HEURÍSTICAS PARA O PROBLEMA DE CAMINHO MAIS CURTO ROBUSTO

AMADEU ALMEIDA COCO

HEURÍSTICAS PARA O PROBLEMA DE CAMINHO MAIS CURTO ROBUSTO

Dissertação apresentada ao Programa de Pós-Graduação em Departamento de 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 Departamento de Ciência da Computação.

Orientador: Thiago Ferreira de Noronha
Coorientadora: Andréa Cynthia Santos

Belo Horizonte
Março de 2013

Coco, Amadeu Almeida
C667h Heurísticas para o problema de caminho mais curto robusto / Amadeu Almeida Coco. — Belo Horizonte, 2013
xxi, 47 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação
Orientador: Thiago Ferreira de Noronha
Coorientadora: Andréa Cynthia Santos

1. Computação - Teses. 2. Programação Heurística - Teses. 3. Otimização Combinatória - Teses. I. Orientador. II. Coorientadora. III. Título.

CDU 519.6*62 (043)

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

FOLHA DE APROVAÇÃO

Heurísticas para o problema de caminho mais curto robusto

AMADEU ALMEIDA COCO

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 - França

PROF. GERALDO ROBSON MATEUS
Departamento de Ciência da Computação - UFMG

PROF. RICARDO MARTINS DE ABREU SILVA
Centro de Informática - UFPE

Belo Horizonte, 07 de março de 2013.

Dedico a todos que de alguma forma me ajudaram nessa caminhada

Agradecimentos

Aos meus pais Amadeu e Aparecida, a minha irmã Amanda e meu tio Lourenço, que sempre ajudaram nos momentos mais difíceis e compartilharam comigo os melhores momentos da minha vida.

Agradeço também ao meu orientador Thiago Noronha e a minha coorientadora Andréa Cynthia Santos pelo apoio e auxílio em toda caminhada pelo mestrado, por sempre me orientarem da melhor forma possível e me ajudarem a amadurecer durante este processo. Também agradeço a ambos pela oportunidade de trabalhar com eles e poder mostrar todo meu potencial presente neste trabalho.

Agradeço a meus colegas do LaPO por compartilharem diversas experiências e me ajudarem no dia a dia, com dicas simples, porém extremamente valiosas.

Agradeço aos amigos por estarem sempre do meu lado e ficarem curiosos com meu trabalho.

"Quando a gente acha que tem todas as respostas,
vem a vida e muda todas as perguntas ..."
(Luis Fernando Veríssimo)

Resumo

O problema de caminho mais curto (SP, do inglês *Shortest Path*) consiste em encontrar um caminho em um grafo $G = (V, A)$, partindo de um vértice de origem $s \in V$ até um vértice de destino $t \in V$, onde o custo total é mínimo. Porém, diversas aplicações para o SP dependem de dados não conhecidos à priori. O Problema do Caminho Mais Curto Robusto (RSP, do inglês *Robust Shortest Path*) generaliza o SP de forma que os dados incertos podem ser modelados como um intervalo ou como um conjunto de cenários discretos. Este trabalho foca no critério de otimização robusta *minmax* com arrependimento relativo utilizando a modelagem por intervalo para as incertezas. Foram desenvolvidas diversas heurísticas para resolver o RSP Intervalar com arrependimento relativo, tais como, heurísticas construtivas, heurísticas baseadas em *pilot method*, e um algoritmo genético de chaves aleatórias. Os resultados dos experimentos computacionais mostram que as heurísticas propostas neste trabalho obtiveram resultados melhores do que as heurísticas adaptadas da literatura.

Palavras-chave: Caminho Mais Curto Robusto, incerteza de dados, heurísticas, formulação matemática.

Abstract

The well-known Shortest Path problem (SP) consists in finding a shortest path from a source node $s \in V$ to a destination node $t \in V$ such that the total cost is minimized. However, the data of several SP applications is uncertain. The Robust Shortest Path problem (RSP) generalizes the SP once the uncertain data can be modeled with costs as an interval or else by a set of discrete scenarios. This dissertation focuses on the robust optimization criterion minmax relative regret with uncertain data modeled in an interval. We developed several heuristics with emphasis on providing efficient and scalable methods for solving the minmax relative regret RSP, such as constructive heuristics, pilot method based heuristics, and a random key genetic algorithm. The computational experiments show that the heuristics proposed in this work obtained better results than the heuristics proposed in the literature.

Keywords: Robust shortest path, uncertain data, heuristics, mathematical modeling.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Seja $G = (V, A)$ um grafo direcionado, no qual V é um conjunto de vértices e A é um conjunto de arcos, onde cada arco $(i, j) \in A$ está associado a um peso $c_{ij} \in \mathbb{R}$. Dados $\{s, t\} \in V$, o Problema do Caminho Mais Curto (SP, do inglês *Shortest Path Problem*) consiste em encontrar um caminho entre o vértice de origem s e o vértice de destino t em G cujo somatório dos custos associados aos arcos no caminho seja mínimo. SP não tem solução se existe um ciclo negativo entre s e t. O SP é estudado desde a década de 50 [Shimbel, 1955] e sua complexidade computacional é $\theta(|V| \cdot \log |V|)$ [Cormen et al., 2009]. Quando o custo associado aos arcos é sempre positivo, o algoritmo mais utilizado para sua resolução é o de Dijkstra [Dijkstra, 1959]. Quando o custo associado aos arcos pode ser negativo, o algoritmo mais utilizado na literatura é o de Bellmann-Ford [Bellman, 1958] com complexidade $\mathcal{O}(|V|^2 \cdot \log |V|)$ [Ahuja et al., 1993; Cormen et al., 2009].

SP modela diversos problemas de interesse prático e teórico [Gallo & Pallottino, 1986]. Entretanto, neste problema cada arco $(i, j) \in A$ está associado a um custo fixo, fato que impede que vários problemas de roteamento encontrados no mundo real possam ser modelados via SP, pois frequentemente os dados deste tipo de problema estão sujeitos a incertezas. Por exemplo, o problema de roteamento de uma ambulância que precisa socorrer um paciente, no menor tempo possível, partindo de um hospital até o local do acidente pode ser modelado como um SP da seguinte forma. Sejam s o vértice que representa o hospital (origem), t o vértice que representa o local do acidente (destino) e G um grafo direcionado que representa a malha viária, onde o peso dos arcos é igual ao comprimento dos respectivos seguimentos de rua, o problema de roteamento de ambulâncias pode ser resolvido calculando-se o caminho mais curto em G. Apesar do comprimento das ruas não variar, existem incertezas associadas ao trafego de veículos, portanto o tempo que um veículo demora para percorrer cada segmento

de rua pode variar. Desta forma, o caminho ótimo para SP, inicialmente o mais barato dentre todos os caminhos, pode acabar sendo um caminho ruim. Por outro lado, um caminho considerado em princípio pior pode passar a ser o melhor na prática. Uma alternativa neste caso seria rotear a ambulância fixando o custo de cada segmento de rua como o maior tempo de percurso já registrado para cada segmento individualmente. Entretanto, esta abordagem pode também levar a caminhos ineficientes, já que nem sempre todos os segmentos de rua estão congestionados ao mesmo tempo.

Os problemas no qual existem incertezas nos valores dos dados são geralmente resolvidos por duas técnicas: *Otimização Estocástica* [Spall, 2003] e *Otimização Robusta* [Ben-Tal & Nemirovski, 2002; Kouvelis & Yu, 1997]. Na Otimização Estocástica, os custos dos arcos do grafo G são definidos por variáveis aleatórias ou cenários discretos, onde a distribuição de probabilidades de cada cenário é conhecida. O Problema do Caminho Mais Curto Estocástico (SSP, do inglês *Stochastic Shortest Path Problem*) [Bertsekas & Tsitsiklis, 1991] consiste em encontrar um caminho entre o vértice origem s e o vértice de destino t em G cujo o custo esperado do somatório dos arcos do caminho seja mínimo.

