda_j$  ou  $\lambda_k \notin P_{g-1}^e$  então
12:           $counter \leftarrow 0$ 
13:          Para  $q \leftarrow 1$  a  $n_c$  faça
14:            Se  $\lambda_k[q] = \lambda_j[q]$  então  $counter \leftarrow counter + 1$ 
15:          fim Se
16:          fim Para
17:          Se  $\frac{counter}{n_c} > DS$  então  $decision \leftarrow -1$  e  $k \leftarrow j$ 
18:          fim Se
19:        fim Se
20:      fim Para
21:    fim Se
22:    Se  $decision = -1$  então  $j \leftarrow j + 1$ 
23:    Senão  $P_g^e \leftarrow P_g^e \cup \{\lambda_j\}$ 
24:    fim Se
25:  fim Enquanto
26:  Se  $j = n_p$  então  $i \leftarrow n_e$ 
27:  Senão  $j \leftarrow j + 1$ 
28:  fim Se
29: fim Para

```

necessário calcular o número de chaves aleatórias iguais entre eles. Finalmente, poderia ser argumentado que, se o número cromossomos com similaridade menor que DS for pequeno, a cardinalidade do conjunto elite poderia ser menor do que n_e . No entanto, como a porcentagem de chaves aleatórias iguais de P_{g-1}^e é inferior ou igual a DS_{g-1} , $P_{g-1}^e \subset P_g$, e $DS_{g-1} \leq DS_g$, então pode-se garantir que sempre haverá n_e elementos em P_g^e . O capítulo 5 apresentará o impacto da função similaridade nos resultados do BRKGA.

3.4 Decodificadores

Esta seção apresenta quatro decodificadores para o PETC, que resultarão em quatro versões do BRKGA para o problema. As duas primeiras versões do BRKGA estão baseadas nas heurísticas construtivas Menor Tempo de Viagem (STT) e *Cluster*. Na terceira e na quarta versões, as chaves aleatórias são usadas para associar cada tarefa a um técnico; as rotas de cada técnico são construídas a partir das chaves aleatórias (terceiro decodificador) ou a partir das prioridades das tarefas, melhorando-as por meio de uma busca em vizinhança (quarto decodificador).

3.4.1 Decodificando com a heurística Menor Tempo de Viagem (*STT*)

Esta versão do BRKGA representa cada solução como um vetor (λ) composto por n elementos que associam uma chave aleatória a uma tarefa. A população inicial contém, além dos cromossomos gerados aleatoriamente, a solução encontrada pela heurística construtiva STT (Seção 2.3), que é codificada pelo seguinte valor em cada chave aleatória i :

$$\lambda[i] = \frac{\rho_i}{\rho_{max}} \quad (3.1)$$

onde ρ_i é definido pela equação (2.2), $\rho_{max} = \max\{\rho_i | i = 1, \dots, n\} + \nu$ e ν é um número pequeno.

Para decodificar cada vetor, ordene as tarefas em ordem decrescente do valor da chave aleatória ($\lambda[i]$); a primeira tarefa classificada será avaliada em todos os técnicos possíveis e um técnico será selecionado, de acordo com os Passos 2 e 3 da heurística STT (descritos na Seção 2.3); esses procedimentos serão repetidos para a segunda, terceira e demais tarefas, até a última tarefa classificada.

3.4.2 Decodificando com a heurística *Cluster*

Na heurística construtiva *Cluster* alguns técnicos são designados para realizar tarefas em *clusters* específicos e, em seguida, as tarefas restantes são programadas considerando todas as tarefas simultaneamente. Seguindo essa mesma ideia, o decodificador *cluster*

tem n chaves aleatórias para representar a prioridade das n tarefas e mais uma chave (a última do vetor) para determinar a porcentagem de técnicos associados aos *clusters*. A população inicial do BRKGA é formada por vetores aleatórios e pela solução encontrada pela heurística construtiva *Cluster*, que é codificado de acordo com a fórmula a seguir:

$$\lambda[i] = \begin{cases} 0.9 + 0.1 \cdot \frac{\mu_i - \mu_{ideal}}{\mu_{max} - \mu_{ideal}}, & \text{se } \mu_i \geq \mu_{ideal} \\ 0.9 \cdot \frac{\rho_i}{\rho_{max}}, & \text{caso contrário} \end{cases}$$

onde ρ_i é definido por (2.2), $\mu_i = \frac{w_i}{p_i}$, $\mu_{max} = \max\{\mu_i | i = 1, \dots, n\} + \nu$ e ν é um número muito pequeno.

Um vetor λ é decodificado usando os passos da heurística construtiva *Cluster* (descrita na Seção 2.6). Nos Passos 1 e 2, os parâmetros μ_i e μ_{ideal} são substituídos por $\lambda[i]$ e 0.9 (testes preliminares indicaram que é melhor fixar μ_{ideal} em 0.9 do que acrescentar uma nova chave aleatória e deixar o algoritmo genético escolher um valor para esse parâmetro). O número de técnicos destinados aos *clusters* no Passo 3 é $n_{tec} = \lceil \lambda[n+1] \cdot m \rceil$. O decodificador termina com o Passo 4, que programa as tarefas restantes através da heurística NT, descrita na Seção 2.2 (as tarefas serão ordenadas pelo valor da chave aleatória e programadas de acordo com os Passos 2 a 4 da heurística construtiva).

3.4.3 Decodificador Aleatório (A)

Nessa versão do BRKGA, cada cromossomo (λ) contém n chaves aleatórias, que são utilizadas para selecionar um técnico para cada tarefa e depois construir a rota de cada técnico. As tarefas são designadas para os técnicos da seguinte maneira: se há dois técnicos capazes de realizar uma tarefa e a chave aleatória for um número no intervalo $(0; 1/2]$, então ela é associada ao primeiro técnico; caso seja um número entre $(1/2; 1]$, é designada ao segundo. Um procedimento análogo é realizado quando uma tarefa tem 3 ou mais técnicos aptos. Uma vez distribuídas todas as tarefas, para evitar a necessidade de uma nova chave aleatória por tarefa para determinar a rota de cada técnico, é extraído um índice ψ de cada chave aleatória, que é a porcentagem de distância entre o valor da chave aleatória e o início do subintervalo: concretamente, se no exemplo anterior a chave aleatória tem valor 0,25, então ela está exatamente na metade do subintervalo $[0; 1/2)$ e tem índice de 50%. Desse modo, as tarefas de cada técnico são ordenadas em ordem crescente do índice ψ e, da primeira à última, são introduzidas na rota do

técnico.

Portanto, o técnico selecionado para cada tarefa é definido por:

$$TC_i = \lceil \lambda[i] \cdot NST_i \rceil \quad (3.2)$$

onde TC_i é o n -th técnico com habilidade para executar a tarefa i e NST_i é o número de técnicos capazes de executar a tarefa i .

O índice (ψ_i) é calculado:

$$\psi_i = \frac{\lambda[i] - \frac{TC_i - 1}{NST_i}}{\frac{1}{NST_i}} = NST_i \cdot \lambda[i] - TC_i + 1 \quad (3.3)$$

e as tarefas serão classificadas em ordem crescente do índice (ψ_i) . Começando da primeira até a última tarefa, todas as posições possíveis na rota do técnico serão avaliadas e a posição que resultar na solução factível com maior ociosidade (z_k) será escolhida. Se não houver solução factível (por conflitos de janelas de tempo, etc.) para introduzir uma tarefa na rota, então ela não é realizada.

Além dos vetores aleatórios, são incluídas na população inicial as soluções codificadas das melhores heurísticas construtivas (STT, NT e Cluster). O seguinte valor é calculado para as respectivas chaves aleatórias:

$$\lambda[i] = \frac{TC_i - \frac{\rho_i}{\rho_{max}}}{NST_i} \quad (3.4)$$

Se essas heurísticas não associarem alguma tarefa a nenhum técnico, então um técnico com habilidade é sorteado e o valor da chave é dado por:

$$\lambda[i] = \frac{TC_i - \frac{\nu}{\rho_{max}}}{NST_i} \quad (3.5)$$

onde ν é um número muito pequeno. Dessa forma, essas tarefas ocupam as últimas posições quando as tarefas de cada técnico forem ordenadas pelo índice ψ .

Para ilustrar o decodificador Aleatório, considere um problema com 7 técnicos e 10 tarefas. A Tabela 3.5 apresenta as habilidades de cada técnico para cada tarefa

Tabela 3.5 – Habilidades de 7 técnicos para 10 tarefas

Tarefa	Técnico						
	1	2	3	4	5	6	7
1	1	0	1	1	1	1	1
2	1	1	1	0	1	0	1
3	0	1	0	0	0	0	0
4	1	1	1	1	1	1	1
5	1	0	1	1	1	0	1
6	0	0	1	0	0	1	1
7	0	1	1	1	1	1	1
8	1	1	0	1	1	1	1
9	1	1	0	1	0	1	1
10	1	1	1	1	1	1	1

(habilidade igual 1 se é capaz de realizá-la e 0, caso contrário). A Tabela 3.6 apresenta o número de técnicos com habilidade para cada tarefa (segunda linha), as chaves aleatórias de um cromossomo (terceira linha), o n -ésimo técnico com habilidade que será associado a essa tarefa (quarta linha), o número do técnico escolhido (quinta linha) e os valores do índice ψ para cada tarefa (última linha). Neste exemplo, o técnico 6 recebe as tarefas 4, 6 e 7, que possuem índices ψ iguais a 0,67, 0,65 e 0,56, respectivamente. Portanto, as tarefas são incluídas na rota deste técnico na seguinte ordem: 7, 6, 4 (para cada tarefa, todas as posições possíveis são testadas e é escolhida a posição na rota que resultar na maior ociosidade z).

Tabela 3.6 – Decodificador para o exemplo de 7 técnicos e 10 tarefas

Tarefa	1	2	3	4	5	6	7	8	9	10
# técnicos com habilidade	6	5	1	7	5	3	6	6	5	7
Chave aleatória	0,65	0,43	0,91	0,81	0,96	0,55	0,76	0,9	0,13	0,41
Técnico escolhido	4º	3º	1º	6º	5º	2º	5º	6º	1º	3º
# Técnico escolhido	5	3	2	6	7	6	6	7	1	3
Índice ψ	0,90	0,15	0,91	0,67	0,80	0,65	0,56	0,40	0,65	0,87

3.4.4 Decodificador Busca em Vizinhança (BV)

O quarto decodificador desenvolvido para o PETC também tem uma chave aleatória para cada tarefa, que associa uma tarefa a um técnico da mesma maneira que o decodificador Aleatório (através da equação (3.2)). Para programar as tarefas de cada técnico, elas são ordenadas pela prioridade w e nessa ordem são inseridas na rota. Uma

vez construída a rota, uma busca em vizinhança tenta melhorá-la. O esquema a seguir apresenta os passos deste decodificador:

1. *Inicialização*: Associe cada uma das tarefas a um técnico, de acordo com a equação (3.2). Selecione o primeiro técnico $k = 1$.
2. *Ordenar as tarefas*: Ordene as tarefas do técnico k por ordem decrescente de prioridade (w). Selecione a primeira tarefa i da *fila* e faça $f_0 = 0$ e $j = 0$.
3. *Programar as tarefas*:
 - (a) avalie todas as posições possíveis para a tarefa i na rota do técnico k . Escolha a posição que resultar na solução factível com maior ociosidade após o retorno à base (z_k).
 - (b) se a tarefa i for a última da *fila*, siga para o próximo passo. Caso contrário, selecione a próxima tarefa i e volte ao Passo 3a.
4. Faça $j = j + 1$. Calcule o valor da função objetivo da solução construída no passo anterior (f_j). Se $f_j > f_0$, faça $f_0 = f_j$ e $j = 0$.
5. *Busca em vizinhança*:
 - (a) se $j > 5$, siga para o Passo 6. Caso contrário, siga para o Passo 5b.
 - (b) entre as tarefas associadas ao técnico k , sorteie duas tarefas distintas (bl_1 e bl_2) e diferentes das tarefas sorteadas na busca em vizinhança anterior. Troque a posição dessas tarefas na *fila*¹. Selecione a primeira tarefa i da nova *fila* e volte ao Passo 3a.
6. *Critério de parada*: se as rotas de todos os técnicos já foram programadas, fim do algoritmo. Caso contrário, selecione o próximo técnico k e volte ao Passo 2.

A população inicial desta versão do BRKGA é formada por vetores aleatórios e pelas soluções das três melhores heurísticas construtivas (STT, NT e Cluster), codificadas pelas equações 3.4 e 3.5.

¹Note que, na primeira busca em vizinhança, as tarefas sorteadas são trocadas na *fila* inicial (Passo 2). Na segunda busca em vizinhança, na *fila* da primeira busca em vizinhança. Na terceira busca em vizinhança, são trocadas as tarefas da *fila* da segunda busca em vizinhança. E assim por diante.

3.4.5 Outros decodificadores

Outros decodificadores foram desenvolvidos ao longo deste trabalho: baseado na heurística construtiva Técnico mais próximo (NT), usando enumerações completas para encontrar a melhor rota para um técnico (em vez da busca em vizinhança), etc. Como os resultados dos testes computacionais não apresentaram um desempenho superior aos decodificadores apresentados anteriormente, essas outras propostas foram omitidas.

Capítulo 4

Limitantes Superiores (*Upper Bound*)

Conforme apresentado na Seção 1.5, o Problema de Escalonamento de Técnicos de Campo é NP-difícil e, por isso, espera-se encontrar soluções ótimas apenas para instâncias de pequenas dimensões. Para avaliar os resultados obtidos através das heurísticas construtivas e das meta-heurísticas para os problemas de médias e de grandes dimensões, convém obter limitantes superiores das instâncias e calcular a porcentagem média da diferença entre o valor obtido pelos métodos heurísticos e pelos limitantes superiores.

A seguir encontram-se algumas propostas de como relaxar o modelo de Programação Linear Inteira Mista (PLIM) apresentado na Seção 1.4 para obter limitantes superiores. As duas primeiras propostas estão baseadas no trabalho de Xu e Chiu (2001). A terceira foi desenvolvida neste trabalho e a última trata-se de uma abordagem clássica em problemas de combinatórios.

Outros modelos foram desenvolvidos, na tentativa de melhorar os resultados obtidos pelos quatro modelos apresentados neste capítulo (por exemplo, modelos intermediários entre o 2 e o 3, mesclando as suas restrições). Como esses modelos não apresentaram um desempenho superior, eles foram omitidos neste trabalho.

4.1 Modelo zero: sem tempos de deslocamentos e janelas de tempo

A proposta mais simples para calcular os limitantes superiores para o PETC é desconsiderar os tempos de deslocamento e as janelas de tempo das tarefas. O problema ficará, portanto, reduzido ao seguinte modelo:

Parâmetros:

- J Conjunto de tarefas, $i = 1, \dots, n$;
- K Conjunto de técnicos, $k = 1, \dots, m$;
- p_i Tempo necessário para a execução da tarefa i ;
- a_k Horário de início do turno do técnico k ;
- b_k Horário de término do turno do técnico k ;
- w_i Prioridade ou peso da tarefa i ;
- s_{ik} Habilidade do técnico k para realizar a tarefa i .

Variáveis:

- y_{ik} Assume 1 se a tarefa i é atribuída ao técnico k e 0, caso contrário.

Formulação:

$$\max \sum_{i \in J} \sum_{\substack{k \in K \\ |s_{ik}=1}} \frac{w_i y_{ik}}{MW} + \frac{1}{MZ} \cdot \left(\sum_{k \in K} (b_k - a_k) - \sum_{i \in J} \sum_{\substack{k \in K \\ |s_{ik}=1}} p_i y_{ik} \right) \quad (4.1)$$

Sujeito a:

$$\sum_{\substack{k \in K \\ |s_{ik}=1}} y_{ik} \leq 1 \quad i \in J \quad (4.2)$$

$$\sum_{\substack{i \in J \\ |s_{ik}=1}} p_i y_{ik} \leq b_k - a_k \quad k \in K \quad (4.3)$$

$$y_{ik} \in \{0, 1\} \quad i \in J, k \in K, s_{ik} = 1 \quad (4.4)$$

Onde:

$$MW = \min_{i \in J} w_i \quad (4.5)$$

$$MZ = \max_{k \in K} (b_k - a_k) + \varepsilon \quad (4.6)$$

ε é um número muito pequeno

O primeiro termo da função objetivo é semelhante ao modelo PLIM original do PETC e considera a soma ponderada das tarefas realizadas. Os dois termos seguintes consideram a soma dos horários de trabalho dos técnicos subtraído do tempo de execução

das tarefas designadas, sendo um valor superior ao da soma dos tempos ociosos dos técnicos no fim do dia.

Pelas restrições (4.2), cada tarefa será alocada a, no máximo, um técnico, evitando designar mais de uma pessoa para a mesma tarefa. O conjunto de restrições (4.3) garantem que a soma dos tempos de processamento das tarefas designadas a cada técnico não superará o seu tempo de trabalho diário. As demais restrições são idênticas às apresentadas na Seção 1.4. Pode-se notar que este modelo é semelhante ao Problema de Múltiplas Mochilas.

Esse modelo foi proposto por Xu e Chiu (2001) e será denominado UB_0 .

4.2 Modelo 1: reintroduzindo tempos de deslocamento e janelas de tempo

Para melhorar o modelo anterior, procurar-se-á nessa segunda proposta reintroduzir o conceito de janelas de tempo das tarefas e de tempo de deslocamento dos técnicos.

Para simplificar o conceito de janela de tempo, determinou-se um conjunto E , que define se duas tarefas i e j podem ser executadas pelo mesmo técnico ou se, pelas janelas de tempo, essas tarefas são *incompatíveis* ou *na prática simultâneas* e, portanto, devem ser executadas por diferentes técnicos. Desse modo, o conjunto E será definido da seguinte maneira:

$$E = \{(i, j) | i \neq j \in J, e_i + p_i + c_{ij} > l_j - p_j \text{ e } e_j + p_j + c_{ji} > l_i - p_i\} \quad (4.7)$$

Para simplificar o tempo de viagem, para cada técnico k um tempo mínimo prévio à execução da tarefa i foi calculado (denominado δ_{ik}), que pode ser o tempo real de deslocamento ou o tempo de espera (diferença entre as janelas de tempo da tarefa i e de seus potenciais predecessores). Os potenciais predecessores de uma tarefa i são a base da empresa e as tarefas j que o técnico k pode executar ($s_{jk} = 1$) antes da tarefa i . Dessa forma, o tempo prévio da base será $\tau_{0ik} = \max\{c_{0i}, e_i - a_k\}$ e o tempo prévio de uma tarefa j ($1 \leq j \leq n$) será:

$$\tau_{jik} = \begin{cases} \max\{c_{ji}, e_i - l_j\}, & \text{se } s_{ik} = s_{jk} = 1 \text{ e } e_j + p_j + c_{ji} \leq l_i - p_i \\ b_k - a_k, & \text{caso contrário} \end{cases} \quad (4.8)$$

E assim, δ_{ik} será definido por:

$$\delta_{ik} = \min\{\tau_{jik} \mid 0 \leq j \leq n, j \neq i\} \quad (4.9)$$

Por último, para determinar o tempo mínimo de retorno dos técnicos até a base, foi definido um vetor Φ :

$$\phi_k = \min\{c_{0i} \mid i \in J, s_{ik} = 1\} \quad (4.10)$$

Dessa forma, chegou-se ao seguinte modelo:

$$\max \sum_{i \in J} \sum_{k \in K \mid s_{ik}=1} \frac{w_i y_{ik}}{MW} + \frac{1}{MZ} \cdot \left(\sum_{k \in K} (b_k - a_k) - \sum_{i \in J} \sum_{k \in K \mid s_{ik}=1} (p_i + \delta_{ik}) y_{ik} \right) \quad (4.11)$$

Sujeito a:

$$\sum_{\substack{k \in K \\ |s_{ik}=1}} y_{ik} \leq 1 \quad i \in J \quad (4.12)$$

$$\sum_{\substack{i \in J \\ |s_{ik}=1}} (p_i + \delta_{ik}) y_{ik} + \phi_k \leq b_k - a_k \quad k \in K \quad (4.13)$$

$$y_{ik} + y_{jk} \leq 1 \quad (i, j) \in E, k \in K, s_{ik} = s_{jk} = 1 \quad (4.14)$$

$$y_{ik} \in \{0, 1\} \quad i \in J, k \in K, s_{ik} = 1 \quad (4.15)$$

A função objetivo deste modelo (4.11) é diferente da do modelo da seção anterior apenas pelo fato de ter sido incluído o parâmetro δ_{ik} . O parâmetro ϕ_k não foi incluído na função objetivo para evitar a inclusão de uma nova variável, que indicaria se um técnico deixou ou não a base. As restrições (4.13) garantem que a soma dos tempos de processamento, dos tempos mínimos para chegar em cada tarefa e dos tempos mínimos para retornar à base não superem os horários de trabalho de cada técnico. Quando duas tarefas são *incompatíveis*, esse par pertence ao conjunto E e (4.14) assegura que elas não serão destinadas ao mesmo técnico. As demais restrições são iguais às do modelo anterior.

Esse segundo modelo está baseado em uma proposta de *upper bound* de Xu e Chiu (2001), na qual foram feitas as seguintes melhorias:

- Nas condições do conjunto E foram incluídas as distâncias entre as tarefas: $e_i + p_i + c_{ij} > l_j - p_j$ e $e_j + p_j + c_{ji} > l_i - p_i$, enquanto que Xu e Chiu (2001) consideraram apenas as janelas de tempo para determinar se duas tarefas não podem pertencer ao mesmo técnico ($e_i + p_i > l_j - p_j$ e $e_j + p_j > l_i - p_i$).
- Para determinar o mínimo tempo de deslocamento e/ou de espera imediatamente anterior à tarefa i no técnico k , aqueles autores consideraram a combinação de todas as tarefas que não pertenciam ao conjunto E , enquanto que no modelo desta seção foi verificado se a tarefa j realmente pode ser executada antes de i ($e_j + p_j + c_{ji} \leq l_i - p_i$).
- A variável ϕ_k (o tempo mínimo necessário para retornar à base) foi acrescentada ao modelo.
- Foram geradas as variáveis y_{ik} somente quando $s_{ik} = 1$.

Esse modelo será denominado UB_1 .

4.3 Modelo 2: excluindo combinações infactíveis

Neste terceiro modelo, em vez das restrições (4.13) e (4.14) do modelo anterior, outra abordagem será proposta para representar os tempos de deslocamentos e as janelas de tempo. Ao invés das distâncias mínimas para reduzir o tempo disponível do técnico para realizar as tarefas (restrição (4.13)), para cada técnico k foi feita uma enumeração completa de todas as tarefas possíveis (ou seja, $s_{ik} = 1$) e, entre as soluções factíveis, determinado o maior valor da soma dos tempos de processamento das tarefas $pmax_k$ (considerando todos os parâmetros do modelo original, como janelas de tempo, tempos de deslocamentos, etc.). Esse mesmo resultado poderia ser obtido resolvendo para cada técnico k o modelo de PLIM original com a função objetivo $\max \sum_{i \in J} p_i y_{ik}$. Analogamente, foi determinado pwm_k , que é a maior soma de tempo de processamento ponderado possível ($w_i p_i$) entre as soluções factíveis. Dessa forma, chegou-se às restrições (4.18) e (4.19) do modelo a seguir.

Para excluir soluções infactíveis, novas restrições foram incluídas para cada instância. Concretamente, para cada técnico foram analisados todos os grupos de uma, duas, três e quatro tarefas e determinado – novamente através de enumerações completas – para

quais dessas combinações não há soluções factíveis (por violar as restrições de janelas de tempo). As restrições (4.20) a (4.23) foram incluídas no modelo para eliminar esses grupos de tarefas para cada técnico. Uma abordagem similar à deste modelo pode ser encontrada no trabalho de Ascheuer; Fischetti e Grötschel (2001).

A seguir encontra-se a formulação do terceiro modelo proposto.

$$\max \sum_{i \in J} \sum_{\substack{k \in K \\ |s_{ik}=1}} \frac{w_i y_{ik}}{MW} + \frac{1}{MZ} \cdot \left(\sum_{k \in K} (b_k - a_k) - \sum_{i \in J} \sum_{\substack{k \in K \\ |s_{ik}=1}} (p_i + \delta_{ik}) y_{ik} \right) \quad (4.16)$$

Sujeito a:

$$\sum_{\substack{k \in K \\ |s_{ik}=1}} y_{ik} \leq 1 \quad i \in J \quad (4.17)$$

$$\sum_{\substack{i \in J \\ |s_{ik}=1}} p_i y_{ik} \leq pmax_k \quad k \in K \quad (4.18)$$

$$\sum_{\substack{i \in J \\ |s_{ik}=1}} w_i p_i y_{ik} \leq pwmax_k \quad k \in K \quad (4.19)$$

$$y_{ik} \leq 0 \quad (i, k) \in A_1 \quad (4.20)$$

$$y_{i_1 k} + y_{i_2 k} \leq 1 \quad (i_1, i_2, k) \in A_2 \quad (4.21)$$

$$y_{i_1 k} + y_{i_2 k} + y_{i_3 k} \leq 2 \quad (i_1, i_2, i_3, k) \in A_3 \quad (4.22)$$

$$y_{i_1 k} + y_{i_2 k} + y_{i_3 k} + y_{i_4 k} \leq 3 \quad (i_1, i_2, i_3, i_4, k) \in A_4 \quad (4.23)$$

$$y_{ik} \in \{0, 1\} \quad i \in J, k \in K \quad (4.24)$$

Onde:

$$A_1 = \{(i, k) | i \in J, k \in K, s_{ik} = 1, a_k + c_{0i} + p_i > l_i \text{ ou } e_i + p_i + c_{i0} > b_k\}$$

$$A_2 = \{(i_1, i_2, k) | i_1, i_2 \in J, k \in K, s_{i_1 k}, s_{i_2 k} = 1, (i_1, k), (i_2, k) \notin A_1, \text{todas as sequências de } i_1 \text{ e } i_2 \text{ são infactíveis para o técnico } k\}$$

$$A_3 = \{(i_1, i_2, i_3, k) | i_1, i_2, i_3 \in J, k \in K, s_{i_1 k}, s_{i_2 k}, s_{i_3 k} = 1, (i_1, k), (i_2, k), (i_3, k) \notin A_1, (i_1, i_2, k), (i_1, i_3, k), (i_2, i_3, k) \notin A_2 \text{ e todas as sequências de } i_1, i_2 \text{ e } i_3 \text{ são infactíveis no técnico } k\}$$

$$A_4 = \{(i_1, i_2, i_3, i_4, k) | i_1, \dots, i_4 \in J, k \in K, s_{i_1 k}, \dots, s_{i_4 k} = 1, (i_1, k), \dots, (i_4, k) \notin A_1, (i_1, i_2, k), \dots, (i_3, i_4, k) \notin A_2, (i_1, i_2, i_3, k), \dots, (i_2, i_3, i_4, k) \notin A_3 \text{ e todas as sequências de } i_1, i_2, i_3 \text{ e } i_4 \text{ são infactíveis no técnico } k\}$$

Esse modelo será denominado UB_2 .

4.4 Modelo 3: relaxando as variáveis inteiras do modelo de PLIM

O quarto modelo é uma abordagem clássica em otimização, onde as variáveis x e y do modelo original são relaxadas (Seção 1.4), definindo-as como reais em vez de binárias ($0 \leq x, y \leq 1$). Esse modelo será denominado UB_3 .

Capítulo 5

Experimentos Computacionais

Este capítulo apresenta os resultados dos experimentos computacionais realizados para o Problema mono-objetivo de Escalonamento de Técnico de Campo. A primeira seção descreve como as instâncias para o problema foram geradas; em seguida é apresentado os resultados obtidos pelo modelo de Programação Linear Inteira Mista (Seção 5.2), pelos quatro modelos de limitantes superiores (Seção 5.3), pelas heurísticas construtivas (Seção 5.4) e pelas diferentes versões do BRKGA (Seção 5.5). O capítulo termina com uma análise dos principais resultados obtidos (Seção 5.6).

Todos os testes foram realizados em um computador com processador 2.93 GHz Intel(R) 870 e memória RAM de 16GB. Os códigos das heurísticas e meta-heurísticas foram escritos em linguagem C.

5.1 Instâncias geradas

Para realizar os experimentos computacionais, 1040 instâncias foram geradas¹, abrangendo uma considerável diversidade de parâmetros que possam influenciar nos resultados dos métodos heurísticos: a distribuição geográfica dos clientes, o tamanho das janelas de tempo, a porcentagem de tarefas com janelas de tempo, os tempos de execução e as prioridades das tarefas, o número total de tarefas e de técnicos. Quatro diferentes distribuições geográficas de tarefas foram adotadas: distribuição uniforme aleatória (R), distribuição em *clusters* (C), distribuição *semi-cluster* (RC) e distribuição radial (RAD). As três primeiras distribuições foram propostas por Solomon (1987) e a última

¹ Como as instâncias geradas por Xu e Chiu (2001) não estão disponibilizadas na Internet, entrou-se em contato com os autores. Infelizmente, eles responderam que não era possível localizá-las ou reunir todas as informações necessárias para reconstruí-las.

é uma nova distribuição proposta neste trabalho, onde há uma maior concentração de tarefas na parte central de uma cidade, que vai reduzindo cada vez mais quando se afasta do centro em direção à periferia. Para cada uma das 4 distribuições geográficas, 20 instâncias foram geradas com 13 diferentes quantidades de tarefas e técnicos (ou 13 casos apresentados na Tabela 5.1), resultando em 1040 instâncias.

Tabela 5.1 – Dimensão das instâncias geradas

Caso	#tarefas (n)	#técnicos (m)
1	16	2
2	26	2
3	30	3
4	39	3
5	45	7
6	64	5
7	80	13
8	100	10
9	120	20
10	150	25
11	200	33
12	500	83
13	999	166

Os parâmetros das instâncias foram gerados através de uma distribuição uniforme discreta:

- Prioridade das tarefas (w_i): 1 (baixa), 2, ..., 10 (alta).
- Tempo de execução das tarefas (p_i): 30, 35, ..., 120 minutos.
- Porcentagem de tarefas com janelas de tempo²: 25, 50, 75 ou 100%.
- Início da janela de tempo das tarefas (e_i): 7, ..., 19 horas.
- Tamanho das janelas de tempo das tarefas: 2, ..., 8 horas.
- Fim da janela de tempo de uma tarefa (l_i): e_i mais o tamanho da janela de tempo.
- Início das janelas de tempo dos técnicos (a_k): 7,0; 7,5; 8,0; ..., 12,0 horas.
- Fim da janela de tempo dos técnicos (b_k): $a_k + 9$ horas.

² Em algumas instâncias, a porcentagem de tarefas com janelas de tempo é inferior a 100% e, portanto, algumas tarefas não terão janelas de tempo, podendo ser executadas em qualquer momento do dia.

- Habilidade de um técnico k executar a tarefas i (s_{ik}): 0 ou 1.
- Tempo de deslocamento: todas as tarefas estão localizadas em uma região onde o tempo de deslocamento entre quaisquer dois locais é menor ou igual a 1,5 hora.

O gerador de números aleatórios de Taillard (1993) e as sementes sugeridas pelo mesmo autor foram utilizados para gerar os números aleatórios para a distribuição uniforme discreta e determinar os parâmetros descritos acima. A distribuição geográfica radial (parâmetro contínuo) também foi gerada pelo gerador proposto por Taillard.

5.2 Soluções do modelo de PLIM

Usando o software ILOG CPLEX, versão 12.6, com tempo de execução máximo de uma hora, foi resolvido o modelo PLIM apresentado na Seção 1.4. O valor ótimo foi encontrado em 52, 10, 6 e 4 instâncias dos casos 1, 2, 3 e 4, respectivamente. Nos casos 5 a 7, o CPLEX encontrou soluções factíveis para todas as instâncias e, nos casos 8 a 13, nenhuma solução factível foi obtida. A Tabela 5.2 apresenta a diferença percentual entre o limitante inferior (LB_{cplex}) e o limitante superior (UB_{cplex}) da função objetivo obtidos pelo CPLEX, i.e.:

$$Diff\% = 100 \cdot \frac{UB_{cplex} - LB_{cplex}}{UB_{cplex}} \quad (5.1)$$

A primeira coluna da tabela identifica o caso. A segunda, a terceira e a quarta colunas mostram, respectivamente, os valores mínimo, médio e máximo da diferença percentual em cada caso. Pelos resultados da Tabela 5.2, nota-se que a diferença entre os limitantes inferiores e superiores dependem da razão tarefa por técnico ($\frac{n}{m}$) e, quanto

Tabela 5.2 – Diferença percentual ($Diff\%$) média em cada caso

Caso	Min.	Média	Max.	$\frac{n}{m}$
1	0,0	5,2	25,8	8
2	0,0	33,3	50,6	13
3	0,0	24,9	37,5	10
4	0,0	38,2	51,6	13
5	0,0	11,9	20,7	6
6	0,0	42,0	52,8	13
7	1,0	20,8	33,7	6

maior é a razão, maior é a diferença percentual. Por outro lado, para proporções similares de tarefas por técnico, o percentual médio aumenta com o crescimento das dimensões das instâncias. Este comportamento era esperado pois, como mencionado na Seção 1.5, o PETC é um problema NP-difícil.

5.3 Limitantes superiores

Com o objetivo de avaliar a qualidade dos modelos propostos de limitantes superiores (Capítulo 4), os resultados do Modelo 1 (UB_1), do Modelo 2 (UB_2) e do Modelo 3 (UB_3) foram comparados com os resultados do Modelo 0 (UB_0). Todos os modelos foram resolvido usando o software CPLEX 12.6 com tempo de processamento máximo de uma hora.

Para os Casos 1 a 4 ($n < 40$), uma comparação com soluções ótimas também foi realizada. O modelo PLIM descrito na Seção 1.4 foi resolvido para as instâncias desses casos com o tempo de resolução limitado a dez horas (partindo da melhor solução obtida pelas meta-heurísticas). Também foi realizada uma enumeração completa, para aumentar o número de soluções ótimas encontradas. Com isso, encontrou-se o valor ótimo para todas as instâncias dos casos 1 e 2, 60 do caso 3 e 18 do caso 4, totalizando 238 instâncias com valores ótimos conhecidos.

