ALGORITMOS EXATOS PARA O PROBLEMA DO CAMINHO MAIS CURTO ROBUSTO E PARA O PROBLEMA DE LOCALIZAÇÃO DE CONCENTRADORES EM ÁRVORE

JOÃO CARLOS ABREU JÚNIOR

ALGORITMOS EXATOS PARA O PROBLEMA DO CAMINHO MAIS CURTO ROBUSTO E PARA O PROBLEMA DE LOCALIZAÇÃO DE CONCENTRADORES EM ÁRVORE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Thiago Ferreira de Noronha
Coorientadora: Andréa Cynthia Santos

Belo Horizonte

Setembro de 2015

Abreu Júnior, João Carlos

A162a Algoritmos exatos para o problema do caminho mais curto robusto e para o problema de localização de concentradores em árvore / João Carlos Abreu Júnior. — Belo Horizonte, 2015
xxiv, 74 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Thiago Ferreira de Noronha
Coorientadora: Andréa Cynthia Santos

1. Computação - Teses. 2. Otimização Matemática - Teses. 3. Pesquisa Operacional - Teses. I. Orientador. II. Coorientadora. III. Título.

CDU 519.6*61 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Algoritmos exatos para o problema do caminho mais curto robusto e para o problema de localização de concentradores em árvore

JOÃO CARLOS ABREU JÚNIOR

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. THIAGO FERREIRA DE NORONHA - Orientador
Departamento de Ciência da Computação - UFMG

PROFA. ANDREA CYNTHIA SANTOS - Coorientadora
Université de Tecnologie de Troyes

PROF. RAFAEL CASTRO DE ANDRADE
Departamento de Estatística e Matemática Aplicada - UFC

PROF. SEBASTIAN ALBERTO URRUTIA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 14 de setembro de 2015.

Para o meu pai e minha irmã.

Agradecimentos

Agradeço primeiramente à Deus.

Agradeço ao meu pai João Carlos que sempre me ensinou a correr atrás e realizar os meus sonhos.

Agradeço a minha irmã Ana Paula que sempre me apoiou mesmo sem entender muito bem o que eu faço.

Agradeço a minha coorientadora Andréa Cynthia pelas constantes ajudas e reuniões e pelo tempo despendido para contribuir com minha formação.

Agradeço também ao meu orientador Thiago Noronha que mudou a minha idéia inicial e me fez entrar para o mestrado. Sou eternamente grato por tudo que você me ensinou. Muito obrigado pelas diversas horas de reuniões que você teve comigo para me ensinar, me corrigir e me orientar. Obrigado também pelas conversas não técnicas que tivemos, além de muito competente é sempre muito bom conversar com você. Além de meu orientador se tornou um grande amigo.

"Seja você quem for, seja qual for a posição social que você tenha na vida, a mais alta ou a mais baixa, tenha sempre como meta muita força, muita determinação e sempre faça tudo com muito amor e com muita fé em Deus, que um dia você chega lá. De alguma maneira você chega lá."

(Ayrton Senna)

Resumo

Este trabalho é dedicado ao estudo de dois problemas de otimização *NP-Difíceis*. O primeiro problema é o problema do caminho mais curto robusto (RSP, do inglês *Robust Shortest Path*) que é uma generalização do problema de caminho mais curto (SP, do inglês *Shortest Path*). O RSP considera que os custos dos arcos são definidos por um intervalo de valores contínuo. Entre os diferentes critérios de otimização robusta, esse trabalho se dedicada ao critério *minmax* com arrependimento relativo. Neste trabalho são dadas três contribuições para o RSP com arrependimento relativo: (i) a primeira formulação por programação linear inteira mista, (ii) desigualdades válidas para essa formulação e (iii) extensão desse problema em grafos com ciclos de custo positivo. Os resultados computacionais mostraram que os algoritmos baseados nas contribuições propostas são capazes de resolver instâncias de até 1500 nós.

O segundo problema que esse trabalho trata é o problema de localização de concentradores em árvore (THLP, do inglês *Tree Hub Location Problem*). Seja $G = (N, A)$ um grafo completo direcionado, onde N é o conjunto de nós e A é o conjunto de arcos. Além disto, $W_{ij} \in \mathbb{R}^*_+$ é a demanda de fluxo do nó $i \in N$ para o nó $j \in N$, c_{ij} é o custo de trafegar cada unidade de fluxo em um arco $(i, j) \in A$

e p um número inteiro positivo. O THLP consiste em selecionar um subconjunto $P \subset N$ com p nós, denominados concentradores (do inglês *Hubs*), e conectá-los na forma de uma árvore. Em seguida, cada um dos nós em $N \setminus P$, denominados nós clientes, é alocado a um único nó em P de modo que exista um único caminho entre cada par de nós $i, j \in N$ e cujo custo total de rotear as demandas W_{ij} seja minimizado. O custo de trafegar uma unidade de fluxo entre dois nós concentradores $k, m \in P$ sofre um fator de desconto α e é dado por $c_{km} \times \alpha$. O estado da arte de algoritmos para este problema é capaz de resolver instâncias de até 100 nós usando um algoritmo baseado em decomposição de Benders, em um elevado tempo computacional. Isto se deve especialmente ao fato de que $O(|N|^2)$ subproblemas devem ser resolvidos a cada iteração do algoritmo, utilizando um algoritmo de programação linear. Neste trabalho, propomos um algoritmo *ad-hoc* que resolve cada subproblema em $O(|N|^2)$. Desta forma, acelara-se o tempo de execução do algoritmo de decomposição de Benders para o THLP. Resultados preliminares indicam que o tempo de resolução dos subproblemas pode ser acelerado em até 29,52%.

Palavras-chave: arrependimento relativo, Problema de Localização de Concentradores em Árvores, Problemas *NP-Difíceis*, Decomposição de Benders.

Abstract

This work is dedicated to the study of two *NP-Hard* optimization problems. The first problem is the robust shortest path problem (RSP) which is a generalization of the shortest path problem (SP). The RSP considers that the costs of the arcs are defined by a range of continuous values. Among the different robust optimization criterion, this work is dedicated to criterion with *minmax* relative regret. This work gave three contributions to RSP with relative regret. The first, is the integer linear programming formulation to this problem. The seccond is the proposal of valid inequalities. The final is that it extends this problem for graphs with positive cost cycles. Computational results show that algorithms based on proposed contributions are able to solve instances up to 1500 nodes.

The second problem this work investigates is the Tree Hub Location problem (THLP). Let $G = (N, A)$ a complete directed graph, where N is the set of nodes and A is the set of arcs. Besides, $W_{ij} \in \mathbb{R}_+$ is the flow demand of node $i \in N$ to node $j \in N$. c_{ij} is the cost travel to each flow unit in an arc $(i, j) \in A$, and p is a positive integer. The THLP is to select a subset $P \subset N$ with p nodes, called hubs, and connect them in the form of a tree. Then, each node in $N \setminus P$, called a customer, is allocated to a single node in P so there is a unique path between each

pair of nodes $i, j \in N$ whos total cost route demands that W_{ij} is minimized. The cost of one unit flow traveling between two hub nodes $k, m \in P$ suffers a discount factor α and is given by $c_{km} \times \alpha$. The state of the art algorithms for this problem is able to resolve instances up to 100 nodes using an algorithm based on Benders decomposition, in a high computational time. This is primarily due to the fact that $O(|N|^2)$ subproblems must be solved at each iteration of algorithm, using a linear programming algorithm. In this work, we propose an algorithm *ad-hoc* solving each subproblem in $O(|N|^3)$. This way, speeds up the execution time of the Benders decomposition algorithm to THLP. Preliminary results indicate that the time resolution of subproblems can be accelerated by up to 29.52%.

Keywords: *minmax relative regret*, Tree Hub Location problem, *NP-Hard* Problems, Benders decomposition.

Lista de Figuras

1.1 Exemplo de uma solução do THLP em um grafo completo e direcionado com 4 nós e número de nós concentradores, $p = 3$, para serem selecionados 7

2.1 Exemplo de um grafo com custo definido em um intervalo positivo de valores 14

2.2 Um grafo Karasan com 8 nós e 3 camadas. 21

2.3 Exemplo de um grafo grid 3×5. 22

3.1 Exemplo do sistema de restrição diferença para o problema dual, em um grafo completo e direcionado com 4 nós e número de nós concentradores, $p = 3$, para serem selecionados 53

Lista de Tabelas

2.1 Comparação entre CPLEX (V1) e CPLEX (V2) para grafos Karasan com 100 e 200 nós. 24
2.2 Comparação entre CPLEX (V1) e CPLEX (V2) para grafos grid com no máximo 100 e 200 nós. 25
2.3 Comparação entre CPLEX (V1) e CPLEX (V2) para grafos Karasan com 1000 e 1500 nós. 27
2.4 Comparação entre CPLEX (V1) e CPLEX (V2) para grafos grid com no máximo com 1000 nós. 28

3.1 Comparação do tempo gasto, em segundos, e do valor ótimo encontrado pela resolução da relaxação linear dos modelos CPLEX (V1) e CPLEX (V2) nas instâncias AP. 57
3.2 Comparação do tempo gasto, em segundos, e do valor ótimo encontrado pela resolução da relaxação linear dos modelos CPLEX (V1) e CPLEX (V2) nas instâncias CAB. 58
3.3 Comparação do tempo gasto, em segundos, e da quantidade de nós na árvore de *branch and bound* pela resolução dos modelos CPLEX (V1) e CPLEX (V2) nas instâncias AP. 60

3.4 Comparação do tempo gasto, em segundos, e da quantidade de nós na árvore de *branch and bound* pela resolução dos modelos CPLEX (V1) e CPLEX (V2) nas instâncias CAB. 62
3.5 Comparação entre Benders (V1) e Benders+ em instâncias AP de 50 nós. 64

Sumário

Agradecimentos **xi**

Resumo **xv**

Abstract **xvii**

Lista de Figuras **xix**

Lista de Tabelas **xxi**

1 Introdução **1**

2 O Problema do Caminho Mais Curto Robusto **9**

2.1 Trabalhos Relacionados 9

2.2 Formulação Linear 16

2.3 Restrições de eliminação de subciclos 18

2.4 Inequações Válidas 19

2.5 Experimentos Computacionais 20

3 Problema de Localização de Concentradores em Árvore **29**

3.1 Trabalhos Relacionados 29

3.2 Formulações de Programação Linear Inteira Mista para o THLP . . 32
3.3 Algoritmo de Decomposição de Benders 38
3.4 Algoritmo Eficiente para resolução do subproblema 47
3.5 Experimentos Computacionais 52

4 Conclusões **65**

Referências Bibliográficas **67**

Capítulo 1

Introdução

Esta dissertação trata de dois problemas de otimização combinatória *NP-Difíceis*: o problema do caminho mais curto robusto (RSP, do inglês *Robust Shortest Path problem*) e o problema de localização de concentradores em árvores (THLP, do inglês *Tree of Hub Location Problem*). O objetivo desta dissertação é avançar no estado da arte dos algoritmos exatos para esses dois problemas.

Considere um grafo orientado e conexo $G = (V, A)$ com um conjunto V de nós e um conjunto A de arcos. Seja $n = |V|$ e $m = |A|$, respectivamente, o número total de nós e arcos de G. Cada arco $(i, j) \in A$ tem um custo associado $c_{ij} \in \mathbb{R}$. O custo de um caminho $P \subseteq A$ de um nó de origem $s \in V$ até um nó de destino $t \in V$ é dado pela soma dos custos dos arcos em P. O problema do caminho mais curto (SP, do inglês *Shortest Path Problem*) consiste em encontrar o caminho de menor custo a partir de uma origem $s \in V$ para um destino $t \in V$. Uma solução existe se ciclos de custo negativo não estão presente no caminho de s para t. Algoritmos de tempo polinomial estão disponíveis para resolver o SP, como Dijkstra [22] e Bellman-Ford [5]. O SP modela problemas práticos e teóricos [29]. Entretanto,

diversas aplicações para o SP possuem incertezas nos dados.

Algumas estratégias podem ser utilizadas para resolver o problema com incerteza como a programação estocástica [54] e a otimização robusta [6, 40]. A programação estocástica é mais aplicada quando a lei de probabilidade associada à incerteza dos dados é conhecida. Uma dificuldade dessa abordagem é que algumas vezes é difícil definir a distribuição de probabilidade associada a incerteza dos dados ou erros podem acontecer na estimativa dos parâmetros. Além disso, a programação estocástica não se aplica quando a otimização requer decisões não repetitivas como projeto de infraestrutura, acidentes ambientais ou nucleares, etc. [1]. Os trabalhos [8, 48] são dedicados ao problema do caminho mais curto estocástico, que é uma extensão do SP, uma vez que minimiza o custo total esperado.

Otimização robusta é uma alternativa para a programação estocástica onde a variabilidade dos dados é representada por valores determinísticos. Neste trabalho, o foco são modelos de otimização robusta onde a incerteza dos dados é modelada por um intervalo de possíveis valores. O livro [40] apresenta alguns modelos de otimização robusta. RSP é uma generalização do SP, onde o custo de cada arco $(i, j) \in A$ é definido por um intervalo $[l_{ij}, u_{ij}]$, com $l_{ij}, u_{ij} \in \mathbb{N}^*$ e $u_{ij} \geq l_{ij}$, $\forall (i, j) \in A$ [34]. Existem diferentes versões do RSP com intervalo de valores na literatura, a diferença entre eles está no critério de otimização utilizado [1, 3, 13, 35, 57].

A versão mais estudada do RSP usa o critério *minmax regret* e é chamada de *minmax regret* RSP. Seja $\mathcal{P} \subseteq A$ um caminho a partir da origem s para o destino t em G. Um cenário r define uma atribuição do custo $c_{ij} \in [l_{ij}, u_{ij}]$, de cada arco $(i, j) \in A$, em um único valor dado por c_{ij}^r. Assim o custo do caminho $\mathcal{P}$ no cenário r é a soma dos custos c_{ij}^r dos arcos presentes em $\mathcal{P}$. O *arrependimento* (do inglês *regret*) de $\mathcal{P}$ no cenário r (também conhecido como o desvio robusto, do inglês

robust deviation) é definido como a diferença entre o custo de $\mathcal{P}$ em r e o custo do caminho mais curto S^r de s até t em r. Em outras palavras, o *desvio robusto* de $\mathcal{P}$ em r é o *arrependimento* de usar $\mathcal{P}$ ao invés de S^r, quando o cenário r ocorre. O *custo robusto* (do inglês, *robust cost*) de $\mathcal{P}$ é o maior desvio robusto de $\mathcal{P}$ em todos os cenários. O *minmax regret* RSP consiste em encontrar o caminho $\mathcal{P}^*$ a partir de s para t com o menor custo robusto. Esse problema é *NP-Difícil*, inclusive para grafos que não possuem ciclos [40].

O primeiro problema estudado nesta dissertação é o problema do caminho mais curto robusto intervalar com arrependimento relativo - *minmax relative regret* RSP. Seja $\mathcal{P} \subseteq A$ um caminho a partir da origem s para o destino t em G. O *arrependimento relativo* (do inglês *relative regret*) de $\mathcal{P}$ no cenário r (também conhecido como *desvio robusto relativo*, do inglês *relative robust deviation*) é definido como $(custo(\mathcal{P}, r) - custo(S^r, r))/custo(S^r, r)$, onde $custo(\mathcal{P}, r)$ denota o custo de $\mathcal{P}$ em r e $custo(S^r, r)$ denota o custo de S^r em r. O *custo robusto relativo* (do inglês *relative robust cost*) de $\mathcal{P}$ é o maior arrependimento relativo de $\mathcal{P}$ em todos os cenários. O *minmax relative regret* RSP consiste em encontrar o caminho $\mathcal{P}^*$, sem a presença de ciclos, a partir de s até t com o menor custo robusto relativo. Esse problema é *NP-Difícil*, inclusive para grafos que não possuem ciclos [3]. Embora o *minmax relative regret* RSP difere do *minmax regret* RSP somente pela função objetivo, ele é mais difícil de resolver porque a função objetivo é não linear.

O *desvio robusto relativo* pode ser considerado uma métrica melhor do que *desvio robusto* porque o *arrependimento* de usar $\mathcal{P}$ ao invés de S^r é normalizado pelo custo de S^r. Por exemplo, sejam dois caminhos $\mathcal{P}'$ e $\mathcal{P}''$, e também dois cenários r' e r'' tal que $custo(\mathcal{P}', r') = 11$, $custo(S^{r'}, r') = 1$, $custo(\mathcal{P}'', r'') = 100$ e $custo(S^{r''}, r'') = 90$. De acordo com o critério *minmax regret*, o *arrependimento* de

$\mathcal{P}'$ em r' $(11-1=10)$ é o mesmo que $\mathcal{P}''$ em r'' $(100-90=10)$. Entretanto, pode-se observar que o custo de $\mathcal{P}'$ é dez vezes maior do que o custo de $S^{r'}$, enquanto o custo de $\mathcal{P}''$ é somente 11% maior do que o custo de $S^{r''}$. De acordo com o critério *minmax relative regret*, o *arrependimento* de $\mathcal{P}'$ em r' $((11-1)/1=10)$, é muito maior do que de $\mathcal{P}''$ em r'' $((100-90)/90=0,11)$.

As contribuições desta dissertação para este problema são as seguintes. Propor a primeira formulação de programação linear inteira mista para o *minmax relative regret* RSP, juntamente com inequações válidas. Adicionar restrições de eliminação de subciclos para essa formulação e propor um novo conjunto de instâncias baseadas em grafos direcionados com ciclos.