Na otimização robusta, uma das técnicas comumente utilizadas é modelar o valor dos dados por um intervalo real de valores, limitado por um valor mínimo L e um valor máximo U. Desta forma, existem infinitos cenários que podem se realizar, já que o valor de cada dado do problema pode variar dentro do seu respectivo intervalo. Neste caso, o objetivo é encontrar a solução S^* que tenha o menor custo no seu respectivo pior cenário. Vale salientar que o pior cenário para uma solução S' pode ser diferente do pior cenário para uma solução S''. A figura 1.1 apresenta um exemplo do pior cenário para os caminhos $P' = \{(s,0),(0,2),(2,t)\}$ e $P'' = \{(s,1),(1,3),(3,t)\}$. Esta técnica é a mais indicada para modelar aplicações que são mais sensíveis ao pior caso.

A otimização robusta foi inicialmente aplicada com sucesso em problemas de planejamento de investimentos financeiros [Gupta & Rosenhead, 1968] e em problemas de tomada de decisões estratégicas [Rosenhead et al., 1972]. Atualmente, os problemas mais estudados na literatura de Otimização Robusta são o Problema do Caminho Mais Curto Robusto (RSP, do inglês *Robust Shortest Path problem*) [Adjiashvili & Zenklusen, 2011; Aissi et al., 2005, 2009; Averbakh & Lebedev, 2004; Ben-Tal & Nemirovski, 2002; Büsing, 2009; Catanzaro et al., 2011; Dias & Climaco, 2000; Karasan et al., 2001; Kasperski & Zieliński, 2006b, 2008, 2010; Montemanni & Gambardella, 2005b; Montemanni et al., 2004; Nie & Wu, 2009; Puhl, 2009; Salazar-Neumann, 2010; Seshadri & Srinivasan, 2010; Wu & Nie, 2010; Yu & Yang, 1997; Zieliński, 2004] e o Problema da Árvore Geradora Mínima Robusta (RST, do inglês *Robust Minimum Spanning Tree*) [Aissi et al., 2005, 2009; Aron & Hentenryck, 2004; Ben-Tal & Nemirovski, 2002; Büsing,

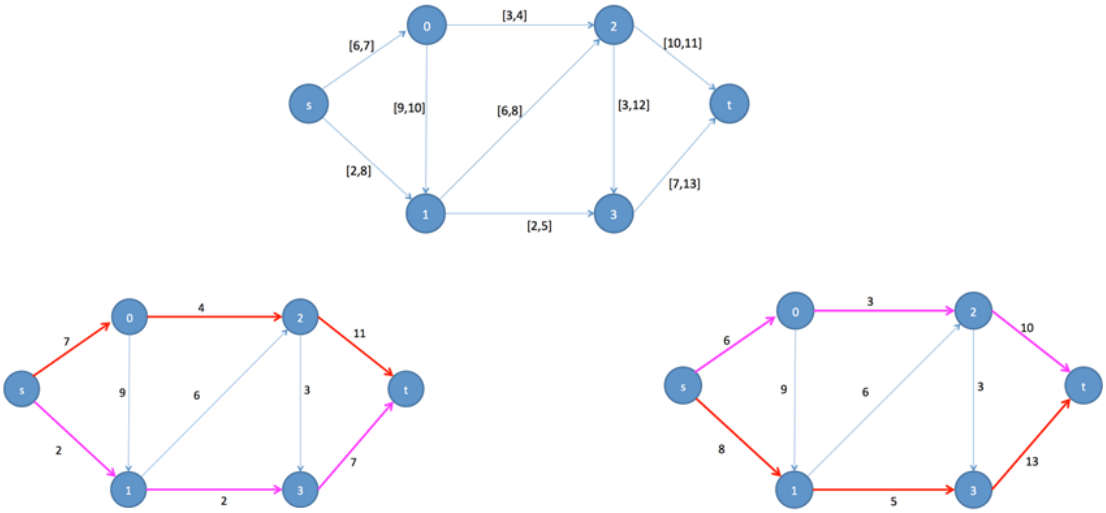

Figura 1.1. Exemplo de grafo para o problema de caminho mais curto robusto. No grafo superior é mostrado o grafo original. No grafo inferior a esquerda é mostrado o pior cenário para P' e no grafo inferior a direita é mostrado o pior cenário para P''.

2009; Kasperski & Zieliński, 2011; Montemanni & Gambardella, 2005a].

RSP é a versão do SP no qual existe incerteza nos valores dos arcos $(i, j) \in A$. As principais formas de modelar esta incerteza encontradas na literatura do RSP são: *modelagem por intervalos* [Averbakh & Lebedev, 2004; Catanzaro et al., 2011; Karasan et al., 2001; Kasperski & Zieliński, 2006b, 2010; Kouvelis & Yu, 1997; Montemanni & Gambardella, 2005b; Montemanni et al., 2004; Salazar-Neumann, 2010; Zieliński, 2004] e a *modelagem por cenários discretos* [Aissi et al., 2009; Dias & Climaco, 2000; Karasan et al., 2001; Kouvelis & Yu, 1997; Yu & Yang, 1997]. Na modelagem por intervalos, cada arco $(i, j) \in A$ pode assumir qualquer valor real no intervalo $[l_{ij}, u_{ij}]$, independentemente dos valores assumidos pelos outros arcos em A, desta forma, infinitos cenários são considerados. Já na modelagem por cenários discretos, um conjunto finito de cenários é definido, onde cada cenário consiste em uma atribuição única de diferentes valores para o custo de todos os arcos pertencentes a A.

Os principais critérios de avaliação utilizados para RSP são *absoluto* [Aissi et al., 2009; Gabrel et al., 2011; Karasan et al., 2001; Kasperski & Zieliński, 2008; Yu & Yang, 1997], *arrependimento absoluto* [Aissi et al., 2009; Averbakh & Lebedev, 2004; Candia-Véjar et al., 2011; Gabrel et al., 2011; Karasan et al., 2001; Kasperski & Zieliński, 2006b, 2007, 2008; Montemanni & Gambardella, 2005b; Montemanni et al., 2004; Yu & Yang, 1997; Zieliński, 2004] e *arrependimento relativo* [Averbakh, 2005; Gabrel et al., 2011; Libura, 2009]. No critério absoluto a solução ótima é dada pelo caminho que apresenta

a melhor solução no pior caso. No critério de avaliação *arrependimento absoluto*, dado $P \subseteq A$ um caminho entre s e t em G, o *desvio robusto* de P no cenário r é definido como a diferença entre (i) o custo do caminho P no cenário r e (ii) o custo do caminho mais curto entre s e t no cenário r, ou seja, o desvio robusto de P em r é o quanto P seria pior, em valor, que o melhor caminho possível caso o cenário r ocorresse. Denota-se por r^P o cenário onde observa-se o pior desvio robusto para o caminho P onde r^P é gerado fixando-se $c_{ij}^r = u_{ij}$, $\forall (i,j) \in P$, onde u_{ij} é o valor máximo do arco, e $c_{ij}^r = l_{ij}$, $\forall (i,j) \in A \setminus P$, onde l_{ij} é o valor mínimo do arco [Karasan et al., 2001]. Seja, também, o *custo robusto* de um caminho P definido como o desvio robusto de P em r^P, então a solução ótima deste critério de avaliação consiste em encontrar o caminho com o menor custo robusto dentre todos os caminhos entre s e t. Já no critério de avaliação *arrependimento relativo*, seja $P \subseteq A$ um caminho entre s e t em G, o *desvio robusto relativo* de P no cenário r é definido por $\frac{a-b}{b}$, onde a é o custo do caminho P no cenário r e b é o custo do caminho mais curto entre s e t no cenário r, ou seja, o desvio robusto relativo de P em r é o quanto P seria pior, relativamente ao melhor caminho possível caso o cenário r ocorresse. Seja o *custo robusto relativo* de um caminho P definido como o desvio robusto relativo de P em r^P, onde o cenário r^p é determinado de forma semelhante a versão com arrependimento absoluto [Karasan et al., 2001]. Portanto, neste critério, a solução é dada pelo caminho com o menor custo robusto relativo dentre todos os caminhos entre s e t.

O problema do caminho mais curto robusto com critério de avaliação absoluto e modelagem intervalar (RSP-IA) consiste em encontrar o caminho cujo somatório dos custos associados aos arcos no caminho seja mínimo no pior dentre todos os cenários. Este cenário é o cenário onde todos os arcos estão em u_{ij}. Esta versão do problema é polinomial [Kouvelis & Yu, 1997; Yu & Yang, 1997] e pode ser resolvida através do Algoritmo de Dijkstra [Dijkstra, 1959].