A Tabela 5.3 apresenta o tempo médio de CPU utilizado para resolver os modelos UB_0 , UB_1 , UB_2 e UB_3 ; o número de vezes em que cada método encontrou o melhor limitante superior ($\#melhor$) e a diferença percentual entre o valor do limitante superior ($UB_{\#}$) (onde $\#$ representa o número do modelo do limitante superior) e o valor ótimo, i.e.:

$$GAP_{Opt} = 100 \cdot \frac{Opt - UB_{\#}}{Opt} \quad (5.2)$$

Os números em **negrito** na Tabela 5.3 e nas próximas tabelas indicam o melhor resultado para cada linha. Como se pode observar na tabela, para todos os casos considerados, os modelos (UB_1) e (UB_2) superaram os resultados obtidos pelo modelo (UB_0). Já com relação ao (UB_3), os resultados foram inferiores aos do modelo (UB_0). Numa comparação entre os Modelos 1 e 2, pode-se observar o desempenho claramente superior do Modelo 2, tanto no número de vezes em que encontrou o melhor limitante

Tabela 5.3 – Comparação entre os modelos de limitantes superiores e os valores ótimos

Caso	UB_0			UB_1			UB_2			UB_3		
	CPU (s)	GAP_{Opt}	#melhor	CPU (s)	GAP_{Opt}	#melhor	CPU (s)	GAP_{Opt}	#melhor	CPU (s)	GAP_{Opt}	#melhor
1	0,2	-20,6	0	0,3	-10,7	3	0,3	-3,5	80	0,1	-27.8	0
2	0,3	-28,3	0	0,3	-15,9	1	0,6	-8,0	80	0,1	-68.4	0
3	0,4	-26,5	0	0,5	-15,4	0	1,0	-6,5	80	0,1	-42.8	0
4	0,7	-36,0	0	1,0	-20,2	0	3,3	-9,3	80	0,1	-73.0	0
Média	0,4	-27,8		0,5	-15,6		1,3	-6,8		0,1	-53.0	

Tabela 5.4 – Comparação entre os modelos propostos de limitantes superiores e o UB_0 .

Caso	UB_0		UB_1			UB_2			UB_3		
	CPU (s)	#melhor	CPU (s)	$Diff_{UB_0}$	#melhor	CPU (s)	$Diff_{UB_0}$	#melhor	CPU (s)	$Diff_{UB_0}$	#melhor
5	0,1	0	144,1	1,9	1	396,5	3,8	80	0,1	-2,5	0
6	148,2	0	674,8	6,7	0	1086,0	11,8	80	0,1	-25,5	0
7	0,2	0	10,8	1,4	14	152,3	1,9	73	0,2	-2,4	0
8	1260,7	0	3326,5	4,3	0	1703,3	7,5	80	0,2	-11,8	0
9	0,3	0	0,9	0,9	80	-	-	-	0,4	-2,5	0
10	0,3	0	1,1	0,6	80	-	-	-	0,9	-2,5	0
11	0,6	0	2,3	0,5	80	-	-	-	2,3	-2,5	0
12	3,5	0	54,8	0,2	80	-	-	-	-	-	-
13	16,8	18	335,2	0,1	80	-	-	-	-	-	-
Média	159,0		505,6	1,8		834,5	6,3		0,6	7,1	

superior (320 vezes no Modelo 2 e 4 vezes no Modelo 1) quanto na diferença percentual média com relação ao valor ótimo.

Para os Casos 5 a 13, foi realizada uma comparação entre os modelos de limitantes superiores. A Tabela 5.4 apresenta o tempo médio de CPU utilizado para resolver cada um dos modelos, o número de vezes que cada modelo encontrou o melhor limitante e a diferença percentual média entre os modelos UB_1 , UB_2 ou UB_3 e UB_0 ($Diff_{UB_0} = 100 \cdot \frac{UB_0 - UB_{\#}}{UB_0}$). Para os Casos 9 a 13, o CPLEX não foi capaz de resolver o Modelo 2, provavelmente devido ao significativo aumento do números de restrições deste modelo³. Como esperado, os resultados da tabela indicam uma relação entre a qualidade dos limitantes obtidos e os respectivos tempos computacionais. O Modelo 2 encontra os melhores limitantes, mas não pôde ser resolvido nas instâncias dos casos 9 a 13. Já o Modelo 1 pôde resolver todas as instâncias e obter limitantes de boa qualidade; repare que os limitantes obtidos por este modelo são sempre melhores ou iguais aos do UB_0 e com um tempo de execução razoável. Assim como nas instâncias pequenas, o Modelo 3 obteve um despenho inferior aos dos demais. Também se pode ressaltar que, ao considerar as 640 instâncias que foram resolvidas por todos os modelos, os limitantes obtidos pelo Modelo 2 foram estritamente melhores em 621 instâncias.

5.4 Heurísticas construtivas

Inicialmente alguns parâmetros da heurística construtiva Cluster foram calibrados. Após alguns testes computacionais, os melhores resultados foram obtidos com os seguintes parâmetros: $\mu_{ideal} = 10$, $t_{cluster} = 12$ minutos e $n_{tec} = 0.8m$. As demais heurísticas construtivas não têm parâmetros a serem calibrados⁴.

Para analisar o desempenho das heurísticas construtivas, foram utilizados cinco indicadores descritos a seguir:

1. **Relação de Desempenho Individual (RDI):** para cada resultado, é calculada a razão:

$$RDI = \frac{M - H}{M - P} \quad (5.3)$$

³Por essa mesma razão, na resolução dos casos 7 e 8 as restrições (4.23) foram omitidas.

⁴No cálculo dos parâmetros μ e ρ foram adotadas horas como unidade para o tempo de processamento p .

onde M representa o melhor valor da função objetivo obtido pelas heurísticas que serão comparadas; P , o pior valor obtido e H , o valor da função objetivo obtido pela heurística analisada. Desse modo, baixos valores médios indicam heurísticas com bom desempenho e altos valores, métodos ineficientes (KIM, 1993).

2. **Número de Instâncias com Melhor Desempenho (NIMD):** uma segunda medida é o número de vezes (número de experimentos) em que o método encontrou a heurística resultado entre todos os métodos comparados.
3. **Diferença percentual com relação à heurística XC ($Diff_{XC}$):** o desempenho dos métodos desenvolvidos são comparados com a heurística construtiva XC pela seguinte relação:

$$Diff_{XC} = 100 \cdot \frac{H - H_{XC}}{H_{XC}} \quad (5.4)$$

onde H é o valor da função objetivo das heurísticas desenvolvidas e H_{XC} é o valor da função objetivo da heurística construtiva XC.

4. **Gap ótimo:** Para as instâncias em que a solução ótima é conhecida (casos 1 a 4), a diferença percentual entre o valor ótimo (Opt) e o valor obtido pelos métodos heurísticos (H) é calculada:

$$GAP_{Opt} = 100 \cdot \frac{Opt - H}{Opt} \quad (5.5)$$

5. **Gap dos limitantes superiores:** Os resultados das heurísticas construtivas são avaliados pela diferença entre o melhor limitante superior (UB) e o valor da solução (SV) obtida pelo método analisado:

$$GAP_{UB} = 100 \cdot \frac{UB - SV}{UB} \quad (5.6)$$

Na Tabela 5.5 são apresentadas as RDI médias e na Tabela 5.6, os NIMD das 80 instâncias de cada caso. As últimas linhas das tabelas apresentam a média geral e o número total, respectivamente. As medidas *RDI* e *NIMD* indicam que as três

Tabela 5.5 – RDI médio das heurísticas construtivas

Caso	XC	NT	STT	EST	TS	Cluster
1	0,48	0,29	0,33	0,72	0,36	0,34
2	0,46	0,38	0,39	0,82	0,44	0,33
3	0,48	0,37	0,34	0,80	0,38	0,30
4	0,58	0,32	0,35	0,71	0,51	0,35
5	0,53	0,19	0,31	0,91	0,42	0,24
6	0,49	0,32	0,31	0,84	0,51	0,24
7	0,45	0,17	0,20	0,96	0,48	0,16
8	0,40	0,20	0,27	0,91	0,65	0,16
9	0,42	0,13	0,17	0,98	0,50	0,14
10	0,37	0,11	0,11	0,96	0,54	0,12
11	0,33	0,11	0,11	0,95	0,57	0,06
12	0,25	0,11	0,03	0,91	0,64	0,08
13	0,23	0,13	0,01	0,89	0,72	0,10
Média	0,42	0,22	0,23	0,87	0,52	0,20

heurísticas com melhor desempenho são: Cluster, NT e STT. A heurística Cluster obtém os melhores resultados médios até o caso 11, enquanto que a STT alcança resultados significativamente melhores para as instâncias grandes (com 500 e 999 tarefas). Repare que, em ambas as medidas, essas três heurísticas superam a XC tanto no valor médio geral quanto no desempenho médio de cada um dos casos. Essa superioridade pode ser

Tabela 5.6 – NIMD das heurísticas construtivas

Caso	XC	NT	STT	EST	TS	Cluster
1	13	29	36	8	26	26
2	20	24	25	3	19	34
3	9	20	21	3	19	23
4	8	24	16	5	14	24
5	9	28	14	1	11	17
6	13	20	16	1	8	24
7	3	23	18	0	5	31
8	7	18	21	1	1	32
9	2	24	19	0	2	33
10	4	25	22	0	1	28
11	1	20	17	0	2	40
12	0	13	52	0	1	14
13	0	3	67	0	0	10
Total	89	271	344	22	109	336

Tabela 5.7 – Porcentagem (%) de melhoria em relação a heurística XC

Caso	NT	STT	EST	TS	Cluster
1	2,2	2,0	-2,3	1,3	1,6
2	1,0	1,2	-3,0	0,4	1,6
3	1,0	1,2	-3,2	1,0	1,8
4	3,0	2,5	-0,7	1,1	2,6
5	2,1	1,5	-3,1	0,7	1,9
6	1,5	1,6	-2,8	0,2	2,2
7	1,7	1,6	-3,5	0,0	1,7
8	1,5	1,1	-3,9	-1,7	1,9
9	1,7	1,5	-3,7	-0,2	1,7
10	1,5	1,4	-3,7	-0,7	1,5
11	1,4	1,3	-4,1	-1,1	1,6
12	0,8	1,2	-4,2	-2,1	1,0
13	0,6	1,2	-3,9	-2,6	0,8
Média	1,5	1,5	-3,2	-0,3	1,7

ressaltada, por exemplo, em uma comparação apenas entre as heurísticas construtivas Cluster e XC, onde os valores médios de RDI (e total de $NIMD$) são 0,20 (833) e 0,79 (219), respectivamente.

A Tabela 5.7 apresenta diferença percentual com relação à heurística XC ($Diff_{XC}$). As colunas apresentam os resultados médios de cada heurística para cada um dos casos e a última linha, a média geral. Os resultados confirmam que as heurísticas NT, STT e Cluster têm em média um desempenho superior à heurística XC em todos os casos. Entre as instâncias pequenas e médias (casos 1 a 11) a Cluster obteve a melhor média (1,8%) e nos problemas, com mais de 500 técnicos (casos 12 e 13), a STT alcançou o melhor desempenho (1,2%).

A média do GAP_{Opt} para as melhores heurísticas foi de 6,5% (NT), 6,6% (STT) e 6,4% (Cluster), conforme pode ser observado na Tabela 5.8. Note que os melhores resultados são obtidos pelas heurísticas NT (casos 1 e 4) e pela Cluster (casos 2 e 3), que têm melhor desempenho nas instâncias de pequenas dimensões.

A Tabela 5.9 apresenta os valores médio e desvio padrão do GAP_{UB} considerando as 80 instâncias de cada caso em cada uma das heurísticas construtivas propostas. Com se esperava, as heurísticas construtivas NT, STT e Cluster superam o o GAP_{UB} médio da heurística XC em todos os casos. A heurística Cluster obtém o melhor GAP_{UB} médio e a melhor média em 8 dos 13 casos. Nos dois maiores casos (instâncias com

Tabela 5.8 – *Gap* de otimalidade das heurísticas construtivas

Caso	XC	NT	STT	EST	TS	Cluster
1	6,7	4,8	5,1	9,0	5,6	5,4
2	7,5	6,7	6,5	10,4	7,2	6,2
3	8,3	7,4	6,9	11,3	7,3	6,6
4	9,1	7,1	7,8	11,6	8,1	7,3
Média	7,9	6,5	6,6	10,6	7,0	6,4

500 e 999 tarefas) a heurística STT obtém os melhores GAP_{UB} médio: 3,8 e 2,9%, respectivamente. Trata-se de um resultado vantajoso, uma vez que o tempo de execução máximo dessa heurística construtiva é de 0,21 segundo. Os tempos máximo de execução das outras heurísticas são: 0,16 (XC), 0,17 (NT), 0,27 (EST), 0,17 (TS) e 0,35 segundo (Cluster).

5.5 BRKGA

5.5.1 Calibragem dos parâmetros do BRKGA

Inicialmente, os seguintes parâmetros do BRKGA foram calibrados: tamanho da população, porcentagem de soluções mutantes, probabilidade do *crossover* parametrizado, tamanho da população e número máximo de gerações. De acordo com Gonçalves e Resende (2011), os seguintes valores foram considerados para cada parâmetro:

- Tamanho do conjunto elite (n_e): 15%, 20% e 25% da população.
- Novas soluções mutantes (n_m): 5%, 10%, 15% e 20% da população.
- Probabilidade do *crossover* parametrizado (p_e): 50%, 60%, 70% e 80%.
- Tamanho da população (n_p): até 1500 cromossomos.

Vários testes foram realizados para calibrar o parâmetro grau de similaridade (DS). Os melhores resultados foram obtidos com um valor inicial (DS^0), que cresce linearmente ao longo das gerações (soluções muito similares são aceitas nas últimas gerações). O parâmetro DS^0 foi calibrado dentro do intervalo de 50% a 90%.

Tabela 5.9 – Média e Desvio Padrão (σ) do GAP_{UB} das heurísticas construtivas

Caso	XC		NT		STT		EST		TS		Cluster	
	Média	σ	Média	σ	Média	σ	Média	σ	Média	σ	Média	σ
1	9,8	4,5	8,0	4,4	8,2	5,0	12,1	4,9	8,7	5,0	8,5	4,5
2	14,3	5,3	13,5	4,5	13,4	4,9	16,9	5,0	14,0	5,4	13,0	5,2
3	13,5	3,9	12,7	4,0	12,6	4,0	16,4	4,5	12,7	4,0	12,0	4,0
4	17,8	5,0	15,5	4,5	15,9	4,4	18,6	4,0	17,0	4,5	15,8	4,8
5	11,0	3,1	9,2	3,0	9,7	3,1	13,8	3,7	10,4	3,0	9,4	2,7
6	19,0	3,3	17,7	3,7	17,7	3,7	21,3	3,8	18,9	3,9	17,3	3,7
7	9,9	3,1	8,4	2,8	8,5	2,6	13,0	3,7	10,0	2,4	8,4	3,0
8	17,6	3,1	16,4	2,7	16,8	2,8	20,8	3,6	19,0	2,8	16,1	2,8
9	8,6	3,2	7,0	2,9	7,2	2,9	11,9	3,9	8,8	2,3	7,0	2,9
10	7,7	3,3	6,4	2,6	6,4	2,6	11,1	3,7	8,4	2,0	6,4	2,6
11	7,4	3,3	6,2	2,9	6,3	3,1	11,2	4,0	8,5	2,3	5,9	2,8
12	5,0	2,6	4,2	2,2	3,8	2,2	9,0	3,2	7,0	1,5	4,1	2,2
13	4,0	2,3	3,4	2,0	2,9	2,0	7,7	3,1	6,5	1,4	3,3	1,9
Média 1–4	13,9	4,7	12,4	4,3	12,5	4,6	16,0	4,6	13,1	4,7	12,3	4,6
Média 5–13	10,0	3,0	8,8	2,7	8,8	2,8	13,3	3,6	10,8	2,4	8,7	2,7

Tabela 5.10 – Parâmetros calibrados para cada versão do BRKGA.

	BRKGA-STT	BRKGA-Cluster	BRKGA-A	BRKGA-BV
n_e	15%	25%	15%	15%
n_m	15%	15%	10%	5%
p_e	60%	60%	70%	80%
DS^0	80%	60%	55%	70%

Diferentes combinações dos valores dos parâmetros foram testadas em 104 instâncias (10% das instâncias de cada caso) nas quatro versões do BRKGA. A Tabela 5.10 apresenta a configuração dos parâmetros n_e , n_m , p_e e DS^0 que resultaram nos melhores resultados médios para cada versão e que serão adotados nos próximos testes computacionais. O tamanho da população (n_p) adotado foi de 1000 cromossomos (para os casos 1 a 11), 500 (caso 12) e 200 (caso 13). Note que, na medida em que aumentam as dimensões das instâncias, o custo da decodificação de cada cromossomo cresce significativamente. Por este motivo, adotou-se um valor menor para o tamanho da população nos casos 12 e 13, de modo a garantir um tempo de execução aceitável. O número necessário de gerações para cada versão do BRKGA atingir a “convergência” foi determinado. Neste trabalho, considerou-se que quando o crescimento do valor das melhores soluções encontradas é inferior a 0,01% nas últimas gerações, então o algoritmo atingiu a “convergência”. A Tabela 5.11 apresenta o número de gerações necessárias para cada

Tabela 5.11 – Número de gerações necessárias para atingir a convergência e tempo de execução (critério de parada) para cada caso.

Caso	Número de gerações				<i>limit_time</i> (s)
	BRKGA-STT	BRKGA-Cluster	BRKGA-A	BRKGA-BV	
1	60	20	20	30	3
2	50	30	30	40	5
3	170	260	360	320	10
4	70	280	770	400	50
5	800	1100	3000	1200	75
6	700	900	1800	1000	80
7	700	1300	4200	1500	180
8	900	1100	3900	1300	250
9	800	1700	7100	2000	480
10	700	1500	6700	1800	600
11	600	1200	7300	2000	900
12	300	200	8000	2200	1800
13	300	200	11000	3000	3600

versão do BRKGA e, na última coluna, o tempo necessário (*limit_time*) para que todas as versões atinjam a convergência.

Com relação à diversificação da população elite, a Figura 5.1 ressalta a diferença entre o desempenho do BRKGA com a elite tradicional e com o conjunto elite formado pela função de similaridade. A diferença média $Diff_{CH}$ de oito instâncias do Caso 8 (com 100 tarefas e 10 técnicos) é calculada por:

$$Diff_{CH} = 100 \cdot \frac{BR - CH}{CH} \quad (5.7)$$

onde BR e CH são os valores obtidos pelo BRKGA e pela heurística construtiva Cluster, respectivamente. A Figura 5.1(a) mostra que o $Diff_{CH}$ médio do BRKGA-A cresce de 7,3% para 10,5% usando a função similaridade. Já no BRKGA-STT, como pode ser observado na Figura 5.1(b), a diferença com relação à heurística construtiva Cluster aumenta de 10,8% para 11,4%. Observe que o BRKGA-STT com a função similaridade continua encontrando melhores soluções após 100 segundos, enquanto que o BRKGA-STT com a elite tradicional não melhora o $Diff_{CH}$ após 100 segundos. O comportamento do BRKGA-BV é similar ao do BRKGA-A (com a variação do $Diff_{CH}$ de 8,4% para 10,7%) e o do BRKGA-Cluster, idêntico ao do BRKGA-STT (com a variação do $Diff_{CH}$ de 10,9% para 11,0%) e seus gráficos foram omitidos.

O desempenho da função similaridade foi analisado também em 140 instâncias com números de tarefas entre 45 e 200 (Casos 5 a 11). Todas as distribuições geográficas de tarefas foram consideradas: R, C, RC e RAD. Em média, quando utilizada a função similaridade do conjunto elite, os resultados melhoram em 2% (BRKGA-A), 1,5% (BRKGA-BV), 0,3% (BRKGA-STT) e 0,2% (BRKGA-Cluster). Ao mesmo tempo, a porcentagem de instâncias com resultados estritamente melhores do que os resultados obtidos pelo BRKGA com a elite tradicional é de 100% (BRKGA-A), 95% (BRKGA-BV), 79% (BRKGA-STT) e 71% (BRKGA-Cluster). Desse modo, pode-se concluir que a função similaridade de fato melhora o desempenho do BRKGA, especialmente do BRKGA-A e do BRKGA-BV, que apresentam uma maior tendência à convergência prematura.

Figura 5.1 – Desempenho do BRKGA com a função similaridade e a elite tradicional:
a) BRKGA-A; b) BRKGA-STT.

Fonte: elaborado pelo autor.

5.5.2 Desempenho do BRKGA

Para comparar as versões do BRKGA, um tempo predeterminado (critério de parada) foi fixado (*limit_time*), suficiente para que todas essas versões atinjam a convergência (já apresentado na Tabela 5.11). Todas as instâncias foram resolvidas uma única vez por cada BRKGA (BIRATTARI, 2004; HUSSIN; STÜTZLE, 2014).

As Tabelas 5.12 e 5.13 apresentam a RDI e o NIMD entre as quatro versões do BRKGA. As duas medidas indicam um desempenho superior do BRKGA-STT e do BRKGA-Cluster, sendo que o primeiro tem um desempenho superior nas instâncias menores (casos 1 a 6) e maiores (casos 12 e 13) e o segundo, nas instâncias de tamanho

Tabela 5.12 – RDI médio das quatro versões do BRKGA

Caso	BRKGA-STT	BRKGA-Cluster	BRKGA-A	BRKGA-BV
1	0,00	0,00	0,00	0,00
2	0,00	0,01	0,01	0,14
3	0,00	0,03	0,08	0,18
4	0,04	0,06	0,19	0,42
5	0,14	0,17	0,60	0,65
6	0,25	0,26	0,47	0,60
7	0,10	0,09	0,74	0,87
8	0,27	0,19	0,75	0,76
9	0,12	0,04	0,83	0,87
10	0,10	0,02	0,82	0,92
11	0,13	0,05	0,79	0,95
12	0,03	0,24	0,78	0,97
13	0,00	0,72	0,63	0,90
Média	0,09	0,15	0,52	0,63

médio (casos 7 a 11).

Outra medida analisada foi a diferença entre os resultados obtidos pelas versões do BRKGA e os resultados da melhor heurística construtiva (Cluster), calculada pela equação (5.7). A Tabela 5.14 apresenta os resultados médios. O BRKGA-STT e o BRKGA-Cluster obtiveram o melhor desempenho médio (5,9%), seguido do BRKGA-A

Tabela 5.13 – NIMD das quatro versões do BRKGA

Caso	BRKGA-STT	BRKGA-Cluster	BRKGA-A	BRKGA-BV
1	80	80	80	80
2	80	78	79	68
3	79	77	71	64
4	75	74	64	45
5	55	49	23	16
6	49	43	28	20
7	34	45	0	2
8	31	44	3	7
9	20	59	0	1
10	19	60	0	1
11	29	51	0	0
12	62	18	0	0
13	79	1	0	0
Total	692	679	348	304

Tabela 5.14 – Melhoria média (em %) de cada versão do BRKGA em relação à melhor heurística construtiva (*Cluster*)

Caso	BRKGA-STT	BRKGA-Cluster	BRKGA-A	BRKGA-BV
1	5,9	5,9	5,9	5,9
2	6,8	6,8	6,8	6,7
3	6,9	6,9	6,9	6,9
4	8,6	8,6	8,5	8,4
5	7,0	6,9	6,7	6,7
6	9,2	9,2	9,0	8,8
7	6,3	6,3	5,7	5,5
8	9,0	9,1	8,2	8,0
9	5,3	5,4	4,5	4,4
10	4,8	4,9	3,8	3,6
11	4,0	4,1	3,0	2,7
12	2,1	1,8	1,3	1,1
13	1,4	0,8	0,8	0,6
Média	5,9	5,9	5,5	5,3

(5,5%) e do BRKGA-BV (5,3%).

Um estudo comparativo com as soluções ótimas foi realizado. A Tabela 5.15 mostra na segunda coluna o número de soluções ótimas encontradas pelos métodos exatos (*#Opt.*) e, nas colunas seguintes, o número de soluções ótimas encontradas por cada BRKGA e o *gap* de otimalidade médio (calculado por (5.5)) entre as instâncias que não atingiram o valor ótimo. Como pode ser observado na última linha da tabela, o BRKGA-STT e o BRKGA-Cluster encontraram, respectivamente, 99% e 97% das soluções ótimas e o BRKGA-Cluster obteve o menor GAP_{Opt} médio para as outras instâncias (0,21%).

Para os casos médios e grandes, a qualidade da solução das heurísticas foi analisada através de uma comparação com o melhor limitante superior obtido, como pode ser observado na Tabela 5.16. As colunas 2 a 4 mostram, respectivamente, os valores mínimo, médio e o desvio padrão do *gap* de otimalidade (GAP_{UB}) dos resultados do BRKGA-STT; as colunas seguintes apresentam os resultados do BRKGA-Cluster, do BRKGA-A e do BRKGA-BV; as últimas três colunas apresentam os valores mínimo, médio e o desvio padrão do GAP_{UB} dos valores ótimos conhecidos (em itálico). Os números em negrito na tabela indicam os melhores valores encontrados para cada caso entre todas as versões do BRKGA.

Tabela 5.15 – Comparação entre os resultados do BRKGA e os valores ótimos conhecidos.

Caso	# <i>Opt.</i>	BRKGA-STT		BRKGA-Cluster		BRKGA-A		BRKGA-BV	
		# <i>Opt.</i>	<i>GAP</i> _{<i>Opt</i>}	# <i>Opt.</i>	<i>GAP</i> _{<i>Opt</i>}	# <i>Opt.</i>	<i>GAP</i> _{<i>Opt</i>}	# <i>Opt.</i>	<i>GAP</i> _{<i>Opt</i>}
1	80	79	0,02	79	0,02	79	0,02	79	0,02
2	80	78	0,58	76	0,30	77	0,76	66	0,45
3	60	60		60		59	0,68	52	0,11
4	18	18		17	0,05	16	0,38	13	0,30
Total (Média)	238	235	0,40	232	0,21	231	0,53	210	0,31

Tabela 5.16 – Média, Mínimo (Min.) e Desvio Padrão (σ) do GAP_{UB} dos métodos heurísticos e das soluções ótimas conhecidas.

Case	BRKGA-STT			BRKGA-Cluster			BRKGA-A			BRKGA-BV			Ótimo		
	Min.	Média	σ	Min.	Média	σ	Min.	Média	σ	Min.	Média	σ	Min.	Média	σ
1	0,3	3,3	2,5	0,3	3,3	2,5	0,3	3,3	2,5	0,3	3,3	2,5	0,3	3,3	2,5
2	0,4	7,4	3,1	0,4	7,4	3,1	0,4	7,4	3,1	0,4	7,4	3,1	0,4	7,3	3,1
3	0,4	6,0	2,7	0,4	6,0	2,7	0,4	6,1	2,7	0,4	6,1	2,7	0,4	6,0 ^a	2,7
4	0,3	8,7	2,8	0,3	8,7	2,8	0,3	8,8	2,8	0,3	8,9	2,9	0,3	8,4 ^b	3,0
5	0,5	3,1	1,4	0,5	3,2	1,4	0,5	3,4	1,5	0,5	3,3	1,4	-	-	-
6	0,6	9,7	2,5	0,6	9,7	2,4	0,6	9,9	2,5	0,6	10,0	2,5	-	-	-
7	0,5	2,7	1,2	0,5	2,7	1,2	0,9	3,3	1,3	0,6	3,4	1,3	-	-	-
8	2,0	8,6	2,1	1,7	8,5	2,1	2,2	9,3	2,1	1,0	9,4	4,0	-	-	-
9	0,4	2,2	1,2	0,4	2,1	1,2	0,7	2,9	1,3	0,7	3,0	1,3	-	-	-
10	0,4	1,9	1,1	0,4	1,8	1,1	0,6	2,9	1,3	0,6	3,0	1,4	-	-	-
11	0,4	2,2	1,5	0,4	2,1	1,5	0,8	3,2	1,8	0,8	3,4	1,7	-	-	-
12	0,4	2,1	1,3	0,4	2,3	1,6	0,8	2,8	1,6	0,7	3,0	1,7	-	-	-
13	0,4	1,9	1,6	0,5	2,5	1,7	0,6	2,5	1,8	0,5	2,8	1,9	-	-	-
Média 1-4 (%)	0,4	6,4	2,8	0,4	6,4	2,8	0,4	6,4	2,8	0,3	6,4	2,8	0,4	6,3	2,8
Média 5-13 (%)	0,6	3,8	1,5	0,6	3,9	1,6	0,9	4,5	1,7	0,7	4,6	1,9	-	-	-

^a resultados médios entre 60 instâncias.^b resultados médios entre 18 instâncias.

Para problemas pequenos, como se esperava, em quase todos os casos os valores mínimo, médio e o desvio padrão do GAP_{UB} das versões do BRKGA e dos valores ótimos são similares. Para os casos 5 a 13, o GAP_{UB} médio do BRKGA-STT e do BRKGA-Cluster foram 3,8% e 3,9%, respectivamente, o que sugere um bom desempenho dessas versões do BRKGA. Observe que esses problemas podem ser divididos em duas categorias: baixa e alta proporção de tarefas por técnico, i.e., instâncias com 6 a 8 e 10 a 13 tarefas por técnico. No primeiro grupo (casos 5, 7, 9, 10, 11, 12 e 13), o valor médio do GAP_{UB} é 2,3% para o BRKGA-STT e 2,4% para o BRKGA-Cluster, enquanto no segundo grupo (casos 6 e 8) o valor médio do GAP_{UB} é 9,2% e 9,1%, respectivamente. Este fato provavelmente está relacionado com a qualidade inferior dos limitantes superiores quando a proporção de tarefas por técnico cresce (como pode ser observado na Tabela 5.3). Nas instâncias em que o valor ótimo é conhecido, os problemas com proporção maior ou igual a 10 (casos 2 a 4) apresentam um *gap* médio de 7,3%, enquanto que os problemas com baixa proporção (caso 1) têm um *gap* médio de 3,3%.

Outra análise que pode ser mencionada é que o BRKGA apresentou um desempenho semelhante para os diferentes valores dos parâmetros adotados. A seguir encontra-se um comentário sobre o desempenho do BRKGA nas diferentes distribuições geográficas de tarefas e em problemas com diferentes tamanhos de janelas de tempo.

A Tabela 5.17 apresenta a média da diferença entre os limitantes superiores e os valores obtidos para cada distribuição geográfica de tarefas (R, C, RC e RAD), para as instâncias pequenas (casos 1 a 4) e para as instâncias médias e grandes (casos 5 a 13). O desempenho das versões do BRKGA nos problemas pequenos é semelhante, como mostra a Tabela 5.17(a). Para os problemas médios e grandes, nota-se que o comportamento das versões do BRKGA para cada distribuição geográfica de pontos coincide com o comportamento médio apresentado na Tabela 5.16, ou seja, as duas melhores versões superam as demais em cerca de 0,7%. Note também que o GAP_{UB} para a distribuição RAD é um pouco superior ao GAP_{UB} das demais distribuições em todas as versões do BRKGA; no entanto, acredita-se que isso seja devido à qualidade dos limitantes superiores, pois para os problemas pequenos o BRKGA-STT encontrou todas as soluções ótimas conhecidas com distribuição RAD e esses valores ótimos apresentam GAP_{UB} médio superior em relação às demais distribuições geográficas de tarefas.

Tabela 5.17 – Valor médio do GAP_{UB} de cada versão do BRKGA para as diferentes distribuições geográficas de tarefas. a) instâncias pequenas (casos 1 a 4). b) instâncias médias e grandes (casos 5 a 13).

(a)				
Distribuição	BRKGA-STT	BRKGA-Cluster	BRKGA-A	BRKGA-BV
R	6,8	6,8	6,8	6,9
C	4,9	4,9	4,9	5,0
RC	6,6	6,6	6,6	6,6
RAD	7,1	7,1	7,1	7,1
Média	6,4	6,4	6,4	6,4

(b)				
Distribuição	BRKGA-STT	BRKGA-Cluster	BRKGA-A	BRKGA-BV
R	3,5	3,5	4,0	4,1
C	3,4	3,4	4,0	4,1
RC	3,7	3,8	4,5	4,6
RAD	4,7	4,9	5,3	5,6
Média	3,8	3,9	4,5	4,6

Tabela 5.18 – Valor médio do GAP_{UB} de cada versão do BRKGA para instâncias com diferentes tamanhos de janelas de tempo: a) instâncias pequenas (casos 1 a 4). b) instâncias médias e grandes (casos 5 a 13).

(a)				
Tamanho da janela de tempo	BRKGA-STT	BRKGA-Cluster	BRKGA-A	BRKGA-BV
2h	4,7	4,7	4,8	4,8
4h	6,8	6,8	6,9	6,9
8h	6,6	6,6	6,6	6,7
2 a 8h	6,8	6,5	6,8	6,9
Média	6,2	6,2	6,3	6,3

(b)				
Tamanho da janela de tempo	BRKGA-STT	BRKGA-Cluster	BRKGA-A	BRKGA-BV
2h	4,0	4,0	4,7	4,9
4h	3,7	3,7	4,2	4,4
8h	3,0	3,0	3,4	3,6
2 a 8h	4,3	4,2	4,9	5,1
Média	3,8	3,7	4,3	4,5

Com relação ao tamanho da janela de tempo, a Tabela 5.18 apresenta os valores médios para as instâncias onde todas as janelas de tempo têm tamanho igual a 2 horas, 4 horas, 8 horas e instâncias onde as janelas de tempo variam entre 2 e 8 horas. Para os problemas pequenos, novamente o comportamento das três versões do BRKGA é semelhante, como é observado na Tabela 5.18(a). Já para as instâncias médias e grandes (Tabela 5.18(b)), as duas melhores versões confirmam o seu desempenho superior para todos os tamanhos de janelas de tempo. Note também que há uma pequena variação entre o GAP_{UB} médio das instâncias com janelas “apertadas” (tamanho igual a 2h) e janelas de tempo “folgadas” (tamanho de 8h), mas que essa variação é principalmente influenciada pela qualidade dos limitantes superiores, pois sabe-se que para as instâncias pequenas quase todos os valores ótimos são encontrados pelas melhores versões do algoritmo e que esses resultados também apresentam uma variação entre as instâncias com janelas de tempo “apertadas” e “folgadas”, como pode ser observado na Tabela 5.18(a).

