



CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E
COMPUTACIONAL

Algoritmos para o Problema Capacitado de Localização de Facilidades com Fonte Única e sua Variante Robusta

Guilherme Barbosa de Almeida

Belo Horizonte
Junho de 2026

Guilherme Barbosa de Almeida

Algoritmos para o problema capacitado de localização de facilidades com fonte única e sua variante robusta

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG.

Área de Concentração: Modelagem Matemática e Computacional

Linha de Pesquisa: Sistemas Inteligentes

Orientador: Prof. Dr. Marcone Jamilson Freitas Souza

Coorientadores: Profa. Dra. Elisangela Martins de Sá e Prof. Dr. Sérgio Ricardo de Souza

Belo Horizonte - MG
Junho de 2026

A447a	Almeida, Guilherme Barbosa de Algoritmos para o problema capacitado de localização de facilidades com fonte única e sua variante robusta / Guilherme Barbosa de Almeida. – 2026. 150 f.
	Tese (doutorado) apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional. Orientador: Marcone Jamilson Freitas Souza. Coorientadora: Elisangela Martins de Sá. Coorientador: Sérgio Ricardo de Souza. Tese (doutorado) – Centro Federal de Educação Tecnológica de Minas Gerais.
	1. Transporte – Modelos matemáticos – Teses. 2. Otimização Robusta – Teses. 3. Resolução de problemas – Teses. 4. Metaheurísticas – Teses. 5. Otimização discreta – Teses. 6. Análise de algoritmos e complexidade de problemas – Teses. I. Souza, Marcone Jamilson Freitas. II. Sá, Elisangela Martins de. III. Souza, Sérgio Ricardo de. IV. Centro Federal de Educação Tecnológica de Minas Gerais. V. Título.
	CDD 519.7



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

**“ALGORITMOS PARA O PROBLEMA CAPACITADO DE
LOCALIZAÇÃO DE FACILIDADES COM FONTE ÚNICA E SUA
VARIANTE ROBUSTA”.**

Tese de Doutorado apresentada por **Guilherme Barbosa de Almeida**, em 29 de junho de 2026, ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, e aprovada pela banca examinadora constituída pelos professores:



Documento assinado digitalmente
MARCONE JAMILSON FREITAS SOUZA
Data: 07/07/2026 11:36:09-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Marcone Jamilson Freitas Souza (Orientador)
Universidade Federal de Ouro Preto



Documento assinado digitalmente
SERGIO RICARDO DE SOUZA
Data: 15/07/2026 10:03:56-0300
Verifique em <https://validar.iti.gov.br>

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



Documento assinado digitalmente
ELISANGELA MARTINS DE SA
Data: 08/07/2026 11:38:10-0300
Verifique em <https://validar.iti.gov.br>

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



Documento assinado digitalmente
REINALDO MORABITO NETO
Data: 07/07/2026 16:43:28-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Reinaldo Morabito Neto
Universidade Federal de São Carlos



Documento assinado digitalmente
BRUNO PETRATO BRUCK
Data: 07/07/2026 17:00:53-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Bruno Petrato Bruck
Universidade Federal da Paraíba



Documento assinado digitalmente
DENIS EMANUEL DA COSTA VARGAS
Data: 08/07/2026 10:21:10-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Dênis Emanuel da Costa Vargas
Centro Federal de Educação Tecnológica de Minas Gerais



Documento assinado digitalmente
GUSTAVO CAMPOS MENEZES
Data: 08/07/2026 11:31:04-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Gustavo Campos Menezes
Centro Federal de Educação Tecnológica de Minas Gerais

Visto e permitida à impressão,

Prof. Dr. Alisson Marques da Silva
Coordenador do Programa de Pós-Graduação Stricto Sensu em
Modelagem Matemática e Computacional

Dedico este trabalho à minha esposa e ao
meu filho, fontes diárias de motivação e
inspiração.

Agradecimentos

A realização desta tese não teria sido possível sem o apoio, incentivo e contribuição de diversas pessoas, às quais expresso minha sincera gratidão.

Agradeço, primeiramente, à minha família, pelo apoio incondicional, pela compreensão nos momentos de ausência e pelo incentivo constante ao longo desta trajetória.

Agradeço ao meu orientador, Prof. Dr. Marcone Jamilson Freitas Souza, e aos coorientadores, Profa. Dra. Elisangela Martins e Prof. Dr. Sérgio Ricardo de Souza, pela orientação, disponibilidade, confiança e pelos ensinamentos transmitidos durante o desenvolvimento deste trabalho.

Agradeço aos professores que lecionaram as disciplinas cursadas no programa pelos ensinamentos que foram fundamentais para o desenvolvimento desta tese.

Agradeço aos colegas de doutorado pelas trocas de experiências, discussões e pela convivência acadêmica ao longo desta caminhada.

Agradeço ao CEFET-MG pela política de afastamento para curso de doutorado, fundamental para que eu pudesse me dedicar integralmente a esta pesquisa.

Por fim, agradeço a todos que, de alguma forma, fizeram parte desta caminhada e contribuíram para a concretização deste trabalho.

Resumo

Esta tese aborda o Problema Capacitado de Localização de Facilidades com Fonte (SSCFLP) e sua variante robusta (RSSCFLP). O SSCFLP consiste em determinar locais para a abertura de facilidades capacitadas para atender às demandas dos clientes, de modo que cada cliente seja atendido por uma única facilidade. Já o RSSCFLP estende o SSCFLP ao considerar incertezas nos custos de alocação e abertura, nas demandas e nas capacidades. Para formular o RSSCFLP, adota-se a abordagem robusta de Bertsimas e Sim. Para resolver os problemas, são propostos três algoritmos: HILS e NCS, para o SSCFLP, e o B&B-GVNS, para o RSSCFLP. O HILS combina a resolução de subproblemas com procedimentos baseados em ILS para obter soluções de qualidade. O NCS combina um método de planos de corte, uma heurística baseada em busca local e um procedimento *cut-and-solve* para encontrar a solução ótima do problema. O B&B-GVNS combina um procedimento de planos de corte, uma heurística baseada em GVNS e um algoritmo *branch-and-bound* aprimorado com heurísticas. Experimentos computacionais indicam que o HILS é competitivo em relação aos algoritmos do estado da arte, produzindo soluções com valores de gap ligeiramente menores em conjuntos mais desafiadores. Os resultados obtidos com o NCS mostraram que ele encontrou mais soluções ótimas ou reduziu o tempo necessário para obtê-las em 85% dos subconjuntos analisados, quando comparado ao algoritmo do estado da arte, e em 73% dos subconjuntos, quando comparado ao CPLEX. Além disso, em vários subconjuntos, o NCS foi mais de 100 vezes mais rápido do que o CPLEX e mais de 10 vezes mais rápido do que o algoritmo do estado da arte para encontrar um número maior ou igual de soluções ótimas. Por fim, o B&B-GVNS apresentou desempenho superior ao do CPLEX em todos os conjuntos testados, com reduções no gap superiores a 20% e 25% nos dois conjuntos mais desafiadores. Esses resultados evidenciam que os algoritmos desenvolvidos nesta tese constituem contribuições metodológicas relevantes para a resolução de problemas capacitados de localização de facilidades e de sua variante robusta.

Palavras-chave: Problema Capacitado de Localização de Facilidades com Fonte Única, Otimização Robusta, *Cut-and-Solve*, Método de planos de corte, *Iterated Local Search*.

Abstract

This thesis addresses the Single-Source Capacitated Facility Location Problem (SSCFLP) and its robust variant (RSSCFLP). The SSCFLP involves determining locations for opening capacitated facilities to meet customer demands, ensuring that each customer is assigned to exactly one facility. The RSSCFLP extends the SSCFLP by considering uncertainty in allocation and opening costs, demands, and capacities. To formulate the RSSCFLP, the robust optimization approach proposed by Dimitris Bertsimas and Melvyn Sim is adopted. To solve these problems, three algorithms are proposed: HILS and NCS for the SSCFLP, and B&B-GVNS for the RSSCFLP. HILS combines the solution of subproblems with procedures based on Iterated Local Search to obtain high-quality solutions. NCS combines a cutting-plane method, a local-search-based heuristic, and a cut-and-solve procedure to find optimal solutions for the problem. B&B-GVNS combines a cutting-plane procedure, a heuristic based on General Variable Neighborhood Search, and a branch-and-bound algorithm enhanced with heuristics. Computational experiments indicate that HILS is competitive with state-of-the-art algorithms, producing solutions with slightly smaller gap values on more challenging instances. The results obtained with NCS showed that it found more optimal solutions or reduced the time required to obtain them in 85% of the analyzed subsets when compared with the state-of-the-art algorithm, and in 73% of the subsets when compared with CPLEX. Moreover, in several subsets, NCS was more than 100 times faster than CPLEX and more than 10 times faster than the state-of-the-art algorithm in obtaining an equal or greater number of optimal solutions. Finally, B&B-GVNS outperformed CPLEX on all tested sets, achieving reductions in gap greater than 20% and 25% on the two most challenging sets. These results demonstrate that the algorithms developed in this thesis constitute relevant methodological contributions to the solution of capacitated facility location problems and its robust variant.

Keywords: Single source capacitated facility location problem, Robust optimization, Cut-and-solve, Cutting-plane method, Iterated Local Search.

Lista de Tabelas

4.1	Visão geral da revisão da literatura: principais abordagens para a resolução do SSCFLP.	62
6.1	Conjuntos de instâncias para testes	96
6.2	Instâncias para treinamento do HILS e suas características.	98
6.3	Resultado da calibração de parâmetros do HILS	98
6.4	Resultados comparativos dos algoritmos HILS, KS, HMEH e CM no conjunto TB4.	100
6.5	Resultados comparativos dos algoritmos HILS e KS no conjunto TB5	100
6.6	Resultados comparativos dos algoritmos HILS e CM no conjunto TB5.	101
6.7	Resultados comparativos dos algoritmos HILS e HMEH no conjunto TB5.	101
6.8	Novos BKVs fornecidos pelo HILS.	102
6.9	Resultado da análise do procedimento de identificação de locais promissores do HILS	103
6.10	Variantes do algoritmo HILS obtidas removendo uma componente . .	104
6.11	Resultado da avaliação de componentes do HILS	105
6.12	Resultado da avaliação do uso de diferentes versões do CPLEX no HILS	107
6.13	Resultados comparativos dos procedimentos PC+PISxy e PC+PISx.	109
6.14	Resultados da avaliação do efeito individual de cada tipo de corte . .	110
6.15	Resultado da calibração de parâmetros do NCS	112
6.16	Resumo dos resultados obtidos pelo algoritmo NCS.	112
6.17	Novos valores ótimos encontrados pelo algoritmo NCS.	112
6.18	Novos BKVs encontrados pelo algoritmo NCS	113
6.19	Resultados comparativos entre NCS-2 e NCS	114
6.20	Resultados comparativos entre NCS, TP-ALG e o CPLEX	117
6.21	Resultados comparativos de LI_0 e LS_0 fornecidos pelos algoritmos NCS e TP-ALG	118
6.22	Resultados comparativos do B&B-GVNS e do CPLEX	122
6.23	Resultado da avaliação de componentes do B&B-GVNS	124
6.24	Análise de robustez do modelo do RSSCFLP	125
6.25	Resultado da avaliação do efeito do incremento de cada parâmetro . .	126
6.26	Custo marginal associado ao aumento das taxas de robustez	126
A.1	Resultados detalhados do HILS no conjunto TB4.	142
A.2	Resultados detalhados do HILS no conjunto TB5	142
A.3	Resultados detalhados do experimento com o NCS	144
A.4	Resultados detalhados do experimento com o H-ALG	148

Lista de Figuras

2.1	Ilustração de uma iteração do ILS	28
3.1	Comparação da solução sem e com a condição de fonte única.	54
5.1	Ramificação da árvore de busca do algoritmo NCS.	86
5.2	Ilustração da perturbação p_4 do B&B-GVNS.	92
6.1	FRAs para os valores de TotalIter e IterOpt.	115

Lista de Siglas

B&B *Branch-and-Bound.*

B&C *Branch-and-Cut.*

B&P *Branch-and-Price.*

BKV *Best Known Value.*

CFLP *Capacitated Facility Location Problem.*

CF Corte de Fenchel.

CI *Cover Inequality.*

CM *Corridor Method.*

CS *Cut-and-Solve.*

FRA Frequência Relativa Acumulada.

GAP *Generalized Assignment Problem.*

GRASP *Greedy Randomized Adaptive Search Procedure.*

GVNS *General Variable Neighborhood Search.*

HMEM *Hypergraph Multi-Exchange Heuristic.*

HILS *Hybrid Iterated Local Search.*

ILS *Iterated Local Search.*

KS *Kernel Search.*

LCIs *Lifted Cover Inequalities.*

LI Limitante inferior.

LPI *Lifted Pack Inequality.*

LS Limitante superior.

NCS *New Cut-and-Solve.*

PC Planos de corte.

PD Problema denso.

PE Problema esparso.

PIS *Partial Integrality Strategy.*

PLFs Problemas de Localização de Facilidades.

PM Programação matemática.

RLG Relaxação lagrangeana.

RLI Relaxação linear.

RSSCFLP *Robust Single-Source Capacitated Facility Location Problem.*

SSCFLP *Single-Source Capacitated Facility Location Problem.*

TP-ALG *Three-Phase Algorithm.*

TRCABG Taxa de Robustez do Custo de Abertura Garantido.

TRCALG Taxa de Robustez do Custo de Alocação Garantido.

TV Taxa de viabilidade.

VND *Variable Neighborhood Descent.*

VNS *Variable Neighborhood Search.*

Lista de Algoritmos

1	Esquema básico do VND (Hansen & Mladenović, 2003)	22
2	Esquema do <i>Basic</i> VNS (Hansen & Mladenović, 2005)	27
3	ILS (Lourenço et al., 2003)	29
4	Algoritmo <i>branch-and-bound</i> para minimização	32
5	Esquema <i>cut-and-solve</i>	33
6	Algoritmo de planos de corte	37
7	HILS	68
8	Algoritmo ILS para os procedimentos ILS-I e ILS-II	72
9	Procedimento PC+PIS	75
10	Separação exata de CIs (Yang et al., 2012)	77
11	NCS	84
12	Busca Local – Fase I	87
13	Busca Local – Fase II	88
14	Esquema do B&B-GVNS	89
15	Procedimento de planos de corte do B&B-GVNS	90
16	VND do B&B-GVNS	91
17	GVNS-I	92
18	GVNS-II	92

Sumário

1	Introdução	13
1.1	Apresentação geral	13
1.2	Justificativa	15
1.3	Objetivos	15
1.3.1	Objetivos específicos	15
1.4	Metodologia	16
1.5	Contribuições	17
1.6	Trabalhos derivados desta tese	18
1.7	Organização do trabalho	18
2	Fundamentação Teórica	20
2.1	Heurísticas	20
2.1.1	Heurísticas baseadas em busca local	21
2.1.2	Meta-heurísticas	23
2.1.3	<i>Variable Neighborhood Search</i>	26
2.1.4	Iterated local search	28
2.2	Matheurísticas	29
2.3	Métodos exatos	30
2.3.1	<i>Branch-and-bound e branch-and-cut</i>	31
2.3.2	<i>Cut-and-solve</i>	32
2.4	Método de planos de corte	34
2.4.1	Caracterização de poliedros por suas facetas	34
2.4.2	Caracterização poliédrica de problemas inteiros	36
2.4.3	Algoritmo de planos de corte	37
2.4.4	Fortalecendo desigualdades válidas em problemas binários	37
2.4.5	<i>Lifted cover inequalities</i>	38
2.5	Dualidade em programação linear	40
2.6	Otimização robusta	41
2.6.1	Ideias gerais	41
2.6.2	Otimização linear robusta	43
2.6.3	Formulação robusta de problemas de otimização discretos	46
3	Formulações matemáticas	51
3.1	Descrição e formulação do SSCFLP	51
3.2	Descrição e formulação do RSSCFLP	55

4	Trabalhos Relacionados	59
4.1	Trabalhos relacionados ao SSCFLP	59
4.2	Trabalhos relacionados ao RSSCFLP	62
4.3	Trabalhos que inspiraram os métodos propostos	64
4.3.1	Algoritmo KS	64
4.3.2	Algoritmo MH	65
4.3.3	Algoritmo TP-ALG	66
5	Métodos de solução	68
5.1	Algoritmo HILS	68
5.1.1	Identificação de locais promissores	70
5.1.2	Construção da solução inicial e procedimento de abertura de facilidades	70
5.1.3	Procedimentos ILS	71
5.2	Algoritmo NCS	73
5.2.1	Uma relaxação parcial forte para o SSCFLP	74
5.2.2	Descrição do NCS	83
5.3	Algoritmo B&B-GVNS	88
6	Experimentos Computacionais	94
6.1	Conjuntos de instâncias de teste	94
6.2	Experimentos com o algoritmo HILS	96
6.2.1	Calibração de parâmetros	97
6.2.2	Resultados e Análises	98
6.3	Experimentos com o algoritmo NCS	107
6.3.1	Experimentos com o procedimento PC+PIS	108
6.3.2	Experimentos com o algoritmo NCS completo	110
6.4	Experimentos com o algoritmo B&B-GVNS	119
6.4.1	Resultados e análises	121
6.4.2	Avaliação do modelo do RSSCFLP	124
7	Conclusões e Sugestões de Trabalhos Futuros	127
7.1	Conclusões	127
7.2	Propostas de trabalhos futuros	128
	Referências	130
A		142
A.1	Resultados detalhados dos experimentos com o algoritmo HILS	142
A.2	Resultados detalhados dos experimentos com o algoritmo NCS	144
A.3	Resultados detalhados dos experimentos com o algoritmo H-ALG . . .	148

Capítulo 1

Introdução

1.1 Apresentação geral

Problemas de localização de facilidades (PLFs) consistem em determinar locais para a instalação (abertura) de facilidades, visando atender às demandas de um conjunto de clientes. Na literatura, o termo “facilidades” engloba instalações físicas, como fábricas, centros de distribuição, hospitais e escolas, bem como equipamentos e unidades de serviço, como antenas, ambulâncias, viaturas e torres de transmissão. Por sua vez, o termo “clientes” abrange indivíduos, regiões, empresas ou pontos de demanda que necessitam ser atendidos pelas facilidades estabelecidas.

As aplicações dos PLFs abrangem diversas áreas. Na logística de transportes, destacam-se aplicações como a localização de garagens em sistemas de tráfego urbano (Ball et al., 1984). No planejamento de redes logísticas, incluem-se, por exemplo, a localização de armazéns (Beasley, 1988) e o planejamento de cadeias de suprimentos (Melo et al., 2006). Na área da saúde, há aplicações relacionadas à localização de serviços de emergência (Pirkul & Schilling, 1988), enquanto, na segurança pública, destacam-se aplicações como a alocação de unidades do corpo de bombeiros (Plane & Hendrick, 1977). Os PLFs também são aplicados em redes de telecomunicações (Gollowitzer & Ljubić, 2011), cadeias humanitárias (Boonmee et al., 2017), problemas de distritamento (Novaes et al., 2009), gestão de resíduos urbanos (Tirkolaee et al., 2020) e no planejamento de infraestrutura para novas tecnologias (Kchaou-Boujelben, 2021).

Na literatura, diversos modelos de PLFs consideram a condição de fonte única. Essa condição impõe que a demanda de cada cliente seja integralmente atendida por uma única facilidade. Por exemplo, Geoffrion & Graves (1974) propuseram um modelo para um sistema de distribuição de produtos no qual a demanda de cada região por um determinado produto deve ser totalmente satisfeita por um único centro de

distribuição. Outro exemplo pode ser encontrado em [Kong \(2021\)](#), que abordaram um problema de alocação de estudantes em escolas primárias, impondo que os estudantes de uma mesma microrregião fossem alocados à mesma escola. Embora a divisão dos estudantes entre diferentes escolas seja possível na prática, a condição de fonte única pode ser adequada quando a demanda é agregada em microrregiões, especialmente por favorecer a organização do transporte escolar e a definição das áreas de atendimento, que podem envolver aspectos administrativos e políticos.

Decisões de localização de facilidades geralmente envolvem altos custos e são difíceis de reverter. Além disso, em aplicações reais, os valores dos parâmetros do problema frequentemente sofrem variações e imprecisões, o que pode resultar em custos superiores aos previstos ou até mesmo na inviabilidade operacional. Diante disso, tem crescido o interesse em modelos de PLFs que incorporam incertezas em seus parâmetros ([Snyder, 2006](#)).

Esta tese aborda o Problema Capacitado de Localização de Facilidades com Fonte Única (*Single Source Capacitated Facility Location Problem* - SSCFLP) e sua variante robusta (*Robust Single Source Capacitated Facility Location Problem* - RSSCFLP). O SSCFLP é um PLF clássico que considera: (i) custos de abertura de facilidades; (ii) limites de capacidade das facilidades; e (iii) a condição de fonte única para a alocação de clientes. Já o RSSCFLP incorpora incertezas nos parâmetros do SSCFLP: demandas, custos de alocação, custos de abertura e capacidades.

O objetivo desta tese é desenvolver métodos de solução para esses problemas. Para isso, são desenvolvidos três algoritmos. O primeiro é um algoritmo heurístico para o SSCFLP, denominado HILS, acrônimo de *Hybrid Iterated Local Search*. O HILS combina procedimentos baseados na meta-heurística *Iterated Local Search* (ILS) com a resolução de subproblemas. O segundo, denominado *New Cut-and-Solve* (NCS), é um algoritmo exato para o SSCFLP que aprimora abordagens baseadas no *framework cut-and-solve*, incorporando novos procedimentos heurísticos e de planos de corte. O terceiro é o B&B-GVNS. O B&B-GVNS é um algoritmo exato para o RSSCFLP, estruturado em três fases: na primeira, aplica-se um procedimento de planos de corte; na segunda, empregam-se procedimentos baseados em *General Variable Neighborhood Search* (GVNS); e, na terceira, utiliza-se um procedimento baseado em *branch-and-bound* (B&B) aprimorado por procedimentos heurísticos.

O restante desta introdução está organizado da seguinte forma. As Seções [1.2](#), [1.3](#), [1.4](#), [1.5](#), [1.6](#) e [1.7](#) apresentam, respectivamente, a justificativa, os objetivos, a metodologia, as contribuições, os trabalhos derivados e a organização desta tese.

1.2 Justificativa

O SSCFLP tem aplicações em diversos problemas do mundo real. Por se tratar de um problema NP-difícil (Delmaire et al., 1999), a literatura geralmente propõe resolvê-lo por meio de métodos heurísticos. Os algoritmos heurísticos do estado da arte são capazes de encontrar soluções de boa qualidade para instâncias com até 1000 clientes e 1000 locais potenciais. No entanto, experimentos realizados em instâncias mais desafiadoras indicam que esses métodos demandam tempo computacional elevado e, ainda assim, nem sempre produzem soluções de alta qualidade.

Ao mesmo tempo, as aplicações do SSCFLP estão frequentemente associadas a decisões estratégicas, que dificilmente podem ser revertidas. Por isso, se mal planejadas, podem acarretar prejuízos elevados. Nesse contexto, a obtenção de soluções ótimas por métodos exatos torna-se relevante, mesmo que tais métodos demandem alto custo computacional. Experimentos com algoritmos exatos do estado da arte mostram que esses algoritmos encontram soluções ótimas para instâncias com até 400 clientes e 80 locais. Entretanto, eles não foram capazes de encontrar soluções ótimas para várias instâncias com 300 clientes e 60 locais dentro do limite de tempo de 50 000 segundos.

Paralelamente, em aplicações reais, os parâmetros do SSCFLP estão sujeitos a variações e imprecisões, o que pode resultar em custos acima do previsto ou até mesmo na inviabilidade das soluções, caso as incertezas nos parâmetros não sejam consideradas. Na literatura, quando a incerteza é abordada, ela costuma restringir-se às demandas dos clientes. Até onde se sabe, não há modelos nem algoritmos que considerem incertezas simultâneas em todos os tipos de parâmetros do problema.

O cenário exposto evidencia uma ampla margem para contribuições relativas a métodos de solução do SSCFLP e do RSSCFLP, o que motiva o desenvolvimento desta tese.

1.3 Objetivos

O objetivo geral desta tese é desenvolver métodos de solução para o SSCFLP e o RSSCFLP.

1.3.1 Objetivos específicos

Para atingir o objetivo geral, estabelecem-se os seguintes objetivos específicos:

- Revisar e analisar a literatura sobre métodos de solução para o SSCFLP e o RSSCFLP;

- Formular o RSSCFLP com base na abordagem de otimização robusta de Bertsimas & Sim (2003);
- Desenvolver heurísticas capazes de obter soluções viáveis de alta qualidade para o SSCFLP e o RSSCFLP;
- Desenvolver métodos de planos de corte para fortalecer as formulações do SSCFLP e do RSSCFLP;
- Desenvolver um algoritmo híbrido para o SSCFLP, combinando meta-heurísticas com a resolução de modelos matemáticos;
- Desenvolver um algoritmo exato para o SSCFLP baseado no *framework cut-and-solve*, incorporando novos procedimentos de planos de corte e heurísticas;
- Desenvolver um algoritmo para resolução do RSSCFLP que integre técnicas de fortalecimento de formulação, meta-heurísticas e métodos exatos;
- Avaliar computacionalmente os algoritmos propostos em conjuntos de instâncias *benchmark* da literatura;
- Comparar o desempenho dos métodos desenvolvidos com os de algoritmos estado da arte e os de *solvers*;

1.4 Metodologia

Para atingir os objetivos propostos, foram definidas as seguintes etapas metodológicas:

- **Revisão da literatura:** realiza-se uma revisão sistemática de trabalhos relacionados ao SSCFLP e suas variantes que consideram incerteza. Essa etapa inclui a classificação dos métodos existentes e a análise crítica dos algoritmos considerados estado da arte;
- **Formulação dos problemas:** são desenvolvidas formulações matemáticas para o SSCFLP e o RSSCFLP, definindo-se conjuntos, parâmetros, variáveis de decisão, função objetivo e restrições. Para o RSSCFLP, a modelagem é baseada na abordagem de otimização robusta de Bertsimas & Sim (2003), permitindo sua formulação como um problema de programação inteira mista;
- **Desenvolvimento de métodos de solução:** são propostos algoritmos heurísticos e exatos para o SSCFLP e o RSSCFLP, combinando diferentes técnicas

de otimização, como meta-heurísticas, matheurísticas, procedimentos de planos de corte e busca baseada em árvores binárias.

- **Experimentação computacional:** são realizados experimentos computacionais em conjuntos de instâncias *benchmark*, com o objetivo de avaliar o desempenho dos algoritmos propostos em termos de qualidade de solução e tempo computacional;
- **Análise de resultados:** os resultados obtidos são comparados com os reportados na literatura, incluindo métodos do estado da arte e *solvers* de programação matemática. São conduzidas análises quantitativas para identificar as principais contribuições dos métodos propostos;
- **Divulgação dos resultados:** os resultados da pesquisa são consolidados nesta tese e submetidos à publicação em periódicos científicos qualificados.

1.5 Contribuições

As principais contribuições desta tese são apresentadas a seguir:

- Desenvolvimento de dois procedimentos baseados na meta-heurística ILS para a solução do SSCFLP;
- Proposição de uma matheurística para a solução do SSCFLP, denominada HILS, que integra procedimentos ILS com a resolução de modelos matemáticos;
- Desenvolvimento de uma nova heurística gulosa para a geração de cortes de cobertura associados às restrições de capacidade do SSCFLP;
- Utilização de dois tipos de planos de corte para problemas da mochila disponíveis na literatura, mas que, até então, não foram explorados no contexto do SSCFLP. Eles são: *Lifted Pack Inequalities* (LPis), gerados por uma função superaditiva apresentada em [Atamturk \(2005\)](#), e *Lifted Cover Inequalities* (LCIs), gerados por uma função superaditiva apresentada em [Letchford & Souli \(2019\)](#);
- Condução de uma avaliação sistemática do impacto de diferentes famílias de planos de corte sobre a qualidade dos limitantes inferiores produzidos por um procedimento que combina planos de corte com a resolução de uma relaxação linear parcial do SSCFLP;
- Desenvolvimento de um novo procedimento de busca local determinística para o SSCFLP baseada em técnicas de clusterização;

- Desenvolvimento de um procedimento de planos de corte para o RSSCFLP por meio da adaptação de procedimentos originalmente propostos para o SSCFLP;
- Desenvolvimento de uma heurística determinística baseada em GVNS para a obtenção de soluções para o RSSCFLP;
- Proposição de um algoritmo para a solução do RSSCFLP, que integra procedimentos de planos de corte, heurísticas e um procedimento baseado em *Branch-and-Bound*, aprimorado com a incorporação de heurísticas;
- Disponibilização pública dos códigos-fontes dos métodos de solução desenvolvidos.

1.6 Trabalhos derivados desta tese

O estudo realizado ao longo do desenvolvimento desta tese deu origem aos seguintes trabalhos científicos:

- de Almeida, G. B., de Sá, E. M., de Souza, S. R., & Souza, M. J. F. (2021). Um algoritmo ILS para o problema de localização de pontos de consolidação. Anais do LIII Simpósio Brasileiro de Pesquisa Operacional (SBPO), João Pessoa, Brasil. doi: 10.59254/sbpo-2021-131653.
- de Almeida, G. B., de Sá, E. M., de Souza, S. R., & Souza, M. J. F. (2023). A hybrid iterated local search matheuristic for large-scale single source capacitated facility location problems. *Journal of Heuristics*, 30, 1–28. doi: 10.1007/s10732-023-09524-9.
- de Almeida, G. B., de Sá, E. M., de Souza, S. R., & Souza, M. J. F. (2025). A new cut-and-solve-based approach for the single-source capacitated facility location problem. Artigo submetido ao periódico *European Journal of Operational Research*.
- de Almeida, G. B., de Sá, E. M., de Souza, S. R., & Souza, M. J. F. (2025). A Hybrid Algorithm for the Robust Single Source Capacitated Facility Location Problem. Artigo a ser submetido após a defesa da tese.

1.7 Organização do trabalho

Além desta introdução, esta tese conta com mais seis capítulos. O Capítulo 2 apresenta a fundamentação teórica necessária ao desenvolvimento deste trabalho.

O Capítulo 3 apresenta as descrições do SSCFLP e do RSSCFLP, incluindo suas formulações matemáticas. No Capítulo 4, apresenta-se a revisão bibliográfica, com ênfase nos principais métodos de solução que fundamentam os algoritmos propostos. O Capítulo 5 é dedicado à descrição detalhada dos métodos desenvolvidos nesta tese. No Capítulo 6, são apresentados os experimentos computacionais realizados, bem como a análise dos resultados obtidos. Por fim, no Capítulo 7, são apresentadas as conclusões do trabalho e as sugestões para pesquisas futuras.

Capítulo 2

Fundamentação Teórica

Os principais fundamentos teóricos que embasam esta tese são apresentados neste capítulo, que está organizado em cinco seções. A Seção 2.1, apresenta a teoria relacionada aos procedimentos heurísticos propostos nesta tese. Já a Seção 2.3 aborda os fundamentos dos algoritmos exatos. Na Seção 2.4, são apresentados os conceitos teóricos associados ao método de planos de corte. A Seção 2.5 apresenta os principais resultados acerca da teoria da dualidade para problemas de programação linear. Por fim, a Seção 2.6 apresenta conceitos relacionados à otimização robusta a abordagem robusta de Bertsimas & Sim (2003) para problemas de otimização discreta.

2.1 Heurísticas

Segundo Silver et al. (1980), um método heurístico é um procedimento para resolver um problema matemático bem definido por meio de uma abordagem intuitiva, na qual a estrutura do problema pode ser interpretada e explorada de forma inteligente para obter uma solução de qualidade. Em um contexto mais moderno e relacionado à Pesquisa Operacional, Talbi (2009) define heurísticas como técnicas que buscam boas soluções para problemas com custo computacional razoável.

Segundo Hansen & Mladenović (2003), diversas instâncias de problemas de otimização que surgem em Pesquisa Operacional e em outras áreas são grandes demais para que uma solução exata seja encontrada em tempo razoável. É bem conhecido que muitos problemas de otimização são NP-difíceis. Para essa classe de problemas que não se conhece um algoritmo com número de passos polinomial no tamanho das instâncias e que encontrar tal algoritmo para qualquer um desses problemas implicaria encontrá-lo para todos os demais. Além disso, em casos em que um problema admite um algoritmo polinomial, o grau desse polinômio pode ser tão elevado que instâncias de tamanho realista não podem ser resolvidas em tempo razoável no pior

caso, e, às vezes, também no caso médio ou na maioria das vezes.

Diante desse cenário, frequentemente é necessário recorrer a heurísticas, que produzem rapidamente uma solução aproximada ou, às vezes, uma solução ótima, porém sem prova de sua otimalidade.

2.1.1 Heurísticas baseadas em busca local

Vários procedimentos heurísticos desenvolvidos nesta tese baseiam-se na busca local. A ideia básica da busca local consiste em partir de uma solução inicial e buscar melhorias sucessivas por meio da exploração de soluções vizinhas definidas por estruturas de vizinhança (Vaessens et al., 1998). Uma estrutura de vizinhança N associa a cada solução $s \in S$ um conjunto $N(s) \subseteq S$ de soluções vizinhas. Uma solução $s \in S$ é denominada um ótimo local em relação a N em um problema de minimização se não existir solução $t \in N(s)$ tal que $f(t) < f(s)$.

Para explorar uma estrutura de vizinhança N em um procedimento de busca local, as estratégias geralmente empregadas são *best improvement* e *first improvement*. A estratégia *best improvement* consiste em explorar completamente a vizinhança e selecionar o melhor vizinho que melhora o valor da função objetivo. Caso não exista tal vizinho, o procedimento é interrompido; caso contrário, a solução corrente é atualizada. A exploração completa dessa vizinhança pode ser computacionalmente custosa. Nesse contexto, uma alternativa consiste na estratégia *first improvement*, na qual os vizinhos são explorados sequencialmente e o primeiro a apresentar melhoria é imediatamente aceito.

Na próxima subseção, apresenta-se o procedimento *Variable Neighborhood Descent* (VND), utilizado como busca local nos procedimentos ILS e GVNS desenvolvidos nesta tese.

2.1.1.1 Variable Neighborhood Descent

O VND baseia-se no fato de que um ótimo local em relação a uma determinada estrutura de vizinhança não é necessariamente um ótimo local em relação a outras. Assim, o método explora diferentes estruturas de vizinhança com o objetivo de obter uma solução que seja um ótimo local em relação a todas elas. O esquema básico do VND é apresentado no Algoritmo 1.

A aplicação do VND envolve algumas decisões importantes. Primeiramente, deve-se considerar a complexidade dos movimentos associados a cada estrutura de vizinhança, uma vez que movimentos mais complexos podem aumentar significativamente o tempo computacional. Além disso, a ordem de exploração das vizinhanças pode influenciar tanto a qualidade das soluções obtidas quanto o tempo de execu-

Algoritmo 1 Esquema básico do VND ([Hansen & Mladenović, 2003](#))

```

1: Inicialização: selecione o conjunto de estruturas de vizinhança  $N_i$ , para  $i = 1, \dots, i_{\max}$ , a serem utilizadas na descida; encontre uma solução inicial  $s$  (ou aplique o procedimento a uma solução  $s$  fornecida);
2: Defina  $i \leftarrow 1$ ;
3: enquanto  $i \leq i_{\max}$  faça
4:   Exploração da vizinhança: encontre o melhor vizinho  $s'$  de  $s$  tal que  $s' \in N_i(s)$ ;
5:   se  $f(s') < f(s)$  então
6:     Atualize  $s \leftarrow s'$ ;
7:     Defina  $i \leftarrow 1$ ;
8:   senão
9:     Defina  $i \leftarrow i + 1$ ;
10:  fim se
11: fim enquanto

```

ção do algoritmo. Em geral, as vizinhanças são ordenadas de modo que estruturas associadas a pequenas modificações na solução corrente sejam exploradas antes de estruturas que promovem alterações mais significativas.

Definida a ordem de exploração das vizinhanças, [Hansen et al. \(2017\)](#) apresentam três critérios para alteração das estruturas de vizinhança ao longo do processo iterativo do VND:

- **Mudança sequencial de vizinhança** (*Sequential neighborhood change step*): se uma melhoria da solução incumbente for obtida em alguma estrutura de vizinhança, a busca é reiniciada a partir da primeira estrutura de vizinhança da lista, de acordo com a ordem previamente estabelecida. Caso contrário, a busca prossegue para a próxima estrutura de vizinhança;
- **Mudança cíclica de vizinhança** (*Cyclic neighborhood change step*): independentemente da ocorrência de melhoria na solução incumbente, a busca continua na próxima estrutura de vizinhança da lista;
- **Mudança do tipo *pipe*** (*Pipe neighborhood change step*): quando uma melhoria da solução corrente é encontrada em alguma estrutura de vizinhança, a busca continua explorando essa mesma vizinhança.

Outra possibilidade consiste em, a cada iteração, formar uma única vizinhança a partir da união das soluções vizinhas geradas por todas as estruturas de vizinhança. Caso nenhuma solução melhoradora seja encontrada nessa vizinhança unificada, o procedimento é encerrado. Caso contrário, a melhor solução encontrada torna-se a nova solução incumbente, e o processo é retomado.

2.1.2 Meta-heurísticas

O termo “meta-heurística” foi introduzido na literatura por [Glover \(1986\)](#), ao caracterizar a heurística de Busca Tabu como uma “meta-heurística” que “sobrepõe” procedimentos a outra heurística. [Sörensen & Glover \(2013\)](#) definem meta-heurísticas como *frameworks* de alto nível que fornecem estratégias independentes do problema a ser resolvido para desenvolver algoritmos heurísticos de otimização. Segundo os autores, algoritmos heurísticos desenvolvidos seguindo os *frameworks* de meta-heurísticas também são referidos na literatura como meta-heurísticas. A maioria dos *frameworks* meta-heurísticos tem origem na década de 1980 e foi proposta como alternativa ou complemento aos métodos clássicos de otimização anteriores, como o método B&B e as técnicas baseadas em programação dinâmica.

Desde então, essas abordagens têm se popularizado como estratégias promissoras para resolver problemas de otimização considerados intratáveis computacionalmente. Ao propor um estudo sobre a história das meta-heurísticas, [Sörensen et al. \(2018\)](#) atesta que elas já eram utilizadas antes da década de 1940, embora não fossem formalmente estudadas.

As meta-heurísticas foram caracterizadas por [Blum & Roli \(2003\)](#) pelas seguintes propriedades fundamentais:

- Meta-heurísticas são estratégias que orientam o processo de busca;
- Seu objetivo é explorar o espaço de soluções de forma eficiente, visando encontrar soluções ótimas ou quase ótimas;
- As técnicas que compõem algoritmos meta-heurísticos variam desde procedimentos simples de busca local até mecanismos mais complexos de aprendizado;
- Algoritmos meta-heurísticos são, em geral, aproximados e geralmente não determinísticos;
- Podem incorporar mecanismos destinados a evitar que a busca fique restrita a regiões limitadas do espaço de soluções;
- Seus conceitos fundamentais permitem uma descrição em nível abstrato;
- Meta-heurísticas não são específicas de um único problema;
- Podem utilizar conhecimento específico do domínio, incorporado na forma de heurísticas controladas por uma estratégia de nível superior;

- Meta-heurísticas mais avançadas utilizam a experiência acumulada durante a busca, geralmente representada por algum tipo de memória, para orientar o processo de exploração.

Conforme [Sörensen & Glover \(2013\)](#), as meta-heurísticas podem ser baseadas em busca local, em heurísticas construtivas ou em heurísticas baseadas em população. As meta-heurísticas baseadas em estruturas de vizinhança incluem o ILS, que utiliza um tipo de perturbação para evitar que busca fique presa em ótimos locais ([Lourenço et al., 2003](#)); Busca em Vizinhança Variável (*Variable Neighborhood Search* - VNS), que explora diferentes estruturas de vizinhanças de perturbação ([Hansen & Mladenović, 2003](#)); Busca Tabu, que explora estruturas de memória para adotar mecanismos para proibir movimentos para soluções ou para conjuntos de soluções que possuam certos atributos ([Glover, 1986](#)); Busca Local Guiada, que introduz penalidades para atributos de soluções para que a busca não fique presa em ótimos locais ([Voudouris & Tsang, 1999](#)); e Recozimento Simulado (*Simulated Annealing*), que usa uma estratégia de movimento aleatório e probabilístico baseado em um processo de recozimento de um sólido cristalino ([Kirkpatrick et al., 1983](#)).

Dentre as meta-heurísticas baseadas em população, a mais conhecida é o Algoritmo Genético, que iterativamente atualiza um conjunto de soluções, chamado de população, utilizando operadores baseados em alterações genéticas no processo evolutivo, visando gerar populações cujas soluções são de melhor qualidade ([Goldberg, 1988](#)). Há também a Busca Dispersa ([Glover, 1998](#)), que mantém um conjunto de soluções de referência, selecionadas com base em critérios de qualidade e diversidade. Novas soluções são geradas por meio de combinações estruturadas dessas soluções, com o objetivo de explorar regiões promissoras do espaço de busca e atualizar o conjunto de referência.

No campo das meta-heurísticas construtivas, destaca-se o Procedimento de Busca Gulosa Randomizada Adaptativa (*Greedy Randomized Adaptive Search Procedure* - GRASP). Ele incorpora aleatoriedade em heurísticas construtivas gulosas para diversificar o processo de construção ([Resende & Ribeiro, 2019](#)). Destaca-se também a família de algoritmos de Colônia de Formigas, em que múltiplos agentes, chamados de formigas, constroem iterativamente soluções, considerando parâmetros chamados de feromônios, que são utilizados para aumentar a probabilidade de que as soluções construídas tenham atributos de boas soluções ([Dorigo et al., 1996](#); [Dorigo & Stützle, 2019](#)).

[Blum & Roli \(2003\)](#) também propuseram classificar meta-heurísticas sob diferentes pontos de vista. A seguir, resumem-se algumas delas.

a) Inspiradas na natureza vs. não inspiradas na natureza.

Uma forma bastante intuitiva de classificar meta-heurísticas baseia-se na origem dos algoritmos. Nesse contexto, há algoritmos inspirados na natureza, como os Algoritmos Genéticos e os Algoritmos de Colônia de Formigas, e algoritmos não inspirados na natureza, como a Busca Tabu e o ILS. No entanto, essa classificação apresenta limitações. Primeiramente, muitos algoritmos híbridos recentes não se enquadram claramente em apenas uma dessas categorias e, em certo sentido, podem pertencer a ambas. Além disso, em alguns casos, é difícil atribuir um algoritmo de forma inequívoca a uma dessas classes, como, por exemplo, ao considerar o uso de memória na Busca Tabu, que também pode ser interpretado como uma característica inspirada na natureza.

(b) Baseadas em população vs. baseadas em solução única.

Outra forma relevante de classificação considera o número de soluções manipuladas simultaneamente durante o processo de busca. Algoritmos que operam sobre uma única solução são denominados métodos de trajetória e incluem meta-heurísticas baseadas em busca local, como a Busca Tabu, o ILS e o VNS. Esses métodos têm em comum a descrição de uma trajetória no espaço de soluções ao longo do processo de busca. Por outro lado, meta-heurísticas baseadas em população realizam a busca por meio da evolução de um conjunto de soluções no espaço de busca.

(c) Função objetivo estática vs. dinâmica.

Meta-heurísticas também podem ser classificadas de acordo com a forma como utilizam a função objetivo. Enquanto alguns algoritmos mantêm a função objetivo original do problema inalterada, outros, como a Busca Local Guiada, modificam essa função ao longo do processo de busca. A ideia central dessa estratégia é evitar que a busca fique presa em ótimos locais, modificando a forma de avaliar as soluções ao longo da busca.

(d) Uso de uma ou de múltiplas estruturas de vizinhança.

A maioria das meta-heurísticas opera com uma única estrutura de vizinhança. Entretanto, algumas abordagens, como o VNS, utilizam múltiplas estruturas de vizinhança, permitindo diversificar a busca por meio da alternância entre elas.

(e) Uso de memória vs. métodos sem memória.

Outra característica importante para a classificação de meta-heurísticas é o uso da informação acumulada ao longo do processo de busca. Algoritmos sem uso de memória baseiam suas decisões apenas no estado atual. Por outro lado, meta-heurísticas que utilizam memória exploram informações históricas da busca. Em

geral, distingue-se entre a memória de curto prazo, que armazena informações recentes, como movimentos realizados ou soluções exploradas, e a memória de longo prazo, que acumula informações mais gerais sobre o processo de busca. O uso de memória é atualmente reconhecido como um dos elementos fundamentais para o desenvolvimento de meta-heurísticas eficazes.

Como tratado acima, há uma grande variedade de *frameworks* de meta-heurísticas disponíveis na literatura que podem ser classificados de várias maneiras. É comum a todas essas abordagens a presença de mecanismos de intensificação e de diversificação. Devido ao destaque dado aos algoritmos propostos nesta tese, as fundamentações das meta-heurísticas VNS e ILS serão abordadas com maior profundidade nas próximas duas subseções.

2.1.3 Variable Neighborhood Search

Esta seção baseia-se no tutorial sobre o método VNS apresentado em [Hansen & Mladenović \(2005\)](#).

O VNS é uma meta-heurística que explora sistematicamente diferentes estruturas de vizinhança de perturbação para evitar a parada prematura em ótimos locais. Uma das principais vantagens do VNS é sua simplicidade conceitual e o número reduzido de parâmetros envolvidos. Além disso, procedimentos baseados em VNS frequentemente produzem soluções de alta qualidade e permitem compreender claramente os mecanismos responsáveis por seu desempenho, o que favorece o desenvolvimento de implementações eficientes.

O esquema básico do VNS é apresentado no Algoritmo 2. Inicialmente, define-se um conjunto de estruturas de vizinhança de perturbação. A partir de uma solução corrente s , gera-se uma nova solução s' por meio de um procedimento de *shaking*, seguido da aplicação de uma busca local, que conduz a um ótimo local s'' .

Três situações podem ocorrer:

- (i) $s'' = s$: a busca retorna ao mesmo ótimo local;
- (ii) $s'' \neq s$ e $f(s'') \geq f(s)$: uma nova solução é encontrada, mas sem melhoria;
- (iii) $s'' \neq s$ e $f(s'') < f(s)$: uma solução melhor é obtida, e a busca é reiniciada a partir dela.

Caso nenhuma melhoria seja encontrada após a exploração de todas as vizinhanças, o processo é repetido até que uma condição de parada seja satisfeita.

Algoritmo 2 Esquema do *Basic* VNS (Hansen & Mladenović, 2005)

```

1: Inicialização: selecione o conjunto de estruturas de vizinhança de perturbação
    $N_k$ , para  $k = 1, \dots, k_{\max}$ , a serem utilizadas na busca; encontre uma solução
   inicial  $s$ ; defina uma condição de parada;
2: enquanto a condição de parada não for satisfeita faça
3:   Defina  $k \leftarrow 1$ 
4:   enquanto  $k \leq k_{\max}$  faça
5:     Perturbação: gere aleatoriamente uma solução  $s'$  na  $k$ -ésima vizinhança
     de  $s$  ( $s' \in N_k(s)$ )
6:     Busca local: aplique um método de busca local utilizando  $s'$  como solução
     inicial
7:     Denote por  $s''$  o ótimo local obtido
8:     se  $f(s'') < f(s)$  então
9:       Atualize  $s \leftarrow s''$ 
10:      Defina  $k \leftarrow 1$ 
11:    senão
12:      Defina  $k \leftarrow k + 1$ 
13:    fim se
14:  fim enquanto
15: fim enquanto

```

Essa estratégia permite equilibrar intensificação e diversificação, concentrando o esforço de busca em regiões promissoras, sem deixar de explorar regiões mais distantes do espaço de soluções quando necessário.

A escolha das estruturas de vizinhança desempenha um papel fundamental no desempenho do VNS. Idealmente, a união das vizinhanças deve ser suficientemente rica para permitir a exploração de todo o espaço viável:

$$X \subseteq \bigcup_{k=1}^{k_{\max}} N_k(s), \quad \forall s \in X.$$

Uma forma prática de obter essa propriedade é por meio de vizinhanças encaixadas, isto é,

$$N_1(s) \subseteq N_2(s) \subseteq \dots \subseteq N_{k_{\max}}(s),$$

o que facilita a implementação e permite controlar o grau de intensificação e diversificação da busca.

Por fim, em alguns problemas, pode ser vantajoso permitir a exploração de soluções inviáveis por meio da introdução de penalizações na função objetivo. Essa estratégia amplia o espaço de busca e pode contribuir para evitar que a busca fique presa em ótimos locais de baixa qualidade.

Nesta tese, são propostos procedimentos heurísticos no B&B-GVNS baseados em

GVNS. O GVNS é uma variante do VNS que utiliza um procedimento VND como método de busca local na linha 6.

2.1.4 Iterated local search

A descrição do ILS apresentada a seguir baseia-se no trabalho de [Lourenço et al. \(2003\)](#).

Dado um problema de minimização e um procedimento de busca local `LocalSearch()` que fornece mínimos locais, seja S^* o conjunto de todos os mínimos locais obtidos ao aplicar `LocalSearch()` a uma solução do espaço de soluções. O ILS permite explorar o espaço S^* por meio de um percurso no qual se desloca de um mínimo local para outro, sem a restrição de utilizar apenas movimentos de melhora. O procedimento ocorre da seguinte forma. Dado um mínimo local corrente s^* , aplica-se inicialmente uma modificação ou perturbação que leva a uma solução intermediária s' . Em seguida, aplica-se um procedimento de busca local `LocalSearch( $s'$ )`, obtendo-se um mínimo local s'' . Se s'' satisfizer um critério de aceitação, ele se torna o próximo elemento do percurso; caso contrário, mantém-se a solução corrente s^* .

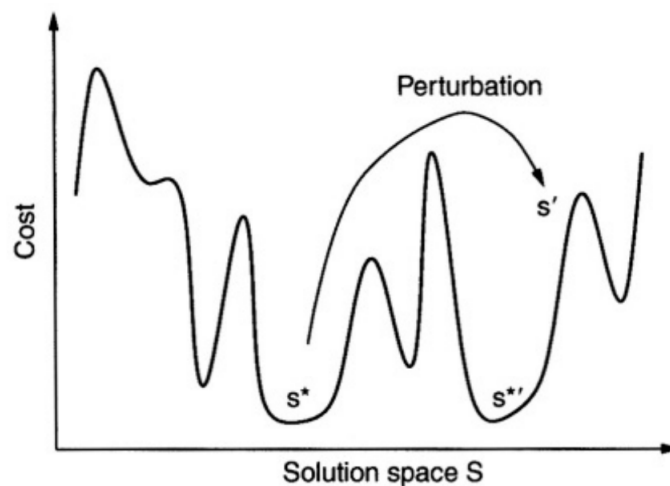


Figura 2.1: Ilustração de uma iteração do ILS
Retirado de: [Lourenço et al. \(2003\)](#).