O problema do caminho mais curto robusto com critério de avaliação arrependimento absoluto e modelagem intervalar (RSP-IR) consiste em encontrar o caminho com o menor arrependimento absoluto dentre todos os caminhos entre s e t. Esta versão do problema é NP-Difícil [Kouvelis & Yu, 1997; Yu & Yang, 1997].

O problema do caminho mais curto robusto com critério de avaliação arrependimento relativo e modelagem intervalar (RSP-IRR) consiste em encontrar o caminho com o menor arrependimento relativo dentre todos os caminhos entre s e t. Esta versão do problema é NP-Difícil [Averbakh, 2005; Kouvelis & Yu, 1997].

O problema do caminho mais curto robusto com critério de avaliação absoluto e modelagem por cenários discretos (RSP-DA) consiste em encontrar o caminho que possui o menor custo máximo, considerando todos os cenários. Esta versão do problema

é NP-Difícil [Aissi et al., 2009; Kouvelis & Yu, 1997; Yu & Yang, 1997].

O problema do caminho mais curto robusto com critério de avaliação arrependimento absoluto e modelagem por cenários discretos (RSP-DR) consiste em minimizar, dentre todos os cenários, o máximo desvio robusto. Este problema é NP-Difícil [Aissi et al., 2009; Kouvelis & Yu, 1997; Yu & Yang, 1997].

O problema do caminho mais curto robusto com critério de avaliação arrependimento relativo e modelagem por cenários discretos (RSP-DRR). O RSP-DRR consiste em minimizar, entre todos os cenários, o máximo desvio robusto relativo. Este problema é NP-Difícil [Averbakh, 2005; Kouvelis & Yu, 1997].

As modelagens para o RSP descritas acima são apresentadas na Tabela 1.1. Na primeira coluna é mostrado o tipo de modelagem de cenário. Nas colunas seguintes são mostrados os nomes da versão de acordo com o grau de conservadorismo da solução utilizada.

Modelagem/Critério de Avaliação	Absoluto	Arrependimento Absoluto	Arrependimento Relativo
Contínuo	RSP-IA	RSP-IR	RSP-IRR
Discreto	RSP-DA	RSP-DR	RSP-DRR

Tabela 1.1. Diversas versões conhecidas para o Problema de Caminho Mais Curto Robusto (RSP) mostradas na literatura

Este trabalho trata do RSP-IRR tal como definido em [Averbakh, 2005; Kouvelis & Yu, 1997]. A Figura 1.2 apresenta um exemplo de RSP-IRR para um grafo com quatro vértices e seis arcos. Cada arco $(i, j) \in A$ está associado a um intervalo $[l_{ij}, u_{ij}]$. Por exemplo, o custo do arco $(s, 1)$ pode assumir qualquer valor real entre $l_{s1} = 4$ e $u_{s1} = 6$. O caminho com menor custo robusto relativo no grafo da Figura 1.2 é o caminho $P^* = \{(s, 0), (0, t)\}$. O custo de P^* no cenário r^{P^*}, onde r^{P^*} é o cenário induzido pelo caminho P^*, ou seja, o cenário onde o arco (i, j) é fixado em u_{ij} caso $(i, j) \in P^*$ e fixado em l_{ij} caso contrário, é $u_{s0} + u_{0t} = 2 + 7 = 9$. O custo do caminho mais curto $P' = \{(s, 1), (1, t)\}$ entre s e t no cenário r^{P^*} é $l_{s1}+l_{1t} = 4+3 = 7$. Portanto, o custo robusto de P^* é $(9 - 7)/7 = 2/7 = 0.286$.

Este trabalho é dividido em seis capítulos. No Capítulo 2 são apresentadas as referências bibliográficas para o RSP. No Capítulo 3 é proposta a primeira formulação linear para o RSP-IRR. No Capítulo 4 são descritas diversas heurísticas para este problema. No Capítulo 5 são realizados os experimentos computacionais para este problema. As conclusões deste trabalho são fornecidas no Capítulo 6.

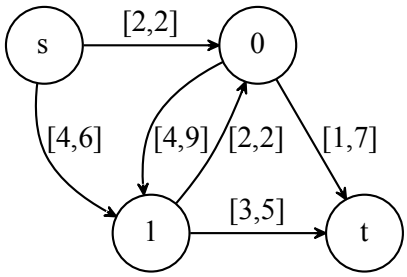

Figura 1.2. Exemplo de grafo para o problema de caminho mais curto robusto.

Capítulo 2

Trabalhos Relacionados

O problema de Caminho Mais Curto Robusto (RSP) é estudado desde o fim da década de 60 [Gupta & Rosenhead, 1968; Rosenhead et al., 1972]. Kouvelis & Yu [1997] sugeriram a utilização de aproximações robustas no auxílio à tomada de decisões quando o conhecimento sobre o estado atual de um problema não pode ser determinado. A aproximação robusta é dada por uma solução que protege contra o pior cenário que possa vir a surgir. Kouvelis & Yu [1997], também, apresentaram pela primeira vez as modelagens intervalar e por cenários discretos, além dos critérios de avaliação absoluto, com arrependimento absoluto e com arrependimento relativo. Por fim, foi provada a complexidade computacional para diversos problemas de otimização robusta. Por exemplo, foi provado que RSP e RST são NP-Difíceis.

Kasperski et al. [2005] apresentaram um *survey* sobre a modelagem intervalar com arrependimento absoluto de RSP, RST, Seleção de Itens Robusto (RSI do inglês *Robust Selecting Itens Problem*) e problema de assinalamento robusto (RAS, do inglês *Robust Assignment Problem*) contendo o estado da arte destes até 2005. Em [Kasperski et al., 2005] é mostrado que apenas o RSI-IR pode ser resolvido polinomialmente [Averbakh, 2001; Conde, 2004]. Outro *survey* sobre os problemas de otimização robusta foi apresentado em [Aissi et al., 2009]. Neste trabalho são apresentados algoritmos e complexidades de diversas versões do RSP, RST e do Problema da Mochila Robusto (RKP, do ingles *Robust Knapsack problem*). Um terceiro *survey* dedicado ao RSP é apresentado em [Gabrel et al., 2011]. Neste *survey* é apresentado o estado da arte para o RSP-DA, RSP-DR, RSP-IA e RSP-IR até 2010.

Yu & Yang [1997] trataram as versões RSP-DA e RSP-DR. Neste trabalho foi provado que estas versões são NP-Difíceis, inclusive quando apenas dois cenários são definidos. Além disso, Yu & Yang [1997] definiram dois algoritmos pseudo-polinomiais, chamados de ARSP e RDSP, para grafos acíclicos, ambos com complexidade $\mathcal{O}(|V|^3 \cdot \Delta)$,

onde Δ é o custo do caminho mais longo no cenário cujo caminho mais longo é o maior dentre todos os cenários. Portanto, o algoritmo ARSP procura um caminho que minimiza o comprimento máximo entre s e t sobre todos os cenários. Já o algoritmo RDSP procura um caminho P entre s e t que, ao longo de todos os cenários, minimiza a diferença máxima entre o custo do caminho P em um cenário r e o caminho mais curto em r.

Orgyczak [1997] desenvolveu o conceito de *minmax* lexicográfico que foi proposto como um refinamento do *minmax* tradicional em problemas de otimização robusta quando a modelagem do problema é dada por cenários discretos. No método *minmax* lexicográfico não se minimiza apenas o pior caso, como no *minmax* tradicional, mas também se minimiza o segundo pior caso, o terceiro, etc. A abordagem *minmax* lexicográfica refina o *minmax* tradicional na forma como é selecionado o conjunto dos resultados, que pode ser dado por mais de uma solução onde todas as soluções consideradas possuem exatamente a mesma distribuição de resultados. O autor considera que o *minmax* lexicográfico é factível com o princípio de otimalidade de Pareto e com o princípio de transferências, ao contrário *minmax* tradicional. O princípio de otimalidade de Pareto se refere a eficiência de um problema da classe *minmax*. Já o princípio de transferências é essencial para medidas de igualdade das soluções. Por fim, o autor aplica o *minmax* lexicográfico no problema de Localização de Facilidades e demonstra que os resultados obtidos são melhores que aqueles obtidos no *minmax* tradicional.