O segundo problema estudado nesta dissertação pertence a classe de problemas de localização de concentradores. Essa classe de problemas é estudada desde 1986 [2, 12, 32]. Considere $G=(N,A)$ um grafo completo e direcionado, onde N é o conjunto de nós e A é o conjunto de arcos. Cada arco $(i,j)\in A$ possui um custo $c_{ij}\in\mathbb{R}_+$ associado. Além disso, para cada par de nós $i,j\in N$ existe uma quantidade de fluxo $W_{ij}\in\mathbb{R}_+^*$ que deve ser enviada do nó de origem i para o nó de destino j. Os problemas da classe de problemas de localização de concentradores consiste em dividir o conjunto N em dois conjuntos denominados, respectivamente, concentradores e clientes, conectar os nós concentradores e alocar os nós clientes aos nós concentradores de forma que o custo de transportar todos as demandas de fluxo W_{ij} seja minimizado. Existe um custo $F_k\in\mathbb{R}_+$ para selecionar um nó $k\in N$ como nó concentrador. Cada nó cliente deve ser alocado em um ou mais nós concentradores. Quando cada nó cliente é alocado para apenas um nó concentrador a alocação é conhecida como simples e quando cada nó cliente é alocado para pelo menos um nó concentrador, a alocação é dita ser múltipla. Todo o fluxo

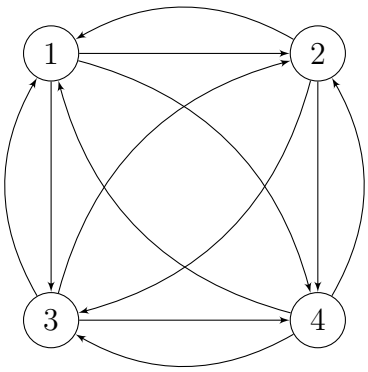
W_{ij} é transportado através dos nós concentradores. O custo de enviar um fluxo W_{ij} do nó de origem i para o nó de destino j é dado pelo soma dos custos dos arcos, presentes no caminho de i até j multiplicado pela quantidade de fluxo W_{ij} enviada de i para j. Quando o arco presente no caminho de i para j conectar dois nós concentradores $k, m \in N$, o custo do arco c_{km} recebe um desconto $\alpha \in [0, 1]$, sendo o novo custo igual a $c_{km} \times \alpha$. O objetivo dos problemas da classe de problemas de localização de concentradores é minimizar o custo de transportar os fluxos através dos nós concentradores selecionados e dos nós clientes alocados. Muitos desses problemas pertencem à classe de problemas *NP-Difíceis* e se diferem quanto a existência de uma quantidade fixa de nós concentradores que precisam ser alocados, quanto a alocação ser simples ou múltipla e quanto a forma de conexão dos nós concentradores.

Nesta dissertação é estudado o problema de localização de concentradores em árvore (THLP, do inglês Tree of Hub Location Problem). Nesse problema o custo de selecionar um nó k do conjunto N como nó concentrador é definido como $F_k = 0$, a quantidade de nós concentradores selecionados é um número fixo $3 \leq p \leq |N|-1$, a alocação é simples e os nós concentradores são conectados através de uma árvore. O THLP foi proposto por Contreras et. al.[18], onde os autores provaram que esse problema é *NP-Difícil* e apresentaram uma aplicação real desse problema na construção da rede de trens de alta velocidade da Espanha. A Figura 1.1 apresenta um exemplo para o THLP em um grafo completo e direcionado com 4 nós, onde 3 nós precisam ser selecionados para serem nós concentradores. Nessa figura, os custos c_{ij} de cada arco $(i, j) \in A$ e as demandas de fluxo W_{ij} para cada par de nós $i, j \in N$ não são apresentados. Na Figura 1.1a é mostrado o grafo completo e direcionado $G = (N, A)$. Na figura 1.1b é destacado os nós selecionados

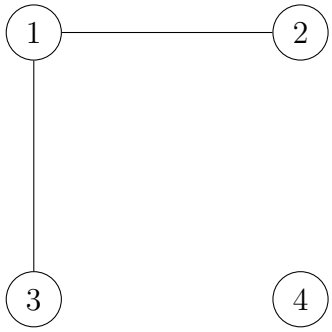
para serem nós concentradores e as conexões entre eles, formando uma árvore. Na figura 1.1c é mostrado a alocação do nó cliente 4 ao nó concentrador 3. A Figura 1.1d apresenta o caminho que o fluxo W_{42}, com origem no nó 4 e destino no nó 2, percorre. O custo de enviar o fluxo W_{42} é dado então por $(c_{43}+(c_{31}+c_{12})\times\alpha)\times W_{42}$. Já a Figura 1.1e apresenta o caminho percorrido para enviar o fluxo W_{24}, do nó de origem 2 para o nó de destino 4. O custo para enviar esse fluxo é dado por $((c_{21}+c_{13})\times\alpha+c_{34})\times W_{24}$. Esse exemplo apresenta a seguinte característica do THLP: embora os nós concentradores estão conectados por meio de uma árvore não direcionada, o fluxo possui uma direção.

O THLP é muito bem resolvido na literatura por um algoritmo de decomposição de Benders proposto por de Sá et. al.[56]. Entretanto, o tempo de execução desse algoritmo chega a diversas horas ou até dias de processamento. A contribuição desta dissertação para esse problema é propor um algoritmo ad-hoc para resolver os subproblemas da decomposição de Benders. Espera-se que este algoritmo seja muito mais eficiente do que o algoritmo de programação linear genérico utilizado na literatura.

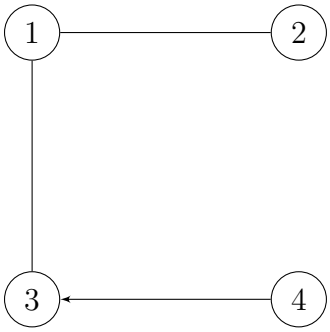
Este trabalho está organizado da seguinte forma: o capítulo 2 é dedicado ao estudo do problema do caminho mais curto robusto intervalar com arrependimento relativo enquanto no capítulo 3 é estudado o problema de localização de concentradores em árvore. As conclusões deste trabalho podem ser encontradas no capítulo 4.



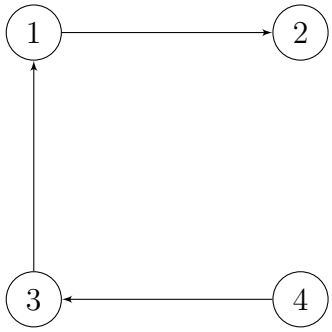
(a) Grafo completo e direcionado



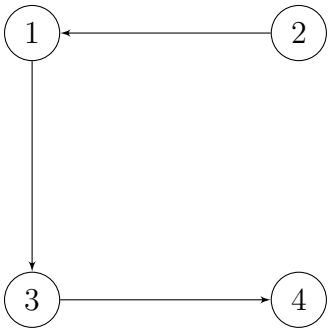
(b) Nós concentradores 1, 2 e 3 e sua árvore de conexão



(c) Alocação do nó cliente 4 no nó concentrador 3



(d) Caminho percorrido pelo fluxo com origem no nó 4 e destino no nó 2



(e) Caminho percorrido pelo fluxo com origem no nó 2 e destino no nó 4

Figura 1.1: Exemplo de uma solução do THLP em um grafo completo e direcionado com 4 nós e número de nós concentradores, $p = 3$, para serem selecionados

Capítulo 2

O Problema do Caminho Mais Curto Robusto

Este capítulo é dedicado ao estudo do problema do caminho mais curto robusto e está organizado da seguinte forma: primeiro, os trabalhos relacionados são apresentados na seção 2.1. Então, a primeira formulação linear inteira mista (MILP, do inglês *Mixed Integer Linear Programming*) é proposta na seção 2.2, a seção 2.3 apresenta as restrições de eliminação de subciclos, na seção 2.4 é proposta inequações válidas para esse problema e os resultados computacionais são reportados na seção 2.5.

2.1 Trabalhos Relacionados

O *minmax regret* RSP foi proposto em [34]. Os autores provam que o pior cenário para o caminho $\mathcal{P}$ pode ser obtido fixando os custos $c^r_{ij} = u_{ij}$ para todo $(i,j) \in \mathcal{P}$, e $c^r_{ij} = l_{ij}$ para todo $(i,j) \in A \setminus \mathcal{P}$. Esse cenário é denominado cenário induzido

por $\mathcal{P}$. Usando esse resultado, a formulação MILP foi definida com as variáveis de decisão $y_{ij} = 1$ se o arco $(i, j) \in A$ está presente na solução e $y_{ij} = 0$ caso contrário. Além disso, variáveis auxiliares $x_i \geq 0$, para todo $i \in V$, mantêm o custo do caminho de menor custo a partir da origem s para o nó i no cenário induzido pelas variáveis y_{ij}. A formulação MILP correspondente é definida pelas equações (2.1) até (2.6).

$$\min \quad z = \sum_{(i,j)\in A} u_{ij} \cdot y_{ij} - x_t \tag{2.1}$$

sujeito à:

$$\sum_{(j,k)\in A} y_{jk} - \sum_{(i,j)\in A} y_{ij} = \begin{cases} 1, & \text{if j = s} \\ -1, & \text{if j = t} \\ 0, & \text{caso contrário} \end{cases} \qquad \forall j \in V \tag{2.2}$$

$$x_j \leq x_i + l_{ij} + (u_{ij} - l_{ij})y_{ij} \quad \forall (i, j) \in A \tag{2.3}$$

$$x_s = 0 \tag{2.4}$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{2.5}$$

$$x_i \geq 0 \quad \forall i \in V \tag{2.6}$$

A função objetivo (2.1) minimiza o custo do *arrependimento* do caminho definido pelas variáveis y_{ij}. As restrições (2.2) são as clássicas restrições de conservação de fluxo e garantem a conectividade a partir do nó de origem s para o nó de destino t. As inequações (2.3) garantem que o custo do menor caminho entre s e t é calculado no cenário onde o custo dos arcos y_{ij} estão fixados no maior valor e o custo dos demais arcos estão fixados no menor valor possível. Essas restrições,

juntamente com a função objetivo, garantem que o máximo *arrependimento* vai ser minimizado. A equação (2.4) garante que o custo do caminho mais curto com origem e destino em s seja igual a zero. Finalmente, o domínio das variáveis y e x são definidas em (2.5) e (2.6), respectivamente. Essa formulação pode resolver eficientemente algumas instâncias através de *solvers* comerciais, como o CPLEX. Por exemplo, instâncias geradas aleatoriamente com até 900 nós e instâncias reais, a partir de uma rede de estradas de Stuttgart, com 2490 nós foi resolvida em poucos segundos em [47]. Instâncias com mais de 1000 nós são resolvidas na otimalidade. O CPLEX interrompeu a execução para instâncias variando entre 10000 e 20000 nós devido a falta de memória. Os gaps para essas instâncias variaram entre 2% até 9% [50].

O *minmax relative regret* RSP foi proposto em [40]. Neste trabalho, vai ser considerado o *minmax relative regret* RSP no pior cenário apresentado para o *minmax regret* RSP. Para provar que essa versão do *minmax relative regret* RSP é *NP-Difícil*, primeiro provamos o Lema 1. Seja P_1 e P_2 dois caminhos de s para t e r_1 e r_2 o pior cenário desses caminhos, conforme apresentado para o *minmax regret* RSP. Além disso, seja S_1 e S_2 o caminho de menor custo entre s e t nos cenários r_1 e r_2, respectivamente.

Lema 1 *Dado dois caminhos P_1 e P_2, e $M > (|V| \cdot U)^2$, com $U = \max_{(i,j)\in A} u_{ij}$.*

$$C_{P_2}^{r_2} - C_{S_2}^{r_2} > C_{P_1}^{r_1} - C_{S_1}^{r_1} \implies \frac{C_{P_2}^{r_2} + M}{C_{S_2}^{r_2} + M} > \frac{C_{P_1}^{r_1} + M}{C_{S_1}^{r_1} + M}, \tag{2.7}$$

Onde $C_P^r = \sum_{(i,j)\in P} c_{ij}^r$ é o custo do caminho P no cenário r.

Prova Temos que

$$\frac{C_{P_2}^{r_2}+M}{C_{S_2}^{r_2}+M}-\frac{C_{P_1}^{r_1}+M}{C_{S_1}^{r_1}+M}=\frac{(C_{S_1}^{r_1}+M)\cdot(C_{P_2}^{r_2}+M)-(C_{S_2}^{r_2}+M)\cdot(C_{P_1}^{r_1}+M)}{(C_{S_2}^{r_2}+M)\cdot(C_{S_1}^{r_1}+M)},$$

que pode ser reescrito como (2.8).

$$\frac{M\cdot(C_{P_2}^{r_2}-C_{S_2}^{r_2}-C_{P_1}^{r_1}+C_{S_1}^{r_1})+C_{P_2}^{r_2}\cdot C_{S_1}^{r_1}-C_{P_1}^{r_1}\cdot C_{S_2}^{r_2}}{(C_{S_2}^{r_2}+M)\cdot(C_{S_1}^{r_1}+M)} \tag{2.8}$$

A partir da equação (2.7), $C_{P_2}^{r_2}-C_{S_2}^{r_2}-C_{P_1}^{r_1}+C_{S_1}^{r_1}>0$, e pela definição $l_{ij},u_{ij}\in\mathbb{N}^*$, $\forall(i,j)\in A$. A desigualdade (2.9) aparece.

$$C_{P_2}^{r_2}-C_{S_2}^{r_2}-C_{P_1}^{r_1}+C_{S_1}^{r_1}\geq 1. \tag{2.9}$$

A equação (2.9), junto com $M\geq(|V|\cdot U)^2>C_{P_1}^{r_1}\cdot C_{S_2}^{r_2}$, onde $U=\max_{(i,j)\in A}u_{ij}$, implica que (2.8) é maior do que zero, o que completa a prova.

Proposição 1 *O* minmax relative regret *RSP, no pior cenário apresentado para o* minmax regret *RSP é* NP-Difícil.

Prova Seja $f:\{\langle G,l,u,s,t\rangle\}\mapsto\{\langle G',l,u,s',t\rangle\}$ uma função que transforma uma instância do *minmax regret* RSP em uma instância do *minmax relative regret* RSP no pior cenário apresentado para o *minmax regret* RSP, como definido por (2.10) - (2.13):

$$V' = V \cup \{s'\} \tag{2.10}$$

$$A' = A \cup \{(s', s)\} \tag{2.11}$$

$$G' = (V', A') \tag{2.12}$$

$$l_{(s',s)} = u_{(s',s)} = (|V| \cdot U)^2 \tag{2.13}$$

$\Rightarrow$ Seja $P \subseteq A$ uma solução ótima para o *minmax relative regret* RSP no pior cenário apresentado para o *minmax regret* RSP. O caminho P, sem o arco (s', s), é também uma solução ótima para o *minmax regret* RSP. Caso contrário, existe uma solução ótima P' para o *minmax regret* RSP com um custo menor do que a solução P e a partir do Lema 1, P' junto com o arco (s', s) é uma solução ótima para o *minmax relative regret* RSP com um custo menor do que a solução ótima P.

$\Leftarrow$ Seja $Q \subseteq A$ uma solução ótima para o *minmax regret* RSP. A partir do Lema 1, Q juntamente com o arco (s', s) é também uma solução ótima para o *minmax relative regret* RSP no pior cenário apresentado para o *minmax regret* RSP.

A formulação para o *minmax relative regret* RSP será composta pelas restrições (2.2) até (2.6), apresentadas anteriormente para o *minmax regret* RSP e a função objetivo é dada por (2.14).

$$\min \frac{\sum_{(i,j)\in A} u_{ij} y_{ij} - x_t}{x_t} \tag{2.14}$$

Esta formulação é válida apenas para grafos sem ciclos, porque considerando um grafo com ciclos e valores não negativos $l_{ij} > 0$ e $u_{ij} > 0$, a solução pode ser melhorada apenas permitindo subciclos, devido a função objetivo (2.14). A figura 2.1 ilustra essa situação, onde s e t são respectivamente a origem e o destino. Se subciclos são proibidos, então para esse exemplo, existem dois caminhos a partir de s para t: $\{(s,1),(1,t)\}$ e $\{(s,2),(2,t)\}$, respectivamente com custo robusto $\left(\frac{20165-120}{120} = 167,04\right)$ e $\left(\frac{20120-165}{165} = 120,94\right)$. Se subciclos são permitidos, a formulação vai fornecer um caminho com $\{(s,1),(1,s),(s,2),(2,t),(t,1),(1,t)\}$ e custo robusto igual a $\left(\frac{60388-20120}{20120} = 2,0014\right)$. Esse problema não aparece em grafos direcionados sem ciclos, como os estudados em [3]. Caso contrário, restrições de eliminação de subciclos precisam ser consideradas.

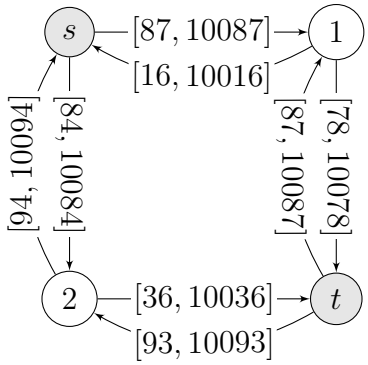


Figura 2.1: Exemplo de um grafo com custo definido em um intervalo positivo de valores

O Survey [1] dedica-se a resultados sobre a complexidade computacional para diversos problemas de otimização robusta, enquanto o survey [28] é dedicado ao RSP com intervalo de dados e RSP com cenários discretos. Uma visão geral sobre problemas de otimização robusta e aplicações, bem como modelos matemáticos para diferentes versões do RSP é encontrado em [40]. Um estudo de algoritmos

exatos, algoritmos aproximativos e heurísticas para problemas de otimização, incluindo o RSP, é apresentado em [13].

O *minmax regret* RSP foi mostrado ser *NP-Difícil* em [40]. Uma formulação MILP para esse problema foi introduzida em [34]. Em [47], os autores propõem um algoritmo *branch and bound* baseado na relaxação combinatória das restrições robustas. Esse trabalho foi estendido em [46] usando um algoritmo de decomposição de Benders baseado na mesma relaxação. Ambos os algoritmos resolveram instâncias geradas aleatoriamente com até 4000 nós e instâncias reais com até 2500 nós.

Os trabalhos [4, 26] insvestigam instâncias que podem ser resolvidas em tempo polinomial ou pseudo-polinomial para problemas clássicos de otimização robusta. Um algoritmo pseudo-polinomial é apresentado em [36] para o *minmax regret* RSP. Esse algoritmo funciona em *series-parallel multidigraphs* e tem complexidade computacional de $O(m \cdot |S^u|^2)$, onde $|S^u|$ representa o número de arcos no caminho de menor custo a partir de s para t no cenário onde o custo dos arcos estão fixados em u_{ij}. Um algoritmo 2-aproximativo para uma ampla classe de problemas de otimização *minmax regret* são apresentados em [16]. Para o *minmax regret* RSP, um algoritmo 2-aproximativo com complexidade, no pior caso, igual a $O(m + n \log n)$ é proposto em [36]. Uma versão melhorada desse algoritmo é dada em [37] para *series-parallel multidigraphs* com um fator de aproximação de $(1 + \epsilon)$ e complexidade, no pior caso, igual a $O(q(n, n/\epsilon))$, onde q é uma função polinomial bivariada e ϵ é um valor no intervalo $(0, 1)$.