O percurso resultante constitui uma busca estocástica no conjunto S^* em que as vizinhanças não são explicitamente definidas, mas sim implicitamente por meio de uma perturbação seguida de uma busca local. Esse procedimento de busca tende a produzir uma amostragem diversificada de S^* desde que as perturbações não sejam excessivamente pequenas nem excessivamente grandes. Se elas forem muito pequenas, frequentemente ocorrerá um retorno à solução s^* , e poucas novas soluções de S^* serão exploradas. Por outro lado, se as perturbações forem muito grandes, as soluções

Algoritmo 3 ILS (Lourengo et al., 2003)

```

1:  $s_0 \leftarrow$  Solução Inicial
2:  $s^* \leftarrow$  Busca Local( $s_0$ )
3: enquanto critério de parada não é satisfeito faça
4:    $s' \leftarrow$  Perturbação( $s^*$ , histórico)
5:    $s'' \leftarrow$  Busca Local( $s'$ )
6:    $s^* \leftarrow$  Critério de Aceitação( $s^*$ ,  $s''$ , histórico)
7: fim enquanto
8: retorne  $s^*$ 

```

geradas serão essencialmente aleatórias e o procedimento se reduzirá a um algoritmo que reinicia aleatoriamente a cada iteração.

O procedimento geral do ILS é ilustrado esquematicamente na Figura 2.1. Lourenço et al. (2003) explicam que, como as perturbações determinísticas podem levar a ciclos de repetição curtos, é recomendável randomizar as perturbações ou torná-las adaptativas, de modo a evitar a ciclagem. Se as perturbações dependerem de alguma das soluções s^* previamente visitadas, o método passa a explorar o histórico da busca. O esquema geral do ILS é apresentado no Algoritmo 3.

2.2 Matheurísticas

Segundo Fischetti & Fischetti (2018), uma primeira abordagem para resolver problemas de otimização, mesmo os considerados difíceis, consiste em, após a análise do problema e das simplificações aceitáveis, estabelecer um modelo de Programação Matemática (PM) eficaz e resolvê-lo por meio de um *software*, geralmente um *solver* de programação linear inteira mista. Devido aos avanços nos métodos de resolução empregados por *solvers* nos últimos anos, essa abordagem pode, de fato, resolver instâncias de interesse ou fornecer, em um período de tempo razoável, soluções de boa qualidade.

Quando isso não ocorre, pode-se insistir na abordagem baseada em PM, buscando melhores resultados por meio de alterações no modelo e/ou do aprimoramento das configurações do *solver*. Alternativamente, pode-se recorrer a heurísticas não baseadas em modelos matemáticos. Nesse caso, o modelo de PM é completamente desconsiderado ou utilizado apenas para ilustrar o problema. Entretanto, ainda existe uma terceira abordagem, que consiste em utilizar modelos de PM como ferramenta central em uma estrutura heurística. Essa hibridização entre modelos de Programação Matemática e meta-heurísticas conduz à abordagem denominada matheurística, que tem se tornado cada vez mais popular nos últimos anos.

Boschetti & Maniezzo (2022) definem matheurísticas como algoritmos heurísticos

baseados em ferramentas matemáticas, como as fornecidas por modelos de Programação Matemática, que são estruturalmente gerais o suficiente para serem aplicadas a diferentes problemas, com poucas adaptações em sua estrutura abstrata. Os resultados podem ser meta-heurísticas híbridas, contendo componentes derivados de modelos matemáticos dos problemas de interesse, mas as próprias técnicas matemáticas também podem definir estruturas gerais para soluções heurísticas.

Conforme as definições de Fischetti & Fischetti (2018) e Boschetti & Maniezzo (2022) descritas anteriormente, métodos heurísticos que exploram relaxações lagrangeanas (RLGs) combinadas com o método do subgradiente são exemplos clássicos de matheurísticas amplamente empregadas para resolver o SSCFLP. Outros exemplos incluem o *Local Branching* (Fischetti & Lodi, 2003), que consiste em um procedimento de busca local no qual as vizinhanças são definidas pela adição de restrições ao modelo de PM; e o *Relaxation-Induced Neighborhood Search* (Danna, 2004), que explora vizinhanças obtidas pela fixação de variáveis no modelo de PM.

Um dos algoritmos mais relevantes da literatura para a resolução do SSCFLP baseia-se na matheurística *Kernel Search* (KS) (Angelelli et al., 2010). O KS é composto por duas fases: inicialização e expansão. Na fase de inicialização, um procedimento que tipicamente utiliza valores de custo reduzido provenientes da solução da relaxação linear (RLI) é empregado para identificar um subconjunto de variáveis promissoras, denominado *kernel*. As variáveis que não pertencem ao *kernel* são particionadas em subconjuntos chamados *buckets*. Em seguida, um *solver* é utilizado para resolver o subproblema restrito às variáveis do *kernel*, possivelmente impondo limites computacionais, como tempo ou memória, a fim de obter uma solução inicial viável.

A fase de expansão consiste em um processo iterativo de construção e resolução de subproblemas. Cada subproblema é composto pelas variáveis do *kernel* corrente e pelas variáveis de um *bucket*. Além disso, duas restrições são adicionadas: uma para garantir que a solução do subproblema contenha pelo menos uma variável do *bucket* com valor não nulo, e outra para garantir que o subproblema produza uma solução melhor do que a solução incumbente. Caso tal solução seja encontrada, ela passa a ser a nova solução incumbente, e as variáveis com valor não nulo nessa solução são incorporadas ao *kernel*. O procedimento continua até que todos os *buckets* sejam explorados.

2.3 Métodos exatos

Os métodos exatos são procedimentos de resolução que buscam encontrar uma solução ótima de um problema de otimização inteiro, quando ela existe, ou demonstrar

que o problema é infactível. Os principais métodos exatos que fundamentam esta tese são os métodos *branch-and-bound*, *branch-and-cut* e *cut-and-solve*, descritos nas subseções a seguir.

2.3.1 *Branch-and-bound e branch-and-cut*

O algoritmo *branch-and-bound* é um método sistemático de enumeração implícita no espaço de soluções factíveis. Ele é estruturado como uma árvore de busca, na qual o espaço de soluções é particionado recursivamente em subconjuntos menores, que definem os subproblemas associados aos nós da árvore de busca. A exploração da árvore é guiada pelos limitantes inferiores (LI) e superiores (LS), obtidos a partir das relaxações lineares dos subproblemas (Lawler & Wood, 1966; Balas & Toth, 1983).

Para descrever o método B&B, considere que o problema a ser resolvido é um problema de minimização P^0 e a seguinte notação é utilizada: P^i é o subproblema associado ao nó i e PL^i sua relaxação linear; $s(PL^i)$ e $opt(PL^i)$ representam, respectivamente, uma solução ótima e o valor ótimo de PL^i ; e l representa a lista de nós ativos da árvore de busca. Inicialmente, $l = \{0\}$, $LS = +\infty$ e $LI = -\infty$.

Em cada iteração, um nó ativo i é selecionado, PL^i é resolvido e quatro possibilidades podem ocorrer:

1. PL^i é infactível;
2. $opt(PL^i) \geq LS$;
3. $opt(PL^i) < LS$ e $s(PL^i)$ é inteira;
4. $opt(PL^i) < LS$ e $s(PL^i)$ é fracionária.

Se ocorre 1, o nó i é podado por infactibilidade, tornando-se inativo. Se ocorrer 2, o nó i é podado por qualidade, pois nem ele nem seus descendentes podem conter soluções melhores do que a incumbente, tornando-se inativo. Se ocorrer 3, a solução incumbente e LS são atualizadas. Além disso, todos os nós ativos j tais que $opt(PL^j) \geq LS$ são podados por qualidade e tornam-se inativos. Nesse caso, o nó i é podado por integralidade.

Se ocorrer 4, a árvore é ramificada no nó i . O espaço de soluções de P^i é particionado em dois subconjuntos, da seguinte forma: seleciona-se uma variável x que assume valor fracionário x^* em $s(PL^i)$; em seguida, adiciona-se a restrição $x \leq \lfloor x^* \rfloor$ a P^i para gerar um dos nós filhos e a restrição $x \geq \lceil x^* \rceil$ a P^i para gerar o outro. Dessa forma, a solução fracionária atual é eliminada, enquanto todas as soluções inteiras viáveis permanecem em algum dos nós-filho gerados.

Algoritmo 4 Algoritmo *branch-and-bound* para minimização

-
- 1: **Passo 0 (inicialização).** Faça $LS \leftarrow +\infty$, $LI \leftarrow -\infty$, $s \leftarrow \emptyset$ e $l \leftarrow \{0\}$.
 - 2: **Passo 1 (seleção de nó).** Selecione o nó ativo i , associado ao problema P^i , da lista de nós ativos l . Se a lista estiver vazia, vá para o Passo 6.
 - 3: **Passo 2 (teste de poda por infactibilidade).** Resolva PL^i . Se PL^i for infactível, remova o nó i da lista l e vá para o Passo 1.
 - 4: **Passo 3 (teste de poda por qualidade).** Se $\text{opt}(PL^i) \geq LS$, remova o nó i da lista l e vá para o Passo 1.
 - 5: **Passo 4 (teste de poda por integralidade).** Se $s(PL^i)$ for inteira e $\text{opt}(PL^i) < LS$, atualize s e LS . Remova da lista l todos os nós ativos j tais que $\text{opt}(PL^j) \geq LS$. Remova o nó i da lista l e volte para o Passo 1.
 - 6: **Passo 5 (ramificação).** Selecione uma variável x com valor fracionário x^* em $s(PL^i)$ e divida P^i em dois subproblemas. Adicione os nós filhos à lista l , remova o nó i da lista l e vá para o Passo 1.
 - 7: **Passo 6 (fim).** Se $LS = +\infty$, não existe solução factível; caso contrário, s é a solução ótima.
-

O algoritmo termina quando não há mais nós ativos. Nesse caso, a solução incumbente é ótima. Caso nenhuma solução factível tenha sido encontrada, conclui-se que o problema é infactível.

O Algoritmo 4 apresenta um pseudocódigo do método *branch-and-bound* para problemas de minimização. Ele é uma adaptação do algoritmo apresentado em Arenales et al. (2007). No Algoritmo 4, PL^i representa a relaxação linear do problema associado ao nó i , enquanto $s(PL^i)$ e $\text{opt}(PL^i)$ representam, respectivamente, uma solução ótima e o valor ótimo de PL^i .

O método *branch-and-cut* é uma extensão do método *branch-and-bound* que incorpora planos de corte ao longo da árvore de busca. Esses cortes fortalecem as formulações dos subproblemas resolvidos nos nós da árvore, possibilitando a obtenção de limitantes inferiores mais apertados, o que pode reduzir o número de nós explorados na árvore de B&B.

2.3.2 Cut-and-solve

A estratégia CS é um *framework* para a resolução exata de problemas combinatórios, introduzido por Climer & Zhang (2006). O método baseia-se em uma árvore de busca em profundidade, na qual cada iteração é guiada por um corte perfurante (*piercing cut*), obtido a partir da solução de uma relaxação.

A cada iteração, o corte perfurante define dois subproblemas: o problema esparso (PE) e o problema denso (PD). O problema esparso restringe o espaço de soluções ao conjunto de soluções que violam o corte perfurante, sendo tipicamente resolvido como um problema inteiro, com o objetivo de encontrar boas soluções factíveis (limitantes

Algoritmo 5 Esquema *cut-and-solve*

```

1: Inicialização:  $PD_0 \leftarrow P$ ,  $LS \leftarrow +\infty$ ,  $s' \leftarrow \emptyset$ 
2:  $LI \leftarrow \text{opt}(RPD_0)$ 
3:  $k \leftarrow 1$ 
4: enquanto  $LS > LI$  faça
5:   Selecione o  $CP_k$ 
6:   Resolva o  $PE_k$ 
7:   se  $\text{opt}(PE_k) < LS$  então
8:     Atualize  $s'$  e  $LS$ 
9:   fim se
10:   $PD_k \leftarrow PD_{k-1}$  com a adição de  $CP_k$ 
11:   $LI \leftarrow \text{opt}(RPD_k)$ 
12:   $k \leftarrow k + 1$ 
13: fim enquanto
14: Retorne  $s'$  e  $LS$ 

```

superiores). Já o problema denso incorpora o corte à formulação, reduzindo o espaço de soluções viáveis e sendo utilizado para obter limitantes inferiores por meio de sua relaxação.

Algoritmo 5 apresenta o pseudocódigo do esquema CS. O primeiro limitante inferior é dado pela solução da relaxação de PD_0 . Em cada iteração, seleciona-se um corte perfurante com base na solução do problema denso anterior (linha 5). O problema esparso correspondente é então resolvido (linha 6). Caso uma solução melhor seja encontrada, a solução incumbente e o limitante superior são atualizados (linha 8). Em seguida, na linha 10 o corte é incorporado ao problema denso, gerando o próximo subproblema da sequência. O processo se repete até que $LS \leq LI$, condição em que a solução incumbente é ótima.

O método *cut-and-solve* requer a definição de três componentes principais: (i) a escolha de uma relaxação adequada para problemas densos; (ii) um procedimento para a geração de cortes perfurantes; e (iii) métodos eficientes para resolver problemas esparsos e relaxações de problemas densos. Essas escolhas dependem fortemente das características específicas do problema (Climer & Zhang, 2006).

Uma vantagem importante do método é que a árvore de busca cresce essencialmente em profundidade, o que reduz significativamente a necessidade de armazenamento de nós e, conseqüentemente, o uso de memória. Além disso, não é necessário definir estratégias de seleção de nós, como ocorre em métodos tradicionais baseados em árvores de busca amplas. Uma desvantagem é que o método exige a resolução de subproblemas inteiros, que podem ser tão desafiadores computacionalmente quanto o próprio problema original. Assim como o B&B, procedimentos de planos de corte podem ser incorporados ao CS para tornar a estratégia mais eficiente. Exemplos de

aplicações do CS podem ser encontrados em [Fang et al. \(2014\)](#), [Luo et al. \(2022\)](#), [Hu et al. \(2022\)](#) e [Yang et al. \(2019\)](#).

2.4 Método de planos de corte

O método de planos de corte constitui uma das principais estratégias empregadas nesta tese para fortalecer formulações de problemas de otimização inteira. Nesta seção, são apresentados os fundamentos teóricos desse método, com base na teoria de desigualdades válidas desenvolvida por [Nemhauser & Wolsey \(1988b\)](#). As demonstrações das proposições apresentadas nesta seção podem ser encontradas nessa referência.

Nesta seção e no restante deste capítulo, considere que $\mathbf{c}$ é um vetor de dimensão n , $\mathbf{A}$ uma matriz $m \times n$ e $\mathbf{b}$ um vetor de dimensão m . Além disso, considere que $\mathcal{P}$ é o poliedro $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} \leq \mathbf{b}\}$ e que a i -ésima linha de $\mathbf{Ax} \leq \mathbf{b}$ é $\mathbf{a}_i^\top \leq b_i$.

2.4.1 Caracterização de poliedros por suas facetas

Esta subseção apresenta definições e resultados fundamentais para caracterizar poliedros por suas facetas.

Definição 2.4.1 (Desigualdade válida) *A desigualdade $\boldsymbol{\pi}^\top \mathbf{x} \leq \pi_0$ (ou $(\boldsymbol{\pi}, \pi_0)$) é chamada de desigualdade válida para $\mathcal{P}$ se for satisfeita por todos os pontos de $\mathcal{P}$. ■*

Definição 2.4.2 (Face) *Se $(\boldsymbol{\pi}, \pi_0)$ é uma desigualdade válida para $\mathcal{P}$ e*

$$F = \{\mathbf{x} \in \mathcal{P} : \boldsymbol{\pi}^\top \mathbf{x} = \pi_0\},$$

então F é chamada de face de $\mathcal{P}$, e diz-se que $(\boldsymbol{\pi}, \pi_0)$ define F . Uma face F é dita própria se $F \neq \emptyset$ e $F \neq \mathcal{P}$. ■

Definição 2.4.3 (Hiperplano suporte) *Uma face F definida por $(\boldsymbol{\pi}, \pi_0)$ é não vazia se, e somente se,*

$$\max\{\boldsymbol{\pi}^\top \mathbf{x} : \mathbf{x} \in \mathcal{P}\} = \pi_0.$$

Nesse caso, diz-se que $(\boldsymbol{\pi}, \pi_0)$ suporta $\mathcal{P}$. ■

Definição 2.4.4 (Faceta) *Uma face F de $\mathcal{P}$ é dita faceta se*

$$\dim(F) = \dim(\mathcal{P}) - 1.$$

Equivalentemente, F é uma face própria maximal, isto é, não existe face F' tal que

$$F \subsetneq F' \subsetneq \mathcal{P}. \quad \blacksquare$$

Definição 2.4.5 (Desigualdades equivalentes) Duas desigualdades válidas

$$\boldsymbol{\pi}^\top \mathbf{x} \leq \pi_0$$

e

$$\boldsymbol{\gamma}^\top \mathbf{x} \leq \gamma_0$$

são ditas equivalentes se

$$(\boldsymbol{\gamma}, \gamma_0) = \lambda(\boldsymbol{\pi}, \pi_0)$$

para algum $\lambda > 0$. ■

Definição 2.4.6 (Desigualdade dominante) Diz-se que a desigualdade $\boldsymbol{\gamma}^\top \mathbf{x} \leq \gamma_0$ domina a desigualdade $\boldsymbol{\pi}^\top \mathbf{x} \leq \pi_0$ se as desigualdades não forem equivalentes e

$$\{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{\gamma}^\top \mathbf{x} \leq \gamma_0\} \subset \{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{\pi}^\top \mathbf{x} \leq \pi_0\}. \quad \blacksquare$$

Nesse caso, diz-se também que $\boldsymbol{\gamma}^\top \mathbf{x} \leq \gamma_0$ é mais forte do que $\boldsymbol{\pi}^\top \mathbf{x} \leq \pi_0$, ou que $\boldsymbol{\pi}^\top \mathbf{x} \leq \pi_0$ é mais fraca.

Uma desigualdade válida maximal é aquela que não é dominada por nenhuma outra desigualdade válida. Toda desigualdade válida maximal para $\mathcal{P}$ define uma face não vazia, e o conjunto de desigualdades válidas maximais contém todas as desigualdades que definem facetas de $\mathcal{P}$.

Proposição 2.4.7 Se F é uma faceta de $\mathcal{P}$, então existe uma desigualdade da forma

$$\mathbf{a}_i^\top \mathbf{x} \leq b_i$$

que define F . ■

Teorema 2.4.8 Seja $\mathcal{P} \subseteq \mathbb{R}^n$ um poliedro de dimensão n . Então, $\mathcal{P}$ admite uma representação irredundante finita da forma

$$\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_i^\top \mathbf{x} \leq b_i, i = 1, \dots, t\},$$

na qual cada desigualdade $\mathbf{a}_i^\top \mathbf{x} \leq b_i$ define uma faceta de $\mathcal{P}$.

Além disso, para cada faceta F_i de $\mathcal{P}$, existe uma única desigualdade válida, a menos de multiplicação por um escalar positivo, que define F_i . ■

2.4.2 Caracterização poliédrica de problemas inteiros

Nesta subseção, apresenta-se a caracterização poliédrica de problemas inteiros.

Considere

$$\mathcal{S} = \mathbb{Z}^n \cap \mathcal{P}.$$

Um problema inteiro (PI) é definido como:

$$\max \{ \mathbf{c}^\top \mathbf{x} : \mathbf{x} \in \mathcal{S} \}.$$

Definição 2.4.9 (*Casca convexa*) A casca convexa de $\mathcal{S}$, denotada por $\text{conv}(\mathcal{S})$, é o menor conjunto convexo que contém $\mathcal{S}$.

Um resultado fundamental da programação inteira estabelece que um problema inteiro pode ser resolvido por meio da otimização linear sobre a casca convexa de suas soluções viáveis.

Teorema 2.4.10 $\text{conv}(\mathcal{S})$ é um poliedro e

$$\max \{ \mathbf{c}^\top \mathbf{x} : \mathbf{x} \in \mathcal{S} \} = \max \{ \mathbf{c}^\top \mathbf{x} : \mathbf{x} \in \text{conv}(\mathcal{S}) \}.$$

Este teorema, juntamente com o Teorema 2.4.8, mostra que a resolução de problemas inteiros pode ser tratada por meio da caracterização completa de $\text{conv}(\mathcal{S})$ por desigualdades lineares que definem suas facetas. Entretanto, obter essa caracterização completa é, em geral, computacionalmente difícil na prática.

Uma alternativa consiste em identificar desigualdades válidas para $\text{conv}(\mathcal{S})$ que não são válidas para $\mathcal{P}$. A adição dessas desigualdades ao modelo original permite construir poliedros intermediários $\mathcal{Q}$ tais que

$$\text{conv}(\mathcal{S}) \subseteq \mathcal{Q} \subseteq \mathcal{P}.$$

Geometricamente, isso corresponde a aproximar externamente $\text{conv}(\mathcal{S})$, ou seja, “por fora”, removendo regiões fracionárias de $\mathcal{P}$ que não contêm soluções inteiras viáveis.

Esse princípio fundamenta os métodos de planos de corte, que buscam fortalecer formulações lineares por meio da adição de desigualdades válidas. Nesta tese, tais métodos são explorados no desenvolvimento de algoritmos para resolver o SSCFLP e o RSSCFLP.

Algoritmo 6 Algoritmo de planos de corte

```

1: Inicialização: defina  $t \leftarrow 1$ ,  $LS \leftarrow +\infty$  e  $LI \leftarrow -\infty$ ;  $\mathcal{S}_1 = \{x \in \mathbb{R}^n : Ax \leq b\}$ .
2: enquanto verdadeiro faça
3:   Resolva a relaxação  $z_t = \min\{cx : x \in \mathcal{S}_t\}$  e considere  $x_t$  a solução obtida;
4:   se  $x_t \in S$  então
5:     retorne  $x_t$  como solução ótima do problema;
6:   fim se
7:   Atualize o limitante inferior:  $LI \leftarrow z_t$ ;
8:   Defina  $t \leftarrow t + 1$  ;
9:    $\mathcal{S}_{t+1} = \mathcal{S}_t \cap \{x \in \mathbb{R}^n : \pi^t x \leq \pi_b\}$  onde  $(\pi^t, \pi_b)$  é uma desigualdade válida para
    $S$  tal que  $\pi^t x_t > \pi_b$ . ;
10: fim enquanto

```

2.4.3 Algoritmo de planos de corte

O algoritmo de planos de corte desempenha um papel importante nesta tese. A fundamentação teórica sobre este tema está descrita em [Nemhauser & Wolsey \(1988a, Cap. II.4\)](#). Nela, os autores introduzem a estratégia de planos de corte como um caso particular da estratégia de relaxação geral para resolver problemas inteiros. Em um algoritmo de relaxação geral, em cada iteração t , uma relaxação $\mathcal{S}_t$ do espaço de soluções é considerada e o problema relaxado correspondente é resolvido. Se a solução ótima da relaxação não for ótima para o PI, ela fornece um limitante inferior e a relaxação $\mathcal{S}_t$ é refinada.

O algoritmo de planos de corte é um caso particular de um algoritmo de relaxação geral. Ele utiliza relaxação linear para definir $\mathcal{S}_t$ na inicialização e, ao iterar, atualiza $\mathcal{S}_t$ adicionando uma desigualdade válida para o PI que elimina a solução fracionária da relaxação corrente. O Algoritmo 6 apresenta o pseudocódigo do esquema de planos de corte.

2.4.4 Fortalecendo desigualdades válidas em problemas binários

No caso específico em que o PI é binário, isto é, $\mathcal{S} \subseteq B^n = \{0,1\}^n$, um princípio geral chamado de *lifting* permite estender para $\mathcal{S}$ uma desigualdade válida para $\mathcal{S}^\delta = \mathcal{S} \cap \{x \in B^n : x_1 = \delta\}$, com $\delta \in \{0,1\}$, conforme as seguintes proposições:

Proposição 2.4.11 *Suponha que:*

$$\sum_{j=2}^n \pi_j x_j \leq \pi_0 \quad (2.1)$$

é válida para $\mathcal{S}^0$. Se $\mathcal{S}^1 = \emptyset$, então $x_1 \leq 0$ é válida para $\mathcal{S}$. Se, por outro lado,

$\mathcal{S}^1 \neq \emptyset$, então:

$$\sum \alpha_1 x_1 + \sum_{j=2}^n \pi_j x_j \leq \pi_0 \quad (2.2)$$

é válida para $\mathcal{S}$ para qualquer $\alpha_1 \leq \pi_0 - \varepsilon$, sendo $\varepsilon = \max \left\{ \sum_{j=2}^n \pi_j x_j : x \in \mathcal{S}^1 \right\}$. Mais ainda, se $\alpha_1 = \pi_0 - \varepsilon$ e a expressão (2.1) fornece uma face de dimensão k de $\text{conv}(\mathcal{S}^0)$, então a expressão (2.2) fornece uma face de dimensão pelo menos $k+1$ de $\text{conv}(\mathcal{S})$. Além disso, se a expressão (2.1) fornece uma faceta de $\text{conv}(\mathcal{S}^0)$, então a expressão (2.2) fornece uma faceta de $\text{conv}(\mathcal{S})$. ■

Proposição 2.4.12 *Suponha que a expressão (2.1) seja válida para $\mathcal{S}^1$; Se $\mathcal{S}^0 = \emptyset$, então $x \geq 1$ é válida para $\mathcal{S}$. Se $\mathcal{S}^0 \neq \emptyset$, então*

$$\gamma_1 x_1 + \sum_{j=2}^n \pi_j x_j \leq \pi_0 + \gamma_1 \quad (2.3)$$

é válida para $\mathcal{S}$ para qualquer $\gamma \geq \varepsilon - \pi_0$, sendo $\varepsilon = \max \left\{ \sum_{j=2}^n \pi_j x_j : x \in \mathcal{S}^0 \right\}$. Mais ainda, se $\gamma_1 = \varepsilon - \pi_0$ e a expressão (2.1) fornece uma face de dimensão k de $\text{conv}(\mathcal{S}^1)$, então a expressão (2.3) fornece uma face de dimensão pelo menos $k+1$ de $\text{conv}(\mathcal{S})$. ■

As duas proposições podem ser utilizadas sequencialmente. Dado $N_1 \subset N = \{1, \dots, n\}$ e seja uma desigualdade $\sum_{j \in N_1} \pi_j x_j \leq \pi_0$ uma desigualdade válida para $\mathcal{S} \cap \{x \in B^n : x_j = 0 \text{ para } j \in N \setminus N_1\}$. Pode-se, então, aplicar o *lifting* de uma variável de cada vez para obter a desigualdade válida

$$\sum_{j \in N \setminus N_1} \alpha_j x_j + \sum_{j \in N_1} \pi_j x_j \leq \pi_0 \quad (2.4)$$

para $\mathcal{S}$. Os coeficientes α_j dependem da ordem de aplicação do *lifting*. Então, considerando diferentes ordens nos elementos de $N \setminus N_1$, obtém-se uma família de desigualdades válidas para $\mathcal{S}$. Os procedimentos de *lifting* das Proposições 2.4.11 e 2.4.12 são referidos na literatura, por exemplo, em Gu et al. (2000), como *up-lifting* e *down-lifting*, respectivamente. A aplicação desses procedimentos para obtenção de LCIs é referida nesta tese como *lifting* sequencial.

2.4.5 Lifted cover inequalities

Nesta subseção, mostra-se como o princípio de *lifting* pode ser utilizado para gerar facetas para problemas da mochila binária.

Considere o conjunto restrição do problema da mochila binária:

$$\mathcal{S} = \left\{ \mathbf{x} \in B^n : \sum_{j \in N} a_j x_j \leq b \right\},$$

sendo $N = \{1, \dots, n\}$, $a_j \in \mathbb{Z}_+$. Como $a_j > b$ implica que $x_j = 0$ para todo $\mathbf{x} \in \mathcal{S}$, assume-se, sem perda de generalidade, que $a_j \leq b$ para todo $j \in N$. Convenientemente, supõe-se que os coeficientes satisfazem $a_1 \geq a_2 \geq a_3 \cdots \geq a_n$ e representam-se os elementos de B^n por vetores característicos, de modo que, para $R \subseteq N$, o vetor $\mathbf{x}^R$ tem componentes $x_j^R = 1$ se $j \in R$ e $x_j^R = 0$, caso contrário.

Com essa notação, define-se que $C \subseteq N$ é uma cobertura para $\mathcal{S}$ se $\mathbf{x}^C \notin \mathcal{S}$, isto é, $\sum_{j \in C} a_j > b$. Uma cobertura C é dita minimal se, para todo $k \in C$, $\sum_{j \in C \setminus \{k\}} a_j \leq b$. Ou seja, a remoção de qualquer elemento de C faz com que o conjunto deixe de ser uma cobertura.

Proposição 2.4.13 *Se C é uma cobertura para $\mathcal{S}$, então*

$$\sum_{j \in C} x_j \leq |C| - 1 \quad (2.5)$$

é uma desigualdade válida para $\mathcal{S}$. ■

A desigualdade (2.5) é chamada de corte de cobertura para $\mathcal{S}$.

Proposição 2.4.14 *Se C é uma cobertura minimal para $\mathcal{S}$ e (C_1, C_2) é uma partição de C com $C_1 \neq \emptyset$, então $\text{conv}(\mathcal{S})$ possui uma faceta dada por:*

$$\sum_{j \in N \setminus C} \alpha_j x_j + \sum_{j \in C_2} \gamma_j x_j + \sum_{j \in C_1} x_j \leq |C_1| - 1 + \sum_{j \in C_2} \gamma_j, \quad (2.6)$$

sendo $\alpha_j \geq 0$ para todo $j \in N \setminus C$ e $\gamma_j \geq 0$ para todo $j \in C_2$. ■

A desigualdade (2.6) é pelo menos tão forte quanto a desigualdade (2.5). Ela também é conhecida como *Lifted Cover Inequality* (LCI). O LCI da sentença (2.6) depende da escolha de C_1 e C_2 , bem como da ordem de aplicação do *up-lifting* das variáveis em $N \setminus C$ e do *down-lifting* das variáveis de C_2 .

Exemplo 2.4.15 *Considere $\mathcal{S} = \{x \in B^5 : 3x_1 + x_2 + x_3 + x_4 + x_5 \leq 4\}$, $C = \{1, 4, 5\}$, $C_1 = \{4, 5\}$ e $C_2 = \{1\}$. Defina a ordem de lifting: x_3 (*up-lifting*), x_1 (*down-lifting*) e x_2 (*up-lifting*).*

Pela Proposição 2.4.13, se $x_1 = 1$, $x_4 + x_5 \leq 1$ é uma faceta de $\text{conv}\{(x_4, x_5) \in B^2 : x_4 + x_5 \leq 4 - 3 = 1\}$. Aplicando a Proposição 2.4.11, temos que o coeficiente

de x_3 é dado por

$$\alpha_3 = 1 - \max\{x_4 + x_5 : x_4 + x_5 \leq 4 - 3x_1 - x_3, x_1 = x_3 = 1\}.$$

Como $\alpha_3 = 1$, segue que $x_3 + x_4 + x_5 \leq 1$ é uma faceta de $\text{conv}\{x \in B^3 : x_3 + x_4 + x_5 \leq 1\}$. Aplicando, em seguida, a Proposição 2.4.12 temos que o coeficiente de x_1 é dado por

$$\gamma_1 = \max\{x_3 + x_4 + x_5 : x_3 + x_4 + x_5 \leq 4\} - 1 = 2.$$

Logo, $2x_1 + x_3 + x_4 + x_5 \leq 3$ é uma faceta de $\text{conv}\{x \in B^4 : 3x_1 + x_3 + x_4 + x_5 \leq 4\}$. Por fim, aplicando a Proposição 2.4.11, obtém-se

$$\alpha_2 = 3 - \max\{2x_1 + x_3 + x_4 + x_5 : 3x_1 + x_3 + x_4 + x_5 \leq 5\}.$$

Então, $\alpha_2 = 0$ e $2x_1 + x_3 + x_4 + x_5 \leq 3$ é uma faceta de $\text{conv}(\mathcal{S})$. ■

2.5 Dualidade em programação linear

Esta seção baseia-se em [Bazaraa et al. \(2005, Capítulo 6\)](#). Considere o seguinte problema de programação linear na forma de minimização:

$$\begin{aligned} \min \quad & \mathbf{c}^T \mathbf{x} \\ \text{sujeito a} \quad & A\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq 0. \end{aligned} \tag{2.7}$$

Denominamos esse problema de *Problema Primal*. Associado ao problema primal, define-se o *Problema Dual*:

$$\begin{aligned} \max \quad & \mathbf{b}^T \mathbf{y} \\ \text{sujeito a} \quad & A^T \mathbf{y} \geq \mathbf{c}, \\ & \mathbf{y} \leq 0. \end{aligned} \tag{2.8}$$

A teoria da dualidade estabelece relações entre os problemas primal e dual. A propriedade da dualidade fraca garante que, para quaisquer soluções viáveis $\mathbf{x}$ e $\mathbf{y}$ dos problemas primal e dual, respectivamente, vale:

$$\mathbf{b}^T \mathbf{y} \leq \mathbf{c}^T \mathbf{x}. \tag{2.9}$$

Assim, toda solução do problema dual fornece um limite inferior para o problema

primal de minimização.

Um dos principais resultados da programação linear é o Teorema da Dualidade Forte, o qual estabelece que, se um dos problemas (primal ou dual) é viável e limitado, o outro também é, e seus valores ótimos coincidem. Isto é,

$$\mathbf{c}^T \mathbf{x}^* = \mathbf{b}^T \mathbf{y}^*, \quad (2.10)$$

em que $\mathbf{x}^*$ e $\mathbf{y}^*$ são soluções ótimas dos problemas primal e dual, respectivamente.

As variáveis duais têm uma interpretação econômica importante. Cada variável dual está associada a uma restrição do problema primal e pode ser interpretada como o custo marginal associado à relaxação do respectivo recurso. Mais especificamente, o valor de uma variável dual representa a taxa de variação do valor ótimo da função objetivo em relação a pequenas variações no lado direito da restrição correspondente. Por esse motivo, variáveis duais também são conhecidas como *preços sombra*.

2.6 Otimização robusta

Esta seção trata de fundamentos teóricos relacionados à Otimização Robusta com foco na formulação e linearização de problemas lineares discretos. Ela é baseada na teoria presente em [Bertsimas & Sim \(2003\)](#), [Bertsimas & Sim \(2004\)](#) e [Bertsimas et al. \(2011\)](#).

2.6.1 Ideias gerais

O paradigma clássico em programação matemática consiste no desenvolvimento de modelos sob a premissa de que os dados de entrada são completamente conhecidos e iguais a valores nominais. Entretanto, essa abordagem não considera a influência das incertezas nos dados sobre a qualidade e a viabilidade das soluções obtidas. À medida que os dados assumem valores não nominais, restrições do modelo podem ser violadas, e a solução ótima correspondente aos dados nominais pode deixar de ser ótima ou até mesmo viável. Essa observação evidencia a necessidade de desenvolver abordagens capazes de lidar com a incerteza dos dados, isto é, abordagens robustas.

O tratamento da incerteza em modelos de programação matemática é reconhecido como um problema central na otimização. Duas abordagens principais têm sido propostas na literatura: (i) programação estocástica e (ii) otimização robusta. A programação estocástica modela a incerteza por meio de cenários associados a diferentes probabilidades. Entretanto, essa abordagem apresenta limitações importantes: (i) a determinação exata da distribuição dos dados e, consequentemente, a construção de

cenários representativos raramente são viáveis na prática; e (ii) o tamanho do modelo resultante cresce significativamente com o número de cenários, o que impõe desafios computacionais relevantes.

Considere um problema de otimização com função objetivo $f_0(\mathbf{x})$ e restrições da forma $f_i(\mathbf{x}, \mathbf{u}_i) \leq 0$, em que os parâmetros $\mathbf{u}_i$ são incertos. A formulação geral da otimização robusta é dada por

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{sujeito a} \quad & f_i(\mathbf{x}, \mathbf{u}_i) \leq 0, \quad \forall \mathbf{u}_i \in \mathcal{U}_i, i = 1, \dots, m. \end{aligned} \tag{2.11}$$

Nessa formulação, $\mathbf{x} \in \mathbb{R}^n$ representa o vetor de variáveis de decisão, enquanto $f_0, f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ são funções reais. Os parâmetros incertos $\mathbf{u}_i \in \mathbb{R}^k$ podem assumir quaisquer valores pertencentes aos conjuntos de incerteza $\mathcal{U}_i \subseteq \mathbb{R}^k$, os quais, são assumidos fechados.

O objetivo do problema (2.11) é determinar uma solução de custo mínimo entre todas aquelas que permanecem viáveis para qualquer realização dos parâmetros incertos pertencentes aos respectivos conjuntos de incerteza. Assim, quando algum dos conjuntos $\mathcal{U}_i$ é contínuo, a formulação possui um número infinito de restrições. Intuitivamente, essa abordagem fornece um mecanismo de proteção da viabilidade para problemas de otimização cujos parâmetros não são conhecidos exatamente.

O fato de a função objetivo não depender explicitamente de parâmetros incertos não implica perda de generalidade. Para considerar esse caso, basta introduzir uma variável auxiliar t e minimizar t sujeito à restrição adicional $\max_{\mathbf{u}_0 \in \mathcal{U}_0} f_0(\mathbf{x}, \mathbf{u}_0) \leq t$. Também não há perda de generalidade ao assumir que o conjunto de incerteza possui a forma

$$\mathcal{U} = \mathcal{U}_1 \times \dots \times \mathcal{U}_m.$$

Caso exista um único conjunto de incerteza $\mathcal{U}$ impondo que $(\mathbf{u}_1, \dots, \mathbf{u}_m) \in \mathcal{U}$, o requisito de viabilidade restrição a restrição conduz a um problema equivalente ao da formulação (2.11), em que cada conjunto $\mathcal{U}_i$ corresponde à projeção de $\mathcal{U}$ sobre as dimensões apropriadas.

A otimização robusta difere significativamente da análise de sensibilidade. Enquanto esta última é normalmente utilizada após a obtenção da solução ótima para avaliar o impacto de pequenas perturbações nos dados do problema, a otimização robusta busca determinar soluções fixas cuja viabilidade seja garantida independentemente da realização dos parâmetros incertos. Em outras palavras, a solução robusta assegura, *a priori*, a satisfação das restrições para qualquer cenário pertencente ao conjunto de incerteza, ainda que esse conjunto represente variações significativas dos parâmetros.

Não é evidente, entretanto, quando o problema (2.11) pode ser resolvido eficientemente. Em princípio, pode-se imaginar que a incorporação de robustez a um problema de otimização implique um aumento substancial da complexidade computacional. Definindo o conjunto viável robusto como

$$\mathcal{X}(\mathcal{U}) = \{\mathbf{x} : f_i(\mathbf{x}, \mathbf{u}_i) \leq 0, \forall \mathbf{u}_i \in \mathcal{U}_i, i = 1, \dots, m\}, \quad (2.12)$$

a tratabilidade do problema está, em grande parte, associada à convexidade de $\mathcal{X}(\mathcal{U})$ em relação às variáveis de decisão, bem como à existência de um procedimento eficiente para verificar a separação.

2.6.2 Otimização linear robusta

No contexto de otimização linear, a contraparte robusta pode ser escrita, sem perda de generalidade, como

$$\begin{aligned} \min \quad & c^\top \mathbf{x} \\ \text{sujeito a} \quad & A\mathbf{x} \leq \mathbf{b}, \quad \forall \mathbf{a}_1 \in \mathcal{U}_1, \dots, \mathbf{a}_m \in \mathcal{U}_m. \end{aligned} \quad (2.13)$$

em que $\mathbf{a}_i$ representa a i -ésima linha da matriz incerta A e assume valores pertencentes ao conjunto de incerteza $\mathcal{U}_i \subseteq \mathbb{R}^n$. Assim, a restrição

$$\mathbf{a}_i^\top \mathbf{x} \leq b_i, \quad \forall \mathbf{a}_i \in \mathcal{U}_i,$$

é equivalente a

$$\max_{\mathbf{a}_i \in \mathcal{U}_i} \mathbf{a}_i^\top \mathbf{x} \leq b_i, \quad \forall i.$$

Esse problema interno é denominado *subproblema de otimização*. A seguir, são apresentados alguns exemplos de conjuntos de incerteza amplamente utilizados.

Incerteza elipsoidal

Em conjuntos de incerteza elipsoidais a incerteza é modelada por um elipsoide centrado no vetor nominal dos parâmetros, permitindo representar situações em que desvios de grande magnitude em todos os coeficientes simultaneamente são considerados pouco prováveis. O tamanho do elipsoide é controlado por um parâmetro que atua como um orçamento de incerteza, possibilitando ao tomador de decisão estabelecer um compromisso entre robustez e desempenho.

Teorema 2.6.1 (*Ben-Tal & Nemirovski, 1998*) *Considere o conjunto de incerteza elipsoidal*

$$\mathcal{U} = \mathcal{U}(\Pi, \mathbf{Q}) = \{\Pi(\mathbf{u}) \mid \|\mathbf{Q}\mathbf{u}\| \leq \rho\},$$

em que $\Pi(\mathbf{u})$ é uma transformação afim de $\mathbb{R}^L$ em $\mathbb{R}^{m \times n}$ e $\mathbf{Q} \in \mathbb{R}^{M \times L}$. Então, o problema robusto (2.13) é equivalente a um problema de programação cônica de segunda ordem.

Em particular, considere o problema

$$\begin{aligned} \min \quad & c^\top \mathbf{x} \\ \text{sujeito a} \quad & \mathbf{a}_i^\top \mathbf{x} \leq b_i, \quad \forall \mathbf{a}_i \in \mathcal{U}_i, i = 1, \dots, m, \end{aligned}$$

com conjunto de incerteza

$$\mathcal{U} = \{(\mathbf{a}_1, \dots, \mathbf{a}_m) : \mathbf{a}_i = \mathbf{a}_i^0 + \Delta_i \mathbf{u}_i, i = 1, \dots, m, \|\mathbf{u}\|_2 \leq \rho\},$$

em que $\mathbf{a}_i^0$ representa o vetor nominal dos coeficientes da i -ésima restrição. Então, a contraparte robusta é dada por

$$\begin{aligned} \min \quad & c^\top \mathbf{x} \\ \text{sujeito a} \quad & (\mathbf{a}_i^0)^\top \mathbf{x} \leq b_i - \rho \|\Delta_i \mathbf{x}\|_2, \quad i = 1, \dots, m. \end{aligned}$$

A ideia central é que, na incerteza elipsoidal, o subproblema

$$\max_{\mathbf{a}_i \in \mathcal{U}_i} \mathbf{a}_i^\top \mathbf{x}$$

corresponde à maximização sobre um conjunto definido por uma restrição quadrática. Como consequência, o problema dual também envolve funções quadráticas, resultando em uma contraparte robusta formulada como um problema de programação cônica de segunda ordem.

Incerteza poliédrica

A incerteza poliédrica pode ser vista como um caso particular da incerteza elipsoidal. Quando o conjunto de incerteza é poliédrico, o subproblema torna-se linear e, consequentemente, a contraparte robusta também pode ser formulada como um problema de programação linear.

Considere o problema

$$\min \quad c^\top \mathbf{x}$$

$$\text{sujeito a } \max_{D_i \mathbf{a}_i \leq d_i} \mathbf{a}_i^\top \mathbf{x} \leq b_i, \quad i = 1, \dots, m.$$

O dual do subproblema (observando que $\mathbf{x}$ não é variável de decisão no problema interno) é

$$\begin{array}{ll} \max & \mathbf{a}_i^\top \mathbf{x} \\ \text{sujeito a} & D_i \mathbf{a}_i \leq d_i \end{array} \quad \Longleftrightarrow \quad \begin{array}{ll} \min & \mathbf{p}_i^\top d_i \\ \text{sujeito a} & \mathbf{p}_i^\top D_i = \mathbf{x}, \\ & \mathbf{p}_i \geq 0. \end{array}$$

Assim, a contraparte robusta pode ser formulada como

$$\begin{array}{ll} \min & c^\top \mathbf{x} \\ \text{sujeito a} & \mathbf{p}_i^\top d_i \leq b_i, \quad i = 1, \dots, m, \\ & \mathbf{p}_i^\top D_i = \mathbf{x}, \quad i = 1, \dots, m, \\ & \mathbf{p}_i \geq 0, \quad i = 1, \dots, m. \end{array}$$

Portanto, o tamanho desse problema cresce apenas de forma polinomial em relação ao tamanho do problema nominal e às dimensões do conjunto de incerteza.

Incerteza com restrição de cardinalidade

[Bertsimas & Sim \(2003\)](#) propuseram uma família de conjuntos de incerteza poliédricos baseada em restrições de cardinalidade. A principal ideia consiste em limitar o número de coeficientes que podem assumir simultaneamente seus desvios máximos em relação aos valores nominais, em vez de considerar que todos possam variar simultaneamente.

Considere uma matriz de coeficientes incertos $A = (a_{ij})$, em que cada coeficiente satisfaz $a_{ij} \in [a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$. Diferentemente do modelo proposto por [Soyster \(1973\)](#), que protege contra o cenário em que todos os coeficientes podem sofrer simultaneamente seus desvios máximos, admite-se que, em cada restrição i , no máximo Γ_i coeficientes possam desviar de seus valores nominais. O parâmetro Γ_i , denominado *orçamento de incerteza*, controla o compromisso entre a robustez da solução e sua qualidade em termos da função objetivo. Quanto maior o valor de Γ_i , maior é o nível de proteção contra a incerteza, porém maior tende a ser o conservadorismo da solução obtida. Na Subseção 2.6.3 é apresentada com detalhes a formulação robusta bem como a linearização de problemas de otimização discretos que consideram conjuntos de incerteza poliédricos baseada em restrições de cardinalidade.

Incerteza baseada em normas

Bertsimas et al. (2011) demonstram que problemas de otimização linear robusta cujos conjuntos de incerteza são definidos por normas gerais podem ser reformulados como problemas convexos, em que as restrições envolvem a norma dual correspondente. Essa abordagem generaliza diversos modelos de incerteza existentes na literatura e permite representar diferentes formas de medir a magnitude das perturbações nos parâmetros do problema.

Considere o conjunto de incerteza

$$\mathcal{U} = \{ \mathbf{A} : \|\mathbf{M}(\text{vec}(\mathbf{A}) - \text{vec}(\bar{\mathbf{A}}))\| \leq \Delta \},$$

em que $\mathbf{M}$ é uma matriz inversível, $\bar{\mathbf{A}}$ representa a matriz nominal, Δ define o nível de incerteza e $\text{vec}(\mathbf{A})$ denota o vetor obtido pela concatenação das linhas de $\mathbf{A}$.

Nesse caso, o problema robusto é equivalente a

$$\begin{aligned} \min \quad & c^\top \mathbf{x} \\ \text{sujeito a} \quad & \bar{\mathbf{A}}_i^\top \mathbf{x} + \Delta \|(\mathbf{M}^\top)^{-1} \mathbf{x}_i\|_* \leq b_i, \quad i = 1, \dots, m, \end{aligned}$$

em que $\mathbf{x}_i$ é um vetor que contém os componentes de $\mathbf{x}$ nas posições correspondentes à i -ésima restrição e zeros nas demais posições, enquanto $\|\cdot\|_*$ representa a norma dual da norma empregada na definição do conjunto de incerteza.

O resultado mostra que a escolha da norma utilizada para definir o conjunto de incerteza determina diretamente a estrutura da contraparte robusta. Em particular, quando são utilizadas as normas l_1 ou l_∞ , obtém-se um problema de programação linear. Por outro lado, quando é utilizada a norma l_2 , a contraparte robusta corresponde a um problema de programação cônica de segunda ordem.

Em síntese, para uma ampla classe de conjuntos de incerteza, os problemas de otimização linear robusta permanecem tratáveis computacionalmente. A natureza do problema equivalente depende da geometria do conjunto de incerteza adotado, sendo possível obter formulações lineares ou cônicas sem perda de exatidão.

2.6.3 Formulação robusta de problemas de otimização discretos

Esta seção apresenta os fundamentos teóricos da formulação robusta de problemas discretos que é aplicada nesta tese para formular o RSSCFLP. Ela se baseia nos trabalhos de Bertsimas & Sim (2003, 2004).

A abordagem de otimização robusta proposta por Bertsimas & Sim (2003) para problemas de otimização discreta apresenta duas características principais:

- (a) permite controlar o grau de conservadorismo da solução;
- (b) é computacionalmente tratável, tanto do ponto de vista teórico quanto do prático.

Quando coeficientes de custo e dados das restrições estão sujeitos à incerteza de valores, a abordagem proposta conduz a um problema de programação inteira robusta de tamanho moderadamente maior, no qual é possível controlar o nível de conservadorismo da solução por meio de limites probabilísticos de violação das restrições.

A construção proposta por Bertsimas & Sim (2003) é descrita a seguir. Sejam $\mathbf{l}$ e $\mathbf{u}$ vetores de dimensão n . Considera-se o seguinte problema nominal de programação inteira mista (MIP), com n variáveis, sendo as k primeiras inteiras:

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} \\ \text{sujeito a} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & x_i \in \mathbb{Z}, \quad i = 1, \dots, k. \end{aligned} \tag{2.14}$$

Sem perda de generalidade, assume-se que a incerteza afeta apenas os coeficientes da matriz $\mathbf{A}$ e do vetor $\mathbf{c}$, e não o vetor $\mathbf{b}$. Caso contrário, pode-se introduzir uma variável adicional x_{n+1} e reescrever o modelo de forma equivalente.

Em aplicações práticas, geralmente estão disponíveis estimativas dos valores nominais dos coeficientes a_{ij} e de seus desvios máximos $\hat{a}_{ij}$. Não se assume conhecimento sobre a distribuição probabilística desses coeficientes. De forma análoga, considera-se que os coeficientes de custo c_j pertencem ao intervalo $[c_j, c_j + d_j]$, em que d_j representa o desvio máximo.

O modelo de incerteza adotado é descrito a seguir:

- (a) **(Incerteza na matriz $\mathbf{A}$):** Seja $N = \{1, 2, \dots, n\}$. Cada elemento a_{ij} , $j \in N$, assume valores no intervalo de incerteza $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$.
- (b) **(Incerteza no vetor de custos $\mathbf{c}$):** Cada elemento c_j , $j \in N$, assume valores no intervalo $[c_j, c_j + d_j]$, em que d_j representa o desvio em relação ao coeficiente de custo nominal c_j .