Dias & Climaco [2000] trabalharam com o conceito de dominância para o RSP-DA. Dados dois caminhos entre s e t, é dito que um caminho P' domina um caminho P'', se e somente se, P' é mais curto que P'' em todos os cenários. Além disto, Dias & Climaco [2000] apresentaram um algoritmo que encontra o conjunto de caminhos não dominados dentro de um grafo. Este algoritmo se inicia com o cálculo do custo do caminho mais curto em cada cenário e cada caminho encontrado é colocado na lista de caminhos não dominados. Em seguida, seleciona-se cada um dos caminhos restantes e testa-se eles são dominados seguindo as regras de dominância definidas em [Dias & Climaco, 2000].

Karasan et al. [2001] trabalharam com as versões RSP-IA e RSP-IR. Neste trabalho apareceu a primeira formulação para o RSP-IR baseada em programação inteira mista. Além disto, Karasan et al. [2001] introduziram o conceito de arco fraco (do inglês, *weak-arc*). Arco Fraco é aquele que jamais estará em uma solução ótima do problema. Baseado nisso, foi proposto um algoritmo de pré-processamento para o problema. Porém, o algoritmo funciona apenas para grafos acíclicos, planares ou em camadas.

Ben-Tal & Nemirovski [2002] trabalharam com diversos problemas de otimização robusta, tais como o problema da Árvore Geradora Mínima Robusta e o do Caminho Mais Curto Robusto. Além disto, os autores mostraram aplicações de otimização robusta nas áreas de projeto de redes e construção de treliças, que são construções compostas por finas barras elásticas de um dado peso, ligadas através de um conjunto de nós.

Bertsimas & Sim [2003] propuseram uma formulação baseada em programação inteira mista e teoria da dualidade [Dantzig, 1963] para problemas de fluxo em redes robustos quando a modelagem é dada por cenários discretos. Neste trabalho, também, foi proposta uma modelagem alternativa ao critério de avaliação absoluto quando a modelagem é dada por cenários discretos. Por fim, os autores propuseram um algoritmo polinomial para o problema de fluxo máximo em redes robusto, nas condições citadas anteriormente.

Averbakh & Lebedev [2004] provaram que o RSP-IR é NP-difícil quando o custo dos arcos varia somente em $\{0, 1\}$. Além disso, os autores provaram que o RSP-IR é polinomial no caso em que o número de arcos com valores incertos é limitado pelo logaritmo de uma função polinomial em relação ao número total de arcos.

Em [Montemanni et al., 2004; Montemanni & Gambardella, 2005a,b], foram propostos algoritmos exatos para RSP-IR. Montemanni et al. [2004] desenvolveram um algoritmo baseado na técnica *Branch-and-Bound* que encontra soluções ótimas em um tempo menor que aquele encontrado pela formulação matemática de [Karasan et al., 2001] implementada no CPLEX[1]. Em [Montemanni & Gambardella, 2005a], os resultados de [Montemanni et al., 2004] foram aprimorados através de técnicas de pré-processamento. Montemanni & Gambardella [2005b] propuseram um algoritmo baseado na decomposição de *Benders*, que obteve resultados semelhantes aos de [Montemanni & Gambardella, 2005b]. Os algoritmos apresentados em [Montemanni & Gambardella, 2005a,b; Montemanni et al., 2004] representaram uma evolução significativa na resolução do RSP-IR, pois foram os primeiros algoritmos exatos apresentados para resolução do problema quando a modelagem é intervalar. Montemanni & Gambardella [2005a,b]; Montemanni et al. [2004] resolveram instâncias com até 4.000 vértices e 160.000 arcos.

Averbakh [2005] estendeu o trabalho de [Kouvelis & Yu, 1997] e trabalhou com a versão robusta de problemas de otimização combinatória, onde a modelagem dos cenários é intervalar e o critério de avaliação da solução é dada por arrependimento relativo. Para esta classe de problemas, o autor propôs encontrar soluções que minimizam

[1] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

a diferença relativa do pior caso. Foi proposto, também, uma formulação genérica não linear para esta versão de problemas. Neste mesmo trabalho, também, foi provado que a complexidade desta versão de problemas robustos é Fortemente NP-Difícil, ou seja, não existem algoritmos aproximativos para esta versão. Por fim, foi apresentado o problema de sequenciamento robusto em uma máquina, que na versão clássica é polinomial e na versão robusta com arrependimento relativo é NP-Difícil.

Dhamdhere et al. [2005] introduziram o conceito de demanda robusta em problemas de otimização robusta intervalares. A motivação de [Dhamdhere et al., 2005] foi as abordagens que consideram o pior caso no modelo estocástico de duas fases [Birge & Louveaux, 1997]. Na primeira fase, dado um vetor aleatório de probabilidades chamado ϵ , as decisões x são tomadas na presença de incertezas em relação às futuras mudanças em ϵ. Já na segunda fase, o valor de ϵ se torna conhecido. Dessa forma, mudanças em x podem ser feitas ou uma nova decisão y pode ser tomada. Já na otimização robusta foi sugerido que na primeira fase o objetivo é encontrar a melhor solução S para um subconjunto de cenários. Enquanto na segunda fase, insere-se os cenários restantes e define-se se a solução S' será mantida com algumas modificações ou se uma nova solução S'' será tomada. Os autores também propuseram um algoritmo que garante, que ao final das duas fases, uma solução S de fator aproximativo 2 para RSP com demanda robusta será encontrada.

Golovin et al. [2006], estendendo o trabalho de [Dhamdhere et al., 2005], estudaram os problemas de corte mínimo robusto e caminho mais curto robusto intervalar e, em ambos, buscou-se um algoritmo aproximativo com fator melhor que o proposto por [Dhamdhere et al., 2005]. Para o problema de corte mínimo foi proposto um algoritmo com um fator de aproximação $1 + \sqrt{2}$. No problema de caminho mais curto robusto a aproximação foi por um fator 1.775. Para encontrar tais valores, os autores investigaram cada um dos cenários da segunda fase individualmente verificando se o atendimento do mesmo pode ser realizada de forma independente, dentro de um custo aceitável, ou seja, se o cenário pode ser atendido de forma independente dentro de uma estimativa segundo os custos deste estágio. Caso o cenário não possa ser atendido de forma independente, a solução encontrada é considerada como solução do primeiro estágio. Golovin et al. [2006] consideraram, também, uma versão chamada de *hitting robust demand*, para os problemas citados, no qual as técnicas propostas em [Dhamdhere et al., 2005] são combinadas com algoritmos utilizados para resolver os problemas de árvore de *Steiner* e Corte-Mínimo de *Steiner*. Para esta versão do problema, os autores obtiveram aproximações semelhantes à primeira versão de [Golovin et al., 2006] para os problemas de corte mínimo com demanda robusta e caminho mais curto com demanda robusta.

Kasperski & Zieliński [2006a] propuseram um algoritmo aproximativo com fator de aproximação 2 para versões robustas intervalares de problemas de otimização robusta. Este algoritmo aproximativo consiste em fixar os arcos do grafo em seus valores médios e rodar um algoritmo que funcione para o versão clássica do problema.

Kasperski & Zieliński [2006b] provaram que o RSP-IR é NP-Difícil mesmo em grafos série paralelos de Eppstein [Eppstein, 1992]. A importância desta prova deve-se ao fato de boa parte dos problemas considerados NP-Difíceis em otimização combinatória são polinomialmente resolvidos para grafos série paralelos de Eppstein [Eppstein, 1992]. Para estes casos, foi proposto um algoritmo pseudo-polinomial, de complexidade $\mathcal{O}(|A| \cdot L^2)$, onde A é o conjunto de arcos do grafo e L é tamanho do maior caminho, em número de arcos, entre origem e destino no cenário, onde todos os arcos estão fixados no limite superior. Este algoritmo, se subdivide em três rotinas. A primeira calcula o caminho mais curto nos trechos em série do grafo. A segunda calcula o caminho mais curto nos trechos paralelos do grafo. A terceira verifica quais são os nós que estão nos limites de um trecho em paralelo ou em série do grafo e marca-os com uma *flag*. Quando os dois nós são encontrados, verifica-se o tipo de trecho entre os nós e chama-se uma das rotinas anteriores, de acordo com o trecho encontrado.