Até o momento, todos os resultados apresentados foram baseados na resolução de cada instância uma única vez por cada versão do BRKGA (BIRATTARI, 2004). Para analisar o componente aleatório do BRKGA, uma comparação foi feita entre os resultados de uma única execução e de dez execuções de um grupo de 104 instâncias (8 instâncias de cada caso). A Tabela 5.19 apresenta os resultados. Para cada versão, é apresentada o GAP_{UB} médio de apenas uma execução (coluna “1 exec.”) das 104 instâncias e, para dez execuções (“10 exec.”), a média dos melhores GAP_{UB} obtidos para cada instância (coluna “Min.”), a média geral (coluna “Med.”) e a média dos piores GAP_{UB} de cada instância (coluna “Max”). Dois pontos podem ser ressaltados a partir dos resultados obtidos. Primeiro, que o GAP_{UB} médio obtido em cada um dos casos com uma e dez execuções é igual ou menor do que 0,1% (exceto para o BRKGA-BV) e a média geral de todos os casos é igual; portanto, constatou-se uma diferença pequena do GAP_{UB} médio ao executar uma ou dez vezes cada instância. Segundo, que a diferença entre as médias dos melhores e dos piores GAP_{UB} (colunas “Min.” e “Max.”) é menor do que 0,4% para as duas melhores versões do BRKGA, o que sugere um comportamento estável para o BRKGA (a maior variação média foi de 0,9%, verificada na versão com o decodificador com busca em vizinhança). De acordo com Birattari (2004), os resultados obtidos com a resolução de apenas uma vez cada uma das 1040 instâncias (média de $GAP_{UB} = 3,8\%$ para o BRKGA-STT) é um resultado mais confiável do desempenho

Tabela 5.19 – Média do GAP_{UB} de uma execução (1 exec.); média dos valores mínimos (Min.), média geral (Med.) e média dos valores máximos (Max.) do GAP_{UB} de 5 execuções de 8 instâncias para cada caso.

Caso	BRKGA-STT				BRKGA-Cluster				BRKGA-A				BRKGA-BV			
	1 exec.	10 exec.			1 exec.	10 exec.			1 exec.	10 exec.			1 exec.	10 exec.		
	Med.	Min.	Med.	Max.	Med.	Min.	Med.	Max.	Med.	Min.	Med.	Max.	Med.	Min.	Med.	Max.
1	3,2	3,2	3,2	3,2	3,2	3,2	3,2	3,2	3,2	3,2	3,2	3,2	3,2	3,2	3,2	3,2
2	7,6	7,6	7,6	7,6	7,6	7,6	7,6	7,6	7,6	7,6	7,6	7,6	7,7	7,6	7,6	7,7
3	5,9	5,9	5,9	6,0	5,9	5,9	5,9	5,9	5,9	5,9	5,9	6,0	6,1	6,0	6,0	6,1
4	7,9	7,9	7,9	7,9	7,9	7,9	7,9	7,9	8,0	7,9	7,9	8,0	7,9	7,9	7,9	7,9
5	3,1	3,0	3,1	3,2	3,1	3,0	3,1	3,2	3,3	3,1	3,3	3,6	3,4	2,5	3,2	3,6
6	9,5	9,3	9,6	10,0	9,6	9,3	9,5	9,7	10,0	9,4	9,9	10,5	9,8	9,6	9,8	10,2
7	2,5	2,3	2,5	2,8	2,5	2,3	2,5	2,7	3,0	2,7	3,1	3,4	3,2	2,6	3,1	3,5
8	8,5	8,2	8,6	9,1	8,2	8,0	8,3	8,7	9,1	8,6	9,2	9,8	8,7	7,6	9,2	10,3
9	2,0	1,8	2,0	2,1	1,8	1,7	1,8	2,0	2,7	2,5	2,8	3,1	2,7	2,4	2,9	3,4
10	1,3	1,1	1,2	1,4	1,2	1,1	1,2	1,3	2,2	1,8	2,2	2,5	2,3	2,0	2,3	2,7
11	1,6	1,4	1,5	1,8	1,5	1,3	1,4	1,6	2,5	2,2	2,5	2,7	3,0	2,4	2,7	3,1
12	1,7	1,4	1,6	1,8	1,9	1,8	1,9	2,1	2,2	2,0	2,2	2,4	2,4	2,1	2,3	2,5
13	1,4	1,3	1,4	1,5	2,0	1,9	2,0	2,1	1,8	1,7	1,8	1,9	2,0	1,9	2,0	2,1
Med. 1-4	6,1	6,1	6,1	6,2	6,1	6,1	6,1	6,1	6,2	6,1	6,2	6,2	6,2	6,2	6,2	6,2
Med. 5-13	3,5	3,3	3,5	3,7	3,5	3,4	3,5	3,7	4,1	3,8	4,1	4,4	4,2	3,7	4,2	4,6

do algoritmo do que o resultado obtido com dez execuções de 104 instâncias (média de $GAP_{UB} = 3,5\%$ para a mesma versão do algoritmo).

5.5.3 Desempenho do BRKGA em instâncias da literatura

Com o objetivo de comparar a meta-heurística com outros métodos da literatura, o desempenho do BRKGA-STT foi avaliado no problema proposto por Kovacs et al. (2012). Os autores utilizaram a meta-heurística *Adaptive Large Neighborhood Search* (ALNS) para resolver um problema semelhante, que considera tarefas com idêntica prioridade, mas com diferentes janelas de tempo. Além disso, em Kovacs et al. (2012) a função objetivo procura minimizar os custos totais de deslocamento e de terceirização. Note que minimizar o custo de terceirização pode ser visto como maximizar o atendimento ponderado das tarefas, enquanto que minimizar o deslocamento total está relacionado com maximizar a ociosidade dos técnicos no fim do dia, ou seja, há um paralelismo entre a função objetivo proposta pelos autores e a adotada no problema deste trabalho (em ambos os problemas, sempre é vantajoso aumentar o deslocamento e executar uma tarefa do que terceirizá-la ou não executá-la). Para adaptar o decodificador STT à nova função objetivo, algumas adaptações foram realizadas. No Passo 2 (ver Seção 2.3 e 3.4.1) ao avaliar uma nova tarefa na rota de cada técnico, será escolhida a posição em que resultar no menor tempo de deslocamento total. No Passo 3, para determinar qual técnico candidato executará a nova tarefa, será escolhido o técnico cuja rota tem o menor aumento de custo de deslocamento. Os experimentos desta seção foram realizados em um computador de 2,93 GHz Intel(R) E7500 com 4 GB de memória RAM.

Todas as instâncias *no team* de Kovacs et al. (2012) foram resolvidas, ou seja, todas as instâncias em que as tarefas são executadas por um único técnico. Concretamente, foram considerados os conjuntos de instâncias denominadas *small*, *complete* e *reduced* com 35, 36 e 36 instâncias, respectivamente. As instâncias *small* têm menores dimensões (até 50 tarefas e 6 técnicos) e é conhecido o valor ótimo. O termo *complete* refere-se às instâncias onde, dado ao elevado número de técnicos, todas as tarefas podem ser executadas sem a necessidade de terceirização. Por outro lado, a expressão *reduced* significa que há um número limitado de técnicos e que é provável que algumas tarefas não poderão ser programadas, ou seja, em princípio será necessário escolher algumas tarefas para serem terceirizadas. Tanto as instâncias *complete* quanto as *reduced* têm

100 tarefas a serem executadas. Assim como em Kovacs et al. (2012), a meta-heurística resolveu cinco vezes cada instância. Os resultados foram comparados com o valor ótimo das instâncias *small* e com os valores médios obtidos pela meta-heurística ALNS.

No conjunto de instâncias *small*, o BRKGA-STT encontrou as soluções ótimas em todas as execuções, superando a heurística ALNS, que obteve o valor ótimo em todas as cinco execuções somente para 71,4% das instâncias (como pode ser observado na Tabela 6 da página 591 de Kovacs et al. (2012)). O tempo médio de CPU por instância para a ALNS e para o BRKGA-STT foram de 3,72 e 2,99 segundos, respectivamente. As Tabelas 5.20 e 5.21 apresentam os resultados obtidos pelo BRKGA-STT para o conjunto de instâncias *complete* e *reduced*, respectivamente. As tabelas apresentam na primeira coluna os nomes das instâncias, na segunda o valor médio (“Med.”) obtido pelas cinco execuções de cada instância, na terceira a diferença percentual entre o resultado médio obtido pela ALNS (fo_A) e o obtido pelo BRKGA-STT (fo_B) dado por $Dif_{med} = 100 \cdot \frac{fo_B - fo_A}{fo_A}$, na quarta o pior valor (“Max.”) obtido entre as cinco execuções de cada instância, na quinta a diferença percentual entre o pior resultado obtido pela ALNS ($fo_{A_{max}}$) e o obtido pelo BRKGA-STT ($fo_{B_{max}}$) dado por $Dif_{max} = 100 \cdot \frac{fo_{B_{max}} - fo_{A_{max}}}{fo_{A_{max}}}$, na sexta o melhor valor (“Min.”) obtido entre as cinco execuções de cada instância, na sétima a diferença percentual entre o melhor resultado obtido pela ALNS ($fo_{A_{min}}$) e o obtido pelo BRKGA-STT ($fo_{B_{min}}$) dado por $Dif_{min} = 100 \cdot \frac{fo_{B_{min}} - fo_{A_{min}}}{fo_{A_{min}}}$, e as últimas três colunas apresentam, respectivamente, o número médio ($|V|$) de tarefas executadas (não terceirizadas) entre os resultados das cinco execuções, o número médio de técnicos necessários ($|T|$) nas cinco execuções e o tempo médio de processamento (coluna CPU (s)). Em itálico estão destacados os resultados em que o BRKGA-STT superou a ALNS.

Tabela 5.20 – Desempenho do BRKGA-STT no conjunto de instâncias *complete* propostas por Kovacs et al. (2012)

Instância	Med.	Dif_{med}	Max.	Dif_{max}	Min.	Dif_{min}	V	T	CPU (s)
C101_5x4	1201,68	8,15	1231,51	9,17	1174,66	6,91	100,0	13,6	92,32
C103_5x4	1124,55	8,41	1143,19	8,94	1100,98	8,09	100,0	12,0	138,14
C201_5x4	1157,56	-1,98	1157,56	-5,80	1157,56	-0,12	100,0	7,0	50,05
C203_5x4	1092,09	4,08	1093,01	3,82	1091,48	4,26	100,0	5,0	79,35
R101_5x4	1718,85	1,96	1726,61	1,73	1712,04	1,99	100,0	21,0	130,60
R103_5x4	1315,80	5,27	1325,11	3,34	1308,40	5,63	100,0	15,6	114,57
R201_5x4	1450,12	0,08	1463,52	0,06	1427,75	-0,87	100,0	6,6	131,01
R203_5x4	1111,87	0,52	1118,77	-0,38	1101,49	0,32	100,0	5,6	138,59
RC101_5x4	1737,49	1,25	1753,98	1,40	1704,93	0,14	100,0	17,4	109,89
RC103_5x4	1481,46	9,40	1495,18	7,71	1458,00	8,97	100,0	13,2	127,88
RC201_5x4	1605,69	-0,10	1605,69	-0,31	1605,69	0,24	100,0	8,0	74,26
RC203_5x4	1180,38	1,19	1180,38	0,15	1180,38	1,62	100,0	6,0	81,83
C101_6x6	1066,40	6,13	1080,86	4,97	1027,79	3,90	100,0	12,4	102,11
C103_6x6	1076,75	19,92	1085,33	19,58	1063,41	18,96	100,0	10,8	143,52
C201_6x6	821,54	0,00	821,54	0,00	821,54	0,00	100,0	4,0	32,58
C203_6x6	689,68	-1,91	690,01	-8,01	689,60	0,00	100,0	4,0	95,80
R101_6x6	1677,54	0,61	1683,37	0,65	1669,71	0,69	100,0	20,0	154,93
R103_6x6	1249,34	1,45	1256,12	1,02	1239,15	1,27	100,0	14,6	137,30
R201_6x6	1300,61	2,39	1320,39	3,17	1264,52	0,20	100,0	5,6	123,44
R203_6x6	949,89	-0,20	953,26	-1,17	946,30	1,50	100,0	5,0	146,46
RC101_6x6	1703,77	1,18	1717,31	1,61	1691,16	0,72	100,0	16,0	145,63
RC103_6x6	1378,68	5,17	1395,43	4,81	1361,02	6,20	100,0	12,4	150,37
RC201_6x6	1427,44	1,67	1448,41	2,62	1403,57	0,59	100,0	6,6	112,59
RC203_6x6	1034,83	1,78	1047,69	1,70	1030,11	2,90	100,0	5,0	145,11
C101_7x4	1456,84	4,14	1487,19	1,71	1423,91	4,93	100,0	15,6	94,61
C103_7x4	1309,31	5,66	1315,36	4,05	1300,13	6,94	100,0	12,4	128,69
C201_7x4	1256,30	-2,02	1256,30	-3,55	1256,30	0,00	100,0	8,0	29,84
C203_7x4	1153,28	0,17	1154,51	0,14	1152,94	0,18	100,0	7,0	64,19
R101_7x4	1826,76	1,83	1834,10	1,13	1820,36	2,47	100,0	22,0	105,46
R103_7x4	1412,75	2,74	1426,51	1,90	1401,02	4,03	100,0	16,2	137,37
R201_7x4	1429,36	1,31	1437,17	0,65	1417,85	1,41	100,0	9,2	79,23
R203_7x4	1171,01	0,35	1174,52	0,45	1168,92	0,35	100,0	8,0	126,38
RC101_7x4	1897,61	2,89	1906,81	2,56	1885,79	3,51	100,0	18,6	133,94
RC103_7x4	1528,59	5,03	1544,32	4,50	1514,05	5,46	100,0	13,4	86,57
RC201_7x4	1719,80	1,09	1745,58	2,35	1713,35	0,91	100,0	9,0	113,91
RC203_7x4	1247,77	0,49	1260,24	0,84	1239,45	0,00	100,0	7,0	62,41
Med.	1332,32	2,78	1342,69	2,15	1320,15	2,90	100,00	10,94	108,91

Tabela 5.21 – Desempenho do BRKGA-STT no conjunto de instâncias *reduced* propostas por Kovacs et al. (2012)

Instância	Med.	GAP_{med}	Max.	GAP_{max}	Min.	GAP_{min}	V	T	CPU (s)
C101_5x4	5763,44	0,52	5880,65	1,28	5685,83	0,52	76,6	8,0	74,41
C103_5x4	2971,30	6,80	3098,04	7,96	2827,17	6,90	91,6	8,0	156,49
C201_5x4	2755,52	0,00	2755,52	0,00	2755,52	0,00	94,0	4,0	72,00
C203_5x4	2386,03	-0,27	2394,83	0,05	2376,09	-0,56	94,0	4,0	101,77
R101_5x4	5636,46	-4,39	5781,31	-6,47	5575,22	-0,13	78,8	12,0	90,35
R103_5x4	1932,31	4,72	2009,70	-0,53	1801,25	5,32	97,2	12,0	106,10
R201_5x4	2869,57	0,53	2903,92	1,33	2838,51	0,00	94,0	4,0	134,29
R203_5x4	2332,27	0,00	2332,42	0,01	2332,23	0,00	94,0	4,0	104,00
RC101_5x4	5129,91	-0,68	5262,37	0,00	5078,63	-0,96	81,8	11,0	99,82
RC103_5x4	2524,94	7,53	2671,35	7,28	2456,30	13,16	94,8	11,0	152,91
RC201_5x4	3096,08	0,14	3109,45	0,51	3086,68	-0,05	94,0	5,0	140,42
RC203_5x4	2536,76	-0,14	2545,26	-0,21	2528,65	0,50	94,0	5,0	154,79
C101_6x6	7755,07	-0,10	7844,77	0,69	7706,74	-0,31	68,0	8,0	100,77
C103_6x6	5084,84	1,11	5218,69	1,61	5041,74	1,23	81,8	8,0	139,38
C201_6x6	3378,22	2,38	3378,22	1,51	3378,22	3,06	90,0	4,0	43,62
C203_6x6	2471,91	0,24	2486,64	0,73	2457,70	-0,10	94,0	3,0	144,17
R101_6x6	6170,52	0,30	6293,28	-0,47	6129,43	2,93	76,8	13,0	130,97
R103_6x6	2330,16	0,04	2443,08	1,60	2244,44	-0,32	95,6	12,2	121,08
R201_6x6	3615,64	2,23	3710,05	3,78	3577,01	2,10	90,8	4,0	146,89
R203_6x6	2463,73	0,72	2476,75	-0,20	2459,98	0,93	94,0	3,0	155,31
RC101_6x6	5508,04	0,76	5733,23	-0,67	5341,20	1,23	81,2	12,0	131,78
RC103_6x6	2586,16	10,07	2846,47	12,83	2264,79	0,04	95,4	11,8	146,97
RC201_6x6	4619,96	2,21	4750,47	2,01	4546,69	2,80	86,6	3,6	153,52
RC203_6x6	2711,80	1,42	2794,68	2,34	2672,69	0,87	94,0	3,0	148,66
C101_7x4	5275,30	0,33	5343,55	0,69	5249,77	0,80	81,0	9,0	108,62
C103_7x4	2091,45	-1,23	2228,37	2,53	1974,84	-2,25	97,6	9,0	152,86
C201_7x4	2776,38	-0,11	2788,23	-0,53	2773,42	0,00	95,0	4,0	55,86
C203_7x4	2288,40	0,27	2301,80	0,00	2261,38	0,00	95,0	4,0	120,31
R101_7x4	5336,78	-0,83	5424,57	-0,24	5286,06	0,88	81,8	14,0	85,25
R103_7x4	2210,41	-0,24	2296,29	-0,78	2117,21	0,58	96,6	13,8	109,69
R201_7x4	2671,42	-0,30	2683,62	0,05	2665,52	-0,28	95,0	5,0	118,49
R203_7x4	2201,93	-0,36	2206,17	-1,05	2199,10	0,00	95,0	5,0	124,95
RC101_7x4	5547,86	-4,34	5679,15	-10,81	5481,27	-0,90	80,8	12,0	64,77
RC103_7x4	2753,69	2,96	3051,77	8,20	2591,39	0,21	94,2	12,0	85,46
RC201_7x4	2914,19	-0,75	2927,04	-0,63	2906,52	-0,46	95,0	5,0	101,45
RC203_7x4	2316,57	1,37	2325,91	1,07	2313,84	1,59	95,0	4,6	127,17
Med.	3528,19	0,91	3610,49	0,98	3471,75	1,09	89,86	7,53	116,82

Os resultados médios do BRKGA-STT ficaram a 2,78% (instâncias *complete*) e 0,91% (*reduced*) dos da ALNS. O melhor desempenho do BRKGA-STT no conjunto de instâncias *reduced* era esperado, dado que essa meta-heurística foi originalmente desenvolvida para problemas em que há um número restrito de técnicos e, consequentemente, é necessário selecionar as tarefas que serão executadas. Outra comparação pode ser feita entre os valores médios da função objetivo obtidos pelo BRKGA-STT para os conjuntos de instâncias *complete* e *reduced* (respectivamente, 1332,32 e 3528,19) e os valores médios obtidos para oito versões da meta-heurística LNS (*Large Neighborhood Search*), que incluem a ALNS. Os resultados encontram-se na Tabela 3 (colunas 6 a 9) na página 589 de Kovacs et al. (2012) e indicam um desempenho competitivo do BRKGA-STT.

Além da comparação com o experimento realizado por Kovacs et al. (2012), outra comparação foi realizada, aplicando o BRKGA-STT a 12 instâncias propostas por Solomon (1987) para o Problema de Roteamento de Veículos com Janelas de Tempo em que o valor ótimo é conhecido. A função objetivo minimiza o custo total de deslocamento e todas as instâncias têm 100 clientes. O problema considera que há um número ilimitado de veículos, que têm restrições de capacidade. Os tempos de deslocamento e as distâncias são calculadas truncando a primeira casa decimal. Cada instância foi resolvida cinco vezes e a média dos resultados obtidos encontra-se na Tabela 5.22. A primeira coluna apresenta o nome da instância; a segunda, o valor da solução ótima; a terceira, a média (“Med.”) dos valores da função objetivo obtidos nas cinco instâncias; a quarta, no número médio de veículos utilizados ($|T|$); a quinta, a diferença percentual entre a média obtida pelo BRKGA-STT (f_{OB}) e o valor ótimo ($f_{O_{opt}}$) dado por $GAP = 100 \cdot \frac{f_{OB} - f_{O_{opt}}}{f_{O_{opt}}}$; e a última coluna, o tempo médio utilizado nas execuções em segundos.

Pelos resultados médios obtidos, pode-se concluir que o BRKGA-STT obteve um desempenho competitivo, com uma diferença média de 0,97% das soluções ótimas e com um tempo médio de execução de 151,54 segundos. O desempenho da ALNS para essas instâncias se encontra na Tabela 5, página 590, de Kovacs et al. (2012). Em média, a ALNS obteve um GAP de 0,38% e com um tempo de processamento de 102,09 segundos.

A partir dos experimentos computacionais realizados com as instâncias de Kovacs et

Tabela 5.22 – Resultados do BRKGA-STT em 12 instâncias do PRVJT propostas por Solomon

Instância	Sol. ótima	Med.	T	GAP	CPU (s)
C101	827,30	827,30	10,0	0,00	0,1
C103	826,30	826,64	10,0	0,04	157,6
C201	589,10	589,10	3,0	0,00	0,1
C203	588,70	588,70	3,0	0,00	80,4
R101	1637,70	1637,70	20,0	0,00	131,6
R103	1208,70	1210,86	14,0	0,18	153,6
R201	1143,20	1153,82	9,0	0,93	171,6
R203	870,80	885,88	6,6	1,73	256,1
RC101	1619,80	1645,58	15,4	1,59	173,5
RC103	1258,00	1305,86	11,2	3,80	277,1
RC201	1261,80	1280,90	9,0	1,51	168,3
RC203	923,70	940,26	5,0	1,79	248,5
Med.	1062,93	1074,38	9,68	0,97	151,54

al. (2012) e de Solomon (1987), pode-se tirar duas conclusões. Primeiro, que o BRKGA necessitou de um tempo médio maior de execução, o que a princípio era esperado por se tratar de uma meta-heurística populacional. Também se deve ressaltar que os resultados foram obtidos em máquinas diferentes, provavelmente com diferentes compiladores. Segundo, que o desempenho inferior em relação à ALNS em parte é decorrente de o BRKGA-STT ter sido desenvolvido para um problema com diferente função objetivo e, ao mesmo tempo, sinaliza que maiores alterações no decodificador (por exemplo, incluir uma busca local no decodificador ou desenvolver outras maneiras de determinar a posição de uma tarefa em uma rota) seriam necessárias para melhor explorar as características desses problemas paralelos e, assim, aumentar a competitividade dos resultados obtidos pelo BRKGA.

5.6 Análise dos resultados da Parte I

A primeira parte deste trabalho apresentou heurísticas construtivas e alguns Algoritmos Genéticos com Chaves Aleatórias Viesadas (*Biased Random Key Genetic Algorithm* – BRKGA) para o Problema mono-objetivo de Escalonamento de Técnicos de Campo.

As melhores heurísticas construtivas, para instâncias de grandes dimensões (500 e 999 tarefas), geraram bons resultados em comparação com os limitantes superiores, com um custo computacional reduzido. Esse bom desempenho pode ser atribuído pelo

fato de as heurísticas construtivas explorarem as características específicas do problema, como as prioridades das tarefas, a distribuição geográfica dos clientes, as habilidades dos técnicos e os tempos de deslocamento, o que indica o potencial dessas heurísticas para problemas de grandes dimensões.

As melhores metas-heurísticas, o BRKGA-STT e BRKGA-Cluster, encontraram 99% e 97% das soluções ótimas para problemas de pequenas dimensões, respectivamente. Para problemas de médias e grandes dimensões, os resultados médios ficaram a 3,8% e 3,9% dos limitantes superiores, respectivamente. O fato de essas duas versões terem superado o desempenho da versão mais aleatória do BRKGA indica a vantagem do uso de decodificadores mais elaborados.

A função similaridade, criada para diversificar o conjunto elite, atingiu o seu objetivo e evitou a convergência prematura do algoritmo. Os valores médios dos resultados foram melhorados (em até 2% em média) e o número de instâncias onde o BRKGA com a medida de similaridade obteve resultados estritamente melhores ficou entre 100% e 71%, dependendo da versão do BRKGA. Pode-se ressaltar que o desempenho do BRKGA melhorou de modo mais expressivo com o decodificador Aleatório, que tinha uma tendência mais acentuada à convergência prematura.

Parte II

Otimização Multiobjetivo

Capítulo 6

Otimização Multiobjetivo: fundamentos teóricos

A maior parte da literatura apresenta o PETC mono-objetivo com diferentes objetivos agregados em uma única função: Tsang e Voudouris (1997) somaram os custos de deslocamento, horas extras e tarefas não executadas; Kovacs et al. (2012) uniram em uma função os custos de deslocamento e de terceirização; Li; Lim e Rodrigues (2005) agruparam o número total de trabalhadores necessários com o tempo total de deslocamento; a função objetivo de Cortés et al. (2014) considera o número de vezes que as janelas de tempo são violadas, o número de clientes atendidos e o tempo total de deslocamento dos técnicos; e Xu e Chiu (2001), como já descrito nos capítulos anteriores, maximizaram o número ponderado de tarefas atendidas e a ociosidade dos técnicos após o retorno à base. Trata-se, portanto, de um problema com diversos possíveis objetivos e que poderia ser melhor explorado por técnicas de otimização multiobjetivo.

Inspirado nos trabalhos de Cordeau et al. (2010), Hashimoto et al. (2011) e Cortés et al. (2014), esse trabalho propõe uma nova função objetivo, que procurará que os clientes mais importantes ou as tarefas mais prioritárias sejam atendidas o mais cedo possível. Trata-se de um novo objetivo que pode ser conflitante com o objetivo adotado até o momento, pois atender as tarefas prioritárias mais cedo muitas vezes implicará no cancelamento da execução de outras menos prioritárias e/ou no aumento do tempo de deslocamento do técnico ao longo do dia e, conseqüentemente, na redução do tempo disponível após o retorno à base. Otimizando as duas funções objetivo simultaneamente, espera-se que as técnicas de otimização multiobjetivo forneçam diversas soluções para cada instância, de modo que o tomador de decisões possa decidir se aceita não executar

algumas tarefas menos importantes ou reduzir a disponibilidade dos técnicos no fim do dia para atender os clientes mais importantes mais cedo, ou se prefere atendê-los mais tarde para executar o máximo de tarefas possíveis.

Assim como em problemas mono-objetivo, onde frequentemente se recorre às heurísticas diante de problemas NP-difíceis, diversas meta-heurísticas surgiram para resolver os problemas multiobjetivos NP-difíceis. Dentre elas, o Algoritmo Genético (AG) tem sido particularmente bem sucedido (KONAK; COIT; SMITH, 2006). Assim como na abordagem mono-objetivo do PETC, neste trabalho pretende-se aplicar e adaptar o *Biased Random-Key Genetic Algorithms (BRKGA)* para o PETC multiobjetivo.

Este capítulo apresentará os fundamentos teóricos de Otimização Multiobjetivo e as medidas de avaliação das soluções geradas. Os próximos capítulos desta Segunda Parte apresentarão uma proposta multiobjetivo para o PETC e o seu modelo de Programação Linear Inteira Mista (Capítulo 7), uma proposta para um BRKGA Multiobjetivo (Capítulo 8) e os resultados computacionais obtidos, incluindo uma comparação do BRKGA Multiobjetivo com um Algoritmo Genético conhecido na literatura (Capítulo 9).

6.1 Fundamentos de Otimização Multiobjetivo

Sendo a otimização uma ferramenta que possibilita a tomada de decisão em uma empresa (no contexto deste estudo, para definir as rotas diárias dos técnicos), pode-se afirmar que diferentes aspectos condicionam essa tomada de decisão (TKINDT; BILLAUT, 2006):

- Muitas vezes não existe um critério claramente definido para uma tomada de decisão, mas sim orientações ou desejos de atingir vários objetivos simultaneamente.
- As decisões são tomadas geralmente por um grupo de pessoas e não por um único indivíduo. Ao mesmo tempo, as intenções dos tomadores de decisão ou as necessidades da empresa podem modificar-se ao longo do tempo.
- Muitas vezes os tomadores de decisão são incapazes de diferenciar uma solução ótima de outras boas soluções.

Nesse contexto, a otimização multiobjetivo pode fornecer um melhor retrato dessa realidade: buscando diferentes ou conflitantes objetivos, pode-se encontrar diversas soluções para um problema e possibilitar ao(s) decisor(es) um protagonismo maior na

tomada de decisão. Em três momentos pode-se atuar no processo de otimização multiobjetivo: antes, durante ou depois da resolução. Dessa forma, os métodos de otimização multiobjetivo podem ser vistos sob três pontos de vista (COELLO; LAMONT; VAN VELDHUIZEN, 2007; TKINDT; BILLAUT, 2006; MIETTINEN, 2008):

1. *Métodos a priori*: onde antes da resolução é definida a ordem de importância dos objetivos, ou o peso ou o grau de importância de cada objetivo.
2. *Métodos iterativos*: quando há uma intervenção durante o processo de resolução, por exemplo alterando o peso ou importância dos objetivos a cada iteração.
3. *Métodos a posteriori*: que geram diversas soluções, permitindo ao decisor escolher a mais satisfatória e que será adotada.

No entanto, essas três classificações não são exclusivas, uma vez que alguns métodos podem ser vistos tanto quanto *a priori* quanto a *a posteriori*. Por exemplo, o método Soma Ponderada (apresentado na Seção 6.1.2) pode ser considerado *a priori* (quando o peso de cada objetivo é definido previamente) ou *a posteriori* (quando várias combinações de pesos são testadas e diversas possíveis soluções são geradas).

A seguir é apresentado o conceito de otimalidade em otimização multiobjetivo e em seguida, alguns métodos multiobjetivo tradicionais e algumas das metas-heurísticas mais encontradas na literatura.

6.1.1 Definição de otimalidade

Do ponto de vista matemático, um problema de minimização multiobjetivo pode ser definido como:

$$\text{Min } F(x), \text{ com } F(x) = [f_1(x); \dots; f_K(x)]^T \text{ e } x \in \mathbf{R}^n \quad (6.1)$$

sujeito a:

$$x \in \mathcal{S}$$

$$\mathcal{S} = \{x/[g_1(x); \dots; g_M(x)]^T \leq 0\}$$

Na otimização mono-objetivo, para quaisquer duas soluções $x, y \in \mathbf{R}$, $f(x) \leq f(y)$ ou $f(x) \geq f(y)$ e, portanto, identificar uma solução ótima é algo trivial. Já na otimização multiobjetivo, pode não ser possível ordenar duas soluções $x, y \in \mathbf{R}^n$, pois não

Figura 6.1 – Exemplo do conceito de dominância

Fonte: elaborado pelo autor.

necessariamente é verdade que $F(x) \leq F(y)$ ou $F(x) \geq F(y)$ (ou seja, $f_i(x) \leq f_i(y)$ ou $f_i(x) \geq f_i(y), \forall i = 1, \dots, k$). O conceito de otimalidade em otimização multiobjetivo é mais elaborado e é necessário adotar algumas definições (DEB, 2001; MIETTINEN, 2008; TKINDT; BILLAUT, 2006; ZITZLER; KNOWLES; THIELE, 2008):

Definição 6.1 *Sejam $x, y \in \mathbf{R}^n$ duas soluções de um problema de minimização multi-objetivo:*

- *se $F(x) \leq F(y)$ e existe i tal que $f_i(x) < f_i(y)$, diz-se que x domina y , o que é representado por $x \prec y$.*
- *se $F(x) \geq F(y)$ e existe i tal que $f_i(x) > f_i(y)$, diz-se que x é dominado por y , o que é representado por $x \succ y$.*
- *se $F(x) \not\leq F(y)$ e $F(x) \not\geq F(y)$, diz-se que x e y são indiferentes.*

A Figura 6.1 apresenta um exemplo gráfico da definição 6.1. Tomando como referência uma solução A de um problema de minimização, a figura destaca as soluções que são dominadas, que dominam e que são indiferentes a A .

Definição 6.2 *Seja uma solução $x \in \mathbf{R}^n$ e um conjunto de soluções $\mathbf{X} \in \mathbf{R}^n$. A solução x é não dominada pelo conjunto $\mathbf{X}$ se, e somente se, não existe $y \in \mathbf{X}$ tal que y domina x ($\nexists y \in \mathbf{X} / y \prec x$).*

Figura 6.2 – Exemplo de conjunto ótimo estrito de Pareto ou soluções eficientes

Fonte: elaborado pelo autor.

Definição 6.3 (Conjunto ótimo estrito de Pareto) *Uma solução de um problema multiobjetivo $x \in \mathcal{S}$ pertence ao conjunto ótimo estrito de Pareto se, e somente se, $\nexists y \in S$ tal que $\forall i = 1, \dots, k$ $f_i(y) \leq f_i(x)$ e tal que $\exists j$ onde $f_j(y) < f_j(x)$.*

Em outras palavras, o conjunto ótimo estrito de Pareto é formado pelos elementos não dominados do conjunto $\mathcal{S}$. Esse conjunto também é conhecido como conjunto de soluções eficientes.

O conjunto ótimo estrito de Pareto pode ser observado no exemplo da Figura 6.2. Para um problema de minimização, as soluções não dominadas foram destacadas em azul e a linha da mesma cor simboliza a fronteira da região factível.

Definição 6.4 (Conjunto ótimo fraco de Pareto) *Uma solução de um problema multiobjetivo $x \in \mathcal{S}$ pertence ao conjunto ótimo fraco de Pareto se, e somente se, $\nexists y \in S$ tal que $\forall i = 1, \dots, k$ $f_i(y) < f_i(x)$.*

Na Figura 6.3 pode-se visualizar um exemplo do conceito de conjunto ótimo fraco de Pareto. Note que apesar de as soluções A e B serem dominadas pela C (as três têm o mesmo valor para f_1 e C tem menor valor para f_2), não há nenhuma solução com valores para as duas funções objetivos estritamente menores do que as soluções A e B. Analogamente, apesar de os pontos E e F serem dominados por D, eles pertencem ao conjunto ótimo fraco de Pareto. Repare também que as soluções entre C e D correspondem ao conjunto ótimo estrito de Pareto, pois o conjunto ótimo estrito de Pareto está contido no conjunto ótimo fraco de Pareto.