Admite-se que $\hat{a}_{ij} = 0$ ou $d_j = 0$ sejam possíveis. A única hipótese imposta aos coeficientes a_{ij} é a simetria de suas distribuições.

Para cada restrição i , introduz-se um parâmetro Γ_i , $i = 0, 1, \dots, m$, com valores no intervalo $[0, |J_i|]$, no qual $J_i = \{j \mid \hat{a}_{ij} > 0\}$. O parâmetro Γ_0 é inteiro, enquanto Γ_i , $i \geq 1$ podem assumir valores não inteiros.

O parâmetro Γ_i controla o nível de proteção contra incertezas na i -ésima restrição. Considera-se a restrição $\mathbf{a}_i^\top \mathbf{x} \leq b_i$. Assume-se que apenas um subconjunto dos coeficientes pode assumir simultaneamente seus piores valores. Especificamente, até $\lfloor \Gamma_i \rfloor$ coeficientes podem variar completamente, e um coeficiente adicional pode variar parcialmente.

Essa modelagem restringe o comportamento adverso dos dados e garante que a solução obtida permaneça viável nesse nível de proteção. Devido à simetria das distribuições, mesmo quando mais coeficientes variam, a violação ocorre com baixa probabilidade. Assim, Γ_i é interpretado como o nível de proteção da restrição i .

De forma análoga, o parâmetro Γ_0 controla o nível de robustez da função objetivo. Valores maiores de Γ_0 aumentam a proteção contra variações adversas nos custos, a custo de um aumento no valor nominal da função objetivo.

A contraparte robusta do problema nominal é dada por:

$$\min \quad \mathbf{c}^\top \mathbf{x} + \max_{\substack{S_0 \subseteq J_0 \\ |S_0| \leq \Gamma_0}} \left\{ \sum_{j \in S_0} d_j |x_j| \right\} \quad (2.15)$$

$$\text{s. a} \quad \sum_j a_{ij} x_j + \max_{\substack{S_i \subseteq J_i \\ |S_i| \leq \lfloor \Gamma_i \rfloor \\ t_i \in J_i \setminus S_i}} \left\{ \sum_{j \in S_i} \hat{a}_{ij} |x_j| + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{it_i} |x_{t_i}| \right\} \leq b_i, \quad \forall i, \quad (2.16)$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \quad (2.17)$$

$$x_i \in \mathbb{Z}, \quad \forall i = 1, \dots, k. \quad (2.18)$$

Teorema 2.6.2 *O Problema (2.15)-(2.18) possui uma formulação linearizada equi-*

valente dada por:

$$\begin{aligned}
\min \quad & \mathbf{c}^\top \mathbf{x} + z_0 \Gamma_0 + \sum_{j \in J_0} p_{0j} \\
s. \quad & a \quad \sum_j a_{ij} x_j + z_i \Gamma_i + \sum_{j \in J_i} p_{ij} \leq b_i, \quad \forall i, \\
& z_0 + p_{0j} \geq d_j y_j, \quad \forall j \in J_0, \\
& z_i + p_{ij} \geq \hat{a}_{ij} y_j, \quad \forall i \neq 0, j \in J_i, \\
& p_{ij} \geq 0, \quad \forall i, j \in J_i, \\
& y_j \geq 0, \quad \forall j, \\
& z_i \geq 0, \quad \forall i, \\
& -y_j \leq x_j \leq y_j, \quad \forall j, \\
& l_j \leq x_j \leq u_j, \quad \forall j, \\
& x_i \in \mathbb{Z}, \quad i = 1, \dots, k.
\end{aligned} \tag{2.19}$$

Prova. Primeiramente, mostra-se como modelar as restrições (2.16) como restrições lineares. Assim, dado um vetor $\mathbf{x}^*$, define-se:

$$\beta_i(\mathbf{x}^*) = \max_{\substack{S_i \cup \{t_i\} \\ S_i \subseteq J_i, |S_i| \leq \lfloor \Gamma_i \rfloor \\ t_i \in J_i \setminus S_i}} \left\{ \sum_{j \in S_i} \hat{a}_{ij} |x_j^*| + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{it_i} |x_{t_i}^*| \right\}.$$

Essa expressão é equivalente a:

$$\begin{aligned}
\beta_i(\mathbf{x}^*) = \max \quad & \sum_{j \in J_i} \hat{a}_{ij} |x_j^*| z_{ij} \\
\text{sujeito a} \quad & \sum_{j \in J_i} z_{ij} \leq \Gamma_i, \\
& 0 \leq z_{ij} \leq 1, \quad \forall i, j \in J_i.
\end{aligned} \tag{2.20}$$

O valor ótimo do Problema (2.20) consiste em $\lfloor \Gamma_i \rfloor$ variáveis iguais a 1 e uma variável igual a $\Gamma_i - \lfloor \Gamma_i \rfloor$. Isso é equivalente à seleção do subconjunto:

$$\{S_i \cup \{t_i\} \mid S_i \subseteq J_i, |S_i| \leq \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}$$

com função de custo correspondente dada por: $\sum_{j \in S_i} \hat{a}_{ij} |x_j^*| + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{it_i} |x_{t_i}^*|$.

Considere agora o dual do problema (2.20):

$$\begin{aligned}
 \min \quad & \sum_{j \in J_i} p_{ij} + \Gamma_i z_i \\
 \text{sujeito a} \quad & z_i + p_{ij} \geq \hat{a}_{ij} |x_j^*|, \quad \forall j \in J_i, \\
 & p_{ij} \geq 0, \quad \forall j \in J_i, \\
 & z_i \geq 0, \quad \forall i.
 \end{aligned} \tag{2.21}$$

Pelo Teorema da Dualidade Forte, como o Problema (2.20) é viável e limitado para todo $\Gamma_i \in [0, |J_i|]$, o problema dual (2.21) também é viável e limitado, e seus valores ótimos coincidem. Assim, $\beta_i(\mathbf{x}^*)$ é igual ao valor da função objetivo do Problema (2.21).

De modo análogo, a função objetivo (2.15) pode ser convertida em uma formulação linear da seguinte maneira:

$$\begin{aligned}
 \beta_0(\mathbf{x}^*) &= \max \left\{ \sum_{j \in S_0} d_j |x_j^*| \mid |S_0| \leq \Gamma_0, S_0 \subseteq J_0 \right\} \\
 &= \max \left\{ \sum_{j \in J_0} d_j |x_j^*| z_{0j} \mid \sum_{j \in J_0} z_{0j} \leq \Gamma_0, 0 \leq z_{0j} \leq 1, \forall j \in J_0 \right\} \\
 &= \min \left\{ \sum_{j \in J_0} p_{0j} + \Gamma_0 z_0 \mid z_0 + p_{0j} \geq d_j |x_j^*|, z_0 \geq 0, p_{0j} \geq 0, \forall j \in J_0 \right\}.
 \end{aligned}$$

Substituindo as maximizações internas do problema (2.15)-(2.18) pelas funções objetivo referentes aos seus respectivos duais e incorporando as restrições duais ao problema, obtemos o modelo equivalente (2.19). ■

A abordagem apresentada nesta Seção 2.6.3 é adotada nesta tese para formular o RSSCFLP (Capítulo 3).

Capítulo 3

Formulações matemáticas

Este capítulo apresenta a descrição matemática dos problemas abordados nesta tese. Está subdividido em duas seções. A Seção 3.1 apresenta a descrição e a formulação do SSCFLP. Já a Seção 3.2 apresenta a descrição e a formulação do RSSCFLP.

3.1 Descrição e formulação do SSCFLP

O SSCFLP consiste em determinar, dentre um conjunto de locais, em quais locais abrir facilidades para satisfazer as demandas dos clientes, considerando que cada facilidade possui capacidade de atendimento limitada e que a demanda de cada cliente deve ser atendida integralmente por uma única facilidade (condição de fonte única).

Para formular o SSCFLP, introduz-se a seguinte notação. Seja $J = \{1, 2, \dots, m\}$ o conjunto de locais disponíveis para a abertura de facilidades. Para cada local $j \in J$, uma facilidade com capacidade b_j pode ser aberta a um custo de abertura f_j . Seja também $I = \{1, 2, \dots, n\}$ o conjunto de clientes. Cada cliente $i \in I$ tem demanda d_i . O custo de alocar um cliente i em uma facilidade aberta no local j é dado por c_{ij} . Denote D a demanda total e considere as variáveis de decisão de alocação x_{ij} , para $i \in I$ e $j \in J$, e as variáveis de decisão de localização y_j , para $j \in J$, definidas como:

$$x_{ij} = \begin{cases} 1, & \text{se o cliente } i \text{ é alocado a uma facilidade no local } j; \\ 0, & \text{caso contrário.} \end{cases} \quad (3.1)$$

$$y_j = \begin{cases} 1, & \text{se uma facilidade é aberta no local } j; \\ 0, & \text{caso contrário.} \end{cases} \quad (3.2)$$

Usando a notação apresentada acima, o SSCFLP é formulado como o seguinte

problema de programação linear binária:

$$\min \sum_{j \in J} f_j y_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \quad (3.3)$$

$$\text{s.t. } \sum_{j \in J} x_{ij} = 1 \quad \forall i \in I \quad (3.4)$$

$$\sum_{i \in I} d_i x_{ij} \leq b_j y_j \quad \forall j \in J \quad (3.5)$$

$$x_{ij} \leq y_j \quad \forall i \in I, j \in J \quad (3.6)$$

$$\sum_{j \in J} b_j y_j \geq D \quad (3.7)$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I, j \in J \quad (3.8)$$

$$y_j \in \{0,1\} \quad \forall j \in J. \quad (3.9)$$

A expressão (3.3) apresenta o objetivo de minimizar os custos totais de abertura de facilidades (primeira componente) e de alocação de clientes a facilidades (segunda componente). As restrições (3.4) são chamadas de restrições de alocação. Elas garantem o respeito à condição de fonte única. Já as restrições (3.5), chamadas de restrições de capacidade, asseguram que os clientes só podem ser alocados em locais com facilidades abertas e que a demanda total atendida por cada facilidade não exceda sua capacidade. As restrições (3.6) também asseguram que os clientes só podem ser alocados em locais com facilidades abertas. Essas restrições, apesar de redundantes, fortalecem o limitante inferior obtido pela relaxação linear do problema, porém aumentam o número total de restrições de $m + n$ para $m + n + mn$. A restrição (3.7) garante que a capacidade total de facilidades abertas é suficiente para suprir a demanda total. Ela é redundante para o problema, mas é explorada nesta tese para obter desigualdades válidas que fortalecem sua formulação. As expressões (3.8) e (3.9) definem as variáveis de decisão como binárias.

O SSCFLP pode ser visto como uma variante do Problema Capacitado de Localização de Facilidades (*Capacitated Facility Location Problem* - CFLP). A diferença entre o CFLP e o SSCFLP é ausência da condição do fonte única no CFLP. A formulação do CFLP é dada por:

$$\min \sum_{j \in J} f_j y_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$\text{s.a: } (3.4) - (3.7)$$

$$x_{ij} \in [0,1] \quad \forall i \in I, j \in J$$

$$y_j \in \{0,1\} \quad \forall j \in J.$$

O SSCFLP é considerado mais desafiador de resolver do que o CFLP, uma vez que, quando a localização das facilidades abertas é conhecida, o subproblema de alocar demandas de clientes a facilidades abertas no CFLP é um problema de programação linear, que pode ser resolvido com eficiência por *solvers*. Por outro lado, no SSCFLP, o subproblema resultante é um Problema de Atribuição Generalizada (*Generalized Assignment Problem* - GAP), que é um problema NP-difícil (Fisher et al., 1986). Se $J' \subset J$ é o conjunto dos locais com facilidades abertas, o GAP é formulado pelas sentenças:

$$\begin{aligned}
 & \min \sum_{i \in I} \sum_{j \in J'} c_{ij} x_{ij} \\
 & \text{s.t.} \sum_{j \in J'} x_{ij} = 1 & \forall i \in I \\
 & \sum_{i \in I} d_i x_{ij} \leq b_j & \forall j \in J' \\
 & x_{ij} \in \{0,1\} & \forall i \in I, j \in J'.
 \end{aligned}$$

A relaxação linear do SSCFLP consiste no problema obtido considerando as variáveis de decisão como contínuas, e pode ser formulada pelas sentenças:

$$\begin{aligned}
 & \min \sum_{j \in J} f_j y_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \\
 & \text{s.t.} \quad (3.4) - (3.7) \\
 & x_{ij} \in [0,1] & \forall i \in I, j \in J \\
 & y_j \in [0,1] & \forall j \in J.
 \end{aligned}$$

No estudo de métodos para resolução do SSCFLP, a relaxação linear, o CFLP e o GAP desempenham papéis importantes. A solução da relaxação linear, apesar de fornecer um limitante inferior considerado como fraco para o problema, fornece custos reduzidos de cada variável, que podem ser utilizados para a identificação de variáveis promissoras. Já a solução do CFLP produz um limitante inferior mais forte do que a relaxação linear e pode ser utilizada para identificar locais promissores. O GAP, por sua vez, pode ser utilizado para encontrar bons limitantes superiores ou a solução ótima, quando se fixam como abertas facilidades em um subconjunto de locais disponíveis do problema.

Exemplo 3.1.1 (Diferença entre o CFLP e o SSCFLP) Para ilustrar a dife-

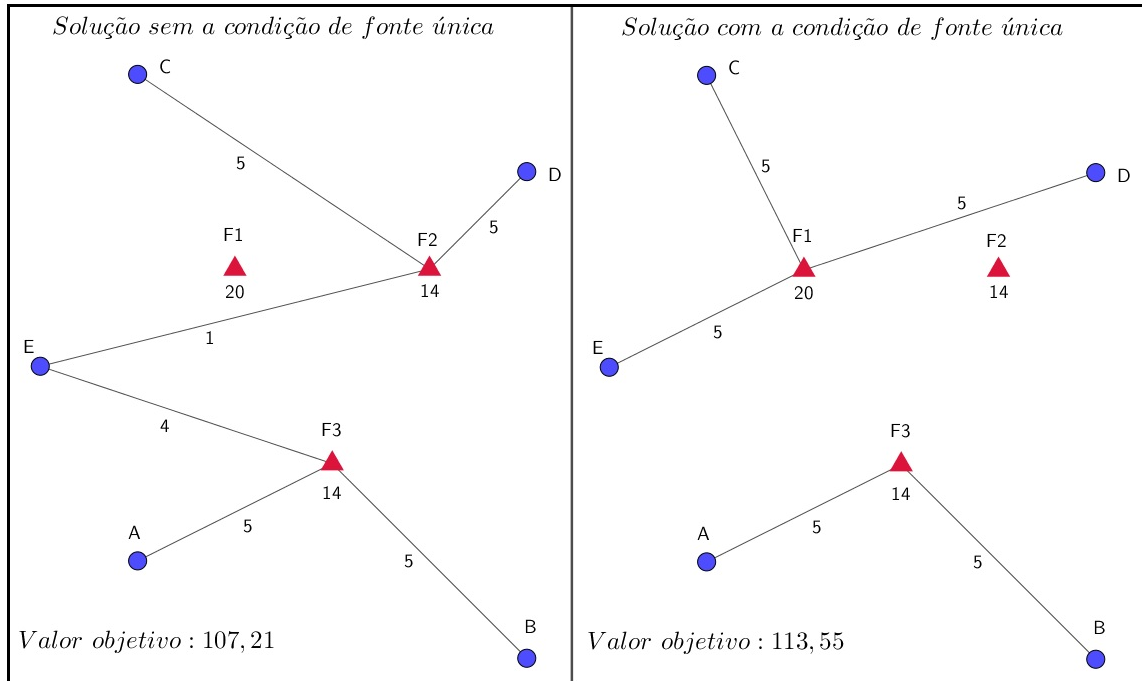


Figura 3.1: Comparação da solução sem e com a condição de fonte única.

rença entre o CFLP e o SSCFLP, considere uma instância em que os clientes são os pontos do plano cartesiano $A = (2,2)$, $B = (6,1)$, $C = (2,7)$, $D = (6,6)$ e $E = (1,4)$, todos com demanda igual a 5. Os locais disponíveis são os pontos $F1 = (3,5)$, $F2 = (5,5)$ e $F3 = (4,3)$, com capacidades de 20, 14 e 14, respectivamente, e custos de abertura de 30, 20 e 20, respectivamente. Os custos de alocação de clientes em facilidades foram considerados iguais às distâncias euclidianas entre esses pontos no plano cartesiano.

A Figura 3.1 apresenta geometricamente as soluções ótimas do problema para esta instância, sem considerar a condição de fonte única (CFLP) e considerando a condição de fonte única (SSCFLP). Nela, os clientes são representados por círculos e os locais potenciais por regiões triangulares. Uma aresta que liga um cliente a uma facilidade indica que uma quantidade de demanda do cliente, representada pelo número próximo ao ponto médio da aresta, é atendida por uma facilidade aberta naquele local. Os números abaixo das regiões triangulares indicam a capacidade das facilidades correspondentes

Observa-se que, na solução que não considera a condição de fonte única, a demanda do cliente E foi dividida e atendida pelas facilidades abertas em F2 e F3, afim de reduzir custos. Com a condição de fonte única, esta divisão é proibida. Isto alterou consideravelmente a solução, modificando inclusive os conjuntos de locais com facilidades abertas.

O exemplo ilustra que a adoção da condição de fonte única aumenta o custo da

solução em relação a modelos que permitem a divisão da demanda. Entretanto, sua adoção não busca necessariamente minimizar o custo operacional, mas representar situações nas quais a divisão da demanda entre diferentes facilidades é indesejável ou nas quais sua restrição favorece a redução de custos associados a problemas secundários. Por exemplo, no problema de alocação de estudantes em escolas primárias abordado em [Kong \(2021\)](#), a demanda por atendimento escolar é agregada em microrregiões e é considerada indivisível. Nesse contexto, a atribuição de cada microrregião a uma única escola primária pode reduzir custos associados ao transporte escolar. Além disso, pode simplificar aspectos administrativos e políticos relacionados à definição das áreas de atendimento de cada escola primária. Esse exemplo evidencia que a adoção da condição de fonte única deve ser avaliada de acordo com o contexto da aplicação, ponderando-se o aumento de custo decorrente dessa condição e as vantagens por ela proporcionadas.

3.2 Descrição e formulação do RSSCFLP

O RSSCFLP é uma variante do SSCFLP em que os parâmetros do problema estão sujeitos a incertezas. Nesta tese, o RSSCFLP é modelado com base na abordagem de otimização robusta para problemas discretos proposta por [Bertsimas & Sim \(2003, 2004\)](#), que permite controlar o grau de proteção contra incertezas por meio de parâmetros de robustez.

Considere a seguinte notação. Sejam I o conjunto de clientes e J o conjunto de locais para a abertura de facilidades. Em cada local j , pode-se abrir uma facilidade com capacidade no intervalo $[b_j - \hat{b}_j, b_j + \hat{b}_j]$, por um custo fixo de abertura no intervalo $[f_j, f_j + \hat{f}_j]$. Além disso, cada cliente $i \in I$ possui uma demanda no intervalo $[d_i - \hat{d}_i, d_i + \hat{d}_i]$ e um custo de alocação no intervalo $[c_{ij}, c_{ij} + \hat{c}_{ij}]$ para cada facilidade j . Os valores de b_j , f_j , $\forall j \in J$, são chamados de valores nominais da capacidade e do custo fixo de abertura de uma facilidade aberta em j , respectivamente, e $\hat{b}_j$ e $\hat{f}_j$ seus respectivos desvios máximos. O valor de d_i , $\forall i \in I$, é chamado de valor nominal da demanda de cada cliente i e $\hat{d}_i$ o seu respectivo desvio máximo. Por fim, o valor de c_{ij} , $\forall i \in I, j \in J$, é chamado de valor nominal do custo de alocação do cliente i na facilidade j , e $\hat{c}_{ij}$ é o respectivo desvio máximo.

Os níveis de robustez Γ^c e Γ^f são estabelecidos para controlar o grau de proteção contra incerteza nos custos de alocação de clientes e de abertura de facilidades, respectivamente. Os valores de Γ^c e Γ^f são inteiros não negativos que podem ser, no máximo, iguais às quantidades positivas de $\hat{c}_{ij}$, $i \in I, j \in J$, e de $\hat{f}_j$, $j \in J$, respectivamente. Além disso, para cada $j \in J$, os níveis de robustez Γ_j^d e Γ_j^b são usados para controlar o grau de proteção contra incerteza nas demandas dos clientes

e na capacidade da facilidade j . Para cada $j \in J$, Γ_j^d é um valor não negativo que pode ser, no máximo, igual às quantidades positivas de $\hat{d}_i$. Já $\hat{b}_j$ é um número não negativo menor ou igual a 1.

Com esta notação, uma solução do RSSCFLP deve estar protegida contra o pior caso possível para variações de até:

- (i) Γ^c parâmetros de custos de alocação em relação aos valores nominais c_{ij} ;
- (ii) Γ^f parâmetros de custos fixos de abertura em relação aos valores nominais f_j ;
- (iii) $\lfloor \Gamma_j^d \rfloor$ parâmetros de demanda em relação aos valores nominais d_i e de um parâmetro de demanda variando no máximo $(\Gamma_j^d - \lfloor \Gamma_j^d \rfloor)\hat{d}_t$, para $t \in I$, em relação ao valor nominal d_t , para cada facilidade $j \in J$;
- (iv) Γ_j^b é a fração do desvio máximo da capacidade b_j de cada facilidade $j \in J$.

Considere as variáveis de decisão de localização $y_j \in \{0,1\}$, $\forall j \in J$, de modo que $y_j = 1$ se uma facilidade é aberta no local j e $y_j = 0$, caso contrário; e variáveis de decisão de alocação $x_{ij} \in \{0,1\}$, $\forall i \in I, j \in J$, tais que $x_{ij} = 1$ se o cliente i é alocado à facilidade j e $x_{ij} = 0$, caso contrário. Adicionalmente, considere a matriz $\mathbf{x} = (x_{ij})_{i \in I, j \in J}$ e os vetores $\mathbf{x}_j = (x_{ij})_{i \in I}$, $\forall j \in J$, e $\mathbf{y} = (y_j)_{j \in J}$ de variáveis de decisão. Então, o RSSCFLP pode ser formulado pelas sentenças:

$$\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} f_j y_j + \beta^c(\mathbf{x}, \Gamma^c) + \beta^f(\mathbf{y}, \Gamma^f) \quad (3.10)$$

$$\text{sujeito a } \sum_{i \in I} d_i x_{ij} + \beta_j^d(\mathbf{x}_j, \Gamma_j^d) \leq b_j y_j - \beta_j^b(y_j, \Gamma_j^b) \quad \forall j \in J. \quad (3.11)$$

$$\sum_{j \in J} x_{ij} = 1 \quad \forall i \in I \quad (3.12)$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I, j \in J \quad (3.13)$$

$$y_j \in \{0,1\} \quad \forall j \in J. \quad (3.14)$$

em que:

$$\beta^c(\mathbf{x}, \Gamma^c) = \max_{\{S | S \subseteq I \times J; |S| \leq \Gamma^c\}} \left\{ \sum_{(i,j) \in S} \hat{c}_{ij} x_{ij} \right\}; \quad (3.15)$$

$$\beta^f(\mathbf{y}, \Gamma^f) = \max_{\{S | S \subseteq J; |S| \leq \Gamma^f\}} \left\{ \sum_{j \in S} \hat{f}_j y_j \right\}; \quad (3.16)$$

$$\beta_j^d(\mathbf{x}_j, \Gamma_j^d) = \max_{\{S \cup \{t\} | S \subseteq I; |S| \leq \Gamma_j^d; t \in I \setminus S\}} \left\{ \sum_{i \in S} \hat{d}_i x_{ij} + (\Gamma_j^d - \lfloor \Gamma_j^d \rfloor) \hat{d}_t x_{tj} \right\} \quad \forall j \in J; \quad (3.17)$$

$$\beta_j^b(y_j, \Gamma_j^b) = \Gamma_j^b \hat{b}_j y_j \quad (3.18)$$

A função objetivo (3.10) minimiza o *custo garantido total*, definido como a soma dos custos nominais de alocação e de abertura de facilidades, com as respectivas variações adversas máximas associadas às incertezas e aos níveis de robustez considerados. As restrições (3.11) asseguram que, para cada facilidade, a capacidade subtraída do valor máximo possível de sua variação deve ser suficiente para suprir a demanda total dos clientes atendidos, mais o valor máximo possível da variação total das demandas. Ao mesmo tempo, essas restrições garantem que apenas locais com facilidades abertas podem atender aos clientes. As restrições (3.12) asseguram que cada cliente deve ser alocado exatamente a uma facilidade. Se a incerteza nos parâmetros não for considerada, os valores de β^c , β^f , β_j^d ($j \in J$) e β_j^b ($j \in J$) são todos nulos, e a formulação acima reduz-se à formulação clássica do SSCFLP, referida como problema nominal.

A linearização de β^c , β_j^d e β^f é obtida seguindo os mesmos passos da demonstração do Teorema 1 de Bertsimas & Sim (2003). Os principais passos são descritos a seguir. Considere que x_{ij} e y_j assumem valores x_{ij}^* ($\forall i \in I$ e $j \in J$) e y_j^* ($\forall j \in J$), respectivamente, e $\mathbf{x}^* = (x_{ij}^*)_{i \in I, j \in J}$, $\mathbf{x}_j^* = (x_{ij}^*)_{i \in I}$, $\forall j \in J$, e $\mathbf{y}^* = (y_j^*)_{j \in J}$. Então:

$$\beta^c(\mathbf{x}^*, \Gamma^c) = \max \left\{ \sum_{i \in I} \sum_{j \in J} \hat{c}_{ij} x_{ij}^* z_{ij} : \sum_{i \in I} \sum_{j \in J} z_{ij} \leq \Gamma^c, z_{ij} \in [0,1] \forall i \in I, j \in J \right\} \quad (3.19)$$

$$\beta^f(\mathbf{y}^*, \Gamma^f) = \max \left\{ \sum_{j \in J} \hat{f}_j y_j^* z_j : \sum_{j \in J} z_j \leq \Gamma^f, z_j \in [0,1] \forall j \in J \right\} \quad (3.20)$$

$$\beta_j^d(\mathbf{x}_j^*, \Gamma_j^d) = \max \left\{ \sum_{i \in I} \hat{d}_i x_{ij}^* z_i : \sum_{i \in I} z_i \leq \Gamma_j^d, z_i \in [0,1] \forall i \in I \right\} \quad \forall j \in J \quad (3.21)$$

Os problemas duais dos três problemas de maximização acima são dados, respectivamente, por:

$$\min \left\{ \lambda^c \Gamma^c + \sum_{i \in I} \sum_{j \in J} \mu_{ij}^c : \lambda^c + \mu_{ij}^c \geq \hat{c}_{ij} x_{ij}^*, \lambda^c \geq 0, \mu_{ij}^c \geq 0, \forall i \in I, j \in J \right\} \quad (3.22)$$

$$\min \left\{ \lambda^f \Gamma^f + \sum_{j \in J} \mu_j^f : \lambda^f + \mu_j^f \geq \hat{f}_j y_j^*, \lambda^f \geq 0, \mu_j^f \geq 0, \forall j \in J \right\} \quad (3.23)$$

$$\min \left\{ \lambda_j^d \Gamma_j^d + \sum_{i \in I} \mu_{ij}^d : \lambda_j^d + \mu_{ij}^d \geq \hat{d}_i x_{ij}^*, \lambda_j^d \geq 0, \mu_{ij}^d \geq 0, \forall i \in I \right\} \quad \forall j \in J \quad (3.24)$$

Por conta da dualidade forte, o valor ótimo da função objetivo dos problemas de

minimização das sentenças (3.22), (3.23) e (3.24) é equivalente ao dos problemas de maximização das sentenças (3.19), (3.20) e (3.21). Portanto, o problema (3.10)-(3.14) para o RSSCFLP pode ser reformulado pelo seguinte problema:

$$\begin{aligned}
& \min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} f_j y_j + \lambda^c \Gamma^c + \sum_{i \in I} \sum_{j \in J} \mu_{ij}^c + \lambda^f \Gamma^f + \sum_{j \in J} \mu_j^f \\
& \text{sujeito a } \sum_{i \in I} d_i x_{ij} + \lambda_j^d \Gamma_j^d + \sum_{i \in I} \mu_{ij}^d \leq b_j y_j - \Gamma_j^b \hat{b}_j y_j & \forall j \in J \\
& \sum_{j \in J} x_{ij} = 1 & \forall i \in I \\
& \lambda^c + \mu_{ij}^c \geq \hat{c}_{ij} x_{ij} & \forall i \in I, j \in J \\
& \lambda^f + \mu_j^f \geq \hat{f}_j y_j & \forall j \in J \\
& \lambda_j^d + \mu_{ij}^d \geq \hat{d}_i x_{ij} & \forall i \in I, j \in J \\
& \lambda^c \geq 0, \mu_{ij}^c \geq 0 & \forall i \in I, j \in J \\
& \lambda^f \geq 0, \mu_j^f \geq 0 & \forall j \in J \\
& \lambda_j^d \geq 0, \mu_{ij}^d \geq 0 & \forall i \in I, j \in J \\
& x_{ij} \in \{0,1\} & \forall i \in I, j \in J \\
& y_j \in \{0,1\} & \forall j \in J.
\end{aligned}$$

Enquanto a formulação do SSCFLP dada pelas sentenças (3.3)-(3.9), desconsiderando as restrições redundantes das sentenças (3.6) e (3.7), possui $m \times n + m$ variáveis e $m + n$ restrições, a formulação linearizada do RSSCFLP possui $m + m \times n + 2(m \times n + m + 1)$ variáveis e $m + n + 2(m \times n) + m$ restrições. Portanto, a incorporação da robustez e sua consequente linearização aumentam significativamente o número de variáveis e de restrições. Em contrapartida, por se tratar de um problema de programação linear inteira mista, técnicas exatas e heurísticas amplamente utilizadas nessa classe de problemas podem ser empregadas para a resolução do modelo do RSSCFLP.

Capítulo 4

Trabalhos Relacionados

Este capítulo apresenta os trabalhos relacionados aos problemas abordados nesta tese. As Seções 4.1 e 4.2 apresentam as revisões da literatura relacionadas ao SSCFLP e ao RSSCFLP. A Seção 4.3 apresenta a descrição de algoritmos existentes na literatura que inspiraram os métodos propostos nesta tese.

4.1 Trabalhos relacionados ao SSCFLP

O SSCFLP é um problema amplamente estudado na literatura. A maior parte das abordagens para a sua resolução é heurística, com prevalência de matheurísticas. Boa parte das abordagens heurísticas utiliza procedimentos baseados em meta-heurísticas. Em menor quantidade, também foram propostos métodos exatos para resolver o SSCFLP. A maior parte deles combina a construção de árvores de busca binária com heurísticas.

Dentre as matheurísticas propostas, a maior parte explora relaxações lagrangeanas. [Barceló & Casanovas \(1984\)](#), [Sridharan \(1991\)](#), [Hindi & Pieńkosz \(1999\)](#), [Cortinhal & Captivo \(2003\)](#), [Chen & Ting \(2008\)](#) e [Oliveira et al. \(2021\)](#) propuseram relaxar as restrições de alocação (3.12) da formulação do SSCFLP apresentada na Seção 3.1 do Capítulo 3. [Barceló & Casanovas \(1984\)](#) propuseram uma heurística que primeiro seleciona locais para a abertura de facilidades e, depois, aloca os clientes. Nela, aplica-se uma busca local para refinar os procedimentos em ambas as fases. [Sridharan \(1991\)](#) explorou iterativamente subproblemas de alocação de clientes a locais com facilidades abertas, selecionados a partir de relaxações. [Hindi & Pieńkosz \(1999\)](#) propuseram uma busca local para a alocação de clientes, utilizando um procedimento de Busca em Vizinhança Restrita. [Cortinhal & Captivo \(2003\)](#) utilizaram heurísticas baseadas em buscas locais e um procedimento baseado na meta-heurística Busca Tabu para encontrar boas soluções. Já [Chen & Ting \(2008\)](#)

propuseram explorar soluções por meio de relaxações, utilizando um procedimento baseado na meta-heurística Colônia de Formigas. Por outro lado, [Klincewicz & Luss \(1986\)](#) e [Caserta & Voß \(2020\)](#) propuseram relaxar as restrições de capacidade (3.5) da formulação do SSCFLP. [Klincewicz & Luss \(1986\)](#) propuseram resolver as relaxações utilizando um procedimento de ascensão dual e aplicaram uma busca local para encontrar soluções. Já [Caserta & Voß \(2020\)](#) propuseram utilizar uma heurística baseada no *framework* Método do Corredor.

Por sua vez, outras matheurísticas existentes exploram soluções de relaxações lineares ao invés de relaxações lagrangeanas. [Daskin & Jones \(1993\)](#) desenvolveram um algoritmo que identifica alocações promissoras a partir da solução da relaxação linear e utiliza busca local para construir soluções de boa qualidade. [Guastaroba & Speranza \(2014\)](#) propuseram um algoritmo baseado no *framework* *Kernel Search*, que também explora relaxações lineares para identificar localizações e alocações promissoras. [Kong \(2021\)](#) desenvolveram uma busca local para o SSCFLP que resolve, iterativamente, subproblemas menores que o problema original. Cada subproblema é composto por um subconjunto de clientes e de locais que é resolvido por um *solver*. Nessa abordagem, a relaxação linear é utilizada para gerar uma solução inicial. Por último, [Bakker et al. \(2025\)](#) combinaram uma técnica de reconhecimento de padrões para identificar regiões promissoras com uma heurística baseada em *Kernel Search*.

Há também abordagens heurísticas que não recorrem a técnicas de programação matemática. A maior parte delas aplica *frameworks* de meta-heurísticas. [Delmaire et al. \(1999\)](#) desenvolveram e compararam duas abordagens híbridas. A primeira é um procedimento Busca Tabu que utiliza um procedimento GRASP para diversificar a busca por soluções. A segunda é um procedimento GRASP Reativo, que utiliza meta-heurística Busca Tabu para refinar soluções. [Cortinhal & Captivo \(2004\)](#) propuseram procedimentos baseados em Algoritmos Genéticos para o SSCFLP. [Ho \(2015\)](#) também utilizou um procedimento Busca Tabu para refinar soluções em um algoritmo Busca Tabu Iterada. [Qin et al. \(2012\)](#) combinaram, iterativamente, dois procedimentos de Recozimento Simulado (*Simulated Annealing*): um para determinar os locais de abertura de facilidades e outro para alocar clientes ao conjunto de facilidades abertas. [Contreras & Díaz \(2008\)](#) apresentaram um procedimento de Busca Dispersa (*Scatter Search*) que utiliza meta-heurística Busca Tabu para refinar soluções. Por fim, [Xu et al. \(2025\)](#) propuseram um algoritmo que combina um procedimento baseado na meta-heurística *Crow Search Algorithm* com resolução de modelos para explorar o espaço de soluções.

Outras duas abordagens heurísticas revisadas são buscas locais que exploram a teoria de grafos. [Ahuja et al. \(2004\)](#) propuseram um algoritmo *Very Large-Scale Neighborhood Search*, em que grafos são utilizados para detectar vizinhanças de me-

lhora. [Tran et al. \(2017\)](#) propuseram aprimorar esse algoritmo adotando novos procedimentos baseados em hipergrafos.

Dentre os métodos exatos, [Neebe & Rao \(1983\)](#) reformularam o SSCFLP como um problema de partição de conjuntos e propuseram um procedimento B&B para resolvê-lo. Já [Holmberg et al. \(1999\)](#) propuseram um procedimento B&B em que relaxações lagrangeanas, combinadas com o método subgradiente, são utilizadas para obter limitantes inferiores. Além disso, procedimentos heurísticos *Repeat Matching* são utilizados para obter limitantes superiores. A estratégia *Branch-and-Price* (B&P) também já foi utilizada para a resolução do SSCFLP no algoritmo proposto por [Díaz & Fernández \(2002\)](#). Nele, em cada iteração, são considerados dois níveis de otimização. O primeiro nível gera locais para a abertura de facilidades, e o segundo nível consiste no subproblema de alocar clientes às facilidades abertas, que é resolvido por meio de um procedimento de geração de colunas. [Yang et al. \(2012\)](#) propuseram um algoritmo baseado no *framework cut-and-solve*. Nele, uma árvore binária de busca é construída explorando dois tipos de subproblemas: um para fornecer limitantes superiores e outro para fornecer limitantes inferiores e selecionar variáveis para ramificar a árvore. Procedimentos de planos de corte são incorporados para fortalecer as formulações dos subproblemas. Além disso, um procedimento Busca Tabu é aplicado para fornecer limites superiores. [Gadegaard et al. \(2018\)](#) propuseram um algoritmo em três fases que aprimora o de [Yang et al. \(2012\)](#). Na primeira fase, um procedimento de planos de corte é utilizado para fortalecer a formulação do SSCFLP; na segunda fase, um procedimento de *Local Branching* fornece um limite superior. A terceira fase aplica o esquema *cut-and-solve*.

A Tabela 4.1 resume as principais abordagens para a resolução do SSCFLP. Cada estudo é classificado como heurístico (H) ou exato (E). Os métodos heurísticos ainda são classificados como matheurísticas (MATH) ou meta-heurísticas (META), enquanto os métodos exatos são categorizados de acordo com sua estratégia principal. Os estudos são apresentados em ordem cronológica de publicação.

De acordo com a Tabela 4.1, a maioria das abordagens propostas para o SSCFLP é heurística, com predominância de matheurísticas. Métodos exatos aparecem com menor frequência e, desde o trabalho de [Yang et al. \(2012\)](#), observa-se a predominância do uso do esquema *cut-and-solve* em seu desenvolvimento. As duas últimas linhas da tabela apresentam os algoritmos HILS e NCS propostos nesta tese. O HILS é classificado de forma semelhante à maior parte dos trabalhos revisados, enquanto o NCS segue a tendência recente de empregar a estratégia *cut-and-solve* em métodos exatos.

Tabela 4.1: Visão geral da revisão da literatura: principais abordagens para a resolução do SSCFLP.

Trabalho	E	H	MATH	META	B&B	B&P	CS	Estratégias
Neebe & Rao (1983)	✓	-	-	-	✓	-	-	Geração de colunas
Barceló & Casanovas (1984)	-	✓	✓	-	-	-	-	RLG, Busca local
Klincewicz & Luss (1986)	-	✓	✓	-	-	-	-	RLG, Busca local
Sridharan (1991)	-	✓	✓	-	-	-	-	RLG
Daskin & Jones (1993)	-	✓	✓	-	-	-	-	RLI, Busca local
(Holmberg et al., 1999)	✓	-	-	-	✓	-	-	RLG, <i>Repeat matching</i>
Hindi & Pieńkosz (1999)	-	✓	✓	-	-	-	-	RLG, Busca local
Delmaire et al. (1999)	-	✓	-	✓	-	-	-	Busca Tabu GRASP Reativo
Díaz & Fernández (2002)	✓	-	-	-	-	✓	-	Geração de colunas
Cortinhal & Captivo (2003)	-	✓	✓	✓	-	-	-	RLG, Busca Tabu
Cortinhal & Captivo (2004)	-	✓	-	✓	-	-	-	Algoritmo Genético
Contreras & Díaz (2008)	-	✓	-	✓	-	-	-	Busca Dispersa
Chen & Ting (2008)	-	✓	✓	✓	-	-	-	RLG, Colônia de Formigas
Qin et al. (2012)	-	✓	-	✓	-	-	-	Recozimento Simulado
Yang et al. (2012)	✓	-	-	✓	-	-	✓	Busca Tabu, Planos de Corte
Ahuja et al. (2004)	-	✓	-	-	-	-	-	Busca local
Guastaroba & Speranza (2014)	-	✓	✓	-	-	-	-	<i>Kernel Search</i>
Ho (2015)	-	✓	-	✓	-	-	-	Busca Tabu Iterada
Tran et al. (2017)	-	✓	-	-	-	-	-	Busca local
Gadegaard et al. (2018)	✓	-	✓	-	-	-	✓	<i>Local branching</i> Planos de Corte
Caserta & Voß (2020)	-	✓	✓	-	-	-	-	RLG, Método do Corredor
Oliveira et al. (2021)	-	✓	✓	✓	-	-	-	RLG, Busca Tabu
Kong (2021)	-	✓	✓	-	-	-	-	RLI, Busca local
Xu et al. (2025)	-	✓	-	✓	-	-	-	<i>Crow Search Algorithm</i>
Bakker et al. (2025)	-	✓	✓	-	-	-	-	<i>Kernel Search</i>
HILS	-	✓	✓	-	-	-	-	ILS
NCS	✓	-	-	-	-	-	✓	Busca local Planos de corte

4.2 Trabalhos relacionados ao RSSCFLP

A revisão realizada nesta tese identificou apenas os trabalhos de [Ryu & Park \(2022\)](#) e [Lin \(2009\)](#) que tratam especificamente de variantes do SSCFLP com dados incertos. Em contrapartida, verifica-se que, de modo geral, a literatura sobre problemas de localização com incerteza é bastante ampla. Diante desse cenário, a presente revisão considera, além desses dois trabalhos, estudos relacionados a outros problemas de localização sob incerteza, priorizando publicações recentes.

Os trabalhos de [Lin \(2009\)](#) e [Ryu & Park \(2022\)](#) consideram incertezas apenas nas demandas dos clientes. Em [Lin \(2009\)](#), o problema é modelado por meio de restrições probabilísticas, que incorporam requisitos de nível de serviço para as facilidades. Nesse modelo, cada facilidade deve atender à demanda com uma probabilidade espe-

cífica. Para resolver o problema, os autores utilizaram o algoritmo de [Holmberg et al. \(1999\)](#) para distribuições de Poisson e combinaram uma heurística lagrangeana com *branch-and-bound* para distribuições normais. Já [Ryu & Park \(2022\)](#) modelaram o problema utilizando a abordagem de otimização robusta de [Bertsimas & Sim \(2003\)](#). Em vez de resolver diretamente o modelo de programação inteira mista obtido, os autores reformularam o problema por meio da decomposição de Dantzig–Wolfe e aplicaram um procedimento *branch-and-price*.

Dentre os problemas de localização mais gerais sujeitos a incertezas, a maioria considera que eles estão presentes apenas nas demandas. [Daskin et al. \(1997\)](#) propuseram um modelo baseado em cenários para o problema das p -medianas, considerando incerteza nas demandas. Nele, o objetivo é minimizar o arrependimento (*regret*). [Baron et al. \(2011\)](#) estudaram o problema de localização multiperíodo com incerteza na demanda, utilizando uma abordagem robusta com diferentes conjuntos de incerteza, incluindo regiões do tipo caixa e elipsoides. [Albareda-Sambola et al. \(2011\)](#) estudaram o problema de localização capacitado, considerando demandas modeladas como variáveis aleatórias de Bernoulli. Nesse modelo, é permitido exceder a capacidade das facilidades mediante o pagamento dos custos de terceirização. Os autores propuseram um modelo estocástico em dois estágios para minimizar os custos totais. [Bieniek \(2015\)](#) propuseram um modelo estocástico para o problema de localização não capacitado com demandas incertas. Eles desenvolveram expressões analíticas para as funções de recursos. [Saif & Delage \(2021\)](#) estudaram versões de um e dois estágios do problema de localização capacitado com incerteza nas demandas, utilizando a abordagem de otimização robusta sob incerteza na distribuição (*distributionally robust optimization*). Para resolver os modelos, os autores propuseram procedimentos baseados em geração de colunas. [Campos et al. \(2025\)](#) estudaram uma variante robusta do problema das p -medianas capacitado, considerando incerteza nas demandas. Os autores utilizaram a abordagem de [Bertsimas & Sim \(2003\)](#) e introduziram um *framework* geral para modelar conjuntos poliédricos de incerteza. Para resolver o problema, foram desenvolvidos algoritmos baseados em B&C, *branch-and-price* e VNS.

Há também trabalhos que consideram incertezas em parâmetros distintos da demanda. [Louveaux & Peeters \(1992\)](#) propuseram um modelo estocástico em dois estágios, com recursos, para um problema de localização com incertezas nas demandas, nos custos e nas receitas. O modelo foi resolvido por meio de um procedimento que explora a formulação do dual. [Armas et al. \(2016\)](#) abordaram o problema não capacitado com fonte única, considerando incerteza nos custos de alocação. Os autores propuseram uma abordagem que combina simulação e heurísticas (*simheuristic*), aplicando um procedimento ILS. [Affy et al. \(2021\)](#) trataram de um problema capaci-

tado que considera facilidades que podem falhar, perdendo parte de suas capacidades e facilidades fortificadas (que não estão sujeitas a falhas). Nesse modelo, cada cliente deve ser alocado a uma facilidade principal e a uma de *backup*. Os autores propuseram uma formulação linearizada e um método iterativo para alocação do orçamento destinado a facilidades fortificadas. Koca (2023) consideraram um problema de localização capacitado com incertezas nas demandas, nos custos de alocação e nas taxas de falha das facilidades. O modelo foi formulado como um problema de programação estocástica em dois estágios e resolvido pelo método *L-shaped*. Cheng et al. (2024) abordaram um problema de localização capacitado em dois estágios com incertezas nas demandas e nas capacidades das facilidades. O modelo foi formulado utilizando otimização robusta sob incerteza na distribuição. Foi considerado um conjunto de ambiguidade por cenário (*scenario-wise ambiguity set*), capaz de capturar correlações entre incertezas de diferentes tipos de parâmetros. A formulação resultante corresponde a um problema de programação inteira misto, que foi resolvido por um *solver* comercial. Por fim, em Pantuso (2025), é proposto um modelo geral de programação estocástica em dois estágios para problemas de localização, considerando incerteza dependente das decisões. Nele, as distribuições de probabilidade dos parâmetros dependem diretamente das decisões de localização. Para resolver o modelo, foi proposto um algoritmo de decomposição exato baseado no método *L-shaped*.

De maneira geral, essa revisão evidencia que são escassos os estudos que tratam simultaneamente de múltiplas fontes de incerteza em problemas de localização. Essa lacuna é particularmente relevante, uma vez que, em aplicações reais, a incerteza é inerente aos dados de entrada.

4.3 Trabalhos que inspiraram os métodos propostos

Os algoritmos KS, *Three-Phase Algorithm* (TP-ALG) e MH propostos em Guastaroba & Speranza (2014), Gadegaard et al. (2018) e Kong (2021) foram revistos com mais profundidade, pois eles serviram de inspiração para os algoritmos desenvolvidos neste trabalho.

4.3.1 Algoritmo KS

O algoritmo KS utiliza os custos reduzidos das variáveis na solução da relaxação linear do problema original para identificar locais e alocações promissores. Inicialmente, forma-se um *kernel*, composto por um conjunto de variáveis promissoras. Parte das variáveis não promissoras é subdividida em uma sequência de *buckets*, e a outra é eliminada da busca. Definidos o *kernel* e a sequência de *buckets*, o subpro-

blema que inclui apenas variáveis do *kernel* é resolvido para fornecer um limitante superior para o SSCFLP. Inicia-se, então, o processo iterativo. Em cada iteração, o *kernel* é perturbado, forçando a inclusão de variáveis do *bucket* corrente. Se houver melhora na solução, o *kernel* é atualizado. O processo é repetido até que todos os *buckets* sejam explorados. Os resultados mostraram que o algoritmo é muito eficiente para instâncias de pequeno e médio porte. Porém, em instâncias mais desafiadoras, o algoritmo obteve soluções de qualidade razoável, mas com alto custo computacional. O principal gargalo do método é que variáveis podem ser eliminadas precocemente da busca, sem chance de retorno antes do processo iterativo. Em instâncias mais difíceis, variáveis sumariamente eliminadas podem permanecer ativas na solução ótima do problema.

4.3.2 Algoritmo MH

O algoritmo MH inicialmente constrói uma solução explorando a solução da relaxação linear parcial do problema original. Essa relaxação apenas relaxa as condições de integralidade das variáveis x_{ij} . Resolvida a relaxação linear parcial, a solução inicial é construída, alocando cada cliente i à facilidade j para a qual x_{ij} assume o maior valor. Essa solução geralmente é infactível, por desconsiderar as capacidades no processo de alocação. O algoritmo MH permite a exploração de soluções infactíveis, penalizando a infactibilidade na avaliação das soluções.

Construída a solução inicial, inicia-se o processo iterativo do MH. Em cada iteração, um subproblema, que é um SSCFLP de dimensão bem menor que o original, é resolvido. Em seguida, constrói-se uma nova solução, mesclando a solução corrente com a do subproblema. A construção dos subproblemas baseia-se em *clusterização*, isto é, no agrupamento de objetos semelhantes, chamados de *clusters*. No MH, os *clusters* são subconjuntos de clientes $I^* \subset I$ e de locais $J^* \subset J$, supostamente próximos entre si. A formação dos *clusters* explora a configuração de clientes alocados em facilidades na solução corrente e os valores de c_{ij} . Mais precisamente, em cada iteração, é selecionado aleatoriamente um cliente $i^* \in I$ e é construído um *cluster* centralizado em i^* . A construção compreende três passos: (i) adicione o local j que atende i^* a J^* e mais Q locais j com menores valores de c_{i^*j} ; (ii) adicione a I^* todos os clientes alocados nas facilidades de J^* ; (iii) por último, para cada cliente $i \in I^*$, adicione a J^* o local j sem facilidade aberta com menor valor de c_{ij} . Resolve-se, então, o subproblema, que é um SSCFLP restrito ao conjunto de clientes I^* e ao conjunto de locais J^* . Uma nova solução é construída da seguinte maneira: preservam-se as alocações de clientes que não pertencem ao *cluster* e alocam-se os clientes que pertencem ao *cluster* de acordo com as alocações da solução do subproblema. Caso a

solução construída seja melhor do que a solução corrente, esta é atualizada.

Os resultados de Kong (2021) mostraram que o algoritmo é competitivo para instâncias de pequeno, médio e grande porte quando comparado ao algoritmo KS. Porém, em conjuntos muito desafiadores, o MH mostrou-se muito mais eficiente. O ponto forte do algoritmo é permitir múltiplas realocações de clientes de forma localmente ótima. A exploração de soluções infactíveis também é um diferencial, visto que elas podem servir de ponte para alcançar soluções de alta qualidade.

4.3.3 Algoritmo TP-ALG

O TP-ALG é um algoritmo em três fases. A primeira aplica um procedimento de planos de corte para fortalecer a formulação do SSCFLP e, posteriormente, resolve a relaxação linear parcial do problema para obter um limitante inferior forte. Esta relaxação é obtida ao relaxar as condições de integralidade apenas das variáveis x_{ij} . No procedimento de planos de corte proposto, os autores utilizaram três tipos de cortes: *Variable Upper Bounds* (VUBs) que são as restrições (3.6) da formulação apresentada na Seção 3.1, e LCIs e cortes de Fenchel (CFs) obtidos explorando as restrições (3.5) e a restrição (3.7).