Kasperski & Zieliński [2007], estendendo [Kasperski & Zieliński, 2006a], procuraram obter aproximações polinomiais para a versão robusta intervalar de problemas de otimização combinatória tais como RSP-IR e RST-IR. Neste mesmo artigo, é apresentado um algoritmo aproximativo com fator $(1 + \epsilon)$ para problemas robustos nos quais a solução do problema clássico é polinomial. Por fim, os autores propõem um algoritmo de complexidade $\mathcal{O}(|A| \cdot \Delta^2)$, onde A é o conjunto de arcos e Δ é igual ao maior valor do conjunto U que contém todos u_{ij}, $\forall (i,j) \in A$.

Escoffier et al. [2008] provaram que o RSP-IR pode ser resolvido em tempo $\mathcal{O}(|V|^2 \cdot |A|^k)$, onde k é o tamanho da instância. Eles, também, provam que para grafos com topologia similar a grafos do tipo série-paralelo [Eppstein, 1992], com topologia similar a árvores e com grafos acíclicos direcionados (DAG, do inglês *Direct Acyclic Graph*), o problema é pseudo polinomial. Os dois casos citados anteriormente são chamados pelos autores de instâncias tratáveis. Por fim, os autores propuseram algoritmos para as versões intervalares com arrependimento absoluto de RSP, Árvore Geradora Mínima Robusta (RST) e emparelhamento perfeito bipartido com pesos robusto intervalar com arrependimento (RBPM, do inglês *Robust minimum weighted Bipartite Perfect Matching*) polinomiais no caso onde o limite superior é dado por uma constante K e pseudo-polinomiais para o segundo quando a topologia do grafo é uma árvore, um grafo série-paralelo [Eppstein, 1992] ou um DAG.

Chen et al. [2009] trabalharam com uma versão do RSP denominada problema

de planejamento de caminhos (PPP, do inglês *Path Planning Problem*). No artigo é proposta uma formulação para o problema quando os arcos estão modelados de forma intervalar. Os autores adaptaram a formulação do RSP-IA [Karasan et al., 2001] para que a função objetivo encontre o caminho que contém a menor soma de riscos. O risco de um arco é calculado através da fórmula $\frac{u_e - x_e}{u_e - l_e}$, onde u_e é o limite superior do arco, l_e é o limite inferior e x_e é o valor do arco no cenário fixado. O objetivo do trabalho é encontrar caminho entre s e t em G que possa ser percorrido em tempo menor ou igual a uma constante definida à priori e que a soma dos riscos do caminho seja minimizado. Por fim, os autores resolvem este problema através de um algoritmo com complexidade $\mathcal{O}(|V|^3 \cdot |A|)$.

Puhl [2009] trabalhou com o problema de Caminho Mais Curto Robusto Recuperável (RRSP, do inglês *Recoverable Robust Shorthest Path Problem*), que combina a otimização robusta com um processo de programação estocástica de duas fases [Birge & Louveaux, 1997]. Neste trabalho, é apresentado duas abordagens de RRSP. A primeira abordagem, chamada de *k-Arc-RRSP*, se divide em duas fases. Na primeira fase, define-se um caminho entre s e t. Na segunda fase, pode-se escolher um novo caminho ou permanecer com o caminho escolhido na primeira fase, porém um novo caminho só pode ser escolhido se ele tiver no máximo k arcos diferentes do caminho definido na primeira fase. Na segunda abordagem, chamada de *Rent-RRSP*, também se divide em duas fases. Na primeira fase, define-se um caminho entre s e t. Na segunda fase, pode-se definir qualquer caminho entre s e t, porém os custos dos arcos utilizados no caminho da primeira fase e não utilizados no caminho relativo à segunda fase devem ser adicionados ao custo final da solução. Tanto *k-Arc-RRSP* quanto *Rent-RRSP* são NP-difíceis.

Nie & Wu [2009] abordaram uma versão do problema RSP Estocástico onde o objetivo é encontrar o caminho que possui o menor desvio robusto. Porém, diferentemente do RSP-IR, o desvio robusto definido pelo autor é dado pela diferença entre o custo do caminho no pior cenário e no melhor cenário. Foi provada que a complexidade desta versão do problema é a mesma que a do RSP-IR.

Libura [2009] considerou um problema de otimização robusta genérico onde o intervalo de cada arco é dado de forma dependente de um parâmetro $\delta \in [0, 1)$. O foco dos autores no artigo não foi encontrar uma solução robusta para um conjunto de cenários correspondente a algum δ, mas verificar a robustez de uma solução para os pesos iniciais do problema. Em particular, os autores estão interessados no maior valor de δ no qual tal solução permanece robusta, visto que o valor δ é chamado de raio de robustez.

Kasperski & Zieliński [2010] propuseram uma modelagem do problema baseada

em lógica *Fuzzy* para o RSP e provaram que a complexidade desta versão do problema é NP-Difícil. Por fim, demonstraram que encontrar o ótimo nesta modelagem é tão difícil quanto na modelagem robusta.

Seshadri & Srinivasan [2010] propuseram uma versão do RSP chamada de *Optimal Reliability Path* (ORP) onde o objetivo é calcular o caminho que possui a máxima confiabilidade, ou seja, aquele que tem a maior probabilidade de ser percorrido em um tempo inferior ao pré-estabelecido. Para isso, Seshadri & Srinivasan [2010] propuseram um algoritmo que encontra o ORP. O algoritmo determina os K caminhos mais curtos e avalia a confiabilidade de cada um destes caminhos. A seguir, é determinado o limite inferior sobre a variância do caminho mais curto. Depois, computa-se o limite superior de confiabilidade dos caminhos. Em seguida, confere-se os critérios de otimalidade e parada estabelecidos pelos autores que são, respectivamente, limite inferior maior ou igual ao menor limite superior, ou número de iterações máximo atingido. Caso as condições anteriores não sejam atendidas é realizado um sorteio no vetor do tempo de ligação das viagens, do cálculo do caminho com máxima confiabilidade condicional e atualização dos limites.

Roy [2010] refere-se ao termo otimização robusta como a necessidade em prevenir impactos indesejáveis tais como a degradação das propriedades a serem mantidas quando ocorre uma mudança de cenário. O autor considerou também que a otimização robusta pode ser relacionada e/ou integrada a noções de flexibilidade, estabilidade, sensibilidade e até de igualdade. Por fim, o autor definiu uma nova medida de robustez alternativa ao *minmax*. Tal método, chamado de *(b,w)-robustness* onde b é não-negativo e caracteriza um valor máximo que não pode ser excedido na maior parte dos cenários. Já a constante w, onde $w \leq max_x min_s(v_s(x))$, define um valor que não deve ser excedido por nenhum cenário. *(b,w)-robustness* é utilizado apenas nos casos em que o número de cenários é modelado por cenários discretos e o número de cenários definidos é muito grande. Por fim, deve-se salientar que assim como o critério *minmax*, o critério *(b,w)-robustness* se divide em *(b,w)-robustness absoluto*, *(b,w)-robustness com arrependimento* e *(b,w)-robustness com arrependimento relativo*.

Catanzaro et al. [2011] estendeu o trabalho de [Karasan et al., 2001] e desenvolveu quatro algoritmos de pré-processamento para RSP-IR em DAGs e grafos com poucos ciclos. O primeiro algoritmo, de complexidade $\mathcal{O}(|V|^2 \cdot |A|)$, onde V é o conjunto de vértices e A é o conjunto de arcos do grafo, procura arcos que, se retirados, resultam em uma solução com custo robusto maior que a solução original no pior caso. O segundo algoritmo, com complexidade $\mathcal{O}(|V|^3)$, procura encontrar vértices que não estão contidos em nenhum caminho mais curto entre origem e destino. O terceiro algoritmo, de complexidade $\mathcal{O}(|V|^2 \cdot |A|)$, parte da árvore geradora mínima gerada pelo

cenário quando todos os arcos do grafo original estão no valor máximo. A seguir é gerado o caminho mais curto entre s e t, considerando apenas os arcos da árvore. No passo seguinte, seleciona-se um arco presente neste caminho, retira-o e recalcula-se o caminho mais curto, caso o custo do novo caminho mais curto seja pior que o anterior, o arco pode estar na solução ótima. O quarto algoritmo, de complexidade $\mathcal{O}(|V|^2 \cdot |A|)$, parte de uma árvore geradora mínima obtida a partir do grafo original no cenário no qual todos os arcos do grafo estão em valor mínimo. A seguir, encontra-se o caminho mais curto dentro dentro da AGM gerada. Em seguida, escolhe-se um arco que não está na árvore e insere-o nela. Depois, recalcula-se o caminho mais curto e, caso o arco inserido não esteja no caminho, ele é considerado como um arco que não estará no caminho ótimo.