Figura 6.3 – Exemplo de conjunto ótimo fraco de Pareto

Fonte: elaborado pelo autor.

As definições apresentadas até o momento – para problemas de minimização – servirão de base para os métodos apresentados nas próximas seções. Note que, para problemas de maximização, como o PETC, as definições são análogas.

6.1.2 Métodos Tradicionais

Há várias técnicas que resolvem os problemas multiobjetivo transformando-os em problemas de otimização mono-objetivo, onde ou os objetivos são agrupados em uma única função objetivo, ou alguns dos objetivos são transformados em restrições e/ou sucessivos problemas são resolvidos incluindo um novo objetivo a cada resolução. Pode-se citar, por exemplo, os métodos de soma ponderada, o método ε -restrito, ordem lexicográfica, ordem lexicográfica com metas, o método de métricas ponderadas, o método de Benson, o método de restrições elásticas, etc. (DEB, 2001; EHRGOTT, 2006; DONOSO; FABREGAT, 2007). A seguir alguns desses métodos são apresentados; sob certas condições, alguns deles podem encontrar soluções ótimas de Pareto.

Soma Ponderada

Método tradicional na literatura, onde os diversos objetivos compõem uma única função objetivo, com diferentes pesos para cada um dos K objetivos:

$$\text{Min } F(x) = \sum_{i=1}^K \alpha_i \cdot f_i(x) \quad (6.2)$$

onde:

$$\sum_1^K \alpha_i = 1$$

$$\alpha_i > 0$$

Um aspecto importante deste método é que, para que os coeficientes α_i realmente reflitam o grau de importância dos objetivos, esses objetivos devem ser normalizados ou terem a mesma ordem de grandeza.

Como os resultados variam de acordo com os coeficientes α_i , pode-se resolver um problema diversas vezes, utilizando diferentes conjuntos de coeficientes, e analisar cada um dos resultados obtidos. O teorema 6.1 apresenta a condição necessária e suficiente para que o método *Soma Ponderada* possa determinar o conjunto ótimo de Pareto. As definições 6.5 e 6.6 fornecem o suporte teórico para a compreensão do teorema (TKINDT; BILLAUT, 2006).

Definição 6.5 (Conjunto Convexo) *Um conjunto $\mathcal{S} \subset \mathbb{R}^K$ é convexo se e somente se $\forall x, y \in \mathcal{S}, \forall \lambda \in [0; 1], \lambda x + (1 - \lambda)y \in \mathcal{S}$.*

Definição 6.6 (Função Convexa) *Uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$ é convexa se e somente se $\forall x, y \in \mathbb{R}^n, \forall \lambda \in [0; 1], f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$.*

Teorema 6.1 (Teorema de Geoffrion) *Seja $\mathcal{S}$ um conjunto convexo de soluções e sejam $f_1, \dots, f_K$ funções convexas em $\mathcal{S}$. Um ponto x^0 pertence ao conjunto Pareto ótimo se e somente se $\exists \alpha \in \mathbb{R}^K$, com $\alpha_i \in]0; 1[$ e $\sum_1^K \alpha_i = 1$, tal que x^0 é uma solução ótima do problema:*

$$\text{Min } F(x) = \sum_1^K \alpha_i \cdot f_i(x) \tag{6.3}$$

sujeito a:

$$x \in \mathcal{S}$$

O teorema 6.1 é conhecido como Teorema de Geoffrion e a sua demonstração encontra-se em Tkindt e Billaut (2006).

Quando as funções $f_1, \dots, f_K$ não são convexas, apenas a condição necessária é válida, ou seja, a solução ótima do problema 6.3 faz parte do conjunto ótimo de Pareto, mas o método de *Soma Ponderada* não é capaz de encontrar todas as soluções ótimas do problema.

Método ε -Restrito

É um método frequentemente encontrado na literatura, que consiste em otimizar um dos K objetivos estabelecendo valores mínimos ou limitantes para os demais objetivos. Por exemplo, sendo o primeiro objetivo escolhido, o problema pode ser formulado da seguinte maneira:

$$\text{Min } f_1(x) \tag{6.4}$$

sujeito a:

$$x \in \mathcal{S}$$

$$f_i(x) \leq \varepsilon_i^1 \quad \forall i \in [2; K]$$

Uma desvantagem do método ε -*Restrito* é a necessidade de um conhecimento prévio do problema, pois limitantes escolhidos inadequadamente podem torná-lo ineficaz. Por outro lado, o problema pode ser visto como um *método iterativo*, onde o decisor pode especificar e modificar os limitantes e analisar o impacto dessas variações no resultado final. Do mesmo modo, o resultado obtido ao otimizar um objetivo pode ser utilizado como valor limitante deste objetivo quando outro for otimizado.

Através desse método, conforme apresenta o teorema a seguir, pode-se determinar o conjunto ótimo de Pareto. A prova deste teorema também se encontra em Tkindt e Billaut (2006).

Teorema 6.2 $x^0 \in \mathcal{S}$ pertence ao conjunto ótimo de Pareto se e somente se para $\forall k \in [1; K] \exists \varepsilon^k = (\varepsilon_1^k, \dots, \varepsilon_{k-1}^k, \varepsilon_{k+1}^k, \dots, \varepsilon_K^k) \in \mathbb{R}^{K-1}$ tal que $f(x^0)$ é a única solução ótima do problema a seguir:

$$\text{Min } f_k(x) \tag{6.5}$$

sujeito a:

$$x \in \mathcal{S}$$

$$f_i(x) \leq \varepsilon_i^k \quad \forall i \in [1; K], i \neq k$$

Ordem Lexicográfica

Neste método os objetivos são ordenados por ordem de importância e a solução ótima é obtida otimizando as funções objetivo, começando pela mais importante ($f_1(x)$), depois

a segunda ($f_2(x)$) e assim por diante, até a última função ($f_K(x)$). Portanto, o primeiro problema é formulado do seguinte modo:

$$\min f_1(x) \tag{6.6}$$

sujeito a:

$$x \in \mathcal{S}$$

Seja x_1^* e $f_1^* = f_1(x_1^*)$ a solução obtida no problema 6.6. O segundo problema tem a seguinte formulação:

$$\min f_2(x) \tag{6.7}$$

sujeito a:

$$f_1(x) = f_1^*$$

$$x \in \mathcal{S}$$

Seja x_2^* e $f_2^* = f_2(x_2^*)$ a solução obtida no problema 6.7. Este procedimento é repetido para os K objetivos, tendo a seguinte formulação geral:

$$\min f_i(x) \tag{6.8}$$

sujeito a:

$$f_j(x) = f_j^*, j = 1, \dots, i - 1$$

$$x \in \mathcal{S}$$

x_K^* é a solução ótima do problema.

Este método, apesar de privilegiar os primeiros objetivos em detrimento dos outros, é similar ao método de *Soma Ponderada* se se escolher aleatoriamente a ordem dos objetivos. Tkindt e Billaut (2006) apresentam a prova matemática de que x_K^* pertence ao conjunto ótimo estrito de Pareto e que x_k^* , $1 \leq k \leq K$, pertence ao conjunto ótimo fraco de Pareto.

Ordem Lexicográfica com Metas

Trata-se de uma derivação do método *Ordem Lexicográfica*, onde os objetivos – além de serem ordenados por ordem de importância – têm uma meta ser atingida. Desse modo,

ao resolver as diversas fases do método, procura-se não minimizar a função objetivo original, mas sim a solução mais próxima da meta.

Seja T_i a meta de cada objetivo i , $1 \leq i \leq K$. Analogamente à *Ordem Lexicográfica*, K problemas são resolvidos com a seguinte formulação:

$$\min |f_i(x) - T_i| \quad (6.9)$$

sujeito a:

$$\begin{aligned} f_j(x) &= f_j^*, j = 1, \dots, i-1 \\ x &\in \mathcal{S} \end{aligned}$$

Ao contrário da *Ordem Lexicográfica*, não se pode garantir que a solução obtida pelo método *Ordem Lexicográfica com Metas* pertença ao conjunto ótimo fraco de Pareto.

6.1.3 Meta-heurísticas

Para encontrar um conjunto de soluções *não dominadas* para um problema multiobjetivo NP-difícil, algoritmos de natureza heurística também foram propostos na literatura, tendo em conta a complexidade desses problemas. Apesar de não existir garantia de que estes algoritmos gerarão soluções que pertencem ao conjunto ótimo fraco de Pareto, eles apresentam um bom desempenho na literatura.

Diversas meta-heurísticas foram utilizadas em problemas multiobjetivos (COELLO; LAMONT; VAN VELDHUIZEN, 2007; DONOSO; FABREGAT, 2007; TKINDT; BILLAUT, 2006; ABRAHAM; JAIN, 2005). A seguir, encontram-se algumas das meta-heurísticas mais utilizadas:

- *Simulated annealing (SA)*: os diferentes critérios são combinados em uma única função, como no método *Soma Ponderada*, e o SA tradicional encontra uma solução para o problema. Variando os pesos das funções, pode-se resolver o problema diversas vezes e obter um conjunto de boas soluções, candidatas a formarem o conjunto potencial ótimo de Pareto. Alguns trabalhos na área de *scheduling* utilizaram essa meta-heurística: Lin e Ying (2013); Mohammadi; Karampourhaghghi e Samaei (2012), e Vieira e Ribas (2008).
- *Busca Tabu (BT)*: explora um conjunto de soluções paralelamente, cada uma com a sua própria lista tabu. Para cada solução determina-se, por exemplo, um vetor

de pesos associado aos objetivos para definir uma direção de busca sobre o espaço objetivo. Ao terminar a busca, as soluções não dominadas formarão o conjunto potencial ótimo de Pareto. Em *scheduling*, alguns artigos que utilizaram a BT multiobjetivo foram: Shahnazari-Shahrezaei; Tavakkoli-Moghaddam e Kazemi-poor (2013) para estudar a alocação de trabalhadores com diferentes habilidades, Li; Pan e Liang (2010) no problema *Job Shop* Flexível; Choobineh; Mohebbi e Khoo (2006) em um problema de uma única máquina com *set up* dependente da sequência das tarefas; e Arroyo (2002) no problema de *Flow Shop* e no problema da mochila.

- *Algoritmos Genéticos (AG)*: como trabalham com populações de soluções, os AG foram muito utilizados nesses problemas e surgiram diversas versões de AG multiobjetivos: *Vector Evaluated Genetic Algorithms* (VEGA) proposto por Schaffer em 1984, *Multi-objective Optimization Genetic Algorithm* (MOGA) sugerido por Fonseca e Fleming em 1993, *Niched Pareto Genetic Algorithm* (NPGA) de Horn, Nafploitis e Goldberg (1994), *Nondominated Sorting Genetic Algorithm* (NSGA) desenvolvido por Srinivas e Deb em 1994, *Strength Pareto Evolutionary Algorithm* (SPEA) de Zitzler e Thiele em 1999 (DEB, 2004; RADHI; BARRANS, 2012; ZITZLER; LAUMANN; BLEULER, 2004; ABRAHAM; JAIN, 2005). Como concepção geral, esses AG procuram selecionar as soluções não dominadas, que recebem uma alta avaliação (ou aptidão) e que terão maior probabilidade de gerar descendentes. Para manter a diversidade da população, alguns métodos de diversificação são utilizados, como ajustar a aptidão das soluções de acordo com o número de soluções dentro de uma vizinhança ou procurar evitar a formação de nichos de soluções. Algumas pesquisas recentes com AG multiobjetivos: Xiao et al. (2016) aplicaram o AG multiobjetivo para o problema de programação de projetos; Tangpattanakul; Jozefowicz e Lopez (2015) utilizaram o AG no problema de sequenciamento de observação de satélites; Arroyo e Armentano (2005) e Moritz et al. (2015) estudaram o problema *Flow Shop* multiobjetivo; Rahmati; Zandieh e Yazdani (2013) estudaram o problema *Job Shop* flexível multiobjetivo, Samanlioglu; Ferrell e Kurz (2008), o problema do Caixeiro Viajante e Ghoseiri e Ghannadpour (2010), o problema de Roteamento de Veículos com Janela de Tempo.

- *Colônia de Formigas (Ant Colony)*: assim como o AG, trabalha com populações de soluções e pode ser bem adaptado a problemas multiobjetivos. Moritz et al. (2015) e Yagmahan e Yenisey (2008) aplicaram a Colônia de Formigas ao problema *Flow Shop* multiobjetivo e Chang et al. (2008), ao *Job Shop* multiobjetivo.
- Outras metas-heurísticas como o GRASP (MARTÍ et al., 2015), a *Iterated Local Search* (GOMES; DAS NEVES; SOUZA, 2014), a ALNS (DEMIR; BEKTAŞ; LAPORTE, 2014), a *Artificial Bee Colony* (ABC) (XIANG; ZHOU; LIU, 2015) e a *Particle Swarm Optimization* (PSO) (SHAHNAZARI-SHAHREZAEI; TAVAKKOLI-MOGHADDAM; KAZEMIPOOR, 2013) também foram aplicados aos problemas multiobjetivos.

6.2 Medidas de avaliação das soluções

Para avaliar os resultados entre os diferentes métodos, quatro medidas comparativas são adotadas (ZITZLER; KNOWLES; THIELE, 2008): porcentagem de soluções eficientes (I_{PSE}), hipervolume (I_H), indicador epsilon multiplicativo (I_ϵ) e medida do conjunto de cobertura (Cov). Essas medidas foram adotadas tendo em conta a literatura recente (AQUINO; ARROYO, 2015; CASJENS et al., 2015; DEMIR; BEKTAŞ; LAPORTE, 2014; LIN; YING, 2013; MARTÍ et al., 2015; MORITZ et al., 2015; TANGPATTANAKUL; JOZEFOWIEZ; LOPEZ, 2015; XIANG; ZHOU; LIU, 2015; XIAO et al., 2016). A seguir encontram-se as definições desses indicadores para problemas de maximização.

6.2.1 Porcentagem de soluções eficientes

A primeira medida representa a quantidade de soluções do conjunto referência encontrada por uma heurística. Esse indicador foi proposto na literatura por Ulungu et al. (1999) como porcentagem da fronteira de Pareto fracamente dominada por um conjunto de soluções. Seja $\mathbf{P}_1$ o conjunto das soluções x não dominadas de um método heurístico H_1 e $\mathbf{Ref}$ o conjunto de pontos z de referência (soluções não dominadas entre todos os métodos avaliados). A porcentagem de soluções eficientes encontradas por H_1 é:

$$I_{PSE}(H_1) = 100 \cdot \frac{|\{z | \exists x \in \mathbf{P}_1 : f(x) \succeq z\}|}{|\mathbf{Ref}|} \quad (6.10)$$

6.2.2 Hipervolume

Proposto na literatura por Zitzler e Thiele (1999), o indicador hipervolume representa o volume do espaço definido pelas soluções fracamente dominadas por um método. Formalmente o hipervolume é definido como:

$$I_H(H_1) := \int_{z_{inferior}}^{z_{superior}} \alpha_{\mathbf{Ref}}(z) dz \quad (6.11)$$

onde $z_{inferior}$ e $z_{superior}$ são os limitantes inferiores e superiores da parte do espaço em que o hipervolume é calculado e $\alpha_{\mathbf{Ref}}$ é a função obtenção (*the attainment function*) (FONSECA; FONSECA; HALL, 2001) definida por

$$\alpha_{\mathbf{Ref}}(z) := \begin{cases} 1 & \text{se } \exists x \in \mathbf{P}_1 : f(x) \succeq z \\ 0 & \text{caso contrário} \end{cases} \quad (6.12)$$

que é igual a 1 quando o método H_1 encontra uma solução x fracamente dominada por $\mathbf{Ref}$.

Para possibilitar a comparação de diferentes métodos e instâncias neste trabalho, define-se também a porcentagem do hipervolume de cada método em relação ao conjunto referencia:

$$I_{PH}(H_1) = 100 \cdot \frac{I_H(\mathbf{H}_1)}{I_H(\mathbf{Ref})} \quad (6.13)$$

6.2.3 Indicador epsilon

Proposto na literatura por Zitzler et al. (2003), o indicador epsilon multiplicativo (I_ϵ) é um número real ϵ tal que se os valores das funções objetivo de cada ponto z do conjunto de referência ($\mathbf{Ref}$) forem multiplicados por ϵ obter-se-á um novo conjunto com soluções fracamente dominadas pelas soluções P_1 de um método H_1 :

$$I_\epsilon(\mathbf{H}_1) = \min\{\epsilon \in R \mid \forall z \in \mathbf{Ref} \exists x \in \mathbf{P}_1 : f(x) \succeq_\epsilon z\} \quad (6.14)$$

onde

$$f(x) \succeq_\epsilon z \iff \forall i \in 1..n : f_i(x) \geq \epsilon \cdot z_i \quad (6.15)$$

Outra medida da família ϵ , que não é utilizada nesse trabalho, é o indicador epsilon

aditivo ($I_{\epsilon+}$), definido analogamente a (6.14) como:

$$f(x) \succeq_{\epsilon+} z \iff \forall i \in 1..n : f_i(x) + \epsilon \geq z_i \quad (6.16)$$

Em outras palavras, a família de indicadores ϵ quantifica um número que transforma um conjunto (**Ref**) em soluções piores do que as soluções de outro conjunto ($\mathbf{P}_1$) em todos os objetivos ¹.

6.2.4 Medida do conjunto de cobertura

O último indicador apresentado é a *medida do conjunto de cobertura* (Cov) (ZITZLER; THIELE, 1999), que é a fração das soluções não dominadas de uma heurística (H_2) que são cobertas pelas soluções eficientes outra heurística (H_1). Uma solução a cobre uma solução b se a domina b ou se $a = b$. Sejam $\mathbf{P}_1$ e $\mathbf{P}_2$ os conjuntos das soluções não dominadas das heurísticas H_1 e H_2 . Essa medida é definida por:

$$Cov(\mathbf{P}_1, \mathbf{P}_2) = \frac{|\{b \in \mathbf{P}_2; \exists a \in \mathbf{P}_1 : a \text{ cobre } b\}|}{|\mathbf{P}_2|} \quad (6.17)$$

Quando $Cov(\mathbf{P}_1, \mathbf{P}_2) = 0$, nenhum ponto do conjunto $\mathbf{P}_2$ é coberto pelo conjunto $\mathbf{P}_1$. Quando $Cov(\mathbf{P}_1, \mathbf{P}_2) = 1$ então todos os pontos de $\mathbf{P}_2$ são dominados ou são iguais a um elemento de $\mathbf{P}_1$. Note que $Cov(\mathbf{P}_1, \mathbf{P}_2)$ não necessariamente é igual a $1 - Cov(\mathbf{P}_2, \mathbf{P}_1)$.

6.2.5 Exemplo

Para visualizar essas medidas de qualidade, considere a seguir o exemplo de um problema de maximização com três métodos heurísticos (H_1 , H_2 e H_3). Seja $\mathbf{P}_1$, $\mathbf{P}_2$ e $\mathbf{P}_3$, respectivamente, os conjuntos das soluções não dominadas de H_1 , H_2 e H_3 , de acordo com a Tabela 6.1 e o Gráfico 6.4. O conjunto referência (**Ref**) formado pelas soluções não dominadas de todos os métodos é: $\mathbf{Ref} = \{(5,0; 7,5); (6,0; 7,2); (6,8; 7,0); (7,1; 6,9); (7,5; 6,6); (8,0; 6,2); (9,5; 4,2); (10,0; 2,0)\}$.

¹Os indicadores ϵ também são definidos entre duas heurísticas H_1 e H_2 (com conjunto de soluções eficientes respectivamente $\mathbf{P}_1$ e $\mathbf{P}_2$), onde $\mathbf{P}_2$ substitui o conjunto **Ref** nas definições apresentadas anteriormente; em vez de $I_\epsilon(\mathbf{H}_1)$ ou $I_{\epsilon+}(\mathbf{H}_1)$ tem-se $I_\epsilon(\mathbf{H}_1, \mathbf{H}_2)$ ou $I_{\epsilon+}(\mathbf{H}_1, \mathbf{H}_2)$.

Tabela 6.1 – Soluções eficientes de três métodos heurísticos

P_1	P_2	P_3
(5,0; 7,3)	(6,0; 7,2)	(5,0; 7,5)
(6,7; 6,9)	(6,8; 7,0)	(6,7; 6,9)
(8,0; 6,2)	(7,1; 6,9)	(8,0; 6,0)
(9,5; 4,2)	(7,5; 6,6)	(9,3; 4,1)
(9,7; 2,0)	(8,0; 5,0)	(10,0; 2,0)

Figura 6.4 – Soluções eficientes de três métodos heurísticos

Fonte: elaborado pelo autor.

Dessa forma, tem-se os seguintes indicadores:

1. Porcentagem de soluções eficientes: o conjunto de referência **Ref** tem 2, 4 e 2 soluções fracamente dominadas pelos conjuntos P_1 , P_2 e P_3 , respectivamente. Como a cardinalidade de **Ref** é 8, tem-se os seguintes índices: $I_{PSE}(H_1) = 25\%$, $I_{PSE}(H_2) = 50\%$ e $I_{PSE}(H_3) = 25\%$.
2. Hipervolume: para calcular esse indicador, é adotado o vetor (4,0; 1,0) como ponto de referência (limite inferior do hipervolume, uma vez que o limite superior é dado pela fronteira encontrada por cada método). A Figura 6.5 apresenta graficamente a área definida pelo ponto de referência e as soluções eficientes de cada um dos métodos heurísticos, que são: $I_H(H_1) = 28,09$, $I_H(H_2) = 23,21$ e $I_H(H_3) = 27,76$. O hipervolume do conjunto **Ref** é $I_H(Ref) = 29,41$ e, portanto, as porcentagens

de hipervolume dos métodos em relação ao conjunto referência são: $I_{PH}(H_1) = 95,5\%$, $I_{PH}(H_2) = 78,9\%$ e $I_{PH}(H_3) = 94,4\%$.

Figura 6.5 – Hipervolume dos métodos heurísticos com relação ao ponto de referência (4,0; 1,0)

Fonte: elaborado pelo autor.

3. Indicador épsilon multiplicativo: para o método heurístico H_1 o indicador é $I_\epsilon(H_1) = 0,94$. Note que o ponto $(7,5; 6,6)$ do conjunto **Ref** somente é dominado por uma solução de H_1 quando multiplicado por 0,94, resultando no ponto $(7,0; 6,2)$, que é dominado pela solução eficiente de H_1 com coordenadas $(8,0; 6,2)$. Com valores superiores a 0,94, o valor da segunda função objetivo de $(\epsilon \cdot 7,5, \epsilon \cdot 6,6)$ seria superior a 6,2, não sendo dominado por nenhuma solução eficiente de H_1 , como pode ser observado na Figura 6.6(a). Analogamente, tem-se que $I_\epsilon(H_2) = 0,80$ e $I_\epsilon(H_3) = 0,91$. A Figura 6.6 apresenta o conjunto referência, as soluções eficientes de cada método e as soluções do conjunto referência multiplicadas por ϵ .

Figura 6.6 – Conjuntos referência e soluções eficientes de cada heurística, com as soluções do conjunto referência multiplicadas por ϵ . As soluções que definem o valor de ϵ estão indicadas em cada uma das figuras.

Fonte: elaborado pelo autor.

Na prática, I_ϵ de um método heurístico H com um conjunto P de soluções eficientes é calculado da seguinte maneira (Zitzler et al., 2003):

$$I_\epsilon(H) = \min_{z \in \mathbf{Ref}} \max_{x \in \mathbf{P}} \min \left\{ \frac{f_1(x)}{f_1(z)}, \frac{f_2(x)}{f_2(z)} \right\} \quad (6.18)$$

4. Medida do conjunto de cobertura: comparando os métodos H_1 e H_2 , pode-se observar que apenas a solução eficiente de H_1 com valores (6,7; 6,9) é coberta por uma solução eficiente de H_2 (6,8; 7,0). Portanto, $Cov(\mathbf{H}_1, \mathbf{H}_2) = 1/5 = 0,2$. A Tabela 6.2 apresenta a medida do conjunto de cobertura entre os três métodos heurísticos do exemplo.

Tabela 6.2 – Medidas do conjunto de cobertura

	$Cov(*, \mathbf{H}_1)$	$Cov(*, \mathbf{H}_2)$	$Cov(*, \mathbf{H}_3)$
$Cov(\mathbf{H}_1, *)$	-	0,2	0,6
$Cov(\mathbf{H}_2, *)$	0,2	-	0,2
$Cov(\mathbf{H}_3, *)$	0,6	0,2	-

A Tabela 6.3 apresenta um resumo dos três primeiros índices de medida, onde as heurísticas que obtiveram os maiores índices apresentam os melhores resultados. O método heurístico H_2 supera os outros métodos com relação ao indicador I_{PSE} . Já com relação aos índices I_{PH} e I_ϵ , a heurística H_1 obteve o melhor desempenho. Com relação ao conjunto de cobertura, pode-se observar na Tabela 6.2 que os três métodos têm igual desempenho, pois $Cov(H_1, H_2) = Cov(H_2, H_1)$, $Cov(H_1, H_3) = Cov(H_3, H_1)$ e $Cov(H_2, H_3) = Cov(H_3, H_2)$. Através desse exemplo, pode-se concluir que a comparação entre os métodos multiobjetivos depende, entre outros fatores, dos indicadores de desempenho adotado (ZITZLER; KNOWLES; THIELE, 2008). Por isso, neste trabalho são adotadas as quatro medidas apresentadas, de modo a buscar uma análise mais abrangente dos métodos desenvolvidos. Antes de concluir esse capítulo, pode-se ressaltar que o índice hipervolume é sensível ao ponto de referência adotado. Por exemplo,

Tabela 6.3 – Índices de medida para o exemplo

Medida	H_1	H_2	H_3
$I_{PSE}(\%)$	25	50	25
$I_{PH}(\%)$	95,5	78,9	94,4
I_ϵ	0,94	0,80	0,91

se o ponto $(0,0; 0,0)$ fosse adotado como referência em vez de $(4,0; 1,0)$, a heurística H_3 superaria as demais com relação a esse índice, pois $I_{PH}(P_1) = 96,3\%$, $I_{PH}(P_2) = 85,6\%$ e $I_{PH}(P_3) = 97,5\%$. Ao apresentar os resultados computacionais no Capítulo 9, é feita uma análise de sensibilidade entre o índice I_{PH} e o ponto de referência adotado.

Capítulo 7

Modelo Multiobjetivo

Este capítulo inicialmente apresenta a proposta de uma nova função objetivo para o PETC. A segunda seção descreve o modelo multiobjetivo de Programação Linear Inteira Mista e a última seção do capítulo apresenta um exemplo do PETC multiobjetivo.

7.1 Proposta de uma nova função objetivo

No modelo mono-objetivo estudado na Parte I deste trabalho (ver Seção 1.4), ao maximizar o número ponderado de tarefas atendidas e, secundariamente, a ociosidade após o retorno à base, pôde-se notar algumas limitações:

- As restrições do modelo garantem que as janelas de tempo são respeitadas, mas não consideram se as tarefas prioritárias poderiam ser atendidas mais cedo. Uma solução ótima pode alocar uma tarefa prioritária para o fim da sua janela de tempo e – como na prática existem atrasos imponderáveis, como o trânsito ou o atraso no processamento de tarefas antecessoras – fatores externos poderão causar o cancelamento da execução de tarefas programadas.
- Em algumas situações, uma solução ótima pode ter ociosidades de técnicos antes da execução das tarefas e, ao mesmo tempo, essas tarefas – que poderiam ser realizadas no início ou próximo do início de sua janela de tempo – são executadas mais tarde.

Para melhorar estes aspectos, propõe-se introduzir uma nova função objetivo para o problema: procurar que as tarefas prioritárias sejam atendidas o mais cedo possível. Esse novo objetivo foi inspirado nos trabalhos de Cordeau et al. (2010), Hashimoto et

al. (2011) e Cortés et al. (2014). Nos dois primeiros trabalhos as tarefas foram divididas em três níveis de prioridade e a soma ponderada do instante de término da última tarefa de cada grupo foi minimizada, de modo que as tarefas do grupo mais prioritário fossem executadas antes das demais. Já no terceiro trabalho, a empresa estabelece uma janela de tempo flexível para cada serviço, que representa a prioridade do cliente, ou seja, quanto mais importante é o cliente, mais cedo é o fim da sua janela de tempo. Ao contrário desses autores, que consideram ou prioridades ou janelas de tempo, o problema desta tese considera simultaneamente janelas de tempo e prioridades e, portanto, é necessária uma nova abordagem para essa proposta.

Seja uma variável *folga* (γ) que é a fração de tempo disponível entre o fim da execução de uma tarefa e o término da sua janela de tempo, representando o quão cedo ela é concluída. Quando uma tarefa começar a ser executada exatamente no início da sua janela de tempo ($t_i = e_i$), então haverá 100% de *folga* ($\gamma = 1$); quando terminar de ser executada exatamente no fim da janela ($t_i = l_i - p_i$), a *folga* é igual a um número muito pequeno ($\gamma = \frac{\varepsilon}{(l_i - e_i - p_i + \varepsilon)}$); e a folga receberá o valor nulo ($\gamma = 0$) somente quando uma tarefa não for realizada. Portanto, a variável *folga* é definida por:

$$\gamma_i = \frac{(l_i - t_i - p_i + \varepsilon)}{(l_i - e_i - p_i + \varepsilon)} \quad (7.1)$$

onde:

ε é um número muito pequeno

Repare que o parâmetro ε também garante que o denominador de (7.1) nunca é igual a zero, o que pode ocorrer quando a amplitude da janela de tempo de uma tarefa for igual ao seu tempo de processamento ($p_i = l_i - e_i$).

Portanto, o novo objetivo é:

$$\max \sum_{i \in J} \frac{w_i \cdot \gamma_i}{MW} \quad (7.2)$$

onde prioridade (w_i) tem o intuito de que, ao maximizar esse novo objetivo, as folgas das tarefas mais prioritárias sejam maiores e, portanto, elas sejam executadas mais cedo. O denominador MW foi incluído para que esse novo objetivo tenha a mesma ordem de grandeza do objetivo proposto no modelo mono-objetivo (ver equação 1.1).

7.2 Modelo PLIM Multiobjetivo

Dessa forma, tem-se o seguinte o modelo multiobjetivo para o PETC:

$$\max f(x, y, t, z) = (f_1(x, y, t, z), f_2(x, y, t, z)) \quad (7.3)$$

sujeito a:

$$\gamma_i = \frac{(l_i - t_i - p_i + \varepsilon)}{(l_i - e_i - p_i + \varepsilon)} \\ x, y, t, z \in X^*$$

onde:

$$f_1(x, y, t, z) = \sum_{i \in J} \sum_{\substack{k \in K \\ |s_{ik}=1}} \frac{w_i y_{ik}}{MW} + \sum_{k \in K} \frac{z_k}{MZ}, \\ f_2(x, y, t, z) = \sum_{i \in J} \frac{w_i \gamma_i}{MW}$$

ε é um número muito pequeno

X^* é formado pelas restrições do modelo mono-objetivo (1.2–1.11), exceto a restrição (1.5), e também pelas restrições:

$$t_i \geq e_i \quad i \in J \quad (7.4)$$

$$t_i \geq l_i - p_i - M \sum_{\substack{k \in K \\ |s_{ik}=1}} y_{ik} + \varepsilon \quad i \in J \quad (7.5)$$

$$t_i \leq l_i - p_i + M \sum_{\substack{k \in K \\ |s_{ik}=1}} y_{ik} + \varepsilon \quad i \in J \quad (7.6)$$

As restrições (7.5) e (7.6) garantem que, quando uma tarefa não for executada, a sua folga é nula no novo objetivo a ser maximizado. No modelo mono-objetivo, quando uma tarefa não era alocada a nenhum técnico ($\sum_{k \in K} y_{ik} = 0$), a variável t_i perdia o seu significado, podendo assumir qualquer valor arbitrário respeitando as demais restrições. Note que, no modelo multiobjetivo, as restrições (7.5) e (7.6) são ativadas quando uma tarefa não for programada na rota de nenhum técnico e, portanto, tem-se que $t_i = l_i - p_i + \varepsilon$, resultando em $\gamma_i = 0$.

Tabela 7.1 – Parâmetros do exemplo: a) prioridades, tempos de execução e janelas de tempo das tarefas. b) Tempo de deslocamento c_{ij} (em minutos) entre as tarefas e a sede da empresa.

(a)										
Tarefa	1	2	3	4	5	6	7	8	9	10
Prioridade (w)	5	10	8	1	9	2	1	3	6	6
Tempo de execução (p)	90 min	75 min	60 min	85 min	45 min	30 min	45 min	40 min	35 min	45 min
Início da janela de tempo (e)	11h20min	10h20min	10h30min	9h25min	16h10min	8h35min	10h25min	10h30min	13h20min	11h25min
Fim da janela de tempo (l)	19h15min	12h25min	15h60min	19h60min	18h25min	15h00min	13h30min	16h40min	16h0min	17h50min

(b)											
c_{ij}	Emp.	1	2	3	4	5	6	7	8	9	10
Emp.	0	15	18	22	25	20	11	21	26	31	25
1	15	0	32	14	32	32	24	21	31	17	15
2	18	32	0	34	20	23	16	36	35	46	42
3	22	14	34	0	25	42	33	35	44	15	29
4	25	32	20	25	0	40	31	45	50	39	46
5	20	32	23	42	40	0	10	20	14	49	33
6	11	24	16	33	31	10	0	20	19	42	30
7	21	21	36	35	45	20	20	0	12	36	14
8	26	31	35	44	50	14	19	12	0	47	26
9	31	17	46	15	39	49	42	36	47	0	25
10	25	15	42	29	46	33	30	14	26	25	0

7.3 Exemplo

Para ilustrar o problema multiobjetivo, a seguir encontra-se um exemplo com dez tarefas e dois técnicos. O primeiro técnico trabalha entre 8h e 17h e pode executar todas as tarefas, exceto as tarefas 1, 3 e 5; o segundo técnico trabalha das 11h às 20h e somente não pode executar as tarefas 2 e 8. A Tabela 7.1 apresenta os demais parâmetros do problema.