Técnicas de pré-processamento para o *minmax regret* RSP são propostas em [14, 34]. A ideia geral é remover arcos dominados, chamados de *arcos-fracos* (do inglês *weak-arcs*), i.e., os arcos que não pertecem a nenhuma uma solução

ótima. Um algoritmo para eliminar *arcos-fracos* é introduzido em [34] para grafos particulares que são acíclicos, planar e em camadas. Em [14], o autor estende o pré-processamento proposto em [34], eliminando nós e aplicando outras estratégias para deduzir *arcos-fracos*. O procedimento para eliminar nós consome $O(n^3)$ no pior caso, e elimina nós que não aparecem em nenhum caminho mais curto entre a origem e o destino. Além disso, os autores apresentam estratégias usando uma árvore geradora mínima para detectar *arcos-fracos* e tem complexidade computacional de $O(m \cdot n^2)$ no pior caso.

O *minmax relative regret* RSP foi introduzido em [40]. Em [3], o autor prova que essa versão do RSP é *NP-Difícil*, mesmo em grafos direcionados acíclicos. Até onde sabemos, não existe na literatura uma formulação linear inteira ou algoritmos exatos para essa versão do RSP.

2.2 Formulação Linear

A função objetivo (2.14) foi linearizada neste trabalho e teve inspiração no trabalho [9], para o caso onde $l_{ij}, u_{ij} \in \mathbb{N}^*$. Primeiro, sem perda de generalidade, pode-se reescrever a função objetivo como

$$\min \frac{\sum_{(i,j)\in A} u_{ij} y_{ij}}{x_t} - \frac{x_t}{x_t} = \sum_{(i,j)\in A} u_{ij} \frac{y_{ij}}{x_t} - 1 \tag{2.15}$$

Em seguida, variáveis $z_{ij} \in \mathbb{R}_+$ são introduzidas, juntamente com as restrições (2.16)-(2.18), para garantir que $z_{ij} = \frac{y_{ij}}{x_t}$, $\forall (i,j) \in A$.

$$z_{ij} \leq y_{ij} \quad \forall (i,j) \in A \tag{2.16}$$

$$z_{ij} \geq \frac{1}{x_t} - (1 - y_{ij}) \quad \forall (i,j) \in A \tag{2.17}$$

$$z_{ij} \in \mathbb{R}_+ \quad \forall (i,j) \in A \tag{2.18}$$

Note que a restrição (2.17) é não linear. Ela pode ser linearizada com a introdução de variáveis w_l, $\forall l \in \{L, \ldots, U\}$, tal que $w_l = 1$ se e somente se $x_t = l$, e $w_l = 0$ caso contrário. O limite inferior L para o valor de x_t é definido como o custo do caminho de menor custo a partir de s para t no cenário onde o custo dos arcos estão fixados em $c^l_{ij} = l_{ij}$, e o limite superior U para o valor de x_t é definido como o custo do caminho de menor custo a partir de s para t no cenário onde o custo dos arcos estão fixados em $c^u_{ij} = u_{ij}$. Portanto, o número de variáveis w_l é igual a $U - L$. Então, troca-se a restrição (2.17) pelas restrições (2.19)-(2.22).

$$\sum_{l \in \{L,\ldots,U\}} z_{ij} \geq \frac{1}{l} \cdot w_l - (1 - y_{ij}) \quad \forall (i,j) \in A \tag{2.19}$$

$$\sum_{l \in \{L,\ldots,U\}} w_l \geq 1 \tag{2.20}$$

$$\sum_{l \in \{L,\ldots,U\}} l \cdot w_l = x_t \tag{2.21}$$

$$w_l \in \{0,1\} \quad \forall l \in \{L, \ldots, U\} \tag{2.22}$$

Finalizando, é possível reescrever a função objetivo não linear (2.14) como a função objetivo linear (2.23) e uma formulação MILP para o *minmax relative*

regret RSP pode ser definida pelas equações (2.2)-(2.6), (2.16), (2.18)-(2.23). O número de restrições dessa formulação é $O(|A|)$, enquanto o número de variáveis é $O(|A| + U)$.

$$\min \sum_{(i,j)\in A} u_{ij} z_{ij} - 1 \tag{2.23}$$

2.3 Restrições de eliminação de subciclos

As restrições de eliminação de subciclos propostas em [44] foram aplicadas com êxito para alguns problemas de otimização clássicos da literatura [21, 52], e são adaptadas aqui para o *minmax relative regret* RSP. Essas restrições estabelecem uma ordem topológica para cada nó visitado no caminho robusto de s até t para eliminação de subciclos.

As restrições (2.24)-(2.26) usam as variáveis t_i, $\forall i \in V$, que determinam a ordem que o nó i aparece no caminho robusto. A restrição (2.24) garante que se o arco $(i, j) \in A$ é escolhido no caminho, i.e. se $y_{ij} = 1$, então $t_i < t_j$. A equação (2.25) define que o primeiro nó visitado em um caminho do nó de origem s até o nó de destino t seja o nó s. O domínio das variáveis t_i é definido em (2.26), $\forall i \in V$. Por exemplo, dado um caminho $\mathcal{P} = \{(s, 1), (1, s), (s, 2), (2, t)\}$, quando os arcos $(s, 1)$ e $(1, s)$ são simultaneamente considerados, a restrição correspondente (2.24) é dada, respectivamente, por $t_s - t_1 \leq -1$ e $t_1 - t_s \leq -1$. Mas, essas inequações não podem ser simultaneamente satisfeitas. Assim, os subciclos são eliminados.

A formulação completa considerada aqui para o *minmax relative regret* é dada pela função objetivo (2.23) e as restrições (2.2)-(2.6), (2.16), (2.18)-(2.26).

$$t_i - t_j + |V|y_{ij} \leq |V| - 1, \qquad \forall (i,j) \in A \tag{2.24}$$

$$t_s = 0 \tag{2.25}$$

$$0 \leq t_i \leq |V| \qquad \forall i \in V \tag{2.26}$$

2.4 Inequações Válidas

Dado o limite inferior L e o limite superior U para o valor de x_t, definido anteriormente, segue que

$$\frac{\sum_{(i,j)\in A} u_{ij}y_{ij} - x_t}{U} \leq \frac{\sum_{(i,j)\in A} u_{ij}y_{ij} - x_t}{x_t} \leq \frac{\sum_{(i,j)\in A} u_{ij}y_{ij} - x_t}{L}.$$

Como

$$\sum_{(i,j)\in A} u_{ij}z_{ij} - 1 = \frac{\sum_{(i,j)\in A} u_{ij}y_{ij} - x_t}{x_t}$$

em uma solução ótima inteira para a formulação MILP descrita anteriormente, as inequações (2.27) e (2.28) são válidas para essa formulação. Embora (2.27) e (2.28) sejam redundantes na formulação inteira, elas melhoram (i) o limite inferior da relaxação linear dessa formulação e (ii) a performance do algoritmo de branch and bound do CPLEX.

$$\sum_{(i,j)\in A} u_{ij} z_{ij} - 1 \geq \frac{\sum_{(i,j)\in A} u_{ij} y_{ij} - x_t}{U} \tag{2.27}$$

$$\sum_{(i,j)\in A} u_{ij} z_{ij} - 1 \leq \frac{\sum_{(i,j)\in A} u_{ij} y_{ij} - x_t}{L} \tag{2.28}$$

A formulação resultante para o *minmax relative regret* é dada pela função objetivo (2.23) e as restrições (2.2)-(2.6), (2.16), (2.18)-(2.28). Essas restrições não somente melhoram a relaxação linear através da imposição de um limite inferior para o valor da função objetivo, mas também pelo fortalecimento da ligação entre as variáveis z_{ij}, y_{ij}, e x_t.

2.5 Experimentos Computacionais

Os experimentos computacionais foram executados em uma máquina Intel Core i7 com 2.67 GHz de clock e 4 GB de memória RAM, rodando o sistema operacional Linux. Dois algoritmos *branch and bound* para o *minmax relative regret* RSP foram implementados usando o ILOG CPLEX versão 12.5, com parâmetros nos valores padrões. O primeiro foi baseado em (2.2)-(2.6), (2.16), (2.18)-(2.26), chamado aqui de CPLEX (V1), e o segundo foi baseado em (2.2)-(2.6), (2.16), (2.18)-(2.28) e é identificado aqui por CPLEX (V2) e difere do primeiro pela adição das inequações válidas (2.27) e (2.28). Dois conjuntos de instâncias são usadas nos experimentos computacionais: instâncias Karasan [34] e instâncias grid, propostas neste trabalho. Esses dois conjunto são descritos abaixo.

Grafos Karasan foram propostos em [34] e foram usados em experimentos

computacionais de [34, 45, 46, 47, 50]. Estes grafos são em camadas [55] e acíclicos [10] que se parecem com redes de telecomunicações. Em grafos Karasan, toda camada C tem um número W de nós. Existe um arco entre cada nó da camada c para cada nó da camada $c+1$, com $c \in \{1, \ldots, C-1\}$. Além disso, existe um arco a partir do nó s para todo nó da camada 1 e existe um arco a partir de cada nó da última camada para o nó t. Um grafo Karasan com 8 nós (incluindo s e t) e 3 camadas é mostrado na figura 2.2. O intervalo de dados $[l_{ij}, u_{ij}]$, $\forall (i,j) \in A$, é definido a seguir. Primeiro, um número aleatório $\theta_{ij} \in [1, \theta_{max}]$ é gerado $\forall (i,j) \in A$, onde $\theta_{max} = 200$. Então, l_{ij} é definido como $U[(1-d) \cdot \theta_{ij}, (1+d) \cdot \theta_{ij}]$, e u_{ij} é definido como $U[l_{ij}, (1+d) \cdot \theta_{ij}]$, onde $d = 0.9$ e $U[a,b]$ denota a parte inteira de um número aleatório uniformimente selecionado no intervalo $[a,b]$. Essas instâncias são nomeadas como K-v-θ_{max}-d-i-W, onde K identifica o tipo da instância, v é o número de nós entre s e t, enquanto i distingue diferentes instâncias geradas com os mesmos valores para os parâmetros. Os valores dos parâmetros mencionados anteriormente θ_{max}, d, e W são também mostrados no nome da instância.

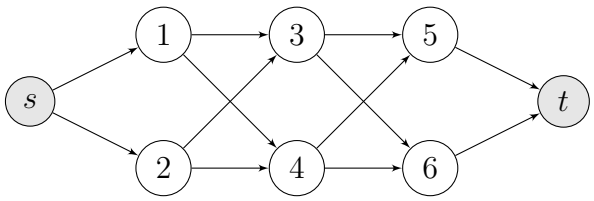


Figura 2.2: Um grafo Karasan com 8 nós e 3 camadas.

Grafos grid são baseados em uma matriz $n \times m$, onde n é o número de linhas e m é o número de colunas. Cada célula da matriz corresponde a um nó e existe arcos bi-direcionais entre cada par de nós cuja respectiva célula da matriz são adjacentes. O nó de origem s é definido sendo o nó superior esquerdo e o de

destino t é definido como sendo o nó inferior direito. Essas instâncias foram criadas porque elas se parecem com interseção de ruas em rede de estradas e porque elas contêm muitos ciclos. Um exemplo de um grid 3×5 é mostrado na figura 2.3. O valor de $[l_{ij}, u_{ij}]$, $\forall (i,j) \in A$, foi gerado da mesma forma como explicado para as instâncias Karasan. As instâncias são nomeadas como G-$n\times m$-i, onde G identifica o tipo da instância, n é o número de linhas, m é o número de colunas e i distingue diferentes instâncias geradas com os mesmos valores para os parâmetros.

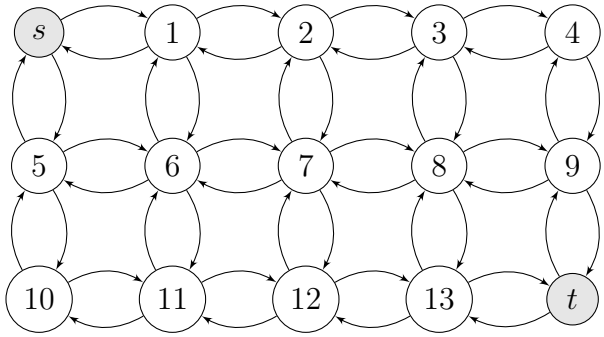


Figura 2.3: Exemplo de um grafo grid 3×5.

No primeiro experimento, foi avaliado o impacto das inequações válidas (2.27) e (2.28) pela comparação de desempenho do CPLEX (V1) e CPLEX (V2) para pequenas instâncias Karasan e Grid com 100 e 200 nós. O tempo máximo para rodar ambos os algoritmos foi setado para 7200 segundos. Os resultados desse experimento são mostrados nas tabelas 2.1 e 2.2, respectivamente. O nome da instância aparece na coluna 1. O limite inferior (lb) e o limite superior (ub) para o valor da solução ótima obtida pelo CPLEX (V1) estão nas colunas 2 e 3, respectivamente. O gap de otimalidade, o tempo de execução e o número de nós explorados na árvore de *branch and bound* são reportados, respectivamente, nas colunas 4 até 6. Os mesmos dados são mostrados para o CPLEX (V2) nas últimas

cinco colunas, respectivamente. Pode ser visto que as inequações válidas (2.27) e (2.28) melhoram o limite inferior da relaxação linear. Para as instâncias Karasan (Tabela 2.1), pode ser visto que quanto menor o número de nós em cada camada, a dificuldade de resolução da instância aumenta. CPLEX (V1) encontrou soluções ótimas para quase todas as instâncias mas K-200-200-0.9-a-2, K-200-200-0.9-b-2, K-200-200-0.9-a-5, e K-200-200-0.9-b-5 não foram resolvidas na otimalidade antes de 7200 segundos, enquanto o CPLEX (V2) resolveu K-200-200-0.9-b-5 além de todas as outras instâncias resolvidas pelo CPLEX (V1). A média relativa do gap de integralidade do CPLEX (V1) foi 20,19%, enquanto que para o CPLEX (V2) foi somente de 2,40%. Para as instâncias grid (Tabela 2.2), CPLEX (V1) resolveu todas as instâncias com até 100 nós mas nenhuma com até 200 nós antes de 7200 segundos, enquanto o CPLEX (V2) resolveu todas as instâncias com 100 nós e metade das instâncias com 200 nós. Além disso, CPLEX (V1) falhou em encontrar soluções para quatro (das dez) instâncias com 200 nós com limite de tempo de 7200 segundos, enquanto CPLEX (V2) encontrou soluções viáveis para todas as instâncias. A média relativa do gap de integralidade do CPLEX (V1) para essas instâncias foi de 47,43%, enquanto que para o CPLEX (V2) foi somente 2,52%. Esses resultados sugerem que as instâncias Grid propostas nesse trabalho são mais difícies de serem resolvidas do que as instâncias Karasan. Além disso, eles indicam que o CPLEX (V2) pode ser aplicado eficientemente para pequenas instâncias Karasan e grid com até 200 nós.

No segundo experimento, foi comparado o CPLEX (V1) e CPLEX (V2) para grandes instâncias Karasan e grid com até 1500 nós. O tempo máximo de execução, para ambos os algoritmos, foi de 7200 segundos. Os resultados desse experimento são mostrados nas tabelas 2.3 e 2.4, respectivamente. O nome das

Tabela 2.1: Comparação entre CPLEX (V1) e CPLEX (V2) para grafos Karasan com 100 e 200 nós.

	CPLEX (V1)					CPLEX (V2)				
Instance	lb (%)	ub (%)	gap (%)	t(s)	Nodes	lb (%)	ub (%)	gap(%)	t(s)	Nodes
K-100-200-0.9-a-2	0,00	0,00	0,00	0,87	1708	0,00	0,00	0,00	0,19	1
K-100-200-0.9-b-2	0,00	0,00	0,00	1,09	2313	0,00	0,00	0,00	0,17	1
K-100-200-0.9-a-5	2,27	2,27	0,00	0,85	52	2,27	2,27	0,00	0,29	1
K-100-200-0.9-b-5	0,79	0,79	0,00	0,97	209	0,79	0,79	0,00	0,33	1
K-100-200-0.9-a-10	0,63	0,63	0,00	1,54	64	0,63	0,63	0,00	0,42	1
K-100-200-0.9-b-10	0,00	0,00	0,00	0,56	4	0,00	0,00	0,00	0,19	1
K-100-200-0.9-a-25	0,00	0,00	0,00	0,42	1	0,00	0,00	0,00	0,31	1
K-100-200-0.9-b-25	1,35	1,35	0,00	0,74	1	1,35	1,35	0,00	0,49	1
K-100-200-0.9-a-50	2,44	2,44	0,00	0,44	1	2,44	2,44	0,00	2,04	1
K-100-200-0.9-b-50	0,00	0,00	0,00	0,64	1	0,00	0,00	0,00	0,46	1
K-200-200-0.9-a-2	-86,38	60,99	147,37	7200,00	1202871	39,35	60,93	21,58	7200,00	1199976
K-200-200-0.9-b-2	-80,98	59,81	140,79	7200,00	521026	38,65	59,68	21,03	7200,00	1480963
K-200-200-0.9-a-5	-9,77	62,37	72,15	7200,00	1175419	54,90	60,20	5,30	7200,00	586265
K-200-200-0.9-b-5	-4,35	39,22	43,57	7200,00	1027303	39,22	39,22	0,00	55,14	19681
K-200-200-0.9-a-10	73,08	73,08	0,00	2453,23	445310	73,08	73,08	0,00	46,75	16325
K-200-200-0.9-b-10	72,09	72,09	0,00	1253,04	321462	72,09	72,09	0,00	9,93	3077
K-200-200-0.9-a-25	45,83	45,83	0,00	8,79	76	45,83	45,83	0,00	4,47	4
K-200-200-0.9-b-25	76,92	76,92	0,00	11,55	209	76,92	76,92	0,00	5,42	77
K-200-200-0.9-a-50	35,71	35,71	0,00	3,46	1	35,71	35,71	0,00	5,01	1
K-200-200-0.9-b-50	25,00	25,00	0,00	6,28	1	25,00	25,00	0,00	5,56	1
Average			20,19					2,40		

Tabela 2.2: Comparação entre CPLEX (V1) e CPLEX (V2) para grafos grid com no máximo 100 e 200 nós.