Candia-Véjar et al. [2011] trabalharam com complexidade de algoritmos de problemas de otimização robusta com modelagem intervalar. Os autores partiram de definições e propriedades comuns aos problemas da classe propostas em [Averbakh & Lebedev, 2004]. A seguir foram apresentadas as complexidades da versão *minmax* com arrependimento absoluto dos problemas de p-medianas, caminho mais curto, árvore geradora mínima, assinalamento, programação linear e caixeiro viajante. Em seguida, os autores apresentaram as formulações matemáticas destes problemas e por fim foram mostradas algoritmos aproximativos e suas respectivas provas.

Hasuike [2011] trabalhou com uma versão do RSP-IR no qual cada rota tem uma importância diferente, ou seja, fatores como o tipo de estrada, o tamanho do veículo e o tipo de carregamento do veículo são considerados. Para esta versão do problema, uma formulação não linear inteira mista foi proposta. Em seguida, foram apresentadas transformações determinísticas na formulação do problema de modo que esta possa ser resolvida de forma analítica. Como a formulação permanece não linear, alguns *solvers* comerciais como o CPLEX[2] não podem ser utilizados para resolvê-la. Por fim, o autor propôs um algoritmo aproximativo com fator de aproximação 2 para obter uma boa solução para o problema.

Conde [2012] trabalhou com problemas de otimização robusta na modelagem intervalar e critério de avaliação com arrependimento absoluto. Assim como [Kasperski & Zieliński, 2006a,b, 2007], Conde [2012] procurou resolver a versão clássica do problema sob o cenário médio, ou seja, quando o custo de todos os arcos são fixados no ponto médio do intervalo de valores possíveis para tal arco. Dessa forma, uma aproximação de fator dois pôde ser obtida para vários problemas de otimização *minmax* com arrependimento absoluto [Kasperski & Zieliński, 2007]. Além disto, o autor estende a

[2]http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

abordagem de aproximação por fator dois para problemas nos quais a incerteza está em algum outro dado além do arco, como por exemplo, custos ou demandas robustas.

Kalai et al. [2012] propuseram o critério α-robustness. Para justificar a proposta, os autores consideraram que o critério *minmax* valoriza em demasiado o pior cenário. A aproximação α-lexicográfica, que funciona apenas para a versão de cenários discretos, estende o critério *minmax* lexicográfico [Orgyczak, 1997] no sentido em que não se considera apenas o pior cenário. Em ambos os critérios, considera-se o custo no segundo pior cenário, o custo no terceiro pior cenário e etc. Mais precisamente, é calculado um vetor de soluções chamado de vetor ideal, onde este vetor é composto pela melhor solução obtida em cada cenário. Após o cálculo do vetor ideal, calcula-se a distância, em termos de resultados, de cada solução possível para o problema em relação a solução ideal. Quanto mais próxima uma solução encontra-se da solução ideal, mais robusta esta é considerada.

Recentemente, Pérez et al. [2012] propuseram um algoritmo baseado em *simulated annealing* para o RSP-IR. Este *simulated annealing* é uma das metaheurísticas pioneiras na resolução do RSP-IR. Os resultados indicam uma melhoria média de 0.3% em relação aos resultados apresentados em [Kasperski & Zieliński, 2006a]. Neste trabalho, também, foram resolvidas instâncias com até 20.000 vértices em tempos inferiores a 1240 segundos.

Até onde sabemos, não existem na literatura de RSP-IRR trabalhos que estudam e avaliam heurísticas ou algoritmos exatos para esta versão do problema. Sendo assim, este trabalho propõe as primeiras heurísticas para o problema com o intuito de resolver instâncias maiores de RSP-IRR em tempo polinomial. Neste trabalho, também, é apresentada a primeira formulação linear para o RSP-IRR.

Capítulo 3

Formulações

Na Seção 3.1, a formulação por programação linear inteira mista para o RSP-IR proposta em [Karasan et al., 2001] é descrita. Na Seção 3.2, a formulação não linear para o RSP-IRR proposta em [Averbakh, 2005] é apresentada. Além disto, na Seção 3.2 é proposta uma linearização para a formulação de [Averbakh, 2005] que funciona apenas para grafos acíclicos. O objetivo da formulação proposta em 3.2 é tentar resolver o RSP-IRR utilizando um *solver* comercial como CPLEX [1].

3.1 Caminho Mais Curto Robusto com Arrependimento Absoluto

Uma formulação por programação linear inteira mista para o RSP-IR foi proposta em [Karasan et al., 2001]. Existem dois conjuntos de variáveis na formulação. Para cada arco $(i, j) \in A$, são definidas variáveis $y_{ij} \in \{0, 1\}$ de forma que $y_{ij} = 1$ se o arco (i, j) faz parte do caminho mais curto robusto, e $y_{ij} = 0$ em caso contrário. As variáveis $x_i \in \mathbb{R}, \forall i \in V$, armazenam o custo do caminho mais curto entre o vértice origem s e o vértice i no cenário induzido pelo caminho definido pelas variáveis y_{ij}. A formulação é definida pelas equações (3.1)-(3.6).

[1] `http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/`

$$\min \sum_{(i,j)\in A} u_{ij}y_{ij} - x_t \tag{3.1}$$

Sujeito a:

$$x_j \leq x_i + l_{ij} + (u_{ij} - l_{ij})y_{ij} \quad \forall (i,j) \in A \tag{3.2}$$

$$\sum_{(j,k)\in A} y_{jk} - \sum_{(i,j)\in A} y_{ij} = \begin{cases} 1; & \text{se j = s.} \\ -1; & \text{se j = t.} \\ 0; & \text{caso contrário} \end{cases} \quad \forall j \in V \tag{3.3}$$

$$x_s = 0, \tag{3.4}$$

$$y \in \{0,1\} \quad \forall (i,j) \in A, \tag{3.5}$$

$$x_j \geq 0 \quad \forall j \in V \tag{3.6}$$

A função objetivo (3.1) minimiza o custo robusto do caminho entre s e t definido pelas variáveis y. As restrições da Equação (3.2) garantem a consistência entre as variáveis x e y e determinam um limite superior para o custo do arrependimento absoluto no vértice j. Estas restrições juntamente com a função objetivo asseguram que o arrependimento máximo será mínimo. A Equação (3.3) define as restrições de conservação do fluxo entre s e t. A Restrição (3.4) fixa $x_s = 0$. As restrições (3.5) e (3.6) definem os domínios das variáveis x e y.

O modelo definido em [Karasan et al., 2001] foi resolvido de forma eficiente pelo *solver* comercial CPLEX. Por exemplo, Montemanni et al. [2004] resolveram instâncias geradas aleatoriamente com 900 vértices e uma instância real, da rede rodoviária de Stuttgart, na Alemanha, com 2490 vértices em poucos segundos. Instâncias com 1000 vértices foram resolvidas na otimalidade e o *solver* CPLEX interrompeu a execução para instâncias variando entre 10000 e 20000 vértices devido à falta de memória. Os *gaps* para estas instâncias variaram entre 2% e 9%.

3.2 Caminho Mais Curto Robusto com Arrependimento Relativo

Uma formulação por programação não linear inteira mista para o RSP-IRR foi proposta em [Averbakh, 2005], baseada na formulação de [Karasan et al., 2001]. A diferença entre as duas formulações está na função objetivo. A função objetivo na formulação de [Karasan et al., 2001] é linear enquanto em [Averbakh, 2005], a função objetivo é

não linear. Vale salientar que a formulação de [Averbakh, 2005] é valida apenas para grafos acíclicos e não pode ser utilizado em grafos com ciclos. A formulação é dada pelas equações (3.7)-(3.12).

$$\min \frac{\sum_{(i,j)\in A} u_{ij}y_{ij} - x_t}{x_t} \tag{3.7}$$

Sujeito a:

$$x_j \leq x_i + l_{ij} + (u_{ij} - l_{ij})y_{ij} \quad \forall (i,j) \in A \tag{3.8}$$

$$\sum_{(j,k)\in A} y_{jk} - \sum_{(i,j)\in A} y_{ij} = \begin{cases} 1; & \text{se j = s.} \\ -1; & \text{se j = t.} \\ 0; & \text{caso contrário} \end{cases} \quad \forall j \in V \tag{3.9}$$

$$x_s = 0, \tag{3.10}$$

$$y \in \{0,1\} \quad \forall (i,j) \in A, \tag{3.11}$$

$$x_j \geq 0 \quad \forall j \in V \tag{3.12}$$

A função objetivo (3.7) minimiza o custo robusto relativo do caminho definido pelas variáveis y. As restrições (3.8)-(3.12) são equivalentes as restrições (3.2)-(3.6). Como a função objetivo é não linear, não é possível utilizar técnicas de programação linear inteira para resolver a formulação (3.7)-(3.12). Neste trabalho, é proposta uma linearização da função objetivo, inspirada no trabalho de [Bisschop, 2005] considerando o caso que $l_{ij}, u_{ij} \in \mathbb{N}$. Assim, pode-se utilizar técnicas de programação linear inteira mista para resolver RSP-IRR.