A Figura 7.1(a) mostra a solução do modelo mono-objetivo, que maximiza f_1 , e a Tabela 7.2 apresenta nas colunas 3 e 4 o momento do início da execução de cada tarefa e a sua respectiva folga. Todas as tarefas são realizadas (o valor da função objetivo é $f_1 = 51,4$), no entanto, pode-se observar que as tarefas 2, 3 e 5, que são as mais prioritárias, estão programadas para o fim de suas janelas de tempo e têm folgas pequenas. Concretamente, a tarefa 2 começa a ser executada às 11h10min e é concluída às 12h25min, que é exatamente o fim de sua janela de tempo.

Para aumentar essas folgas, o exemplo é resolvido pelo método ϵ -restrito, onde o valor mínimo adotado para a primeira função objetivo é de 45,0 (ou seja, admite-se que algumas tarefas com soma das prioridades de até 6 possam não ser executadas). A Figura 7.1(b) apresenta a solução obtida, onde os valores das funções objetivo são

Tabela 7.2 – Momento de início da execução das tarefas (t) e respectivas folgas (γ) de três soluções do exemplo

tarefa	prioridade	mono-objetivo		ϵ -restrito (1)		ϵ -restrito (2)	
		$\max f_1$		$\max f_2$		$\max f_2$	
		s.a. $f_1 > 45,0$		s.a. $f_1 > 51,0$		s.a. $f_1 > 51,0$	
		t	γ	t	γ	t	γ
1	5	15h34min	34%	12h36min	80%	13h51min	61%
2	10	11h10min	0% ^a	10h20min	100%	10h20min	100%
3	8	14h20min	15%	11h22min	81%	11h22min	81%
4	1	9h25min	100%	17h35min	11%	17h35min	11%
5	9	17h36min	6%	16h10min	100%	16h10min	100%
6	2	12h41min	31%	8h35min	100%	8h35min	100%
7	1	11h21min	60%	-	-	12h11min	24%
8	3	13h30min	45%	14h49min	21%	14h54min	20%
9	6	13h30min	93%	13h27min	94%	13h32min	90%
10	6	12h20min	84%	12h17min	85%	12h51min	75%
Média		-	47%	-	75%	-	66%

^a Valor próximo de zero: $\gamma = \frac{\epsilon}{(l_i - e_i - p_i + \epsilon)}$

Figura 7.1 – Exemplo com 10 tarefas e 2 técnicos: a) solução do modelo mono-objetivo ($\max f_1$); b) solução do modelo multiobjetivo (método ϵ -restrito: $\max f_2$ s.a. $f_1 \geq 45,0$); c) solução do modelo multiobjetivo (método ϵ -restrito: $\max f_2$ s.a. $f_1 \geq 51,0$)

Fonte: elaborado pelo autor.

$f_1 = 50,2$ e $f_2 = 43,0$, e nas colunas 5 e 6 da Tabela 7.2 encontra-se o momento de início da execução de cada tarefa e as respectivas folgas (com subtítulo ϵ -restrito (1)). Ao priorizar a segunda função objetivo, a tarefa 7 (prioridade 1) deixou de ser executada e as folgas de todas as tarefas de prioridade maior que 3 aumentaram. Note que as duas tarefas mais prioritárias agora são executadas no início de sua janela de tempo, com folga igual a 100% e que a média das folgas das tarefas executadas aumentou de 47% para 75%.

Finalmente, pode-se considerar uma terceira possibilidade, em que o decisor queira executar todas as tarefas e ao mesmo tempo aumentar a folga das tarefas prioritárias.

Nesse caso, o método ϵ -restrito exige que $f_1 \geq 51,0$, uma vez que a soma das prioridades de todas as tarefas é 51. A solução encontra-se na Figura 7.1(c) e as variáveis t e γ nas duas últimas colunas da Tabela 7.2 (com subtítulo ϵ -restrito (2)). Os valores das funções objetivo são $f_1 = 51,2$ e $f_2 = 41,4$. Note que as tarefas com *folgas* pequenas têm baixas prioridades e as tarefas com altas prioridades têm *folgas* grandes; ao mesmo tempo, a média das *folgas* diminuiu de 75% para apenas 66% e as três tarefas mais prioritárias mantiveram igual valores de *folga* em relação à segunda resolução.

A partir deste pequeno exemplo, pode-se concluir que soluções intermediárias, entre os pontos extremos, podem apresentar um bom equilíbrio entre as duas funções objetivo e tornarem-se opções vantajosas para o decisor. Ou seja, a otimização multiobjetivo pretende oferecer as soluções extremas e boas soluções intermediárias, de modo que seja possível avaliar até que ponto se aceita não executar algumas tarefas menos importantes para atender os clientes mais importantes mais cedo, ou até que ponto se prefere atendê-los mais tarde para executar o máximo de tarefas possíveis.

Capítulo 8

BRKGA Multiobjetivo

Conforme apresentado na Seção 6.1.3, diversos AG foram desenvolvidos para problemas multiobjetivos, mas até o momento, apenas dois trabalhos adaptaram o BRKGA para um problema multiobjetivo. Tangpattanakul; Jozefowicz e Lopez (2015) desenvolveram um BRKGA multiobjetivo para estudar o problema de sequenciamento de observações de satélites. Já Roque; Fontes e Fontes (2013) associaram o BRKGA ao NSGAI para estudar o problema de redução de custos e do impacto ambiental em usinas de energia.

Tradicionalmente, os AG multiobjetivos trabalham com o conceito de Pareto: gera-se um conjunto de soluções, onde as não dominadas formam uma estimativa do conjunto de Pareto. Para desenvolver uma versão multiobjetivo do BRKGA, é necessário determinar inicialmente como classificar as soluções de uma população. Tangpattanakul; Jozefowicz e Lopez (2015) e Roque; Fontes e Fontes (2013) optaram por incluir no conjunto elite apenas as soluções não dominadas. Portanto, foi adotado um tamanho do conjunto elite variável, dependente do número de soluções não dominadas encontradas; quando esse número excede um valor máximo, então um método (distância de *Crowding*, que é apresentado na seção 8.1.2) determina quais soluções não dominadas são incluídas no conjunto elite.

Na primeira seção deste capítulo serão apresentados quatro métodos para classificar as soluções obtidas pelo BRKGA multiobjetivo e, em seguida (Seção 8.2), diferentes estratégias para formar o conjunto elite. O capítulo termina (Seção 8.3) com uma descrição de três diferentes decodificadores desenvolvidos para o PETC biobjetivo.

8.1 Métodos de classificação das soluções

Quatro métodos serão apresentados nessa seção: fronteiras de Pareto, fronteiras de Pareto com distância de *crowding*, fronteiras de Pareto com densidade de vizinhança e fronteiras de Pareto com densidade baseada em células (COELLO; LAMONT; VAN VELDHUIZEN, 2007; KONAK; COIT; SMITH, 2006; DEB, 2004; BAGCHI, 1999).

8.1.1 Fronteiras de Pareto

Uma primeira forma de classificar as soluções obtidas por um método é unicamente classificá-las de acordo com fronteiras de dominância. As soluções não dominadas são classificadas em primeiro lugar formando a primeira fronteira ($F1$); as soluções não dominadas obtidas sem considerar $F1$ formam a segunda fronteira ($F2$), sendo classificadas em segundo lugar; as soluções não dominadas obtidas sem considerar $F1$ e $F2$ formam a terceira fronteira ($F3$) e são classificadas em terceiro lugar; e assim por diante.

8.1.2 Distância de *Crowding*

Tanto Tangpattanakul; Jozefowicz e Lopez (2015) quanto Roque; Fontes e Fontes (2013) inspiraram-se no *Nondominated Sorting Genetic Algorithm II* (NSGA-II), proposto por Deb et al. (2002), para classificar as soluções. No NSGA-II, para cada solução é calcu-

Figura 8.1 – Distância de *crowding*

Fonte: elaborado pelo autor.

Algoritmo 3 Pseudocódigo da distância de *Crowding*

```

1: Seja  $l = |F_j|$ , onde  $l$  é o número de pontos da  $j$ -ésima linha frontal
2: Para  $i \leftarrow 1$  a  $l$  faça
3:    $d_i = 0$  (distância inicial)
4: fim Para
5: Para Para cada objetivo  $k$  faça
6:   ordene as soluções em ordem crescente do valor da função objetivo
7:    $d_1 = d_l = \infty$  (os pontos extremos recebem um valor muito alto)
8:   Para  $i \leftarrow 2$  a  $l - 1$  faça
9:      $d_i = d_i + |f_k(i + 1) - f_k(i - 1)|$  (distância dos vizinhos)
10:  fim Para
11: fim Para

```

lado um índice, que é o número de soluções que a dominam. As soluções que possuem índice *zero* são as não dominadas e formarão a (primeira) linha frontal de Pareto; a partir daí são definidas a segunda e as demais linhas frontais.

Para preservar a diversidade da população, o NSGA-II utiliza a distância das soluções mais próximas na mesma linha frontal, como pode ser observado na Figura 8.1. A distância de *crowding* de uma solução v é igual à soma da diferença entre os valores das funções objetivo ($\Delta f_1 + \Delta f_2$) de suas vizinhas v_1 e v_2 . Os pontos extremos recebem um valor superior aos demais e as soluções de uma mesma linha frontal são ordenadas em ordem decrescente da distância de *crowding*. O Algoritmo 3 apresenta o pseudocódigo da distância de *crowding*.

Dessa forma, a primeira proposta para classificar as soluções baseado na distância de *crowding* é ordenar os cromossomos (λ) por ordem crescente do índice das linhas frontais (primeiro as soluções não dominadas fazem parte da primeira fronteira $F1$, depois as soluções dominadas apenas pelas soluções de $F1$ formam a segunda fronteira $F2$ e assim por diante) e dentro de uma mesma linha frontal, ordenar as soluções pela ordem decrescente da distância de *crowding*.

8.1.3 Densidade de vizinhança

O método densidade de vizinhança encontra as fronteiras de Pareto do conjunto de soluções analisado e procura penalizar aquelas soluções localizadas em áreas com grande concentração, de modo que as soluções mais isoladas – que estariam em áreas menos exploradas pela busca – sejam favorecidas e se obtenha uma maior diversidade nas fronteiras de Pareto ao longo das gerações do AG. Essa ideia foi proposta por Goldberg

e Richardson (1987) e utilizada em diversos outros trabalhos (por exemplo, KIM; LIOU, 2013; DEB; GOLDBERG, 1989; ZITZLER; LAUMANN; THIELE, 2001).

Concretamente, as linhas frontais são determinadas ($F1, F2, \dots$) e cada solução (λ_i) de uma população P recebe uma avaliação $f'(\lambda_i)$ de acordo com o seguinte critério:

$$f'(\lambda_i) = \begin{cases} 1 & \text{se } \lambda_i \in F1 \\ \alpha_i \cdot nc(\lambda_i) & \text{caso contrário} \end{cases} \quad (8.1)$$

Onde:

$$nc(\lambda_i) = \sum_{\lambda_j \in P} \max \left\{ \frac{\sigma - d(\lambda_i, \lambda_j)}{\sigma}, 0 \right\} \quad (8.2)$$

$$d(\lambda_i, \lambda_j) = \sqrt{\left(\frac{f_1(\lambda_i) - f_1(\lambda_j)}{f_1^{\max} - f_1^{\min}} \right)^2 + \left(\frac{f_2(\lambda_i) - f_2(\lambda_j)}{f_2^{\max} - f_2^{\min}} \right)^2} \quad (8.3)$$

sendo d é a distância Euclidiana normalizada ($f_1^{\max}$, $f_1^{\min}$, $f_2^{\max}$ e $f_2^{\min}$ são os valores máximos e mínimos de cada objetivo dentro da população P), α_i é o índice da fronteira de λ_i e σ é o tamanho da vizinhança (ou nicho). As soluções são ordenadas por ordem crescente de $f'(\lambda_i)$, ou seja, as soluções melhores classificadas são as soluções da primeira linha frontal e, em seguida, as soluções das linhas fronteiras seguintes que estão em regiões mais isoladas.

A Figura 8.2 apresenta um esboço da vizinhança de uma solução λ_i numa população

Figura 8.2 – População P e a densidade de vizinhança de uma solução

Fonte: elaborado pelo autor.

P . Há outras 5 soluções dentro da vizinhança σ e quanto mais próximo os vizinhos estiverem de λ_i , menor será a distância, mais próximo de 1 será a fração $\frac{\sigma - d(\lambda_i, \lambda_j)}{\sigma}$ na equação (8.2) e maior será a penalidade $nc(\lambda_i)$; por outro lado, as soluções que estão na periferia da vizinhança acrescentarão um valor pequeno (próximo de zero) na penalidade $nc(\lambda_i)$. Repare também que sempre teremos $nc(\lambda_i) \geq 1$, pois $d(\lambda_i, \lambda_i) = 0$.

8.1.4 Densidade baseada em células

Abordagem adotada em vários trabalhos (por exemplo, LONG et al., 2015; YANG et al., 2013; KNOWLES; CORNE, 2000; LU; YEN, 2003; CORNE; KNOWLES; OATES, 2000), na densidade baseada em células – após as fronteiras de Pareto serem determinadas – o espaço das funções objetivo é dividido em células ou hipercubos: cada eixo da função objetivo é dividido em tamanhos iguais, como pode ser observado na Figura 8.3.

Concretamente, cada solução λ_i receberá a seguinte avaliação:

$$f'(\lambda_i) = \begin{cases} 1 & \text{se } \lambda_i \in F1 \\ \alpha_i \cdot cel(\lambda_i) & \text{caso contrário} \end{cases} \quad (8.4)$$

onde α_i é o índice da fronteira de λ_i e $cel(\lambda_i)$ é o número de soluções localizadas na mesma célula de λ_i .

Figura 8.3 – Espaço das funções objetivo e sua divisão em hipercubos, com o número de soluções em cada célula.

Fonte: elaborado pelo autor.

As soluções serão ordenadas por ordem crescente de $f'(\lambda_i)$ e, portanto, os pontos das primeiras fronteiras e que estiverem em células com menores densidades serão favorecidos. Repare que na Figura 8.3, as soluções com valores da função objetivo dentro da célula com densidade 7 serão muito penalizadas e deverão ocupar as últimas posições da classificação.

8.2 Estratégias para formar o conjunto elite

Após as soluções do BRKGA multiobjetivo serem classificadas por um dos métodos apresentados na seção anterior, é necessário decidir quais cromossomos formam parte do conjunto elite. As seguintes estratégias são adotadas neste trabalho:

1. *Tamanho do conjunto elite fixo ou variável:* uma primeira estratégia será incluir no conjunto elite da próxima geração apenas as soluções da primeira fronteira, adotando assim um tamanho variável para o conjunto elite, que será igual ao número de soluções não dominadas de cada população. Outra estratégia seria adotar um tamanho fixo para o conjunto elite, incluindo nesse conjunto as soluções da primeira fronteira e completando-o com as soluções melhores classificadas por um dos métodos anteriores. Em ambos os casos, se o número de soluções não dominadas superar o tamanho máximo do conjunto elite (primeira estratégia) ou o tamanho do conjunto elite (segunda estratégia), as soluções da primeira fronteira são ordenadas pela distância de *crowding*, pela densidade de vizinhança ou pela densidade de células, e as melhores classificadas formam o conjunto elite.
2. *Função similaridade:* Tendo em conta o bom desempenho da função similaridade no BRKGA mono-objetivo, também será avaliado o impacto de utilizar a função similaridade para selecionar os cromossomos que serão incluídos no conjunto elite. O objetivo dessa estratégia será aumentar a diversificação na busca e procurar melhorar a qualidade das soluções de fronteira. Inicialmente, toda a primeira fronteira poderá ser incluída no conjunto, que será completado pelos cromossomos melhores classificados por um dos métodos anteriores que têm uma porcentagem de chaves aleatórias iguais às soluções já incluídas no conjunto elite menores do que o grau de similaridade permitido. Outra possibilidade será incluir apenas os dois pontos extremos da primeira fronteira e também avaliar os demais pontos da

primeira fronteira de acordo com a medida de similaridade: dessa forma, evitar-se-ia incluir duas soluções da primeira fronteira que sejam muito semelhantes.

Logicamente, também será estudado o desempenho do BRKGA sem utilizar a função similaridade, tendo em conta que os métodos de classificação das soluções também de certa forma procuram – por meio dos valores das funções objetivo de cada cromossomo – diversificar as soluções.

8.3 Decodificadores

Baseado no desempenho dos decodificadores no problema mono-objetivo, decidiu-se implementar três decodificadores para o problema multiobjetivo: o *Aleatório*, o Menor Tempo de Viagem (*STT*) e um novo decodificador denominado *Balanceado*. No caso dos dois primeiros, foram duas as principais mudanças da versão mono para a multiobjetivo: foi criada uma nova regra para decidir a posição de uma nova tarefa na rota de um técnico (que leva em conta simultaneamente ambas as funções objetivo) e, para que isso seja feito, uma nova chave aleatória foi incluída em cada cromossomo. A seguir encontram-se os detalhes de cada decodificador.

8.3.1 Aleatório

Neste decodificador cada cromossomo terá $n + 1$ chaves aleatórias. As primeiras n serão as chaves de cada uma das n tarefas. Assim como no decodificador do problema mono-objetivo, para um cromossomo λ cada tarefa i será associada ao técnico TC_i dado por:

$$TC_i = \lceil \lambda[i] \cdot NST_i \rceil \quad (8.5)$$

onde $NST_i = \sum_{k \in K} s_{ik}$ é o número de técnicos com habilidade para executar a tarefa i .

Também será calculado o índice (ψ_i) para cada tarefa, definido por:

$$\psi_i = \frac{\lambda[i] - \frac{TC_i - 1}{NST_i}}{\frac{1}{NST_i}} = NST_i \cdot \lambda[i] - TC_i + 1 \quad (8.6)$$

As tarefas de um mesmo técnico k serão ordenadas pelo índice (ψ_i) . Para tentar inseri-las na rota do técnico – diferentemente do decodificador do problema mono-objetivo, que escolhia a posição que maximizava a ociosidade no fim do dia (z_k) – será escolhida a posição que maximiza uma soma ponderada dos dois objetivos do problema:

$$f_k^{sp} = \lambda[n+1] \cdot f_1 + (\lambda[n+1] - 1) \cdot f_2 \quad (8.7)$$

onde o fator de ponderação é dado pela última chave aleatória $(\lambda[n+1])$ do cromossomo e apenas as tarefas programadas no técnico k são consideradas $(y_{ik} = 1)$. A ideia da soma ponderada f_k^{sp} é representar o quanto as tarefas programadas para o técnico k representam para as funções objetivo do problema e poder avaliar se a inclusão de uma nova tarefa irá melhorar ou piorar essa ponderação entre os objetivos f_1 e f_2 .

A seguir encontra-se a descrição do pseudocódigo do decodificador aleatório.

1. *Inicialização:* Calcule o técnico escolhido TC_i para cada tarefa i e o seu índice ψ_i .
2. *Para cada um dos técnicos:*
 - (a) Faça $f_k = 0$. Ordene as tarefas associadas ao técnico k em ordem crescente do índice ψ . Selecione a primeira tarefa i classificada.
 - (b) Avalie todas as posições possíveis da tarefa i na rota do técnico; escolha a posição que resultar na solução factível com maior valor de f_k^{sp} , dado pela equação (8.7).
 - (c) Se $f_k^{sp} - f_k \geq 0$, inclua a tarefa i na rota do técnico k , faça $f_k = f_k^{sp}$. Caso contrário, a tarefa i não será programada.
 - (d) Se i é a última tarefa do técnico k , siga para o próximo passo. Caso contrário, selecione a próxima tarefa i e volte para o Passo 2b.

8.3.2 Menor tempo de viagem - *STT*

Assim como no decodificador Aleatório, esse decodificador tem $n+1$ chaves aleatórias. Cada uma das n tarefas tem uma chave aleatória e a última será um fator de ponderação entre os objetivos, que permitirá decidir se uma nova tarefa poderá ou não ser incluída

na rota de um técnico. A seguir encontram-se os principais detalhes do decodificador STT.

1. *Ordenar as tarefas:* Ordene as tarefas em ordem decrescente do valor da chave aleatória. Selecione a tarefa i melhor classificada. Para todo técnico k , faça $f_k = 0$.
2. *Análise dos técnicos candidatos:* para cada técnico k capaz ($s_{ik} = 1$), avalie todas as posições possíveis da tarefa i na rota do técnico. Escolha a posição que resultar na solução factível com maior valor de f_k^{sp} , dado pela equação (8.7). Desconsidere os técnicos onde $f_k^{sp} < f_k$.
3. *Escolha do técnico:* se não há técnicos candidatos, siga para o próximo passo. Caso contrário, selecione o técnico que pode executar a tarefa i com menor tempo total de viagem (desde o início até o fim do seu dia de trabalho). Se mais de um técnico tem o mesmo tempo total de viagem, então o segundo e o terceiro critérios de decisão serão a maior ociosidade (z_k) e o menor número de identificação do técnico, respectivamente. Faça essa atribuição. Para o técnico escolhido, faça $f_k = f_k^{sp}$.
4. *Critério de parada:* se a tarefa i é a última classificada no primeiro passo, fim. Caso contrário, selecione a próxima tarefa i e retorne para o Passo 2.

8.3.3 Balanceado

O decodificador Balanceado difere-se do decodificador STT na forma de escolher qual técnico realizará uma tarefa. Em vez de escolher o técnico com menor tempo de viagem, no decodificador Balanceado a chave aleatória $\lambda[n + 1]$ (por meio da soma ponderada) determinará qual técnico será escolhido; ou seja, a tarefa será inserida na rota do técnico que puder executá-la com o maior acréscimo na soma ponderada definida por (8.7) e, portanto, com maior ganho para as funções objetivo do problema.

Concretamente, o pseudocódigo desse decodificador tem os mesmos passos que o pseudocódigo do STT, exceto o Passo 3. Nesse passo será escolhido o técnico k que poderá executar a tarefa i com maior valor de $f_k^{sp} - f_k$. Se dois ou mais técnicos tiverem

o mesmo valor, então o segundo e o terceiro critérios de decisão serão o menor tempo de viagem total e o menor número de identificação do técnico, respectivamente.

Capítulo 9

Resultados Computacionais

Nos experimentos computacionais para o PETC multiobjetivo, foram utilizadas as mesmas instâncias geradas para o problema mono-objetivo (Seção 5.1). Os testes foram realizados em um computador com processador 2.93 GHz Intel(R) 870 e memória RAM de 16GB. Os códigos das heurísticas e meta-heurísticas foram escritos em linguagem C.

A primeira seção deste capítulo apresenta os principais dados das instâncias de dimensões pequenas onde as fronteiras ótimas de Pareto foram obtidas. Em seguida, é feita uma descrição de como os parâmetros do BRKGA multiobjetivo foram calibrados (Seção 9.2) e as estratégias adotadas para formar o conjunto elite são apresentadas (Seção 9.3). A comparação entre os decodificadores adotados e entre o BRKGA e um AG conhecido na literatura encontram-se, respectivamente, nas Seções 9.4 e 9.5. O capítulo é encerrado com as principais conclusões dos experimentos computacionais do PETC multiobjetivo.

9.1 Fronteiras ótimas de Pareto e características das instâncias

Assim como na otimização mono-objetivo (ver Seção 5.3), optou-se por fazer uma enumeração completa para encontrar as soluções que formam o conjunto ótimo de Pareto de algumas instâncias de pequenas dimensões. A Tabela 9.1 apresenta na primeira coluna o número do caso e na segunda, o respectivo número de instâncias em que o conjunto ótimo de Pareto é encontrado; nas colunas seguinte, os números mínimo, médio e máximo de soluções dos conjuntos de Pareto (cardinalidade dos conjuntos); e na última coluna, o tempo médio da enumeração completa em cada caso. No total, foram encon-

Tabela 9.1 – Instâncias com fronteira ótima de Pareto conhecida.

Caso	# instâncias	# soluções fronteira ótima			Tempo médio (segundos)
		Mínimo	Média	Máximo	
1	80	1	12,6	41	5
2	80	2	16,6	51	1.792
3	54	3	24,8	86	42.877
4	20	4	18,4	34	77.347
Média			18,1		30.505

tradas as fronteiras de Pareto de 234 instâncias. Como já se esperava, os tempos médios de execução da enumeração completa têm um crescimento expressivo com o aumento das dimensões das instâncias.

Para exemplificar, é apresentado o conjunto de Pareto ótimo encontrado para uma instância do caso 1. A Tabela 9.2 e a Figura 9.1 apresentam as 15 soluções encontradas. Note que as soluções não formam um conjunto convexo e, por isso, alguns dos métodos tradicionais poderiam não encontrar todas as soluções. De fato, em um teste feito com o método Soma Ponderada, apenas 7 das 15 soluções da fronteira de Pareto foram encontradas; concretamente, as soluções 1, 2, 3, 4, 8, 13 e 15 da Tabela 9.2¹. Já no método Método ε -Restrito, todas as 15 soluções de Pareto foram encontradas. Optou-se por utilizar a enumeração completa tendo em conta o número médio elevado de soluções do conjunto Pareto e as limitações dos métodos tradicionais.

Conforme já citado no início deste capítulo, para os experimentos computacionais no contexto multiobjetivo foram utilizadas as mesmas instâncias geradas na Parte I do trabalho, ou seja, o tamanho das janelas de tempo das tarefas varia entre 2 e 8 horas. Poderia ser argumentado que, para instâncias com janelas de tempo apertadas (2 horas), o segundo objetivo proposto (f_2) perderia o sentido – uma vez que o tempo não utilizado das janelas de tempo ($t_i - e_i - p_i$) é pequeno, entre zero e 90 min – e que se igualaria ao primeiro objetivo. Para estudar essa hipótese, o Apêndice C apresenta um exemplo com 16 tarefas e 2 técnicos, onde todas as tarefas têm janelas de tempo apertadas. A fronteira de Pareto tem 6 soluções e o exemplo descreve a influência das

¹ No estudo do método Soma Ponderada, utilizou-se o fator de ponderação α_1 e α_2 (com $\alpha_2 = 1 - \alpha_1$) das funções objetivo f_1 e f_2 , respectivamente, com precisão de até cinco casas decimais. Desse modo, por exemplo, o método encontrou a solução 15 para todos os valores de α_1 menores ou iguais a 0,21711 e a solução 13 para valores de α_1 entre 0,21712 e 0,40403; ou seja, o método não encontrou a solução 14 e testes com valores intermediários a 0,21711 e 0,21712 também não encontraram essa solução, mostrando a dificuldade do método para encontrar alguns pontos da fronteira de Pareto. Logicamente, para valores de α_1 maiores ou iguais a 0,40404, o método encontrou as soluções 8, 4, 3, 2 e 1.

Tabela 9.2 – Pareto ótimo de uma instância com 16 tarefas e 2 técnicos.

Solução	f_1	f_2
1	53,194	20,795
2	53,192	21,189
3	53,173	23,446
4	53,125	23,611
5	52,291	24,404
6	52,280	24,813
7	51,265	26,076
8	51,223	27,313
9	49,324	27,544
10	49,321	27,841
11	47,302	28,356
12	46,222	29,437
13	46,217	30,705
14	44,318	30,936
15	44,316	31,233

Figura 9.1 – Fronteira ótima de Pareto de uma instância com 16 tarefas e 2 técnicos.

Fonte: elaborado pelo autor.

duas funções objetivo em três das soluções, concluindo que maximizar as folgas não perde o sentido em instâncias com janelas de tempo apertadas. Outra análise que se pode fazer é considerar o número de soluções das fronteiras de Pareto destas instâncias. Para os casos 1, 2, 3 e 4, onde as fronteiras são conhecidas, essa média é de 3,5; 4,0; 4,3 e 12,8 pontos. Essas médias são inferiores à média geral, apresentada na Tabela 9.1, o que indica que a segunda função objetivo tem um impacto menor nessas instância. Por outro lado, o número de pontos da fronteira tende a crescer com o aumento do tamanho das instâncias, o que sinaliza que a segunda função objetivo – mesmo nas instâncias com janelas de tempo apertadas – tem um impacto significativo com o aumento do número de tarefas.

9.2 Calibragem dos parâmetros do BRKGA

Os seguintes parâmetros do BRKGA foram inicialmente calibrados: tamanho da população elite, porcentagem de soluções mutantes, probabilidade do *crossover* parametrizado, tamanho da população e número máximo de gerações. Nessa calibração utilizou-se uma faixa maior para os valores dos parâmetros dos que os utilizados no BRKGA mono-objetivo (ver Seção 5.5.1), pois os sugeridos por Gonçalves e Resende (2011) referem-se apenas à otimização mono-objetivo. Os seguintes valores foram avaliados para cada um dos parâmetros:

- tamanho da população elite (n_e): 15%, 20%, ..., 40% da população.
- novas soluções mutantes (n_m): 5%, 10%, 15% e 20% da população.
- probabilidade do crossover parametrizado (p_e): 50%, 60%, 70% e 80%.
- tamanho da população (n_p): até 1500 cromossomos.

O desempenho de todas as versões do BRKGA multiobjetivo foi avaliado com diferentes combinações de valores dos parâmetros em 104 instâncias (10% das instâncias de cada um dos casos). A Tabela 9.3 apresenta os valores de n_e , n_m e p_e que resultaram nos melhores índices médios (de porcentagem de soluções eficientes (I_{PSE}), hipervolume (I_H) e indicador epsilon multiplicativo (I_ϵ)) para cada BRKGA multiobjetivo. Assim como no caso mono-objetivo, o tamanho da população (n_p) adotado foi de 1000 (para os casos 1 a 11), 500 (caso 12) e 200 cromossomos (caso 13). A título de exemplo, o

Tabela 9.3 – Parâmetros calibrados para cada uma das versões do BRKGA multiobjetivo

	BRKGA-A	BRKGA-STT	BRKGA-Balanceado
n_e	35%	35%	35%
n_m	10%	15%	15%
p_e	70%	60%	60%

Apêndice D apresenta os resultados da calibração do tamanho do conjunto elite para o BRKGA-Balanceado e os critérios de escolha para $n_e = 35\%$.

O número máximo de gerações para cada uma das versões do BRKGA multiobjetivo e o critério de parada adotado estão na Tabela 9.4. Considerou-se que o algoritmo convergiu quando a variação do hipervolume nas últimas gerações é inferior a 0,01%. O critério de parada representa o tempo necessário para todas as versões do BRKGA multiobjetivo atingirem essa convergência. Numa comparação entre o critério de parada do BRKGA mono com o multiobjetivo, o multiobjetivo precisou de em média quatro vezes mais de tempo computacional para atingir a convergência, o que pode ser justificado pelo maior grau de dificuldade de uma meta-heurística em um problema multiobjetivo, onde procura-se encontrar soluções eficientes em boa quantidade e qualidade.

Tabela 9.4 – Número de gerações necessárias para a convergência do BRKGA multiobjetivo e critério de parada (*limit_time*) para cada caso.

Caso	Número de gerações			<i>limit_time</i>
	BRKGA-STT	BRKGA-Cluster	BRKGA-A	segundos
1	400	400	400	10
2	430	430	430	20
3	1240	910	900	50
4	7600	1520	1510	100
5	10340	3180	3180	300
6	8930	2550	2590	400
7	10510	1910	1900	800
8	14030	2550	2580	1300
9	19910	1810	1810	1800
10	21000	2100	2120	3000
11	20900	2090	2060	5000
12	24270	910	920	7200
13	32270	880	880	10800

9.3 Composição do conjunto elite

9.3.1 Classificação das soluções

Conforme apresentado na Seção 8.1, quatro métodos de classificação das soluções foram avaliados no BRKGA multiobjetivo: classificá-las somente pelas fronteiras (denominado F1 nessa seção) e incluir apenas a primeira fronteira no conjunto elite; utilizar a distância de *crowding*; a densidade de vizinhança ou a densidade baseada em células. Na primeira estratégia, o tamanho do conjunto elite é variável, sendo igual ao número de soluções não dominadas de cada população; nas demais, o conjunto elite tem um tamanho fixo, de acordo com a Tabela 9.3. Em todos os casos, se o número de soluções não dominadas superar o valor máximo do conjunto elite (primeira estratégia) ou o tamanho do conjunto elite (demais estratégias), a distância de *crowding* (na estratégia F1) ou os próprios métodos de classificação (demais estratégias) selecionam as melhores soluções da primeira fronteira, que são incluídas no conjunto elite.

A seguir é apresentado o desempenho dessas diferentes estratégias de classificação para cada um dos três decodificadores. O objetivo é avaliar qual dos critérios de classificação apresenta melhor desempenho médio e é escolhido para o BRKGA. Nesta e nas próximas seções as soluções das heurísticas construtivas NT, STT, EST e Cluster (apresentadas no Capítulo 2) foram incluídas na população inicial do BRKGA.

Os métodos de classificação por linhas frontais e distância de *crowding* não têm parâmetros para serem calibrados. Já na densidade de vizinhança e na densidade de células, é necessário determinar, respectivamente, o valor do tamanho da vizinhança (σ) e do tamanho das células. Para calibrar estes parâmetros foram realizados testes em 104 instâncias. O tamanho da vizinhança σ foi avaliado no intervalo entre 0,01 e 0,20 e os resultados indicam que o melhor desempenho médio é obtido quando $\sigma = 0,05 \cdot \left(1 - \frac{\text{time}}{\text{limit_time}}\right)$, ou seja, o tamanho da vizinhança na primeira geração do BRKGA multiobjetivo é igual a 0,05 e esse valor decresce linearmente, sendo na última geração próximo zero. Com relação ao tamanho das células, foram avaliados valores entre 1 e 50 e os melhores resultados foram obtidos quando o tamanho da célula é igual a $\left\lceil \sqrt{n} \cdot \left(1 - \frac{\text{time}}{\text{limit_time}}\right) \right\rceil$, onde n é o número de tarefas da instância; analogamente ao tamanho da vizinhança, o tamanho das células nas primeiras gerações é igual a $\lceil \sqrt{n} \rceil$ e esse valor diminui até chegar a 1 nas últimas gerações.