Se a solução obtida na primeira fase for inteira, ela é ótima e o algoritmo termina. Caso contrário, passa-se à fase seguinte. Na segunda fase, o TP-ALG utiliza um procedimento *local branching* que primeiro explora vizinhanças definidas pelas variáveis de localização e, depois, vizinhanças definidas pelas variáveis de alocação, para gerar um bom limitante superior. Se a solução da segunda fase for ótima, o algoritmo termina. Senão, passa-se para a terceira fase.

A terceira fase aplica o esquema *cut-and-solve* descrito na Subseção 2.3.2. O problema original é o problema denso inicial que ocupa o topo da árvore de busca. Em cada iteração, o problema esparso é SSCFLP restrito aos locais promissores, isto é, locais com facilidades abertas na solução da relaxação parcial do problema denso da iteração anterior. Já o problema denso é o problema denso da iteração anterior atualizado com a adição de uma restrição que garante que pelo menos uma facilidade deve ser aberta em um local não promissor da iteração corrente.

Os resultados apresentados em Gadegaard et al. (2018) mostram que o TP-ALG é mais eficiente do que o algoritmo de Yang et al. (2012) na maioria das instâncias testadas. Além disso, o procedimento *cut-and-solve* da fase III é comparável ao CPLEX, versão 12.3, em instâncias de pequeno porte e mais eficiente em instâncias de grande porte. Apesar disso, o TP-ALG não conseguiu encontrar a solução ótima em várias instâncias testadas, mesmo com um tempo limite de 50000 segundos. Em suas análises de resultados os autores em Gadegaard et al. (2018) revelam que a

principal dificuldade do TP-ALG é resolver na otimalidade os subproblemas esparsos gerados pelo esquema *cut-and-solve*.

Capítulo 5

Métodos de solução

Este capítulo apresenta os algoritmos propostos nesta tese. A Seção 5.1 apresenta o algoritmo HILS. O algoritmo NCS é apresentado na Seção 5.2. Por fim, o algoritmo B&B-GVNS é apresentado na Seção 5.3.

5.1 Algoritmo HILS

O algoritmo HILS alterna iterativamente entre a resolução de subproblemas com um *solver* de programação matemática e a aplicação de procedimentos ILS. O Algoritmo 7 apresenta o seu pseudocódigo.

Algoritmo 7 HILS

- 1: Identifique locais potenciais promissores
 - 2: Gere uma solução inicial
 - 3: Aplique o procedimento de abertura de facilidades
 - 4: Refine a solução aplicando o procedimento VND
 - 5: **enquanto** o critério de parada não for satisfeito **faça** ▷ Fase I
 - 6: Utilize o *solver* para resolver o SSCFLP Reduzido com tempo limite `tilim1`
 - 7: Refine a solução aplicando o procedimento ILS–I
 - 8: Refine a solução aplicando o procedimento ILS–II
 - 9: **fim enquanto**
 - 10: **enquanto** o critério de parada não for satisfeito **faça** ▷ Fase II
 - 11: Utilize o *solver* para resolver o GAP corrente com limite de tempo `tilim2`
 - 12: Refine a solução aplicando o procedimento ILS–II
 - 13: **fim enquanto**
-

Inicialmente, obtém-se a solução ótima da relaxação linear do SSCFLP e aplica-se um procedimento de identificação de locais promissores (linha 1). O subproblema resultante é um SSCFLP restrito aos locais promissores. Ele é denominado SSCFLP Reduzido. Na linha 2, uma heurística construtiva gera uma solução inicial. Em

seguida, nas linhas 3–4, aplica-se um procedimento de abertura e um procedimento de busca local VND para refinar a solução.

O restante do algoritmo contém duas fases. A primeira fase consiste no laço das linhas 5 a 9, cujo objetivo é encontrar a solução ótima do SSCFLP Reduzido. Na linha 6, um *solver* é utilizado para resolver o SSCFLP Reduzido, com um limite de tempo definido pelo parâmetro `tilim1`. Nas linhas 7 e 8, aplicam-se dois procedimentos ILS, denominados ILS–I e ILS–II. O primeiro perturba a solução corrente, modificando os locais com facilidades abertas. Caso seja aceita uma abertura de facilidade em um local j' não promissor, j' é incluído no conjunto de locais promissores do algoritmo, e o SSCFLP Reduzido é atualizado. No ILS–II, as perturbações alteram apenas a alocação dos clientes, preservando os locais com facilidades abertas. A primeira fase termina quando ocorre uma iteração sem melhoria ou quando a solução ótima do SSCFLP Reduzido é obtida na linha 6 e não é melhorada na linha 7.

A segunda fase corresponde ao laço definido nas linhas 10 a 13. Ela é executada apenas se a solução ótima do SSCFLP Reduzido não tiver sido encontrada na fase anterior. Nessa fase, busca-se a solução ótima do problema de atribuição generalizada corrente (GAP), obtida ao se manter fixas as facilidades abertas na solução da primeira fase. Na linha 11, um *solver* é utilizado para resolver o subproblema resultante, com um limite de tempo definido pelo parâmetro `tilim2`. Na linha 12, aplica-se o procedimento ILS–II. De maneira análoga à primeira fase, a segunda fase termina quando ocorre uma iteração sem melhoria ou quando a solução ótima do GAP é encontrada.

Espera-se que a solução ótima do SSCFLP Reduzido também seja a solução ótima do SSCFLP ou, ao menos, uma boa solução. Entretanto, os procedimentos da primeira fase podem não ser suficientes para encontrá-la. Quando isso ocorre, espera-se que a solução ótima do GAP obtida na primeira fase seja uma boa solução para o SSCFLP. A estratégia de combinar, em cada fase, a resolução de subproblemas por *solvers* e procedimentos heurísticos que exploram vizinhanças baseia-se nas seguintes premissas:

- A busca por soluções por meio da exploração de vizinhanças pode melhorar limitantes superiores em problemas de minimização, mas pode ficar presa em um ótimo local, que pode não ser o ótimo global;
- A resolução de subproblemas por *solvers* pode ser utilizada para a busca não ficar presa em ótimos locais em relação às vizinhanças estabelecidas e, simultaneamente, refinar as soluções.

Nas subseções seguintes, detalham-se os procedimentos do algoritmo proposto. A Subseção 5.1.1 descreve o procedimento de identificação de locais promissores.

A Subseção 5.1.2 descreve os procedimentos de construção da solução inicial e de abertura de facilidades. Por fim, a Subseção 5.1.3 detalha os procedimentos ILS.

5.1.1 Identificação de locais promissores

Considere $\tilde{y}_j$ e $\hat{c}(y_j)$, $j \in J$, como os valores assumidos pelas variáveis de decisão de localização e seus respectivos custos reduzidos na solução ótima da relaxação linear do SSCFLP, e σ a média dos valores positivos de $\hat{c}(y_j)$. O conjunto de locais promissores é definido como o subconjunto $J^0 \subseteq J$ de locais $j \in J$ tais que $\hat{c}(y_j) \leq \delta\sigma$, em que $\delta \geq 0$ é um parâmetro do HILS. Esse subconjunto inclui os locais $j \in J$ tais que $\tilde{y}_j = 1$, pois, para eles, $\hat{c}(y_j) < 0$. Então, o parâmetro δ determina os locais $j \in J$ tais que $\tilde{y}_j < 1$, que também são incluídos em J^0 .

5.1.2 Construção da solução inicial e procedimento de abertura de facilidades

Após a identificação de J^0 , os locais promissores são ordenados do mais promissor ao menos promissor. No HILS, considera-se que locais com maiores valores de $\tilde{y}_j$ são mais promissores. Se dois locais possuem o mesmo valor de $\tilde{y}_j$, o local com menor custo reduzido $\hat{c}(y_j)$ é considerado mais promissor. Se, para dois locais, ambos os valores de $\tilde{y}_j$ e $\hat{c}(y_j)$ são iguais, então o local com o menor valor de f_j/b_j (razão entre o custo de abertura e a capacidade) é considerado mais promissor.

Para gerar a solução inicial, seleciona-se inicialmente um conjunto $J' \subset J^0$ de facilidades a serem abertas. Para isso, adicionam-se locais a J' , na ordem definida no parágrafo anterior, até que a capacidade total das facilidades selecionadas seja maior ou igual à demanda total dos clientes. Definido o conjunto J' , os clientes são ordenados em ordem não crescente de demanda e alocados sequencialmente de acordo com a seguinte regra: para cada cliente i , verifica-se inicialmente se existe alguma facilidade $j \in J'$ tal que $x_{ij}^* = 1$, em que x_{ij}^* representa o valor assumido por x_{ij} na solução da relaxação linear, e cuja capacidade residual seja suficiente para atender o cliente i . Nesse caso, o cliente é alocado à facilidade j . Caso contrário, o cliente é alocado à facilidade de J' com capacidade disponível e que minimize o custo de alocação c_{ij} . Se, para algum cliente, não existir facilidade em J' com capacidade residual suficiente para atendê-lo, todas as alocações realizadas até então são desfeitas, o próximo local da ordenação é adicionado a J' e o procedimento de alocação é reiniciado a partir do primeiro cliente. O processo é repetido até que todos os clientes sejam alocados com sucesso.

A impossibilidade de obtenção de uma solução factível pelo procedimento de ge-

ração descrito acima somente pode ocorrer em instâncias altamente degeneradas, nas quais o procedimento não consegue alocar todos os clientes mesmo após a abertura de facilidades em todos os locais disponíveis. Se isso ocorrer, o HILS falha e não retorna uma solução heurística para o problema.

Após gerar a solução inicial, aplica-se uma busca local, explorando soluções que possuem mais facilidades abertas do que a solução inicial. O procedimento consiste em abrir uma facilidade em um determinado local, desalocar todos os clientes e aplicar a mesma regra de alocação de clientes descrita anteriormente. Para a exploração da vizinhança nesse procedimento, adota-se a estratégia *Best Improvement*. Caso facilidades sejam abertas em locais não promissores, esses locais passam a integrar o conjunto de locais promissores e o SSCFLP Reduzido é atualizado.

5.1.3 Procedimentos ILS

Nesta subseção, são detalhados os procedimentos ILS-I e ILS-II do algoritmo HILS. Ambos aplicam procedimentos VND como componente de busca local. Os procedimentos VND e ILS foram apresentados na fundamentação teórica desta tese (Capítulo 2, nas seções 2.1.3 e 2.1.4, respectivamente). Por isso, a teoria acerca desses procedimentos será omitida nesta Seção.

Primeiro, apresentam-se os procedimentos VND. Para isso, considere os seguintes movimentos:

- (i) Realoca um cliente: consiste em realocar um cliente de uma facilidade que possui mais de um cliente alocado para outra facilidade aberta;
- (ii) Troca de dois clientes: consiste em trocar a alocação de dois clientes;
- (iii) Fecha uma facilidade: consiste em fechar uma facilidade, realocando todos os clientes atendidos por ela para outra facilidade.

O procedimento VND do ILS-I utiliza as estruturas de vizinhança definidas pelos três movimentos acima, enquanto o procedimento VND do ILS-II utiliza apenas as definidas pelos dois primeiros movimentos. Em ambos os VND, movimentos que resultam na violação de capacidades não são considerados. O critério *Pipe Neighborhood Change Step* é empregado para alterar as estruturas de vizinhança, e a estratégia *Best Improvement* é adotada para explorar as vizinhanças. Além disso, as estruturas foram ordenadas segundo o critério de que movimentos que modificam menos as soluções devem ser realizados antes dos que modificam mais. Com esse critério, a ordem adotada é a mesma em que os movimentos foram apresentados acima.

Para descrever os procedimentos ILS, considere as perturbações:

Algoritmo 8 Algoritmo ILS para os procedimentos ILS-I e ILS-II

Entrada: solução corrente, procedimento VND, movimento de perturbação , ILSmax

```

1:  $s \leftarrow$  solução corrente
2:  $\text{Iter} \leftarrow 0$ 
3:  $l \leftarrow 1$ 
4: enquanto  $\text{Iter} < \text{ILSmax}$  faça
5:    $\text{Iter} \leftarrow \text{Iter} + 1$ 
6:    $s' \leftarrow$  solução obtida modificando  $s$  pelo movimento de perturbação no nível  $l$ 
7:    $s'' \leftarrow$  solução obtida refinando  $s'$  através do procedimento VND
8:   se  $s''$  for melhor do que  $s$  então
9:      $s \leftarrow s''$ 
10:     $\text{Iter} \leftarrow 0$ 
11:     $l \leftarrow 1$ 
12:   senão
13:      $l \leftarrow l + 1$  ▷ apenas para ILS-II
14:   fim se
15: fim enquanto
16: Return Solução refinada  $s$ 

```

- (i) Perturbação 1 (Abre facilidade): utilizada no procedimento ILS-I, abre uma facilidade em um local escolhido aleatoriamente e realoca para ela clientes com menores custos de alocação c_{ij} cujas demandas podem ser satisfeitas até que não seja mais possível realocar clientes para ela;
- (ii) Perturbação 2 (Troca de alocação de clientes): utilizada no procedimento ILS-II, consiste em realizar uma troca aleatória na alocação de dois clientes.

A Perturbação 2 é aplicada no procedimento ILS-II considerando níveis $l \in \mathbb{Z}$, $l \geq 1$. Cada nível l define um número de trocas aleatórias igual a $\lfloor \gamma \times l \times n \rfloor$, em que γ é um parâmetro do ILS-II e n é o número total de clientes. A consideração de níveis ocorre da seguinte maneira: se não há melhora da solução corrente, aumenta-se o nível de perturbação; caso contrário, retorna-se ao primeiro nível ($l = 1$). Essa estratégia possibilita que o número de trocas aleatórias cresça a cada iteração sem melhora e que esse crescimento ocorra mais rapidamente em problemas de grande porte. Por outro lado, como cada Perturbação 1 modifica a solução consideravelmente, ela é aplicada no algoritmo ILS-I sem considerar o aumento de nível de perturbação. Observa-se que o ILS-I pode alterar os locais de facilidades abertas e, ao mesmo tempo, a alocação de clientes, enquanto ILS-II pode alterar apenas a alocação de clientes.

O Algoritmo 8 descreve o pseudocódigo dos procedimentos ILS-I e ILS-II. O processo iterativo ocorre nas linhas 4 a 15. Na linha 6, o movimento de perturbação no nível l é aplicado na solução corrente s , gerando a solução intermediária s' . Em

seguida, conforme linha 7, o procedimento VND é aplicado à solução s' , obtendo-se a solução s'' . Se s'' for melhor do que a solução corrente s , a solução corrente é atualizada (linhas 8- 11). Caso contrário, na linha 13, aumenta-se o nível da perturbação (apenas no procedimento ILS-II). A parada desses procedimentos (linha 4) ocorre quando o número de iterações sem melhora atinge os valores máximos permitidos, definidos pelos parâmetros ILSmaxI para o ILS-I e ILSmaxII para o ILS-II.

5.2 Algoritmo NCS

O algoritmo NCS é composto por duas fases. A primeira fase aplica um procedimento de planos de corte para fortalecer a formulação e resolve uma relaxação linear parcial para obter um limitante inferior forte para o SSCFLP. A segunda fase aplica o *framework cut-and-solve* para explorar o espaço de soluções do problema. Um procedimento heurístico é incorporado à segunda fase para torná-la mais eficiente.

O NCS difere dos algoritmos apresentados por Yang et al. (2012) e Gadegaard et al. (2018) principalmente nos procedimentos de geração de planos de corte propostos e na heurística empregada. A heurística utilizada no NCS baseia-se em estratégias de *clustering* que agrupam locais e clientes supostamente próximos para a construção de vizinhanças. Essas abordagens mostraram-se muito eficientes na resolução de problemas de localização, particularmente quando aplicadas a problemas de grande porte (Kong, 2021; Gnägi & Baumann, 2021). Deve-se ressaltar que Yang et al. (2012) propuseram um procedimento Busca Tabu, enquanto Gadegaard et al. (2018) aplicaram uma matheurística *Local Branching*.

No que se refere aos procedimentos de geração de planos de corte, Yang et al. (2012) e Gadegaard et al. (2018) utilizaram VUBs (definidos na Seção 4.3.3), LCIs obtidos por *lifting* sequencial (conforme Seção 2.4.5), e cortes de Fenchel (CFs). Para obter os cortes de cobertura necessários à geração de LCIs, Yang et al. (2012) propuseram um procedimento de separação exata, enquanto Gadegaard et al. (2018) adotaram uma heurística construtiva. A abordagem proposta nesta tese considera os tipos de desigualdades utilizados por esses autores e incorpora dois tipos adicionais de cortes: (i) LPIs e (ii) outros tipos de LCIs, ambos gerados por meio de funções superaditivas. Além disso, é introduzida uma nova heurística gulosa para a geração de cortes de cobertura. Diferentemente dos LCIs obtidos por *lifting* sequencial, dos CFs e dos cortes de cobertura obtidos por separação exata, os procedimentos adicionais propostos nesta tese não requerem a resolução de problemas da mochila para obter desigualdades válidas. Essa proposta pode aumentar a eficiência do procedimento de planos de corte da primeira fase. Dado o grande número de tipos de cortes considerados, propõe-se uma avaliação sistemática da eficácia de diferentes famílias

de planos de corte. Os resultados são utilizados para identificar um conjunto eficaz de cortes a serem empregados no NCS.

O restante desta seção está organizado da seguinte forma. A Subseção 5.2.1 apresenta os procedimentos utilizados para obter uma relaxação parcial forte para o SSCFLP e a Subseção 5.2.2 descreve o algoritmo exato proposto.

5.2.1 Uma relaxação parcial forte para o SSCFLP

Uma estratégia importante para resolver o SSCFLP consiste em obter uma relaxação parcial forte, isto é, um modelo derivado do SSCFLP no qual as restrições de integralidade de um subconjunto das variáveis de decisão são relaxadas e que fornece bons limitantes inferiores. Para isso, o NCS combina um procedimento de planos de corte (PC) e a estratégia de integralidade parcial (*partial integrality strategy* – PIS) (Yang et al., 2012). A abordagem PIS mantém a integralidade das variáveis de localização (y_j), relaxando apenas as restrições de integralidade das variáveis de alocação (x_{ij}). Essa estratégia combinada é denominada procedimento PC+PIS.

Os planos de corte empregados na primeira fase podem ser divididos em três grupos. Seja, então, $\mathbf{x}_j = (x_{1j}, \dots, x_{nj})$ e $\mathbf{y} = (y_1, \dots, y_m)$. O primeiro grupo consiste em desigualdades da forma $x_{ij} \leq y_j$ (VUBs). O segundo grupo é composto por planos de corte obtidos explorando desigualdades válidas para os m politopos da mochila:

$$\mathcal{X}_j^y = \text{casca convexa de } \left\{ \mathbf{x}_j \in \{0,1\}^n : \sum_{i \in I} d_i x_{ij} \leq b_j y_j \right\}, \quad j \in J, \quad (5.1)$$

que são definidos pelas restrições de capacidade do SSCFLP. O terceiro grupo é formado por planos de corte obtidos explorando desigualdades válidas para o politopo da mochila:

$$\mathcal{Y} = \text{casca convexa de } \left\{ \mathbf{y} \in \{0,1\}^m : \sum_{j \in J} b_j y_j \geq D \right\}. \quad (5.2)$$

O Algoritmo 9 descreve o procedimento PC+PIS. Nele, $\mathcal{F}$ é a formulação do SSCFLP descrita na Seção 3.1, sem as restrições redundantes (3.6); $\text{PL}(\mathcal{F})$ denota a relaxação linear de $\mathcal{F}$, e $\mathcal{F}$ -parcialmente-relaxado denota o modelo da relaxação parcial forte retornado pelo procedimento.

Inicialmente, aplica-se o procedimento PC. Nesse procedimento, iterativamente, planos de corte violados pela solução ótima de $\text{PL}(\mathcal{F})$ são adicionados ao modelo $\mathcal{F}$ até que a diferença relativa entre dois limitantes inferiores consecutivos seja inferior ao parâmetro **TolerCuttingPlane**. Ressalta-se que é necessário selecionar os procedimentos de geração de planos de corte para explorar os politopos da mochila. Além

Algoritmo 9 Procedimento PC+PIS

```

1: Entrada:  $\mathcal{F}$ , TolerPlanosCortantes
2: Saída:  $\mathcal{F}$ -parcialmente-relaxado
3: Selecione procedimentos de geração de planos cortantes para explorar os politopos da mochila  $\mathcal{X}_j^y$  e  $\mathcal{Y}$ 
4: Resolva o  $PL(\mathcal{F})$  e considere  $s^*$  a sua solução ótima
5:  $LI \leftarrow$  valor ótimo de  $PL(\mathcal{F})$ 
6: ParePC  $\leftarrow$  Falso
7: enquanto ParePC = Falso faça
8:    $LI^* \leftarrow LI$ 
9:   Adicione ao modelo  $\mathcal{F}$  todos os planos cortantes violados por  $s^*$  obtidos explorando VUBs
10:  para  $j \in J$  faça
11:    Adicione ao modelo  $\mathcal{F}$  um plano cortante violado por  $s^*$  obtido explorando  $\mathcal{X}_j^y$ 
12:  fim para
13:  Adicione ao modelo  $\mathcal{F}$  um plano cortante violado obtido explorando o politopo  $\mathcal{Y}$ 
14:  Resolva  $PL(\mathcal{F})$  e considere  $s^*$  a solução de  $PL(\mathcal{F})$ 
15:   $LI \leftarrow$  valor ótimo de  $PL(\mathcal{F})$ 
16:  se  $(LI - LI^*)/LI^* \geq \text{TolerPlanosCortantes}$  então
17:    ParePC  $\leftarrow$  Verdadeiro
18:  fim se
19: fim enquanto
20:  $\mathcal{F}$ -parcialmente-relaxado  $\leftarrow$   $\mathcal{F}$  fortalecido pela adição de cortes
21:  $\mathcal{F}$ -parcialmente-relaxado  $\leftarrow$  o modelo obtido de  $\mathcal{F}$ -parcialmente-relaxado relaxando a integralidade das variáveis  $x_{ij}$ ,  $\forall i \in I, j \in J$ 

```

disso, múltiplos cortes violados podem ser adicionados na mesma iteração. Após a aplicação do PC, emprega-se o PIS no modelo de $\mathcal{F}$, fortalecido pela adição de cortes (linha 21).

Algumas considerações sobre a geração de cortes que exploram $\mathcal{X}_j^y$ e $\mathcal{Y}$ são importantes. Dado $j \in J$, para obter desigualdades válidas para $\mathcal{X}_j^y$ pode-se acrescentar a variável y_j em desigualdades válidas para o politopo:

$$\mathcal{X}_j = \text{casca convexa de } \left\{ \mathbf{x}_j \in \{0,1\}^n : \sum_{i \in I} d_i x_{ij} \leq b_j \right\}.$$

Mais precisamente, considere $\sum_{i \in I} \sigma_{ij} x_{ij} \leq \kappa_j$, com $\sigma_{ij} \geq 0$, para $i \in I$, e $\kappa_j \geq 0$, a sentença geral que representa um plano de corte para $\mathcal{X}_j$. Então,

$$\text{se } \sum_{i \in I} \sigma_{ij} x_{ij} \leq \kappa_j \text{ é válida para } \mathcal{X}_j, \text{ então } \sum_{i \in I} \sigma_{ij} x_{ij} \leq \kappa_j y_j \text{ é válida para } \mathcal{X}_j^y. \quad (5.3)$$

Outra observação é que se pode substituir a restrição que define $\mathcal{Y}$ por outra equivalente (Yang et al., 2012), da seguinte forma. Considerando as variáveis auxiliares $\tilde{y}_j = 1 - y_j, j \in J$, ao substituir y_j por $1 - \tilde{y}_j$ na restrição $\sum_{j \in J} b_j y_j \geq D$ obtém-se a restrição da mochila

$$\sum_{j \in J} b_j \tilde{y}_j \leq \sum_{j \in J} b_j - D, \quad (5.4)$$

que tem a mesma estrutura da restrição que define $\mathcal{X}_j$ e é utilizada para gerar planos cortantes para $\mathcal{Y}$.

As subseções 5.2.1.1, 5.2.1.2 e 5.2.1.3, a seguir, descrevem, respectivamente, como LCIs, LPIs e CFs são construídos para obter planos cortantes violados para $\mathcal{X}_j^y$. A construção desses tipos de planos cortantes para $\mathcal{Y}$ segue os mesmos passos. Nestas subseções, denota-se por s^* uma solução fracionária do SSCFLP e por x_{ij}^* e y_j^* os valores assumidos pelas variáveis x_{ij} e y_j em s^* .

5.2.1.1 *Lifted Cover Inequalities*

Se C é uma cobertura para j , a desigualdade

$$\sum_{i \in C} x_{ij} \leq |C| - 1$$

é válida para $\mathcal{X}_j$. Aplicando a implicação (5.3), obtém-se a desigualdade mais forte:

$$\sum_{i \in C} x_{ij} \leq (|C| - 1)y_j, \quad (5.5)$$

que é válida para $\mathcal{X}_j^y$. As desigualdades (5.5) são referidas nesta seção como cortes de cobertura (*cover inequalities* - CIs).

Considere que um LCI é uma desigualdade válida obtida fortalecendo um CI pela aplicação de um procedimento de *lifting*. Esta tese considera quatro tipos de procedimentos para gerar CIs e dois tipos de procedimentos para construir LCIs a partir de CIs. No procedimento PC+PIS, se um corte de cobertura violado é obtido, ele é selecionado para gerar um LCI violado. Por outro lado, mesmo que nenhum corte de cobertura violado seja obtido em um dos procedimentos, o mais próximo de ser violado obtido nos procedimentos é selecionado para gerar um LCI que pode ser violado. Observa-se que um corte de cobertura violado satisfaz:

$$\sum_{i \in C} x_{ij}^* > (|C| - 1)y_j^*. \quad (5.6)$$

A condição (5.6) é equivalente a $\sum_{i \in C} (y_j^* - x_{ij}^*) < y_j^*$. Por isso, quanto menor o valor de $\sum_{i \in C} (y_j^* - x_{ij}^*)$, mais próximo de ser violado é o corte de cobertura.

Geração de cortes de cobertura

Dois dos procedimentos propostos para gerar CI são heurísticas gulosas extraídas da literatura. Ambas ordenam os clientes seguindo um certo critério e incluem os clientes em C até que a condição $\sum_{i \in C} d_i > b_j$ seja satisfeita. A heurística gulosa

Algoritmo 10 Separação exata de CIs (Yang et al., 2012)**Entrada:** $\mathcal{X}_j, s^*$ **Saída:** CI violado ou $\emptyset$

```

1:  $f_j(k, r) \leftarrow \infty$  para  $k = 1, \dots, n$  e  $r = 0, \dots, b_j$ 
2:  $f_j(0, 0) \leftarrow 0$ 
3:  $g_j(k) \leftarrow \infty$  para  $k = 1, \dots, n$ 
4: para  $k = 1, \dots, n$  faça
5:   para  $r = 0, \dots, b_j$  faça
6:     se  $f_j(k-1, r) < f_j(k, r)$  então
7:        $f_j(k, r) \leftarrow f_j(k-1, r)$ 
8:     fim se
9:   fim para
10:  para  $r = 0, \dots, b_j - d_k$  faça
11:    se  $f_j(k-1, r) + (y_j^* - x_{kj}) < f_j(k, r + d_k)$  então
12:       $f_j(k, r + d_k) \leftarrow f_j(k-1, r) + (y_j^* - x_{kj})$ 
13:    fim se
14:  fim para
15:  para  $r = b_j - d_k + 1, \dots, b_j$  faça
16:    se  $f_j(k-1, r) + (y_j^* - x_{kj}) < g_j(k)$  então
17:       $g_j(k) \leftarrow f_j(k-1, r) + (y_j^* - x_{kj})$ 
18:    fim se
19:  fim para
20:  se  $g_j(k) < y_j^*$  então
21:    retorne um CI violado
22:  fim se
23: fim para
24: retorne  $\emptyset$ 

```

proposta por Gu et al. (2000) ordena os clientes em ordem não crescente de x_{ij}^* , enquanto a heurística de Crowder et al. (1983) ordena os clientes em ordem não decrescente de $(y_j^* - x_{ij}^*)/d_i$. Para quebrar empates na ordenação de clientes, ambas as heurísticas utilizam a ordem não crescente de d_i . Esses dois procedimentos são denotados por CIGulosaA e CIGulosaB, respectivamente.

O terceiro procedimento é uma heurística gulosa introduzida nesta tese. Ela é baseada na observação de que a maior parte de cortes de cobertura violados é gerada por coberturas C compostas por clientes em $K_j^1 = \{i \in I : x_{ij}^* = 1\}$ e por alguns clientes no conjunto $K_j = \{i \in I : 0 < x_{ij}^* < 1\}$. Então, a heurística começa considerando $C = K_j^1$ e ordenando os clientes de K_j em ordem não-crescente de valores de x_{ij}^* . Em seguida, testa todas as possibilidades de inserir um, dois ou três clientes de K_j em C e verifica se C gera um corte de cobertura violado. Se sim, o procedimento termina. A inserção de quatro ou mais clientes não é considerada, pois o aumento no custo computacional, devido ao risco de falha do procedimento, geralmente não compensa o benefício de considerá-los. Essa heurística é referida como CIGulosaNova.

O último procedimento, denominado CIExato, consiste na separação exata de cortes do tipo (5.5) para $\mathcal{X}_j^y$, proposta por Yang et al. (2012). Ele é uma adaptação

do procedimento de separação exata de CIs para politopos da mochila proposto por [Kaparis & Letchford \(2010\)](#).

Considere

$$g_j(k) = \min \left\{ \sum_{i=1}^k (y_j^* - x_{ij}^*) v_i : \sum_{i=1}^k d_i v_i \geq b_j + 1, v_i \in \{0,1\}^k \right\}. \quad (5.7)$$

Existe um CI violado se, e somente se, $g_j(k) < y_j^*$ para algum valor de k . Se $g_j(k) < y_j^*$, seja $I^{\leq k} = \{i \in I; i \leq k\}$. Então, a cobertura $C = \{i \in I^{\leq k}; v_i^* = 1\}$, em que v_i^* é o valor da variável v_i na solução ótima do problema que define $g_j(k)$ na sentença (5.7) para $i \in I^{\leq k}$, gera um CI violado.

Considere, também, para $k = 1, 2, \dots, n$ e $r = 0, 1, \dots, b_j$:

$$f_j(k, r) = \min \left\{ \sum_{i=1}^k (y_j^* - x_{ij}^*) v_i : \sum_{i=1}^k d_i v_i = r, v_i \in \{0,1\}^k \right\}.$$

Dado $\mathcal{X}_j$ e s^* , o Algoritmo 10 resolve o problema de separação exata de CIs, isto é, retorna um CI violado ou $\emptyset$, caso nenhum CI violado exista.

Lifting CIs através de uma função superaditiva

Um dos procedimentos de *lifting* considerados no NCS constrói uma função g superaditiva em $\mathbb{Z}_+$, isto é, $g(d_1) + g(d_2) \leq g(d_1 + d_2)$ para todo $d_1, d_2 \in \mathbb{Z}_+$. Essa função permite modificar os coeficientes do lado esquerdo da expressão (5.5), incluindo não apenas os coeficientes iguais a 1 de x_{ij} para $i \in C$, mas também os coeficientes iguais a 0 de x_{ij} para $i \notin C$. Para isto, utiliza-se a função superaditiva de [Letchford & Souli \(2019\)](#), apresentada a seguir.

Seja C uma cobertura para a facilidade j , e seja $\bar{d}$ o único valor positivo tal que $\sum_{i \in C} \min\{d_i, \bar{d}\} = b_j$. Defina $d_i^- = \min\{d_i, \bar{d}\}$, $C^- = \{i \in C : d_i \leq \bar{d}\}$ e $C^+ = \{i \in C : d_i > \bar{d}\}$. Seja $S^-(r)$ a soma dos r maiores valores de d_i^- para $r \geq 1$, com $S^-(0) = 0$. Também, considere as funções:

$$f(z) = \begin{cases} 0, & \text{se } z = 0 \\ h, & \text{se } S^-(h) < z \leq S^-(h+1), \text{ para algum } h \in \{0, \dots, |C| - 1\} \end{cases}$$

$$g(z) = \begin{cases} f(z) + 1/2, & \text{se } z = h\bar{d}, \text{ para algum } h \in \{1, \dots, |C^+| - 1\} \\ f(z), & \text{caso contrário.} \end{cases}$$

Então,

$$\sum_{i \in C^-} x_{ij} + \sum_{i \in I \setminus C^-} g(d_i) x_{ij} \leq (|C| - 1) y_j \quad (5.8)$$

é uma desigualdade válida para $\mathcal{X}_j^y$ pelo menos tão forte quanto a desigualdade (5.5). Denominamos como LCIFunçãoSuperaditiva ao procedimento para construir a LCI (5.8) a partir de um CI gerada por uma cobertura C .

***Lifting* CIs através de *lifting* sequencial**

Nesta tese, utiliza-se o *lifting* sequencial para derivar, a partir de CIs, LCIs da forma:

$$\sum_{i \in C_1} x_{ij} + \sum_{i \in C_2} \beta_{ij} x_{ij} + \sum_{i \in I \setminus C} \alpha_{ij} x_{ij} \leq (|C_1| - 1 + \sum_{i \in C_2} \beta_{ij}) y_j, \quad (5.9)$$

em que C_1 e C_2 formam uma partição de C , com $C_1 \neq \emptyset$. Os coeficientes β_{ij} e α_{ij} são calculados por meio da resolução de problemas da mochila, conforme detalhado na Subseção 2.4.5.

Para construir LCIs do tipo (5.9) utilizando *lifting* sequencial, é necessário definir um dos conjuntos C_1 e C_2 para iniciar o processo, bem como uma ordem para o cálculo dos coeficientes α_{ij} e β_{ij} . Nesta tese, essas definições são feitas seguindo Hoffman & Padberg (1991): (i) considera-se $C_2 = \{i \in C : x_{ij}^* = 1\}$ e, inicialmente, determinam-se os coeficientes α_{ij} para o conjunto $T_1 = \{i \in I \setminus C : 0 < x_{ij}^* < 1\}$; (ii) calculam-se β_{ij} para o conjunto C_2 ; (iii) determinam-se α_{ij} para $T_0 = \{i \in I \setminus C : x_{ij}^* = 0\}$. Nesse procedimento, os elementos de C_2 e T_0 são ordenados em ordem não crescente de d_i , enquanto os de T_1 são ordenados em ordem não crescente de x_{ij}^* .

Denomina-se o procedimento para construção de LCIs do tipo (5.9), utilizando o *lifting* sequencial descrito nesta subseção, como LCILiftingSequencial.

5.2.1.2 *Lifted pack inequalities*

Um conjunto de clientes $P \subset I$ é um pacote para uma facilidade $j \in J$ se P não é uma cobertura para a facilidade, isto é:

$$\sum_{i \in P} d_i \leq b_j.$$

Seja $\mu_j = b_j - \sum_{i \in P} d_i$ a capacidade residual da facilidade j . A desigualdade:

$$\sum_{i \in I \setminus P} (d_i - \mu_j)^+ x_{ij} \leq 0, \quad (5.10)$$

em que $(d_i - \mu_j)^+ = \max(d_i - \mu_j, 0)$ é válida para $\mathcal{X}_j \cap \{\mathbf{x}_j \in \{0,1\}^n; x_{ij} = 1 \ \forall i \in P\}$ e é chamada de desigualdade de empacotamento (*pack inequality* - PI) (Atamturk, 2005). Para construir PIs, clientes são ordenados em ordem não crescente de valores de x_{ij}^* e adicionados a P enquanto a condição $\sum_{i \in P} d_i \leq b_j$ seja satisfeita.

Como no caso de CIs, PIs são fortalecidos para obter LPs. Para isso, propõe-se utilizar a função superaditiva descrita em Atamturk (2005). Seja $I' = \{i \in I \setminus P : d_i > \mu_j\}$ e $S(r)$ a soma das r maiores demandas de clientes em I' , com r um número inteiro maior ou igual a 1 e $S(0) = 0$. Considere:

$$g(z) = \begin{cases} h\mu_j + z, & \text{se } -S(h+1) + \mu_j \leq z \leq -S(h), \text{ para algum } h \in \{0, 1, \dots, |I'| - 1\} \\ h\mu_j - S(h), & \text{se } -S(h) \leq z \leq -S(h) + \mu_j, \text{ para algum } h \in \{0, 1, \dots, |I'| - 1\} \\ |I'|\mu_j - S(|I'|), & \text{se } z \leq -S(|I'|) + \mu_j. \end{cases}$$

Então, a desigualdade:

$$\sum_{i \in P} g(-d_i)(1 - x_{ij}) + \sum_{i \in I \setminus P} (d_i - \mu_j)^+ x_{ij} \leq 0 \quad (5.11)$$

é válida para $\mathcal{X}_j$. Aplicando a implicação (5.3), obtém-se o LPI:

$$\sum_{i \in P} -g(-d_i)x_{ij} + \sum_{i \in I \setminus P} (d_i - \mu_j)^+ x_{ij} \leq \left(\sum_{i \in P} -g(-d_i) \right) y_j. \quad (5.12)$$

O procedimento para construir LPs descrito nesta subseção é denominado LPIFuncaoSuperaditiva.

5.2.1.3 Cortes de Fenchel

Os cortes de Fenchel (CFs) são desigualdades obtidas ao se resolver o problema de separação exata de uma solução fracionária da casca convexa do conjunto de soluções viáveis de um problema de otimização inteira (Boyd, 1993). Nesta tese, para gerar CFs para $\mathcal{X}_j^y$, adota-se o procedimento proposto por Gadegaard et al. (2018), que reduz o problema de separação e utiliza um esquema de geração de colunas para resolvê-lo.

Dado $j \in J$, o problema de separação pode ser restringido às variáveis com valores fracionários, conforme descrito a seguir. Seja $K_j = \{i \in I : 0 < x_{ij}^* < 1\}$; $\mathbf{x}_{K_j} = (x_{ij})_{i \in K_j}$ e $\mathbf{x}_{K_j}^* = (x_{ij}^*)_{i \in K_j}$ os vetores coluna das variáveis x_{ij} e dos valores x_{ij}^* ; $\bar{b}_j = b_j - \sum_{i \in K_j^1} d_i$; e $\mathcal{X}_{K_j}$ a casca convexa de $\left\{ \mathbf{x}_{K_j} \in \{0,1\}^{|K_j|} : \sum_{i \in K_j} d_i x_{ij} \leq \bar{b}_j \right\}$. Introduzindo o vetor coluna de variáveis de decisão $\boldsymbol{\lambda}_{K_j} = (\lambda_{ij})_{i \in K_j}$ e utilizando o sobrescrito T para denotar a transposta de um vetor, o problema de separar $\mathbf{x}_{K_j}^*$ de

$\mathcal{X}_{K_j}$ é dado por:

$$v_{K_j} = \max \left\{ \boldsymbol{\lambda}_{K_j}^T \cdot \mathbf{x}_{K_j}^* : \boldsymbol{\lambda}_{K_j}^T \cdot \mathbf{x}_{K_j} \leq 1, \forall \mathbf{x}_{K_j} \in E(\mathcal{X}_{K_j}), \boldsymbol{\lambda}_{K_j} \in [0,1]^{|K_j|} \right\},$$

em que $E(\mathcal{X}_{K_j})$ é o conjunto de pontos extremos de $\mathcal{X}_{K_j}$. Note que $\mathbf{x}_{K_j} \in E(\mathcal{X}_{K_j})$ não é uma variável de decisão nesse problema.

Se $v_{K_j} > 1$, os valores de λ_{ij} , $i \in K_j$, na solução ótima fornecem o corte de Fenchel

$$\sum_{i \in K_j} \lambda_{ij} x_{ij} \leq 1, \quad (5.13)$$

que separa $\mathbf{x}_{K_j}^*$ de $\mathcal{X}_{K_j}$. Esse corte é, então, estendido para o corte de Fenchel

$$\sum_{i \in K_j^1} x_{ij} + \sum_{i \in K_j} \lambda_{ij} x_{ij} \leq 1 + |K_j^1|, \quad (5.14)$$

que separa s^* de $\mathcal{X}_j$. Por outro lado, se $v_{K_j} \leq 1$, não existe plano de corte violado. Em vez de resolver o problema que define v_{K_j} diretamente, resolve-se sua formulação dual por meio de um procedimento de geração de colunas. Para cada $\mathbf{x}_{K_j} \in E(\mathcal{X}_{K_j})$, denote por $\mu_{\mathbf{x}_{K_j}}$ a variável dual associada à restrição $\boldsymbol{\lambda}_{K_j}^T \cdot \mathbf{x}_{K_j} \leq 1$. Então, o problema dual de v_{K_j} é dado por:

$$\min \left\{ \sum_{\mathbf{x}_{K_j} \in E(\mathcal{X}_{K_j})} \mu_{\mathbf{x}_{K_j}} : \sum_{\mathbf{x}_{K_j} \in E(\mathcal{X}_{K_j})} \mu_{\mathbf{x}_{K_j}} \mathbf{x}_{K_j} \geq \mathbf{x}_{K_j}^*, \mu_{\mathbf{x}_{K_j}} \geq 0, \forall \mathbf{x}_{K_j} \in E(\mathcal{X}_{K_j}) \right\}.$$

O procedimento de geração de colunas inicializa o problema mestre restrito com as colunas da matriz identidade $|K_j| \times |K_j|$. Em seguida, de forma iterativa, os valores duais da solução do problema mestre restrito, denotados por λ'_{ij} , $i \in K_j$, fornecem a desigualdade candidata $\sum_{i \in K_j} \lambda'_{ij} x_{ij} \leq 1$. Se essa desigualdade não for violada por $\mathbf{x}_{K_j}^*$, então não há corte de Fenchel. Caso contrário, ela é selecionada para verificação de validade em $\mathcal{X}_{K_j}$, que consiste em verificar se:

$$\max \left\{ \sum_{i \in K_j} \lambda'_{ij} x_{ij} : \sum_{i \in K_j} d_i x_{ij} \leq \bar{b}_j, x_{ij} \in \{0,1\}, \forall i \in K_j \right\} \leq 1. \quad (5.15)$$

Se a desigualdade candidata for violada, mas não válida, a solução do problema de maximização em (5.15) fornece uma nova coluna para o problema mestre restrito. Se for violada por $\mathbf{x}_{K_j}^*$ e válida para $\mathcal{X}_{K_j}$, obtém-se um corte de Fenchel. Para evitar tempo de execução excessivo, especialmente em casos de convergência lenta, introduz-se o parâmetro **MaxIterFc**, que limita o número máximo de iterações.

Uma limitação importante desse procedimento é que os coeficientes da desigualdade candidata $\sum_{i \in K_j} \lambda'_{ij} x_{ij} \leq 1$ podem ser fracionários. Isto pode causar erros de arredondamento na verificação de validade na expressão (5.15). Para evitar que desigualdades não válidas sejam incluídas na formulação do SSCFLP, são consideradas três estratégias. Suponha que a desigualdade candidata seja violada, mas possua coeficientes fracionários. A primeira estratégia, utilizada por [Kaparis & Letchford \(2010\)](#), divide a desigualdade pelo menor coeficiente fracionário. Se os coeficientes da nova desigualdade forem inteiros, ela é selecionada para verificação de validade. Caso contrário, aplica-se a segunda estratégia, proposta por [Gadegaard et al. \(2018\)](#). Essa estratégia multiplica a desigualdade candidata por um número grande M e arredonda os coeficientes resultantes para o inteiro mais próximo. Se a nova desigualdade não for violada, adiciona-se uma nova coluna ao problema mestre restrito e inicia-se uma nova iteração. Se for violada, mas não válida, utiliza-se a terceira estratégia, proposta por [Yang et al. \(2012\)](#), que também multiplica a desigualdade candidata por M , mas substitui cada coeficiente resultante pela sua parte inteira. Se a desigualdade obtida não for violada ou não for válida, adiciona-se uma nova coluna ao problema mestre restrito e inicia-se uma nova iteração.

Exemplo 5.2.1 *O exemplo a seguir ilustra a aplicação das três estratégias apresentadas no parágrafo anterior. Considere o politopo determinado pela restrição da mochila $27x_1 + 5x_2 + 31x_3 + 31x_4 + 12x_5 + 5x_6 + 33x_7 + 7x_8 + 15x_9 \leq 82$, a desigualdade candidata $0,3x_1 + 0,6x_2 + 0,36x_3 + 0,36x_4 + 0,16x_5 + 0,6x_6 + 0,4x_7 + 0,1x_8 + 0,2x_9 \leq 1$ e $M = 10^6$. A primeira estratégia produz o corte $3x_1 + 6,6x_2 + 3,6x_3 + 3,6x_4 + 1,6x_5 + 6,6x_6 + 4x_7 + 1x_8 + 2x_9 \leq 10$, que não possui coeficientes inteiros. A segunda estratégia fornece o corte $300000x_1 + 666667x_2 + 366667x_3 + 366667x_4 + 166667x_5 + 666667x_6 + 400000x_7 + 100000x_8 + 200000x_9 \leq 1000000$. Essa desigualdade não é válida, pois o valor ótimo do problema de verificação de validade é 1000001. A aplicação da terceira estratégia produz o corte $300000x_1 + 666666x_2 + 366666x_3 + 366666x_4 + 166666x_5 + 666666x_6 + 400000x_7 + 100000x_8 + 200000x_9 \leq 1000000$, válido para o politopo, pois o valor ótimo do problema de verificação de validade é 1000000.*

Se o procedimento descrito acima gera um plano de corte violado, aplica-se *lifting* sequencial para fortalecê-lo. Primeiro, aplica-se *up-lifting* nas variáveis x_{ij} com $i \in K_j^1$, e, posteriormente, *down-lifting* nas variáveis x_{ij} com $i \in K_j$. Por fim, aplica-se a implicação (5.3) para obter um plano de corte violado ainda mais forte para $\mathcal{X}_j^y$. Denota-se por *FCGeracaoColunas* o procedimento de seleção de cortes de Fenchel descrito nesta subseção.

5.2.2 Descrição do NCS

O NCS inicia aplicando o procedimento PC+PIS descrito na Subseção 5.2.1. A relaxação parcial resultante é resolvida para obter um limitante inferior forte. Se a solução for inteira, ela é ótima e o algoritmo é encerrado. Caso contrário, o algoritmo prossegue para o seu laço principal (segunda fase).

No laço principal, conjuntos de locais promissores e não promissores são definidos iterativamente para gerar dois tipos de subproblemas: (i) o problema esparso (PE) e o (ii) problema denso (PD). O modelo PD inicial corresponde ao SSCFLP fortalecido pelo procedimento de planos de corte. Em cada iteração, o conjunto de locais promissores é composto pelas facilidades abertas na solução da relaxação parcial do PD da iteração anterior. Essa relaxação é obtida por meio da abordagem PIS. Já o conjunto de locais não promissores é definido como o complemento do conjunto de locais promissores.

Uma vez definidos os locais promissores e não promissores, o PE é obtido ao restringir os locais potenciais do SSCFLP ao conjunto de locais promissores. O PE é, então, resolvido e, se for viável, seu valor ótimo fornece um limitante superior (LS) para o SSCFLP, atualizando o LS incumbente, se aplicável. Já o PD é obtido adicionando ao modelo do PD da iteração anterior a condição de que pelo menos uma facilidade deve ser aberta em um local não promissor. O ponto-chave dessa construção é que, em cada iteração, o espaço de soluções do PD consiste nas soluções do SSCFLP que ainda não foram exploradas ao resolver PEs da iteração corrente ou de iterações anteriores. Tipicamente, o PD cobre um espaço de soluções muito grande. Assim, em vez de resolvê-lo diretamente, o NCS resolve sua relaxação parcial, o que pode levar a três situações:

- (i) Se a solução for inteira, ela fornece um LS para o SSCFLP e atualiza o melhor LS, se aplicável. O algoritmo é encerrado com valor ótimo igual ao LS incumbente;
- (ii) Se a solução não for inteira e seu valor ótimo for menor que o LS incumbente, então esse valor fornece um LI para o SSCFLP, e uma nova iteração é iniciada;
- (iii) Se a solução não for inteira e seu valor ótimo for maior ou igual ao LS incumbente, então o LS incumbente corresponde ao valor ótimo e o algoritmo é encerrado.

Um mecanismo importante incorporado ao NCS consiste em utilizar, em cada iteração, o LS incumbente como cota superior no modelo do PE. Essa estratégia exclui da região viável de cada PE as soluções com valores objetivos superiores ao valor corrente de LS, o que pode acelerar a sua resolução. No entanto, sua aplicação

Algoritmo 11 NCS**Entrada:** SSCFLP, TLimNCS, NCSTolerGap, U **Saída:** s'

```

1:  $PD_0 \leftarrow \text{SSCFLP}$ ,  $LS \leftarrow +\infty$ ,  $s' \leftarrow \emptyset$ ,  $k \leftarrow 0$ 
2: Aplique o procedimento PC+PIS e resolva  $RPD_0$ 
3: se  $s(RPD_0)$  é inteira então
4:    $s' \leftarrow \emptyset$ 
5:   retorne
6: fim se
7:  $LI \leftarrow \text{opt}(RPD_0)$ 
8: enquanto tempo decorrido pelo NCS  $\leq$  TLimNCS e  $\Delta(LI, LS)\% \geq \text{NCSTolerGap}$  faça
9:    $k \leftarrow k + 1$ 
10:  Considere o conjunto locais promissores  $\Omega_k = \{j \in J : y_j^* = 1 \text{ em } s(RPD_{k-1})\}$ 
11:  Formule o  $PE_k$  considerando as expressões do modelo SSCFLP e a restrição  $\sum_{j \in J \setminus \Omega_k} y_j \leq 0$ 
12:  se  $LS = +\infty$  e  $|\Omega_k| > U$  então
13:    Aplique o procedimento heurístico. Atualize  $LS$  e  $s'$ , se aplicável
14:  fim se
15:  se  $LS < +\infty$  então
16:    Adicione  $LS$  como cota superior para  $PE_k$ 
17:  fim se
18:  Resolva  $PE_k$ . Atualize  $LS$  e  $s'$ , se aplicável
19:  se  $\Omega_j = J$  então
20:    retorne
21:  fim se
22:  Formule  $PD_k$  adicionando  $\sum_{j \in J \setminus \Omega_k} y_j \geq 1$  ao modelo de  $PD_{k-1}$  e resolva  $RPD_k$ 
23:  se  $s(RPD_k)$  é inteira então
24:    se  $\text{opt}(RPD_k) < LS$  então
25:       $LS \leftarrow \text{opt}(RPD_k)$ ,  $s' \leftarrow s(RPD_k)$ 
26:    retorne
27:  fim se
28: fim se
29: fim enquanto

```

só é possível quando há uma cota superior disponível, isto é, quando pelo menos uma solução viável já foi encontrada pelo algoritmo. Assim, adota-se o seguinte procedimento: em cada iteração, caso não exista uma cota superior disponível e o PE tenda a ser difícil de resolver — isto é, quando o número de locais promissores é maior que um parâmetro U , aplica-se o procedimento heurístico desenvolvido nesta tese para encontrar uma solução de boa qualidade para o PE, fornecendo uma boa cota superior. Esse procedimento é detalhado na subseção seguinte.