Inicialmente, a função objetivo (3.7) é reescrita como:

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

considerando que: $\frac{x_t}{x_t} = 1$, então a equação anterior pode ser reescrita por

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

Em seguida, são introduzidas variáveis $z_{ij} \in \mathbb{R}$.

$$z_{ij} = \frac{y_{ij}}{x_t} \quad , \forall (i,j) \in A,$$

Desta forma a função objetivo pode ser escrita como (3.13).

$$\min \sum_{(i,j)\in A} u_{ij} z_{ij} - 1 \tag{3.13}$$

Para garantir que

$$z_{ij} = \frac{y_{ij}}{x_t} \quad \forall (i,j) \in A$$

são acrescentadas as restrições (3.14)-(3.16)

$$z_{ij} \leq y_{ij} \forall (i,j) \in A \tag{3.14}$$

$$z_{ij} \geq \frac{1}{x_t} - (1 - y_{ij}) \quad \forall (i,j) \in A \tag{3.15}$$

$$z_{ij} \in [0,1] \quad \forall (i,j) \in A \tag{3.16}$$

A restrição 3.14 garante que $z_{ij} = 0$ quando $y_{ij} = 0$. Já a restrição 3.15 garante que $z_{ij} = \frac{1}{x_t}$ quando $y_{ij} = 1$. Note que a restrição (3.15) permanece não linear. Seja L, o valor do caminho mais curto no cenário em que todos os arcos $(i,j) \in A$ estão fixados em l_{ij} e U, o valor do caminho mais curto no cenário em que todos os arcos $(i,j) \in A$ estão fixados em u_{ij}. Para linearizar a restrição (3.15), cria-se as variáveis w_l, para todo $l \in \{L, ..., U\}$, onde $w_l = 1$ se $x_t = l$ e $w_l = 0$, caso contrário. Dessa forma, a restrição (3.15) pode ser substituída pelas restrições (3.17)-(3.20).

$$z_{ij} \geq \sum_{l\in\{L,...,U\}} (\frac{1}{l} \cdot w_l) - (1 - y_{ij}) \quad \forall (i,j) \in A \tag{3.17}$$

$$\sum_{l\in\{L,...,U\}} w_l = 1 \tag{3.18}$$

$$\sum_{l\in\{L,...,U\}} l \cdot w_l = x_t \tag{3.19}$$

$$w_l \in \{0,1\} \quad \forall l \in [L,U] \tag{3.20}$$

Portanto, a formulação linear do RSP-IRR é constituída pela função objetivo (3.13) e as restrições (3.8)-(3.12), (3.14) e (3.16)-(3.20).

Capítulo 4

Heurísticas para RSP-IRR

Kasperski & Zieliński [2006b, 2007, 2008] desenvolveram duas heurísticas, com fator de aproximação 2 para o RSP-IR. Estas heurísticas podem ser aplicadas diretamente para RSP-IRR. A primeira, chamada aqui de U-Kasperski, recebe como entrada um grafo $G = (V, A)$, os limites inferior l_{ij} e superior u_{ij} para o custo de cada arco $(i, j) \in A$, além do vértice de origem $s \in V$ e do vértice de destino $t \in V$. Em seguida, fixa-se o cenário r^u, onde o custo de todos os arcos estão em seu respectivo valor máximo, ou seja, $c^u_{ij} = u_{ij}$, $\forall (i, j) \in A$. Em seguida, calcula-se o caminho mais curto P^u entre s e t no cenário r^u, utilizando o algoritmo de Dijkstra. Por fim, o custo robusto de P^u é calculado e esta solução é retornada. A complexidade assintótica desta heurística no pior caso é $O(|V| \cdot \log |V|)$, assumindo-se que $|V| \cdot \log |V| \geq |A|$.

A segunda heurística, chamada aqui de M-Kasperski, recebe os mesmos parâmetros de entrada que U-Kasperski. Primeiramente, fixa-se o cenário r^m, onde o custo de todos os arcos estão em seu respectivo valor médio, ou seja, $c^m_{ij} = (l_{ij} + u_{ij})/2$, $\forall (i, j) \in A$. Em seguida, calcula-se o caminho mais curto P^m entre s e t no cenário r^m, utilizando algoritmo de Dijkstra. Por fim, o custo robusto de P^m é calculado e esta solução é retornada. A complexidade desta heurística no pior caso é $O(|V| \cdot \log |V|)$, assumindo-se que $|V| \cdot \log |V| \geq |A|$. Vale salientar que, apesar de estar provado que as heurísticas são 2-aproximadas para RSP-IR, não está provado que elas são 2-aproximadas para RSP-IRR.

Pérez et al. [2012] propuseram uma heurística baseada em *Simulated Annealing* para o RSP-IR, inspirada no trabalho de [Kirkpatrick et al., 1983]. Esta heurística também pode ser aplicada diretamente para RSP-IRR. A heurística, chamada aqui de SA-RSP, recebe como entrada um grafo $G = (V, A)$, os limites inferior l_{ij} e superior u_{ij} para o custo de cada arco $(i, j) \in A$, os vértices de origem $s \in V$ e destino $t \in V$. Esta heurística recebe como parâmetros (i) uma temperatura inicial t_0, (ii) uma temperatura

final t_f, (iii) um parâmetro de redução de temperatura β (onde $\beta < 1$), (iv) um valor inteiro L que representa o número de soluções aceitas necessárias para que a temperatura seja diminuída e, (v) um critério de aceitação C dado por $e^{-\Delta f_i/T} < \theta$, onde Δf_i é a diferença entre a solução encontrada e a solução atual, T é a temperatura na iteração corrente e $\theta \in U[0,1]$ é um valor definido de forma aleatória a cada iteração. Inicialmente, executa-se o algoritmo U-Kasperski [Kasperski & Zieliński, 2006b, 2007] para que uma solução inicial seja obtida. Enquanto $t_0 \geq t_f$, executa-se os quatro passos seguintes. (i) Gera-se um novo conjunto de arcos A', onde alguns arcos da solução corrente são retirados e outros arcos pertencentes a A, mas não a solução corrente são inseridos em A' de forma que a relação $A' \subset A$ sempre seja mantida, (ii) executa-se U-Kasperski [Kasperski & Zieliński, 2006b, 2007] utilizando A' ao invés de A como parâmetro de entrada, (iii) aceita-se a nova solução gerada no passo (ii) caso C seja satisfeito e, (iv) após L iterações nas quais uma solução é aceita, calcula-se a nova temperatura $t_0 = t_0 * \beta$. Por fim, retorna-se a melhor solução encontrada.

Neste capítulo são propostas três novas heurísticas para resolver RSP-IRR. Na Seção 4.1 são apresentadas heurísticas baseadas em k caminhos mais curtos [Martins & Pascoal, 2003; Yen, 1971]. Em seguida, na Seção 4.2 é apresentada uma heurística baseada na metaheurística Pilot Method [Duin & Voss, 1994]. Por fim, na Seção 4.3 é apresentada uma heurística baseada em algoritmos genéticos com chaves aleatórias [Bean, 1994; Gonçalves et al., 2009]. No Capítulo 5, estas heurísticas são comparadas com M-Kasperski [Kasperski & Zieliński, 2006b, 2007], U-Kasperski [Kasperski & Zieliński, 2006b, 2007] e SA-RSP [Pérez et al., 2012].