As Tabelas 9.5, 9.6 e 9.7 apresentam os indicadores porcentagem de soluções efici-

entes (I_{PSE}), porcentagem de hipervolume (I_{PH}) e indicador epsilon multiplicativo (I_ϵ) para as três versões do BRKGA. Para calcular o hipervolume, é necessário definir o ponto de referência. Nesta e nas próximas seções, é adotado como ponto de referência as seguintes coordenadas:

$$\left(\left\lfloor \min_{x \in \mathbf{Ref}} f_1(x) - 1 \right\rfloor ; \left\lfloor \min_{x \in \mathbf{Ref}} f_2(x) - 1 \right\rfloor \right) \quad (9.1)$$

Os resultados das tabelas estão separados entre problemas pequenos (casos 1 a 4), onde os índices foram calculados em relação ao conjunto ótimo de Pareto, e instâncias médias e grandes (casos 5 a 13), índices calculados em relação ao conjunto de referência (soluções não dominadas entre todos os métodos avaliados).

Em média, os índices dos resultados obtidos pela distância de *crowding* tendem a superar os demais no BRKGA-A e BRKGA-STT; apenas no BRKGA-Balanceado o método densidade de vizinhança obteve melhores índices. Tendo em conta esses resultados, optou-se por adotar nas próximas seções o método distância de *crowding* para o BRKGA-A e BRKGA-STT, e o método densidade de vizinhança para o BRKGA-Balanceado.

Note também que o desempenho do método, que inclui no conjunto elite apenas as soluções da primeira fronteira, é em média inferior a todos os outros (apenas no BRKGA-A superou alguns dos outros métodos). Esse resultado reforça a hipótese de que incluir no conjunto elite soluções estratégicas das outras fronteiras tende a melhorar o desempenho do BRKGA.

Tabela 9.5 – Desempenho do **BRKGA-A** para os diferentes métodos de ordenação:
a) instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto.
b) instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.

(a)				
	F1	<i>Crowding</i>	Vizinhança	Células
I_{PSE} (%)	82,6	92,5	61,6	63,4
I_{PH} (%)	99,7	100,0	99,1	99,2
I_{ϵ}	0,995	0,998	0,986	0,988

(b)				
	F1	<i>Crowding</i>	Vizinhança	Células
I_{PSE} (%)	42,9	56,4	4,4	3,7
I_{PH} (%)	92,8	94,4	59,3	59,6
I_{ϵ}	0,988	0,991	0,943	0,944

Tabela 9.6 – Desempenho do **BRKGA-STT** para os diferentes métodos de ordenação:
a) instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto.
b) instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.

(a)				
	F1	<i>Crowding</i>	Vizinhança	Células
I_{PSE} (%)	86,5	93,2	92,9	94,1
I_{PH} (%)	99,8	100,0	100,0	100,0
I_{ϵ}	0,996	0,999	0,999	0,999

(b)				
	F1	<i>Crowding</i>	Vizinhança	Células
I_{PSE} (%)	8,1	59,1	41,0	22,0
I_{PH} (%)	72,6	96,9	94,5	86,6
I_{ϵ}	0,974	0,992	0,991	0,986

Tabela 9.7 – Desempenho do **BRKGA-Balanceado** para os diferentes métodos de ordenação: a) instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.

(a)				
	F1	<i>Crowding</i>	Vizinhança	Células
I_{PSE} (%)	88,8	92,2	95,6	95,4
I_{PH} (%)	99,9	100,0	100,0	100,0
I_ϵ	0,998	0,999	1,000	1,000
(b)				
	F1	<i>Crowding</i>	Vizinhança	Células
I_{PSE} (%)	12,4	44,4	54,2	31,5
I_{PH} (%)	79,2	93,6	97,6	95,4
I_ϵ	0,979	0,989	0,994	0,992

9.3.2 Estratégias adicionais

Após definir o melhor método de classificação para cada versão do BRKGA, foram avaliados diversos modos de formar o conjunto elite do BRKGA multiobjetivo. Tradicionalmente, o BRKGA inclui no conjunto elite os n_e cromossomos melhores classificados (estratégia utilizada nos resultados apresentados até agora). Tendo em conta o bom desempenho da função similaridade no BRKGA mono-objetivo (descrita na Seção 3.3), foi avaliado nesta seção se a função similaridade no BRKGA multiobjetivo melhora ou não os índices de desempenho.

Além disso, tendo em conta a importância dos pontos extremos da fronteira de Pareto (KONAK; COIT; SMITH, 2006), também foi analisada a estratégia de incluir na população inicial do BRKGA multiobjetivo duas soluções do BRKGA mono-objetivo, uma que procura maximizar apenas f_1 e outra, f_2 . Concretamente, dado o mesmo critério de parada (*limit_time*), são comparadas as estratégias de unicamente utilizar o BRKGA multiobjetivo e de utilizar parte do tempo inicial rodando o BRKGA mono-objetivo e no restante do tempo, utilizar o BRKGA multiobjetivo com as soluções do mono-objetivo na população inicial.

Após testes preliminares em 104 instâncias, os seguintes valores para similaridade inicial (DS^0) foram adotados: 55% (BRKGA-A) e 80% (BRKGA-STT e BRKGA-Balanceado). Com relação à estratégia de dedicar parte do tempo de execução para o BRKGA mono-objetivo, esse tempo inicial foi avaliado entre 5% e 40% do tempo total.

Os testes preliminares indicaram os melhores resultados quando os primeiros 20% do tempo são dedicados ao BRKGA mono-objetivo (10% para cada um dos objetivos) e os 80% do tempo restante para o BRKGA multiobjetivo.

A Tabela 9.8 apresenta os índices de desempenho do BRKGA-A com as estratégias multiobjetivo ou mono e multiobjetivo (denominadas nas tabelas *multi* e *mono/multi*, respectivamente), e utilizando a elite tradicional (*trad.*) ou a elite formada pela função similaridade (*similar*). Para os problemas de pequenas dimensões, os resultados foram similares. Já para as instâncias médias e grandes, os índices de avaliação obtidos com a função similaridade superaram de forma significativa os índices dos resultados sem a função similaridade. Esse comportamento também pode ser observado, por exemplo, nas soluções eficientes obtidas para uma instância de 100 tarefas e 10 técnicos (caso 8), apresentado na Figura 9.2. Note que as soluções eficientes obtidas com a elite tradicional (Figuras 9.2(a) e (c)) não atingem nenhum dos pontos da linha de referência (melhores soluções entre os quatro métodos comparados). Já com relação à estratégia mono/multiobjetivo, os índices de desempenho foram um pouco superiores em relação aos obtidos apenas pelo multiobjetivo. Dessa forma, optou-se por associar o decodificador Aleatório à estratégia mono/multiobjetivo com similaridade.

Tabela 9.8 – Desempenho do **BRKGA-A** com distância de *crowding* para diferentes modos de construir o conjunto elite: a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.

(a)				
Estratégia Elite	multi trad.	multi similar.	mono/multi trad.	mono/multi similar.
I_{PSE} (%)	92,5	94,0	94,6	94,6
I_{PH} (%)	100,0	100,0	100,0	100,0
I_{ϵ}	0,998	0,999	0,999	0,999
(b)				
Estratégia Elite	multi trad.	multi similar.	mono/multi trad.	mono/multi similar.
I_{PSE} (%)	16,4	42,9	19,8	49,5
I_{PH} (%)	83,5	94,6	84,9	94,4
I_{ϵ}	0,978	0,988	0,982	0,990

Figura 9.2 – Soluções de referência e soluções eficientes encontradas pelo BRKGA-A com diferentes estratégias: a) somente multiobjetivo com elite tradicional; b) somente multiobjetivo com elite formada pela função similaridade; c) mono e multiobjetivo com elite tradicional; d) mono e multiobjetivo com elite formada pela função similaridade.

Fonte: elaborado pelo autor.

Os resultados dos índices obtidos para o BRKGA-STT encontram-se na Tabela 9.9. Para os problemas de pequenas dimensões, os resultados foram similares. Para as instâncias médias e grandes os índices de avaliação obtidos com a estratégia mono e multiobjetivo apresentaram melhores resultados. A Figura 9.3 mostra um exemplo das soluções obtidas para uma instância de 100 tarefas e 10 técnicos. Note neste exemplo que somente na Figura 9.3(d) os dois pontos extremos são atingidos. No entanto, tendo em conta os índices da Tabela 9.9, optou-se por associar o decodificador STT à estratégia mono/multiobjetivo com a elite tradicional.

Tabela 9.9 – Desempenho do **BRKGA-STT** com distância de *crowding* para diferentes modos de construir o conjunto elite: a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.

(a)				
Estratégia Elite	multi trad.	multi similar.	mono/multi trad.	mono/multi similar.
I_{PSE} (%)	93,2	93,6	95,4	95,3
I_{PH} (%)	100,0	100,0	100,0	100,0
I_ϵ	0,999	0,999	1,000	1,000

(b)				
Estratégia Elite	multi trad.	multi similar.	mono/multi trad.	mono/multi similar.
I_{PSE} (%)	24,3	23,2	45,7	40,1
I_{PH} (%)	93,0	92,9	96,6	96,4
I_ϵ	0,983	0,983	0,993	0,993

Figura 9.3 – Soluções de referência e soluções eficientes encontradas pelo BRKGA-STT com diferentes estratégias: a) somente multiobjetivo com elite tradicional; b) somente multiobjetivo com elite formada pela função similaridade; c) mono e multiobjetivo com elite tradicional; d) mono e multiobjetivo com elite formada pela função similaridade.

Fonte: elaborado pelo autor.

Os resultados dos índices obtidos para o BRKGA-Balanceado encontram-se na Tabela 9.10. Para os problemas de pequenas dimensões, novamente os resultados foram similares. Para as instâncias de dimensões médias e grandes, os índices de avaliação obtidos na estratégia multiobjetivo sem a função similaridade (formando a elite do modo tradicional) apresentaram melhores resultados. A Figura 9.4 mostra um exemplo das soluções obtidas para uma instância de 100 tarefas e 10 técnicos. Note que a Figura 9.4(a) encontrou grande parte dos pontos de referência e, baseado nos índices da Tabela 9.10, optou-se por associar o decodificador Balanceado à estratégia multiobjetivo sem a função similaridade.

Tabela 9.10 – Desempenho do **BRKGA-Balanceado** com densidade de vizinhança para diferentes modos de construir o conjunto elite: a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.

(a)				
Estratégia Elite	multi trad.	multi similar.	mono/multi trad.	mono/multi similar.
I_{PSE} (%)	95,6	94,9	94,9	94,7
I_{PH} (%)	100,0	100,0	100,0	100,0
I_ϵ	1,000	1,000	0,999	0,999

(b)				
Estratégia Elite	multi trad.	multi similar.	mono/multi trad.	mono/multi similar.
I_{PSE} (%)	50,2	48,3	18,7	15,6
I_{PH} (%)	97,2	97,3	91,5	89,8
I_ϵ	0,992	0,991	0,988	0,988

Figura 9.4 – Soluções de referência e soluções eficientes encontradas pelo BRKGA-Balanceado com diferentes estratégias: a) somente multiobjetivo com elite tradicional; b) somente multiobjetivo com elite formada pela função similaridade; c) mono e multi-objetivo com elite tradicional; d) mono e multi-objetivo com elite formada pela função similaridade.

Fonte: elaborado pelo autor.

Em resumo, para as versões BRKGA-A e BRKGA-STT notou-se que a estratégia mono/multi superou em média os índices dos algoritmos que unicamente utilizaram a estratégia multiobjetivo; por outro lado, o BRKGA-Balanceado obteve melhores resultados utilizando apenas o BRKGA multiobjetivo. Esses resultados parecem refletir a diferença entre os três decodificadores. O BRKGA-Balanceado, por explorar igualmente as duas funções objetivo, não necessita ter na sua população inicial boas soluções para cada um deles. Já o decodificador Aleatório, não está baseado nas características das funções objetivo, e o menor tempo de viagem (STT), tende a privilegiar a maximização do número ponderado de tarefas atendidas (objetivo f_1); por esses motivos, ambos se beneficiam quando na população inicial há boas soluções para cada um dos objetivos isolados.

Com relação ao uso da função similaridade, o BRKGA-A (que nos problemas mono-objetivo tinha maior tendência à convergência prematura) obteve melhores índices utilizando essa função, como pode ser observado na Tabela 9.8b. Já nas demais versões do BRKGA, obteve-se resultados semelhantes com e sem o uso da função similaridade (como pode ser visto nas Tabelas 9.9b e 9.10b).

9.4 Comparação dos decodificadores

Após definir na seção anterior o método de classificação e a estratégia para formar o conjunto elite utilizando ou não a função similaridade e incluindo ou não os resultados do BRKGA mono-objetivo na população inicial do multiobjetivo, nessa seção são comparados os resultados das três versões do BRKGA.

A Tabela 9.11 apresenta essa comparação. Nas instâncias pequenas (casos 1 a 4), o BRKGA-STT e BRKGA-Balanceado obtiveram os melhores resultados para a porcentagem de soluções ótimas de Pareto (I_{PSE}) encontradas; note que, como o número médio de soluções nas fronteiras de Pareto é 18,1, essas duas versões estão encontrando em média 17,3 das soluções ótimas de Pareto. Nos demais índices, o desempenho das três versões é praticamente igual para as instâncias pequenas. Já para as instâncias médias e grandes (casos 5 a 13), o BRKGA-Balanceado superou os demais em todos os índices. Para melhor analisar os casos médios e grandes, a Tabela 9.12 apresenta essas medidas para os casos 5 a 7, 8 a 10, e 11 a 13; pode-se concluir que – segundo a medida de hipervolume e indicador epsilon – o desempenho do BRKGA-Balanceado tem uma

Tabela 9.11 – Desempenho das três versões do BRKGA: a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.

(a)			
	BRKGA-A	BRKGA-STT	BRKGA-Balanceado
I_{PSE} (%)	94,6	95,4	95,6
I_{PH} (%)	100,0	100,0	100,0
I_ϵ	0,999	1,000	1,000

(b)			
	BRKGA-A	BRKGA-STT	BRKGA-Balanceado
I_{PSE} (%)	22,0	47,9	49,4
I_{PH} (%)	81,4	90,9	95,6
I_ϵ	0,978	0,969	0,991

maior superioridade em relação às outras versões à medida que aumenta o tamanho das instâncias.

Com relação à “Medida do conjunto de cobertura”, obteve-se os seguintes resultados: $Cov(\text{BRKGA-A}, \text{BRKGA-STT}) = 0,46$ e $Cov(\text{BRKGA-STT}, \text{BRKGA-A}) = 0,71$;

Tabela 9.12 – Desempenho das três versões do BRKGA em relação ao conjunto de referência: a) Casos 5 a 7. b) Casos 8 a 10. c) Casos 11 a 13.

(a)			
	BRKGA-A	BRKGA-STT	BRKGA-Balanceado
I_{PSE} (%)	40,8	52,0	65,0
I_{PH} (%)	95,1	97,1	98,9
I_ϵ	0,988	0,990	0,994

(b)			
	BRKGA-A	BRKGA-STT	BRKGA-Balanceado
I_{PSE} (%)	13,6	42,2	44,2
I_{PH} (%)	82,8	93,5	95,8
I_ϵ	0,980	0,980	0,991

(c)			
	BRKGA-A	BRKGA-STT	BRKGA-Balanceado
I_{PSE} (%)	11,5	49,5	39,0
I_{PH} (%)	66,3	82,0	92,2
I_ϵ	0,965	0,938	0,988

$Cov(\text{BRKGA-A}, \text{BRKGA-Balanceado}) = 0,43$ e $Cov(\text{BRKGA-Balanceado}, \text{BRKGA-A}) = 0,87$; o que demonstra o desempenho superior do BRKGA-STT e do BRKGA-Balanceado em relação ao BRKGA-A. Entre o BRKGA-STT e o BRKGA-Balanceado, a “Medida do conjunto de cobertura” indica um desempenho semelhante dessas duas versões, pois $Cov(\text{BRKGA-STT}, \text{BRKGA-Balanceado}) = Cov(\text{BRKGA-Balanceado}, \text{BRKGA-STT}) = 0,60$.

A Figura 9.5 apresenta um exemplo das soluções eficientes obtidas para cada uma das versões do BRKGA para uma instância com 100 tarefas e 10 técnicos (trata-se da mesma instância apresentada como exemplo na seção anterior). Note que o BRKGA-A atingiu poucos pontos do conjunto Referência, que os pontos extremos foram obtidos pelo BRKGA-STT (para f_1) e BRKGA-Balanceado (para f_2) e que, pela visualização dos gráficos, é difícil comparar os desempenhos do BRKGA-Balanceado e do BRKGA-STT.

Dos índices analisados, apenas o Hipervolume tem um parâmetro a ser definido, que é o ponto de referência. Nos testes realizados, esse ponto foi definido por (9.1). A seguir, encontram-se três outros pontos de referência, utilizados para a análise de sensibilidade deste parâmetro.

$$\left(\left\lfloor \min_{x \in \mathbf{Ref}} 0,99 \cdot f_1(x) \right\rfloor ; \left\lfloor \min_{x \in \mathbf{Ref}} 0,99 \cdot f_2(x) \right\rfloor \right) \quad (9.2)$$

$$\left(\left\lfloor \min_{x \in \mathbf{Ref}} 0,95 \cdot f_1(x) \right\rfloor ; \left\lfloor \min_{x \in \mathbf{Ref}} 0,95 \cdot f_2(x) \right\rfloor \right) \quad (9.3)$$

$$(0; 0) \quad (9.4)$$

A Tabela 9.13 apresenta a análise de sensibilidade, que confirmam a superioridade do BRKGA-Balanceado.

Tabela 9.13 – Análise de sensibilidade do Hipervolume

ponto de referência definido por	BRKGA-A	BRKGA-STT	BRKGA-Balanceado
(9.1)	79,1	90,5	95,6
(9.2)	81,4	90,6	95,9
(9.3)	87,3	91,4	96,8
(9.4)	98,4	97,4	99,2

Figura 9.5 – Soluções de referência e soluções eficientes encontradas pelo: a) BRKGA-A; b) BRKGA-STT; c) BRKGA-Balanceado.

Fonte: elaborado pelo autor.

9.5 Comparação com um AG da literatura

Proposto por Deb et al. (2002), o *Nondominated Sorting Genetic Algorithm II* (NSGA II) é um AG multiobjetivo muito empregado na literatura (AQUINO; ARROYO, 2015). Inicialmente o NSGA-II cria uma população inicial P_0 de N cromossomos e, através dos operadores (seleção, recombinação e mutação), é gerada uma nova população de filhos Q_0 também de tamanho N . A união dos pais com os filhos resulta em um conjunto R_0 , de tamanho $2N$ e todos os indivíduos são avaliados e classificados de acordo com as linhas frontais e a distância de *crowding* (conforme descrito na Seção 8.1.2). Os N melhores cromossomos de R_0 formam a nova população P_1 . Os novos pais P_1 geram os novos filhos Q_1 , que são classificados para gerar os pais da segunda geração P_2 , e assim por diante.

A seguir encontra-se a descrição dos operadores do NSGA-II. Para formar novas soluções, dois cromossomos são selecionados através de sorteios binários ou *binary tournament selection* (GOLDBERG; DEB, 1999): com igual probabilidade para todos os N cromossomos pais, dois sorteios de dois cromossomos são realizados e os cromossomos melhores classificados de cada sorteio (pelas linhas frontais e pela distância de *crowding*) são escolhidos para gerar dois novos cromossomos. O cruzamento é feito pelo *crossover* em dois pontos (troca dos genes localizados entre os dois pontos sorteados) e em seguida os genes podem ser alterados aleatoriamente com uma probabilidade de mutação p_{mut} .

Para comparar apenas as diferenças entre as estratégias do BRKGA-Balanceado e do NSGA-II (elitismo, seleção, *crossover* e mutação), optou-se por utilizar a mesma representação do cromossomos para o PETC, ou seja, o NSGA-II também foi construído com cromossomos de $n + 1$ números aleatórios entre 0 e 1, que são transformados em soluções do PETC através do decodificador Balanceado. Para calibrar os parâmetros do NSGA-II, o seu desempenho foi avaliado em 104 instâncias com diferentes tamanhos de população ($N = 250, 500, 750, 1000$) e diferentes probabilidade de mutação (p_{mut} entre 0,05% e 10,0%). Os melhores resultados médios foram obtidos com $p_{mut} = 2\%$ e com tamanho da população N de 750 (para os casos 1 a 11), 500 (caso 12) e 200 cromossomos (caso 13).

A Tabela 9.14 apresenta os índices de desempenho do BRKGA-Balanceado e o NSGA-II, em testes computacionais em 1040 instâncias. Para as instâncias pequenas, os dois algoritmos têm um desempenho semelhante. Já para instâncias médias e grandes

Tabela 9.14 – Desempenho do BRKGA e do NSGA-II: a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto. b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.

(a)		
	BRKGA-Balanceado	NSGA-II
I_{PSE} (%)	93,9	94,2
I_{PH} (%)	100,0	100,0
I_{ϵ}	1,000	0,999
(b)		
	BRKGA-Balanceado	NSGA-II
I_{PSE} (%)	66,0	43,7
I_{PH} (%)	97,0	89,0
I_{ϵ}	0,996	0,992

(casos 5 a 13) o BRKGA-Balanceado obteve um desempenho significativamente superior². Com relação à “Medida do conjunto de cobertura”, obteve-se os seguintes resultados: $Cov(\text{BRKGA-Balanceado}, \text{NSGA-II}) = 0,89$ e $Cov(\text{NSGA-II}, \text{BRKGA-Balanceado}) = 0,47$.

A análise estatística dos índices de desempenho confirmou que os índices médios do BRKGA-Balanceado são maiores do que os do NSGA-II nas instâncias médias e grandes, o que permite concluir que de fato o BRKGA-Balanceado tem um desempenho superior. Para realizar essa análise, inicialmente o teste de Kolmogorov-Smirnov foi aplicado e a hipótese de normalidade da distribuição dos resultados dos índices foi rejeitada com nível de significância $\alpha = 1\%$. Em decorrência deste resultado utilizou-se o teste não-paramétrico de soma de classificação de Wilcoxon. Para as instâncias pequenas, a hipótese nula não foi rejeitada e, portanto, não há uma diferença entre as médias apresentadas na Tabela 9.14(a). Já com relação às instâncias médias e grandes, a hipótese nula foi rejeitada com nível de significância $\alpha = 1\%$ (o valor-P resultou em 0,00 para todos os índices) e conclui-se que a média dos índices do BRKGA-Balanceado são superiores.

Para exemplificar a diferença de desempenho entre os dois métodos, a Figura 9.6 apresenta os resultados obtidos novamente para uma instância com 100 tarefas e 10

² No cálculo do Hipervolume, novamente foi feita uma análise de sensibilidade dessa medida em relação ao ponto de referência adotado. Além do ponto definido por (9.1), outros três foram utilizados: (9.2), (9.3) e (9.4). Os resultados obtidos para todos os pontos de referência confirmaram a superioridade do BRKGA em relação ao NSGA-II.

Tabela 9.15 – Desempenho do BRKGA e do NSGA-II, ambos com distância de *crowding*:
a) índices das instâncias pequenas (casos 1 a 4) em relação ao conjunto ótimo de Pareto.
b) índices das instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.

(a)			
Meta-heurística Classificação	BRKGA-Balanceado Vizinhança	BRKGA-Balanceado <i>Crowding</i>	NSGA-II <i>Crowding</i>
I_{PSE} (%)	95,6	92,2	95,0
I_{PH} (%)	100,0	100,0	100,0
I_ϵ	1,000	0,999	0,999

(b)			
Meta-heurística Classificação	BRKGA-Balanceado Vizinhança	BRKGA-Balanceado <i>Crowding</i>	NSGA-II <i>Crowding</i>
I_{PSE} (%)	56,1	47,8	21,2
I_{PH} (%)	97,7	94,9	85,5
I_ϵ	0,994	0,991	0,987

técnicos. Note que os resultados obtidos pelo BRKGA superam nitidamente os do NSGA-II, que obteve poucos pontos do conjunto Referência.

O BRKGA e o NSGA-II diferenciam-se em diversos aspectos: na estratégia de elitismo, na forma de classificar as soluções, de fazer o *crossover* e na diversificação (soluções mutantes ou mutações, respectivamente). Se por um lado o NSGA-II é mais simples de ser implementado, por outro ele parece ter uma menor capacidade de diversificação do que o BRKGA, que é promovida pelas soluções mutantes e pela classificação por vizinhança na versão BRKGA-Balanceado. Para aprofundar na comparação entre o BRKGA-Balanceado (com vizinhança) e o NSGA-II (com distância de *crowding*), esses dois métodos foram comparados com o BRKGA-Balanceado (com distância de *crowding*) e obteve-se os índices apresentados na Tabela 9.15³. Para os problemas pequenos, os métodos obtiveram índices semelhantes; já para as instâncias de médias e grandes dimensões, o desempenho do BRKGA-Balanceado (com distância de *crowding*) foi superior ao do NSGA-II, o que indica que as estratégias de elitismo, *crossover* e diversificação são, provavelmente, mais responsáveis pelo desempenho superior do BRKGA-Balanceado do que o método de classificação das soluções.

³ A Tabela 9.15 apresenta a média dos índices dos resultados obtidos em 104 instâncias, diferentemente da Tabela 9.14, que refere-se a 1040 instâncias.

Figura 9.6 – Soluções de referência e soluções eficientes encontradas por: a) BRKGA-Balanceado; b) NSGA-II.

(a) BRKGA-Balanceado

(b) NSGA-II

Fonte: elaborado pelo autor.

9.6 Conclusões do capítulo

Neste capítulo diversas versões multiobjetivo do BRKGA foram desenvolvidas, com diferentes estratégias para classificar as soluções e formar o conjunto elite. Pôde-se verificar o desempenho dessas estratégias dependem principalmente da qualidade do decodificador. Os melhores resultados foram obtidos pelo decodificador Balanceado, que explora igualmente as duas funções objetivo. Quando isso não acontece, pode ser vantajoso utilizar parte do tempo da busca para encontrar boas soluções para cada um dos objetivos isolados e incluí-las na população inicial do algoritmo multiobjetivo. Já a função similaridade mostrou-se vantajosa apenas quando o decodificador tem maior tendência à convergência prematura, como é o caso do decodificador Aleatório.

Na comparação entre o BRKGA e o NSGA-II, o BRKGA obteve resultados superiores em todos os índices de comparação, como pode ser observado na Tabela 9.14 e confirmado pelos testes estatísticos.

Capítulo 10

Conclusões e futuros trabalhos

Este trabalho apresentou um estudo mono e biobjetivo para o Problema de Escalonamento de Técnicos de Campo.

Na primeira parte, heurísticas construtivas e alguns Algoritmos Genéticos com Chaves Aleatórias Viesadas (*Biased Random Key Genetic Algorithm* – BRKGA) foram apresentados. As melhores heurísticas construtivas geraram bons resultados com um esforço computacional reduzido para instâncias de grandes dimensões, que ficaram em média a 3,8% (500 tarefas) e 2,9% (999 tarefas) dos limitantes superiores. A melhor versão do BRKGA encontrou 99% das soluções ótimas dos 238 problemas de pequenas dimensões e, para os 720 problemas de médias e grandes dimensões, os resultados ficaram em média a 3,8% dos limitantes superiores. Os decodificadores mais elaborados obtiveram um desempenho melhor do que os decodificadores mais aleatórios.

Para evitar a convergência prematura do BRKGA e melhorar seus resultados, foi proposta uma função similaridade para diversificar a população elite do algoritmo, selecionando cromossomos com uma porcentagem de genes diferentes. Os testes computacionais indicaram que, com a utilização da função similaridade, o algoritmo apresentou um desempenho médio superior de até 2% e com resultados estritamente melhores em mais de 70% das instâncias.

A segunda parte do trabalho apresentou uma abordagem biobjetivo para o problema, onde uma segunda função objetivo buscou que as tarefas mais prioritárias fossem realizadas o mais cedo possível. Diversas estratégias para classificar a população do algoritmo e escolher as melhores soluções (estratégias de elitismo) do BRKGA foram propostas e avaliadas. A melhor versão do algoritmo foi capaz de encontrar 94% das soluções da fronteira ótima de Pareto (entre 234 instâncias pequenas) e, para instâncias médias e

grandes, superar o desempenho do NSGA-II nos índices de avaliação adotados.

A seguir, apresentam-se algumas propostas para possíveis futuros trabalhos:

1. Aprofundar no estudo sobre a modelagem do problema com o intuito de investigar formulações mais próximas da sua envoltória convexa.
2. Investigar outros objetivos para o problema, como por exemplo o balanceamento do trabalho dos técnicos.
3. Inserir novas conceitos no modelo, como a realização de tarefas com adiantamento ou atraso (penalizando essa violação das janelas de tempo) ou incluir na função objetivo o custo da não execução de uma tarefa.
4. Diversos decodificadores foram propostos para o BRKGA. Algumas novas estratégias de codificadores poderiam ser desenvolvidas, com outras representações do cromossomo e com buscas locais, tanto para o enfoque mono quanto multiobjetivo.
5. Comparar os resultados obtidos pelo BRKGA e pelo NSGA-II com outros Algoritmos Genéticos multiobjetivos, como o *Strength Pareto Evolutionary Algorithm 2* (SPEA2) ou o *Indicator-Based Evolutionary Algorithm* (IBEA), e outras metas-heurísticas multiobjetivo, como *Ant Colony* ou *Artificial Bee Colony*.
6. Adaptar os decodificadores desenvolvidos neste trabalho para problemas semelhantes, como os problemas de times de técnicos para realizar as tarefas (KOVACS et al., 2012; CORDEAU et al., 2010).
7. Fazer uma comparação entre a função similaridade e diferentes estratégias da literatura para gerar diversidade na população, avaliando quão efetivas e diferentes elas são.
8. Neste trabalho não foram adotadas estratégias computacionais de processamento paralelo. Para futuros trabalho, essas técnicas poderiam ser desenvolvidas e o BRKGA proposto eventualmente adaptado ou otimizado.

Parte III

Apêndices

Apêndice A

Relação do PETC com outros problemas NP-difíceis

A.1 Extensão do problema de Roteamento de Veículo com Janelas de Tempo

O PRVJT pode ser representado pelo modelo PLIM a seguir (SOLOMON, 1987). Será adotada a notação utilizada por Arenales, Armentano e Morabito (2007), com uma barra superior em cima dos parâmetros e variáveis deste modelo para distingui-los do PLIM do PETC.

Parâmetros:

$\bar{C} = \{1, \dots, \bar{n}\}$	Conjunto de nós que representam os $\bar{n}$ clientes;
$\bar{N} = \bar{C} \cup \{0, \bar{n} + 1\}$	Conjunto de todos os nós, onde os nós 0 e $\bar{n} + 1$ representam a base;
$\bar{K}$	Conjunto de veículos, $k = 1, \dots, \bar{m}$;
$\bar{Q}_k$	Capacidade do veículo $k \in \bar{K}$;
$\bar{E}$	Conjunto de arcos associados às conexões entre os nós: $\bar{E} = \{(i, j) : i, j \in \bar{N}, i \neq j, i \neq \bar{n} + 1, j \neq 0\}$;
$\bar{\mu}_{ij}$	Custo associado ao arco (i, j) ;
$\bar{\nu}_{ij}$	Tempo para o arco (i, j) ser percorrido;
$\bar{p}_i$	Tempo de serviço do cliente i ;
$\bar{d}_i$	Demanda do cliente i ;
$\bar{e}_i$	Início da janela de tempo do cliente i ;
$\bar{l}_i$	Fim da janela de tempo do cliente i .

Variáveis:

- $\bar{x}_{ijk}$ Assume 1 se o veículo $\bar{k}$ percorre o arco (i,j) , $\forall k \in \bar{K}, \forall (i,j) \in \bar{E}$, e 0, caso contrário;
- $\bar{t}_i$ Instante em que o cliente i começa a ser servido.

Formulação do PRVJT:

$$\min \sum_{k \in \bar{K}} \sum_{(i,j) \in \bar{E}} \bar{\mu}_{ij} \bar{x}_{ijk} \quad (\text{A.1})$$

Sujeito a:

$$\sum_{k \in \bar{K}} \sum_{j \in \bar{N}} \bar{x}_{ijk} = 1 \quad i \in \bar{C} \quad (\text{A.2})$$

$$\sum_{i \in \bar{N}} \bar{x}_{ihk} = \sum_{j \in \bar{N}} \bar{x}_{hjk} \quad h \in \bar{C}, k \in \bar{K} \quad (\text{A.3})$$

$$\sum_{j \in \bar{N}} \bar{x}_{0jk} = 1 \quad k \in \bar{K} \quad (\text{A.4})$$

$$\sum_{i \in \bar{N}} \bar{x}_{i,\bar{n}+1,k} = 1 \quad k \in \bar{K} \quad (\text{A.5})$$

$$\sum_{i \in \bar{C}} \bar{d}_i \sum_{j \in \bar{N}} \bar{x}_{ijk} \leq \bar{Q}_k \quad k \in \bar{K} \quad (\text{A.6})$$

$$\bar{e}_i \leq \bar{t}_i \leq \bar{l}_i \quad i \in \bar{C} \quad (\text{A.7})$$

$$\bar{t}_i + \bar{p}_i + \bar{\nu}_{ij} \leq \bar{t}_j + M(1 - \bar{x}_{ijk}) \quad k \in \bar{K}, (i,j) \in \bar{E} \quad (\text{A.8})$$

$$\bar{x}_{ijk} \in \{0, 1\} \quad i, j \in \bar{C}, k \in \bar{K} \quad (\text{A.9})$$

$$\bar{t}_i \in \mathbf{R}_+ \quad i \in \bar{C} \quad (\text{A.10})$$

Onde:

$$M \text{ é um número suficientemente grande} \quad (\text{A.11})$$

A função objetivo (A.1) minimiza o custo total das rotas percorridas. As restrições (A.2) asseguram que cada cliente é atendido por exatamente um veículo. As restrições (A.3) a (A.5) representam o fluxo de rede: cada veículo deverá deixar a base uma vez, sairá do cliente i somente se tiver entrado neste mesmo cliente e retornará ao depósito também uma única vez. A capacidade $\bar{Q}_k$ de cada veículo k não pode ser excedida, como garante (A.6). As restrições (A.7) garantem que as janelas de tempo serão respeitadas.