O pseudocódigo do algoritmo NCS é apresentado no Algoritmo 11. Nesse algoritmo, $k \in \mathbb{Z}_+$ é um contador de iterações; PD_k ($k \geq 0$) e PE_k ($k \geq 1$) denotam, respectivamente, o PD e o PE da iteração k ; RPD_k ($k \geq 0$) denota a relaxação parcial de PD_k ; e $s(P)$ e $\text{opt}(P)$ denotam, respectivamente, a solução e o valor ótimo do problema P , com $P = PE_k$ ou $P = RPD_k$.

O algoritmo utiliza dois critérios de parada na linha 8. O primeiro critério para o algoritmo quando o tempo de execução excede o parâmetro TLimNCS e o segundo critério para quando uma solução ótima é encontrada. Seja $\Delta(v_1, v_2)\%$ o desvio

percentual relativo entre dois valores v_1 e $v_2 \neq 0$, dado por $\Delta(v_1, v_2)\% = |v_1 - v_2|/|v_2| \times 100\%$. Considera-se uma solução ótima quando $\Delta(\text{LS}, \text{LI})\%$ é menor que o parâmetro do algoritmo `NCSTolerGap`.

Suponha que todos os PEs e RPDs sejam resolvidos dentro do limite de tempo `TimLimNCS`. Nesse caso, se o SSCFLP for viável, a solução s' é ótima; caso contrário, $s' = \emptyset$. Se o algoritmo for interrompido devido ao limite de tempo e $s' \neq \emptyset$, então s' corresponde a uma solução heurística para o SSCFLP.

A Figura 5.1 ilustra o processo de ramificação da árvore de busca do algoritmo NCS. O nó raiz, denotado por PD_0 , corresponde ao SSCFLP. Inicialmente, a relaxação parcial de PD_0 é resolvida e, caso uma solução inteira não seja obtida, o nó é ramificado. Os locais com facilidades abertas na solução da relaxação parcial de PD_0 definem o conjunto de locais promissores Ω_1 da primeira iteração ($k = 1$). A ramificação gera os nós filhos PE_1 e PD_1 . O nó PE_1 corresponde ao SSCFLP restrito aos locais de Ω_1 . Já o nó PD_1 é obtido pela inclusão de uma restrição que impõe a abertura de pelo menos uma facilidade em um local não promissor, isto é, pertencente a $J \setminus \Omega_1$. Em seguida, resolvem-se o problema PE_1 e a relaxação parcial de PD_1 . Caso um critério de parada não seja satisfeito, a árvore é ramificada a partir de PD_1 . Nesse caso, os locais com facilidades abertas na solução da relaxação parcial de PD_1 definem o conjunto de locais promissores da segunda iteração ($k = 2$), denotado por Ω_2 . Os dois novos nós filhos são denotados por PE_2 e PD_2 . O nó PE_2 corresponde ao SSCFLP restrito aos locais de Ω_2 , enquanto o nó PD_2 é obtido pela adição, ao modelo de PD_1 , de uma restrição que exige a abertura de pelo menos uma facilidade em um local que não pertence a Ω_2 . O problema PE_2 e a relaxação parcial de PD_2 são resolvidos. Caso um critério de parada continue não sendo satisfeito, uma nova ramificação é realizada a partir de PD_2 . Esse processo é repetido sucessivamente até que um critério de parada seja satisfeito.

5.2.2.1 Uma heurística para resolver problemas esparsos

A heurística proposta para resolver os PEs gera uma solução inicial, possivelmente infactível, e aplica uma busca local em duas fases. Ela é baseada na heurística introduzida por Kong (2021).

Seja PE_k , $k \geq 1$, um problema esperso no Algoritmo 11 (NCS). A solução inicial é construída alocando cada cliente $i \in I$ ao local $j \in \Omega_k$ com o maior valor de x_{ij} na solução $s(\text{RPD}_{k-1})$. Dessa forma, a condição de fonte única é garantida por construção; no entanto, a solução gerada pode ser inviável, uma vez que o processo de alocação não considera as restrições de capacidade. Para avaliar soluções inviáveis, introduz-se o parâmetro $\rho > 0$, que representa o custo unitário associado à demanda

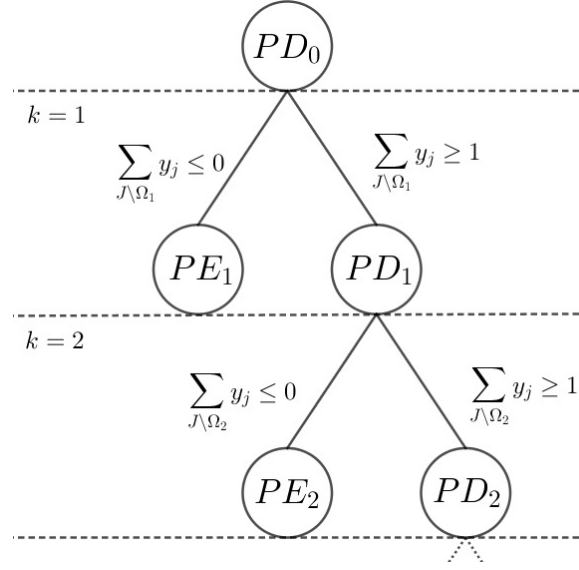


Figura 5.1: Ramificação da árvore de busca do algoritmo NCS.

que excede a capacidade. O valor de ρ escolhido deve ser suficientemente grande, de modo que qualquer solução viável tenha valor de função objetivo inferior ao de qualquer solução inviável.

Dada uma solução corrente s , na primeira fase seleciona-se iterativamente um subconjunto de facilidades $J^* \subset \Omega_k$, com $|J^*| = Q$, e resolve-se o SSCFLP restrito a J^* e ao subconjunto $I^* \subset I$ de clientes alocados às facilidades de J^* . Em seguida, obtém-se uma nova solução, mesclando a solução corrente com a solução ótima do SSCFLP restrito, da seguinte forma: para cada cliente $i \in I$, se $i \notin I^*$, sua alocação em s é mantida; caso contrário, i é alocado à facilidade $j \in J^*$ que o atende na solução do SSCFLP restrito. A estratégia *first improvement* é adotada para atualizar as soluções na primeira fase.

Dado Q , inicialmente igual ao parâmetro $Q_{\min}$, os SSCFLPs restritos são gerados para cada $j \in \Omega_k$, em ordem não crescente de $\sum_{i \in I^j} d_i - b_j$, em que I^j é o conjunto de clientes alocados à facilidade j . Dado j , o conjunto J^* é construído da seguinte forma: (i) adiciona-se j a J^* e seleciona-se o cliente $i' \in I^j$ com o menor valor de $c_{i'j}$; (ii) adicionam-se a J^* as $Q - 1$ facilidades com os menores valores de $c_{i'j}$. Caso todos os SSCFLPs restritos sejam explorados sem melhoria na solução corrente, o valor de Q é incrementado em uma unidade (linha 15), e o processo iterativo é reiniciado.

Dois critérios de parada são utilizados na primeira fase. Seja Q' o maior valor de Q para o qual a solução corrente é melhorada. O algoritmo é interrompido quando $Q > Q' + R$, em que R é um parâmetro da heurística, ou quando o tempo de execução excede o parâmetro $TLimHeur$. O Algoritmo 12 apresenta o pseudocódigo da primeira fase do procedimento de busca local.

Se a solução obtida na primeira fase da busca local for inviável, a segunda fase

Algoritmo 12 Busca Local – Fase I**Entrada:** Solução Inicial s , R , $Q_{\min}$, $TLimHeur$ **Saída:** s refinada

```

1:  $Q \leftarrow Q_{\min}$ ,  $Q' \leftarrow Q$ 
2: enquanto tempo decorrido pela Fase I  $\leq TLimHeur$  e  $Q \leq Q' + R$  faça
3:   Ordene as facilidades de  $\Omega_k$ 
4:   para cada  $j \in \Omega_k$  faça
5:     Selecione  $J^*$  e resolva o SSCFLP restrito
6:     Construa uma nova solução  $s'$ 
7:     se  $s'$  é melhor que  $s$  então
8:        $s \leftarrow s'$ ,  $Q \leftarrow Q_{\min}$ 
9:       se  $Q > Q'$  então
10:         $Q' \leftarrow Q$ 
11:     fim se
12:     Recomece na linha 2
13:   fim se
14:   fim para
15:    $Q \leftarrow Q + 1$ 
16: fim enquanto

```

é executada. Trata-se de uma adaptação da primeira, que visa aumentar a possibilidade de se obter uma solução factível. Mais precisamente, a segunda fase segue os procedimentos descritos no algoritmo 12, porém com três modificações: (i) os SSCFLPs restritos são construídos para cada facilidade $j \in \Omega_k$ com capacidade violada, em vez de considerar todas as facilidades $j \in \Omega_k$; (ii) na seleção de J^* e I^* , realizam-se os mesmos passos da primeira fase, mas, após isso, todas as facilidades com capacidade residual positiva que ainda não estejam incluídas também são adicionadas a J^* e, conseqüentemente, todos os clientes alocados a elas são incluídos em I^* ; (iii) se todos os SSCFLPs restritos forem construídos e resolvidos para cada $j \in \Omega_k$ com capacidade violada, sem melhoria da solução, a segunda fase é encerrada.

Se a segunda fase não conseguir gerar uma solução factível, a aplicação da heurística não produz efeito e o PE_k é resolvido sem considerar uma cota superior. O Algoritmo 13 apresenta o pseudocódigo da segunda fase da busca local.

Algoritmo 13 Busca Local – Fase II**Entrada:** solução s inviável para PE_k , $Q_{\min}$ **Saída:** s refinada

```

1: Melhora  $\leftarrow$  Verdadeiro
2:  $Q \leftarrow Q_{\min}$ 
3: enquanto Melhora = Verdadeiro faça
4:   Melhora  $\leftarrow$  Falso
5:   Selecione e ordene as facilidades de  $\Omega_k$  com capacidade violada
6:   para cada  $j \in \Omega_k$  com capacidade violada faça
7:     Selecione  $I^*$  e  $J^*$ 
8:     Resolva o SSCFLP restrito a  $I^*$  e  $J^*$ 
9:     Construa uma nova solução  $s'$  mesclando a solução do SSCFLP restrito com a solução  $s$ 
10:    se  $s'$  é melhor que  $s$  então
11:       $s \leftarrow s'$ 
12:      Melhora  $\leftarrow$  Verdadeiro
13:      Recomece na linha 3
14:    fim se
15:  fim para
16: fim enquanto
17: retorne  $s$ 

```

5.3 Algoritmo B&B-GVNS

O B&B-GVNS é composto por três fases. A fase I fortalece a formulação por meio de um procedimento de planos de corte. No melhor dos casos, a fase I fornece uma solução inteira e o algoritmo termina sem necessidade de executar as fases seguintes.

A fase II começa gerando uma solução inicial, aplicando a heurística construtiva descrita a seguir. Sejam x_{ij}^* e y_j^* os valores fracionários das variáveis x_{ij} e y_j da solução da relaxação linear obtida na fase anterior. Para cada cliente i , seleciona-se o local $j' = \arg \max_{j \in J} x_{ij}^*$. Em seguida, abre-se uma facilidade em j' , caso ainda não haja uma facilidade aberta nesse local, e aloca-se i nessa facilidade. Ao final do processo, obtém-se uma solução que geralmente é infactível, pois as restrições de capacidade não são consideradas na alocação dos clientes.

Após a geração da solução inicial, a fase II aplica, em sequência, um procedimento VND e dois procedimentos baseados na meta-heurística GVNS, denominados GVNS-I e GVNS-II. Ambos os GVNS utilizam a mesma componente VND, diferindo apenas nas estruturas de perturbação: o GVNS-I altera os locais com facilidades, bem como a alocação de clientes, enquanto o GVNS-II modifica apenas a alocação de clientes. Esses três procedimentos são detalhados na Subseção 5.3.0.2.

Na fase III do B&B-GVNS, seleciona-se um algoritmo baseado em *Branch-and-Bound* para resolver problemas de otimização com variáveis inteiras. Esse algoritmo pode ser, por exemplo, implementado em *solvers* como o CPLEX ou o Gurobi. Em seguida, aprimora-se esse algoritmo da seguinte forma: a cada ω iterações, aplica-se a heurística construtiva da linha 2 à solução da relaxação linear do nó corrente e, em

seguida, aplica-se o procedimento VND. Caso seja encontrada uma solução melhor do que a incumbente, aplica-se o GVNS-II para refiná-la ainda mais. Esse procedimento tem como objetivo acelerar a obtenção de limitantes superiores de boa qualidade ao longo da árvore de busca, melhorando o desempenho global do algoritmo. O GVNS-I não é utilizado nessa fase, pois experimentos preliminares indicaram perda de desempenho quando ele também é incluído.

O Algoritmo 14 apresenta o pseudocódigo do B&B-GVNS.

Algoritmo 14 Esquema do B&B-GVNS

Fase I

- 1: Fortaleça a formulação do RSSCFLP, aplicando um procedimento de planos de corte;

Fase II

- 2: Gere uma solução inicial (possivelmente infactível) aplicando a heurística construtiva;
- 3: Refine a solução inicial aplicando em sequência o VND, o GVNS-I e o GVNS-II;

Fase III

- 4: Selecione um algoritmo baseado em B&B para resolver MIPs;
 - 5: Aprimore o algoritmo baseado em B&B incorporando a cada ω iterações a heurística construtiva, o VND e o GVNS-II;
 - 6: Resolva o RSSCFLP utilizando o algoritmo baseado em B&B aprimorado.
-

5.3.0.1 Procedimento de planos de corte

No método de planos de corte, desigualdades válidas que eliminam soluções fracionárias são adicionadas iterativamente ao modelo de um problema inteiro ou inteiro misto, conforme descrito na Subseção 5.3.0.1 do Capítulo 2.

A componente de planos de corte do B&B-GVNS explora dois tipos de desigualdades válidas. O primeiro consiste no conjunto de desigualdades $x_{ij} \leq y_j, \forall i \in I, j \in J$, amplamente utilizado na literatura para fortalecer a formulação de problemas de localização. O segundo tipo é composto por desigualdades válidas, derivadas das restrições (3.11), considerando um caso específico de variação das demandas dos clientes. Mais precisamente, para cada facilidade j , considere um conjunto $S_j \subseteq I$ tal que $|S_j| \leq \Gamma_j^d$ e um cliente $t_j \in I \setminus S_j$. Então, a desigualdade:

$$\sum_{i \in I} d_i x_{ij} + \sum_{i \in S_j} \hat{d}_i x_{ij} + (\Gamma_j^d - \lfloor \Gamma_j^d \rfloor) \hat{d}_{t_j} x_{t_j j} \leq b_j y_j - \Gamma_j^b \hat{b}_j y_j \quad (5.16)$$

é válida para o RSSCFLP. Ela possui a estrutura de uma restrição de mochila, para a qual há extensa literatura sobre a geração de desigualdades válidas para esse tipo de restrição. No algoritmo B&B-GVNS propõe-se gerar LCIs e CFs seguindo os mesmos procedimentos utilizados para gerar esses cortes, explorando as restrições de

capacidade $\sum_{i \in I} d_i x_{ij} \leq b_j y_j$ do SSCFLP no algoritmo NCS descrito na Subseção 5.2.1 deste capítulo.

Para cada solução fracionária s^* do RSSCFLP e para cada $j \in J$, considere o vetor $\mathbf{s}_j^* = (x_{1j}^*, x_{2j}^*, \dots, x_{nj}^*, y_j^*)$ e X_j^y . Além disso, seja X_j^y a casca convexa do conjunto $\{(x_{1j}, \dots, x_{nj}, y_j) \in \{0,1\}^{n+1} : (5.16)\} \forall j \in J$. O pseudocódigo do procedimento de planos de corte proposto encontra-se no Algoritmo 15.

Algoritmo 15 Procedimento de planos de corte do B&B-GVNS

Entrada: Modelo do RSSCFLP

- 1: **enquanto** Critério de parada não é satisfeito **faça**
 - 2: Resolva a relaxação linear e obtenha a solução fracionária s^*
 - 3: Adicione ao modelo todas as desigualdades da forma $x_{ij} \leq y_j$ violadas por s^*
 - 4: **para** $j \in J$ **faça**
 - 5: Selecione S_j e t_j e obtenha a desigualdade (5.16)
 - 6: Adicione ao modelo um LCI ou um CF válido para X_j^y violado por $\mathbf{s}_j^*$
 - 7: **fim para**
 - 8: **fim enquanto**
 - 9: **retorne** Modelo do RSSCFLP fortalecido
-

Na linha 5, S_j e t_j são escolhidos de modo a aumentar a probabilidade de se obter um plano de corte violado. Para isso, adota-se a seguinte regra: sejam x_{ij}^* e y_j^* os valores assumidos pelas variáveis x_{ij} e y_j na solução da relaxação linear. Incluem-se em S_j os $|S_j|$ clientes com maiores valores de $\hat{d}_i x_{ij}^*$ e escolhe-se t_j como o cliente $i \in I \setminus S_j$ com maior valor de $(\Gamma_j^d - \lfloor \Gamma_j^d \rfloor) \hat{d}_{t_j} x_{t_j j}^*$.

5.3.0.2 Procedimentos VND, GVNS-I e GVNS-II

O procedimento VND do B&B-GVNS explora dois tipos de estruturas de vizinhança, denotadas por V_1 e V_2 . Elas são definidas, respectivamente, pelos movimentos:

- (i) m_1 (Realoca um cliente): realoca um cliente de uma facilidade que tem mais de um cliente alocado para outra facilidade;
- (ii) m_2 (Troca dois clientes): troca a alocação de dois clientes alocados em duas facilidades diferentes.

A estratégia *Pipe Neighborhood Change* é utilizada para definir a regra de trocas de estruturas. Ela consiste em manter a busca em uma estrutura de vizinhança enquanto houver melhora na solução corrente. Para explorar as vizinhanças, adota-se a estratégia *Best Improvement*. Como a complexidade de m_1 é menor do que a de m_2 , a estrutura V_1 é explorada, inicialmente, antes da estrutura V_2 . O Algoritmo 16 apresenta o pseudocódigo do procedimento VND proposto.

Algoritmo 16 VND do B&B-GVNS**Entrada:** Estruturas de vizinhança V_1 e V_2 ; solução corrente s

```

1:  $i \leftarrow 1$ 
2: enquanto  $i \leq 2$  faça
3:    $s' \leftarrow$  melhor solução em  $V_i(s)$ 
4:   se  $f(s') < f(s)$  então
5:      $s \leftarrow s', i \leftarrow 1$ 
6:   senão
7:      $i \leftarrow i + 1$ 
8:   fim se
9: fim enquanto
10: retorne solução refinada  $s$ 

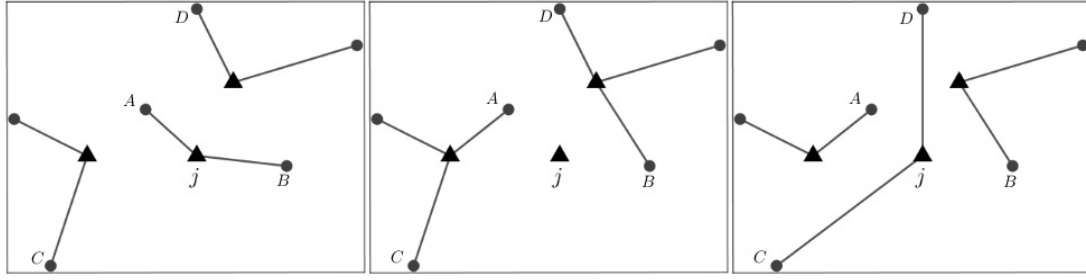
```

Para a descrição dos procedimentos GVNS, considere as perturbações que alteram o conjunto de locais com facilidades abertas:

- (i) p_1 (Abre uma facilidade): denote por α o número inteiro mais próximo da média de clientes atendidos por cada facilidade aberta. Em seguida, selecione um local j' para abrir uma nova facilidade. Por fim, realoque os α clientes i com menores custos de alocação para j' , proibindo que as facilidades de origem fiquem sem clientes atribuídos.
- (ii) p_2 (Fecha uma facilidade): selecione uma facilidade para fechar e realoque cada cliente atendido por ela para a facilidade aberta com menor custo de alocação.
- (iii) p_3 (Substitui facilidade): substitua o local de uma facilidade aberta, preservando os clientes alocados.

Considere também as perturbações p_4 e p_5 que alteram os clientes atendidos por uma e duas facilidades, respectivamente:

- (iv) p_4 (Realocação múltipla I): selecione uma facilidade j e seja $L(j)$ o conjunto de clientes atendidos por ela. Inicialmente, realoque cada cliente $i \in L(j)$ para a facilidade aberta com menor valor de c_{ij} . Em seguida, selecione $|L(j)|$ clientes $i \notin L(j)$ com os menores valores de c_{ij} e realoque-os para a facilidade j . A Figura 5.2 ilustra p_4 . Essa ilustração assume que c_{ij} representa a distância euclidiana entre o cliente i e a facilidade j . Inicialmente, os clientes A e B de j são realocados para as facilidades mais próximas. Em seguida, os clientes C e D, que são os clientes mais próximos de j distintos de A e B, são realocados para j .
- (v) p_5 (Realocação múltipla II): selecione duas facilidades e troque a alocação de seus clientes.

Figura 5.2: Ilustração da perturbação p_4 do B&B-GVNS.**Algoritmo 17** GVNS-I**Entrada:** Estruturas de vizinhança N_1 , N_2 e N_3 ; solução corrente s ; parâmetro α .

```

1: Pare  $\leftarrow$  Falso
2: enquanto Pare = Falso faça
3:   para  $i = 1$  até 3 faça
4:      $S_i \leftarrow \alpha$  melhores soluções de  $N_i(s)$ 
5:   fim para
6:    $S \leftarrow \bigcup_{i=1}^3 S_i$ 
7:   Aplique o VND em cada solução de  $S$  e considere  $s'$  a melhor solução obtida
8:   se  $s'$  é melhor do que  $s$  então
9:      $s \leftarrow s'$ 
10:  senão
11:    Pare  $\leftarrow$  Verdadeiro
12:  fim se
13: fim enquanto
14: retorne solução refinada  $s$ 

```

Algoritmo 18 GVNS-II**Entrada:** Estruturas de vizinhança N_4 e N_5 ; solução corrente s , parâmetro α .

```

1:  $i \leftarrow 4$ 
2: enquanto  $i \leq 5$  faça
3:    $S_i \leftarrow \alpha$  melhores soluções de  $N_i(s)$ 
4:    $s' \leftarrow$  melhor solução obtida aplicando o VND em soluções de  $S_i$ 
5:   se  $s'$  é melhor do que  $s$  então
6:      $s \leftarrow s'$ ,  $i \leftarrow 4$ 
7:   senão
8:      $i \leftarrow i + 1$ 
9:   fim se
10: fim enquanto
11: retorne Solução refinada  $s$ 

```

Denote por N_i as estruturas de vizinhança definidas pelas perturbações p_i , $i = 1, \dots, 5$. Em ambos os procedimentos GNVS, a estratégia de exploração de uma estrutura N_i a partir de uma solução corrente s consiste em, inicialmente, obter o conjunto $N_i(s)$ de todas as soluções obtidas ao aplicar p_i a s . Em seguida, a

vizinhança é reduzida selecionando uma quantidade, definida pelo parâmetro α , de soluções perturbadas mais bem avaliadas para aplicação do VND.

Os pseudocódigos dos procedimentos GNVS-I e GVNS-II são apresentados nos Algoritmos 17 e 18. O GVNS-I explora as estruturas N_1 , N_2 e N_3 . Movimentos que alteram locais com facilidades abertas são difíceis de reverter. Então, em cada iteração, opta-se por garantir que a melhor solução que pode ser obtida ao explorar essas estruturas seja selecionada. Para isso, em cada iteração, une-se a um único conjunto S as soluções perturbadas obtidas ao explorar N_1 , N_2 e N_3 , e aplica-se o VND a todas as soluções de S . Já no GVNS-II, a estrutura N_4 é explorada antes da N_5 porque, geralmente, p_4 fornece soluções com modificações menores do que p_5 .

Capítulo 6

Experimentos Computacionais

Este capítulo apresenta os experimentos computacionais realizados com os algoritmos HILS, NCS e B&B-GVNS. A Seção 6.1 descreve os conjuntos de instâncias utilizados nos testes computacionais. Os experimentos realizados com os algoritmos HILS, NCS e B&B-GVNS, bem como seus respectivos resultados e análises, são apresentados nas Seções 6.2, 6.3 e 6.4, respectivamente.

6.1 Conjuntos de instâncias de teste

Os conjuntos de instâncias considerados nos testes computacionais incluem seis *benchmarks* clássicos do SSCFLP. Nesta tese, esses conjuntos são denominados TB X , com $X = 1, \dots, 6$.

O conjunto TB1 proposto em [Holmberg et al. \(1999\)](#) é composto pelas instâncias h1–h71, que são divididas em subconjuntos com diferentes características. Nos subconjuntos h1-h12, h13-h24 e h56-h71 as coordenadas dos nós do SSCFLP são geradas aleatoriamente em um quadrado e o custo de atendimento é proporcional à distância euclidiana. No subconjunto h25-h40, o custo de alocação considera uma parcela fixa mais uma proporcional à demanda do cliente. Já o subconjunto h41-h55 adapta instâncias clássicas de [Solomon \(1987\)](#) para roteamento de veículos, acrescentando custos de abertura e capacidades às instâncias. As instâncias de TB1 são consideradas de fácil resolução, pois podem ser resolvidas na otimalidade rapidamente pelo CPLEX.

O conjunto TB2, composto pelas instâncias y1-y20 foi proposto em [Yang et al. \(2012\)](#). Nele os custos de alocação entre clientes e facilidades são proporcionais às distâncias euclidianas entre os nós que os representam. Já o custo fixo de abertura é composto por uma parcela aleatória mais uma parcela proporcional à raiz quadrada da capacidade da facilidade. Em relação ao conjunto de [Holmberg et al. \(1999\)](#), as ins-

tâncias propostas por [Yang et al. \(2012\)](#) são mais difíceis de serem resolvidas. Ainda sim, o CPLEX consegue resolver boa parte dessas instâncias rapidamente, enquanto algumas exigem algumas horas de processamento. O conjunto TB3, proposto em [Gadegaard et al. \(2018\)](#), contém as instâncias n1-n45. Elas foram geradas de maneira similar às instâncias de [Yang et al. \(2012\)](#), mas as regras de geração foram ajustadas para levar em conta que altos custos de abertura levam a menores custos de alocação. Essa adaptação aumenta o conflito, característico de problemas de localização, de minimizar ao mesmo tempo os custos de abertura e de alocação. Experimentos na literatura mostram que esse conjunto é significativamente mais desafiador do que o conjunto TB2 já que várias instâncias não têm valores ótimos conhecidos.

O conjunto TB4 proposto por [Beasley \(1988\)](#) contém 12 instâncias divididas nos subconjuntos capa, capb e capc. As instâncias de TB4 possuem 1000 clientes e 100 locais (mais de 1000 nós), sendo consideradas de grande porte para o SSCFLP. Apesar disso, elas podem ser resolvidas pelo CPLEX rapidamente. O conjunto TB5, composto por 100 instâncias, foi gerado de maneira similar ao conjunto de [Yang et al. \(2012\)](#). As principais diferenças desse conjunto em relação aos demais são o maior porte das instâncias, que chegam a ter 1500 clientes e 1000 locais, e a igualdade entre o número de clientes e de locais em 80 das 100 instâncias. Com isso, enquanto nos demais conjuntos o número de locais disponíveis chega, a no máximo 100, no conjunto TB5 ele é no mínimo 300. O conjunto TB5 pode ser considerado o *benchmark* mais desafiador para o SSCFLP, uma vez que quase metade de suas instâncias ainda não possui valores ótimos conhecidos.

Por fim, o conjunto de [Delmaire et al. \(1999\)](#) é composto por 57 instâncias de pequeno porte que foram geradas de forma que os custos de alocação possuam duas componentes: uma parcela fixa associada à facilidade e uma parcela proporcional à demanda do cliente. Além disso, os custos fixos de abertura também são proporcionais à capacidade da facilidade, com um fator de proporcionalidade aleatório. Esse conjunto é mais desafiador do que o de [Holmberg et al. \(1999\)](#), pois suas instâncias possuem capacidades bastante restritivas: a capacidade total é apenas 1% superior à demanda total. Apesar disso, experimentos da literatura mostram que elas podem ser resolvidas rapidamente pelo CPLEX.

Em conjunto, os seis *benchmarks* considerados abrangem uma ampla variedade de características do SSCFLP, incluindo diferentes procedimentos de geração e portes de instâncias. Essa diversidade permite avaliar os métodos propostos sob cenários bastante distintos, tornando as conclusões experimentais mais abrangentes e representativas. A Tabela 6.1 apresenta um resumo dos *benchmarks* considerados. Nela apresenta-se para cada conjunto a mesma divisão em subconjuntos adotada nos trabalhos que propuseram as instâncias. As colunas “Inst”, “Opt”, $|I|$ e $|J|$ indicam,

Tabela 6.1: Conjuntos de instâncias para testes

Conjunto	Subconjunto	#Inst	#Opt	$ I $	$ J $
TB1 (Holmberg) Holmberg et al. (1999)	h1-h12	12	12	50	10
	h13-h24	12	12	50	20
	h25-h40	16	16	150	30
	h41-h55	15	15	70-100	10-30
	h56-h71	16	16	200	30
TB2 (Yang) Yang et al. (2012)	y1-y5	5	5	200	30
	y6-y10	5	5	200	60
	y11-y15	5	5	300	60
	y16-y20	5	5	400	80
TB3 (Gadegaard) Gadegaard et al. (2018)	n1-n5	5	5	100	50
	n6-n10	5	5	100	50
	n11-n15	5	5	100	50
	n16-n20	5	3	200	50
	n21-n25	5	5	200	50
	n26-n30	5	5	200	50
	n31-n35	5	0	300	60
	n36-n40	5	5	300	60
TB4 (OR-4) Beasley (1990)	capa	4	4	1000	100
	capb	4	4	1000	100
	capc	4	4	1000	100
TB5 (Avella) Avella & Boccia (2009)	i300	20	15	300	300
	i300i500	20	20	1500	300
	i500	20	14	500	500
	i700	20	6	700	700
	i1000	20	0	1000	1000
TB6 (Díaz) Díaz & Fernández (2002)	d1-d6	6	6	20	10
	d7-17	11	11	30	15
	d18-d25	8	8	40	20
	d26-d33	8	8	50	20
	d34-d41	8	8	60	30
	d42-d49	8	8	75	30
	d50-d57	8	8	90	30

respectivamente, o número de instâncias, o número de ótimos conhecidos, de clientes e de locais em cada subconjunto.

6.2 Experimentos com o algoritmo HILS

O algoritmo HILS foi testado nos conjuntos *benchmark* de grande porte TB4 e TB5 da Tabela 6.1. A qualidade das soluções fornecidas foi comparada à dos algoritmos heurísticos do estado da arte KS ([Guastaroba & Speranza, 2014](#)), *Hypergraph Multi-*

Exchange Heuristic (HMEH) (Tran et al., 2017) e *Corridor Method* (CM) (Caserta & Vof, 2020). Idealmente, todos os algoritmos deveriam ser executados na mesma máquina e com a mesma versão de *solver*, quando aplicável. No entanto, não foi possível obter acesso aos códigos-fonte dos demais algoritmos. Consequentemente, a comparação com o HILS foi feita com base nos resultados reportados nos estudos que o propuseram.

O algoritmo proposto foi implementado na linguagem C++ e executado em um computador Intel Core i5-4210U de 2.40 GHz, 64 bits, com 8 GB de memória RAM e sistema operacional Ubuntu 18.04.5 LTS. O *solver* CPLEX 22.1 foi utilizado para resolver SSCFLPs reduzidos e problemas GAP nas linhas 6 e 11 do Algoritmo 7. Na implementação, as restrições redundantes dadas pelas sentenças (3.6) não foram incluídas na formulação do SSCFLP para o conjunto TB5. Essa escolha foi feita porque, em execuções preliminares, observou-se que o aumento no número de restrições geradas por essas restrições tornava o algoritmo menos eficiente nesse conjunto de dados. Essa decisão também foi adotada por Guastaroba & Speranza (2014) para resolver subproblemas em instâncias desse conjunto.

6.2.1 Calibração de parâmetros

O algoritmo HILS possui os seguintes parâmetros: (i) `tilim1` e `tilim2`, que determinam, respectivamente, os tempos para resolver o SSCFLP reduzido e o GAP pelo *solver* nas fases I e II do algoritmo; (ii) δ , que determina o valor máximo para os custos reduzidos das variáveis de localização na resolução de $\mathcal{LP}(\text{SSCFLP})$ para os locais inicialmente considerados promissores; (iii) γ , que determina o número de trocas aleatórias no procedimento ILS–II; e (iv) `ILSmaxI` e `ILSmaxII`, que determinam o número máximo de iterações sem melhoria dos procedimentos ILS–I e ILS–II.

O *software* IRACE (López-Ibáñez et al., 2016) foi utilizado para ajustar automaticamente os parâmetros do HILS. Para realizar a calibração, cinco instâncias com diferentes características foram selecionadas para treinamento. As características avaliadas foram: (i) o número total de clientes e locais potenciais $m + n$; (ii) a razão $r1$ entre o número de clientes e o número de locais potenciais; (iii) a razão $r2$ entre a demanda total e a capacidade total; (iv) a razão $r3$ entre o número de locais com facilidades abertas ou parcialmente abertas na solução da relaxação linear e o número total de locais; e (v) a razão $r4$ entre o custo total de alocação de clientes e o custo total de abertura de facilidades na solução da relaxação linear. As instâncias selecionadas e os valores de $m + n$, $r1$, $r2$, $r3$ e $r4$ são apresentados na Tabela 6.2.

Três valores possíveis de cada parâmetro foram fornecidos ao IRACE, com base em execuções preliminares. A Tabela 6.3 apresenta, para cada parâmetro do algoritmo

Tabela 6.2: Instâncias para treinamento do HILS e suas características.

Instância	$m + n$	$r1$	$r2$	$r3$	$r4$
capa3	1100	0,1	0,042	0,080	1,786
i300_16	600	1,0	0,050	0,063	2,266
i300i500_7	1800	0,2	0,100	0,357	3,327
i500_2	1000	1,0	0,200	0,176	0,296
i1000_9	2000	1,0	0,100	0,087	0,654

HILS, os valores fornecidos ao *software* IRACE. Os valores em negrito correspondem à melhor configuração encontrada pela ferramenta.

Tabela 6.3: Resultado da calibração de parâmetros do HILS

Parâmetro	Valores	Parâmetro	Valores
tilim1	{600, 800, 1000 }	γ	{0,000, 0,004, 0,008 }
tilim2	{100, 200 , 300}	ILSmaxI	{100, 150 , 200}
δ	{0,00, 0,25 , 0,50}	ILSmaxII	{90, 120 , 150}

6.2.2 Resultados e Análises

O algoritmo HILS foi executado cinco vezes para cada instância dos conjuntos TB4 e TB5. Para cada instância, apurou-se o melhor valor de função objetivo encontrado, denotado por f_{Best} , e calculou-se a média $\bar{f}$ dos valores das funções objetivo e a média t dos tempos de execução. Em seguida, calculou-se o desvio relativo percentual entre $\bar{f}$ e o valor ótimo f^* , denotado por Δ , utilizando a expressão:

$$\Delta = \frac{\bar{f} - f^*}{f^*} \times 100 \quad (6.1)$$

Para instâncias do conjunto TB5 sem valores ótimos conhecidos, os valores de Δ foram calculados utilizando os limitantes inferiores disponíveis na literatura, em substituição a f^* . Em [Guastaroba & Speranza \(2014\)](#) e [Tran et al. \(2017\)](#), foi utilizado o mesmo conjunto de limitantes inferiores para o cálculo dos desvios relativos, denominado LI1. Por outro lado, em [Caserta & Voß \(2020\)](#), foi utilizado um conjunto distinto, denominado LI2. Por isso, para cada instância de TB5, foi calculado o desvio relativo Δ_1 utilizando-se o limitante inferior correspondente de LI1 e o desvio relativo Δ_2 utilizando-se o limitante inferior de LI2.

Os resultados detalhados dos experimentos estão disponíveis na Seção [A.1](#) do Apêndice [A](#). O restante desta seção está organizado da seguinte forma. Na Subseção [6.2.2.1](#), são apresentados os resultados comparativos do HILS com os algoritmos estado da arte. A Subseção [6.2.2.2](#) mostra uma análise do procedimento inicial de

identificação de locais promissores. Na Subseção 6.2.2.3, analisa-se a influência dos componentes do HILS na qualidade das soluções geradas. Por fim, na Subseção 6.2.2.4, analisa-se a influência do uso de uma versão mais antiga do CPLEX nos resultados obtidos.

6.2.2.1 Comparação com algoritmos estado da arte

Para a comparação dos resultados do HILS com os obtidos por outros algoritmos da literatura, foram desconsiderados os resultados das instâncias utilizadas na calibração, a fim de evitar a superestimação de desempenho nessas instâncias. Os conjuntos TB4* e TB5* foram definidos como os obtidos pela remoção das instâncias utilizadas na calibração dos conjuntos TB4 e TB5, respectivamente. Para comparação com o algoritmo HMEH, não foi possível utilizar o conjunto TB5*, pois em Tran et al. (2017) não foram reportados resultados para cada instância, o que impossibilitou ajustar os resultados desse algoritmo no conjunto TB5*.

A Tabela 6.4 compara os resultados obtidos pelos algoritmos HILS, KS, HMEH e CM no conjunto de dados TB4*. As Tabelas 6.5 e 6.6 apresentam os resultados comparativos do algoritmo HILS com os algoritmos KS e CM para o conjunto T5*, respectivamente. Por sua vez, a Tabela 6.7 apresenta os resultados obtidos pelo algoritmo proposto e pelo HMEH no conjunto TB5. Nessas tabelas, as colunas $\bar{\Delta}$, $\bar{\Delta}_1$, $\bar{\Delta}_2$ e Opt apresentam, respectivamente, as médias de Δ , Δ_1 , Δ_2 e o número de soluções ótimas obtidas em cada subconjunto de instâncias. As colunas $\bar{t}$ apresentam os tempos médios de execução, em segundos. A linha Avg./Sum apresenta a média dos valores das colunas $\bar{\Delta}$, $\bar{\Delta}_1$, $\bar{\Delta}_2$, $\bar{t}$ e a soma dos valores da coluna Opt. Todos os subconjuntos de TB4 e TB5 com instâncias removidas devido à calibração aparecem nas tabelas de resultados com o símbolo *.

Para o conjunto de dados TB4*, o HILS obteve $\bar{\Delta} = 0,000$ em todos os subconjuntos e encontrou soluções ótimas para todas as instâncias. Esses resultados são equivalentes aos obtidos pelos algoritmos KS e CM e superiores aos do HMEH, que apresentou valor médio de $\bar{\Delta}$ igual a 0,047 e encontrou soluções ótimas em 7 das 11 instâncias.

Para o conjunto TB5*, a média dos valores de $\bar{\Delta}_1$ do HILS foi de 0,55, enquanto a média correspondente do KS foi de 0,63. De forma análoga, a média dos valores de $\bar{\Delta}_2$ do HILS foi de 0,41, enquanto a média correspondente do CM foi de 0,49. Já no conjunto TB5, o algoritmo HILS apresentou $\bar{\Delta}_1 = 0,55$, enquanto o HMEH apresentou $\bar{\Delta}_1 = 0,60$. Observa-se que os valores de $\bar{\Delta}_1$ e $\bar{\Delta}_2$ obtidos pelo HILS foram, em geral, inferiores aos dos demais algoritmos. Além disso, em todos os subconjuntos, os valores obtidos pelo HILS foram menores ou iguais aos dos demais

Tabela 6.4: Resultados comparativos dos algoritmos HILS, KS, HMEH e CM no conjunto TB4.

TB4*	HILS			KS		
	$\bar{\Delta}$	Opt	$\bar{t}$	$\bar{\Delta}$	Opt	$\bar{t}$
capa*	0,000	3	257,31	0,00	3	22,61
capb	0,000	4	181,25	0,00	4	51,42
capc	0,000	4	116,02	0,00	4	30,61
Avg./Sum	0,000	11	184,86	0,00	11	36,00

TB4*	HMEH			CM		
	$\bar{\Delta}$	Opt	$\bar{t}$	$\bar{\Delta}$	Opt	$\bar{t}$
capa*	0,002	2	39,16	0,000	3	49,33
capb	0,076	3	51,53	0,000	4	47,75
capc	0,053	2	34,66	0,000	4	31,00
Avg./Sum	0,047	7	42,02	0,000	11	42,09

Tabela 6.5: Resultados comparativos dos algoritmos HILS e KS no conjunto TB5

TB5*	HILS			KS		
	$\bar{\Delta}_1$	Opt	$\bar{t}$	$\bar{\Delta}_1$	Opt	$\bar{t}$
i300*	0,47	12	753,87	0,59	11	2322,80
i300i500*	0,00	19	133,54	0,00	19	339,59
i500*	0,52	7	2057,36	0,60	7	4163,30
i700	0,77	4	4493,49	0,90	2	5244,69
i1000*	0,98	0	7294,37	1,06	0	6476,36
Avg./Sum	0,55	42	2962,68	0,63	39	3725,34

algoritmos, exceto no subconjunto i500, no qual o HMEH apresentou o menor valor. Esses resultados indicam que, para o conjunto TB5, o HILS gera soluções de qualidade ligeiramente superior às dos demais algoritmos.

Também foi analisado o número de soluções ótimas encontradas pelos algoritmos. O HILS obteve 42 soluções ótimas no conjunto TB5*. Esse número é superior ao obtido pelo KS (“Opt” = 39) e igual ao obtido pelo CM. No conjunto TB5, o HILS encontrou 42 soluções ótimas, enquanto o HMEH encontrou 28. Destaca-se, ainda, que no subconjunto i300*, o HILS foi o algoritmo que encontrou o maior número de soluções ótimas. Esses resultados indicam que o HILS encontra soluções ótimas com frequência igual ou superior à dos demais algoritmos.

Em [Guastaroba & Speranza \(2014\)](#) e [Caserta & Voß \(2020\)](#), foram disponibilizados resultados detalhados para cada instância. Esses resultados foram utilizados na realização de dois testes estatísticos de Wilcoxon ([Wilcoxon, 1945](#)) no conjunto

Tabela 6.6: Resultados comparativos dos algoritmos HILS e CM no conjunto TB5.

TB5*	HILS			CM		
	$\overline{\Delta}_2$	Opt	$\bar{t}$	$\overline{\Delta}_2$	Opt	$\bar{t}$
i300*	0,15	12	753,87	0,24	11	848,84
i3001500*	0,00	19	133,54	0,00	19	28,32
i500*	0,28	7	2057,36	0,31	8	984,68
i700	0,62	4	4493,49	0,78	4	912,90
i1000*	0,97	0	7294,37	1,10	0	944,68
Avg./Sum	0,41	42	2962,68	0,49	42	745,65

Tabela 6.7: Resultados comparativos dos algoritmos HILS e HMEH no conjunto TB5.

TB5	HILS			HMEH		
	$\overline{\Delta}_1$	Opt	$\bar{t}$	$\overline{\Delta}_1$	Opt	$\bar{t}$
i300	0,45	12	718,53	0,54	10	428,03
i3001500	0,00	19	132,57	0,01	14	1159,33
i500	0,58	7	2163,82	0,52	4	2982,72
i700	0,77	4	4493,49	0,82	0	4992,14
i1000	0,97	0	7321,22	1,10	0	8582,74
Avg./Sum	0,55	42	2965,93	0,60	28	3628,99

TB5*. Um deles considerou os desvios relativos pareados obtidos pelos algoritmos HILS e KS, enquanto o outro considerou os desvios relativos pareados obtidos pelos algoritmos HILS e CM. Em ambos os casos, adotou-se uma hipótese alternativa unilateral, segundo a qual o algoritmo HILS fornece soluções de melhor qualidade do que os demais. Os resultados evidenciaram, com significância estatística, que o HILS fornece soluções de melhor qualidade do que o algoritmo KS (p -valor = 0,0013) e do que o algoritmo CM (p -valor = 0,0152).

Por fim, avaliou-se a capacidade do algoritmo HILS de gerar soluções melhores do que as então conhecidas para as 55 instâncias do conjunto TB5 cujos valores ótimos são desconhecidos. Para isso, o valor de f_{best} de cada instância foi comparado com o respectivo melhor valor conhecido da literatura (*Best Known Value* - BKV). O algoritmo HILS atualizou 22 BKVs, o que corresponde a 40% das instâncias sem valores ótimos conhecidos. A Tabela 6.8 apresenta esses resultados. Nela, a coluna “Melhoria(%)” apresenta o percentual de melhoria, calculado pela fórmula $100 \times (\text{BKV} - f_{\text{best}})/\text{BKV}$.

Tabela 6.8: Novos BKVs fornecidos pelo HILS.

Inst.	BKV	f_{best}	t	Melhoria (%)
i300_2	16140,00 ⁽⁶⁾	16138,88	2068,89	0,007
i300_3	15687,38 ⁽⁵⁾	15686,29	2183,93	0,007
i300_4	18312,60 ⁽⁶⁾	18215,87	2094,75	0,528
i300_5	18315,44 ⁽⁶⁾	18288,06	2141,77	0,149
i500_1	26824,44 ⁽¹⁾	26774,78	4654,67	0,185
i500_3	28362,14 ⁽²⁾	28275,24	2721,74	0,306
i700_1	37616,61 ⁽⁴⁾	37474,45	15778,03	0,380
i700_2	35076,86 ⁽¹⁾	34899,95	9752,17	0,504
i700_3	34977,47 ⁽¹⁾	34903,76	6746,31	0,210
i700_7	21437,82 ⁽⁴⁾	21436,12	6139,86	0,008
i700_8	20823,17 ⁽¹⁾	20798,33	4021,74	0,119
i700_10	22274,57 ⁽⁴⁾	22219,51	3431,56	0,247
i1000_1	50732,70 ⁽³⁾	50371,91	14365,09	0,711
i1000_2	51677,00 ⁽¹⁾	51525,90	10841,45	0,292
i1000_3	48141,97 ⁽¹⁾	47922,16	13123,62	0,457
i1000_4	49909,44 ⁽²⁾	47922,16	9931,21	0,234
i1000_5	51824,38 ⁽¹⁾	51642,78	10740,55	0,350
i1000_7	27522,12 ⁽¹⁾	27397,48	4946,24	0,453
i1000_8	27638,39 ⁽³⁾	27546,78	5812,37	0,331
i1000_9	27127,70 ⁽⁴⁾	27076,45	6061,24	0,329
i1000_10	27469,73 ⁽¹⁾	27388,50	7631,24	0,296
i1000_20	21618,06 ⁽⁴⁾	21613,16	6186,31	0,023

O algoritmo HILS atualizou as soluções de 10 das 20 instâncias do subconjunto i1000, que pode ser considerado o subconjunto mais difícil de instâncias a ser resolvido, uma vez que não há solução ótima conhecida para nenhuma instância. A maior melhoria ocorreu para a instância i1000_1, com um ganho de 0,711%.

Os tempos computacionais obtidos pelo HILS não são diretamente comparáveis aos reportados por outros autores, pois as configurações das máquinas utilizadas diferem significativamente das adotadas nos experimentos dos demais algoritmos. Considerando que o SSCFLP é aplicado em decisões estratégicas, a principal exigência em relação ao tempo computacional é que ele seja viável na prática. Os tempos médios obtidos para os conjuntos TB4*, TB5* e TB5 foram, respectivamente, 184,86, 2962,68 e 2965,93 segundos. O maior valor de t observado foi de 12988,66 segundos, correspondente à instância i1000_1. Esse tempo é viável em aplicações práticas do SSCFLP, uma vez que decisões estratégicas de localização geralmente são tomadas em horizontes de planejamento de médio e longo prazo.

6.2.2.2 Análise do procedimento identificação de locais promissores

O procedimento inicial para a seleção de locais promissores é semelhante ao do algoritmo KS. Recorde que ele seleciona os locais $j \in J$ que satisfazem a condição $\hat{c}(y_j) \leq \delta \times \sigma$, em que $\hat{c}(y_j)$ é o custo reduzido de y_j na solução da relaxação linear do SSCFLP, σ é a média dos custos reduzidos positivos das variáveis y_j e δ é um parâmetro do procedimento. Para avaliar esse procedimento, apurou-se para cada instância de TB4 e TB5 com valor ótimo conhecido (57 instâncias) e para cada valor de $\delta \in \{0, 0,25, 0,50\}$: o percentual de locais considerados promissores e se o conjunto de locais promissores coincidiu com o de locais com facilidades abertas na solução ótima.

A Tabela 6.9 apresenta um resumo dos resultados. As colunas Min, Avg e Max representam, respectivamente, os percentuais mínimo, médio e máximo de locais considerados promissores para os valores 0,00, 0,25 e 0,50 de δ . A coluna N mostra o número de instâncias em que o conjunto de locais promissores não coincidiu com o de locais com facilidades abertas na solução ótima.

Tabela 6.9: Resultado da análise do procedimento de identificação de locais promissores do HILS

		$\delta = 0,00$				$\delta = 0,25$			
Data set	Inst	Min	Avg	Max	N	Min	Avg	Max	N
TB4	12	4	28,9	96,0	1	8,0	39,2	98,0	0
TB5	45	9	84,5	99,0	7	20,7	87,3	99,7	1
		$\delta = 0,50$							
Data set	#Inst	Min	Avg	Max	N				
TB4	12	23,0	46,3	98,0	0				
TB5	45	32,7	90,5	99,8	1				

Os resultados mostram que o número de locais selecionados pode variar consideravelmente de uma instância para outra, independentemente do valor de δ . Esse comportamento pode ser observado pela grande diferença entre os valores das colunas “Min” e “Max” em ambos os conjuntos. Valores próximos de 100 na coluna “Max” indicam que há instâncias em que o procedimento considera quase todos os locais como promissores. Verificou-se que isso ocorre porque, em determinadas instâncias, há um grande número de locais $j \in J$ com $\hat{c}(y_j)$ positivos, porém muito próximos de zero. Para esses casos, a condição $\hat{c}(y_j) \leq \delta \times \sigma$ tende a ser satisfeita para qualquer valor de $\delta \geq 0$. Por outro lado, os baixos valores nas colunas “Min” indicam que há

instâncias em que o procedimento considera apenas uma pequena fração dos locais como promissores. Como esperado, o aumento do valor de δ elevou o percentual médio de locais considerados promissores.