	CPLEX (V1)					CPLEX (V2)				
Instance	lb (%)	ub (%)	gap (%)	t(s)	Nodes	lb (%)	ub (%)	gap (%)	t(s)	Nodes
g_3x30_a	0,50	0,50	0,00	1,33	118	0,50	0,50	0,00	0,58	1
g_3x30_b	0,70	0,70	0,00	1,91	655	0,70	0,70	0,00	0,70	22
g_4x25_a	0,96	0,96	0,00	2,96	2539	0,95	0,95	0,00	0,81	1
g_4x25_b	0,00	0,00	0,00	2,04	386	0,00	0,00	0,00	0,38	1
g_5x20_a	0,63	0,63	0,00	3,48	1335	0,63	0,63	0,00	0,61	1
g_5x20_b	0,00	0,00	0,00	1,66	377	0,00	0,00	0,00	0,29	1
g_6x17_a	0,00	0,00	0,00	2,32	92	0,00	0,00	0,00	0,31	1
g_6x17_b	0,00	0,00	0,00	0,87	306	0,00	0,00	0,00	0,20	1
g_7x14_a	0,00	0,00	0,00	1,29	368	0,00	0,00	0,00	0,30	1
g_7x14_b	1,49	1,49	0,00	2,03	464	1,49	1,49	0,00	0,63	2
g_4x50_a	-98,99	-	-	7200,00	176946	33,31	45,19	11,89	7200,00	333410
g_4x50_b	-87,98	34,36	122,34	7200,00	79988	27,98	34,36	6,38	7200,00	267725
g_5x40_a	-95,05	-	-	7200,00	163330	33,54	40,60	7,06	7200,00	365738
g_5x40_b	-92,57	-	-	7200,00	153183	28,03	28,03	0,00	1029,16	36778
g_6x34_a	-89,60	33,64	123,24	7200,00	239289	33,64	33,64	0,00	327,48	23130
g_6x34_b	-90,15	-	-	7200,00	111683	39,99	55,58	15,59	7200,00	497787
g_7x29_a	-89,81	44,99	134,80	7200,00	155818	44,98	44,99	0,00	3994,81	298610
g_7x29_b	-93,82	56,14	149,95	7200,00	246237	45,09	54,27	9,49	7200,00	323021
g_8x25_a	-62,99	26,88	89,86	7200,00	223165	26,88	26,88	0,00	99,96	5566
g_8x25_b	-82,46	56,14	138,60	7200,00	314228	40,77	40,77	0,00	439,39	34039
Average			47,43					2,52		

instâncias são mostrados na coluna 1. O limite inferior (lb) e o limite superior (ub) para o valor ótimo da solução obtida pelo CPLEX (V1) são dados na coluna 2 e 3, respectivamente. O gap de integralidade, o tempo de execução e o número de nós explorados na árvore de *branch and bound* são reportados nas colunas de 4 a 6. Os mesmos dados são mostrados para o CPLEX (V2) nas últimas cinco colunas, respectivamente. Pode ser visto que as inequações válidas (2.27) e (2.28) melhoram muito o limite inferior da relaxação linear. Para as instâncias Karasan (Tabela 2.3), CPLEX (V1) resolveu somente uma instância, enquanto CPLEX (V2) conseguiu resolver seis (das 20) instâncias antes de 7200 segundos. A média relativa do gap de integralidade do CPLEX (V1) foi 1051,98%, enquanto que para o CPLEX (V2) foi apenas 16,68%. Para as instâncias grid (Tabela 2.4), nenhuma instância foi resolvida pelo CPLEX (V1) e CPLEX (V2) antes de 7200 segundos. Entretanto, o CPLEX (V2) encontrou soluções inteiras viáveis para todas as 10 instâncias com uma média de gap igual a 18,78%, enquanto que o CPLEX (V1) não encontrou soluções para nenhuma instância desse conjunto. O fato que o CPLEX (V2) encontrou soluções com quase 20% de média para o gap com até 7200 segundos de execução motiva o estudo de heurísticas para enfrentar grandes instâncias do *minmax relative regret* RSP.

Tabela 2.3: Comparação entre CPLEX (V1) e CPLEX (V2) para grafos Karasan com 1000 e 1500 nós.

	CPLEX (V1)					CPLEX (V2)				
Instance	lb (%)	ub (%)	gap (%)	t(s)	Nodes	lb (%)	ub (%)	gap (%)	t(s)	Nodes
K-1000-200-0.9-a-5	-95,48	68,86	164,34	7200,00	20799	38,70	61,49	22,79	7200,00	22155
K-1000-200-0.9-b-5	-100,00	87,93	187,93	7200,00	14008	40,87	67,35	26,48	7200,00	44525
K-1000-200-0.9-a-10	-100,00	106,97	206,97	7200,00	17959	49,26	93,05	43,79	7200,00	76124
K-1000-200-0.9-b-10	-94,75	100,76	195,50	7200,00	41556	44,56	76,74	32,19	7200,00	62477
K-1000-200-0.9-a-25	-98,63	142,36	240,99	7200,00	36759	53,57	65,07	11,50	7200,00	164165
K-1000-200-0.9-b-25	-94,43	182,19	276,62	7200,00	19798	53,71	77,93	24,22	7200,00	96155
K-1000-200-0.9-a-50	-73,13	83,72	156,86	7200,00	21969	68,09	68,09	0,00	1100,99	10967
K-1000-200-0.9-b-50	-65,75	65,91	131,66	7200,00	24498	65,91	65,91	0,00	754,09	5871
K-1000-200-0.9-a-100	84,21	84,21	0,00	2432,07	1851	84,21	84,21	0,00	1521,14	2261
K-1000-200-0.9-b-100	-8,53	95,00	103,53	7200,00	13828	90,00	90,00	0,00	1716,96	1707
K-1500-200-0.9-a-5	-99,72	103,72	203,44	7200,00	2175	34,16	53,13	18,97	7200,00	3441
K-1500-200-0.9-b-5	-98,47	97,96	196,43	7200,00	3204	30,93	45,20	14,27	7200,00	5230
K-1500-200-0.9-a-10	-99,11	105,50	204,61	7200,00	3158	36,58	57,71	21,13	7200,00	5818
K-1500-200-0.9-b-10	-99,61	69,56	169,17	7200,00	9404	36,58	56,39	19,81	7200,00	10923
K-1500-200-0.9-a-25	-93,05	265,97	359,02	7200,00	8026	38,65	52,88	14,23	7200,00	41947
K-1500-200-0.9-b-25	-100,00	283,33	383,33	7200,00	8626	48,47	87,57	39,10	7200,00	9560
K-1500-200-0.9-a-50	-77,73	66,67	144,40	7200,00	4338	59,26	59,26	0,00	2197,76	9618
K-1500-200-0.9-b-50	-100,00	11911,43	12011,43	7200,00	9	52,65	75,51	22,86	7200,00	37770
K-1500-200-0.9-a-100	-99,35	1171,43	1270,78	7200,00	1169	81,25	81,25	0,00	4934,37	570
K-1500-200-0.9-b-100	-77,06	4355,56	4432,62	7200,00	9	83,60	105,88	22,28	7200,00	20884
Average			1051,98					16,68		

Tabela 2.4: Comparação entre CPLEX (V1) e CPLEX (V2) para grafos grid com no máximo com 1000 nós.

	CPLEX (V1)					CPLEX (V2)				
Instance	lb (%)	ub (%)	gap (%)	t(s)	Nodes	lb (%)	ub (%)	gap (%)	t(s)	Nodes
g_6x60_a	-98,67	-	-	7200,00	44388	30,86	42,45	11,59	7200,00	383891
g_6x60_b	-99,08	-	-	7200,00	35895	35,75	50,02	14,28	7200,00	174914
g_7x70_a	-100,00	-	-	7200,00	9720	37,62	58,78	21,17	7200,00	80621
g_7x70_b	-100,00	-	-	7200,00	15996	35,60	53,51	17,92	7200,00	116668
g_8x80_a	-100,00	-	-	7200,00	9507	32,24	46,74	14,50	7200,00	52630
g_8x80_b	-98,72	-	-	7200,00	9797	35,09	52,75	17,66	7200,00	60023
g_9x90_a	-98,43	-	-	7200,00	13802	33,20	50,64	17,44	7200,00	35137
g_9x90_b	-98,94	-	-	7200,00	5997	34,99	51,56	16,57	7200,00	46045
g_10x100_a	-100,00	-	-	7200,00	1273	37,20	57,07	19,87	7200,00	18444
g_10x100_b	-98,66	-	-	7200,00	2425	32,98	47,42	14,45	7200,00	24643
Average			-					18,78		

Capítulo 3

Problema de Localização de Concentradores em Árvore

Este capítulo é dedicado ao estudo do problema de localização de concentradores em árvore e está organizado da seguinte forma: Na seção 3.1 são discutidos trabalhos relacionados ao THLP. Na seção 3.2 são apresentas duas formulações lineares inteiras mistas presentes na literatura para o THLP. Já na seção 3.3 é descrito o algoritmo de decomposição de Benders para esse problema, proposto por de Sá et. al.[56]. Em seguida, na seção 3.4 o algoritmo proposto neste trabalho para resolver os subproblemas da decomposição de Benders de forma eficiente é apresentado. Finalmente, na seção 3.5, são apresentados os experimentos computacionais.

3.1 Trabalhos Relacionados

O'Kelly apresenta em [49] uma formulação inteira quadrática e duas heurísticas para o problema de localização de concentradores (do inglês, p-hub location pro-

blem). Em [39], Klincewicz apresenta várias heurísticas para o problema de localização de concentradores e resultados computacionais foram realizados para comparar essas heurísticas com as apresentadas em [49]. Em [53] os autores apresentam uma linearização da formulação proposta em [49] para o problema de localização de concentradores com alocação simples. Resultados computacionais mostraram que essa linearização é capaz de resolver instâncias de até 25 nós para o problema de localização de concentradores com alocação simples. Em [24], os autores apresentam um algoritmo de branch and bound capaz de resolver instâncias de até 100 nós, na versão de alocação simples, e 200 nós na versão de alocação múltipla do problema de localização de concentradores.

Em [11], Campbell, apresenta formulações para quatro tipos de problemas de localização de concentradores. Além disso, segundo [18], esse trabalho é o primeiro a apresentar uma formulação para o problema de localização de concentradores com alocação múltipla. Em [38], Klincewicz apresenta o dual da formulação do problema de alocação de concentradores não capacitado, presente em [11], e propõe um algoritmo de branch and bound para essa formulação dual. O algoritmo proposto é capaz de resolver instâncias de até 25 nós do problema de localização de concentradores não capacitado. Em [53], os autores apresentam uma nova formulação para o problema de localização de concentradores com alocação múltipla. Essa formulação é capaz de resolver instâncias de até 25 nós para esse problema. Os resultados computacionais também mostraram que essa formulação possui uma excelente relaxação linear. A relaxação linear apresentou gap de no máximo 1%. Em [25], os autores apresentam três novas formulações para o problema de localização de concentradores com alocação múltipla. Os testes computacionais mostraram que essas formulações são capazes de resolver instâncias de até 25 nós. Em [43], os

autores apresentam um algoritmo de branch and bound capaz de resolver instâncias de até 40 nós para o problema de localização de concentradores com alocação múltipla. Em [31], os autores investigaram o poliedro das soluções viáveis do problema de localização de concentradores com alocação múltipla e apresentam uma nova formulação para esse problema. Experimentos computacionais não foram apresentados nesse trabalho. Em [42], os autores apresentam novas formulações para o problema de localização de concentradores com alocação múltipla que são capazes de resolver instâncias de até 30 nós. Marín propõe, em [41], um algoritmo de branch and cut que resolve instância de até 50 nós para o problema de localização de concentradores euclidiano com alocação múltipla. Em [20], os autores propõe um algoritmo de branch and bound capaz de resolver instâncias de até 120 nós para o problema de localização de concentradores com alocação múltipla. Em [30], os autores apresentam uma formulação inteira com $O(|N|)^4$ variáveis para o problema de localização de concentradores com alocação múltipla. Eles mostram uma comparação do número de variáveis e restrições com outras formulações presentes na literatura para esse problema. Um algoritmo baseado em decomposição de Benders foi proposto e experimentos computacionais mostraram que esse algoritmo foi capaz de resolver instâncias de até 50 nós para esse problema. Em [27] os autores apresentam um survey de recentes publicações para problemas de localização de concentradores. Esse trabalho contêm modelos matemáticos e aplicações no mundo real desses problemas.

O THLP foi proposto por Contreras et. al.[18]. Nesse traballho os autores provam que o problema é *NP-Difícil* e apresentam uma formulação linear inteira mista contendo $O(|N|^2)$ variáveis binárias, $O(|N|^3)$ variáveis contínuas e $O(|N|^3)$ restrições. Com essa formulação é possível resolver instâncias de THLP com até

25 nós, com um tempo computacional de até 40 horas.

Em [17], Contreras et. al. apresentam outra MILP baseada em caminhos entre cada par de nós para o THLP. Essa formulação possui $O(|N|^2)$ variáveis binárias, $O(|N|^4)$ variáveis contínuas e $O(|N|^4)$ restrições. Um algoritmo de relaxação Lagrangiana e uma heurística para o THLP também são propostos. Os resultados computacionais mostram que os gaps obtidos pela relaxação Lagrangiana são de no máximo 10% para instâncias com até 100 nós. Entretanto, os tempos computacionais para resolver essa relaxação chega a 5200 segundos.

O algoritmo de decomposição de Benders, proposto por de Sá et. al.[56], resolve instâncias do THLP com até 100 nós. Esse algoritmo é baseado na formulação proposta por Contreras et. al.[17]. Os resultados computacionais mostram que o tempo computacional consumido para resolver instâncias de 100 nós para o THLP com esse algoritmo de decomposição de Benders é superior a 80 horas. Uma heurística baseda em algoritmos genéticos com chaves aleatórias para o THLP é apresentada por Pessoa et. al.[51]. As abordagens de [17], [18] e [56] são o estado da arte para o THLP e são detalhadas a seguir.

3.2 Formulações de Programação Linear Inteira Mista para o THLP

As duas formulações de programação linear inteira mista encontradas na literatura de THLP são detalhadas nesta seção. Nessas formulações as constantes W_{ij} representam a quantidade de fluxo que deve ser enviada do nó de origem $i \in N$ para o nó de destino $j \in N$ e as constantes c_{ij} representam o custo do arco $(i, j) \in A$.

Já as constantes $O_i = \sum_{j \in N} W_{ij}$, para todo $i \in N$, representam a soma dos fluxos com origem nó $i \in N$ e com destino em todos os outros nós do conjunto N, enquanto as constantes $D_i = \sum_{j \in N} W_{ji}$, para todo $i \in N$, representam a soma dos fluxos enviados a partir de todos os nós em N com destino ao nó $i \in N$. Além disso, a constante $0 \leq \alpha \leq 1$ representa o fator de desconto que será aplicado ao custo c_{km} quando os dois nós k, m do arco $(k, m) \in A$ são nós concentradores.

A primeira formulação descrita foi proposta por Contreras et. al.[18] e contém $O(|N|^2)$ variáveis binárias, $O(|N|^3)$ variáveis contínuas e $O(|N|^2)$ restrições. Nessa formulação as variáveis z_{kk} modelam os nós escolhidos como nós concentradores, portanto, elas são iguais a 1 quando o nó k é escolhido como nó concentrador e 0 caso contrário. As variáveis $y_{km} \in \{0, 1\}$ modelam a árvore de conexão dos nós concentradores. Essas variáveis são iguais a 1 quando os nós concentradores k e m estão conectados e 0 caso contrário. Já as variáveis z_{ik}, com $i \neq k$, modelam a alocação do nó cliente i a um nó concentrador k. Essas variáveis são iguais a 1 quando o nó cliente $i \in N$ está alocado ao nó concentrador $k \in N$, e 0 caso contrário. As variáveis $x_{ikm} \in \mathbb{R}_+$ modelam a quantidade de fluxo com origem no nó $i \in N$ e com destino a todos os outros nós do conjunto N, transportado através do arco $(k, m) \in A$, onde $k, m \in N$ são nós concentradores e estão conectados, isto é, $y_{km} = 1$.

A função objetivo (3.1) minimiza o custo de transportar os fluxos dos nós clientes até os nós concentradores somado ao custo de transportar os fluxos entre os nós concentradores, aplicando o fator de desconto. A restrição (3.2) garante que todo nó cliente $i \in N$ vai ser alocado a somente um nó concentrador $k \in N$. Desta forma, a alocação é garantida ser simples. A restrição (3.3) garante que exatamente p nós do conjunto N são escolhidos como nós concentradores. As

restrições (3.4) e (3.5) conectam as variáveis y e z. Elas garantem que nós clientes só podem ser alocados a nós concentradores e que a árvore de conexão, definida pelas variáveis y_{km}, só exista entre os nós concentradores. Assim essas restrições impõem $y_{km} = 0$, caso $k \in N$ ou $m \in N$ não sejam concentradores. Quando $y_{km} = 1$, a restrição (3.6) garante que a quantidade de fluxo com origem no nó $i \in N$, transportado através do arco (k, m), é limitado pela oferta de fluxo com origem no nó i. Já quando $y_{km} = 0$, essa restrição garante que o fluxo com origem no nó $i \in N$ não será transportado através do arco (k, m). As restrições (3.7) garantem a conservação de fluxo com origem no nó $i \in N$ e destino no nó $k \in N$. O lado esquerdo dessas restrições corresponde ao fluxo com origem no nó $i \in N$ transportado diretamente para o nó $k \in N$, caso i seja um nó cliente e esteja alocado para o nó concentrador k, somado com o fluxo com origem no nó $i \in N$ e que é transportado até o nó concentrador $k \in N$ através dos outros nós concentradores $m \in N$. Já o lado direito dessas restrições corresponde ao fluxo com origem no nó $i \in N$, transportado através do nó concentrador k para os outros nós concentradores, somado com o fluxo que tem origem no nó $i \in N$ e destino a todos os outros nós clientes que estão alocados ao nó concentrador $k \in N$. A restrição (3.8) garante o que p - 1 arestas conectam os nós concentradores. Por fim, as restrições (3.9) e (3.10) definem o domínio das variáveis x_{ikm}, z_{km} e y_{km}.

$$\min \quad \sum_{i\in N}\sum_{k\in N}(c_{ik}O_i + c_{ki}D_i)z_{ik} + \sum_{i\in N}\sum_{k\in N}\sum_{m\in N, m\neq k}\alpha c_{km}x_{ikm} \tag{3.1}$$

sujeito a:

$$\sum_{k\in N} z_{ik} = 1 \quad \forall i \in N \tag{3.2}$$

$$\sum_{k\in N} z_{kk} = p \tag{3.3}$$

$$z_{km} + y_{km} \leq z_{mm} \quad \forall k, m \in N; m > k \tag{3.4}$$

$$z_{mk} + y_{km} \leq z_{kk} \quad \forall k, m \in N; m > k \tag{3.5}$$

$$x_{ikm} + x_{imk} \leq O_i y_{km} \quad \forall i, k, m \in N; m > k \tag{3.6}$$

$$O_i z_{ik} + \sum_{m\in N, m\neq k} x_{imk} = \sum_{m\in N, m\neq k} x_{ikm} + \sum_{m\in N} W_{im} z_{mk} \quad \forall i, k \in N; k \neq i \tag{3.7}$$

$$\sum_{k\in N}\sum_{m\in N} y_{km} = p - 1 \tag{3.8}$$

$$x_{ikm} \geq 0 \quad \forall i, k, m \in N \tag{3.9}$$

$$z_{km}, y_{km} \in \{0, 1\} \quad \forall k, m \in N \tag{3.10}$$

A segunda formulação apresentada para o THLP, proposta por Contreras et. al.[17], contém $O(|N|^2)$ variáveis binárias, $O(|N|^4)$ variáveis contínuas e $O(|N|^4)$ restrições. Assim como na formulação anterior as variáveis z_{kk} indicam os nós concentradores, as variáveis z_{ik}, com $i \neq k$, representam as alocações dos nós clientes aos nós concentradores e as variáveis y_{km} definem as arestas da árvore que conecta os nós concentradores. Já as variáveis x possuem outro significado nessa formulação. As variáveis x_{ij}^{km} definem o caminho pelo qual o fluxo com origem no nó $i \in N$ e destino no nó $j \in N$ é transportado. Quando $x_{ij}^{km} = 1$, o fluxo com origem

no nó i e destino no nó j é transportado através do arco (k, m), caso contrário $x_{ij}^{km} = 0$. A formulação é apresentada em (3.11)-(3.24). A função objetivo (3.11) minimiza a soma (i) dos custos de transportar o fluxo dos nós clientes até os nós concentradores com (ii) os custos de transportar o fluxo, com o fator de desconto aplicado, entre os nós concentradores. As restrições (3.12) e (3.13) garantem que todo nó cliente $i \in N$ será alocado a exatamente um nó concentrador $k \in N$. As restrições (3.14) e (3.15) definem que y_{km} só pode pertencer a árvore que conecta os nós concentradores, quando k e m são nós concentradores.