As restrições (A.8) significam que o cliente j será atendido após o veículo k terminar o atendimento do cliente i e percorrer o arco (i, j) .

Para facilitar as relações entre os dois problemas, considere o PETC a seguir (que será denominado PETC-2), onde a função objetivo será minimizar o custo de deslocamento total: μ_{ij} será o custo para um técnico sair da tarefa i e ir para a tarefa j . Os demais parâmetros e variáveis do PETC não foram alterados, com exceção de y_{ik} que foi substituído por $\sum_{j \in J \setminus i} x_{jik}$, mantendo o mesmo significado do PETC. Será exigido que todos os técnicos deixem a base e que todas as tarefas sejam atendidas; portanto, as restrições (A.13) e (A.15) são de igualdade.

Formulação do PETC-2:

$$\min \sum_{k \in K} \sum_{i \in J} \sum_{j \neq i \in J} \mu_{ij} x_{ijk} \quad (\text{A.12})$$

Sujeito a:

$$\sum_{k \in K} \sum_{j \neq i \in J} x_{ijk} = 1 \quad i \in J \quad (\text{A.13})$$

$$\sum_{j \in J \cup \{0\} \setminus i} x_{ijk} = \sum_{j \in J \cup \{0\} \setminus i} x_{jik} \quad k \in K, i \in J \quad (\text{A.14})$$

$$\sum_{i \in J} x_{0ik} = 1 \quad k \in K \quad (\text{A.15})$$

$$e_i \leq t_i \leq l_i - p_i \sum_{k \in K} \sum_{j \neq i \in J} x_{ijk} \quad i \in J \quad (\text{A.16})$$

$$t_i + p_i + c_{ij} \leq t_j + M(1 - x_{ijk}) \quad k \in K, i \neq j \in J \quad (\text{A.17})$$

$$a_k + c_{0j} \leq t_j + M(1 - x_{0jk}) \quad k \in K, j \in J \quad (\text{A.18})$$

$$t_i + p_i + c_{i0} \leq b_k + M(1 - x_{i0k}) \quad k \in K, i \in J \quad (\text{A.19})$$

$$x_{ijk} \in \{0, 1\} \quad i, j \in J, k \in K \quad (\text{A.20})$$

$$t_i \in \mathbf{R}_+ \quad i \in J, k \in K \quad (\text{A.21})$$

Pode-se notar que os modelos do PRVJT e do PETC-2 têm diversas restrições equivalentes:

- A restrição (A.2), que associa cada cliente a um único veículo, é idêntica à (A.13), que destina cada tarefa a um único técnico.

- O fluxo de rede é garantido pelas equações (A.3) a (A.5) no PRVJT. Segundo Arenales, Armentano e Morabito (2007), as restrições (A.5) são redundantes e estão apresentadas no modelo somente para enfatizar a estrutura de rede. Dessa forma, as restrições (A.14) e (A.15) garantem o fluxo de rede no PETC-2.
- As janelas de tempo dos clientes no PRVJT e das tarefas no PETC-2 precisam ser respeitadas, o que é feito pelas restrições (A.7) e (A.16). Repare que, na formulação apresentada, as janelas de tempo do PRVJT indicam apenas o intervalo em que o cliente precisa começar a ser atendido e no PETC-2, o intervalo em que a tarefa deve começar e terminar de ser executada: portanto, é imediato que as duas restrições são equivalentes.
- As restrições (A.8) do PRVJT associa o instante de início do atendimento de um cliente ao atendimento do cliente anterior na rota de um técnico. Essa mesma função é realizada no PETC-2 pela restrição (A.17), onde o início da execução de uma tarefa depende de sua antecessora.
- As restrições das variáveis (A.9) e (A.10) no PRVJT e (A.20) e (A.21) no PETC-2 também são idênticas.
- Falta apenas relacionar as restrições (A.6), que garante que a capacidade de um veículo será respeitada, com as restrições (A.17) a (A.19), o que será feito a seguir.

Pela analogia dos dois problemas, as restrições que garantem que a capacidade de cada veículo não é excedida devem ser equivalentes a que as tarefas realizadas pelos técnicos não superem o seu tempo de trabalho ou, em outras palavras, a janela de tempo de cada técnico deve ser respeitada. Pretende-se mostrar que é possível relaxar as restrições (A.17) a (A.19) para encontrar uma expressão equivalente a (A.6).

As restrições (A.17) a (A.19) podem ser re-escritas da seguinte maneira:

$$t_i + p_i + c_{ij} \leq t_j + M(1 - x_{ijk}) \iff x_{ijk}(t_i + p_i + c_{ij} - t_j) \leq 0, k \in K, i \neq j \in J \quad (\text{A.22})$$

$$a_k + c_{0j} \leq t_j + M(1 - x_{0jk}) \iff x_{0jk}(a_k + c_{0j} - t_j) \leq 0, k \in K, j \in J \quad (\text{A.23})$$

$$t_i + p_i + c_{i0} \leq b_k + M(1 - x_{i0k}) \iff x_{i0k}(t_i + p_i + c_{i0} - b_k) \leq 0, k \in K, i \in J \quad (\text{A.24})$$

Somando as essas restrições re-escritas, para cada técnico teremos que:

$$\sum_{j \in J} x_{0jk}(a_k + c_{0j} - t_j) + \sum_{i \in J} \sum_{j \in J} x_{ijk}(t_i + p_i + c_{ij} - t_j) + \sum_{i \in J} x_{i0k}(t_i + p_i + c_{i0} - b_k) \leq 0, k \in K \quad (\text{A.25})$$

Na equação (A.25), a primeira parte está relacionada com a primeira tarefa do técnico k ; a segunda parte consiste no somatório que envolve a primeira e a segunda tarefas, a segunda e a terceira tarefas, e assim sucessivamente, até a penúltima e a última tarefa; a terceira parte representa a última tarefa do técnico. Ou seja, há uma soma e uma subtração de todas as variáveis t que, portanto, podem ser retiradas da expressão, ficando a restrição (A.25) equivalente a:

$$\sum_{j \in J} x_{0jk}(a_k + c_{0j}) + \sum_{i \in J} \sum_{j \in J} x_{ijk}(p_i + c_{ij}) + \sum_{i \in J} x_{i0k}(p_i + c_{i0} - b_k) \leq 0, k \in K \quad (\text{A.26})$$

Considerando o caso em que todas as tarefas e a base são equidistantes (iguais a $\hat{c}$), seja d_i a soma entre o tempo de processamento de uma tarefa e a sua distância às outras ($d_i = p_i + \hat{c}$). A restrição (A.26) pode ser re-escrita da seguinte forma:

$$\sum_{j \in J} x_{0jk}(a_k + \hat{c}) + \sum_{i \in J} \sum_{j \in J} x_{ijk}d_i + \sum_{i \in J} x_{i0k}(d_i - b_k) \leq 0, k \in K \quad (\text{A.27})$$

Como a restrição (A.15) garante que os técnicos deixam a base exatamente uma vez e, portanto, retornarão apenas uma vez, podemos reescrever a expressão (A.27):

$$\sum_{i \in J} d_i \sum_{j \in J \cup \{0\}} x_{ijk} \leq b_k - a_k - \hat{c}, k \in K \quad (\text{A.28})$$

Seja o valor $b_k - a_k - \hat{c} = \hat{Q}_k$, então:

$$\sum_{i \in J} d_i \sum_{j \in J \cup \{0\}} x_{ijk} \leq \hat{Q}_k, k \in K \quad (\text{A.29})$$

A equação (A.29) é equivalente a (A.6) e, portanto, o modelo relaxado de PETC-2 é equivalente ao modelo do PRVJT.

Para finalizar essa seção, sobre a relação entre o PETC e o PRVJT, falta unicamente fazer a ligação entre o PETC-2 e o modelo original (PETC). A principal diferença é que - pela limitação do número de técnicos - no PETC não será possível atender todas as

tarefas, gerando duas consequências: as restrições (A.13) e (A.15) do PETC-2 passarão a ser desigualdades (≤ 1) e a função objetivo maximizará o número ponderado de tarefas atendidas. Para minimizar as distâncias percorridas pelos técnicos, um objetivo secundário foi incluído na função objetivo do PETC, maximizando a ociosidade dos técnicos no fim do dia de trabalho.

Dadas essas alterações na função objetivo e em algumas restrições, pode-se mencionar que o PETC é um problema semelhante (e não propriamente equivalente) ao PETC-2 e ao PRVJT.

A.2 Redução ao problema de máquinas paralelas

Neste apêndice apresentaremos uma redução do problema de máquinas paralelas ao PETC. Kahlbacher e Cheng (1993) provaram que o problema de minimizar o número de tarefas atrasadas em máquinas paralelas idênticas (PMPI) é NP-difícil.

Deseja-se mostrar que uma instância desse problema pode ser convertida em tempo polinomial a uma instância do PETC e, portanto, o problema de máquinas paralelas que minimiza o número de tarefas atrasadas é reduzível ao PETC (GAREY; JOHNSON, 1979).

1. Inicialmente, vamos mostrar que o problema de minimizar o número de tarefas não realizadas em máquinas paralelas idênticas (que será chamado de PMPI-2) é reduzível ao problema de minimizar o número ponderado de tarefas não realizadas em máquinas paralelas idênticas com tempo dependente do *setup* e outros parâmetros: janelas de tempo para as tarefas e para as máquinas, prioridades das tarefas e se as máquinas podem ou não processar cada tarefa. Chamaremos esse problema de PMPIS.

Será utilizada uma barra superior em cima dos parâmetros e das variáveis do modelo de máquinas paralelas para distingui-los do PLIM do PETC.

Parâmetros:

- $\bar{J}$ Conjunto de tarefas, $i = 1, \dots, \bar{n}$;
- $\bar{K}$ Conjunto de máquinas, $k = 1, \dots, \bar{m}$;
- $\bar{p}_i$ Tempo de processamento da tarefa i ;
- $\bar{e}_i$ Horário de chegada da tarefa i ;
- $\bar{l}_i$ Horário de saída da tarefa i ;
- $\bar{a}_k$ Horário de início de funcionamento da máquina k ;
- $\bar{b}_k$ Horário de término de funcionamento da máquina k ;
- $\bar{c}_{ij}$ Tempo de *setup* entre as tarefas i e j , onde $i, j \in \bar{J} \cup \{0\}$,
e 0 é uma tarefa fictícia que representa o *setup* do início e fim do dia;
- $\bar{w}_i$ Prioridade ou peso da tarefa i ;
- $\bar{s}_{ik}$ Indica se a máquina k pode realizar a tarefa i .

Variáveis:

- $\bar{y}_{ik}$ Assume 1 se a tarefa i é realizada pela máquina k e 0, caso contrário;
- $\bar{x}_{ijk}$ Assume 1 se a tarefa i precede a tarefa j na máquina k e 0, caso contrário;
- $\bar{t}_i$ Início da execução da tarefa i ;
- $\bar{z}_k$ Tempo disponível da máquina k no fim do dia.

Formulação do PMPIS:

$$\min \sum_{i \in J} \frac{\bar{w}_i(1 - \sum_{k \in K} \bar{y}_{ik})}{MW} - \sum_{k \in K} \frac{\bar{z}_k}{MZ} \quad (\text{A.30})$$

Sujeito a:

$$\sum_{k \in K} \bar{y}_{ik} \leq 1 \quad i \in \bar{J} \quad (\text{A.31})$$

$$\sum_{j \in J \cup \{0\} \setminus i} \bar{x}_{ijk} = \sum_{j \in J \cup \{0\} \setminus i} \bar{x}_{jik} = \bar{y}_{ik} \quad k \in \bar{K}, i \in \bar{J} \quad (\text{A.32})$$

$$\sum_{i \in J} \bar{x}_{0ik} \leq 1 \quad k \in \bar{K} \quad (\text{A.33})$$

$$\bar{e}_i \leq \bar{t}_i \leq \bar{l}_i - \bar{p}_i \sum_{k \in K} \bar{y}_{ik} \quad i \in \bar{J} \quad (\text{A.34})$$

$$\bar{t}_i + \bar{p}_i + \bar{c}_{ij} \leq \bar{t}_j + M(1 - \bar{x}_{ijk}) \quad k \in \bar{K}, i \neq j \in \bar{J} \quad (\text{A.35})$$

$$\bar{a}_k + \bar{c}_{0j} \leq \bar{t}_j + M(1 - \bar{x}_{0jk}) \quad k \in \bar{K}, j \in \bar{J} \quad (\text{A.36})$$

$$\bar{t}_i + \bar{p}_i + \bar{c}_{i0} \leq \bar{b}_k - \bar{z}_k + M(1 - \bar{x}_{i0k}) \quad k \in \bar{K}, i \in \bar{J} \quad (\text{A.37})$$

$$\bar{y}_{ik} \leq \bar{s}_{ik} \quad (\text{A.38})$$

$$0 \leq \bar{z}_k \leq \bar{b}_k - \bar{a}_k \quad k \in \bar{K} \quad (\text{A.39})$$

$$\bar{x}_{ijk}, \bar{y}_{ik} \in \{0, 1\} \quad i, j \in J, k \in \bar{K} \quad (\text{A.40})$$

$$\bar{t}_i, \bar{z}_k \in \mathbf{R}_+ \quad i \in J, k \in \bar{K} \quad (\text{A.41})$$

Onde:

$$MW = \min_{i \in J} \bar{w}_i \quad (\text{A.42})$$

$$MZ = \max_{k \in K} (\bar{b}_k - \bar{a}_k) + \varepsilon \quad (\text{A.43})$$

ε é um número muito pequeno

$$M > \max \left(\max_{i \in \bar{J}} \bar{l}_i, \max_{k \in \bar{K}} \bar{a}_k \right) + \max_{i, j \in \bar{J} \cup \{0\}} \bar{c}_{ij} \quad (\text{A.44})$$

As equações (A.31) garantem que caso uma tarefa seja realizada, ela será processada por uma única máquina. As restrições (A.32) garante que as tarefas terão uma única antecessora e uma única sucessora. Cada máquina terá apenas uma tarefa na primeira

posição, como mostra (A.33). As janelas de tempo das tarefas são respeitadas pelas restrições (A.34) a (A.37). Em A.38 é garantido que as tarefas serão realizadas por máquinas que podem processá-las. O tempo disponível (ou ocioso) de uma máquina k no fim do dia está no intervalo mostrado em (A.39). As demais restrições definem as variáveis.

Assumindo que o tempo de *setup* é zero, que todas as prioridades são iguais a um ($w_i = 1$), que todas as máquinas podem processar todas as tarefas ($s_{ik} = 1, \forall i \in J, k \in K$), que as máquinas estão disponíveis entre o instante da chegada da tarefa mais cedo ($\bar{a}_k \leq \min_{i \in J} \bar{e}_i$) e o horário necessário para processar todas as tarefas ou o instante da saída da tarefa mais tarde ($\max_{i \in J} \bar{l}_i \leq \bar{b}_k$), que $MZ \rightarrow \infty$ ou $\frac{1}{MZ} = 0$, o modelo anterior pode ser re-escrito da maneira a seguir, que será chamado de PMPI-2:

Formulação do PMPI-2:

$$\min \sum_{i \in J} (1 - \sum_{k \in K} \bar{y}_{ik}) \quad (\text{A.45})$$

Sujeito a:

$$\sum_{k \in K} \bar{y}_{ik} \leq 1 \quad i \in \bar{J} \quad (\text{A.46})$$

$$\sum_{j \in J \cup \{0\} \setminus i} \bar{x}_{ijk} = \sum_{j \in J \cup \{0\} \setminus i} \bar{x}_{jik} = \bar{y}_{ik} \quad k \in \bar{K}, i \in \bar{J} \quad (\text{A.47})$$

$$\sum_{i \in J} \bar{x}_{0ik} \leq 1 \quad k \in \bar{K} \quad (\text{A.48})$$

$$\bar{e}_i \leq \bar{t}_i \quad i \in \bar{J} \quad (\text{A.49})$$

$$\bar{T}_i \geq \bar{t}_i + \bar{p}_i - \bar{l}_i \quad i \in \bar{J} \quad (\text{A.50})$$

$$\bar{T}_i \leq M(1 - \sum_{k \in K} \bar{y}_{ik}) \quad i \in \bar{J} \quad (\text{A.51})$$

$$\bar{t}_i + \bar{p}_i \leq \bar{t}_j + M(1 - \bar{x}_{ijk}) \quad k \in \bar{K}, i \neq j \in \bar{J} \quad (\text{A.52})$$

$$\bar{x}_{ijk}, \bar{y}_{ik} \in \{0, 1\} \quad i, j \in J, k \in \bar{K} \quad (\text{A.53})$$

$$\bar{T}_i, \bar{t}_i \in \mathbf{R}_+ \quad i \in J, k \in \bar{K} \quad (\text{A.54})$$

Onde:

$$M \text{ é um número suficientemente grande} \quad (\text{A.55})$$

Pode-se notar que o PMPIS e o PMPI-2 tem diversas equivalências:

- A função objetivo (A.30) ficou reduzida a (A.45).
- As restrições (A.31) a (A.33) são idênticas a (A.46) a (A.48), garantido que uma tarefa não será designada a mais de uma máquina e que, uma vez designada, essa tarefa terá uma única antecessora e uma única sucessora.
- A restrição (A.49) garante que o início da janela de tempo da tarefa será respeitado, do mesmo modo que a primeira desigualdade de (A.34).
- As restrições (A.50) e (A.51) são equivalentes à segunda desigualdade de (A.34). Note que em (A.51), quando uma tarefa for designada a um técnico, $\bar{T}_i$ será igual a 0 e, portanto, pela restrição (A.50) a tarefa deverá ser concluída antes do término da sua janela de tempo.
- Uma vez que assumimos que o tempo de *setup* é zero, a restrição (A.52) é idêntica à restrição (A.35), que garantem que uma tarefa sucessora somente começará a ser executada após o término da sua antecessora.
- As restrições (A.36) a (A.39) deixam de existir pelos valores adotados já descritos no parágrafo anterior à formulação do PMPI-2.
- As variáveis assumem valores equivalentes. Note que a variável T existe apenas para associar as restrições (A.50) e (A.51).

Por tanto, pela equivalência dos dois modelos, pode-se concluir que o PMPI-2 é reduzível ao PMPIS.

2. Vamos agora mostrar que o problema de minimizar o número de tarefas atrasadas em máquinas paralelas idênticas (PMPI) é reduzível ao problema de minimizar o número de tarefas não realizadas em máquinas paralelas idênticas (chamado anteriormente de PMPI-2). No PMPI todas as tarefas serão realizadas e, portanto, o modelo de PLIM do PMPI será:

Variáveis:

- $\bar{y}_{ik}$ Assume 1 se a tarefa i é realizada pela máquina k sem atraso e 0, caso contrário;
- $\bar{x}_{ijk}$ Assume 1 se a tarefa i precede a tarefa j na máquina k e 0, caso contrário;
- $\bar{t}_i$ Início da execução da tarefa i ;
- $\bar{T}_i$ Variável real positiva assumindo os valores necessários de acordo com as restrições a seguir.

Formulação do PMPI:

$$\min \sum_{i \in J} (1 - \sum_{k \in K} \bar{y}_{ik}) \quad (\text{A.56})$$

Sujeito a:

$$\sum_{k \in K} \sum_{j \in J \cup \{0\} \setminus i} \bar{x}_{ijk} = 1 \quad i \in \bar{J} \quad (\text{A.57})$$

$$\sum_{j \in J \cup \{0\} \setminus i} \bar{x}_{ijk} = \sum_{j \in J \cup \{0\} \setminus i} \bar{x}_{jik} \quad k \in \bar{K}, i \in \bar{J} \quad (\text{A.58})$$

$$\sum_{i \in J} \bar{x}_{0ik} \leq 1 \quad k \in \bar{K} \quad (\text{A.59})$$

$$\bar{e}_i \leq \bar{t}_i \quad i \in \bar{J} \quad (\text{A.60})$$

$$\bar{T}_i \geq \bar{t}_i + \bar{p}_i - \bar{l}_i \quad i \in \bar{J} \quad (\text{A.61})$$

$$\bar{T}_i \leq M(1 - \sum_{k \in K} \bar{y}_{ik}) \quad i \in \bar{J} \quad (\text{A.62})$$

$$\bar{t}_i + \bar{p}_i \leq \bar{t}_j + M(1 - \bar{x}_{ijk}) \quad k \in \bar{K}, i \neq j \in \bar{J} \quad (\text{A.63})$$

$$\bar{x}_{ijk}, \bar{y}_{ik} \in \{0, 1\} \quad i, j \in J, k \in \bar{K} \quad (\text{A.64})$$

$$\bar{T}_i, \bar{t}_i \in \mathbf{R}_+ \quad i \in J, k \in \bar{K} \quad (\text{A.65})$$

Onde:

$$M \text{ é um número suficientemente grande} \quad (\text{A.66})$$

Podemos ressaltar as seguintes diferenças do PMPI para o PMPI-2:

- A variável $\bar{y}_{ik}$, que representava as tarefas realizadas no PMPI-2, foi substituída pela variável $\bar{\bar{y}}_{ik}$, que indica se uma tarefa foi realizada sem atraso.
- A restrição (A.46) foi re-escrita usando a variável $\bar{x}_{ijk}$ e passou a ser uma igualdade, como se observa em (A.57).
- A restrição (A.47) deixou de ter a última igualdade, mas manteve o seu mesmo significado em (A.58).
- As demais restrições não sofreram alterações.

Queremos mostrar que o resultado do PMPI-2 será igual ao do PMPI e, que portanto, as formulações são equivalentes. Vamos tentar fazer isso por absurdo:

1. Vamos supor que o PMPI-2 tenha uma solução menor do que o PMPI. Então existem mais tarefas realizadas no PMPI-2 do que tarefas sem atraso no PMPI, ou seja, existe uma designação de tarefas a máquinas e/ou uma sequência em uma máquina que resulta em uma melhor solução para o PMPI. Mas repare que as restrições (A.49) a (A.51), que são reponsáveis pelas janelas de tempo, e a restrição (A.52), que é responsável pela sequencias de tarefas nas máquinas no PMPI-2, são idênticas às restrições (A.60) a (A.63), que têm exatamente as mesmas finalidades no PMPI. Portanto, é contraditório que o PMPI-2 tenha uma solução menor do que o PMPI.
2. Analogamente, é contraditório que o PMPI tenha uma solução menor do que o PMPI-2.

Portanto, por absurdo, podemos concluir que os problemas são equivalentes.

3. Já mostramos anteriormente que o PMPI, que é NP-difícil, é reduzível ao PMPI-2 e que o PMPI-2 é reduzível ao PMPIS. Falta agora o passo final, provando que PMPIS é reduzível ao PETC e que, portanto, o PETC é NP-difícil.

Como as restrições do PETC e do PMPIS são iguais, basta provar que as funções objetivos também são idênticas, ou seja, que maximizar o número ponderado de tarefas

realizadas tem o mesmo sentido de minimizar o número ponderado de tarefas não realizadas. Se isso não fosse verdade, haveria alguma tarefa de uma solução que deixaria ou passaria a ser feita com a outra função objetivo, o que é contraditório, uma vez que as restrições são as mesmas. Portanto, a formulação do PMPIS é idêntica a do PETC e o PMPIS é reduzível ao PETC, o que conclui a demonstração. O PETC é realmente NP-difícil. **(c.q.d.)**

Apêndice B

Pseudocódigo da heurística construtiva STT

As seguintes variáveis foram utilizadas no pseudocódigo da heurística Menor Tempo de Viagem (*Shorter Travel Time* - STT):

- $n_tarefas_do_técnico[j]$ é o número de tarefas já incluídas na rota do técnico j .
- $técnico_avaliado[j]$ é uma variável binária que assume 1 quando existe uma solução factível para o técnico j assumir uma tarefa i ; caso contrário, a variável é igual a zero.
- $TTV[j]$ é o tempo total de viagem caso o técnico j realize uma tarefa i .
- $prioritária[i]$ é a i -ésima tarefa classificada pelo método utilizado na primeira fase.
- $posição$ é a posição que a nova tarefa poderia ocupar na rota do técnico que está sendo avaliado no laço das linhas 19–51.
- $posição_melhor$ é a melhor posição que a nova tarefa poderá ocupar na rota do técnico j , no laço das linhas 19–51. Concretamente, será a rota que resultar no maior z (tempo ocioso) no final do dia.
- $novo_TTV$ armazena o tempo total de viagem da rota do técnico avaliado (nas linhas 19–51) que resultar no maior z .
- $novo_z$ armazena o maior tempo ocioso do técnico avaliado entre as soluções factíveis para esse técnico realizar a nova tarefa.

Algoritmo 4 Pseudocódigo da heurística construtiva Menor Tempo de Viagem (STT)

```

1: Para  $i \leftarrow 1$  a  $n$  faça
2:    $priorit\acute{a}ria[i] \leftarrow i$ 
3:   Se  $\frac{n}{m} < 10$  então
4:      $vetor[j] \leftarrow w_i - \frac{NST_i}{m+1}$ 
5:   Senão
6:      $vetor[j] \leftarrow w_i + \frac{w_i}{p_i} + \frac{p_i}{l_i - e_i}$ 
7:   fim Se
8: fim Para
9: Merge-Sort(1,  $n$ , vetor, prioritaria) (ordena prioritária de acordo com os va-
  lores do vetor)
10: Para  $j \leftarrow 1$  a  $m$  faça
11:    $n\_tarefas\_do\_t\acute{e}cnico[j] \leftarrow 0$ 
12:    $z[j] \leftarrow b[j] - a[j]$ 
13: fim Para
14: Para  $i \leftarrow 1$  a  $n$  faça (laço para tentar alocar todas as  $n$  tarefas)
15:   Para  $j \leftarrow 1$  a  $m$  faça
16:      $t\acute{e}cnico\_avaliado[j] \leftarrow 0$ 
17:      $TTV[j] \leftarrow 0$ 
18:   fim Para
19:   Para  $j \leftarrow 1$  a  $m$  faça (avaliar todos os técnicos)
20:     Se  $s[priorit\acute{a}ria[i]][j] = 1$  então (se o técnico tem habilidade)
21:       criar lista ligada, cópia da rota atual do técnico  $j$  (linha  $j$  da matriz rota)
22:       inserir a tarefa prioritária[ $i$ ] na última posição da lista ligada
23:        $posi\tilde{c}\tilde{a}o \leftarrow n\_tarefas\_do\_t\acute{e}cnico[j] + 1$ 
24:        $posi\tilde{c}\tilde{a}o\_melhor \leftarrow 0$ 
25:        $novo\_TTV \leftarrow b[i] - a[i]$ 
26:        $novo\_z \leftarrow 0$ 
27:       Enquanto  $posi\tilde{c}\tilde{a}o > 0$  faça (escolher melhor posição factível)
28:          $TTV[j] \leftarrow \text{calcula\_TTV}()$ 
29:         Se  $TTV[j] > -1$  então
30:           Se  $novo\_z < candidata\_z[j]$  então
31:              $novo\_TTV \leftarrow TTV[j]$ 
32:              $novo\_z \leftarrow candidata\_z[j]$ 
33:              $posi\tilde{c}\tilde{a}o\_melhor \leftarrow posi\tilde{c}\tilde{a}o$ 
34:           fim Se
35:            $posi\tilde{c}\tilde{a}o \leftarrow posi\tilde{c}\tilde{a}o - 1$ 
36:           Se  $posi\tilde{c}\tilde{a}o > 0$  então
37:             trocar a tarefa prioritária[ $i$ ] com a antecessora
38:           fim Se
39:         fim Se
40:       fim Enquanto

```

```

41:      Se  $novo\_TTV < b[j] - a[j]$  então (armazena dados da melhor so-
      lução do técnico  $j$ )
42:           $técnico\_avaliado[j] \leftarrow 1$ 
43:           $TTV[j] \leftarrow novo\_TTV$ 
44:           $candidata\_z[j] \leftarrow novo\_z$ 
45:          Remover a tarefa prioritária $[i]$  da primeira posição da lista
46:          Inserir a tarefa prioritária $[i]$  na posição_melhor da lista
47:          Copiar a lista para a linha  $j$  da matriz rota_candidata
48:           $n\_tarefas\_candidata\_rota[j] \leftarrow n\_tarefas\_do\_técnico[j] + 1$ 
49:      fim Se
50:  fim Se
51: fim Para
52:  $menor\_TTV \leftarrow b[j] - a[j]$ 
53:  $técnico\_escolhido \leftarrow -1$ 
54: Para  $j \leftarrow 1$  a  $m$  faça (laço para identificar o técnico escolhido)
55:     Se  $técnico\_avaliado[j] = 1$  então
56:         Se  $TTV[j] < menor\_TTV$  então
57:              $menor\_TTV \leftarrow TTV[j]$ 
58:              $técnico\_escolhido \leftarrow j$ 
59:     Senão
60:         Se  $TTV[j] = menor\_TTV$  e  $candidata\_z[técnico\_escolhido] < can-$ 
        didata_ $_z[j]$ ) então
61:              $técnico\_escolhido \leftarrow j$ 
62:         fim Se
63:     fim Se
64: fim Se
65: fim Para
66: Se  $técnico\_escolhido > -1$  então (fixar a rota do técnico escolhido)
67:      $z[técnico\_escolhido] \leftarrow candidata\_z[técnico\_escolhido]$ 
68:      $n\_tarefas\_do\_técnico[técnico\_escolhido] \leftarrow n\_tarefas\_candidata\_rota$ 
    [técnico_ $escolhido$ ]
69:     Copiar a linha  $técnico\_escolhido$  da matriz rota_candidata na linha téc-
    nico_ $escolhido$  da matriz rota
70: fim Se
71: fim Para

```

- *calcula_TTV()* é uma função que calcula o tempo total de viagem do técnico avaliado para uma rota dada. Caso a rota não seja factível, a função retorna valor -1 .
- *candidata_z[j]* a função *calcula_TTV()* armazena nessa variável o tempo ocioso do técnico j no final do dia para uma rota dada, caso seja factível.
- *menor_TTV*: caso diversos técnicos com a *melhor_habilidade* possam executar a tarefa, essa variável armazena o menor tempo de viagem entre eles.
- *técnico_escolhido* recebe o número do técnico que será escolhido para executar a tarefa, de acordo com os critérios de escolha: menor tempo de viagem e maior ociosidade no fim do dia.
- matriz *rota* é a matriz com a sequência de tarefas já designadas aos técnicos.
- matriz *rota_candidata*: ao avaliar todos os técnicos e seleccionar aqueles que têm uma solução factível para executar uma tarefa i , a matriz *rota_candidata* armazena essas rotas factíveis. Quando um técnico j for escolhido, a linha j dessa matriz será copiada para a linha j da matriz *rota*.

O Algoritmo 4 apresenta o pseudocódigo da heurística construtiva STT.

Apêndice C

Exemplo multi em problemas com janelas de tempo apertadas

Para ilustrar as soluções para um problema com janelas de tempo apertadas, a seguir encontra-se um exemplo com 16 tarefas e dois técnicos, onde todas as tarefas têm janelas de tempo igual a 2 horas. O primeiro técnico trabalha entre 8h e 17h e pode executar todas as tarefas, com exceção das tarefas 3, 8, 11, 12 e 15; o segundo técnico trabalha das 11h às 20h e pode executar as tarefas, exceto 6, 7, 9, 13, 14 e 16. A Tabela C.2 apresenta os demais parâmetros do problema. As prioridades das tarefas variam entre 1 (baixa) e 10 (alta).

A fronteira ótima de Pareto contém 6 soluções, apresentadas na Tabela C.1 e na Figura C.1. A seguir será feita uma análise da diferença entre as soluções extremas e uma intermediária, para verificar a influência de cada função objetivo nos resultados. Tanto pela descrição das três soluções apresentadas quanto pela Figura C.1, pode-se concluir que nas instâncias com janelas de tempo apertadas há um impacto menor da segunda função objetivo (o número de soluções da fronteira de Pareto tende a ser menor), mas que ela continua tendo um importante papel para evitar que clientes prioritários sejam atendidos no fim de suas janelas de tempo, como ocorre na solução 1.

Tabela C.1 – Soluções da fronteira de Pareto para o exemplo

Solução	f_1	f_2
1	76,3	48,9
2	76,1	59,5
3	72,2	60,1
4	67,2	60,7
5	66,4	60,8
6	65,4	61,7

Figura C.1 – Soluções da fronteira de Pareto para o exemplo

Fonte: elaborado pelo autor.

Tabela C.2 – Parâmetros do exemplo: a) prioridades, tempos de execução e janelas de tempo das tarefas. b) Tempo de deslocamento c_{ij} (em minutos) entre as tarefas e a sede da empresa.

(a)

Tarefa	Prioridade	Tempo de execução	Início da janela	Fim da janela
	w_i	p_i	e_i	l_i
1	10	40min	10h35min	12h35min
2	7	35min	14h10min	16h10min
3	9	30min	17h05min	19h05min
4	4	30min	11h30min	13h30min
5	9	1h10min	11h20min	13h20min
6	1	1h40min	12h15min	14h15min
7	7	35min	12h45min	14h45min
8	1	40min	16h25min	18h25min
9	4	1h35min	9h00min	11h00min
10	5	45min	13h00min	15h00min
11	8	1h30min	15h40min	17h40min
12	4	55min	15h35min	17h35min
13	4	50min	13h55min	15h55min
14	9	35min	9h20min	11h20min
15	6	1h55min	14h35min	16h35min
16	8	1h00min	11h55min	13h55min

(b)

c_{ij}	base	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
base	0	37	29	37	17	45	55	23	67	40	48	38	70	67	36	2	34
1	37	0	13	23	31	10	36	21	42	11	18	24	39	35	47	38	4
2	29	13	0	14	18	23	47	22	54	23	30	31	42	38	49	29	12
3	37	23	14	0	21	33	59	35	65	34	41	44	35	32	62	36	24
4	17	31	18	21	0	41	59	28	69	38	46	42	56	53	50	16	29
5	45	10	23	33	41	0	29	25	33	8	9	21	40	36	48	46	12
6	55	36	47	59	59	29	0	32	13	25	20	18	67	64	36	57	35
7	23	21	22	35	28	25	32	0	44	18	25	15	59	55	28	25	17
8	67	42	54	65	69	33	13	44	0	32	24	29	66	63	49	69	42
9	40	11	23	34	38	8	25	18	32	0	8	14	47	43	41	41	10
10	48	18	30	41	46	9	20	25	24	8	0	16	48	45	43	49	18
11	38	24	31	44	42	21	18	15	29	14	16	0	60	57	28	40	21
12	70	39	42	35	56	40	67	59	66	47	48	60	0	4	86	70	42
13	67	35	38	32	53	36	64	55	63	43	45	57	4	0	82	67	39
14	36	47	49	62	50	48	36	28	49	41	43	28	86	82	0	38	44
15	2	38	29	36	16	46	57	25	69	41	49	40	70	67	38	0	35
16	34	4	12	24	29	12	35	17	42	10	18	21	42	39	44	35	0

Figura C.2 – Solução 1 do exemplo

Fonte: elaborado pelo autor.