Para os dois valores de $\delta = 0,25$ e $\delta = 0,50$, obteve-se $N = 0$ no conjunto TB4 e $N = 1$ no conjunto TB5. Esse resultado mostra que a identificação de locais promissores falhou em selecionar exatamente o conjunto de locais com facilidades abertas na solução ótima em apenas uma instância dentre 57 (ou seja, 1,75% do total de instâncias). A escolha $\delta = 0,25$ mostrou-se mais vantajosa, pois os percentuais médios de locais considerados promissores são menores do que os correspondentes para $\delta = 0,50$, reduzindo ainda mais o espaço de busca sem eliminar a solução ótima.

6.2.2.3 Análise das componentes do algoritmo HILS

Para analisar o efeito de cada componente no desempenho do HILS, foram realizados experimentos com variantes obtidas pela remoção de uma de suas componentes. Cada variante foi denominada HILS-X, sendo X igual a A, B, C, D, E ou F, conforme descrito na Tabela 6.10. Esta tabela apresenta, na primeira coluna, o nome da variante; na segunda, a componente removida; e, na terceira, as linhas correspondentes removidas no pseudocódigo do HILS descrito pelo Algoritmo 7 da Seção 5.1.

Tabela 6.10: Variantes do algoritmo HILS obtidas removendo uma componente

Variante	Componente removida	Linhas no pseudocódigo
HILS-A	Procedimento de abertura	linha 3
HILS-B	Procedimento VND	linha 4
HILS-C	Resolução do SSCFLP Restrito pelo <i>solver</i>	linha 6
HILS-D	Procedimento ILS-I	linha 7
HILS-E	Resolução do GAP pelo <i>solver</i>	linha 11
HILS-F	Procedimento ILS-II	linhas 8 e 12

Cada variante HILS-X foi executada cinco vezes em cada instância utilizada na calibração, bem como na instância i1000_1, que apresentou o maior tempo de execução no experimento com o HILS. Para cada variante, foram calculados os desvios percentuais relativos entre as médias dos valores da função objetivo e o valor ótimo para a instância capa3, e entre as médias dos valores da função objetivo e os limitantes inferiores de LI2 para as demais instâncias.

As colunas $d(\Delta)$ da Tabela 6.11 apresentam a diferença entre os desvios relativos obtidos nas variantes HILS-X e os obtidos pelo algoritmo HILS em sua versão completa. Valores positivos de $d(\Delta)$ indicam pior desempenho da variante HILS-X. Por exemplo, na instância i1000_1, houve um aumento de +0,13 no valor de Δ ao utili-

Tabela 6.11: Resultado da avaliação de componentes do HILS

	HILS-A		HILS-B		HILS-C	
	$d(\Delta)$	$Dev(t)(\%)$	$d(\Delta)$	$Dev(t)(\%)$	$d(\Delta)$	$Dev(t)(\%)$
capa3	0,00	-4,13	0,00	+0,69	+0,06	+67,54
i300_16	0,00	-13,68	0,00	-15,29	+0,36	-21,09
i3001500_7	0,00	-4,40	0,00	-0,21	+0,10	+861,94
i500_2	+0,04	-10,26	-0,01	+50,03	+0,33	-57,34
i1000_9	+0,19	-10,84	+0,17	5,68	+0,11	-54,07
i1000_1	+0,13	+12,55	+0,12	-7,35	+0,13	-30,44

	HILS-D		HILS-E		HILS-F	
	$d(\Delta)$	$Dev(t)(\%)$	$d(\Delta)$	$Dev(t)(\%)$	$d(\Delta)$	$Dev(t)(\%)$
capa3	0,00	-15,50	0,00	-2,08	0,00	-3,95
i300_16	0,00	+20,44	0,00	-0,03	0,00	+0,91
i3001500_7	0,00	-52,07	0,00	+0,75	0,00	-2,90
i500_2	+0,04	+16,79	-0,03	+5,46	-0,02	+2,05
i1000_9	0,00	+25,10	+0,07	-36,86	-0,01	-6,99
i1000_1	+0,15	+23,86	+0,11	-21,87	+0,07	-8,34

zar a variante HILS-A, indicando que essa variante gera soluções que apresentam, em média, um desvio relativo médio 0,13 pontos percentuais maior do que o obtido pelo algoritmo HILS. Por sua vez, as colunas $Dev(t)(\%)$ apresentam os desvios percentuais relativos dos tempos médios obtidos pelas variantes HILS-X em relação aos tempos médios obtidos pelo algoritmo HILS em sua versão completa. Valores positivos de $Dev(t)(\%)$ indicam o percentual de tempo adicional consumido pela variante HILS-X em comparação com a versão completa. Por outro lado, valores negativos indicam o percentual de redução de tempo pela variante HILS-X em relação ao tempo da versão completa. Por exemplo, na instância capa3, a variante HILS-C consumiu 67,54% mais tempo do que sua versão completa, enquanto, na instância i300_16, consumiu 21,09% menos tempo. Os resultados mais significativos estão destacados em negrito.

Os resultados da variante HILS-A mostram que, ao remover o procedimento de abertura, o algoritmo obteve soluções piores, especialmente nas instâncias i1000_9 ($d(\Delta) = +0,19$) e i1000_1 ($d(\Delta) = +0,13$). Esse pior desempenho também é observado quando se remove o procedimento VND, uma vez que $d(\Delta) = +0,17$ na instância i1000_9 e $d(\Delta) = +0,12$ na i1000_1 para a variante HILS-B. Os resultados da HILS-C mostram que, sem a componente de resolução do SSCFLP reduzido pelo *solver*, há piora na qualidade das soluções em todas as instâncias, sendo essa piora mais significativa nas instâncias i300_16 e i500_2, para as quais se tem $d(\Delta) = +0,36$ e $d(\Delta) = +0,33$, respectivamente. Os resultados das variantes HILS-D, HILS-E e

HILS-F mostram que a remoção dos procedimentos ILS-I, resolução do GAP pelo *solver* na fase II e ILS-II piorou a qualidade das soluções na instância i1000_1, com valores de $d(\Delta)$ iguais a +0,15, +0,11 e +0,07, respectivamente. Conclui-se que todas as componentes do algoritmo HILS contribuem positivamente para o desempenho do algoritmo, em termos de qualidade das soluções geradas, em pelo menos uma das instâncias testadas. Além disso, indicam que esse benefício é mais significativo para a componente responsável pela resolução do SSCFLP reduzido pelo *solver* e menos significativo para o procedimento ILS-II.

Quando uma componente é removida do algoritmo HILS, não há tempo gasto com sua execução e, caso ela consuma muito tempo de execução, espera-se uma redução considerável no tempo total de execução do algoritmo. Os resultados indicam que esse fato ocorre principalmente para a componente que resolve o SSCFLP reduzido utilizando o *solver* nas instâncias i300_16 ($\text{Dev}(t) = -21,09\%$), i500_2 ($\text{Dev}(t) = -57,34\%$), i1000_9 ($\text{Dev}(t) = -54,07\%$) e i1000_1 ($\text{Dev}(t) = -30,44\%$), bem como no procedimento de resolução do GAP pelo *solver* na fase II do HILS nas instâncias i1000_9 ($\text{Dev}(t) = -36,86\%$) e i1000_1 ($\text{Dev}(t) = -21,87\%$). Entretanto, ao desconsiderar uma componente, as soluções obtidas em determinada etapa do algoritmo podem ser piores do que aquelas obtidas pela versão completa do HILS na mesma etapa, fazendo com que os procedimentos posteriores consumam mais tempo do que na versão completa do algoritmo, aumentando o tempo total do algoritmo. Os resultados indicam que isso ocorre com a componente VND na instância i500_2 ($\text{Dev}(t) = +50,03\%$), com a componente de resolução do SSCFLP reduzida pelo *solver* nas instâncias capa3 ($\text{Dev}(t) = +67,54\%$) e i300i500_7 ($\text{Dev}(t) = +861,94\%$), e com a componente ILS-I nas instâncias i300_16 ($\text{Dev}(t) = +20,44\%$), i1000_9 ($\text{Dev}(t) = +25,10\%$) e i1000_1 ($\text{Dev}(t) = +23,86\%$). Para as demais componentes, como os valores absolutos de $\text{Dev}(t)(\%)$ são pequenos, torna-se difícil concluir se as variações de tempo decorrem da natureza estocástica do algoritmo ou da remoção da componente. Essa análise indica que o VND pode reduzir consideravelmente o tempo de execução em algumas instâncias, enquanto a resolução do GAP pelo *solver* pode aumentá-lo significativamente em determinados casos. Além disso, os procedimentos ILS-I e de resolução do SSCFLP reduzido pelo *solver* podem tanto aumentar quanto reduzir o tempo total de execução, dependendo das características da instância considerada.

6.2.2.4 Análise do efeito de se utilizar uma versão do CPLEX mais antiga

O uso de *solvers* para a resolução de modelos de programação matemática pode tornar o desempenho do HILS dependente da eficiência desses *solvers*. Retome que

os experimentos com o HILS utilizaram o CPLEX, versão 22.1. Para analisar o efeito do uso de uma versão mais antiga, o HILS foi executado com o CPLEX versão 12.7, mantendo-se os mesmos parâmetros utilizados no HILS com CPLEX 22.1.

A Tabela 6.12 apresenta os resultados comparativos. Nela, a coluna $d(\Delta)$ indica a diferença entre os valores de Δ obtidos pelo HILS com CPLEX 12.7 e os obtidos pelo HILS com CPLEX 22.1. Já a coluna $Dev(t)\%$ apresenta os desvios percentuais relativos entre os valores de t obtidos pelo HILS com CPLEX 12.7 e os obtidos pelo HILS com CPLEX 22.1. O valor médio de $d(\Delta)$ obtido foi de 0,006, indicando que a influência da versão do CPLEX sobre a qualidade das soluções geradas pelo algoritmo HILS foi insignificante. Por outro lado, a média dos valores de $Dev(t)\%$ foi de 41,24%, indicando um aumento significativo no tempo de execução ao se utilizar a versão mais antiga.

Tabela 6.12: Resultado da avaliação do uso de diferentes versões do CPLEX no HILS

	HILS com CPLEX 12.7		HILS com CPLEX 22,1		$d(\Delta)$	$Dev(t)\%$
	Δ	t	Δ	t		
capa3	0,000	206,40	0,000	121,24	0,000	41,26
i300_16	0,000	76,17	0,000	47,04	0,000	38,25
i3001500_7	0,000	182,15	0,000	110,42	0,000	39,38
i500_2	1,208	9402,44	1,222	4186,64	-0,014	55,47
i1000_9	1,058	13562,27	1,019	7831,37	0,040	42,26
i1000_1	1,433	18776,16	1,421	12988,66	0,011	30,82
				Avg.	0,006	41,24

6.3 Experimentos com o algoritmo NCS

Esta seção apresenta os experimentos computacionais realizados com o NCS. Os procedimentos do algoritmo foram implementados em C++. Os testes foram realizados em uma máquina Intel Core i5-11400H de 2.70 GHz, com 16 GB de memória RAM e sistema operacional Ubuntu 24.04 LTS. Nos procedimentos que requerem a solução exata de subproblemas, utilizou-se o solver CPLEX, versão 22.1.1 configurado para resolver os problemas de forma determinística (`ParallelMode` = 1) e utilizando apenas uma *thread* (`Threads` = 1). Uma exceção foi feita para os problemas da mochila com coeficientes inteiros presentes nos procedimentos `LCILiftingSequencial` e `FCGeraçãoColunas`, descritos nas Subseções 5.2.1.1 e 5.2.1.3. Nesses casos, foi utilizado o algoritmo Minknap de [Pisinger \(1997\)](#). O NCS foi testado nos conjuntos TB1-TB5 apresentados na Tabela 6.1.

Esta seção está organizada da seguinte forma. Inicialmente, a Subseção 6.3.1 apresenta os experimentos com a componente PC+PIS do NCS (Algoritmo 9). Em seguida, a Subseção 6.3.2 apresenta os experimentos com o algoritmo NCS configurado com a versão da componente PC+PIS que apresentou desempenho superior.

6.3.1 Experimentos com o procedimento PC+PIS

Inicialmente, foram conduzidos experimentos com o procedimento PC+PIS proposto na Seção 5.2.1 para comparar versões que utilizam diferentes procedimentos de geração de planos de corte. O objetivo é selecionar uma versão a ser utilizada na linha 2 do Algoritmo 11 (NCS).

Primeiramente, a Subseção 6.3.1.1 compara duas versões que utilizam todos os procedimentos descritos nas Subseções 5.2.1.1–5.2.1.3. A primeira, denominada PC+PIS_{xy}, gera planos de corte ao explorar os polítopos da mochila $\mathcal{X}_j^y$ e $\mathcal{Y}$. A segunda, denominada PC+PIS_x, explora apenas $\mathcal{X}_j^y$, desativando a linha 13 do Algoritmo 9. Dessa forma, é possível avaliar se há benefícios na geração de planos de corte a partir do polítopo $\mathcal{Y}$. Observe que esses cortes são obtidos a partir da restrição redundante (5.4) da formulação do SSCFLP apresentada no Capítulo 3. Embora Yang et al. (2012) e Gadegaard et al. (2018) tenham utilizado tais cortes, eles não apresentaram evidências experimentais de suas vantagens quando o método de planos de corte é combinado com a estratégia PIS.

Na Subseção 6.3.1.2, seleciona-se a melhor versão entre PC+PIS_{xy} e PC+PIS_x e analisa-se o efeito de cada procedimento de geração de planos de corte, com o objetivo de selecionar uma família de cortes efetiva para o algoritmo NCS.

Em todos os experimentos com versões do procedimento PC+PIS, adotaram-se os seguintes parâmetros: `TolerCuttingPlane` = 0.0001 como critério de parada do procedimento de planos de corte, `MaxIterFc` = 50 e $M = 10^6$ no procedimento `FCGeraçãoColunas`. Os experimentos foram conduzidos em uma amostra de 50 instâncias dos conjuntos *benchmark* considerados nesta Seção. A amostra foi selecionada da seguinte maneira: para cada subconjunto de TB1-TB5 foram selecionadas aleatoriamente $\max\{1, \lfloor n/5 \rfloor\}$ instâncias, em que n representa o número de instâncias do subconjunto.

As relaxações parciais obtidas pelas versões do PC+PIS foram resolvidas com um limite de tempo de 3600 segundos. Para avaliar a qualidade dos LIs fornecidos por cada versão na amostra, foi calculado, para cada instância, o valor de $\Delta(\text{BKV}, \text{LI})\%$. Em seguida, foi considerada a média desses valores, denotada por $\overline{\Delta}(\text{LI})\%$. Assim, quanto menor o valor de $\overline{\Delta}(\text{LI})\%$, melhor a qualidade dos LIs.

Para avaliar o custo computacional de cada versão, foi utilizada a média geo-

métrica, denotada por $\hat{t}$, dos tempos de execução em cada instância da amostra, incluindo o tempo de resolução das relaxações parciais resultantes. Optou-se pela média geométrica (em vez da média aritmética) por reduzir o impacto de valores de tempo de execução muito discrepantes entre instâncias de diferentes conjuntos *benchmark*.

6.3.1.1 Experimentos com as versões PC+PISxy e PC+PISy

Em ambas as versões, PC+PISx e PC+PISxy, os LCIs são gerados primeiro, seguidos por LPIs e, por fim, por FCs. Cada procedimento é aplicado apenas se o anterior não produzir um plano de corte violado. LCIs são gerados antes de LPIs porque estudos como o de [Kaparis & Letchford \(2010\)](#) indicam que essa ordem é mais eficiente para identificar planos de corte violados em polítopos da mochila. Para gerar CIs, os procedimentos propostos na Subseção 5.2.1.1 são aplicados em ordem não decrescente de complexidade: CIGulosaA, CIGulosaB, CIGulosaNova e CIExato. Conforme discutido na Subseção 5.2.1.1, um CI é sempre selecionado, seja ele violado ou o mais próximo de ser violado, para gerar um LCI. Devido à sua menor complexidade, o procedimento LCIFunçãoSuperaditiva é aplicado antes do LCILiftingSequencial; esse último é utilizado apenas se o primeiro não gerar um plano de corte violado.

Os resultados das versões PC+PISxy e PC+PISx são apresentados na Tabela 6.13. A versão PC+PISx apresenta desempenho superior ao da PC+PISxy tanto em termos de qualidade do limitante inferior (menor $\overline{\Delta}(\text{LI})\%$) quanto de tempo de execução (menor $\hat{t}$). Esses resultados indicam que a geração de planos de corte exclusivamente a partir de $\mathcal{X}_j^y$ é mais eficaz do que a exploração conjunta de $\mathcal{X}_j^y$ e $\mathcal{Y}$ no Algoritmo 9.

Tabela 6.13: Resultados comparativos dos procedimentos PC+PISxy e PC+PISx.

Versão	$\overline{\Delta}(\text{LI})\%$	$\hat{t}$
PC+PISxy	0,1153	5,88
PC+PISx	0,1081	4,22

6.3.1.2 Avaliação do efeito de cada procedimento de geração de planos de corte

Nesta subseção, avalia-se o efeito de cada procedimento de geração de planos de corte na versão PC+PISx sobre a qualidade dos LIs e o tempo computacional. Para isso, são conduzidos experimentos com versões, denotadas por PC+PISx-X, em que $1 \leq X \leq 8$, cada uma obtida pela remoção de um procedimento da versão

PC+PISx. Para cada versão, são calculados os indicadores $r(LI)$ e $r(\hat{t})$, que representam, respectivamente, as razões entre os valores de $\overline{\Delta}(LI)\%$ e de $\hat{t}$ obtidos e os valores correspondentes da versão PC+PISx. Por fim, o efeito de cada procedimento é classificado da seguinte forma: se a qualidade dos LIs da versão PC+PISx-X for pior ($r(LI) > 1$), o procedimento removido tem efeito positivo na qualidade dos LIs; caso contrário, tem efeito negativo. De forma análoga, se o tempo de execução da versão PC+PISx-X for maior ($r(\hat{t}) > 1$), o procedimento removido tem efeito positivo no tempo computacional; caso contrário, tem efeito negativo.

Tabela 6.14: Resultados da avaliação do efeito individual de cada tipo de corte

Versão	Procedimento removido	$\overline{\Delta}(LI)\%$	$\hat{t}$	$r(LI)$	$r(\hat{t})$	Efeito	
						Qualidade dos LIs	Tempo de execução
PC+PISx		0,1081	4,22				
PC+PISx-1	CIGulosaA	0,1180	4,52	1,0913	1,07	Positivo	Positivo
PC+PISx-2	CIGulosaB	0,1132	4,25	1,0472	1,01	Positivo	Positivo
PC+PISx-3	CIGulosaNova	0,1186	4,01	1,0971	0,95	Positivo	Negativo
PC+PISx-4	CIExato	0,1142	3,24	1,0561	0,77	Positivo	Negativo
PC+PISx-5	LCIFunçãoSuperaditiva	0,1116	4,41	1,0320	1,05	Positivo	Positivo
PC+PISx-6	LCILiftingSequencial	0,1235	4,17	1,1427	0,99	Positivo	Negativo
PC+PISx-7	LPIFunçãoSuperaditiva	0,1079	4,17	0,9981	0,98	Negativo	Negativo
PC+PISx-8	FCGeraçãoColunas	0,1672	2,04	1,5467	0,48	Positivo	Negativo

A Tabela 6.14 apresenta os resultados dessa análise. Observa-se que CIGulosaA, CIGulosaB e LCIFunçãoSuperaditiva mostram-se altamente benéficos, pois apresentam efeito positivo tanto na qualidade dos LIs quanto no tempo de execução. Os procedimentos CIGulosaNova, CIExato, LCILiftingSequencial e FCGeraçãoColunas apresentam efeito positivo na qualidade dos LIs, mas efeito negativo no tempo de execução. Além disso, LPIFunçãoSuperaditiva é o único procedimento com efeito negativo sobre a qualidade dos LIs; nesse procedimento, o efeito também foi negativo no tempo de execução.

Para aplicação no NCS, adota-se o critério de que os procedimentos que melhoram a qualidade dos LIs devem ser selecionados, mesmo que aumentem o tempo computacional. Portanto, a versão PC+PISx-7 é selecionada para ser utilizada na linha 2 do Algoritmo 11 (NCS).

Ressalta-se que as conclusões obtidas a partir da análise do efeito individual de cada procedimento de geração de planos de corte podem ser aplicadas em outros métodos de solução, inclusive no contexto de outros problemas combinatórios.

6.3.2 Experimentos com o algoritmo NCS completo

Nos experimentos com o algoritmo NCS (Algoritmo 11), os critérios de parada foram definidos da seguinte forma: $NCSGapToler = 0.01$ e $TLimNCS = 28800$ segundos

(8 horas). Utilizou-se a versão PC+PISx-7 do procedimento PC+PIS na linha 2, adotando as mesmas configurações empregadas na Subseção 6.3.1. Um limite de tempo de 3600 segundos (1 hora) para a resolução da relaxação parcial obtida com o procedimento PC+PISx-7 foi imposto. Essa configuração evita que todo o tempo de execução do algoritmo NCS seja consumido pela resolução da relaxação parcial obtida via PC+PISx-7.

Para resolver os problemas esparsos nos conjuntos de dados TB3 e TB5, configurou-se `NodeFileInd` = 3 no CPLEX 22.1.1. Essa configuração armazena a árvore de busca em um arquivo compactado no Como os custos de alocação dos clientes e os custos fixos de abertura são inteiros nos conjuntos TB1, TB2 e TB3, definiu-se, para esses conjuntos, a tolerância relativa de gap como 0 (`MIPGap` = 0) e a tolerância absoluta como 0,99 (`AbsMIPGap` = 0,99) nos modelos dos problemas esparsos. Para os conjuntos TB1, TB2 e TB3, definiu-se para esses conjuntos a tolerância relativa de gap como 0 (`MIPGap` = 0) e a tolerância absoluta como 0,99 (`AbsMIPGap` = 0,99) nos modelos dos problemas esparsos. Além disso, nesses conjuntos, utilizou-se o critério de otimalidade $LS - LI < 1$, em vez de $\Delta(LS, LI)\% < \text{NCSGapToler}$, de modo a garantir a obtenção da solução ótima sem considerar uma margem de tolerância. Para os problemas densos, definiu-se a tolerância de gap igual a 0 (`MIPGap` = 0) em todos os conjuntos de dados.

O parâmetro U do Algoritmo 11 e os parâmetros $Q_{\min}$, R e TimLimHeur do Algoritmo 12 foram calibrados utilizando o pacote IRACE (López-Ibáñez et al., 2016). Para o treinamento, foram selecionadas as seguintes instâncias: p15 e p63 do conjunto TB1; y60-200-4 do conjunto TB2; g50-200-2-3 do conjunto TB3; capb2 do conjunto TB4; e i300_8 e i1000_20 do conjunto TB5. Foi definida a expressão $LS_0 + 10^{-6} \times t(LS_0)$ como função de avaliação no IRACE, em que LS_0 é o valor da função objetivo da primeira solução viável encontrada pelo algoritmo NCS, e $t(LS_0)$ é o tempo de execução, em segundos, necessário para obter essa solução. O segundo termo representa uma penalização mínima associada ao tempo de execução. Essa penalização é utilizada para desempatar soluções com o mesmo valor de função objetivo, favorecendo menores tempos.

A Tabela 6.15 reporta o conjunto de parâmetros fornecidos ao IRACE e os retornados no resultado da calibração.

Utilizando a configuração descrita nesta subseção, o algoritmo NCS foi executado em todas as instâncias dos conjuntos TB1 a TB5 da Tabela 6.1. Os resultados detalhados dos experimentos estão disponíveis na Seção A.3 no Apêndice A.

Tabela 6.15: Resultado da calibração de parâmetros do NCS

Parâmetro	Valores fornecidos	Valor na melhor configuração
U	{6, 8, 10, 12}	10
Q_{min}	{5, 6, 7, 8, 9}	6
R	{1, 2, 3, 4, 5}	4
TimLimHeur	{1800, 3600, 4800}	3600

Tabela 6.16: Resumo dos resultados obtidos pelo algoritmo NCS.

Conjunto	Inst	Opt	$\bar{t}$	Novos Opt	Opt Conhecidos Não Encontrados	Soluções Factíveis	Novos BKVs
TB1	71	71	0,86	0	0	71	0
TB2	20	20	28,11	0	0	20	0
TB3	45	44	1102,55	6	0	45	1
TB4	12	12	132,01	0	0	12	0
TB5	100	58	13283,72	2	0	100	26
Total	248	205		8	0	248	27

Tabela 6.17: Novos valores ótimos encontrados pelo algoritmo NCS.

Instância	Valor ótimo	t	Instância	Valor ótimo	t
n19 (TB3)	25953	24,58	n33 (TB3)	34878	11,32
n20 (TB3)	25328	331,32	n34 (TB3)	34820	366,83
n31 (TB3)	34860	169,34	i700_13 (TB5)	17284,72	17029,47
n32 (TB3)	36549	18982,61	i1000_18 (TB5)	20752,92	9346,50

6.3.2.1 Análise dos resultados

A Tabela 6.16 apresenta um resumo dos principais resultados obtidos com o NCS. Nas colunas 3 e 4 são apresentados o número de soluções ótimas obtidas (Opt) e o tempo médio de execução ($\bar{t}$). As demais colunas reportam o número de novos valores ótimos encontrados (Novos Opt), o número de valores ótimos conhecidos não encontrados pelo NCS (Opt Conhecidos Não Encontrados), o número de instâncias para as quais o algoritmo retornou uma solução viável (Soluções Factíveis) e o número de novos melhores valores conhecidos (Novos BKVs).

O algoritmo NCS encontrou soluções ótimas para todas as instâncias dos conjuntos TB1, TB2 e TB4; para 44 das 45 instâncias de TB3; e para 58 das 100 instâncias de TB5. No total, foram obtidos 205 valores ótimos dentre 248 instâncias, incluindo 8 novos valores ótimos, apresentados na Tabela 6.17. Como o número total de “Opt Conhecidos Não Encontrados” é igual a zero, os valores ótimos encontrados pelo NCS incluem todos os previamente reportados na literatura. Em relação ao tempo de execução, observa-se uma variação significativa entre os conjuntos de dados ($\bar{t} < 1$ para TB1 e $\bar{t} > 13000$ para TB5).

Adicionalmente, o NCS atualizou 27 melhores valores conhecidos (BKVs), cor-

respondendo a 63% das instâncias sem soluções ótimas conhecidas (43 instâncias). Os novos BKVs são apresentados na Tabela 6.18. As melhorias obtidas são relativamente pequenas, variando de 0,004% a 0,22%. Do ponto de vista prático, tais melhorias podem ser pouco relevantes em aplicações reais, especialmente diante da incerteza inerente aos dados de entrada. Por outro lado, dada a maturidade da literatura sobre o SSCFLP e a proximidade entre os valores conhecidos, mesmo pequenas melhorias são difíceis de alcançar, o que indica que a abordagem proposta é capaz de explorar o espaço de soluções de forma mais eficaz. Assim, sob a perspectiva algorítmica, essas melhorias representam uma contribuição relevante para o avanço dos métodos de solução para o SSCFLP.

Tabela 6.18: Novos BKVs encontrados pelo algoritmo NCS

Inst	BKV atual	Novo BKV	Melhoria%
TB3			
n35	34142,00	34140,00	0,006
TB5			
i300_3	15666,23	15649,41	0,107
i500_1	26731,63	26698,33	0,125
i500_2	28460,43	28431,44	0,102
i500_3	28275,24	28212,43	0,222
i500_4	28489,69	28450,80	0,137
i500_5	24969,40	24920,99	0,194
i500_9	16418,30	16399,40	0,115
i700_1	37343,34	37302,35	0,110
i700_2	34817,44	34783,64	0,097
i700_3	34759,07	34739,12	0,057
i700_5	38230,13	38158,53	0,187
i700_9	21102,39	21090,04	0,059
i700_10	22210,50	22188,50	0,099
i700_11	17188,47	17180,27	0,048
i700_12	18201,06	18194,03	0,039
i700_15	18220,75	18218,21	0,014
i1000_1	50104,98	50049,24	0,111
i1000_3	47737,73	47701,98	0,075
i1000_5	51415,41	51408,93	0,013
i1000_10	27388,50	27364,31	0,088
i1000_12	22231,18	22226,96	0,019
i1000_13	22745,61	22728,86	0,074
i1000_14	22405,92	22401,92	0,018
i1000_15	22704,67	22687,95	0,074
i1000_16	21390,24	21386,85	0,016
i1000_17	21234,62	21233,77	0,004
i1000_19	20597,01	20591,21	0,028
i1000_20	21601,55	21598,54	0,014

O algoritmo NCS não encontrou soluções ótimas em 43 das 248 instâncias. Todos os valores na coluna “Opt Conhecidos Não Encontrados” são iguais a zero, indicando que nenhuma dessas instâncias possui solução ótima conhecida reportada na literatura.

Tabela 6.19: Resultados comparativos entre NCS-2 e NCS

Conjunto	Inst	Opt	Dif Opt	$r(\bar{t})$	Soluções Viáveis
TB1	71	71	0	0,88	71
TB2	20	20	0	1,10	20
TB3	45	40	-4	4,26	40
TB4	12	12	0	1,11	12
TB5	100	58	0	1,14	58
Total	248	201	-4		201

tura. Os valores na coluna “Soluções Factíveis” indicam que, para todas as instâncias de todos os conjuntos de dados, o NCS retorna uma solução ótima ou, ao menos, uma solução viável.

Além disso, em 30 das 43 instâncias, o algoritmo atingiu o limite de tempo ao resolver o primeiro problema esparso. Portanto, mesmo com as heurísticas propostas, a resolução dos PEs pode representar um desafio para o algoritmo NCS. Esse resultado está em consonância com a conclusão de [Gadegaard et al. \(2018\)](#), que identificaram a resolução dos PEs como o principal gargalo de seu algoritmo.

6.3.2.2 Análise do procedimento heurístico proposto

Também foi avaliado o efeito da heurística desenvolvida neste trabalho sobre o algoritmo NCS. Para isso, considerou-se uma variante denominada NCS-2, que não aplica a heurística. Executou-se o NCS-2 em todas as instâncias dos conjuntos *benchmark* considerados nos experimentos com o NCS, e compararam-se os resultados obtidos com os do NCS. A Tabela 6.19 apresenta os resultados obtidos. A coluna “Dif Opt” reporta a diferença entre o número de soluções ótimas encontradas pelo NCS-2 e pelo NCS, enquanto a coluna $r(\bar{t})$ reporta a razão entre seus tempos médios de execução.

Os resultados mostram que, para os conjuntos TB2, TB4 e TB5, o NCS-2 apresenta desempenho inferior ao do NCS, pois encontrou o mesmo número de soluções ótimas, porém com tempo médio de execução maior ($r(\bar{t}) > 1$). Para o conjunto TB3, o NCS-2 apresenta desempenho significativamente pior: encontrou quatro soluções ótimas a menos (Dif Opt = -4), com tempo médio de execução mais de quatro vezes maior do que o do NCS ($r(\bar{t}) = 4,26$). Observa-se ainda que os valores nas colunas “Soluções Viáveis” e “Opt” para o NCS-2 são iguais. Esse resultado indica que, enquanto o NCS encontrou soluções viáveis para todas as instâncias, o NCS-2 encontrou soluções viáveis apenas quando atingiu a solução ótima.

Conclui-se que a heurística proposta melhora o desempenho do algoritmo NCS em três aspectos principais: reduz o tempo de execução; aumenta a probabilidade de encontrar soluções ótimas; e permite que o NCS retorne soluções viáveis mesmo

quando soluções ótimas não são encontradas.

6.3.2.3 Análise do crescimento da árvore de busca

Considerando as 205 instâncias para as quais o algoritmo NCS encontrou soluções ótimas, analisa-se o número total de iterações (TotalIter) e a iteração em que o valor de LS do Algoritmo 11 coincide, pela primeira vez, com o valor ótimo (IterOpt). Para isso, foram construídos gráficos das frequências relativas acumuladas (FRAs) de TotalIter e IterOpt. Nesse último, consideraram-se apenas as instâncias para as quais TotalIter > 0. Esses gráficos são apresentados nas Figuras 6.1a e 6.1b.

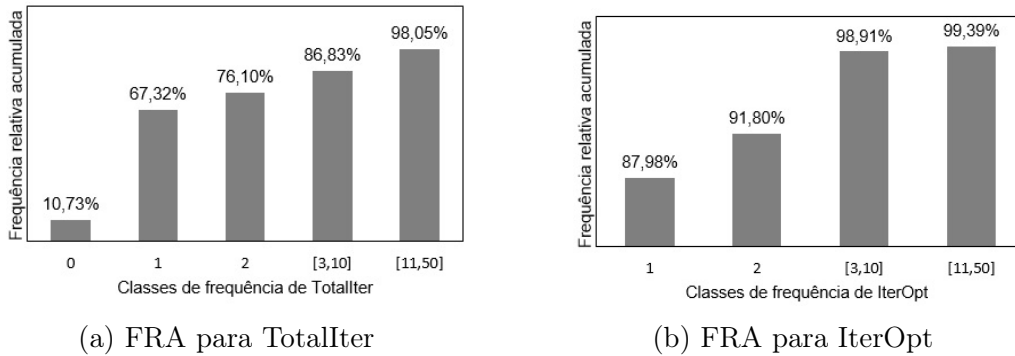


Figura 6.1: FRAs para os valores de TotalIter e IterOpt.

A Figura 6.1a mostra que, para aproximadamente 11% das instâncias, a solução ótima foi encontrada ao resolver o modelo RDP_0 . Além disso, o algoritmo foi finalizado antes do segundo nível da árvore em aproximadamente 67% das instâncias e antes do terceiro nível em aproximadamente 76% das instâncias. Portanto, na maioria das instâncias, foram necessárias apenas poucas iterações para resolver o SSCFLP com o algoritmo NCS.

Adicionalmente, desconsiderando os casos em que TotalIter = 0, a Figura 6.1b mostra que a solução ótima foi obtida na primeira iteração para 88% das instâncias e até a décima iteração para aproximadamente 99% delas. Esse resultado sugere que o primeiro conjunto de localizações promissoras já fornece o conjunto de facilidades abertas na solução ótima, ou que são necessárias apenas poucas modificações para alcançá-la.

6.3.2.4 Comparação com o algoritmo exato estado da arte e o solver CPLEX

Esta seção compara o desempenho do algoritmo NCS com o do algoritmo TP-ALG e com o solver CPLEX nos conjuntos *becnhmark* de TB1 a TB5 da Tabela 6.1. Para o experimento com o TP-ALG, utilizou-se o código disponibilizado pelos autores em

<http://github.com/SuneGadegaard/SSCFLPsolver>, configurando o CPLEX como solver. Para o experimento com o CPLEX, este foi configurado para operar sob as mesmas condições adotadas no experimento com o NCS. Para ambos os algoritmos foi imposto um limite de tempo de 28800 segundos.

A Tabela 6.20 apresenta os resultados da comparação. Eles são reportados em 26 subconjuntos dos conjuntos de dados considerados. Para cada conjunto, foram considerados os mesmos subconjuntos definidos nos estudos que os introduziram na literatura. A análise considera que o NCS é mais eficiente em um subconjunto quando o outro algoritmo encontra um número de soluções ótimas menor ou igual ao do NCS ($\text{Dif Opt} \leq 0$) e apresenta tempo de execução maior ($r(\bar{t}) > 1$). Os valores em negrito na tabela indicam tais casos. Além disso, são considerados o número de instâncias em que a solução obtida pelo CPLEX ou TP-ALG (desconsiderando a prova de otimalidade) é melhor que a do NCS (coluna B) e o número de instâncias em que é pior (coluna W).

De acordo com a Tabela 6.20, o algoritmo NCS obteve desempenho superior ao do CPLEX em 19 dos 26 subconjuntos. Dentre esses, em quatro subconjuntos encontrou mais soluções ótimas com menor tempo computacional, enquanto nos demais obteve o mesmo número de soluções ótimas com menor tempo. Em nenhum subconjunto o CPLEX encontrou mais soluções ótimas do que o NCS, ou seja, $\text{Dif Opt} \leq 0$ em todos os 26 subconjuntos. Destaca-se, ainda, que o CPLEX consumiu mais de 10 vezes o tempo do NCS ($r(\bar{t}) > 10$) nos subconjuntos y16-y20, n1-n5, n6-n10, n16-n20, n21-n25 e n36-n40, e mais de 100 vezes nos subconjuntos n6-n10 e n16-n20. Esses resultados indicam que o NCS é mais eficiente do que o CPLEX. Ademais, considerando todas as instâncias, o NCS apresentou desempenho superior, identificando seis valores ótimos a mais. Nos casos em que a otimalidade não foi comprovada, os valores de B e W obtidos para o CPLEX ($B = 8$ contra $W = 29$) indicam que o NCS produz soluções heurísticas de melhor qualidade.

O desempenho superior do NCS em relação ao CPLEX pode ser atribuído à estratégia *cut-and-solve* que o fundamenta. Esse esquema permite direcionar a busca por meio da fixação de múltiplas variáveis, conduzindo o algoritmo a nós da árvore de busca que levam à solução ótima do SSCFLP. Como consequência, são necessárias poucas iterações para encontrar soluções ótimas, conforme mostrado na Subseção 6.3.2.3. O uso de uma relaxação parcial forte e da heurística proposta contribui adicionalmente para a eficiência do método.

Em comparação com o algoritmo TP-ALG, o NCS foi mais eficiente em 22 dos 26 subconjuntos. Em dois desses subconjuntos, encontrou mais soluções ótimas em menos tempo, enquanto nos demais encontrou o mesmo número de soluções ótimas em menos tempo. Em nenhum subconjunto, o TP-ALG encontrou mais soluções ótimas

Tabela 6.20: Resultados comparativos entre NCS, TP-ALG e o CPLEX

Subset	Inst	NCS		CPLEX solver				TP-ALG			
		Opt	$\bar{t}$	Dif Opt	B	W	$r(\bar{t})$	Dif Opt	B	W	$r(\bar{t})$
TB1											
h1-h12	12	12	0,03	0	0	0	0,68	0	0	0	2,93
h13-h24	12	12	0,04	0	0	0	1,28	0	0	0	4,34
h25-h40	16	16	0,68	0	0	0	1,25	0	0	0	2,41
h41-h55	15	15	0,13	0	0	0	0,95	0	0	0	2,52
h56-h71	16	16	2,97	0	0	0	0,36	0	0	0	0,72
TB2											
y1-y5	5	5	13,01	0	0	0	3,24	0	0	0	1,53
y6-y10	5	5	40,93	0	0	0	7,72	0	0	0	0,96
y11-y15	5	5	21,09	0	0	0	7,33	0	0	0	1,40
y16-y20	5	5	39,59	0	0	0	35,10	0	0	0	1,12
TB3											
n1-n5	5	5	108,33	0	0	0	62,99	0	0	0	5,55
n6-n10	5	5	23,25	0	0	0	191,37	0	0	0	2,41
n11-n15	5	5	1,70	0	0	0	3,38	0	0	0	5,61
n16-n20	5	5	92,37	-1	0	1	143,45	-1	0	1	66,61
n21-n25	5	5	18,09	0	0	0	25,40	0	0	0	18,14
n26-n30	5	5	0,92	0	0	0	3,62	0	0	0	5,21
n31-n35	5	4	9666,02	-2	0	1	2,27	-3	0	4	2,52
n36-n40	5	5	6,71	0	0	0	46,90	0	0	0	6,44
n41-n45	5	5	5,60	0	0	0	2,68	0	0	0	3,67
TB4											
capA	4	4	197,44	0	0	0	0,54	0	0	0	0,70
capB	4	4	129,93	0	0	0	2,46	0	0	0	3,38
capC	4	4	68,65	0	0	0	1,15	0	0	0	13,65
TB5											
i300	20	16	6831,15	0	2	2	0,88	0	0	4	0,98
i300i500	20	20	170,26	0	0	0	0,19	0	0	0	1,87
i500	20	13	11933,84	-2	0	5	1,28	0	0	7	1,02
i700	20	7	20867,48	0	2	8	0,95	0	0	12	1,04
i1000	20	2	26644,81	-1	4	12	1,06	0	0	18	1,02
Total	248	205	-	-6	8	29	-	-4	0	46	-

do que o NCS. Destaca-se, ainda, que o TP-ALG demandou mais de dez vezes o tempo do NCS ($r(\bar{t}) > 10$) nos subconjuntos n16–n20, n21–n25 e capC. Além disso, o NCS encontrou quatro soluções ótimas adicionais, considerando todas as instâncias dos experimentos (Dif Opt = -4). Esses resultados evidenciam a superioridade do NCS em relação ao TP-ALG ao encontrar mais soluções ótimas ou a mesma quantidade de soluções ótimas mais rapidamente. A comparação entre B e W para o TP-ALG (B = 0 versus W = 46) indica que, quando o NCS e o TP-ALG não produzem a mesma solução, o NCS produz consistentemente soluções de melhor qualidade. No conjunto TB5, considerando as 42 instâncias para as quais nenhum dos algoritmos encontrou a solução ótima, o TP-ALG obteve soluções piores em 41 delas (97,6%), e o mesmo valor em apenas uma. Esse resultado evidencia a superioridade do NCS em termos

Tabela 6.21: Resultados comparativos de LI_0 e LS_0 fornecidos pelos algoritmos NCS e TP-ALG

Conjunto	$\overline{\Delta}(LI_0)\%$		Opt(LI_0)		$\bar{t}(LI_0)$	
	NCS	TP-ALG	NCS	TP-ALG	NCS	TP-ALG
TB1	0,062	0,262	37	36	0,22	0,15
TB2	0,086	0,624	0	0	6,79	3,96
TB3	0,014	0,189	0	0	0,85	0,96
TB4	0,001	0,050	1	1	55,66	45,65
TB5	0,208	0,501	4	1	854,14	669,38
Conjunto	$\overline{\Delta}(LS_0)\%$		Opt(LS_0)		$\bar{t}(LS_0)$	
	NCS	TP-ALG	NCS	TP-ALG	NCS	TP-ALG
TB1	0,009	0,000	28	35	0,38	0,73
TB2	0,000	0,920	19	3	17,86	23,29
TB3	0,000	0,616	42	16	29,64	21,70
TB4	0,001	0,001	11	2	55,79	17,67
TB5	0,209	1146,633	31	12	1796,73	27,59

da qualidade das soluções heurísticas em conjuntos mais desafiadores.

Para analisar mais profundamente essa comparação, recorda-se que o NCS, utilizando a versão PC+PISx-7, difere do TP-ALG principalmente nos procedimentos de geração de planos de corte e na obtenção do primeiro limite superior. Para relacionar essas diferenças aos resultados, o primeiro limitante inferior (LI_0) obtido a partir da relaxação parcial e o valor da primeira solução viável (LS_0) foram considerados para ambos os algoritmos, bem como os tempos necessários para obtê-los, denotados por $t(LI_0)$ e $t(LS_0)$. Em seguida, para cada conjunto de dados, calcularam-se as médias de $\Delta(BKV, LI_0)\%$, $\Delta(LS_0, MLI)\%$, $t(LI_0)$ e $t(LS_0)$, em que MLI é o melhor limitante inferior conhecido. Essas médias foram denotadas por $\overline{\Delta}(LI_0)\%$, $\overline{\Delta}(LS_0)\%$, $\bar{t}(LI_0)$ e $\bar{t}(LS_0)$, respectivamente. Além disso, apurou-se o número de instâncias em que LI_0 e LS_0 coincidiram com o valor ótimo, denotados por Opt(LI_0) e Opt(LS_0), respectivamente.

Os resultados são apresentados na Tabela 6.21. Os valores na coluna $\overline{\Delta}(LI_0)\%$ são menores para o NCS em todos os conjuntos, e os valores de Opt(LI_0) são maiores em dois conjuntos (TB1 e TB5) e iguais nos demais. O tempo médio para a obtenção de LI_0 é maior para o NCS na maioria dos conjuntos. Entretanto, obter LI_0 de alta qualidade apresenta vantagens importantes: além de fornecer um bom limitante inferior, essa solução é utilizada para selecionar o conjunto inicial de locais promissores. Assim, considera-se que o NCS apresenta vantagem nesse aspecto.

Em relação à coluna $\overline{\Delta}(LS_0)\%$, o NCS apresenta valores menores nos conjuntos TB2, TB3, TB4 e TB5 e valores maiores apenas no conjunto TB1. Os valores na coluna Opt(LS_0) indicam que, nesses quatro conjuntos, o número de instâncias em que LS_0 coincide com o valor ótimo é consideravelmente maior para o NCS. Quanto

ao tempo necessário para obter LS_0 , os valores na coluna $\bar{t}(LS_0)$ são menores para o NCS em TB1 e TB2 e maiores nos demais conjuntos. Esses resultados indicam que o NCS tende a produzir valores de LS_0 de maior qualidade, que coincidem com os valores ótimos com maior frequência, embora possa demandar mais tempo computacional para obtê-los. Isso confere ao NCS vantagem na redução do espaço de busca dos problemas esparsos, desde a primeira iteração. Esse aspecto é particularmente importante, uma vez que a resolução dos PEs constitui o principal desafio computacional em ambos os algoritmos. O desempenho superior do NCS em relação ao TP-ALG sugere que o tempo adicional necessário para obter LS_0 é compensado pelos benefícios associados à sua maior qualidade.

A análise desses resultados indica que a obtenção de LI_0 e LS_0 de melhor qualidade pelo NCS desempenha um papel-chave na superioridade do NCS em relação ao TP-ALG.

6.4 Experimentos com o algoritmo B&B-GVNS

Instâncias para o RSSCFLP foram derivadas de conjuntos *benchmark* TB1, TB2, TB3 e TB6 da Tabela 6.1. Para isso, valores nominais foram substituídos por intervalos de incerteza, considerando um parâmetro δ , que determina a amplitude desses intervalos da seguinte forma. Seja v o valor nominal de um parâmetro do SSCFLP. Se v é o valor de uma demanda ou de uma capacidade, o intervalo de incerteza correspondente é $[v - \delta v, v + \delta v]$. Se v é o valor de um custo de alocação ou de um custo de abertura, então o intervalo de incerteza correspondente é $[v, v + \delta v]$.

Nos experimentos com o B&B-GVNS, adotou-se $\delta = 20\%$ na geração das instâncias. Além disso, assumiu-se que cada nível de robustez do RSSCFLP fosse proporcional ao número total (ou estimado) de parâmetros sujeitos à incerteza correspondente. Dada uma solução do RSSCFLP, a quantidade total de custos de alocação incertos é igual ao número de clientes. Já cada facilidade possui uma única capacidade sujeita à incerteza. A quantidade de custos de abertura foi estimada pelo número Q de facilidades abertas na solução da relaxação linear do problema nominal. Por fim, a quantidade de clientes alocados em cada facilidade foi estimada por $P_j = \text{rnd}(b_j/\bar{d})$, em que $\bar{d}$ representa a demanda média dos clientes e $\text{rnd}(\cdot)$ denota o operador de arredondamento para o inteiro mais próximo. Introduzindo os parâmetros κ^c , κ^f , κ^d e κ^b como constantes de proporcionalidade, as seguintes expressões para os níveis de robustez do modelo RSSCFLP foram obtidas:

$$\Gamma^f = \text{rnd}(\kappa^f \times Q); \quad \Gamma_j^d = \kappa^d \times P_j, \forall j \in J; \quad \Gamma_j^b = \kappa^b, \forall j \in J; \quad \Gamma^c = \text{rnd}(\kappa^c \times |I|). \quad (6.2)$$

Para determinar as constantes de proporcionalidade da expressão (6.2), realizou-

se um experimento utilizando uma amostra $\mathcal{S}$, na qual os parâmetros foram ajustados iterativamente com base nas taxas de robustez obtidas. As taxas foram apuradas por meio de simulações de Monte Carlo, nas quais se assumiu que os parâmetros incertos seguem distribuições normais com média igual ao respectivo valor nominal e desvio padrão igual a $0,1v$, em que v representa o valor nominal do parâmetro. A amostra $\mathcal{S}$ foi formada selecionando-se aleatoriamente, em cada um dos conjuntos TB1, TB2, TB3 e TB6 da Tabela 6.1, uma quantidade de instâncias igual ao quociente inteiro entre o número total de instâncias do conjunto e 10.

Dada uma solução $s \in \mathcal{S}$, define-se o custo total de alocação garantido por s , $\Lambda^c(s)$, e o custo total de abertura garantido por s , $\Lambda^f(s)$, pelas expressões:

$$\Lambda^c(s) = \lambda^c \Gamma^c + \sum_{i \in I} \sum_{j \in J} (c_{ij} x_{ij} + \mu_{ij}^c), \quad \Lambda^f(s) = \lambda^f \Gamma^f + \sum_{j \in J} (f_j y_j + \mu_j^f). \quad (6.3)$$

Na sentença (6.3), x_{ij} , μ_{ij}^c , y_j , μ_j^f , λ^c e λ^f correspondem aos valores das respectivas variáveis na solução s . O valor de $\Lambda^c(s)$ representa o custo total de alocação no pior caso para o nível de robustez Γ^c associado à solução s . De forma análoga, $\Lambda^f(s)$ representa o custo total de abertura no pior caso para o nível de robustez Γ^f associado à solução s .

Considerando essas definições as taxas consideradas foram as seguintes:

- (i) Taxa de Viabilidade (TV): média dos percentuais, calculados para cada $s \in \mathcal{S}$, de cenários simulados nos quais cada s é viável;
- (ii) Taxa de Robustez do Custo de Alocação Garantido (TRCALG): média dos percentuais, calculados para cada $s \in \mathcal{S}$, de cenários simulados nos quais o valor de $\Lambda^c(s)$ é maior ou igual ao custo total de alocação de s ;
- (iii) Taxa de Robustez do Custo de Abertura Garantido (TRCABG): média dos percentuais, calculados para cada $s \in \mathcal{S}$, de cenários simulados nos quais $\Lambda^f(s)$ é maior ou igual ao custo total de abertura observado no cenário simulado.

A regra de ajuste adotada no experimento identifica a menor dentre as taxas para determinar qual parâmetro deve ser incrementado em 0,1 da seguinte maneira: o parâmetro κ^c é incrementado quando a menor taxa é TRCALG, enquanto κ^f é incrementado quando a menor taxa é TRCABG. Caso a menor taxa seja TV, incrementa-se κ^d ou κ^b . Nesse caso, ambas as possibilidades são testadas, selecionando-se a que produz o maior valor de TV. A regra de ajuste foi aplicada no experimento até que todas as três taxas fossem superiores a 90%. O aumento de κ^d ou de κ^b podem tornar algumas instâncias de $\mathcal{S}$ ineficazes. Por isso, o seguinte mecanismo de prevenção foi

adicionado ao experimento: caso o aumento de um desses parâmetros torne alguma instância de $\mathcal{S}$ inactível, a atualização desse parâmetro no experimento é bloqueada.