A restrição (3.16) assegura que exatamente p nós do conjunto N são selecionados para serem nós concentradores enquanto a restrição (3.17) garante que esses nós concentradores sejam conectados através de $p-1$ arestas, definidas pelas variáveis y. As restrições (3.18) garantem que o fluxo com origem no nó i e destino no nó j será transportado através do único caminho que conectam esses nós. As restrições (3.19) garantem que o fluxo com origem no nó $i \in N$ e destino no nó $j \in N$ pode ser transportado através do arco (k, m), se k e m são nós concentradores e estão conectados. As restrições (3.20) e (3.21) estabelecem que o fluxo com origem no nó $i \in N$ e destino no nó $j \in N$ só pode ser transportado pelo arco (k, m), se os nós $k, m \in N$ são nós concentradores. O domínio das variáveis x, y e z é dado, respectivamente, nas restrições de (3.22) - (3.24). As restrições (3.14), (3.15), (3.20) e (3.21) não estão presentes na formulação proposta por Contreras et. al.[17], elas foram adicionandos por de Sá et. al.[56]. Essas restrições são redundantes porém melhoram o limite inferior fornecido pela relaxação linear [56].

$$\min \quad \sum_{i \in N} \sum_{\substack{k \in N \\ k \neq i}} (c_{ik} O_i + c_{ki} D_i) z_{ik} +$$

$$\sum_{i \in N} \sum_{\substack{j \in N \\ j > i}} \sum_{k \in N} \sum_{\substack{m \in N \\ m \neq k}} (c_{km} W_{ij} + c_{mk} W_{ji}) \times \alpha \times x_{ij}^{km} \tag{3.11}$$

sujeito a:

$$z_{ik} \leq z_{kk} \qquad \forall i, k \in N : i \neq k \tag{3.12}$$

$$\sum_{k \in N} z_{ik} = 1 \qquad \forall i \in N \tag{3.13}$$

$$y_{km} \leq z_{kk} \qquad \forall k, m \in N : k < m \tag{3.14}$$

$$y_{km} \leq z_{mm} \qquad \forall k, m \in N : k < m \tag{3.15}$$

$$\sum_{k \in N} z_{kk} = p \tag{3.16}$$

$$\sum_{k \in N} \sum_{\substack{m \in N \\ m > k}} y_{km} = \sum_{k \in N} z_{kk} - 1 \tag{3.17}$$

$$\sum_{\substack{k \in N \\ k \neq m}} x_{ij}^{km} + z_{im} = \sum_{\substack{r \in N \\ r \neq m}} x_{ij}^{mr} + z_{jm} \qquad \forall i, j, m \in N : i < j, m \qquad (3.18)$$

$$x_{ij}^{km} + x_{ij}^{mk} \leq y_{km} \qquad \forall i, j, k, m \in N : i < j, k < m \qquad (3.19)$$

$$\sum_{\substack{m \in N \\ m \neq k}} x_{ij}^{km} \leq z_{kk} \qquad \forall i, j, k \in N : i < j, k \qquad (3.20)$$

$$\sum_{\substack{m \in N \\ m \neq k}} x_{ij}^{mk} \leq z_{kk} \qquad \forall i, j, k \in N : i < j, k \qquad (3.21)$$

$$x_{ij}^{km} \geq 0 \qquad \forall i, j, k, m \in N : i < j, k \neq m \qquad (3.22)$$

$$y_{km} \in \{0, 1\} \qquad \forall k, m \in N : k < m \qquad (3.23)$$

$$z_{ik} \in \{0, 1\} \qquad \forall i, k \in N \qquad (3.24)$$

3.3 Algoritmo de Decomposição de Benders

Nesta seção é descrito o algoritmo de decomposição de Benders para o THLP proposto por de Sá et. al.[56]. Essa decomposição de Benders utiliza a formulação linear inteira mista apresentada em [17] pois, segundo de Sá et. al.[56], essa formulação apresenta uma relaxação linear melhor quando comparada com a relaxação linear da formulação apresentada em [18]. Entretanto, os tempos computacionais para resolver essa relaxação em instâncias de THLP com até 25 nós chega a 2300 segundos [17].

A decomposição de Benders proposta em [7] consiste em dividir um problema de programação linear ou inteira em dois problemas. Um problema é denominado problema mestre e o outro é conhecido como subproblema. Um algoritmo de decomposição de Benders consiste em encontrar a solução do problema mestre, essa solução é um parâmetro de entrada para o subproblema. A partir da solução do

problema mestre o algoritmo de decomposição de Benders resolve o subproblema. Em seguida, cortes válidos obtidos através da solução dos subproblemas são adicionados ao problema mestre que é resolvido novamente. Esse ciclo se repete até que a solução ótima do problema seja encontrada ou outro critério de parada seja satisfeito. Os cortes adicionados ao problema mestre, a partir da solução do subproblema, são cortes de otimalidade ou de viabilidade. Sempre que o subproblema possuir uma solução ilimitada, o corte adicionado ao problema mestre será um corte de viabilidade. Já quando o subproblema possuir solução ótima, o corte adicionado ao problema mestre será um corte de otimalidade.

A decomposição de Benders proposta em [56] para a modelagem do THLP apresentada em [17] consiste em manter as variáveis z e y no problema mestre e as variáveis x no subproblema, tornando o subproblema um problema de programação linear. Com essa separação de variáveis, um dos objetivos do problema mestre é encontrar uma árvore conectando os nós concentradores e os nós clientes. Enquanto o objetivo do subproblema é encontrar o valor mínimo de transportar o fluxo entre todos os pares de nós $i, j \in N$, com $i < j$. Esse subproblema possui $O(|N|^4)$ variáveis lineares e pode ser decomposto em $O(|N|^2)$ subproblemas independentes, um para cada par de nós $i, j \in N$. Cada um desses subproblemas possui $O(|N^2|)$ variáveis lineares. Assim, o objetivo de cada um deles é encontrar o valor mínimo de transportar o fluxo entre cada par de nós $i, j \in N$. Vale ressaltar que nos subproblemas só existe um caminho entre cada par de nós $i, j \in N$, pois os nós estão conectados por uma árvore, o que facilita a resolução desses subproblemas. Na próxima seção, essa característica é explorada para criar um algoritmo eficiente para resolver os subproblemas.

Os subproblemas, um para cada par de nós $i, j \in N$, com $i < j$, são forma-

dos pela função objetivo (3.25) e as restrições (3.26)-(3.30). Essas restrições são semelhantes, respectivamente, as restrições (3.18)-(3.22). A diferença entre elas é que as restrições (3.26)-(3.30) possuem constantes $\overline{z}$ e $\overline{y}$ enquanto as restrições (3.18)-(3.22) possuem variáveis z e y.

$$\text{min Q(i, j)} = \sum_{k \in N} \sum_{\substack{m \in N \\ m \neq k}} (c_{km} W_{ij} + c_{mk} W_{ji}) \times \alpha \times x_{ij}^{km} \tag{3.25}$$

sujeito a:

$$\sum_{\substack{k \in N \\ k \neq m}} x_{ij}^{km} - \sum_{\substack{r \in N \\ r \neq m}} x_{ij}^{mr} = \overline{z}_{jm} - \overline{z}_{im} \qquad \forall m \in N \tag{3.26}$$

$$x_{ij}^{km} + x_{ij}^{mk} \leq \overline{y}_{km} \qquad \forall k, m \in N : k < m \tag{3.27}$$

$$\sum_{\substack{m \in N \\ m \neq k}} x_{ij}^{km} \leq \overline{z}_{kk} \qquad \forall k \in N \tag{3.28}$$

$$\sum_{\substack{m \in N \\ m \neq k}} x_{ij}^{mk} \leq \overline{z}_{kk} \qquad \forall k \in N \tag{3.29}$$

$$x_{ij}^{km} \geq 0 \qquad \forall k, m \in N : k \neq m \tag{3.30}$$

Após associar as variáveis u_{ijm}, e_{ijkm}, s_{ijk} e t_{ijm}, respectivamente, às restrições (3.26)-(3.29), o dual do subproblema para cada par de nós $i, j \in N$, com $i < j$, é escrito com a função objetivo (3.31) e as restrições (3.32)-(3.37).

$$\max \overline{Q}(\text{i, j}) = \sum_{m \in N} (\overline{z}_{jm} - \overline{z}_{im}) u_{ijm} - \sum_{k \in N} \sum_{\substack{m \in N \\ k<m}} \overline{y}_{km} e_{ijkm} - \sum_{k \in N} \overline{z}_{kk} (s_{ijk} + t_{ijk}) \tag{3.31}$$

sujeito a:

$$u_{ijm} - u_{ijk} - e_{ijkm} - s_{ijk} - t_{ijm} \leq \alpha(c_{km} W_{ij} + c_{mk} W_{ji}) \quad \forall k, m \in N : k < m \tag{3.32}$$

$$u_{ijm} - u_{ijk} - e_{ijmk} - s_{ijk} - t_{ijm} \leq \alpha(c_{km} W_{ij} + c_{mk} W_{ji}) \quad \forall k, m \in N : k > m \tag{3.33}$$

$$u_{ijm} \in \mathbb{R} \qquad \forall m \in N \tag{3.34}$$

$$e_{ijkm} \geq 0 \qquad \forall k, m \in N : k < m \tag{3.35}$$

$$t_{ijm} \geq 0 \qquad m \in N \tag{3.36}$$

$$s_{ijm} \geq 0 \qquad \forall m \in N \tag{3.37}$$

Já o problema mestre é formado pela função objetivo (3.38) e as restrições (3.12)-(3.17), (3.23), (3.24) e (3.39)-(3.41). Nesse problema as variáveis η_{ij}, uma para cada subproblema, representa o custo da solução ótima do subproblema. As restrições (3.39) e (3.40) são, respectivamente, os cortes de otimalidade e viabilidade adicionados a partir de uma solução do dual do subproblema. Nessas restrições, os conjuntos H e G são, respectivamente, o conjunto de cortes de otimalidade e de viabilidade.

O problema mestre não garante a formação de uma árvore conectando os nós concentradores porque a restrição de conservação de fluxo não está presente nesse

problema. Quando os nós concentradores não são conectados por meio de uma árvore na solução do problema mestre, os subproblemas são ilimitados [56]. Assim, cortes de viabilidade devem ser adicionados ao problema mestre. Esses cortes não ajudam a melhorar o limite inferior da solução do THLP e portanto devem ser evitados [56]. Para evitar que os subproblemas sejam ilimitados as restrições (3.42)-(3.50) são adicionadas ao problema mestre. Essas restrições garantem que a solução do problema mestre forme uma árvore conectando os nós concentradores. A ideia dessas restrições é formar uma árvore a partir de um novo nó raiz até todos os nós concentradores. Para isso, o nó raiz possui p unidades de fluxo que devem ser enviadas para os nós concentradores, sendo que cada nó concentrador deve receber exatamente uma unidade de fluxo.

$$\min \quad \sum_{i\in N}\sum_{k\in N,k\neq i}(c_{ik}O_i + c_{ki}D_i)z_{ik} + \sum_{i\in N}\sum_{j\in N,j>i}\eta_{ij} \tag{3.38}$$

sujeito a:

$(3.12) - (3.17), (3.23), (3.24)$

$$\eta_{ij} \geq \sum_{m\in N}(z_{jm} - z_{im})u^h_{ijm} - \sum_{k\in N}\sum_{\substack{m\\ k<m}\in N} y_{km}e^h_{ijkm}$$

$$-\sum_{k\in N} z_{kk}(s^h_{ijk} + t^h_{ijk}) \forall i,j \in N, h \in H : i < j \tag{3.39}$$

$$\sum_{m\in N}(z_{jm} - z_{im})u^h_{ijm} - \sum_{k\in N}\sum_{\substack{m\\ k<m}\in N} y_{km}e^h_{ijkm} -$$

$$\sum_{k\in N} z_{kk}(s^h_{ijk} + t^h_{ijk}) \leq 0 \quad \forall i < j \quad i,j \in N \quad h \in G \tag{3.40}$$

$$\eta_{ij} \geq 0 \quad \forall i < j \quad i,j \in N \tag{3.41}$$

Nas restrições (3.42)-(3.50), as variáveis q_k indicam que o novo nó raiz, denominado por 0, está conectado ao nó concentrador k, quando q_k igual a 1, e 0 caso contrário. As variáveis f_{km}, com $k \neq m$, representam o fluxo transportado através do arco $(k, m) \in A$, onde k e m são nós concentradores. A restrição (3.42) garante que a quantidade de fluxo ofertada pelo nó raiz é igual ao número de nós concentradores alocados. As restrições (3.43) são as de conservação de fluxo e garantem que a quantidade de fluxo que permanece em cada nó concentrador k é igual a 1. As restrições (3.44) e (3.45) garantem o limite superior de fluxo nas variáveis f_{km}. As restrições (3.46) garantem o limite superior de fluxo que será ofertado para o nó concentrador k. A restrição (3.47) garante que o nó raiz só está conectado a um nó. As restrições (3.48) garantem que o nó raiz só pode se conectar a nós concentradores. As restrições (3.49) e (3.50) definem os domínios das variáveis f_{km} e q_k, respectivamente.

$$\sum_{m \in N} f_{0m} = \sum_{k \in N} z_{kk} \tag{3.42}$$

$$\sum_{m \in N \cup \{0\}: m \neq k} f_{mk} - \sum_{m \in N: m \neq k} f_{km} = z_{kk} \qquad \forall k \in N \tag{3.43}$$

$$f_{km} \leq |N| y_{km} \qquad \forall k, m \in N : k < m \tag{3.44}$$

$$f_{km} \leq |N| y_{mk} \qquad \forall k, m \in N : k > m \tag{3.45}$$

$$f_{0k} \leq |N| q_k \qquad \forall k \in N \tag{3.46}$$

$$\sum_{k \in N} q_k = 1 \tag{3.47}$$

$$q_k \leq z_{kk} \qquad \forall k \in N \tag{3.48}$$

$$f_{km} \geq 0 \qquad \forall k \in N \cup \{0\}, m \in N : k \neq m \tag{3.49}$$

$$q_k \in \{0, 1\} \qquad \forall k \in N \tag{3.50}$$

O algoritmo de decomposição de Benders para o THLP proposto em [56] é divido em duas fases: a primeira fase, denominada aquecimento (do inglês *warm-start*), tem o objetivo de acelerar a convergência do algoritmo de Benders. Para isso, essa fase resolve a relaxação linear do problema mestre. A segunda fase do algoritmo, denominada inteira (do inglês *integer*), tem como objetivo encontrar a solução ótima para o THLP.

Um pseudo-código da fase de aquecimento é apresentada no Algoritmo 1. Nesse algoritmo, a variável LB é o limite inferior do problema obtido pela resolução do problema mestre, a variável h é a quantidade de iterações que o algoritmo vai executar e a variável λ é um multiplicador utilizado para gerar uma nova solução viável não inteira para o problema mestre. Na linha 2 desse algoritmo, as variáveis

são inicializadas e na linha 3 é gerada uma solução inicial (z^0, y^0), onde $z^0_{kk} = \frac{1}{2}$ para todo nó $k \in N$, $z^0_{ik} = \frac{1}{2|N|-2}$ para toda alocação do nó $i \in N$ para o nó $k \in N$, com $i \neq k$ e $y^0_{km} = \frac{|N|-2}{|N|^2-|N|}$ para todo $k, m \in N$, com $k < m$. Essa solução inicial é uma solução viável não inteira para o problema mestre. O laço das linhas 4 - 19 é executado cinco vezes. Nas linhas 5, 6 e 7, os seguintes passos são realizados, respectivamente: (i) os subproblemas com o parâmetro (z^0, y^0) são resolvidos, (ii) os cortes de otimalidade são adicionados ao problema mestre que (iii) é então resolvido. Em seguida, nas linhas 8 e 9 são atualizados respectivamente, a solução do problema mestre e o valor do limite inferior do THLP, obtido pela solução do problema mestre. Posteriormente, na linha 10, os subproblemas parametrizados com a solução atual do problema mestre são resolvidos. As linhas 12 e 13 são executadas se o subproblema for ilimitado, caso contrário as linhas 15 e 16 serão executadas. A linha 12 adiciona os cortes de viabilidade no problema mestre e a linha 13 encontra o melhor valor de λ para gerar uma nova solução viável a partir de (z^0, y^0). A linha 15 adiciona os cortes de otimalidade no problema mestre e a linha 16 fixa o valor do parâmetro λ. A linha 18 atualiza a solução inicial (z^0, y^0). Essa nova solução é uma combinação linear convexa da solução (z^0, y^0) com a solução atual (z, y) do problema mestre. A linha 19 atualiza o número de iterações executadas.