A solução 1, que maximiza f_1 , é apresentada na Figura C.2 e detalhada na Tabela C.3. Note que as tarefas com prioridades superiores a 6 são programadas, mas que as duas últimas tarefas do técnico 1 têm folgas inferiores 0,05, ou seja, o técnico 1 termina de executá-las faltando menos de 3 minutos para o fim de sua janelas de tempo. Caso ocorram atrasos inesperados, essas tarefas estarão sujeitas a serem canceladas (uma vez que não é permitida a violação das janelas de tempo). Espera-se que a segunda função objetivo tenha um impacto direto no aumento dessas folgas.

Tabela C.3 – Informações das tarefas executadas por cada técnico na solução 1:
a) Técnico 1. b) Técnico 2.

Tarefa	14	1	4	16	7	Empresa
Horário de chegada do técnico	8h36min	10h42min	11h53min	12h52min	14h09min	15h07min
Tempo de espera	44min	-	-	-	-	-
Início da execução	9h20min	10h42min	11h53min	12h52min	14h09min	-
Fim da execução	9h55min	11h22min	12h23min	13h52min	14h44min	-
Folga (γ)	1,00	0,91	0,74	0,05	0,01	-

(a)

Tarefa	5	10	2	11	3	Empresa
Horário de chegada do técnico	11h45min	13h05min	14h20min	15h27min	17h54min	19h01min
Tempo de espera	-	-	-	13min	-	-
Início da execução	11h45min	13h05min	14h20min	15h40min	17h54min	-
Fim da execução	12h55min	13h50min	14h55min	17h10min	18h24min	-
Folga (γ)	0,49	0,94	0,88	1,00	0,45	-

(b)

Figura C.3 – Solução 6 do exemplo

Fonte: elaborado pelo autor.

A solução 6, que tem o máximo valor para f_2 , é apresentada na Figura C.3 e detalhada na Tabela C.4. Em relação à solução 1, os técnicos têm diferentes tarefas programadas e distintas rotas. Note que as folgas têm valor superior a 0,9 (exceto a tarefa 7, com folga 0,68), mas que algumas tarefas com alta prioridade não serão realizadas (como as tarefas 5 e 11, de prioridades 9 e 8, respectivamente). A seguir, um solução intermediária que pode ser vantajosa de acordo com a Figura C.1 será analisada.

Tabela C.4 – Informações das tarefas executadas por cada técnico na solução 6:
a) Técnico 1. b) Técnico 2.

Tarefa	14	1	16	7	2	Empresa
Horário de chegada do técnico	8h36min	10h42min	11h26min	13h12min	14h08min	15h14min
Tempo de espera	44min	-	29min	-	2min	-
Início da execução	9h20min	10h42min	11h55min	13h12min	14h10min	-
Fim da execução	9h55min	11h22min	12h55min	13h47min	14h45min	-
Folga (γ)	1,00	0,91	1,00	0,68	1,00	-

(a)

Tarefa	4	10	15	3	Empresa
Horário de chegada do técnico	11h17min	12h46min	14h34min	17h06min	18h13min
Tempo de espera	13min	14min	1min	-	-
Início da execução	11h30min	13h00min	14h35min	17h06min	-
Fim da execução	12h00min	13h45min	16h30min	17h36min	-
Folga (γ)	1,00	1,00	1,00	0,98	-

(b)

Figura C.4 – Solução 2 do exemplo

Fonte: elaborado pelo autor.

A solução 2 é apresentada na Figura C.4 e detalhada na Tabela C.5. Em relação à solução 1, as tarefas e a rota do técnico 2 são idênticas. Já o técnico 1 deixa de ter a tarefa 4 em sua rota e recebe a tarefa 13 (ambas têm prioridade 4), mantendo o mesmo somatório de prioridades das tarefas realizadas (que é 76) e aumentando significativamente as folgas das tarefas 16 e 7 para 1,00 e 0,68, respectivamente. Com o aumento dessas folgas, o técnico tem a sua disponibilidade no final do dia reduzida de 1h53min para 21 min (pois há um aumento nos tempos de deslocamentos e de esperas).

Tabela C.5 – Informações das tarefas executadas por cada técnico na solução 2:
a) Técnico 1. b) Técnico 2.

Tarefa	14	1	16	7	13	Empresa
Horário de chegada do técnico	8h36min	10h42min	11h26min	13h12min	14h42min	16h39min
Tempo de espera	44min	-	29min	-	-	-
Início da execução	9h20min	10h42min	11h55min	13h12min	14h42min	-
Fim da execução	9h55min	11h22min	12h55min	13h47min	15h32min	-
Folga (γ)	1,00	0,91	1,00	0,68	0,32	-

(a)

Tarefa	5	10	2	11	3	Empresa
Horário de chegada do técnico	11h45min	13h05min	14h20min	15h27min	17h54min	19h01min
Tempo de espera	-	-	-	13min	-	-
Início da execução	11h45min	13h05min	14h20min	15h40min	17h54min	-
Fim da execução	12h55min	13h50min	14h55min	17h10min	18h24min	-
Folga (γ)	0,49	0,94	0,88	1,00	0,45	-

(b)

Apêndice D

Calibração dos parâmetros do BRKGA multiobjetivo

Para exemplificar como o tamanho do conjunto elite para o BRKGA-Balanceado foi determinado, apresenta-se a seguir os resultados obtidos para os diversos tamanhos possíveis do conjunto elite. Os experimentos foram realizados com os outros parâmetros fixos ($n_m = 15\%$, $p_e = 60\%$ e tamanho da população (n_p) = 1000 para os casos 1 a 11, 500 para o caso 12 e 200 cromossomos para o caso 13).

A Tabela D.1 apresenta os resultados obtidos. Para os problemas pequenos (Tabela D.1(a)), há um desempenho semelhante para os resultados com diferentes tamanhos de n_e . Já para os problemas médios e grandes, os melhores valores para as medidas porcentagem de hipervolume (I_{PH}) e epsilon multiplicativo (I_ϵ) ocorrem quando $n_e = 35\%$ e, por isso, esse valor foi adotado. Na medida porcentagem de soluções eficientes (I_{PSE}), quando $n_e = 35\%$ obtém-se o terceiro melhor desempenho para os problemas médios e grandes.

Outro aspecto que se pode destacar é que algumas vezes vários valores adotados obtêm desempenhos semelhantes. Neste caso, nota-se que a diferença entre os índices de medida quando $n_e = 20\%$, 25% e 35% não é grande e intui-se – considerando essa e todas as demais calibrações realizadas – que vários conjuntos de parâmetros podem gerar bons resultados e que, portanto, o BRKGA é robusto (pouco sensível às variações dos parâmetros). Seria necessário desenvolver um estudo sistemático e com análise estatística dos resultados para confirmar essa suposição.

Tabela D.1 – Desempenho do **BRKGA-Balanceado** para determinação do tamanho do conjunto elite: a) instâncias pequenas (casos 1 a 4) em relação ao conjunto de referência. b) instâncias médias e grandes (casos 5 a 13) em relação ao conjunto de referência.

(a)						
n_e	15%	20%	25%	30%	35%	40%
I_{PSE} (%)	98,1	95,8	97,2	95,9	96,1	94,8
I_{PH} (%)	100,0	100,0	100,0	100,0	99,9	100,0
I_ϵ	1,000	0,999	1,000	0,999	0,998	0,999

(b)						
n_e	15%	20%	25%	30%	35%	40%
I_{PSE} (%)	40,5	41,6	39,0	34,7	40,0	35,8
I_{PH} (%)	96,7	96,8	96,6	96,1	97,4	96,4
I_ϵ	0,990	0,991	0,991	0,991	0,993	0,990

Referências Bibliográficas

ABRAHAM, A.; JAIN, L.. **Evolutionary multiobjective optimization**. Springer, London, 2005.

ANDRADE, C. E.; TOSO, R. F.; RESENDE, M. G.; MIYAZAWA, F. K. Biased random-key genetic algorithms for the winner determination problem in combinatorial auctions. **Evolutionary computation**. v. 23, n. 2, p. 279–307, 2015.

AQUINO, R. F.; ARROYO, J. E. C. Um algoritmo memético para o problema multiobjetivo de roteamento de veículos com janelas de tempo. In: XLVII SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL (SBPO), 2015, Porto de Galinhas – PE. Anais do XLVII Simpósio Brasileiro de Pesquisa Operacional. p. 4154–4165.

ARENALES, M. N.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. **Pesquisa operacional**: Para Cursos de Engenharia. Elsevier Brasil, 2007.

ARROYO, J. E. C. **Heurísticas e metaheurísticas para otimização combinatória multiobjetivo**. 2002. 232 páginas. Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Campinas, 2002.

ARROYO, J. E. C.; ARMENTANO, V. A. Genetic local search for multi-objective flowshop scheduling problems. **European Journal of Operational Research**, v. 167, n. 3, p. 717–738, 2005.

ASCHEUER, N.; FISCHETTI, M.; GRÖTSCHEL, M. Solving the asymmetric traveling salesman problem with time windows by branch-and-cut. **Mathematical Programming**, v. 90, n. 3, p. 475–506, 2001.

BAGCHI, T. P. **Multiobjective scheduling by genetic algorithms**. Massachusetts: Springer Science & Business Media, 1999. 358 p.

BAÑOS, R.; ORTEGA, J.; GIL, C.; FERNÁNDEZ, A.; DE TORO, F. A Simulated Annealing-based parallel multi-objective approach to vehicle routing problems with time windows. **Expert Systems with Applications**. v. 40, n. 5, p. 1696–1707, 2013.

BIRATTARI, M. **On the Estimation of the Expected Performance of a Metaheuristic on a Class of Instances**. Bruxelas, Bélgica: Institut de Recherches Interdisciplinaires et de Developpements en Intelligence Artificielle, 2004. (Technical Report No. TR/IRIDIA/2004-001).

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. **ORSA journal on computing**, v. 6, n. 2, p. 154–160, 1994.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM Computing Surveys (CSUR)**, v. 35, n. 3, p. 268–308, 2003.

BRANCO, F. J. C.; NAGANO, M. S. Avaliação de métodos heurísticos construtivos

para o problema de programação de operações no-wait flow shop. **Revista Produção Online**, v. 9, n. 1, 2009.

BRANDÃO, J. S.; NORONHA, T. F.; RESENDE, M. G.; RIBEIRO, C. C. A biased random-key genetic algorithm for single-round divisible load scheduling. **International Transactions in Operational Research**, n. 22, p. 823–839, 2015.

BURIOL, L. S.; HIRSCH, M. J.; PARDALOS, P. M.; QUERIDO, T.; RESENDE, M. G.; RITT, M. A biased random-key genetic algorithm for road congestion minimization. **Optimization Letters**, v. 4, n. 4, p. 619–633, 2010.

CASJENS, S.; SCHWENDER, H.; BRUNING, T.; ICKSTADT, K. A novel crossover operator based on variable importance for evolutionary multi-objective optimization with tree representation. **Journal of Heuristics**, v. 21, n. 1, p. 1–24, 2015.

CHANG, P.; LIN, K.; PAI, P.; ZHONG, C.; LIN, C.; HUNG, L. Ant colony optimization system for a multi-quantitative and qualitative objective job-shop parallel-machine-scheduling problem. **International Journal of Production Research**, v. 46, n. 20, p. 5719–5759, 2008.

CHOOBINEH, F. F.; MOHEBBI, E.; KHOO, H. A multi-objective tabu search for a single-machine scheduling problem with sequence-dependent setup times. **European Journal of Operational Research**, v. 175, n. 1, p. 318–337, 2006.

COELLO, C. A. C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. **Computer methods in applied mechanics and engineering**, v. 191, n. 11, p. 1245–1287, 2002.

COELLO, C. A. C.; LAMONT, G. B.; VAN VELDHUIZEN, D. A. **Evolutionary algorithms for solving multi-objective problems**. New York: Kluwer Academic, 2007. 800 p.

CORDEAU, J.; LAPORTE, G.; PASIN, F.; ROPKE, S. Scheduling technicians and tasks in a telecommunications company. **Journal of Scheduling**, v. 13, n. 4, p. 393 – 409, 2010.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Introduction to algorithms. Estados Unidos, Massachusetts: 2001. Segunda edição. 1180 p.

CORNE, D. W.; KNOWLES, J. D.; OATES, M. J. The Pareto envelope-based selection algorithm for multiobjective optimization. In: **Parallel Problem Solving from Nature PPSN VI**. Springer Berlin Heidelberg, 2000. p. 839–848.

CORTÉS, C. E.; GENDREAU, M.; ROUSSEAU, L. M.; SOUYRIS, S.; WEINTRAUB, A. Branch-and-price and constraint programming for solving a real-life technician dispatching problem. **European Journal of Operational Research**, v. 238, n. 1, p. 300–312, 2014.

DARWIN, C. **A origem das espécies**. Hemus, 2003.

DAVIS, M. M.; CHASE, R. B.; AQUILANO, N. J. **Fundamentos da administração da produção**. Bookman, 2001.

DEB, K. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. **Evolutionary computation**, v. 7, n. 3, p. 205–230, 1999.

DEB, K. **Multi-objective optimization using evolutionary algorithms**. Londres: John Wiley & Sons, 2001. 496 p.

DEB, K. Introduction to Evolutionary Multiobjective Optimization. In: **Metaheuris-**

- tics for Multiobjective Optimisation.** Springer Berlin Heidelberg, p. 59–96, 2004.
- DEB, K.; PRATAP, A.; AGRAWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. **Evolutionary Computation**, v. 6, n. 2, p. 182–197, 2002.
- DEB, K.; GOLDBERG, D. E. An investigation of niche and species formation in genetic function optimization. In: **Proceedings of the 3rd international conference on genetic algorithms.** Morgan Kaufmann Publishers Inc., 1989. p. 42–50.
- DEMIR, E.; BEKTAŞ, T.; LAPORTE, G. The bi-objective pollution-routing problem. *European Journal of Operational Research*, v. 232, n. 3, p. 464–478, 2014.
- DOHN, A.; KOLIND, E.; CLAUSEN, J. The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. **Computers & Operations Research**, v. 36, n. 4, p. 1145–1157, 2009.
- DONOSO, Y.; FABREGAT, R. **Multi-objective optimization in computer networks using metaheuristics.** Nova Iorque: CRC Press, 2007. 449 p.
- DUAN, X. Multiobjective Optimal Reactive Power Flow Using Elitist Nondominated Sorting Genetic Algorithm-Comparison and Improvement. **Journal of Electrical Engineering & Technology**, v. 5, n. 1, p. 70–78, 2010.
- DUARTE, A.; MARTÍ, R.; RESENDE, M. G. C.; SILVA, R. M. A. Improved heuristics for the regenerator location problem. **International Transactions in Operational Research**, v. 21, n. 4, p. 541–558, 2014.
- EHRGOTT, M. **Multicriteria optimization.** Berlin: Springer Science & Business Media, 2006. 323 p.
- FONSECA, V. G.; FONSECA, C. M.; HALL, A. O. Inferential performance assessment of stochastic optimisers and the attainment function. In: **Evolutionary multi-criterion optimization.** Springer Berlin Heidelberg, p. 213–225, 2001.
- G1.GLOBO.COM. Consumidores reclamam de atrasos na entrega de compras pela internet. *Jornal Nacional, Rede Globo de Televisão*, 1 set. 2015. Disponível em: <<http://g1.globo.com/jornal-nacional/noticia/2015/09/consumidores-reclamam-de-atrasos-na-entrega-de-compras-pela-internet.html>>. Acesso em: 2 set. 2015.
- GAREY, M.R.; JOHNSON, D.S. **Computers and Intractability: A Guide to the Theory of NP-Completeness.** W. H. Freeman, 1979.
- GHOSEIRI, K.; GHANNADPOUR, S. F. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. **Applied Soft Computing**, v. 10, n. 4, p. 1096–1107, 2010.
- GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning.** Addison-Wesley, 1989.
- GOLDBERG, D. E.; DEB, K. A comparative analysis of selection schemes used in genetic algorithms. In: RAWLINS, G. J. E. **Foundations of genetic algorithms.** Morgan Kaufmann Publishers Inc., 1999. p. 69–93.
- GOLDBERG, D. E.; HOLLAND, J. H. **Genetic algorithms and machine learning.** Machine learning, v. 3, n. 2, p. 95–99, 1989.
- GOLDBERG, D. E.; RICHARDSON, J. Genetic algorithms with sharing for multimodal function optimization. In: **Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms.**

Hillsdale, NJ: Lawrence Erlbaum, 1987. p. 41-49.

GOMES, H. C.; DAS NEVES, F. A.; SOUZA, M. J. F. Multi-objective metaheuristic algorithms for the resource-constrained project scheduling problem with precedence relations. **Computers & Operations Research**, v. 44, p. 92–104, 2014.

GONÇALVES, J. F.; MENDES, J. J. M.; RESENDE, M. G. C. A hybrid genetic algorithm for the job shop scheduling problem. **European journal of operational research**, v. 167, n. 1, p. 77–95, 2005.

GONÇALVES, J. F.; RESENDE, M. G. C. Biased random-key genetic algorithms for combinatorial optimization. **Journal of Heuristics**, v. 17, n. 5, p. 487–525, 2011.

GONÇALVES, J. F.; RESENDE, M. G. C. A biased random key genetic algorithm for 2D and 3D bin packing problems. **International Journal of Production Economics**, v. 145, n. 2, p. 500–510, 2013.

GONÇALVES, J. F.; RESENDE, M. G. C. A biased random-key genetic algorithm for the unequal area facility layout problem. **European Journal of Operational Research**, v. 246, n. 1, p. 86–107, 2015.

GONÇALVES, J. F.; RESENDE, M. G. C.; MENDES, J. J. M. A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. **Journal of Heuristics**, v. 17, n. 5, p. 467–486, 2011.

GONÇALVES, J. F.; RESENDE, M. G. C.; TOSO, R. F. An experimental comparison of biased and unbiased random-key genetic algorithms. **Pesquisa Operacional**, v. 34, n. 2, p. 143–164, 2014.

GONÇALVES, J. F.; SOUSA, P. S. A.. A genetic algorithm for lot sizing and scheduling under capacity constraints and allowing backorders. **International Journal of Production Research**, v. 49, n. 9, p. 2683–2703, 2011.

GRASAS, A.; RAMALHINHO, H.; PESSOA, L. S.; RESENDE, M. G.; CABALLÉ, I.; BARBA, N. On the improvement of blood sample collection at clinical laboratories. **BMC Health Services Research**, v. 14, n. 1, p. 12, 2014.

HASHIMOTO, H.; BOUSSIER, S.; VASQUEZ, M.; WILBAUT, C. A GRASP-based approach for technicians and interventions scheduling for telecommunications. **Annals of Operations Research**, v. 183, n. 1, p. 143–161, 2011.

HO, J. C.; CHANG, Y. Minimizing the number of tardy jobs for m parallel machines. **European Journal of Operational Research**, v. 84, n. 2, p. 343–355, 1995.

HUSSIN, M. S.; STÜTZLE, T. Tabu search vs. simulated annealing as a function of the size of quadratic assignment problem instances. **Computers & Operations Research**, v. 43, p. 286–291, 2014.

KAHLBACHER, H. G.; CHENG, T. C. E. Parallel machine scheduling to minimize costs for earliness and number of tardy jobs. **Discrete Applied Mathematics**, v. 47, n. 2, p. 139–164, 1993.

KIM, H.; LIOU, M. New fitness sharing approach for multi-objective genetic algorithms. **Journal of Global Optimization**, v. 55, n. 3, p. 579–595, 2013.

KIM, Y. D. Heuristics for flowshop scheduling problems minimizing mean tardiness. **Journal of the Operational Research Society**, v. 44, n. 1, p. 19–28, 1993.

KONAK, A.; COIT, D. W.; SMITH, A. E. Multi-objective optimization using genetic algorithms: A tutorial. **Reliability Engineering & System Safety**, v. 91, n. 9,

p. 992–1007, 2006.

KNOWLES, J. D.; CORNE, D. W. Approximating the nondominated front using the Pareto archived evolution strategy. **Evolutionary computation**, v. 8, n. 2, p. 149–172, 2000.

KOVACS, A. A.; PARRAGH, S. N.; DOERNER, K. F.; HARTL, R. F. Adaptive large neighborhood search for service technician routing and scheduling problems. **Journal of scheduling**, v. 15, n. 5, p. 579–600, 2012.

LALLA-RUIZ, E.; GONZÁLEZ-VELARDE, J. L.; MELIÁN-BATISTA, B.; MORENO-VEGA, J. M. Biased random key genetic algorithm for the tactical berth allocation problem. **Applied Soft Computing**, v. 22, p. 60–76, 2014.

LENSTRA, J. K.; KAN, A. R.; BRUCKER, P. Complexity of machine scheduling problems. **Annals of discrete mathematics**, v. 1, p. 343–362, 1977.

LI, J.; PAN, Q.; LIANG, Y. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. **Computers & Industrial Engineering**, v. 59, n. 4, p. 647–662, 2010.

LI, Y.; LIM, A.; RODRIGUES, B. Manpower allocation with time windows and job-teaming constraints. **Naval Research Logistics (NRL)**, v. 52, n. 4, p. 302–311, 2005.

LIN, S.; YING, K. Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start simulated-annealing algorithm. **Computers & Operations Research**, v. 40, n. 6, p. 1625–1647, 2013.

LONG, Q.; WU, C.; HUANG, T.; WANG, X. A genetic algorithm for unconstrained multi-objective optimization. **Swarm and Evolutionary Computation**, v. 22, p. 1–14, 2015.

LU, H.; YEN, G. G. Rank-density-based multiobjective genetic algorithm and benchmark test function study. **Evolutionary Computation**, v. 7, n. 4, p. 325–343, 2003.

MARTÍ, R.; CAMPOS, V.; RESENDE, M. G.; DUARTE, A. Multiobjective grasp with path relinking. **European Journal of Operational Research**, v. 240, n. 1, p. 54–71, 2015.

MICHALEWICZ, Z. **Genetic algorithms+ data structures= evolution programs**. Springer Science & Business Media, 2013.

MIETTINEN, K. **Nonlinear multiobjective optimization**. Nova Iorque: Springer Science & Business Media, 2012. 298 p.

MIETTINEN, K. Introduction to Multiobjective Optimization: Noninteractive Approaches In: JÜRGEN, B.; DEB, K.; MIETTINEN, K.; SŁOWINSKI, R. **Multiobjective Optimization Interactive and Evolutionary Approaches**. Berlin Heidelberg, Springer-Verlag, 2008. p. 1–28.

MOHAMMADI, G.; KARAMPOURHAGHGH, A.; SAMAEI, F. A multi-objective optimisation model to integrating flexible process planning and scheduling based on hybrid multi-objective simulated annealing. **International Journal of Production Research**, v. 50, n. 18, p. 5063–5076, 2012.

MORÁN-MIRABAL, L. F.; GONZÁLEZ-VELARDE, J. L.; RESENDE, M. G. C. Randomized heuristics for the family traveling salesperson problem. **International Transactions in Operational Research**, v. 21, n. 1, p. 41–57, 2014.

- MORITZ, R. L.; REICH, E.; SCHWARZ, M.; BERNT, M.; MIDDENDORF, M. Refined ranking relations for selection of solutions in multi objective metaheuristics. **European Journal of Operational Research**, v. 243, n. 2, p. 454–464, 2015.
- NORONHA, T. F.; RESENDE, M. G. C.; RIBEIRO, C. C. A biased random-key genetic algorithm for routing and wavelength assignment. **Journal of Global Optimization**, v. 50, n. 3, p. 503–518, 2011.
- OMBUKI, B.; ROSS, B. J.; HANSHAR, F. Multi-objective genetic algorithms for vehicle routing problem with time windows. **Applied Intelligence**, v. 24, n. 1, p. 17–30, 2006.
- OSMAN, I. H.; LAPORTE, G. Metaheuristics: A bibliography. **Annals of Operations Research**, v. 63, n. 5, p. 511–623, 1996.
- OVERHOLTS II, D. L.; BELL, J. E.; AROSTEGUI, M. A. A location analysis approach for military maintenance scheduling with geographically dispersed service areas. **Omega**, v. 37, n. 4, p. 838–852, 2009.
- PAN, Q.; WANG, L. Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. **Omega**, v. 40, n. 2, p. 218–229, 2012.
- PILLAC, V.; GUERET, C.; MEDAGLIA, A. L. A parallel matheuristic for the technician routing and scheduling problem. **Optimization Letters**, v. 7, n. 7, p. 1525–1535, 2012.
- PINEDO, M. **Scheduling: theory, algorithms, and systems**. Springer Science & Business Media, 2012.
- PREFEITURA.SP.GOV.BR. Decreto 49.487, de 12 de maio de 2008. Disponível em: <http://www3.prefeitura.sp.gov.br/cadlem/secretarias/negocios_juridicos/cadlem/integra.asp?alt=13052008D%20494870000>. Acesso em: 2 fev. 2016.
- PREFEITURA.SP.GOV.BR. Decreto 49.800, de 23 de julho de 2008. Disponível em: <https://www3.prefeitura.sp.gov.br/caminhoes_zmrc/Documentos/Decreto4980008.pdf>. Acesso em: 2 fev. 2016.
- PROCON.SP.GOV.BR. Cadastro de reclamações fundamentadas 2014. Disponível em: <<http://www.procon.sp.gov.br/reclamacoes.asp>>. Acesso em: 2 fev. 2016.
- PROCON.SP.GOV.BR. Lei número 13.747, de 7 de outubro de 2009 (alterada pela Lei Estadual número 14.951/13). Disponível em: <<http://www.procon.sp.gov.br/texto.asp?id=2924>>. Acesso em: 10 fev. 2016.
- QU, R.; BURKE, E. K.; MCCOLLUM, B. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. **European Journal of Operational Research**, v. 198, n. 2, p. 392–404, 2009.
- RADHI, H. E.; BARRANS, S. M. Comparison between Multiobjective Population-Based Algorithms in Mechanical Problem. **Applied Mechanics and Materials**, v. 110, p. 2383–2389, 2012.
- RAHMATI, S. H. A.; ZANDIEH, M.; YAZDANI, M. Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem. **The International Journal of Advanced Manufacturing Technology**, v. 64, n. 5-8, p. 915–932, 2013.
- REIS, R.; RITT, M.; BURIOL, L. S.; RESENDE, M. G. C. A biased random-key genetic algorithm for OSPF and DEFT routing to minimize network congestion. **International Transactions in Operational Research**, v. 18, n. 3, p. 401–423, 2011.

- RESENDE, M. G. C.; TOSO, R. F.; GONÇALVES, J. F.; SILVA, R. M. A. A biased random-key genetic algorithm for the Steiner triple covering problem. **Optimization Letters**, v. 6, n. 4, p. 605–619, 2012.
- RONCONI, D. P. A note on constructive heuristics for the flowshop problem with blocking. **International Journal of Production Economics**, v. 87, n. 1, p. 39–48, 2004.
- RONCONI, D. P.; HENRIQUES, L. R. S. Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. **Omega**, v. 37, n. 2, p. 272–281, 2009.
- ROPKE, S.; PISINGER, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. **Transportation science**, v. 40, n. 4, p. 455–472, 2006.
- ROQUE, L. A. C.; FONTES, D. B. M. M.; FONTES, F. A. C. C. BRKGA Adapted to Multiobjective Unit Commitment-Solving Pareto Frontier for UC Multiobjective Problem using BRKGA, SPEA2, NPGA, and NSGA II Techniques. In: ICORES, p. 64–72, 2012.
- SAMANLIOGLU, F.; FERRELL JR., W. G.; KURZ, M. E. A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem. **Computers & Industrial Engineering**, v. 55, n. 2, p. 439–449, 2008.
- SAWIK, T. A lexicographic approach to bi-objective scheduling of single-period orders in make-to-order manufacturing. **European journal of operational research**, v. 180, n. 3, p. 1060–1075, 2007.
- SHAHNAZARI-SHAHREZAEI, P.; TAVAKKOLI-MOGHADDAM, R.; KAZEMIPOOR, H. Solving a new fuzzy multi-objective model for a multi-skilled manpower scheduling problem by particle swarm optimization and elite tabu search. **The International Journal of Advanced Manufacturing Technology**, v. 64, n. 9-12, p. 1517–1540, 2013.
- SILVA JUNIOR, I. C.; CARNEIRO JUNIOR, S.; OLIVEIRA, E. J.; PEREIRA, J. L. R.; GARCIA, P. A. N. Metodologia heurística construtiva para alocação de bancos de capacitores em sistemas de distribuição de energia elétrica. **Sba: Controle & Automação Sociedade Brasileira de Automatica**, v. 19, n. 1, p. 72–82, 2008.
- SOBREIRO, V. A.; NAGANO, M. S. Proposta de uma heurística construtiva baseada na TOC para definição de mix de produção. **Produção**, v. 23, n. 3, p. 468–477, 2013.
- SOLOMON, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. **Operations research**, v. 35, n. 2, p. 254–265, 1987.
- TAILLARD, E. Benchmarks for basic scheduling problems. **European Journal of Operational Research**, v. 64, n. 2, p. 278–285, 1993.
- TANG, H.; MILLER-HOOKS, E.; TOMASTIK, R. Scheduling technicians for planned maintenance of geographically distributed equipment. **Transportation Research Part E: Logistics and Transportation Review**, v. 43, n. 5, p. 591–609, 2007.
- TANGPATTANAKUL, P.; JOZEFOWIEZ, N.; LOPEZ, P. A multi-objective local search heuristic for scheduling Earth observations taken by an agile satellite. **European Journal of Operational Research**, v. 245, n. 2, p. 542–554, 2015.
- THOMAS, J.; CHAUDHARI, N. S. Hybrid approach for 2d strip packing problem using genetic algorithm. **Advances in Computational Intelligence**, p. 566–574, 2013.
- TKINDT, V.; BILLAUT, J. C. **Multicriteria scheduling: theory, models and**

algorithms. Springer Science & Business Media, 2006.

TSANG, E.; VOUDOURIS, C. Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. **Operations Research Letters**, v. 20, n. 3, p. 119–127, 1997.

ULUNGU, E. L.; TEGHEM, J. F. P. H.; FORTEMPS, P. H.; TUYTTENS, D. MOSA method: a tool for solving multiobjective combinatorial optimization problems. **Journal of Multicriteria Decision Analysis**, v. 8, n. 4, p. 221, 1999.

VALENTE, J.; GONÇALVES, J. F. A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties. **Computers & Operations Research**, v. 36, n. 10, p. 2707–2715, 2009.

VALENTE, J. M. S.; MOREIRA, M. R. A.; SINGH, A.; ALVES, R. A. F. S. Genetic algorithms for single machine scheduling with quadratic earliness and tardiness costs. **The International Journal of Advanced Manufacturing Technology**, v. 54, n. 1-4, p. 251–265, 2011.

VIEIRA, G. E.; RIBAS, P. C. Fractional factorial analysis to the configuration of simulated annealing applied to the multi-objective optimization of master production scheduling problems. **International Journal of Production Research**, v. 46, n. 11, p. 3007–3026, 2008.

WEISSTEIN, E. W. **CRC Concise Encyclopedia of Mathematics.** CRC press, 2010.

XIANG, Y.; ZHOU, Y.; LIU, H. An elitism based multi-objective artificial bee colony algorithm. **European Journal of Operational Research**, v. 245, n. 1, p. 168–193, 2015.

XIAO, J.; WU, Z.; HONG, X. X.; TANG, J. C.; TANG, Y. Integration of electromagnetism with multi-objective evolutionary algorithms for RCPSP. **European Journal of Operational Research**, v. 251, n. 1, p. 22–35, 2016.

XU, J.; CHIU, S. Y. Effective heuristic procedures for a field technician scheduling problem. **Journal of Heuristics**, v. 7, n. 5, p. 495–509, 2001.

YAGMAHAN, B.; YENISEY, M. M. Ant colony optimization for multi-objective flow shop scheduling problem. **Computers & Industrial Engineering**, v. 54, n. 3, p. 411–420, 2008.

YANG, S.; LI, M.; LIU, X.; ZHENG, J. A grid-based evolutionary algorithm for many-objective optimization. **IEEE Transactions on Evolutionary Computation**, v. 17, n. 5, p. 721–736, 2013.

ZITZLER, E.; DEB, K.; THIELE, L. Comparison of multiobjective evolutionary algorithms: Empirical results. **Evolutionary Computation**, v. 8, n. 2, p. 173–195, 2000.

ZITZLER, E.; KNOWLES, J.; THIELE, L. Quality Assessment of Pareto Set Approximations. In: JÜRGEN, B.; DEB, K.; MIETTINEN, K.; SŁOWINSKI, R. **Multiobjective Optimization Interactive and Evolutionary Approaches.** Berlin Heidelberg, Springer-Verlag, 2008. p. 373–404.

ZITZLER, E.; LAUMANN, M.; BLEULER, S. A tutorial on evolutionary multiobjective optimization. In: **Metaheuristics for Multiobjective Optimisation.** Springer Berlin Heidelberg, p. 3–37, 2004.

ZITZLER, E.; LAUMANN, M.; THIELE, L. **SPEA2: improving the strength Pareto evolutionary algorithm**. Swiss Federal Institute Technology: Zurich, Switzerland; 2001.

ZITZLER, E.; THIELE, L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. **Evolutionary Computation**, v. 3, n. 4, p. 257–271, 1999.

ZITZLER, E.; THIELE, L.; LAUMANN, M.; FONSECA, C. M.; DA FONSECA, V. G. Performance assessment of multiobjective optimizers: an analysis and review. **Evolutionary Computation**, v. 7, n. 2, p. 117–132, 2003.