Os resultados obtidos nesse experimento foram: $\kappa^c = 0,1$, $\kappa^f = 0,3$, $\kappa^d = 0,6$ e $\kappa^b = 0,3$. Esses valores foram adotados em todos os experimentos computacionais com o B&B-GVNS.

O CPLEX 22.1.1 foi utilizado como algoritmo baseado em B&B na linha 4 do Algoritmo 14. A solução fornecida na Fase II foi utilizada como solução inicial para o CPLEX na Fase III do algoritmo. Além disso, configuraram-se $\alpha = 50$ nos GVNS-I e GVNS-II e $\omega = 100$ no procedimento da linha 5 do Algoritmo 14 (B&B-GVNS). O CPLEX, com suas configurações padrão, também foi executado no mesmo conjunto de instâncias utilizado no experimento com o B&B-GVNS. Para ambos os algoritmos, foi definido um tempo limite de 7200 segundos. Para cada um deles, apuraram-se o LS, o LI e o tempo de execução t . Os resultados detalhados dos experimentos com ambos os algoritmos constam na Seção A.3 do Apêndice A.

6.4.1 Resultados e análises

O desempenho do B&B-GVNS foi comparado ao do CPLEX. Para cada instância e algoritmo, verificou-se inicialmente se uma solução factível foi encontrada. Em caso afirmativo, apuraram-se os valores de LS e LI retornados, o valor de $\text{gap}\% = 100 \times (\text{LS} - \text{LI}) / \text{LS}$ e se a solução ótima foi encontrada. Reporta-se que o B&B-GVNS encontrou soluções factíveis em todas as 193 instâncias do experimento, enquanto o CPLEX não encontrou soluções factíveis em duas delas, a saber, d30 e d33 de TB6. Isso indica que o B&B-GVNS apresenta vantagem em relação à capacidade de retornar soluções factíveis em instâncias cuja viabilidade é desafiadora. Quanto à quantidade de soluções ótimas encontradas, reporta-se que os algoritmos apresentaram o mesmo desempenho, pois ambos as encontraram nas mesmas instâncias.

O restante da análise consistiu em avaliar, em cada subconjunto, três critérios de comparação entre os algoritmos: (i) menor média de $\text{gap}\%$; (ii) maior quantidade de LS de melhor qualidade na comparação instância por instância; e (iii) menor média de t . Para auxiliar a análise, calcularam-se as variações percentuais ($\text{dev}\%$) entre as médias de $\text{gap}\%$ e de t obtidas pelo B&B-GVNS e os valores correspondentes do CPLEX. Considerou-se que um algoritmo apresentou desempenho superior ao outro quando obteve vantagem em pelo menos um dos três critérios, sem apresentar desvantagem nos demais. As médias dos valores de $\text{gap}\%$ no subconjunto d26-d33 bem como em todo o conjunto TB6 (d1-d57) foram calculadas para ambos os algoritmos sem considerar as instâncias d30 e d33, visto que o CPLEX não encontrou soluções viáveis para elas. Os resultados são apresentados na Tabela 6.22.

Tabela 6.22: Resultados comparativos do B&B-GVNS e do CPLEX

Conjunto	Inst	Opt	CPLEX		B&B-GVNS				Desempenho
			gap%	t	gap% (dev%)	LS melhor	LS pior	t (dev%)	
TB1									
h1-h12	12	12	0,00	0,92	0,00	0	0	-31,43	Superior
h13-h24	12	12	0,00	2,94	0,00	0	0	-16,77	Superior
h25-h40	16	16	0,23	1386,07	0,00	0	0	0,54	Inferior
h41-h55	15	15	0,00	3,31	0,00	0	0	-24,22	Superior
h56-h71	16	13	0,24	1669,82	-2,50	2	1	-9,08	Superior
h1-h71	71	68	0,11	690,00	-1,27	2	1	-4,75	Superior
TB2									
y1-y5	5	0	4,14	7200,00	-19,47	4	1	0,00	Superior
y6-y10	5	0	3,34	7200,00	-32,49	5	0	0,00	Superior
y11-y15	5	0	2,26	7200,00	-18,64	4	1	0,00	Superior
y16-y20	5	0	1,97	7200,00	-33,44	4	1	0,00	Superior
y1-y20	20	0	2,93	7200,00	-25,38	17	3	0,00	Superior
TB3									
g1-g5	5	0	16,88	7200,00	-15,03	5	0	0,00	Superior
g6-g10	5	0	14,72	7200,00	-15,23	5	0	0,00	Superior
g11-g15	5	0	11,73	7200,00	-6,18	5	0	0,00	Superior
g16-g20	5	0	17,34	7200,00	-18,68	5	0	0,00	Superior
g21-g25	5	0	15,45	7200,00	-19,48	5	0	0,00	Superior
g26-g30	5	0	14,17	7200,00	-30,28	4	0	0,00	Superior
g31-g35	5	0	17,93	7200,00	-24,36	5	0	0,00	Superior
g36-g40	5	0	16,43	7200,00	-26,72	5	0	0,00	Superior
g41-g45	5	0	15,39	7200,00	-23,08	5	0	0,00	Superior
g1-g45	45	0	15,56	7200,00	-20,25	44	0	0,00	Superior
TB6									
d1-d6	6	6	0,00	0,77	0,00	0	0	-39,83	Superior
d7-d17	11	11	0,00	1,30	0,00	0	0	15,87	Inferior
d18-d25	8	8	0,00	506,94	0,00	0	0	-45,92	Superior
d26-d33	6	6	0,00	4,79	0,00	2	0	-20,38	Superior
d34-d41	8	5	2,50	2984,40	5,78	1	1	-4,37	-
d42-d49	8	2	3,21	5401,69	-10,23	3	2	-0,01	Superior
d50-d57	8	2	4,00	5400,69	-8,89	4	1	0,00	Superior
d1-d57	55	40	1,41	2079,95	-5,56	10	4	-2,55	Superior

A coluna “Opt” indica a quantidade de ótimos encontrados por ambos os algoritmos. Nos subconjuntos de TB1, o B&B-GVNS apresentou desempenho superior do que o CPLEX em 4 de 5 subconjuntos e, no outro, apresentou desempenho inferior devido a uma variação na média de t de +0,54%. Considerando todo o conjunto, o B&B-GVNS apresentou desempenho superior, com redução de gap% e de t de 1,27% e 4,75%, respectivamente. Além disso, o B&B-GVNS encontrou dois LSs de melhor

qualidade e apenas um com qualidade pior do que o CPLEX.

Nos subconjuntos de TB2 e TB3, que totalizam 13 subconjuntos, o B&B-GVNS apresentou desempenho superior em todos eles, reduzindo consideravelmente o valores de gap% e obtendo LSs de melhor qualidade para todas as instâncias de 10 subconjuntos. Ressalta-se que a redução percentual dos gap%s foi superior a 15% em 12 subconjuntos e ultrapassou 30% em três deles (y6-y10, y16-y20 e g26-g30). Considerando todas as instâncias, o B&B-GVNS reduziu o gap% em 25,38% no conjunto TB2 e em 20,25% no conjunto TB3. Além disso, o B&B-GVNS obteve 17 de 20 (85%) e 44 de 45 valores (98%) de LSs de melhor qualidade nos conjuntos TB2 e TB3, respectivamente, e nenhum valor de LS com qualidade pior em TB3.

Dentre os sete subconjuntos de TB6, o B&B-GVNS apresentou desempenho superior em 5 deles (d1-d6, d18-d25, d26-d33, d42-d49 e d50-d57) e inferior em apenas um (d7-d17). No subconjunto d34-d41, o B&B-GVNS apresentou vantagem ao reduzir t , empate no número de LSs de melhor e de pior qualidade, e desvantagem ao aumentar o gap%. Isto tornou a comparação inconclusiva pelo critério adotado. Destaca-se que, nos subconjuntos d1-d7 e d18-d25, o B&B-GVNS reduziu o tempo do CPLEX em 39,83% e 45,92%, respectivamente, para obter valores ótimos em todas as instâncias. Considerando todas as instâncias do conjunto, o B&B-GVNS apresentou redução de gap% de 5,56%, obteve 10 valores de LS com qualidade melhor e 4 com qualidade pior, além de redução de t de 2,55%.

Considerando todos os 25 subconjuntos do experimento, o B&B-GVNS apresentou desempenho superior em 22 deles (88%), inferior em apenas 2 (8%) e, em 1 deles, a comparação de desempenho foi inconclusiva. Na análise por conjunto, o B&B-GVNS apresentou desempenho superior em todos eles. Os resultados indicam que o B&B-GVNS supera o CPLEX na resolução do RSSCFLP, com vantagem mais evidente em conjuntos de instâncias mais desafiadores. Os três principais componentes que diferenciam o B&B-GVNS da resolução direta pelo CPLEX são o procedimento de planos de corte (Fase I), a heurística primal (Fase II) e o aprimoramento do CPLEX por meio da incorporação de heurísticas (Fase III). Os resultados sugerem que a aplicação desses procedimentos traz benefícios que justificam a superioridade do B&B-GVNS. Para verificar se todos eles contribuem efetivamente para o desempenho superior do B&B-GVNS, avaliou-se o desempenho do B&B-GVNS quando cada um deles é removido. Para isso, consideram-se B&B-GVNS-1, B&B-GVNS-2 e B&B-GVNS-3 como variantes do B&B-GVNS obtidas pela remoção, respectivamente, do procedimento de planos de corte, da heurística primal e do aprimoramento via incorporação de heurísticas ao algoritmo baseado em B&B. Essas variantes foram executadas na amostra $\mathcal{S}$ apresentada na Subseção 6.4, e os resultados foram comparados com os obtidos pela versão original do B&B-GVNS. Os resultados são apresentados na Tabela 6.23.

Tabela 6.23: Resultado da avaliação de componentes do B&B-GVNS

	B&B-GVNS	B&B-GVNS-1 dev(%)	B&B-GVNS-2 dev(%)	B&B-GVNS-3 dev(%)
gap%	2,99	+0,51	+5,89	+2,73
t	2454,62	+0,19	+0,81	+14,59

Os resultados mostram que, para todas as variantes, as variações percentuais dos valores de gap% e de t em relação aos obtidos pelo B&B-GVNS são positivas. Assim, a remoção de qualquer um dos procedimentos reduz o desempenho do algoritmo, o que sugere que todos eles contribuem para o desempenho superior do B&B-GVNS em relação ao CPLEX. Em termos de qualidade das soluções, a heurística primal baseada em GVNS é o procedimento que mais beneficia o B&B-GVNS, pois sua remoção foi a que gerou o maior impacto negativo nos valores de gap% (+5,89%). Já em termos de tempo computacional, a incorporação de heurísticas nos nós da árvore de B&B é o procedimento de maior impacto, uma vez que sua remoção resultou no maior aumento no tempo computacional (+14,59%).

6.4.2 Avaliação do modelo do RSSCFLP

Para avaliar o modelo do RSSCFLP, o experimento descrito na seção anterior para a determinação das constantes de proporcionalidade da expressão (6.2) foi repetido para todas as combinações de $\delta \in \{0,10; 0,20; 0,30\}$ e $\sigma \in \{0,05; 0,10; 0,15\}$, totalizando nove experimentos.

Em cada iteração de cada experimento, calculou-se, além dos indicadores de robustez definidos na Seção 6.4, a média do aumento percentual dos custos garantidos das soluções da amostra em relação aos custos das respectivas soluções nominais, denotada por ACG%. Esse indicador estima o aumento do custo garantido da solução ótima do problema robusto em relação ao valor ótimo do problema nominal para a configuração de parâmetros obtida em cada iteração.

A Tabela 6.24 apresenta um resumo dos resultados de cada experimento. Observa-se que, em todos os casos, os valores de parâmetros obtidos garantiram valores de TV%, TRCALG% e TRCABG% maiores do que 90%, exceto quando $\sigma = 0,15$, nos quais TV% alcançou valores de 65,7% a 70,6%. Esses resultados indicam que o procedimento produz soluções com elevadas taxas de robustez para cenários de baixa e moderada variabilidade e mantém elevadas taxas de robustez dos custos mesmo em cenários de maior variabilidade, embora, neste último caso, a taxa de viabilidade seja reduzida.

Para os três valores de σ , os quatro parâmetros na configuração final foram maiores

Tabela 6.24: Análise de robustez do modelo do RSSCFLP

δ	Valores de parâmetros				TV	TRCALG	TRCABG	ACG
(%)	κ^c	κ^f	κ^d	κ^b	(%)	(%)	(%)	(%)
$\sigma = 0,05$								
10%	0,1	0,3	0,4	0,3	92,8	99,8	96,9	6,6
20%	0,1	0,1	0,4	0	94,7	100,0	95,9	8,2
30%	0,1	0,1	0,3	0	95,6	100,0	98,4	12,5
$\sigma = 0,10$								
10%	0,1	0,5	0,7	1,1	92,0	94,8	94,1	12,8
20%	0,1	0,3	0,6	0,3	94,0	99,8	96,4	19,2
30%	0,1	0,2	0,5	0,1	95,4	100,0	95,8	22,1
$\sigma = 0,15$								
10%	0,2	0,7	0,9	1,5	70,6	96,9	90,8	22,3
20%	0,1	0,4	0,5	0,5	67,1	98,1	94,4	22,9
30%	0,1	0,3	0,4	0,2	65,7	99,8	96,4	23,9

para $\delta = 0,10$ do que para $\delta = 0,30$. Além disso, os valores de ACG foram maiores quando $\delta = 0,30$ do que quando $\delta = 0,10$. Esses resultados sugerem que, à medida que o intervalo de incerteza diminui, os níveis de robustez necessários para garantir as taxas de robustez desejadas aumentam. Adicionalmente, há uma tendência do procedimento gerar soluções menos conservadoras, o que resulta em valores de custo garantido menores. Esse comportamento pode ser explicado pelo fato de que quando os níveis de incerteza são relativamente pequenos, os ajustes nos parâmetros resultam em variações menores nas taxas de robustez. Dessa forma, a porcentagem-alvo tende a ser alcançada com menor folga do que em níveis de incerteza maiores.

Analisando os resultados, com foco no efeito da variabilidade da distribuição normal, observa-se que, para um mesmo valor de δ , os valores dos parâmetros obtidos para $\sigma = 0,15$ são maiores ou iguais aos obtidos para $\sigma = 0,05$, e os valores de ACG% são maiores para $\sigma = 0,15$ do que para $\sigma = 0,05$. Esses resultados indicam que a maior variabilidade dos parâmetros acarreta a necessidade de soluções mais conservadoras, o que se reflete em valores de custo garantidos mais elevados.

Também foi analisado o efeito do ajuste de cada parâmetro nas taxas de robustez e nos custos das soluções do RSSCFLP. Para isso, calculou-se a variação média das taxas de robustez e de ACG% quando cada parâmetro é incrementado individualmente 0,1, considerando os dados dos nove experimentos realizados. Os resultados são apresentados na Tabela 6.25.

Os resultados mostram que incrementos em κ^c e κ^f aumentam significativamente TRCALG e TRCABG, respectivamente, sem alterar significativamente as demais

Tabela 6.25: Resultado da avaliação do efeito do incremento de cada parâmetro

Parâmetro	Variação média causada pelo incremento			
	TV%	TRCALG%	TRCABG%	ACG%
κ^c	0,48	59,07	-0,04	2,48
κ^f	-0,10	0,01	16,29	1,06
κ^d	8,42	0,00	-0,08	1,19
κ^b	4,86	-0,01	-0,09	1,09

Tabela 6.26: Custo marginal associado ao aumento das taxas de robustez

Taxa de robustez	Parâmetro	Cálculo	Custo marginal (%)
TRCALG	κ^c	2,48/59,07	0,04
TRCABG	κ^f	1,06/16,29	0,07
TV	κ^d	1,19/8,42	0,14
TV	κ^b	1,09/4,86	0,22

taxas de robustez. De forma análoga, incrementos em κ^d e κ^b aumentam significativamente TV, com impacto desprezível nas demais taxas. Os aumentos mais relevantes estão destacados em negrito na tabela.

A partir dos resultados da Tabela 6.25, é possível estimar o custo marginal de cada tipo de proteção, definido como a razão entre a variação em ACG% e a variação correspondente da taxa de robustez. Os valores obtidos são apresentados na Tabela 6.26.

Observa-se que aumentar a robustez associada aos custos de alocação (TRCALG) apresenta o menor custo marginal, seguido da robustez dos custos de abertura (TRCABG). Em contrapartida, elevar a taxa de viabilidade (TV) é relativamente mais oneroso. Além disso, entre os dois mecanismos que influenciam TV, o incremento de κ^d apresenta menor custo marginal do que o incremento de κ^b , indicando que aumentar a robustez associada às demandas é mais eficiente do que aumentar a robustez associada às capacidades.

Esses resultados sugerem que aumentar a robustez associada aos custos de alocação é relativamente menos oneroso, enquanto aumentar a proteção contra inviabilidade é mais custoso. Além disso, no que se refere à inviabilidade, observa-se que é mais eficiente aumentar a robustez por meio dos parâmetros associados à variação da demanda (κ^d) do que por meio daqueles associados à variação da capacidade (κ^b).

Os códigos-fonte das implementações computacionais desenvolvidas nesta tese, bem como os resultados completos dos experimentos computacionais realizados com os algoritmos HILS, NCS e B&B-GVNS, estão disponíveis publicamente em <https://github.com/gbalmeida>.

Capítulo 7

Conclusões e Sugestões de Trabalhos Futuros

7.1 Conclusões

Esta tese aborda o Problema de Localização Capacitado com Fonte Única (SSCFLP) e sua variante robusta (RSSCFLP). O SSCFLP consiste em localizar facilidades capacitadas para atender às demandas de clientes, de modo que cada cliente seja atendido por uma única facilidade. Já o RSSCFLP é uma extensão do SSCFLP que considera incertezas em todos os tipos de parâmetros: custos de alocação, custos de abertura, demandas e capacidades. A abordagem robusta de Bertsimas & Sim é proposta para formular este último problema.

Para resolver os problemas, são propostos três algoritmos: o HILS e o NCS para resolver o SSCFLP; e o B&B-GVNS para resolver o RSSCFLP. O HILS é um algoritmo heurístico que, primeiramente, explora a solução da relaxação linear para identificar locais promissores e reduzir o espaço de busca. Em seguida, alterna entre a resolução de subproblemas por meio de um *solver* e a aplicação de procedimentos ILS para encontrar boas soluções. O NCS é um algoritmo exato que, em sua primeira fase, fortalece a formulação do SSCFLP por meio de um procedimento de planos de corte e resolve uma relaxação parcial do modelo fortalecido para obter um limitante inferior forte para o problema. Em sua segunda fase, o NCS aplica um esquema *cut-and-solve*, incorporando um procedimento de busca local baseado em clusterização para encontrar a solução ótima do problema. Já o B&B-GVNS é um algoritmo exato para resolver o RSSCFLP composto por três fases: a primeira fortalece a formulação por meio de um procedimento de planos de corte, a segunda aplica procedimentos heurísticos GVNS para encontrar um bom limitante superior e a terceira aprimora um algoritmo B&B, incorporando buscas locais baseadas em VND e GVNS para resolver

o problema.

Experimentos computacionais foram realizados com os algoritmos em conjuntos *benchmark* de instâncias. Os resultados mostraram que o algoritmo HILS produziu soluções com valores de gap% ligeiramente melhores do que os obtidos por algoritmos do estado da arte e atualizou 40% dos BKVs dentre as instâncias cujos valores ótimos não são conhecidos. Já os resultados para o NCS mostraram que ele encontrou mais valores ótimos ou reduziu o tempo para obter a mesma quantidade em 22 dos 26 subconjuntos analisados, quando comparado ao algoritmo exato do estado da arte, e em 19 dos 26 subconjuntos, quando comparado ao CPLEX. Além disso, o NCS foi mais de 100 vezes mais rápido do que o CPLEX e mais de 10 vezes mais rápido do que o algoritmo do estado da arte em vários subconjuntos para encontrar um número maior ou igual de soluções ótimas. Por fim, os experimentos com o B&B-GVNS mostraram que ele apresentou desempenho superior ao do CPLEX nos conjuntos de instâncias considerados. Nos dois conjuntos mais desafiadores, a vantagem do B&B-GVNS se mostrou mais evidente. Em um deles, o B&B-GVNS reduziu o gap% em mais de 20% e encontrou soluções melhores em 85% das instâncias. No outro, a redução do gap% foi superior a 25% com soluções melhores em 98% das instâncias.

Os resultados dos experimentos mostraram que os algoritmos desenvolvidos nesta tese constituem contribuições metodológicas relevantes para a resolução dos problemas abordados.

7.2 Propostas de trabalhos futuros

Como perspectiva para trabalhos futuros, sugere-se a proposição de duas outras abordagens para o RSSCFLP. Uma delas é propor modelos robustos que consideram correlação entre variáveis incertas. Uma direção nesse sentido é se basear na abordagem proposta em [Jalilvand-Nejad et al. \(2016\)](#). A outra abordagem é propor modelos robustos que consideram planos de contingência, como no modelo proposto em [Munari et al. \(2019\)](#) para o problema de roteamento de veículos.

Sugere-se, também, o desenvolvimento de algoritmos para o Problema Capacitado Modular de Localização de Facilidades com Fonte Única (*Single-Source Modular Capacitated Facility Location Problem* – SSMCLP) e para o Problema Capacitado de Localização de Facilidades com Fonte Única em Duas Escalas (*Two-Echelon Single-Source Capacitated Facility Location Problem* – TSCFLP). O SSMCLP é uma extensão do SSCFLP que considera capacidades modulares, isto é, a cada facilidade deve ser atribuída uma capacidade dentre um conjunto de valores possíveis ([Correia & Eugénia Captivo, 2006](#)). Já o TSCFLP estende o SSCFLP ao considerar dois níveis de localização de facilidades ([Tragantalerngsak et al., 2000](#)).

Como o SSCFLP é um subproblema de ambas as extensões, os procedimentos de separação de planos de corte utilizados no NCS podem ser explorados para fortalecer as formulações dessas extensões. Além disso, a estratégia de considerar relaxações lineares parciais, adotada no NCS, pode ser empregada para obter limitantes inferiores de alta qualidade. Caso o SSMCLP e o TSCFLP apresentem comportamento semelhante ao observado no SSCFLP, a solução de relaxações parciais fortes pode fornecer o conjunto de locais com facilidades abertas na solução ótima ou uma aproximação desse conjunto.

Para desenvolvimento de algoritmos exatos, o êxito da estratégia *cut-and-solve* no algoritmo NCS e em outros problemas de localização indica que sua aplicação constitui uma direção promissora de pesquisa para o SSMCLP e o TSCFLP. Um ponto importante em um estudo que siga essa direção é investigar a viabilidade de resolver os problemas esparsos gerados por esse esquema. Como alternativa ao *cut-and-solve*, procedimentos de planos de corte podem ser integrados a um procedimento *branch-and-bound* aprimorado por heurísticas. Esse caminho é promissor, pois se mostrou bastante eficaz no RSSCFLP, que também é uma extensão do SSCFLP.

Para a obtenção de algoritmos heurísticos ou para acelerar a convergência de métodos exatos, o uso de meta-heurísticas constitui uma direção promissora de pesquisa. As meta-heurísticas são amplamente aplicadas na literatura para resolução de diversos problemas de otimização. Nesta tese, as meta-heurísticas ILS e GVNS foram aplicadas com êxito para resolver o SSCFLP e o RSSCFLP, respectivamente. Em investigações nessa direção, as estruturas de vizinhança propostas nesta tese podem ser reutilizadas, adaptadas ou estendidas.

Referências

- Afify, B., Soeanu, A., & Awasthi, A. (2021). Separation linearization approach for the capacitated facility location problem under disruption. *Expert Systems with Applications*, 169, 114187. doi: <https://doi.org/10.1016/j.eswa.2020.114187>.
- Ahuja, R. K., Orlin, J. B., Pallottino, S., Scaparra, M. P., & Scutellà, M. G. (2004). A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science*, 50, 749–760. doi: 10.1287/mnsc.1030.0193.
- Albareda-Sambola, M., Fernández, E., & da Gama, F. S. (2011). The facility location problem with bernoulli demands. *Omega*, 39, 335–345. doi: <https://doi.org/10.1016/j.omega.2010.08.002>.
- Angelelli, E., Mansini, R., & Grazia Speranza, M. (2010). Kernel search: A general heuristic for the multi-dimensional knapsack problem. *Computers & Operations Research*, 37, 2017–2026. URL: <https://www.sciencedirect.com/science/article/pii/S0305054810000328>. doi: <https://doi.org/10.1016/j.cor.2010.02.002>. Metaheuristics for Logistics and Vehicle Routing.
- Arenales, M., Armentano, V., Morabito, R., & Yanasse, H. (2007). *Pesquisa operacional*. Coleção Campus-ABEPRO engenharia de produção. Elsevier. URL: <https://books.google.com.br/books?id=xjLOGAAACAAJ>.
- Armas, J., Juan, A., Marquès, J., & Pedroso, J. P. (2016). Solving the deterministic and stochastic uncapacitated facility location problem: from a heuristic to a simheuristic. *Journal of the Operational Research Society*, . doi: 10.1057/s41274-016-0155-6.
- Atamturk, A. (2005). Cover and pack inequalities for (mixed) integer programming. *Annals OR*, 139, 21–38. doi: 10.1007/s10479-005-3442-1.
- Avella, P., & Boccia, M. (2009). A cutting plane algorithm for the capacitated facility location problem. *Computational Optimization and Applications*, 43, 39–65. doi: 10.1007/s10589-007-9125-x.

- Bakker, H., Guastaroba, G., Nickel, S., & Speranza, M. G. (2025). Enhancing kernel search with pattern recognition: the single-source capacitated facility location problem. URL: <https://arxiv.org/abs/2512.08576>. arXiv:2512.08576.
- Balas, E., & Toth, P. (1983). Branch and bound methods. In E. Lawler, J. Lenstra, A. Rinnooy Kan, & D. Shmoys (Eds.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons.
- Ball, M., Assad, A., Bodin, L., Golden, B., & Spielberg, F. (1984). Garage location for an urban mass transit system. *Transportation Science*, 18, 56–75. URL: <https://doi.org/10.1287/trsc.18.1.56>. doi: 10.1287/trsc.18.1.56. arXiv:<https://doi.org/10.1287/trsc.18.1.56>.
- Barceló, J., & Casanovas, J. (1984). A heuristic lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research*, 15, 212–226. doi: 10.1016/0377-2217(84)90211-X.
- Baron, O., Milner, J., & Naseraldin, H. (2011). Facility location: A robust optimization approach. *Production and Operations Management*, 20, 772–785. doi: <https://doi.org/10.1111/j.1937-5956.2010.01194.x>.
- Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2005). *Linear Programming and Network Flows*. (3rd ed.). New York: John Wiley & Sons.
- Beasley, J. (1988). An algorithm for solving large capacitated warehouse location problems. *European Journal of Operational Research*, 33, 314–325. URL: <https://www.sciencedirect.com/science/article/pii/0377221788901750>. doi: [https://doi.org/10.1016/0377-2217\(88\)90175-0](https://doi.org/10.1016/0377-2217(88)90175-0).
- Beasley, J. E. (1990). OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41, 1069–1072. URL: <https://doi.org/10.1057/jors.1990.166>. doi: 10.1057/jors.1990.166.
- Ben-Tal, A., & Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, 23, 769–805. URL: <http://www.jstor.org/stable/3690632>.
- Bertsimas, D., Brown, D. B., & Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM Review*, 53, 464–501. URL: <https://doi.org/10.1137/080734510>. doi: 10.1137/080734510. arXiv:<https://doi.org/10.1137/080734510>.
- Bertsimas, D., & Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, 98, 49–71. URL: <https://doi.org/10.1007/s10107-003-0396-4>. doi: 10.1007/s10107-003-0396-4.

- Bertsimas, D., & Sim, M. (2004). The price of robustness. *Operations Research*, 52, 35–53. URL: <https://doi.org/10.1287/opre.1030.0065>. doi: 10.1287/opre.1030.0065.
- Bieniek, M. (2015). A note on the facility location problem with stochastic demands. *Omega*, 55, 53–60. doi: <https://doi.org/10.1016/j.omega.2015.02.006>.
- Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35, 268–308. URL: <https://doi.org/10.1145/937503.937505>. doi: 10.1145/937503.937505.
- Boonmee, C., Arimura, M., & Asada, T. (2017). Facility location optimization model for emergency humanitarian logistics. *International Journal of Disaster Risk Reduction*, 24, 485–498. URL: <https://www.sciencedirect.com/science/article/pii/S2212420916302576>. doi: <https://doi.org/10.1016/j.ijdr.2017.01.017>.
- Boschetti, M. A., & Maniezzo, V. (2022). Matheuristics: using mathematics for heuristic design. *4OR*, 20, 173–208.
- Boyd, E. A. (1993). Generating fenchel cutting planes for knapsack polyhedra. *SIAM Journal on Optimization*, 3, 734–750. doi: 10.1137/0803038.
- Campos, R. A., Chagas, G. O., Coelho, L. C., & Munari, P. (2025). Exact methods and a variable neighborhood search for the robust capacitated p-median problem. *Computers & Operations Research*, 173, 106851. doi: <https://doi.org/10.1016/j.cor.2024.106851>.
- Caserta, M., & Voß, S. (2020). A general corridor method-based approach for capacitated facility location. *International Journal of Production Research*, 58, 3855–3880. URL: <https://doi.org/10.1080/00207543.2019.1636320>. doi: 10.1080/00207543.2019.1636320.
- Chen, C.-H., & Ting, C.-J. (2008). Combining Lagrangian heuristic and Ant Colony System to solve the Single Source Capacitated Facility Location Problem. *Transportation Research Part E: Logistics and Transportation Review*, 44, 1099–1122.
- Cheng, C., Yu, Q., Adulyasak, Y., & Rousseau, L.-M. (2024). Distributionally robust facility location with uncertain facility capacity and customer demand. *Omega*, 122, 102959. doi: <https://doi.org/10.1016/j.omega.2023.102959>.
- Climer, S., & Zhang, W. (2006). Cut-and-solve: An iterative search strategy for combinatorial optimization problems. *Artificial Intelligence*,

170, 714–738. URL: <https://www.sciencedirect.com/science/article/pii/S0004370206000233>. doi: <https://doi.org/10.1016/j.artint.2006.02.005>.

Contreras, I., & Díaz, J. (2008). Scatter search for the single source capacitated facility location problem. *Annals of Operations Research*, 157, 73–89. doi: 10.1007/s10479-007-0193-1.

Correia, I., & Eugénia Captivo, M. (2006). Bounds for the single source modular capacitated plant location problem. *Computers & Operations Research*, 33, 2991–3003. doi: 10.1016/j.cor.2005.02.030. Part Special Issue: Constraint Programming.

Cortinhal, M. J., & Captivo, M. E. (2003). Upper and lower bounds for the single source capacitated location problem. *European Journal of Operational Research*, 151, 333–351. doi: 10.1016/S0377-2217(02)00829-9.

Cortinhal, M. J., & Captivo, M. E. (2004). Genetic algorithms for the single source capacitated location problem. In *Metaheuristics: Computer Decision-Making* (pp. 187–216). Boston, MA: Springer US. doi: 10.1007/978-1-4757-4137-7_9.

Crowder, H., Johnson, E. L., & Padberg, M. (1983). Solving large-scale zero-one linear programming problems. *Operations Research*, 31, 803–834. doi: 10.1287/opre.31.5.803.

Danna, E. (2004). Integrating local search techniques into mixed integer programming. *4OR*, 2, 321–324. doi: 10.1007/s10288-004-0042-x.

Daskin, M. S., Hesse, S. M., & Reville, C. S. (1997). α -reliable p-minimax regret: A new model for strategic facility location modeling. *Location Science*, 5, 227–246. doi: [https://doi.org/10.1016/S0966-8349\(98\)00036-9](https://doi.org/10.1016/S0966-8349(98)00036-9).

Daskin, M. S., & Jones, P. C. (1993). A new approach to solving applied location/allocation problems. *Computer-Aided Civil and Infrastructure Engineering*, 8, 409–421. doi: <https://doi.org/10.1111/j.1467-8667.1993.tb00226.x>.

Delmaire, H., Díaz, J. A., Fernández, E., & Ortega, M. (1999). Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. *INFOR: Information Systems and Operational Research*, 37, 194–225. doi: 10.1080/03155986.1999.11732381.

Díaz, J. A., & Fernández, E. (2002). A branch-and-price algorithm for the single source capacitated plant location problem. *Journal of the Operational Research Society*, 53, 728–740. doi: 10.1057/palgrave.jors.2601353.

- Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26, 29–41. doi: 10.1109/3477.484436.
- Dorigo, M., & Stützle, T. (2019). Ant colony optimization: Overview and recent advances. In M. Gendreau, & J.-Y. Potvin (Eds.), *Handbook of Metaheuristics* (pp. 311–351). Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-91086-4_10. doi: 10.1007/978-3-319-91086-4_10.
- Fang, Y., Chu, F., Mammari, S., & Che, A. (2014). A cut-and-solve-based algorithm for optimal lane reservation with dynamic link travel times. *International journal of production research*, 52, 1003–1015.
- Fischetti, M., & Fischetti, M. (2018). Matheuristics. In R. Martí, P. M. Pardalos, & M. G. C. Resende (Eds.), *Handbook of Heuristics* (pp. 121–153). Cham: Springer volume 1-2. doi: 10.1007/978-3-319-07124-4_14.
- Fischetti, M., & Lodi, A. (2003). Local branching. *Mathematical Programming*, 98, 23–47. doi: 10.1007/s10107-003-0395-5.
- Fisher, M. L., Jaikumar, R., & Van Wassenhove, L. N. (1986). A multiplier adjustment method for the generalized assignment problem. *Management science*, 32, 1095–1103.
- Gadegaard, S., Klose, A., & Nielsen, L. (2018). An improved cut-and-solve algorithm for the single-source capacitated facility location problem. *EURO Journal on Computational Optimization*, 6, 1–27. doi: <https://doi.org/10.1007/s13675-017-0084-4>.
- Geoffrion, A. M., & Graves, G. W. (1974). Multicommodity distribution system design by benders decomposition. *Management Science*, 20, 822–844. URL: <http://www.jstor.org/stable/2630093>.
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13, 533–549. doi: [https://doi.org/10.1016/0305-0548\(86\)90048-1](https://doi.org/10.1016/0305-0548(86)90048-1). Applications of Integer Programming.
- Glover, F. (1998). A template for scatter search and path relinking. In *Artificial Evolution* (pp. 13–54). Springer volume 1363 of *Lecture Notes in Computer Science*.
- Gnägi, M., & Baumann, P. (2021). A matheuristic for large-scale capacitated clustering. *Computers & Operations Research*, 132, 105304. URL: <https://www.sciencedirect.com/science/article/pii/S0305054821000952>. doi: <https://doi.org/10.1016/j.cor.2021.105304>.

- Goldberg, D. E. (1988). *Genetic Algorithms in Search Optimization and Machine Learning*. Addison Wesley.
- Gollowitzer, S., & Ljubić, I. (2011). Mip models for connected facility location: A theoretical and computational study. *Computers & Operations Research*, 38, 435–449. URL: <https://www.sciencedirect.com/science/article/pii/S0305054810001334>. doi: <https://doi.org/10.1016/j.cor.2010.07.002>.
- Gu, Z., Nemhauser, G. L., & Savelsbergh, M. W. (2000). Sequence Independent Lifting in Mixed Integer Programming. *Journal of Combinatorial Optimization*, 4, 109–129. URL: https://ideas.repec.org/a/spr/jcomop/v4y2000i1d10.1023_a1009841107478.html. doi: 10.1023/A:1009841107478.
- Guastaroba, G., & Speranza, M. (2014). A heuristic for BILP problems: The single source capacitated facility location problem. *European Journal of Operational Research*, 238, 438–450. URL: <https://www.sciencedirect.com/science/article/pii/S0377221714003166>. doi: <https://doi.org/10.1016/j.ejor.2014.04.007>.
- Hansen, P., & Mladenović, N. (2003). *A Tutorial on Variable Neighborhood Search*. Technical Report LES CAHIERS DU GERAD, HEC MONTREAL AND GERAD. Available at <https://www.cs.uleth.ca/~benkoczi/OR/read/vns-tutorial.pdf>.
- Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, 5, 423–454. doi: 10.1007/s13675-016-0075-x.
- Hansen, P., & Mladenović, N. (2005). Variable neighborhood search. In E. Burke, & G. Kendall (Eds.), *Search Methodologies*. Springer.
- Hindi, K. S., & Pieńkosz, K. (1999). Efficient solution of large scale, single-source, capacitated plant location problems. *Journal of the Operational Research Society*, 50, 268–274. doi: 10.1057/palgrave.jors.2600698.
- Ho, S. C. (2015). An iterated tabu search heuristic for the single source capacitated facility location problem. *Applied Soft Computing*, 27, 169–178. doi: <https://doi.org/10.1016/j.asoc.2014.11.004>.
- Hoffman, K. L., & Padberg, M. (1991). Improving lp-representations of zero-one linear programs for branch-and-cut. *ORSA Journal on Computing*, 3, 121–134. doi: 10.1287/ijoc.3.2.121.

- Holmberg, K., Rönnqvist, M., & Yuan, D. (1999). An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research*, 113, 544–559. URL: <https://www.sciencedirect.com/science/article/pii/S0377221798000083>. doi: [https://doi.org/10.1016/S0377-2217\(98\)00008-3](https://doi.org/10.1016/S0377-2217(98)00008-3).
- Hu, P., Chu, F., Liu, M., Wang, S., & Wu, P. (2022). An integrated approach for a new flexible multi-product disassembly line balancing problem. *Computers & operations research*, 148, 105932.
- Jalilvand-Nejad, A., Shafaei, R., & Shahriari, H. (2016). Robust optimization under correlated polyhedral uncertainty set. *Computers & Industrial Engineering*, 92, 82–94. URL: <https://www.sciencedirect.com/science/article/pii/S0360835215004763>. doi: <https://doi.org/10.1016/j.cie.2015.12.006>.
- Kaparis, K., & Letchford, A. (2010). Separation algorithms for 0-1 knapsack polytopes. *Mathematical Programming*, 124, 69–91. doi: 10.1007/s10107-010-0359-5.
- Kchaou-Boujelben, M. (2021). Charging station location problem: A comprehensive review on models and solution approaches. *Transportation Research Part C: Emerging Technologies*, 132, 103376. doi: <https://doi.org/10.1016/j.trc.2021.103376>.
- Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220, 671–680. URL: <https://www.science.org/doi/abs/10.1126/science.220.4598.671>. doi: 10.1126/science.220.4598.671. arXiv:<https://www.science.org/doi/pdf/10.1126/science.220.4598.671>.
- Klincewicz, J. G., & Luss, H. (1986). A lagrangian relaxation heuristic for capacitated facility location with single-source constraints. *Journal of the Operational Research Society*, 37, 495–500. doi: 10.1057/jors.1986.84.
- Koca, E. (2023). Two-stage stochastic facility location problem with disruptions and restricted shortages. *Computers & Industrial Engineering*, 183, 109484. doi: <https://doi.org/10.1016/j.cie.2023.109484>.
- Kong, Y. (2021). A matheuristic algorithm for the single-source capacitated facility location problem and its variants. doi: 10.48550/arXiv.2112.12974.
- Lawler, E. L., & Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations Research*, 14, 699–719. doi: 10.1287/opre.14.4.699.

- Letchford, A. N., & Souli, G. (2019). On lifted cover inequalities: A new lifting procedure with unusual properties. *Operations Research Letters*, 47, 83–87. URL: <https://www.sciencedirect.com/science/article/pii/S016763771830422X>. doi: <https://doi.org/10.1016/j.orl.2018.12.005>.
- Lin, C. (2009). Stochastic single-source capacitated facility location model with service level requirements. *International Journal of Production Economics*, 117, 439–451. doi: <https://doi.org/10.1016/j.ijpe.2008.11.009>.
- Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In F. Glover, & G. A. Kochenberger (Eds.), *Handbook of Metaheuristics* (pp. 320–353). Boston, MA: Springer US. URL: https://doi.org/10.1007/0-306-48056-5_11. doi: 10.1007/0-306-48056-5_11.
- Louveaux, F. V., & Peeters, D. (1992). A dual-based procedure for stochastic facility location. *Operations Research*, 40, 564–573. doi: 10.1287/opre.40.3.564. arXiv:<https://doi.org/10.1287/opre.40.3.564>.
- Luo, J.-Y., Chen, L., Chen, W.-K., Yuan, J.-H., & Dai, Y.-H. (2022). A cut-and-solve algorithm for virtual machine consolidation problem. URL: <http://arxiv.org/licenses/nonexclusive-distrib/1.0>.
- López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3, 43–58. doi: 10.1016/j.orp.2016.09.002.
- Melo, M., Nickel, S., & Saldanha-da-Gama, F. (2006). Dynamic multi-commodity capacitated facility location: a mathematical modeling framework for strategic supply chain planning. *Computers & Operations Research*, 33, 181–208. doi: <https://doi.org/10.1016/j.cor.2004.07.005>.
- Munari, P., Moreno, A., De La Vega, J., Alem, D., Gondzio, J., & Morabito, R. (2019). The robust vehicle routing problem with time windows: Compact formulation and branch-price-and-cut method. *Transportation Science*, 53, 1043–1066. URL: <https://doi.org/10.1287/trsc.2018.0886>. doi: 10.1287/trsc.2018.0886. arXiv:<https://doi.org/10.1287/trsc.2018.0886>.
- Neebe, A. W., & Rao, M. R. (1983). An algorithm for the fixed-charge assigning users to sources problem. *Journal of the Operational Research Society*, 34, 1107–1113. doi: 10.2307/2581020.

- Nemhauser, G., & Wolsey, L. (1988a). General algorithms. In *Integer and Combinatorial Optimization* chapter II.4. (pp. 349–382). John Wiley & Sons, Ltd. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118627372.ch11>. doi: <https://doi.org/10.1002/9781118627372.ch11>.
- Nemhauser, G., & Wolsey, L. (1988b). The theory of valid inequalities. In *Integer and Combinatorial Optimization* chapter II.1. (pp. 203–258). John Wiley & Sons, Ltd. doi: 10.1002/9781118627372.ch8.
- Novaes, A. G., Souza de Cursi, J., da Silva, A. C., & Souza, J. C. (2009). Solving continuous location–districting problems with voronoi diagrams. *Computers & Operations Research*, 36, 40–59. doi: <https://doi.org/10.1016/j.cor.2007.07.004>. Part Special Issue: Operations Research Approaches for Disaster Recovery Planning.
- Oliveira, O., Matos, T., & Gamboa, D. (2021). A dual RAMP algorithm for single source capacitated facility location problems. *Annals of Mathematics and Artificial Intelligence*, 89, 815–834. doi: 10.1007/s10472-021-09756-0.
- Pantuso, G. (2025). Solution of stochastic facility location problems with combinatorially many decision-dependent distributions. *Journal of Combinatorial Optimization*, 50. URL: <https://doi.org/10.1007/s10878-025-01371-7>. doi: 10.1007/s10878-025-01371-7.
- Pirkul, H., & Schilling, D. A. (1988). The siting of emergency service facilities with workload capacities and backup service. *Management Science*, 34, 896–908. URL: <https://doi.org/10.1287/mnsc.34.7.896>. doi: 10.1287/mnsc.34.7.896. arXiv:<https://doi.org/10.1287/mnsc.34.7.896>.
- Pisinger, D. (1997). A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, 45, 758–767. doi: 10.1287/opre.45.5.758.
- Plane, D. R., & Hendrick, T. E. (1977). Mathematical programming and the location of fire companies for the denver fire department. *Operations Research*, 25, 563–578. URL: <https://doi.org/10.1287/opre.25.4.563>. doi: 10.1287/opre.25.4.563. arXiv:<https://doi.org/10.1287/opre.25.4.563>.
- Qin, J., Ni, L.-L., & Shi, F. (2012). Combined simulated annealing algorithm for the discrete facility location problem. *The Scientific World Journal*, 2012, 576392. URL: <https://doi.org/10.1100/2012/576392>. doi: <https://doi.org/10.1100/2012/576392>.

- Resende, M. G. C., & Ribeiro, C. C. (2019). Greedy randomized adaptive search procedures: Advances and extensions. In M. Gendreau, & J.-Y. Potvin (Eds.), *Handbook of Metaheuristics* (pp. 169–220). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-91086-4_6. doi: 10.1007/978-3-319-91086-4_6.
- Ryu, J., & Park, S. (2022). A branch-and-price algorithm for the robust single-source capacitated facility location problem under demand uncertainty. *EURO Journal on Transportation and Logistics*, 11, 100069. doi: 10.1016/j.ejtl.2021.100069.
- Saif, A., & Delage, E. (2021). Data-driven distributionally robust capacitated facility location problem. *European Journal of Operational Research*, 291, 995–1007. URL: <https://www.sciencedirect.com/science/article/pii/S0377221720308304>. doi: <https://doi.org/10.1016/j.ejor.2020.09.026>.
- Silver, E. A., Victor, R., Vidal, V., & de Werra, D. (1980). A tutorial on heuristic methods. *European Journal of Operational Research*, 5, 153–162. URL: <https://www.sciencedirect.com/science/article/pii/0377221780900843>. doi: [https://doi.org/10.1016/0377-2217\(80\)90084-3](https://doi.org/10.1016/0377-2217(80)90084-3).
- Snyder, L. V. (2006). Facility location under uncertainty: a review. *IIE Transactions*, 38, 547–564. URL: <https://doi.org/10.1080/07408170500216480>. doi: 10.1080/07408170500216480. arXiv:<https://doi.org/10.1080/07408170500216480>.
- Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35, 254–265. URL: <http://www.jstor.org/stable/170697>.
- Sörensen, K., & Glover, F. W. (2013). Metaheuristics. In S. I. Gass, & M. C. Fu (Eds.), *Encyclopedia of Operations Research and Management Science* (pp. 960–970). Boston, MA: Springer US. URL: https://doi.org/10.1007/978-1-4419-1153-7_1167. doi: 10.1007/978-1-4419-1153-7_1167.
- Sörensen, K., Sevaux, M., & Glover, F. (2018). A history of metaheuristics. In R. Martí, P. M. Pardalos, & M. G. C. Resende (Eds.), *Handbook of Heuristics* (pp. 791–808). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-07124-4_4. doi: 10.1007/978-3-319-07124-4_4.
- Soyster, A. L. (1973). Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21, 1154–1157. URL: <http://www.jstor.org/stable/168933>.

- Sridharan, R. (1991). A lagrangian heuristic for the capacitated plant location problem with side constraints. *Journal of the Operational Research Society*, 42, 579–585. doi: 10.1057/jors.1991.117.
- Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. John Wiley & Sons. doi: 10.1002/9780470496916.
- Tirkolaei, E. B., Mahdavi, I., Esfahani, M. M. S., & Weber, G.-W. (2020). A robust green location-allocation-inventory problem to design an urban waste management system under uncertainty. *Waste Management*, 102, 340–350. doi: <https://doi.org/10.1016/j.wasman.2019.10.038>.
- Tragantalerungsak, S., Holt, J., & Rönnqvist, M. (2000). An exact method for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 123, 473–489. doi: 10.1016/S0377-2217(99)00105-8.
- Tran, T. H., Scaparra, M. P., & O’Hanley, J. R. (2017). A hypergraph multi-exchange heuristic for the single-source capacitated facility location problem. *European Journal of Operational Research*, 263, 173–187. doi: 10.1016/j.ejor.2017.04.032.
- Vaessens, R., Aarts, E., & Lenstra, J. (1998). A local search template. *Computers & Operations Research*, 25, 969–979. URL: <https://www.sciencedirect.com/science/article/pii/S0305054897000932>. doi: [https://doi.org/10.1016/S0305-0548\(97\)00093-2](https://doi.org/10.1016/S0305-0548(97)00093-2).
- Voudouris, C., & Tsang, E. (1999). Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113, 469–499. URL: <https://www.sciencedirect.com/science/article/pii/S037722179800099X>. doi: [https://doi.org/10.1016/S0377-2217\(98\)00099-X](https://doi.org/10.1016/S0377-2217(98)00099-X).
- Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1, 80–83. URL: <http://www.jstor.org/stable/3001968>. doi: <https://doi.org/10.2307/3001968>.
- Xu, W., Luis, M., & Yuce, B. (2025). Solving the single-source capacitated facility location problem using a crow search-based hybrid method. *IEEE Access*, 13, 82847–82859. doi: 10.1109/ACCESS.2025.3565153.
- Yang, Z., Chen, H., Chu, F., & Wang, N. (2019). An effective hybrid approach to the two-stage capacitated facility location problem. *European Journal of Operational Research*, 275, 467–480.

Yang, Z., Chu, F., & Chen, H. (2012). A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research*, 221, 521–532. doi: 10.1016/j.ejor.2012.03.04.

Apêndice A

A.1 Resultados detalhados dos experimentos com o algoritmo HILS

Tabela A.1: Resultados detalhados do HILS no conjunto TB4.