Já para a fase inteira é apresentado um pseudo-código no Algoritmo 2. Nesse algoritmo as variáveis LB e UB representam, respectivamente, os valores dos limites inferiores e superiores do THLP. O limite inferior é obtido pelo valor da função objetivo do problema mestre enquanto o limite superior é a soma entre: (i) o valor da função objetivo do problema mestre e (ii) a soma da função objetivo de cada um dos subproblemas. A variável h contabiliza o número de iterações necessárias para

```
1  begin
2      LB ← 0, h ← 0
3      Encontrar uma solução inicial (z^0, y^0)
4      while h < 5 do
5          Resolver subproblemas com (z^0, y^0)
6          Adicionar os cortes (3.39) para o problema mestre
7          Resolver o problema mestre
8          (z^h, y^h) ← Solução do problema mestre
9          LB ← Valor da solução do problema mestre
10         Resolver os subproblemas com (z^h, y^h)
11         if Subproblema é ilimitado then
12             Adicionar os cortes (3.40) no problema mestre
13             Encontrar o melhor valor para λ
14         else
15             Adicionar os cortes (3.39) no problema mestre
16             λ ← 0.5
17         end
18         (z^0, y^0) ← (1 - λ)(z^0, y^0) + λ(z^h, y^h)
19         h ← h + 1
20     end
21 end
```

Algoritmo 1: Fase aquecimento do algoritmo de decomposição de Benders para o THLP

resolver o problema na otimalidade e a variável λ é o multiplicador para atualizar a solução inicial (z^0, y^0). Na linha 2 deste algoritmo as variáveis são inicializadas. Na linha 3, uma solução inicial para as variáveis y e z é retornada pelo algoritmo de aquecimento. O laço das linhas 4 - 15 é executado até que o limite superior seja igual ao limite inferior, caracterizando assim a solução ótima para o THLP. A linha 5 resolve o subproblema considerando a solução z^0 e y^0 do problema mestre. Os cortes de otimalidade são adicionados na linha 6. Em seguida, o problema mestre é resolvido e a solução encontrada é retornada respectivamente nas linhas 7 e 8. O limite inferior para o THLP é atualizado na linha 9. A linha 10 resolve o

subproblema utilizando a solução atual do problema mestre. A linha 11 adiciona cortes de otimalidade no problema mestre. A linha 12 atualiza a solução inicial (z^0, y^0). A linha 13 atualiza, caso seja necessário, o limite superior da solução e a linha 14 atualiza o número de iterações do algoritmo.

```
1  begin
2      UB ← ∞, LB ← 0, h ← 0
3      (z^0, y^0) ← solução retornada pelo algoritmo da fase aquecimento
4      while UB ≠ LB do
5          Resolver subproblemas com (z^0, y^0)
6          Adicionar os cortes (3.39) no problema mestre
7          Resolver o problema mestre
8          (z^h, y^h) ← Solução do problema mestre
9          LB ← valor da solução do problema mestre
10         Resolver subproblemas com (z^h, y^h)
11         Adicionar o corte (3.39) no problema mestre
12         (z^0, y^0) ← 0.5(z^0, y^0) + 0.5(z^h, y^h)
13         Atualizar UB, se necessário
14         h ← h + 1
15     end
16 end
```

Algoritmo 2: Fase inteira do algoritmo de decomposição de Benders para o THLP

3.4 Algoritmo Eficiente para resolução do subproblema

Os subproblemas da decomposição de Benders proposto em [56] podem ser modelados como problemas de programação linear com $O(|N|^2)$ variáveis reais e $O(|N|^2)$ restrições. Eles são resolvidos por um algoritmo de programação linear no algoritmo proposto em [56]. Nesta seção é proposto um algoritmo ad-hoc para reso-

lução desses subproblemas, quando a solução do problema mestre é uma solução inteira viável. A ideia geral do algoritmo consiste em resolver, por inspeção, o subproblema primal para encontrar seu valor ótimo e então resolver o dual do subproblema através de um sistema de restrições de diferença [19]. Este algoritmo é detalhado a seguir.

Um sistema de restrições de diferença é formado por restrições, onde cada restrição é composta de exatamente duas variáveis: uma com o sinal positivo e a outra com sinal negativo. Em [19] é mostrado que um sistema de restrições de diferença pode ser resolvido por um algoritmo de caminho mais curto de fonte única, como por exemplo Dijkstra [19] e Bellman-Ford [19]. Para resolver um sistema de restrições de diferença por um algoritmo que resolva o problema de caminho mais curto é necessário criar o grafo de restrição [19] correspondente ao sistema de restrições de diferença. Nesse grafo, cada variável do sistema de restrições de diferença é um nó e cada restrição é um arco que tem origem no nó que representa a variável com sinal negativo e destino no nó que representa a variável com sinal positivo. O lado direito da restrição é o custo do arco. Então um nó, denominado o, é adicionado e um arco para cada um dos outros nós é também adicionado, com origem em o. Os custos desses arcos são fixados em 0. Assim, existe um caminho do nó de origem o para todos os outros nós. Para resolver o sistema de restrições de diferença basta calcular o caminho de menor custo com origem no nó o para os outros nós. Cada uma das variáveis do sistema de restrições de diferença recebe o valor do caminho mais curto entre o e o nó que representa essa variável.

Os subproblemas primais, um para cada par de nós $i, j \in N$, compostos pela função objetivo (3.25) e as restrições (3.26)-(3.30), podem ser resolvidos por

inspeção sempre que uma solução $\overline{y}$ e $\overline{z}$ do problema mestre é uma solução viável inteira. Uma solução viável inteira do problema mestre forma uma árvore conectando todos os nós do conjunto N. Desta forma, existe um único caminho entre cada par de nós $i, j \in N$. Quando o caminho entre um nó de origem $i \in N$ para um nó de destino $j \in N$ é conhecido, as variáveis de fluxo x_{ij}^{km} do subproblema primal são iguais a W_{ij}, se o arco $(k, m) \in A$ estiver presente no caminho de i para j, e 0 caso contrário. Para encontrar o caminho entre cada par de nós $i, j \in N$ basta utilizar o algoritmo de busca em largura no grafo onde o conjunto de nós é o mesmo conjunto N e o conjunto de arestas é igual a solução inteira retornada pela resolução do problema mestre.

Os subproblemas duais, um para cada par de nós $i, j \in N$, compostos pela função objetivo (3.31) e as restrições (3.32)-(3.37), podem ser resolvidos utilizando o algoritmo de Bellman-Ford para resolver o sistema de restrições de diferença formado pelas variáveis $u_{ijm}, \forall i, j, m \in N$ e depois calcular o valor das variáveis e_{ijkm}, s_{ijk} e t_{ijm}, obtendo uma solução completa para o subproblema dual.

Considerando apenas as variáveis $u_{ijm}, \forall i, j, m \in N$, com $i < j$, as restrições (3.32) e (3.33) do subproblema dual formam um sistema de restrições de diferença cujo grafo de restrição correspondente é um grafo completo e direcionado com $|N|$ nós. Nesse grafo, cada nó representa uma variável u_{ijm} e cada arco representa uma restrição do subproblema. Os arcos adicionados nesse grafo tem origem no nó que representa a variável u_{ijk} com sinal negativo e destino no nó que representa a variável u_{ijm} com sinal positivo. O custo do arco é o lado direito da restrição. Para adicionar a função objetivo do problema dual no grafo de restrição é preciso alterar os valores dos custos dos arcos entre os nós que representam as duas variáveis que aparecem na função objetivo do subproblema dual. Isso é necessário pois o valor

da função objetivo do subproblema dual já é conhecido e portanto os valores das variáveis que aparecem na função objetivo também. Quando $\overline{z}_{jm}$ e $\overline{z}_{ik}$, para um determinado $m, k \in N$ forem iguais a 1, a função objetivo do dual será $u_{ijm} - u_{ijk}$. Essa diferença é igual a solução ótima do subproblema primal, denominada aqui por obj, ficando $u_{ijm} - u_{ijk} = obj$. Assim a função objetivo do dual pode ser reescrita em duas restrições: $u_{ijm} - u_{ijk} \leq obj$ e $u_{ijm} - u_{ijk} \geq obj$. Multiplicando essa última restrição por -1 teremos $u_{ijk} - u_{ijm} \leq -obj$. Portanto, o custo do arco com origem no nó que representa a variável u_{ijk} e com destino no nó que representa a variável u_{ijm} é alterado para obj. Já o custo do arco com origem no nó que representa a variável u_{ijm} e destino no nó que representa a variável u_{ijk} é alterado para $-obj$. Com essa alteração no grafo de restrição, o caminho de menor custo entre u_{ijk} e u_{ijm} é igual à função objetivo do problema primal. Feito essa alteração, basta calcular o caminho de menor custo de u_{ijk} para os outros nós e assim obter os valores para todas as variáveis u_{ijr} $\forall r \in N$.

Após obter os valores das variáveis duais u_{ijm}, as variáveis duais s_{ijm} e t_{ijk} podem ser fixadas em 0, pois suas restrições correspondentes (3.28) e (3.29) do subproblema primal são redundantes sempre que a solução z e y do problema mestre é inteira. Quando a solução do problema mestre é inteira o limite superior das variáveis x_{ij}^{km} é dado pela restrição (3.27) e então as restrições (3.28) e (3.29) são redundantes, pois elas definem esse mesmo limite superior. Já as variáveis e_{ijkm} podem ser obtidas da seguinte forma: as variáveis e_{ijkm} que aparecem na função objetivo do dual do subproblema são fixadas em 0, pois elas devem ser positivas e aparecem com sinal negativo em um problema de maximização. As demais variáveis e_{ijkm} são iguais ao máximo entre 0 e a diferença entre: (i) diferença absoluta entre as variáveis u_{ijm} e u_{ijk} e (ii) o lado direito das restrições do dual.

O algoritmo proposto por esse trabalho, apresentado no algoritmo 3, consiste em resolver o subproblema primal com o algoritmo de busca em largura (linha 2), construir o grafo de restrição (linha 3) e resolvê-lo com o algoritmo de Bellman-Ford para encontrar os valores das variáveis duais u_{ijm} (linha 4). Em seguinda, na linha 5, são calculados os valores das variáveis duais e_{ijkm}. A ordem de complexidade para resolver o subproblema primal por um algoritmo de busca em largura é $O(|N|)$, enquanto para resolver o sistema de restrição de diferença pelo algoritmo de Bellman-Ford é $O(|N|^3)$, portanto a ordem de complexidade do algoritmo proposto para resolver o subproblema dual é $O(|N|^3)$.

```
1 begin
2 |   Resolver o subproblema primal com o algoritmo de busca em largura
3 |   Criar o grafo de restrição a partir do subproblema dual e do valor
  |   ótimo do problema primal
4 |   Calcular o caminho mais curto partindo de um nó para todos os
  |   outros com o algoritmo Bellman-Ford para encontrar o valor das
  |   variáveis u_ijm
5 |   Calcular o valor das variáveis e_ijkm
6 end
```

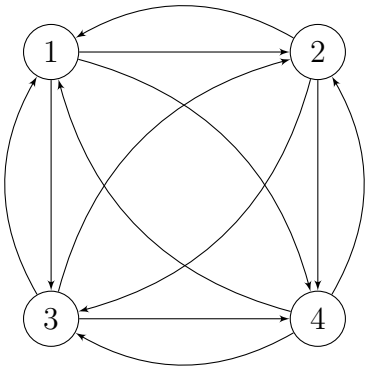
Algoritmo 3: Algoritmo ad-hoc para resolução dos subproblemas da decomposição de Benders

A Figura 3.1 apresenta um exemplo de como o algoritmo proposto para resolver os subproblemas da decomposição de Benders para o THLP constrói o grafo de restrição de diferença. Na Figura 3.1a é apresentado um grafo completo e direcionado com 4 nós. Os custos c_{ij} e os fluxos W_{ij} não são apresentados nessa figura apenas para facilitar a visualização. Nesse exemplo, o número de nós concentradores para serem alocados é igual a 3, o nó $i = 2$ e o nó $j = 4$. A Figura 3.1b mostra os nós concentradores 1, 2 e 3 conectados por meio de uma árvore e o nó cliente 4 alocado ao nó concentrador 3. A Figura 3.1c destaca o caminho

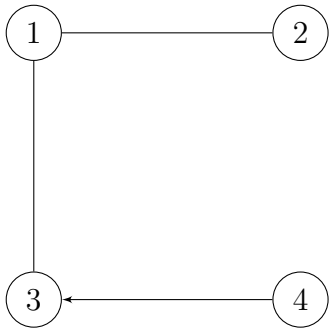
utilizado para enviar o fluxo do nó concentrador 2 para o nó cliente 4, enquanto, a Figura 3.1d destaca o caminho utilizado para enviar o fluxo do nó cliente 4 para o nó concentrador 2. A Figura 3.1e apresenta o problema dual considerando apenas as variáveis u. Já a Figura 3.1f apresenta o grafo de restrições associado a esse sistema de restrição de diferença.

3.5 Experimentos Computacionais

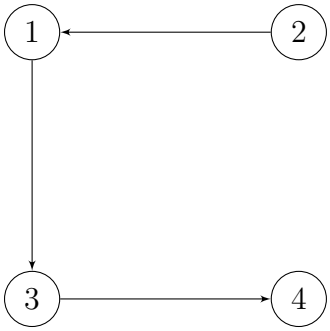
Os experimentos computacionais foram executados em uma máquina Intel Xeon E5405 com 2.00 GHz de clock e 16 GB de memória RAM, com o sistema operacional Linux. Dois algoritmos de *branch and bound* para o THLP foram implementados usando o resolvedor ILOG CPLEX versão 12.6, com parâmetros nos valores padrões. O primeiro foi baseado na modelagem apresentada em [18] e é composto pela função objetivo (3.1) e as restrições (3.2)-(3.10), chamado aqui de CPLEX (V1), e o segundo foi baseado na formulação proposta em [17] juntamente com as desigualdades válidades propostas em [56] e é composta pela função objetivo (3.11) e as restrições (3.12)-(3.24) e é identificado aqui por CPLEX (V2). O algoritmo de Benders proposto em [56], chamado aqui de Benders (V1), e o algoritmo proposto na seção 3.4, identificado aqui por Benders+, foram codificados utilizando a linguagem de programação C++. Estes algoritmos utilizam o CPLEX na versão 12.6 para resolver os problemas de programação linear inteira mista. Nessas decomposições de Benders, o problema mestre é composto pela função objetivo (3.38) e as restrições (3.12)-(3.17), (3.23), (3.24) e (3.39)-(3.50). Já o subproblema, para cada par de nós $i, j \in N$, com $i < j$, é composto função objetivo (3.31) e as restrições (3.32)-(3.37).



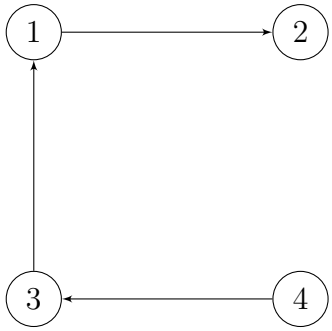
(a) Grafo completo e direcionado



(b) Nós concentradores 1, 2 e 3, sua árvore de conexão e a alocação do nó cliente 4 ao nó concentrador 3



(c) Caminho utilizado para transportar o fluxo com origem no nó 2 e destino no nó 4

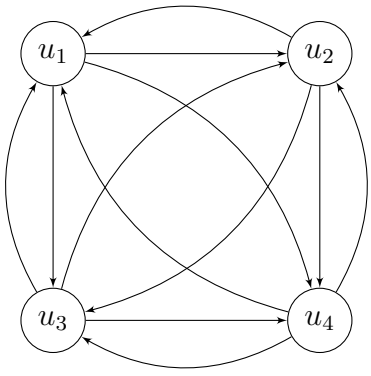


(d) Caminho utilizado para transportar o fluxo com origem no nó 4 e destino no nó 2

$$\max \overline{Q}(2,4) = u_3 - u_2 = ((c_{21} + c_{13}) \times W_{24} + (c_{31} + c_{12}) \times W_{42}) \times \alpha$$

$$u_2 - u_1 \leq \alpha \times (c_{12} \times W_{42} + c_{21} \times W_{24})$$
$$u_3 - u_1 \leq \alpha \times (c_{13} \times W_{42} + c_{31} \times W_{24})$$
$$u_4 - u_1 \leq \alpha \times (c_{14} \times W_{42} + c_{41} \times W_{24})$$
$$u_3 - u_2 \leq \alpha \times (c_{23} \times W_{42} + c_{32} \times W_{24})$$
$$u_4 - u_2 \leq \alpha \times (c_{24} \times W_{42} + c_{42} \times W_{24})$$
$$u_4 - u_3 \leq \alpha \times (c_{34} \times W_{42} + c_{43} \times W_{24})$$
$$u_1 - u_2 \leq \alpha \times (c_{21} \times W_{42} + c_{12} \times W_{24})$$
$$u_1 - u_3 \leq \alpha \times (c_{31} \times W_{42} + c_{13} \times W_{24})$$
$$u_1 - u_4 \leq \alpha \times (c_{41} \times W_{42} + c_{14} \times W_{24})$$
$$u_2 - u_3 \leq \alpha \times (c_{32} \times W_{42} + c_{23} \times W_{24})$$
$$u_2 - u_4 \leq \alpha \times (c_{42} \times W_{42} + c_{24} \times W_{24})$$
$$u_3 - u_4 \leq \alpha \times (c_{43} \times W_{42} + c_{34} \times W_{24})$$

(e) Problema dual com apenas as variáveis u, onde as restrições formam um sistema de restrições de diferença



(f) Grafo de restrições associado ao sistema de restrições de diferença das variáveis u

Figura 3.1: Exemplo do sistema de restrição diferença para o problema dual, em um grafo completo e direcionado com 4 nós e número de nós concentradores, $p = 3$, para serem selecionados

Dois conjuntos de instâncias de testes são utilizadas nos experimentos computacionais e são descritas a seguir. As instâncias AP (do inglês *Australian Post*) são baseadas no fluxo de correspondência do serviço de correio de algumas cidades da Austrália e foram propostas em [23]. As instâncias CAB (do inglês *Civil Aeronautics Board*) são baseadas no fluxo de passageiros aéreos entre 25 cidades americanas na década de 70 e foram propostas em [49]. Essas instâncias são utilizadas na literatura para problemas similares ao problema de localização de concentradores e são definidas em um grafo completo e direcionado, onde existe um custo $c_{ij} \in \mathbb{R}_+$ para cada par de nós $i, j \in N$ e uma demanda $W_{ij} \in \mathbb{R}_+^*$ do nó de origem i para o nó de destino j.

Foram executados três experimentos para comparar o estado da arte em formulações e algoritmos exatos para o THLP. Os dois primeiros experimentos comparam os modelos CPLEX (V1) e CPLEX (V2), enquanto o terceiro experimento compara os algoritmos Benders (V1) e Benders+. O primeiro experimento tem o objetivo de quantificar o quanto a relaxação do modelo CPLEX (V2) é melhor do que a relaxação do modelo CPLEX (V1). O segundo experimento foi realizado com o objetivo de comparar os tempos computacionais para resolver os modelos CPLEX (V1) e CPLEX (V2). O terceiro experimento tem o objetivo de verificar o ganho de tempo computacional que o algoritmo proposto na seção 3.4 proporcionou.

No primeiro experimento, foi avaliado a relaxação linear dos modelos CPLEX (V1) e CPLEX (V2) pela comparação dos tempos computacionais e do valor ótimo obtido pela resolução da relaxação linear desses modelos em instâncias AP e CAB. Os resultados desse experimento são mostrados nas Tabelas 3.1 e 3.2. A Tabela 3.1 apresenta os resultados obtidos pela resolução da relaxação

linear dos modelos CPLEX (V1) e CPLEX (V2) nas instâncias AP, enquanto os resultados obtidos pela resolução da relaxação linear dos modelos CPLEX (V1) e CPLEX (V2) nas instâncias CAB estão presentes na Tabela 3.2. Nessas tabelas, as colunas 1, 2 e 3 mostram, respectivamente, a quantidade de nós $|N|$ da instância, a quantidade p de nós que devem ser selecionados como concentradores, e o valor α de desconto que deve ser aplicado nos arcos que conectam dois nós concentradores. As colunas 4 e 5 contém o valor ótimo da relaxação linear dos modelos CPLEX (V1) e CPLEX (V2), respectivamente. A coluna 6 mostra o desvio relativo percentual entre o valor ótimo obtido pela relaxação linear dos modelos CPLEX (V1) e CPLEX (V2), calculado pela razão entre: (i) a diferença entre o valor ótimo obtido pela relaxação linear do modelo CPLEX (V2) e o valor ótimo obtido pela relaxação linear do modelo CPLEX (V1) e (ii) o valor ótimo obtido pela relaxação linear do modelo CPLEX (V1). As colunas 7 e 8 apresentam o tempo de execução, em segundos, para resolver a relaxação linear dos modelos CPLEX (V1) e CPLEX (V2). A coluna 9 apresenta quantas vezes a resolução da relaxação linear do modelo CPLEX (V1) foi mais rápida do que a relaxação do modelo CPLEX (V2). Este valor é obtido pela divisão das colunas 8 e 7.

Pode-se observar que o tempo máximo para resolver a relaxação linear do modelo CPLEX (V1) nas instâncias AP foi 18,21 segundos, como pode ser observado na coluna 7 da Tabela 3.1. Já para resolver a relaxação linear do modelo CPLEX (V2), nessas mesmas instâncias, foi necessário até 1571,84 segundos, conforme a coluna 8 dessa mesma tabela. A resolução da relaxação linear do modelo CPLEX (V1) foi no mínimo 2 e no máximo 276 vezes mais rápido do que a resolução da relaxação linear do modelo CPLEX (V2) nas instâncias AP, conforme a coluna 9 da Tabela 3.1. Por outro lado, o desvio relativo percentual mínimo foi de

1,04 % enquanto o máximo foi de 20,77 %, como pode ser observado na coluna 6 da Tabela 3.1.

Considerando as instâncias CAB, pode-se observar que o tempo máximo para resolver a relaxação linear do modelo CPLEX (V1) foi 10,28 segundos, como pode ser observado na coluna 7 da Tabela 3.2. Enquanto para resolver a relaxação linear do modelo CPLEX (V2), nessas mesmas instâncias, foi necessário até 3396,81 segundos, conforme a coluna 8 dessa mesma tabela. A resolução da relaxação linear do modelo CPLEX (V1) foi no mínimo 2 e no máximo 383 vezes mais rápido do que a resolução da relaxação linear do modelo CPLEX (V1) nas instâncias CAB, conforme a coluna 9 da Tabela 3.2. Porém, o desvio relativo percentual mínimo foi de 0,32 % enquanto o máximo foi de 19,89 %, como pode ser observado na coluna 6 da Tabela 3.2.

No segundo experimento, foi avaliado o desempenho dos algoritmos de *branch and bound* baseados nos modelos CPLEX (V1) e CPLEX (V2) pela comparação dos tempos computacionais obtidos na execução desses algoritmos em instâncias AP e CAB. O tempo máximo de execução do algoritmo de *branch and bound* para esses modelos foi setado para 144000 segundos (40 horas). Os resultados desse experimento são mostrados nas Tabelas 3.3 e 3.4. A Tabela 3.3 apresenta os resultados obtidos pelo algoritmo de *branch and bound* utilizando os modelos CPLEX (V1) e CPLEX (V2) nas instâncias AP, enquanto os resultados obtidos pelo algoritmo de *branch and bound* com os modelos CPLEX (V1) e CPLEX (V2) nas instâncias CAB estão presentes na Tabela 3.4. Nessas tabelas, as colunas 1, 2 e 3 apresentam as características da instância. A coluna 1 é a quantidade de nós $|N|$ da instância, a coluna 2 o número p de nós concentradores que devem ser alocados e a coluna 3 é o fator de desconto α que deve ser aplicado nas arestas

Tabela 3.1: Comparação do tempo gasto, em segundos, e do valor ótimo encontrado pela resolução da relaxação linear dos modelos CPLEX (V1) e CPLEX (V2) nas instâncias AP.

			Valor da Função Objetivo			Tempo(s)		
$\|N\|$	p	α	CPLEX (V1)	CPLEX (V2)	Desv.(%)	CPLEX (V1)	CPLEX (V2)	$\frac{CPLEX\ (V2)}{CPLEX\ (V1)}$
10	3	0,2	51188,93	52541,03	2,64	0,25	0,54	2,16
10	3	0,5	58912,28	63166,88	7,22	0,25	0,59	2,36
10	3	0,8	66436,79	72640,83	9,34	0,25	0,59	2,36
10	5	0,2	32315,14	34329,67	6,23	0,26	0,56	2,15
10	5	0,5	43970,28	49418,78	12,39	0,25	0,59	2,36
10	5	0,8	55023,77	64013,26	16,34	0,26	0,66	2,54
10	8	0,2	17752,00	20291,17	14,30	0,25	0,58	2,32
10	8	0,5	32143,76	38143,59	18,67	0,25	0,56	2,24
10	8	0,8	46284,29	55896,98	20,77	0,26	0,56	2,15
15	3	0,2	58357,01	59294,82	1,61	0,48	7,44	15,50
15	3	0,5	65839,25	68424,92	3,93	0,64	7,69	12,02
15	3	0,8	72910,15	77174,96	5,85	0,63	9,62	15,27
15	5	0,2	42857,30	44541,10	3,93	0,46	4,36	9,48
15	5	0,5	54353,85	58688,63	7,98	0,67	9,29	13,87
15	5	0,8	64206,16	71170,34	10,85	0,79	16,84	21,32
15	8	0,2	27949,63	30436,92	8,90	0,41	9,17	22,37
15	8	0,5	41838,68	48041,33	14,83	0,51	10,35	20,29
15	8	0,8	55264,69	64708,28	17,09	0,60	11,93	19,88
20	3	0,2	57641,62	58761,19	1,94	1,19	90,63	76,16
20	3	0,5	67032,99	69515,96	3,70	2,45	127,51	52,05
20	3	0,8	74017,64	78177,63	5,62	2,68	127,55	47,59
20	5	0,2	44807,49	46480,37	3,73	1,48	101,82	68,80
20	5	0,5	56877,39	61061,41	7,36	2,60	144,88	55,72
20	5	0,8	66915,93	73379,51	9,66	2,65	166,46	62,82
20	8	0,2	33015,32	35275,90	6,85	1,20	179,88	149,90
20	8	0,5	46929,98	52228,04	11,29	2,09	161,30	77,18
20	8	0,8	59699,68	67980,61	13,87	2,91	214,57	73,74
25	3	0,2	59981,62	60602,29	1,04	4,32	496,82	115,01
25	3	0,5	68224,63	70130,92	2,79	8,28	816,40	98,60
25	3	0,8	75322,95	79442,48	5,47	18,21	1498,30	82,28
25	5	0,2	45881,23	47432,70	3,38	3,50	801,12	228,89
25	5	0,5	57268,05	61046,70	6,60	7,62	1289,01	169,16
25	5	0,8	67382,13	73569,91	9,18	8,14	1488,59	182,87
25	8	0,2	35204,04	37295,69	5,94	3,80	1050,53	276,46
25	8	0,5	48450,89	53651,77	10,73	7,63	1330,25	174,35
25	8	0,8	60842,41	68378,90	12,39	10,16	1571,84	154,71

Tabela 3.2: Comparação do tempo gasto, em segundos, e do valor ótimo encontrado pela resolução da relaxação linear dos modelos CPLEX (V1) e CPLEX (V2) nas instâncias CAB.

			Valor da Função Objetivo			Tempo(s)		
$\|N\|$	p	α	CPLEX (V1)	CPLEX (V2)	Desv.(%)	CPLEX (V1)	CPLEX (V2)	$\frac{CPLEX\ (V2)}{CPLEX\ (V1)}$
10	3	0,2	491,17	494,52	0,68	0,07	0,55	7,86
10	3	0,5	597,58	612,98	2,58	0,27	0,55	2,04
10	3	0,8	682,56	718,97	5,33	0,27	0,54	2,00
10	5	0,2	310,19	322,92	4,11	0,26	0,54	2,08
10	5	0,5	458,65	499,38	8,88	0,27	0,55	2,04
10	5	0,8	598,01	667,39	11,60	0,27	0,63	2,33
10	8	0,2	164,43	190,52	15,86	0,26	0,54	2,08
10	8	0,5	343,50	411,83	19,89	0,26	0,58	2,23
10	8	0,8	519,75	622,78	19,82	0,26	0,60	2,31
15	3	0,2	1875,23	1915,21	2,13	0,50	5,53	11,06
15	3	0,5	2234,52	2324,40	4,02	0,56	11,96	21,36
15	3	0,8	2548,18	2666,09	4,63	0,87	10,23	11,76
15	5	0,2	1243,64	1299,64	4,50	0,43	6,07	14,12
15	5	0,5	1775,92	1935,08	8,96	0,73	12,43	17,03
15	5	0,8	2231,00	2454,20	10,00	0,98	16,17	16,50
15	8	0,2	0798,02	872,27	9,30	0,45	6,85	15,22
15	8	0,5	1404,59	1590,30	13,22	0,59	10,89	18,46
15	8	0,8	1982,02	2250,29	13,54	0,75	14,71	19,61
20	3	0,2	4156,66	4170,15	0,32	1,11	86,87	78,26
20	3	0,5	5149,49	5234,94	1,66	2,60	115,55	44,44
20	3	0,8	5991,49	6279,35	4,80	4,04	171,04	42,34
20	5	0,2	2683,92	2808,68	4,65	0,87	113,10	130,00
20	5	0,5	4009,72	4384,31	9,34	2,29	184,42	80,53
20	5	0,8	5197,14	5663,54	8,97	3,09	172,93	55,96
20	8	0,2	1893,18	2041,65	7,84	0,75	142,82	190,43
20	8	0,5	3289,99	3662,06	11,31	1,95	196,41	100,72
20	8	0,8	4632,22	5247,01	13,27	3,16	297,97	94,29
25	3	0,2	6498,95	6554,65	0,86	3,92	810,55	206,77
25	3	0,5	8088,72	8274,01	2,29	8,47	968,12	114,30
25	3	0,8	9568,20	9923,90	3,72	10,28	1396,87	135,88
25	5	0,2	4586,97	4791,05	4,45	3,38	1295,03	383,14
25	5	0,5	6621,18	7190,74	8,60	8,96	1672,27	186,64
25	5	0,8	8464,57	9173,35	8,37	8,20	1404,84	171,32
25	8	0,2	3511,16	3752,85	6,88	4,02	1261,02	313,69
25	8	0,5	5615,60	6264,08	11,55	8,49	1845,52	217,38
25	8	0,8	7608,65	8613,56	13,21	9,13	3396,81	372,05

que conectam nós concentradores. A coluna 4 apresenta o valor objetivo obtido. As colunas 5 e 6 são o tempo gasto, em segundos, até encontrar o valor ótimo ou o tempo máximo ser atingido, nesse caso, essa coluna apresenta o valor 144000. A coluna 7 é a divisão do tempo gasto pelo algoritmo de *branch and bound* com o modelo CPLEX (V2) pelo tempo gasto pelo algoritmo de *branch and bound* com o modelo CPLEX (V1). Nas colunas 8 e 9 estão a quantidade de nós na árvore de *branch and bound* que o CPLEX avaliou para resolver a instância, respectivamente, com os modelos CPLEX (V1) e CPLEX (V2).

Pode-se observar que em até 40 horas, o algoritmo de *branch and bound* com o modelo CPLEX (V1) não foi capaz de resolver a instância AP com 25 nós, 8 nós concentradores e $\alpha = 0,8$ enquanto o algoritmo de *branch and bound* com o modelo CPLEX (V2) foi capaz de resolver todas as instâncias AP, conforme pode ser observado pelas colunas 5 e 6 da Tabela 3.3. O número de nós avaliados pelo algoritmo *branch and bound* baseado no modelo CPLEX (V2) é sempre menor ou igual ao números de nós avaliados pelo algoritmo baseado no modelo CPLEX (V1). Conforme pode ser observado nas colunas 8 e 9 da Tabela 3.3. Com o modelo CPLEX (V1), o algoritmo de *branch and bound* resolveu metade das instâncias no primeiro nó, conforme a coluna 8 (3.3). Já com o modelo CPLEX (V2) o algoritmo *branch and bound* resolveu mais de 80% das instâncias no primeiro nó, conforme pode ser observado na coluna 9 dessa tabela.

Também pode-se ver que em até 40 horas o algoritmo de *branch and bound* baseado no modelo CPLEX (V1) não foi capaz de resolver a instância CAB com 25 nós, 8 nós concentradores e $\alpha = 0,8$ enquanto o algoritmo de *branch and bound* baseado no modelo CPLEX (V2) foi capaz de resolver todas as instâncias CAB, conforme pode ser observado nas colunas 5 e 6 da Tabela 3.4. O número de nós

Tabela 3.3: Comparação do tempo gasto, em segundos, e da quantidade de nós na árvore de *branch and bound* pela resolução dos modelos CPLEX (V1) e CPLEX (V2) nas instâncias AP.

				Tempo(s)			#Nós	
$\|N\|$	p	α	Obj	CPLEX (V1)	CPLEX (V2)	$\frac{CPLEX\ (V2)}{CPLEX\ (V1)}$	CPLEX (V1)	CPLEX (V2)
10	3	0,2	52541,03	0,18	0,27	1,50	1	1
10	3	0,5	63166,88	0,27	0,29	1,07	1	1
10	3	0,8	72640,83	0,53	0,29	0,55	1	1
10	5	0,2	34340,01	0,45	0,38	0,84	1	1
10	5	0,5	49418,78	2,06	0,29	0,14	1	1
10	5	0,8	64013,26	3,24	0,32	0,10	70	1
10	8	0,2	20513,41	4,89	0,80	0,16	172	1
10	8	0,5	39288,19	5,39	1,39	0,26	339	39
10	8	0,8	57953,45	5,21	1,50	0,29	605	56
15	3	0,2	59294,82	0,91	1,74	1,91	1	1
15	3	0,5	68424,92	2,26	2,69	1,19	1	1
15	3	0,8	77174,96	3,28	3,46	1,05	1	1
15	5	0,2	44541,10	4,11	2,25	0,55	1	1
15	5	0,5	58688,63	33,59	3,68	0,11	33	1
15	5	0,8	71170,34	34,45	4,55	0,13	211	1
15	8	0,2	30738,47	30,96	9,50	0,31	254	1
15	8	0,5	48151,49	108,34	11,41	0,11	2488	1
15	8	0,8	65213,51	455,81	15,47	0,03	5208	111
20	3	0,2	58761,19	1,70	30,99	18,23	1	1
20	3	0,5	69515,96	14,99	35,62	2,38	1	1
20	3	0,8	78177,63	18,85	38,29	2,03	1	1
20	5	0,2	46480,37	17,47	27,08	1,55	1	1
20	5	0,5	61061,41	216,9	43,51	0,20	133	1
20	5	0,8	73592,97	781,05	93,17	0,12	4423	9
20	8	0,2	35296,99	272,19	61,14	0,23	275	1
20	8	0,5	52294,29	975,02	64,28	0,07	1289	1
20	8	0,8	68272,78	6245,99	139,62	0,02	7205	17
25	3	0,2	60602,29	3,98	116,51	29,27	1	1
25	3	0,5	70130,92	22,78	261,99	11,50	1	1
25	3	0,8	79442,48	106,25	677,83	6,38	1	1
25	5	0,2	47432,70	32,37	308,60	9,53	1	1
25	5	0,5	61046,70	693,46	542,99	0,78	1	1
25	5	0,8	73569,91	1936,06	657,70	0,34	1881	1
25	8	0,2	37295,69	1078,05	393,20	0,37	92	1
25	8	0,5	54043,74	40187,62	1043,99	0,03	11939	46
25	8	0,8	69429,77	144000,00	13230,78	0,09	31252	819

avaliados pelo algoritmo *branch and bound* com o modelo CPLEX (V2) é sempre menor ou igual ao número de nós avaliados pelo algoritmo baseado no modelo CPLEX (V1). Conforme pode ser observado nas colunas 8 e 9 da Tabela 3.4. Com o modelo CPLEX (V1), o algoritmo de *branch and bound* resolveu mais da metade das instâncias no primeiro nó, conforme a coluna 8 dessa tabela. Já com o modelo CPLEX (V2) o algoritmo *branch and bound* não resolveu no primeiro nó, apenas 4 instâncias, conforme pode ser observado na coluna 9 dessa tabela.