Instância	f	Δ	t
capa1	19241056,93	0,0000	495,3171
capa2	18438329,78	0,0000	368,7422
capa3	17765201,95	0,0000	136,6790
capa4	17160612,23	0,0000	53,5671
capb1	13657464,23	0,0000	208,8847
capb2	13362529,34	0,0000	206,7318
capb3	13199213,19	0,0000	167,8539
capb4	13083203,74	0,0000	170,0019
capc1	11647410,50	0,0000	152,9768
capc2	11570437,68	0,0000	111,2675
capc3	11519169,78	0,0000	91,4229
capc4	11505861,86	0,0000	49,1745

Tabela A.2: Resultados detalhados do HILS no conjunto TB5

Instância	LB1	LB2	f_{best}	$\bar{f}$	Δ_1	Δ_2	t
i300_1	16350,66	16552,75	16555,77	16559,76	1,28	0,04	2060,70
i300_2	15948,44	16059,34	16138,88	16142,32	1,22	0,52	2088,30
i300_3	15474,84	15606,1	15686,29	15700,64	1,46	0,61	2067,19
i300_4	17989,97	18143,89	18215,87	18224,04	1,30	0,44	2074,28
i300_5	18037,61	18191,11	18288,06	18314,26	1,53	0,68	2073,87
i300_6	11251,19	11271,22	11326,43	11326,43	0,67	0,49	643,32
i300_7	11392,52	11461,16	11470,31	11470,31	0,68	0,08	1065,33
i300_8	11449,67	11449,67	11449,67	11449,67	0,00	0,00	182,20
i300_9	10932,88	10932,88	10932,88	10932,88	0,00	0,00	564,10
i300_10	11232,77	11324,34	11324,34	11324,34	0,82	0,00	305,26
i300_11	10046,94	10046,94	10046,94	10046,94	0,00	0,00	71,37
i300_12	9359,64	9359,64	9359,64	9359,64	0,00	0,00	89,16
i300_13	10103,49	10103,49	10103,49	10103,49	0,00	0,00	557,53
i300_14	9738,05	9738,05	9738,05	9738,05	0,00	0,00	168,01
i300_15	9902,26	9902,26	9902,26	9902,26	0,00	0,00	109,33
i300_16	9168,08	9168,08	9168,08	9168,08	0,00	0,00	47,04
i300_17	9181,07	9181,07	9181,07	9181,07	0,00	0,00	45,18
i300_18	9581,95	9581,95	9581,95	9581,95	0,00	0,00	70,82
i300_19	9062,16	9062,16	9062,16	9062,16	0,00	0,00	53,35
i300_20	9078,22	9077,85	9077,85	9077,85	0,00	0,00	34,19
i3001500_1	154999,14	154999,14	154999,14	154999,14	0,00	0,00	167,08
i3001500_2	159438,60	159438,03	159438,03	159438,03	0,00	0,00	141,35
i3001500_3	157300,15	157300,15	157300,15	157300,15	0,00	0,00	217,10

Continua na próxima página

Continuação da Tabela A.2

Instância	LB1	LB2	f_{best}	$\bar{f}$	Δ_1	Δ_2	t
i3001500_4	157796,28	157796,28	157796,28	157796,28	0,00	0,00	227,56
i3001500_5	161306,00	161305,97	161305,97	161305,97	0,00	0,00	167,39
i3001500_6	156669,28	156667,31	156667,31	156667,31	0,00	0,00	99,37
i3001500_7	157031,55	157031,55	157031,55	157031,55	0,00	0,00	110,42
i3001500_8	157802,83	157796,21	157796,21	157796,21	0,00	0,00	126,66
i3001500_9	156968,46	156968,46	156968,46	156968,46	0,00	0,00	105,12
i3001500_10	157764,39	157757,62	157757,62	157757,62	0,00	0,00	104,58
i3001500_11	150015,13	150015,13	150015,13	150015,13	0,00	0,00	127,81
i3001500_12	154937,67	154937,67	154937,67	154937,67	0,00	0,00	124,64
i3001500_13	151608,42	151608,42	151608,42	151608,42	0,00	0,00	149,75
i3001500_14	151848,05	151848,05	151848,05	151848,05	0,00	0,00	96,16
i3001500_15	156480,89	156480,89	156480,89	156480,89	0,00	0,00	152,14
i3001500_16	155495,62	155493,77	155493,77	155493,77	0,00	0,00	90,00
i3001500_17	156038,04	156038,04	156038,04	156038,04	0,00	0,00	103,31
i3001500_18	156799,93	156790,75	156790,75	156790,75	-0,01	0,00	122,98
i3001500_19	155947,13	155947,13	155947,13	155947,13	0,00	0,00	95,05
i3001500_20	156426,14	156426,14	156426,14	156426,14	0,00	0,00	122,98
i500_1	26412,41	26566,69	26774,79	26796,49	1,45	0,86	4110,22
i500_2	28130,74	28268,41	28589,85	28613,85	1,72	1,22	4186,64
i500_3	27904,51	28067,69	28275,24	28292,10	1,39	0,80	3317,50
i500_4	28159,09	28268,36	28568,65	28622,97	1,65	1,25	4334,35
i500_5	24702,77	24805,56	24969,40	25018,14	1,28	0,86	3407,45
i500_6	15756,82	15842,27	15853,72	15863,65	0,68	0,13	2546,63
i500_7	16109,28	16163,11	16205,15	16205,15	0,60	0,26	2377,03
i500_8	16041,73	16081,54	16126,43	16126,43	0,53	0,28	2423,36
i500_9	16327,71	16346,25	16418,30	16419,98	0,57	0,45	2419,95
i500_10	15815,13	15857,97	15886,54	15886,54	0,45	0,18	3418,29
i500_11	13497,71	13497,71	13497,71	13497,71	0,00	0,00	483,74
i500_12	14675,02	14736,38	14750,13	14750,82	0,52	0,10	2293,16
i500_13	13666,25	13709,76	13715,96	13717,42	0,37	0,06	2244,14
i500_14	13629,54	13629,54	13629,54	13629,54	0,00	0,00	768,81
i500_15	13896,76	13940,08	13947,12	13947,12	0,36	0,05	2322,82
i500_16	12618,68	12618,68	12618,68	12618,68	0,00	0,00	656,78
i500_17	13386,17	13386,17	13389,69	13389,91	0,03	0,03	858,47
i500_18	12852,52	12852,52	12852,52	12852,52	0,00	0,00	296,01
i500_19	13521,52	13521,52	13521,52	13521,52	0,00	0,00	593,29
i500_20	12362,26	12362,2	12362,20	12362,20	0,00	0,00	217,77
i700_1	36905,93	37054,6	37474,45	37499,09	1,61	1,20	11525,07
i700_2	34311,71	34488,56	34899,95	34913,25	1,75	1,23	8114,25
i700_3	34294,63	34485,24	34903,76	34925,42	1,84	1,28	9291,44
i700_4	38090,90	38260,98	38777,67	38804,42	1,87	1,42	8264,87
i700_5	37802,10	37950,49	38384,32	38407,56	1,60	1,20	7362,83
i700_6	19910,67	19881,11	20037,10	20069,19	0,80	0,95	3967,66
i700_7	21297,30	21295,25	21436,12	21442,83	0,68	0,69	4581,73
i700_8	20659,96	20702,95	20798,33	20798,64	0,67	0,46	4213,99
i700_9	20979,88	20976,65	21114,32	21126,75	0,70	0,72	4428,14
i700_10	22055,41	22039,4	22219,51	22244,24	0,86	0,93	3854,70
i700_11	17120,15	17105,77	17192,42	17223,74	0,61	0,69	3619,69
i700_12	18130,42	18135,97	18222,45	18245,95	0,64	0,61	2836,71
i700_13	17239,96	17277,92	17295,88	17307,83	0,39	0,17	2603,01
i700_14	17337,63	17374,91	17383,87	17384,81	0,27	0,06	2504,83
i700_15	18145,49	18167,98	18222,15	18230,74	0,47	0,35	2716,34
i700_16	16029,55	16029,55	16029,55	16032,37	0,02	0,02	1574,57
i700_17	16199,55	16199,55	16199,55	16199,55	0,00	0,00	1169,08
i700_18	16443,07	16443,07	16443,07	16444,53	0,01	0,01	2058,33
i700_19	16399,79	16399,79	16399,79	16399,79	0,00	0,00	2394,10
i700_20	15434,21	15476,99	15520,41	15532,68	0,64	0,36	2788,44
i1000_1	49509,81	49681,02	50371,91	50387,05	1,77	1,42	12988,66
i1000_2	50688,57	50842,16	51525,90	51556,17	1,71	1,40	11377,37
i1000_3	47202,64	47362,62	47922,16	47958,86	1,60	1,26	11717,57
i1000_4	48868,54	49029,12	49792,86	49897,22	2,10	1,77	10949,60
i1000_5	50743,54	50971,44	51642,79	51686,51	1,86	1,40	11017,22
i1000_6	27823,84	27804,13	28146,38	28156,87	1,20	1,27	7644,37
i1000_7	27252,32	27210,23	27397,48	27439,65	0,69	0,84	6288,52
i1000_8	27375,37	27307,81	27546,78	27587,69	0,78	1,02	5793,65
i1000_9	26857,09	26816,77	27076,45	27089,92	0,87	1,02	7831,37
i1000_10	27186,99	27178,05	27388,50	27409,74	0,82	0,85	7725,24
i1000_11	22180,33	22117,85	22364,70	22381,11	0,91	1,19	4832,55
i1000_12	22160,39	22110,95	22245,60	22246,53	0,39	0,61	5566,72

Continua na próxima página

Continuação da Tabela A.2

Instância	LB1	LB2	f_{best}	$\bar{f}$	Δ_1	Δ_2	t
i1000_13	22657,09	22592,9	22822,78	22827,94	0,75	1,04	6692,55
i1000_14	22312,01	22273,28	22512,40	22518,45	0,93	1,10	6172,02
i1000_15	22629,44	22572,17	22805,38	22808,30	0,79	1,05	5724,93
i1000_16	21331,81	21322,81	21439,99	21472,62	0,66	0,70	5751,72
i1000_17	21188,89	21209,83	21297,19	21300,13	0,53	0,43	5055,86
i1000_18	20713,43	20739,2	20752,92	20752,92	0,19	0,07	3478,28
i1000_19	20537,45	20529,02	20643,17	20658,97	0,59	0,63	4921,24
i1000_20	21560,86	21541,51	21613,16	21630,05	0,32	0,41	4895,05

A.2 Resultados detalhados dos experimentos com o algoritmo NCS

Tabela A.3: Resultados detalhados do experimento com o NCS

Instância	BKV	BKLB	LS	LI	Opt?	t	LS ₀	LI ₀
				TB1				
p1	8848,00	8848,00	8848,00	8848,00	1	0,01	8848,00	8848,00
p2	7913,00	7913,00	7913,00	7913,00	1	0,05	7913,00	7913,00
p3	9314,00	9314,00	9314,00	9314,00	1	0,01	9314,00	9314,00
p4	10714,00	10714,00	10714,00	10714,00	1	0,09	10714,00	10700,57
p5	8838,00	8838,00	8838,00	8838,00	1	0,02	8838,00	8837,50
p6	7777,00	7777,00	7777,00	7777,00	1	0,01	7777,00	7777,00
p7	9488,00	9488,00	9488,00	9488,00	1	0,04	9488,00	9477,82
p8	11088,00	11088,00	11088,00	11088,00	1	0,09	11088,00	11087,50
p9	8462,00	8462,00	8462,00	8462,00	1	0,02	8462,00	8462,00
p10	7617,00	7617,00	7617,00	7617,00	1	0,02	7626,00	7613,03
p11	8932,00	8932,00	8932,00	8932,00	1	0,01	8932,00	8932,00
p12	10132,00	10132,00	10132,00	10132,00	1	0,03	10132,00	10132,00
p13	8252,00	8252,00	8252,00	8252,00	1	0,02	8252,00	8252,00
p14	7137,00	7137,00	7137,00	7137,00	1	0,01	7137,00	7137,00
p15	8808,00	8808,00	8808,00	8808,00	1	0,01	8808,00	8808,00
p16	10408,00	10408,00	10408,00	10408,00	1	0,04	10408,00	10408,00
p17	8227,00	8227,00	8227,00	8227,00	1	0,02	8227,00	8227,00
p18	7125,00	7125,00	7125,00	7125,00	1	0,01	7125,00	7125,00
p19	8886,00	8886,00	8886,00	8886,00	1	0,12	8907,00	8879,44
p20	10486,00	10486,00	10486,00	10486,00	1	0,12	10507,00	10479,44
p21	8068,00	8068,00	8068,00	8068,00	1	0,01	8068,00	8068,00
p22	7092,00	7092,00	7092,00	7092,00	1	0,01	7092,00	7092,00
p23	8746,00	8746,00	8746,00	8746,00	1	0,03	8746,00	8746,00
p24	10273,00	10273,00	10273,00	10273,00	1	0,06	10273,00	10273,00
p25	11630,00	11630,00	11630,00	11630,00	1	0,54	11630,00	11613,64
p26	10771,00	10771,00	10771,00	10771,00	1	0,25	10771,00	10750,21
p27	12322,00	12322,00	12322,00	12322,00	1	0,50	12322,00	12298,80
p28	13722,00	13722,00	13722,00	13722,00	1	0,57	13722,00	13702,25
p29	12371,00	12371,00	12371,00	12371,00	1	1,95	12371,00	12362,63
p30	11331,00	11331,00	11331,00	11331,00	1	1,88	11331,00	11297,79
p31	13331,00	13331,00	13331,00	13331,00	1	1,79	13331,00	13299,67
p32	15331,00	15331,00	15331,00	15331,00	1	2,19	15331,00	15301,17
p33	11629,00	11629,00	11629,00	11629,00	1	0,43	11629,00	11627,82
p34	10632,00	10632,00	10632,00	10632,00	1	0,08	10632,00	10632,00
p35	12232,00	12232,00	12232,00	12232,00	1	0,22	12232,00	12231,62
p36	13832,00	13832,00	13832,00	13832,00	1	0,31	13832,00	13831,60
p37	11258,00	11258,00	11258,00	11258,00	1	0,07	11258,00	11258,00
p38	10551,00	10551,00	10551,00	10551,00	1	0,05	10551,00	10551,00
p39	11824,00	11824,00	11824,00	11824,00	1	0,04	11824,00	11824,00
p40	13024,00	13024,00	13024,00	13024,00	1	0,04	13024,00	13024,00
p41	6589,00	6589,00	6589,00	6589,00	1	0,13	6589,00	6585,00
p42	5663,00	5663,00	5663,00	5663,00	1	0,23	5663,00	5660,18
p43	5214,00	5214,00	5214,00	5214,00	1	0,04	5214,00	5214,00
p44	7028,00	7028,00	7028,00	7028,00	1	0,07	7028,00	7028,00
p45	6251,00	6251,00	6251,00	6251,00	1	0,03	6251,00	6251,00
p46	5651,00	5651,00	5651,00	5651,00	1	0,14	5651,00	5651,00
p47	6228,00	6228,00	6228,00	6228,00	1	0,03	6228,00	6228,00
p48	5596,00	5596,00	5596,00	5596,00	1	0,03	5596,00	5596,00

Continua na próxima página

Continuação da Tabela A.3

Instância	BKV	BKLB	LS	LI	Opt?	t	LS ₀	LI ₀
p49	5302,00	5302,00	5302,00	5302,00	1	0,03	5302,00	5302,00
p50	8741,00	8741,00	8741,00	8741,00	1	0,25	8741,00	8710,95
p51	7414,00	7414,00	7414,00	7414,00	1	0,60	7414,00	7362,88
p52	9178,00	9178,00	9178,00	9178,00	1	0,09	9178,00	9175,88
p53	8531,00	8531,00	8531,00	8531,00	1	0,03	8531,00	8531,00
p54	8777,00	8777,00	8777,00	8777,00	1	0,10	8777,00	8773,80
p55	7654,00	7654,00	7654,00	7654,00	1	0,14	7654,00	7640,10
p56	21103,00	21103,00	21103,00	21103,00	1	4,97	21120,00	21073,00
p57	26039,00	26039,00	26039,00	26039,00	1	11,25	26045,00	25966,89
p58	37239,00	37239,00	37239,00	37239,00	1	19,47	37245,00	37146,84
p59	27282,00	27282,00	27282,00	27282,00	1	3,76	27282,00	27262,47
p60	20534,00	20534,00	20534,00	20534,00	1	0,03	20534,00	20534,00
p61	24454,00	24454,00	24454,00	24454,00	1	0,05	24454,00	24454,00
p62	32643,00	32643,00	32643,00	32643,00	1	1,76	32643,00	32632,50
p63	25105,00	25105,00	25105,00	25105,00	1	1,05	25105,00	25103,41
p64	20530,00	20530,00	20530,00	20530,00	1	0,04	20530,00	20530,00
p65	24445,00	24445,00	24445,00	24445,00	1	0,08	24445,00	24445,00
p66	31415,00	31415,00	31415,00	31415,00	1	1,63	31415,00	31379,66
p67	24848,00	24848,00	24848,00	24848,00	1	0,72	24848,00	24829,30
p68	20538,00	20538,00	20538,00	20538,00	1	0,03	20538,00	20538,00
p69	24532,00	24532,00	24532,00	24532,00	1	0,74	24532,00	24532,00
p70	32321,00	32321,00	32321,00	32321,00	1	1,01	32321,00	32283,13
p71	25540,00	25540,00	25540,00	25540,00	1	0,96	25540,00	25540,00
TB2								
y30-200-1	30181,00	30181,00	30181,00	30181,00	1	24,49	30181,00	30132,04
y30-200-2	28923,00	28923,00	28923,00	28923,00	1	20,86	28923,00	28884,15
y30-200-3	28131,00	28131,00	28131,00	28131,00	1	4,58	28131,00	28115,26
y30-200-4	28152,00	28152,00	28152,00	28152,00	1	10,71	28152,00	28136,63
y30-200-5	27646,00	27646,00	27646,00	27646,00	1	4,45	27646,00	27642,46
y60-200-1	27977,00	27977,00	27977,00	27977,00	1	10,19	27977,00	27926,73
y60-200-2	29704,00	29704,00	29704,00	29704,00	1	108,43	29704,00	29635,26
y60-200-3	27993,00	27993,00	27993,00	27993,00	1	17,68	27993,00	27977,21
y60-200-4	27691,00	27691,00	27691,00	27691,00	1	19,30	27693,00	27650,67
y60-200-5	29195,00	29195,00	29195,00	29195,00	1	38,04	29195,00	29153,97
y60-300-1	35648,00	35648,00	35648,00	35648,00	1	21,21	35648,00	35609,32
y60-300-2	35474,00	35474,00	35474,00	35474,00	1	16,18	35474,00	35461,53
y60-300-3	33872,00	33872,00	33872,00	33872,00	1	40,76	33872,00	33852,17
y60-300-4	33096,00	33096,00	33096,00	33096,00	1	8,72	33096,00	33084,02
y60-300-5	30918,00	30918,00	30918,00	30918,00	1	18,57	30918,00	30909,00
y80-400-1	39318,00	39318,00	39318,00	39318,00	1	37,86	39318,00	39296,82
y80-400-2	37076,00	37076,00	37076,00	37076,00	1	38,24	37076,00	37064,00
y80-400-3	43859,00	43859,00	43859,00	43859,00	1	44,51	43859,00	43827,52
y80-400-4	37344,00	37344,00	37344,00	37344,00	1	23,62	37344,00	37331,95
y80-400-5	43508,00	43508,00	43508,00	43508,00	1	53,73	43508,00	43472,09
TB3								
g50-100-2-1	18294,00	18294,00	18294,00	18294,00	1	16,34	18294,00	18290,14
g50-100-2-2	19688,00	19688,00	19688,00	19688,00	1	265,98	19688,00	19682,69
g50-100-2-3	19075,00	19075,00	19075,00	19075,00	1	41,89	19075,00	19072,43
g50-100-2-4	18620,00	18620,00	18620,00	18620,00	1	18,81	18620,00	18616,61
g50-100-2-5	18502,00	18502,00	18502,00	18502,00	1	198,64	18502,00	18496,77
g50-100-3-1	16948,00	16948,00	16948,00	16948,00	1	70,20	16948,00	16941,99
g50-100-3-2	15063,00	15063,00	15063,00	15063,00	1	1,95	15063,00	15059,86
g50-100-3-3	15107,00	15107,00	15107,00	15107,00	1	15,48	15107,00	15103,46
g50-100-3-4	14347,00	14347,00	14347,00	14347,00	1	7,90	14348,00	14343,50
g50-100-3-5	14813,00	14813,00	14813,00	14813,00	1	20,71	14813,00	14808,34
g50-100-5-1	12072,00	12072,00	12072,00	12072,00	1	1,32	12072,00	12070,18
g50-100-5-2	11898,00	11898,00	11898,00	11898,00	1	4,71	11898,00	11893,75
g50-100-5-3	11125,00	11125,00	11125,00	11125,00	1	1,41	11125,00	11123,82
g50-100-5-4	11817,00	11817,00	11817,00	11817,00	1	0,40	11817,00	11815,15
g50-100-5-5	11489,00	11489,00	11489,00	11489,00	1	0,67	11489,00	11486,00
g50-200-2-1	25992,00	25992,00	25992,00	25992,00	1	55,17	25992,00	25987,74
g50-200-2-2	25868,00	25868,00	25868,00	25868,00	1	2,65	25868,00	25865,75
g50-200-2-3	26930,00	26930,00	26930,00	26930,00	1	48,11	26930,00	26926,36
g50-200-2-4	25953,00	25953,00	25953,00	25953,00	1	24,58	25953,00	25949,75
g50-200-2-5	25328,00	25328,00	25328,00	25328,00	1	331,32	25328,00	25323,79
g50-200-3-1	20701,00	20701,00	20701,00	20701,00	1	75,25	20701,00	20696,79
g50-200-3-2	22021,00	22021,00	22021,00	22021,00	1	10,35	22021,00	22016,51
g50-200-3-3	20038,00	20038,00	20038,00	20038,00	1	1,74	20038,00	20036,15
g50-200-3-4	20595,00	20595,00	20595,00	20595,00	1	1,39	20595,00	20594,28

Continua na próxima página

Continuação da Tabela A.3

Instância	BKV	BKLB	LS	LI	Opt?	t	LS ₀	LI ₀
g50-200-3-5	21168,00	21168,00	21168,00	21168,00	1	1,73	21168,00	21166,04
g50-200-5-1	16659,00	16659,00	16659,00	16659,00	1	0,53	16659,00	16658,70
g50-200-5-2	16138,00	16138,00	16138,00	16138,00	1	1,30	16138,00	16136,61
g50-200-5-3	17755,00	17755,00	17755,00	17755,00	1	0,84	17755,00	17754,43
g50-200-5-4	15858,00	15858,00	15858,00	15858,00	1	1,47	15858,00	15855,73
g50-200-5-5	16884,00	16884,00	16884,00	16884,00	1	0,45	16884,00	16883,20
g60-300-2-1	34860,00	34860,00	34860,00	34860,00	1	169,34	34861,00	34857,75
g60-300-2-2	36549,00	36549,00	36549,00	36549,00	1	18982,61	36549,00	36542,86
g60-300-2-3	34878,00	34878,00	34878,00	34878,00	1	11,32	34878,00	34875,10
g60-300-2-4	34820,00	34820,00	34820,00	34820,00	1	366,83	34820,00	34815,82
g60-300-2-5	37140,00	37136,00	37140,00	37136,00	0	28800,00	37140,00	37135,61
g60-300-3-1	27903,00	27903,00	27903,00	27903,00	1	8,26	27903,00	27900,71
g60-300-3-2	27594,00	27594,00	27594,00	27594,00	1	1,67	27594,00	27593,39
g60-300-3-3	29231,00	29231,00	29231,00	29231,00	1	9,44	29231,00	29228,96
g60-300-3-4	27439,00	27439,00	27439,00	27439,00	1	1,58	27439,00	27437,29
g60-300-3-5	28033,00	28033,00	28033,00	28033,00	1	12,60	28033,00	28030,35
g60-300-5-1	21045,00	21045,00	21045,00	21045,00	1	8,75	21045,00	21043,10
g60-300-5-2	22589,00	22589,00	22589,00	22589,00	1	9,49	22589,00	22587,55
g60-300-5-3	21449,00	21449,00	21449,00	21449,00	1	3,95	21449,00	21447,50
g60-300-5-4	21466,00	21466,00	21466,00	21466,00	1	1,23	21466,00	21464,24
g60-300-5-5	21860,00	21860,00	21860,00	21860,00	1	4,60	21860,00	21859,03
TB4								
capa1	19242450,15	19240491,80	19242450,15	19240822,45	1	108,93	19242450,15	19240822,45
capa2	18438610,06	18437208,01	18438610,06	18438046,54	1	191,55	18438610,06	18438046,54
capa3	17765201,95	17765201,95	17765201,95	17765201,95	1	345,04	17765201,95	17765201,95
capa4	17160863,22	17160452,65	17160863,22	17160496,94	1	144,23	17160863,22	17160496,94
capb1	13657857,21	13656594,25	13657857,21	13657109,74	1	70,73	13657857,21	13656379,58
capb2	13362626,19	13361927,89	13362626,19	13362427,96	1	136,18	13362626,19	13361927,45
capb3	13199599,31	13198556,43	13199599,31	13198556,43	1	158,00	13199599,31	13198556,43
capb4	13083300,84	13082342,37	13083300,84	13082944,77	1	154,80	13083300,84	13082516,50
capc1	11647778,90	11646596,97	11647778,90	11646880,82	1	81,65	11647778,90	11646596,97
capc2	11570437,68	11569392,46	11570437,68	11570437,68	1	71,47	11570437,68	11570340,29
capc3	11519262,20	11518657,92	11519262,20	11518956,21	1	65,69	11519262,20	11518743,74
capc4	11505914,79	11505767,39	11505914,79	11505767,39	1	55,79	11505914,79	11505767,39
TB5								
i300_1	16555,77	16555,77	16555,77	16554,36	1	10529,15	16562,68	16517,87
i300_2	16135,82	16100,32	16138,37	16059,88	0	28800,00	16138,37	16059,88
i300_3	15649,41	15627,71	15649,41	15593,67	0	28800,00	15652,39	15593,67
i300_4	18239,87	18187,84	18239,87	18159,25	0	28800,00	18242,71	18159,25
i300_5	18291,05	18247,54	18297,82	18193,36	0	28800,00	18307,10	18193,36
i300_6	11326,43	11326,43	11326,43	11325,30	1	3486,52	11346,75	11291,90
i300_7	11470,31	11470,31	11470,31	11469,37	1	5879,97	11488,06	11432,21
i300_8	11449,67	11449,67	11449,67	11449,67	1	102,23	11449,67	11430,92
i300_9	10932,88	10932,88	10932,88	10931,99	1	266,95	10932,88	10912,98
i300_10	11324,34	11324,34	11324,34	11323,41	1	180,77	11324,34	11292,43
i300_11	10046,94	10046,94	10046,94	10045,94	1	26,41	10050,50	10033,79
i300_12	9359,64	9359,64	9359,64	9358,85	1	74,02	9359,64	9346,13
i300_13	10103,49	10103,49	10103,49	10103,35	1	600,48	10104,57	10085,13
i300_14	9738,05	9738,05	9738,05	9738,05	1	91,05	9738,05	9723,65
i300_15	9902,26	9902,26	9902,26	9902,26	1	105,27	9902,26	9884,51
i300_16	9168,08	9168,08	9168,08	9167,69	1	19,10	9170,49	9163,88
i300_17	9181,07	9181,07	9181,07	9180,29	1	12,15	9181,07	9177,85
i300_18	9581,95	9581,95	9581,95	9581,46	1	20,83	9581,95	9575,12
i300_19	9062,16	9062,16	9062,16	9062,05	1	14,71	9062,16	9060,65
i300_20	9077,85	9077,85	9077,85	9077,27	1	13,49	9077,85	9070,43
i3001500_1	154999,14	154999,14	154999,14	154995,28	1	242,06	155000,71	154966,39
i3001500_2	159438,03	159438,60	159438,03	159437,06	1	230,24	159446,39	159413,81
i3001500_3	157300,15	157300,15	157300,15	157293,81	1	487,48	157300,15	157266,89
i3001500_4	157796,28	157796,28	157796,28	157788,71	1	1151,86	157801,19	157749,70
i3001500_5	161305,97	161305,97	161305,97	161305,97	1	94,89	161306,00	161284,82
i3001500_6	156667,31	156667,31	156667,31	156659,63	1	50,42	156667,31	156651,25
i3001500_7	157031,55	157032,17	157041,57	157032,17	1	100,96	157041,57	157018,54
i3001500_8	157796,21	157796,21	157796,21	157796,21	1	50,05	157796,21	157779,15
i3001500_9	156968,46	156968,46	156968,46	156958,44	1	30,59	156968,46	156966,76
i3001500_10	157757,62	157757,62	157757,62	157744,20	1	36,66	157757,62	157748,11
i3001500_11	150015,13	150015,13	150015,13	150015,13	1	22,26	150015,13	150015,13
i3001500_12	154937,67	154937,67	154937,67	154937,67	1	460,50	154937,67	154904,39
i3001500_13	151608,42	151608,42	151608,42	151605,60	1	26,21	151608,42	151607,65
i3001500_14	151848,05	151848,05	151848,05	151843,84	1	27,83	151848,05	151842,51

Continua na próxima página

Continuação da Tabela A.3

Instância	BKV	BKLB	LS	LI	Opt?	t	LS ₀	LI ₀
i3001500_15	156480,89	156480,89	156483,12	156475,91	1	255,53	156483,12	156445,14
i3001500_16	155493,77	155493,77	155493,77	155493,77	1	23,63	155493,77	155493,77
i3001500_17	156038,04	156038,04	156038,04	156038,04	1	31,15	156038,04	156038,04
i3001500_18	156790,75	156791,41	156791,41	156791,41	1	28,52	156791,41	156786,73
i3001500_19	155947,13	155947,13	155947,13	155944,83	1	26,51	155947,13	155947,13
i3001500_20	156426,14	156426,14	156428,88	156423,89	1	27,82	156428,88	156423,78
i500_1	26698,33	26613,52	26698,33	26613,52	0	28800,00	26698,33	26613,52
i500_2	28431,44	28322,31	28431,44	28322,31	0	28800,00	28481,80	28322,31
i500_3	28212,43	28108,98	28212,43	28108,98	0	28800,00	28224,65	28108,98
i500_4	28450,80	28330,03	28450,80	28330,03	0	28800,00	28450,80	28330,03
i500_5	24920,99	24858,60	24920,99	24858,60	0	28800,00	24920,99	24858,60
i500_6	15853,35	15853,35	15853,35	15852,13	1	5576,24	15853,35	15820,81
i500_7	16205,15	16205,15	16205,15	16203,53	1	13448,96	16220,84	16160,30
i500_8	16115,43	16103,41	16115,43	16075,39	0	28800,00	16143,77	16075,39
i500_9	16399,40	16399,40	16399,40	16371,21	0	28800,00	16409,14	16371,21
i500_10	15886,54	15884,95	15886,54	15884,95	1	4049,65	15886,54	15852,99
i500_11	13497,71	13497,71	13497,71	13496,40	1	971,47	13506,12	13482,99
i500_12	14736,38	14736,38	14736,38	14735,02	1	1732,38	14736,74	14707,01
i500_13	13715,96	13715,96	13715,96	13715,35	1	4553,99	13722,57	13687,95
i500_14	13629,54	13629,54	13629,54	13628,64	1	2237,45	13634,90	13610,57
i500_15	13947,12	13947,12	13947,12	13945,73	1	725,58	13947,38	13923,32
i500_16	12618,68	12618,68	12618,68	12617,49	1	547,82	12623,12	12600,51
i500_17	13386,17	13386,17	13386,17	13384,84	1	1623,57	13386,17	13363,00
i500_18	12852,52	12852,52	12852,52	12851,24	1	166,47	12852,52	12844,96
i500_19	13521,52	13521,52	13521,52	13520,28	1	1370,52	13528,48	13504,67
i500_20	12362,20	12362,20	12362,20	12361,11	1	72,70	12362,20	12357,33
i700_1	37302,35	37145,20	37302,35	37145,20	0	28800,00	37526,91	37145,20
i700_2	34783,64	34564,27	34783,64	34564,27	0	28800,00	34829,77	34564,27
i700_3	34739,12	34556,67	34739,12	34556,67	0	28800,00	34797,96	34556,67
i700_4	38534,35	38317,15	38622,27	38306,15	0	28800,00	38666,34	38306,15
i700_5	38158,53	38029,67	38158,53	38029,67	0	28800,00	38158,53	38029,67
i700_6	20051,28	19971,35	20051,28	19971,35	0	28800,00	20059,11	19971,35
i700_7	21433,56	21366,16	21436,26	21359,44	0	28800,00	21436,26	21359,44
i700_8	20816,45	20750,00	20816,45	20747,16	0	28800,00	20818,55	20747,16
i700_9	21090,04	21041,21	21090,04	21029,06	0	28800,00	21090,04	21029,06
i700_10	22188,50	22136,55	22188,50	22136,55	0	28800,00	22188,50	22136,55
i700_11	17180,27	17173,19	17180,27	17145,94	0	28800,00	17204,97	17145,94
i700_12	18194,03	18179,99	18194,03	18156,82	0	28800,00	18194,03	18156,82
i700_13	17284,72	17277,92	17284,77	17283,27	1	10604,94	17293,27	17259,98
i700_14	17383,87	17383,87	17383,87	17382,14	1	19737,60	17385,11	17361,13
i700_15	18218,21	18208,97	18218,21	18178,54	0	28800,00	18218,21	18178,54
i700_16	16029,55	16029,55	16029,55	16028,34	1	636,07	16029,55	16015,59
i700_17	16199,55	16199,55	16199,55	16197,94	1	477,01	16199,55	16190,01
i700_18	16443,07	16443,07	16443,07	16441,44	1	389,64	16445,29	16432,70
i700_19	16399,79	16399,79	16399,79	16398,26	1	1489,87	16399,79	16389,05
i700_20	15492,02	15492,02	15492,02	15490,62	1	9614,38	15492,02	15465,23
i1000_1	50051,78	49744,73	50049,24	49738,94	0	28800,00	50049,24	49738,94
i1000_2	51277,80	50904,05	52521,77	50904,05	0	28800,00	52521,77	50904,05
i1000_3	47690,07	47449,49	47701,98	47449,49	0	28800,00	47749,30	47449,49
i1000_4	49408,86	49100,11	49425,26	49100,11	0	28800,00	49425,26	49100,11
i1000_5	51363,39	51033,98	51408,93	51033,98	0	28800,00	51408,93	51033,98
i1000_6	28043,73	27906,29	28048,89	27906,29	0	28800,00	28057,76	27906,29
i1000_7	27409,44	27316,76	27409,44	27316,76	0	28800,00	27409,44	27316,76
i1000_8	27543,01	27440,91	27673,87	27440,91	0	28800,00	27885,78	27440,91
i1000_9	26992,81	26867,37	27090,08	26892,84	0	28800,00	27120,20	26892,84
i1000_10	27364,31	27266,53	27364,31	27266,53	0	28800,00	27364,31	27266,53
i1000_11	22247,61	22205,27	22248,93	22205,27	0	28800,00	22252,04	22205,27
i1000_12	22226,96	22197,34	22226,96	22184,51	0	28800,00	22244,27	22184,51
i1000_13	22728,86	22670,58	22728,86	22670,58	0	28800,00	22736,53	22670,58
i1000_14	22401,92	22349,59	22401,92	22349,59	0	28800,00	22429,97	22349,59
i1000_15	22687,95	22660,72	22687,95	22653,20	0	28800,00	22687,95	22653,20
i1000_16	21386,85	21372,31	21386,85	21359,38	0	28800,00	21386,85	21359,38
i1000_17	21233,77	21231,64	21233,77	21231,64	1	8427,62	21234,27	21217,44
i1000_18	20752,92	20750,88	20752,92	20750,88	1	5490,11	20768,15	20736,04
i1000_19	20591,21	20577,58	20591,21	20560,86	0	28800,00	20591,21	20560,86
i1000_20	21598,54	21597,95	21598,54	21582,72	0	28800,00	21599,35	21574,49

A.3 Resultados detalhados dos experimentos com o algoritmo H-ALG

Tabela A.4: Resultados detalhados do experimento com o H-ALG

Instância	LS	LI	gap%	t	LI ₀	LS ₀
			TB1			
p1	9933,00	9932,19	0,00	0,66	9736,03	8726,00
p2	8622,00	8621,49	0,00	0,28	8538,37	10650,00
p3	10590,00	10589,06	0,00	1,42	10298,50	12216,00
p4	12181,00	12180,16	0,00	1,21	11875,50	10282,00
p5	10012,00	10012,00	0,00	0,46	9870,59	8727,00
p6	8652,00	8652,00	0,00	0,20	8591,83	10772,00
p7	10772,00	10772,00	0,00	0,41	10600,00	13146,00
p8	12892,00	12891,23	0,00	2,01	12543,90	9028,00
p9	9028,00	9028,00	0,00	0,12	8996,00	7971,00
p10	7971,00	7970,13	0,00	0,12	7957,08	9605,00
p11	9605,00	9604,41	0,00	0,22	9529,66	11178,00
p12	11085,00	11084,76	0,00	0,45	10905,90	9412,00
p13	9377,00	9376,19	0,00	0,92	9117,54	7881,00
p14	7817,00	7817,00	0,00	0,41	7698,54	10000,00
p15	9937,00	9936,43	0,00	2,39	9722,03	12058,00
p16	11935,00	11935,00	0,00	9,59	11596,10	9250,00
p17	9249,00	9248,07	0,00	2,20	8877,28	7840,00
p18	7840,00	7840,00	0,00	0,47	7655,99	9960,00
p19	9960,00	9960,00	0,00	1,22	9664,76	11940,00
p20	11940,00	11939,12	0,00	10,87	11456,50	8622,00
p21	8622,00	8622,00	0,00	0,22	8573,95	7486,00
p22	7486,00	7486,00	0,00	0,16	7459,87	9295,00
p23	9275,00	9275,00	0,00	0,38	9190,68	10975,00
p24	10955,00	10955,00	0,00	0,55	10841,10	12708,00
p25	12671,00	12670,01	0,00	26,89	12519,40	10781,00
p26	10771,00	10770,24	0,00	13,47	10717,30	13318,00
p27	13293,00	13292,02	0,00	25,90	13142,60	15450,00
p28	15173,00	15172,01	0,00	94,11	14859,40	14091,00
p29	14033,00	14032,01	0,00	86,88	13804,20	12590,00
p30	12577,00	12576,03	0,00	139,11	12337,40	15345,00
p31	15337,00	15213,28	0,81	7201,34	14905,30	18116,00
p32	18097,00	17750,99	1,91	7201,57	17458,40	12769,00
p33	12759,00	12758,03	0,00	21,81	12598,00	11655,00
p34	11652,00	11652,00	0,00	39,00	11414,60	13736,00
p35	13732,00	13731,01	0,00	203,37	13315,00	15826,00
p36	15815,00	15653,60	1,02	7201,22	15202,00	12195,00
p37	12093,00	12092,03	0,00	18,18	11928,10	11091,00
p38	11091,00	11091,00	0,00	3,79	11064,60	12653,00
p39	12653,00	12652,07	0,00	12,40	12497,90	14133,00
p40	14133,00	14132,33	0,00	12,64	13829,90	7118,00
p41	7102,00	7101,08	0,00	0,94	6990,44	6261,00
p42	6220,00	6219,02	0,00	6,38	6001,30	5481,00
p43	5481,00	5481,00	0,00	1,91	5428,27	7648,00
p44	7648,00	7647,77	0,00	0,52	7601,02	6706,00
p45	6688,00	6688,00	0,00	1,37	6605,44	6155,00
p46	6155,00	6155,00	0,00	1,88	6025,59	7191,00
p47	7187,00	7186,10	0,00	5,49	7056,10	6073,00
p48	6073,00	6072,24	0,00	0,71	6019,37	5785,00
p49	5780,00	5779,65	0,00	1,63	5633,15	9612,00
p50	9551,00	9550,34	0,00	2,04	9444,13	8202,00
p51	8106,00	8105,32	0,00	4,32	7934,63	10852,00
p52	10575,00	10574,09	0,00	2,95	10404,70	9575,00
p53	9570,00	9569,16	0,00	2,53	9414,80	10217,00
p54	10144,00	10143,23	0,00	1,60	9984,63	8427,00
p55	8419,00	8418,08	0,00	3,41	8312,23	23458,00
p56	23150,00	23149,04	0,00	125,07	22727,40	29970,00
p57	29530,00	29472,94	0,19	7202,46	28723,50	44661,00
p58	44457,00	43394,85	2,39	7203,81	42408,80	32741,00
p59	32367,00	31982,74	1,19	7204,56	30912,40	21779,00
p60	21700,00	21699,12	0,00	8,56	21598,00	26176,00
p61	26176,00	26175,18	0,00	21,64	25936,30	36647,00
p62	36305,00	36304,01	0,00	754,55	35538,40	27200,00

Continua na próxima página

Continuação da Tabela A.2

Instância	LS	LI	gap%	t	LI ₀	LS ₀
p63	27075,00	27074,05	0,00	46,84	26761,20	21544,00
p64	21538,00	21537,29	0,00	5,01	21507,10	25676,00
p65	25676,00	25675,10	0,00	5,17	25613,30	34521,00
p66	34233,00	34232,01	0,00	515,66	33254,90	27701,00
p67	27223,00	27222,01	0,00	129,57	26591,20	21986,00
p68	21875,00	21875,00	0,00	33,48	21803,20	26586,00
p69	26561,00	26560,08	0,00	57,71	26272,80	36279,00
p70	36133,00	36132,01	0,00	930,18	35485,10	28625,00
p71	28580,00	28579,09	0,00	57,85	27763,00	28683
TB2						
y30-200-1	39240,00	37685,87	3,96	7206,28	36720,30	39398,00
y30-200-2	37257,00	35990,98	3,40	7206,28	35155,20	37369,00
y30-200-3	36194,00	35071,86	3,10	7204,38	34212,90	36315,00
y30-200-4	35616,00	34618,94	2,80	7203,01	33930,00	35722,00
y30-200-5	35843,00	34621,69	3,41	7205,87	33766,40	36019,00
y60-200-1	35718,00	34811,73	2,54	7225,40	33782,10	36165,00
y60-200-2	37936,00	37252,76	1,80	7221,15	36313,90	38448,00
y60-200-3	35821,00	34928,50	2,49	7211,03	34171,10	35872,00
y60-200-4	35618,00	35128,22	1,38	7213,32	34178,10	36439,00
y60-200-5	37441,00	36285,90	3,09	7219,78	35402,90	37648,00
y60-300-1	45047,00	44106,97	2,09	7219,61	43182,70	45941,00
y60-300-2	45681,00	44461,17	2,67	7223,94	43356,00	45958,00
y60-300-3	41932,00	41220,03	1,70	7250,73	40221,20	42722,00
y60-300-4	41105,00	40339,98	1,86	7233,43	39493,10	41192,00
y60-300-5	36912,00	36588,29	0,88	7218,71	35704,20	37257,00
y80-400-1	49224,00	48394,74	1,68	7344,32	47434,30	50453,00
y80-400-2	45375,00	44658,90	1,58	7346,14	43430,50	46529,00
y80-400-3	55217,00	54300,05	1,66	7267,19	53287,50	55884,00
y80-400-4	44698,00	44647,06	0,11	7273,61	43568,10	45730,00
y80-400-5	54984,00	54142,91	1,53	7304,99	53203,20	56446,00
TB3						
g50-100-2-1	25013,00	21479,84	14,13	7201,93	21338,60	25014,00
g50-100-2-2	26808,00	23062,23	13,97	7201,69	22937,00	26816,00
g50-100-2-3	26381,00	22344,32	15,30	7201,69	22194,10	26381,00
g50-100-2-4	25283,00	21864,29	13,52	7201,67	21675,60	25290,00
g50-100-2-5	25095,00	21387,90	14,77	7201,86	21268,60	25098,00
g50-100-3-1	22700,00	19583,01	13,73	7203,13	19286,20	22719,00
g50-100-3-2	19941,00	17597,81	11,75	7202,11	17272,00	19962,00
g50-100-3-3	20332,00	17862,93	12,14	7201,99	17530,50	20364,00
g50-100-3-4	19274,00	16687,96	13,42	7201,45	16471,70	19274,00
g50-100-3-5	19506,00	17292,07	11,35	7202,09	16987,40	19522,00
g50-100-5-1	16082,00	14314,92	10,99	7202,43	13994,60	16093,00
g50-100-5-2	16000,00	13935,71	12,90	7202,12	13663,00	16011,00
g50-100-5-3	14456,00	13038,54	9,81	7201,78	12625,10	14506,00
g50-100-5-4	15377,00	13856,29	9,89	7202,59	13591,40	15396,00
g50-100-5-5	15568,00	13787,44	11,44	7202,57	13416,20	15592,00
g50-200-2-1	35827,00	30742,44	14,19	7204,45	30518,40	35827,00
g50-200-2-2	34907,00	30194,36	13,50	7203,62	29886,50	34907,00
g50-200-2-3	36410,00	31186,69	14,35	7204,57	30942,70	36411,00
g50-200-2-4	35935,00	30766,40	14,38	7204,55	30561,90	35935,00
g50-200-2-5	34820,00	29921,66	14,07	7204,30	29716,20	34820,00
g50-200-3-1	27891,00	24216,67	13,17	7204,29	23906,50	27896,00
g50-200-3-2	29400,00	25639,59	12,79	7205,43	25356,00	29407,00
g50-200-3-3	26896,00	23680,81	11,95	7207,53	23372,50	26929,00
g50-200-3-4	27478,00	24320,54	11,49	7204,45	24018,00	27478,00
g50-200-3-5	28635,00	24971,45	12,79	7205,11	24689,00	28638,00
g50-200-5-1	22242,00	19899,38	10,53	7208,34	19519,20	22255,00
g50-200-5-2	21014,00	19061,31	9,29	7205,09	18704,50	21028,00
g50-200-5-3	22912,00	20615,00	10,03	7205,01	20299,90	22944,00
g50-200-5-4	20523,00	18590,09	9,42	7205,68	18212,00	20526,00
g50-200-5-5	22564,00	20282,05	10,11	7214,61	19971,80	22593,00
g60-300-2-1	46951,00	40652,11	13,42	7212,02	40434,80	46951,00
g60-300-2-2	49144,00	42733,33	13,04	7212,81	42481,80	49144,00
g60-300-2-3	47517,00	40882,81	13,96	7212,68	40678,20	47517,00
g60-300-2-4	46907,00	40840,20	12,93	7211,66	40567,80	46907,00
g60-300-2-5	50826,00	43478,06	14,46	7211,86	43285,00	50826,00
g60-300-3-1	37900,00	32989,10	12,96	7212,61	32693,30	37900,00
g60-300-3-2	36541,00	32265,83	11,70	7212,78	31937,80	36541,00
g60-300-3-3	39101,00	34384,97	12,06	7216,80	34042,50	39119,00

Continua na próxima página

Continuação da Tabela A.2

Instância	LS	LI	gap%	t	LI ₀	LS ₀
g60-300-3-4	36379,00	32357,07	11,06	7211,76	32012,00	36379,00
g60-300-3-5	37329,00	32688,96	12,43	7218,22	32389,10	37336,00
g60-300-5-1	28437,00	24896,62	12,45	7222,22	24524,80	28452,00
g60-300-5-2	30092,00	26013,56	13,55	7214,64	25627,00	30093,00
g60-300-5-3	28409,00	25137,31	11,52	7239,90	24820,40	28422,00
g60-300-5-4	28807,00	25596,31	11,15	7228,29	25283,70	29627,00
g60-300-5-5	28875,00	25838,55	10,52	7229,20	25411,40	28887,00
TB6						
d1	7006,00	7006,00	0,00	0,07	6846,06	7034,00
d2	13266,00	13265,36	0,00	0,35	11849,90	13480,00
d3	12954,00	12954,00	0,00	0,09	12793,40	12954,00
d4	16127,00	16126,50	0,00	1,38	14953,60	16774,00
d5	14766,00	14766,00	0,00	0,08	14357,60	14766,00
d6	10252,00	10251,11	0,00	0,81	8152,76	10255,00
d7	12711,00	12711,00	0,00	0,13	12527,90	12711,00
d8	18160,00	18160,00	0,00	0,45	17521,00	18416,00
d9	9993,00	9993,00	0,00	0,28	9703,49	10172,00
d10	40592,00	40591,08	0,00	10,56	37564,80	41140,00
d11	13163,00	13163,00	0,00	0,66	11906,90	14058,00
d12	12836,00	12836,00	0,00	0,30	12296,10	12836,00
d13	13987,00	13987,00	0,00	0,22	13506,60	14050,00
d14	18428,00	18427,20	0,00	1,25	16576,60	18518,00
d15	18861,00	18861,00	0,00	0,73	16520,70	21313,00
d16	24993,00	24993,00	0,00	1,72	22504,90	28353,00
d17	23518,00	23518,00	0,00	0,28	22998,00	23923,00
d18	30312,00	30311,80	0,00	11,50	28361,60	31410,00
d19	33992,00	33991,50	0,00	24,86	32204,40	34181,00
d20	46581,00	46580,01	0,00	2139,30	41808,20	48703,00
d21	23783,00	23782,04	0,00	13,18	19445,60	28039,00
d22	10954,00	10954,00	0,00	0,50	10362,30	11048,00
d23	17141,00	17140,06	0,00	2,14	15960,30	17333,00
d24	18026,00	18026,00	0,00	0,43	17652,60	18095,00
d25	20381,00	20381,00	0,00	1,50	19086,30	20441,00
d26	13790,00	13790,00	0,00	0,61	13457,20	13790,00
d27	25559,00	25558,62	0,00	1,11	24370,50	26031,00
d28	23901,00	23900,03	0,00	14,80	22339,80	24225,00
d29	24840,00	24839,06	0,00	1,63	23725,30	25340,00
d30	30116,00	25491,39	15,36	7200,00	210757,36	37720,00
d31	14944,00	14944,00	0,00	1,18	14464,40	15159,00
d32	22210,00	22209,26	0,00	3,54	21150,10	22774,00
d33	74629,00	72274,43	3,16	7200,00	62254,88	76673,00
d34	17458,00	17458,00	0,00	1,06	17111,80	17652,00
d35	16657,00	16657,00	0,00	3,57	16154,50	17454,00
d36	36067,00	36066,01	0,00	1209,52	33818,70	36676,00
d37	31196,00	31195,14	0,00	14,83	29950,40	32350,00
d38	83750,00	77900,92	6,98	7200,91	73861,30	86007,00
d39	76154,00	71450,78	6,18	7201,49	67346,60	77818,00
d40	105822,00	97376,66	7,98	7200,76	92353,30	108090,00
d41	41301,00	41301,00	0,00	1,75	39016,70	41562,00
d42	22956,00	22955,63	0,00	10,41	22114,20	23165,00
d43	18234,00	18234,00	0,00	0,07	18234,00	67247,00
d44	64816,00	59840,15	7,68	7201,75	57384,10	40224,00
d45	39514,00	39310,09	0,52	7201,04	37673,00	93933,00
d46	90520,00	88712,70	2,00	7201,04	82604,70	114158,00
d47	112886,00	106866,87	5,33	7201,64	102918,00	105563,00
d48	101043,00	98715,87	2,30	7201,25	95015,30	134131,00
d49	130272,00	123489,35	5,21	7201,14	118265,00	24894,00
d50	24886,00	24886,00	0,00	1,73	24655,00	26559,00
d51	25536,00	25535,63	0,00	4,33	25026,50	71211,00
d52	67687,00	63213,14	6,61	7201,77	60691,10	59882,00
d53	58123,00	56520,28	2,76	7201,23	54490,90	90174,00
d54	89627,00	85863,78	4,20	7201,48	81228,10	134080,00
d55	130073,00	115052,90	11,55	7201,27	111161,00	117812,00
d56	117410,00	114447,16	2,52	7200,82	107915,00	103999,00
d57	100650,00	99126,43	1,51	7200,94	92286,10	10022,00