No terceiro experimento, avaliou-se o impacto do algoritmo proposto para resolver os subproblemas da decomposição de Benders quando as variáveis z e y formam uma solução viável inteira para o problema mestre. Para isso os algoritmos Benders (V1) e Benders+ foram executados em instâncias AP de 50 nós, sendo o tempo limite de execução setado para 86400 segundos (24 horas). Os resultados desse experimento são mostrados na Tabela 3.5. Nessa tabela são apresentadas apenas informações da segunda fase, denominada inteira, de cada algoritmo, uma vez que a primeira fase, denominada aquecimento, é idêntica para os dois algoritmos. A quantidade p de concentradores e o fator de desconto α são dados, respectivamente, nas colunas 1 e 2. As colunas 3, 4, 5, 6 e 7 são referentes ao algoritmo Benders (V1). A coluna 3 é o gap da solução retornada pelo algoritmo, calculado por (UB - LB) / UB, onde UB e LB são respectivamente o limite superior e o limite inferior da solução. A coluna 4 armazena o número de iterações que o algoritmo executou, as colunas 5 e 6 apresentam os tempos totais gastos, em segundos, para resolver respectivamente o problema mestre e o subproblema. A coluna 7 mostra a razão entre o tempo total gasto para resolver o subproblema e o número de iterações. As colunas 8, 9, 10, 11 e 12 apresentam, respectivamente, essas mesmas informações para o algoritmo Benders+. A coluna 13 indica a razão entre:

Tabela 3.4: Comparação do tempo gasto, em segundos, e da quantidade de nós na árvore de *branch and bound* pela resolução dos modelos CPLEX (V1) e CPLEX (V2) nas instâncias CAB.

				Tempo(s)			#Nós	
$\|N\|$	p	α	Obj	CPLEX (V1)	CPLEX (V2)	$\frac{CPLEX\ (V2)}{CPLEX\ (V1)}$	CPLEX (V1)	CPLEX (V2)
10	3	0,2	494,52	0,15	0,28	1,87	1	1
10	3	0,5	612,98	0,32	0,28	0,88	1	1
10	3	0,8	718,97	0,28	0,27	0,96	1	1
10	5	0,2	322,92	0,27	0,27	1,00	1	1
10	5	0,5	499,38	0,61	0,28	0,46	1	1
10	5	0,8	667,39	1,57	0,31	0,20	1	1
10	8	0,2	190,52	0,61	0,28	0,46	1	1
10	8	0,5	411,83	2,30	0,29	0,13	41	1
10	8	0,8	631,57	3,89	1,30	0,33	341	1
15	3	0,2	1915,21	0,99	2,45	2,48	1	1
15	3	0,5	2324,40	1,48	3,09	2,09	1	1
15	3	0,8	2666,07	3,55	4,12	1,16	1	1
15	5	0,2	1299,64	2,77	1,97	0,71	1	1
15	5	0,5	1935,08	25,77	4,48	0,17	1	1
15	5	0,8	2454,20	34,63	4,75	0,14	54	1
15	8	0,2	876,36	23,61	4,51	0,19	38	1
15	8	0,5	1590,30	49,26	3,74	0,07	180	1
15	8	0,8	2250,29	70,57	3,83	0,05	782	1
20	3	0,2	4170,15	1,00	21,82	21,82	1	1
20	3	0,5	5234,94	3,60	35,03	9,73	1	1
20	3	0,8	6279,35	25,89	58,66	2,27	1	1
20	5	0,2	2808,68	6,05	22,82	3,77	1	1
20	5	0,5	4384,03	156,99	66,61	0,42	1	1
20	5	0,8	5663,54	317,81	66,71	0,21	199	1
20	8	0,2	2057,03	132,31	64,21	0,49	109	15
20	8	0,5	3700,18	712,30	137,51	0,19	1298	25
20	8	0,8	5268,77	9355,64	219,11	0,02	16859	17
25	3	0,2	6554,65	2,96	276,08	93,27	1	1
25	3	0,5	8274,01	17,97	424,32	23,61	1	1
25	3	0,8	9923,90	51,41	483,21	9,40	1	1
25	5	0,2	4791,05	50,46	352,43	6,98	1	1
25	5	0,5	7190,67	2040,30	701,25	0,34	242	1
25	5	0,8	9172,79	5301,23	561,82	0,11	494	1
25	8	0,2	3752,85	1464,58	457,35	0,31	122	1
25	8	0,5	6263,47	64567,72	673,93	0,01	17232	1
25	8	0,8	8387,60	144000,00	10231,38	0,07	24412	216

(i) a diferença entre o tempo de resolução dos subproblemas em Benders (V1) e o tempo de resolução dos subproblemas em Benders+ e (ii) o tempo de resolução dos subproblemas em Benders (V1).

Os resultados computacionais indicam que o algoritmo Benders+ permitiu reduzir em até 29,52% os tempos computacionais para resolver os subproblemas, como mostrado na última coluna da Tabela 3.5.

Tabela 3.5: Comparação entre Benders (V1) e Benders+ em instâncias AP de 50 nós.

		Benders (V1)					Benders+					
p	α	Gap(%)	#Iter.	MP(s)	SP(s)	SP/#Iter.	Gap(%)	#Iter.	MP(s)	SP(s)	SP/#Iter.	% *speedup*
3	0,2	0,00	1	17,18	579,38	579,38	0,00	1	17,17	411,97	411,97	28,89
3	0,5	0,00	2	1511,00	1173,55	586,78	0,00	2	1625,34	850,64	425,32	27,52
3	0,8	0,00	2	6777,70	1173,45	586,72	0,00	2	7180,10	846,16	423,08	27,89
5	0,2	0,00	2	124,46	1173,33	586,66	0,00	2	118,73	847,93	423,96	27,73
5	0,5	0,00	2	18136,76	1187,93	593,97	0,00	2	18705,20	897,95	448,98	24,41
5	0,8	0,02	2	81490,64	1163,19	581,59	0,03	2	81640,61	835,47	417,73	28,17
8	0,2	0,08	7	78753,87	4116,52	588,07	0,11	7	79752,70	2943,11	420,44	28,51
8	0,5	0,01	2	81560,22	1166,66	583,33	0,04	2	81709,88	832,98	416,49	28,60
8	0,8	5,23	1	82130,57	580,22	580,22	5,23	1	82174,91	408,92	408,92	29,52

Capítulo 4

Conclusões

Este trabalho se dedicou ao estudo de dois problemas de otimização *NP-Difíceis*. O primeiro problema estudado foi o problema do caminho mais curto robusto e este trabalho apresentou a primeira formulação de programação linear inteira mista para o *minmax relative regret* e inequações válidas. Experimentos computacionais foram executados para instâncias clássicas baseadas em grafos Karasan e em novas instâncias baseadas em grafos grid. O *branch and bound* do CPLEX baseado nessa formulação encontrou soluções ótimas para a maioria das pequenas instâncias Karasan e grid com até 200 nós. Em instâncias Karasan com até 1500 nós o gap médio obtido foi de 16,68%, enquanto em instâncias grid com até 1500 nós o gap médio obtido foi de 18,78%. As instâncias grid propostas neste trabalho foram mais difícies de serem resolvidas do que instâncias Karasan encontradas na literatura. O método de linearização e as inequações válidas aplicadas a função objetivo do *minmax relative regret* RSP podem também ser aplicadas a outros problemas de otimização robusta *minmax relative regret*. Os resultados obtidos para esse problema foram publicados no Journal of Global Optimization [15].

O segundo problema estudado foi o problema de localização de concentradores em árvore. O estado da arte em formulações e algoritmos exatos para o THLP foram avaliados de forma independente em um conjunto maior de instâncias. Além disso, foi proposto um algoritmo ad-hoc para resolver os subproblemas da decomposição de Benders apresentada em [56]. Experimentos computacionais foram executados em instâncias clássicas para problemas de localização de concentradores. Em instâncias com até 25 nós, foram comparadas a resolução, através do algoritmo de *branch and bound* do CPLEX baseado nos modelos CPLEX (V1) e CPLEX (V2). O algoritmo de *branch and bound* baseado no modelo CPLEX (V1) não foi capaz de resolver todas as instâncias em até 40 horas. Porém, o algoritmo de *branch and bound* baseado no modelo CPLEX (V2) foi capaz de resolver todas as instâncias em até 40 horas, sendo que não foi capaz de resolver apenas quatro instâncias em até 10 minutos. Para verificar a eficiência do algoritmo proposto para resolver os subproblemas da decomposição de Benders apresentada em [56], experimentos computacionais foram executados em instâncias de 50 nós. Os resultados computacionais mostraram que o algoritmo proposto reduziu em até 29,52% os tempos computacionais para a resolução desses subproblemas. Esses resultados preliminares são promissores e permitirão realizar pesquisas futuras para melhorar ainda mais a qualidade das soluções produzidas. O algoritmo proposto para resolver os subproblemas da decomposição de Benders podem ser utilizados em outros algoritmos de decomposição de Benders onde o parâmetro de entrada dos subproblemas é uma árvore. Os resultados preliminares desse trabalho para o THLP foram aceitos para publicação nos anais do XLVII Simpósio Brasileiro de Pesquisa Operacional [33].

Referências Bibliográficas

[1] Aissi, H., Bazgan, C., Vanderpooten, D.: Min-max and min-max regret versions of combinatorial optimization problems: A survey. European Journal of Operational Research **197**, 427–438 (2009)

[2] Alumur, S., Kara, B.Y.: Network hub location problems: The state of the art. European Journal of Operational Research **190**(1), 1 – 21 (2008)

[3] Averbakh, I.: Computing and minimizing the relative regret in combinatorial optimization with interval data. Discrete Optimization **2**, 273–287 (2005)

[4] Averbakh, I., Lebedev, V.: Interval data minmax regret network optimization problems. Discrete Applied Mathematics **138**, 289–301 (2004)

[5] Bellman, R.: On a routing problem. Quarterly of Applied Mathematics **16**, 87–90 (1958)

[6] Ben-Tal, A., Nemirovski, A.: Robust optimization – methodology and applications. Math. Programming **92**, 453–480 (2002)

[7] Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Computational Management Science **2**(1), 3 – 19 (2005)

[8] Bertsekas, D.P., Tsitsiklis, J.N.: An analysis of stochastic shortest path problems. Mathematics of Operations Research **16**, 580–595 (1991)

[9] Bisschop, J.: AIMMS - Optimization modeling. Integer Linear Programming Tricks. Paragon Decision Technology B.V., Haarlem (2005)

[10] Bondy, J.A., Murty, U.S.R.: Graph theory with applications. Elsevier Science Ltd (1976)

[11] Campbell, J.F.: Integer programming formulations of discrete hub location problems. European Journal of Operational Research (72), 387 – 405 (1994)

[12] Campbell, J.F., O'Kelly, M.E.: Twenty-five years of hub location research. Transportation Science **46**(2), 153–169 (2012)

[13] Candia-Véjar, A., Álvarez-Miranda, E., Maculan, N.: Minmax regret combinatorial optimization problems: an algorithmic perspective. RAIRO-Operation Reserach **45**, 101–129 (2011)

[14] Catanzaro, D., Labbé, M., Salazar-Neumann, M.: Reduction approaches for robust shortest path problems. Computers & Operations Research **38**, 1610–1619 (2011)

[15] Coco, A.A., Júnior, J.C.A., Noronha, T.F., Santos, A.C.: An integer linear programming formulation and heuristics for the minmax relative regret robust shortest path problem. Journal of Global Optimization **60**(2), 265–287 (2014)

[16] Conde, E.: On a constant factor approximation for minmax regret problems using a symmetry point scenario. European Journal of Operational Research **219**, 452–457 (2012)

[17] Contreras, I., Fernandez, E., Marin, A.: Tight bounds from a path based formulation for the tree of hub location problem. Computers & Operations Research **36**(12), 3117 – 3127 (2009)

[18] Contreras, I., Fernandez, E., Marin, A.: The tree of hubs location problem. European Journal of Operational Research **202**(2), 390 – 400 (2010)

[19] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Third Edition, 3rd edn. The MIT Press (2009)

[20] Cánovas, L., García, S., Marín, A.: Solving the uncapacitated multiple allocation hub location problem by means of a dual-ascent technique. European Journal of Operational Research (179), 990 – 1007 (2007)

[21] Desrochers, M., Laporte, G.: Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. Operations Research Letters **10**, 27–36 (1991)

[22] Dijkstra, E.W.: A note on two problems in connection with graphs. Numerische Mathematik **1**, 269–271 (1959)

[23] Ernst, A.T., Krishnamoorthy, M.: Efficient algorithms for the uncapacitated single allocation p-hub median problem. Location Science **4**(3), 139 – 154 (1996)

[24] Ernst, A.T., Krishnamoorthy, M.: An exact solution approach based on shortest-paths for p-hub median problems. INFORMS Journal on Computing (10), 149 – 162 (1998)

[25] Ernst, A.T., Krishnamoorthy, M.: Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem. European Journal of Operational Research (104), 100 – 112 (1998b)

[26] Escoffier, B., Monnot, J., Spanjaard, O.: Some tractable instances of interval data minmax regret problems: bounded distance from triviality. In: Proceedings of the 34th conference on current trends in theory and practice of computer science, pp. 280–291. Smokovec, Slovakia (2008)

[27] Farahani, R.Z., Hekmatfar, M., Arabani, A.B., Nikbakhsh, E.: Survey: Hub location problems: A review of models, classification, solution techniques, and applications. Comput. Ind. Eng. **64**(4), 1096 – 1109 (2013)

[28] Gabrel, V., Murat, C., Wu, L.: New models for the robust shortest path problem: complexity, resolution and generalization. Annals of Operations Research pp. 1–24 (2011)

[29] Gallo, G., Pallottino, S.: Shorthest Path Methods: A Unifying Approach. Mathematical Progamming Study **26**, 38–64 (1986)

[30] Gelareh, S., Nickel, S.: Hub location problems in transportation networks. Transportation Research Part E: Logistics and Transportation Review **47**(6), 1092 – 1111 (2011)

[31] Hamacher, H.W., Labbé, M., Nickel, S., Sonneborn, T.: Adapting polyhedral properties from facility to hub location problems. Discrete Applied Mathematics (145), 104 – 116 (2004)

[32] Hekmatfar, M., Pishvaee, M.: Hub location problem. In: R. Zanjirani Farahani, M. Hekmatfar (eds.) Facility Location, Contributions to Management Science, pp. 243–270 (2009)

[33] Júnior, J.C.A., Noronha, T.F., Santos, A.C.: O problema de localização de concentradores em árvores: um procedimento para acelerar um algoritmo baseado em decomposição de Benders. XLVII Simpósio Brasileiro de Pesquisa Operacional (2015)

[34] Karasan, O.E., Yaman, H., Ç. Pinar, M.: The robust shortest path problem with interval data. Tech. rep., Bilkent University, Department of Industrial Engineering (2001)

[35] Kasperski, A., Kobylański, P., Kulej, M., Zieliński, P.: Minimizing maximal regret in discrete optimization problems with interval data, pp. 193–208. Akademicka Oficyna Wydawnicza EXIT, Warszawa (2005)

[36] Kasperski, A., Zieliński, P.: The robust shortest path problem in series–parallel multidigraphs with interval data. Operations Reserach Letters **34**, 69–76 (2006)

[37] Kasperski, A., Zieliński, P.: On the existence of an FPTAS for minmax regret combinatorial optimization with interval data. Operations Reserach Letters **35**, 525–532 (2007)

[38] Klincewicz, J.: A dual algorithm for the uncapacitated hub location problem. Location Science (4), 173 – 184 (1996)

[39] Klincewicz, J.G.: Heuristics for the p-hub location problem. European Journal of Operational Research (53), 25 – 37 (1991)

[40] Kouvelis, P., Yu, G.: Robust discrete optimization and its applications. Kluver Academic Publishers (1997)

[41] Marín, A.: Uncapacitated euclidean hub location: Strengthened formulation, new facets and a relax-and-cut algorithm. Journal of Global Optimization (33), 393 – 422 (2005)

[42] Marín, A., Cánovas, L., Landete, M.: New formulations for the uncapacitated multiple allocation hub location problem. European Journal of Operational Research (172), 274 – 292 (2006)

[43] Mayer, G., Wagner, B.: Hublocator: an exact solution method for the multiple allocation hub location problem. Computers and Operations Research (29), 715 – 739 (2002)

[44] Miller, C., Tucker, A., Zemlin, R.: Integer programming formulations and traveling salesman problems. Journal of the ACM **7**, 326 – 329 (1960)

[45] Montemanni, R., Gambardella, L.M.: A branch and bound algorithm for the robust spanning tree problem with interval data. European Journal of Operational Research **161**, 771–779 (2005)

[46] Montemanni, R., Gambardella, L.M.: The robust shortest path problem with interval data via Benders decomposition. 4OR **3**, 315–328 (2005)

[47] Montemanni, R., Gambardella, L.M., Donati, A.V.: A branch and bound algorithm for the robust shortest path problem with interval data. Operations Reserach Letters **32**, 225–232 (2004)

[48] Nie, Y., Wu, X.: Shortest path problem considering on-time arrival probability. Transportation Research Part B **43**, 597–613 (2009)

[49] O'Kelly, M.E.: A quadratic integer program for the location of interacting hub facilities. European Journal of Operational Research (32), 393 – 404 (1987)

[50] Pérez, F., Astudillo, C.A., Bardeen, M., Candia-Véjar, A.: A simulated annealing approach for the minmax regret path problem. In: Proceedings of Congresso Latino Americano de Investigación Operativa/Simpósio Brasileiro de Pesquisa Operacional 2012. Rio de Janeiro, Brazil (2012)

[51] Pessoa, L., Santos, A.C., Resende, M.: A biased random-key genetic algorithm for the tree of hubs location problem. In: Proceedings of the 13ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Angers, France, pp. 458–459 (2012)

[52] Santos, A., Duhamel, C., Aloise, D.: Modeling the mobile oil recovery problem as a multiobjective vehicle routing problem. Modelling, Computation and Optimization in Information Systems and Management Sciences pp. 283–292 (2008)

[53] Skorin-Kapov, D., Skorin-Kapov, J., O'Kelly, M.: Tight linear programming relaxations of uncapacitated p-hub median problems. European Journal of Operational Research (94), 582 – 593 (1996)

[54] Spall, J.C.: Introduction to Stochastic Search and Optimization. Wiley (2003)

[55] Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. IEEE Transactions on Systems, Man, and Cybernetics **2**, 109–125 (1981)

[56] de Sá, E.M., de Camargo, R.S., de Miranda, G.: An improved Benders decomposition algorithm for the tree of hubs location problem. European Journal of Operational Research **226**(2), 185 – 202 (2013)

[57] Yu, G., Yang, J.: On the robust shortest path problem. Computers & Operations Research **25**, 457–468 (1998)