4.1 K-Shortest Path

A heurística K-Shortest Path (KSP) é apresentada em duas versões (i) Disjoint K-Shortest Path (D-KSP) e (ii) Yen's K-Shortest Path (Y-KSP) que é baseada no algoritmo de Yen [Martins & Pascoal, 2003; Yen, 1971] para calcular os k caminhos mais curtos em um grafo.

A heurística D-KSP recebe como entrada um grafo $G = (V, A)$, os limites inferior l_{ij} e superior u_{ij} para o custo de cada arco $(i, j) \in A$, além do vértice de origem $s \in V$ e de destino $t \in V$. Primeiramente, fixa-se o cenário r^m, onde o custo de todos os arcos estão em seu respectivo valor médio, ou seja, $c_{ij}^m = (l_{ij}+u_{ij})/2, \forall (i, j) \in A$. Em seguida, D-KSP calcula todos os caminhos mais curtos disjuntos entre s e t, ou seja, encontra o caminho mais curto entre s e t no cenário r^m, armazena-se o caminho resultante P^m, retiram-se os arcos de P^m de G e repete-se este procedimento, enquanto existir um

caminho entre s e t em G. O custo robusto relativo de todos os caminhos gerados é avaliado. Aquele com o menor custo robusto relativo é retornado. Nesta heurística, pode-se ter no máximo $|A|$ caminhos. Sendo assim, a complexidade computacional de D-KSP no pior caso é $\mathcal{O}(|A| \cdot |V| \cdot \log |V|)$, assumindo-se que $|V| \cdot \log |V| \geq |A|$.

A heurística Y-KSP recebe como entrada um grafo $G = (V, A)$, os limites inferior l_{ij} e superior u_{ij} para o custo de cada arco $(i, j) \in A$, além do vértice de origem $s \in V$ e do vértice de destino $t \in V$. Primeiramente, fixa-se o cenário r^m, onde o custo de todos os arcos estão em seu respectivo valor médio, ou seja, $c_{ij}^m = (l_{ij} + u_{ij})/2$, $\forall (i, j) \in A$. Em seguida, a heurística Y-KSP calcula os k caminhos mais curtos entre s e t em G no cenário r^m, utilizando o algoritmo de Yen [Martins & Pascoal, 2003; Yen, 1971] cuja complexidade no pior caso é $\mathcal{O}(|V|^2 \cdot \log |V|)$, assumindo-se que $|V| \cdot \log |V| \geq |A|$. Como o número de caminhos cresce exponencialmente em função de $|A|$, apenas os k caminhos mais curtos são considerados, onde k é um parâmetro a ser ajustado. O custo robusto relativo de todos os caminhos gerados é avaliado e aquele com o menor custo robusto relativo é retornado. Sendo assim, a complexidade computacional de Y-KSP é $\mathcal{O}(|V|^2 \cdot \log |V|)$, assumindo-se que $|V| \cdot \log |V| \geq |A|$, passa a ser escrita como $\mathcal{O}(k \cdot |V|^2 \cdot \log |V|)$.

4.2 Pilot Method

Pilot Method [Duin & Voss, 1994, 1999; Voss et al., 2005] é uma metaheurística que consiste em utilizar uma heurística construtiva gulosa H para construir uma nova heurística mais eficaz H' para resolver um problema de otimização. Esta última, em geral possui quatro propriedades. Estas propriedades são (i) H' é uma heurística gulosa, (ii) H' é um procedimento de tentativa e erro, (iii) H' faz fixação de valores para uma ou mais variáveis e (iv) H' verifica as consequências de uma possível escolha. Em princípio, H' funciona como uma heurística construtiva tradicional que insere iterativamente um elemento por vez na solução. Entretanto, em vez de usar um critério guloso local para avaliar um elemento que pode entrar na solução, o critério usado por H' consiste em (i) inserir o elemento individualmente na solução (ii) executar a heurística H até o fim, e (iii) utilizar o custo da solução resultante como custo guloso do elemento. A cada iteração, estes três passos são executados para todos os elementos e aquele que obtém o melhor custo guloso é inserido como parte da solução.

Pilot Method foi utilizado com sucesso para problemas de otimização combinatória NP-difíceis, tais como o Problema do Caixeiro Viajante [Duin & Voss, 1999], o Problema da Árvore de *Steiner* [Duin & Voss, 1994, 1999], problemas de árvores

geradoras [Martins, 2007; Xiong et al., 2006] , o Problema de Carregamento de Contêineres [Eley, 2002], o Problema de Design de Redes SDH/WDM [Höller et al., 2008] e problemas de sequenciamento [Bertsekas & Castanon, 1999; Fink & Voss, 2003; Meloni et al., 2004].

Neste trabalho foi desenvolvida uma heurística baseada Pilot Method, chamada aqui de PM, para resolver o RSP-IRR. Ela recebe como entrada um grafo $G = (V, A)$, os limites inferior l_{ij} e superior u_{ij} para o custo de cada arco $(i, j) \in A$, além do vértice de origem $s \in V$ e do vértice de destino $t \in V$. A seguir os custos de todos os arcos no grafo são fixados no cenário r^m onde $c_{ij}^m = (l_{ij} + u_{ij})/2$, $\forall (i, j) \in A$.

O Algoritmo 1 apresenta um pseudocódigo de PM. Na Linha 2, o caminho mestre P' é inicializado com o vértice s e a melhor solução P^* é inicializada como vazia. O laço das linhas 3-12 é repetido enquanto P' não é um caminho entre os vértices s e t, ou seja, enquanto o vértice t não é inserido em P'. Na Linha 4, seleciona-se o vértice w que é o vértice que foi inserido em P' na última iteração. O laço das linhas 5-9 é realizado para cada vértice $v \in \delta^+(w)$, onde $\delta^+(w)$ é o conjunto de vértices v no qual $(w, v) \in A$. Na Linha 6, executa-se Dijkstra entre os vértices v e t. Na Linha 7, concatena-se o caminho P' (entre s e w) e P_v (entre u e t) formando o caminho P'_v entre s e t. Na Linha 8, verifica-se se o custo robusto relativo de P'_v é menor que o de P^*_v, onde P^*_v é a solução com menor custo robusto relativo encontrada na iteração atual. Em caso positivo, P^*_v recebe a solução P'_v e o vértice v é passado para v^* na Linha 9. Na Linha 10, verifica-se se o custo robusto relativo de P^*_v é menor que P^*. Neste caso, P^* recebe a solução P^*_v na linha 11. Na Linha 12 insere-se v^* no fim de P'. Por fim, na Linha 13, retorna-se P^*. Vale salientar que P^* é o melhor dentre todos os caminhos avaliados ao longo da heurística. Consequentemente, P^* pode ser diferente de P'.

A complexidade da heurística PM é igual a complexidade da heurística piloto Dijkstra $\mathcal{O}(|V| \cdot \log |V|)$ [Cormen et al., 2009] executada por $|A|$ vezes, ou seja, uma vez para cada arco do grafo. Dessa forma, a complexidade no pior caso das heurísticas baseadas em Pilot Method é $\mathcal{O}(|A| \cdot |V| \cdot \log |V|)$, assumindo-se que $|V| \cdot \log |V| \geq |A|$.

4.3 Biased Random Key Genetic Algorithm

Uma extensão das heurísticas de Kasperski et al. [Kasperski & Zieliński, 2006b, 2007, 2008] proposta neste trabalho é denominada *Fixed Scenario Heuristic* (FSH). Ela recebe como entrada um grafo $G = (V, A)$, os limites inferior l_{ij} e superior u_{ij} para o custo de cada arco $(i, j) \in A$, além do vértice de origem $s \in V$ e de destino $t \in V$. Além dos dados do problema, FSH também recebe como parâmetro um possível cenário r',

Algoritmo 1: Pilot Method (PM)

Entrada: $G = (V, A)$, l_{ij}, u_{ij}, s, t
Saída: P^*, Melhor solução encontrada

```
1  início
2      P' ← s e P* ← ∅
3      enquanto P' não é um caminho entre s e t faça
4          Seja u o último vértice inserido em P'
5          para cada v ∈ δ+(w) faça
6              P_v ← Dijkstra (u, t, G, r^m)
7              P'_v ← Concatenar (P', P_v)
8              se CustoRobusto (P_v) < CustoRobusto(P*_v) ou P*_v = ∅ então
9                  P*_v ← P'_v, v* ← v
10         se CustoRobusto(P*_v) < CustoRobusto(P*) então
11             P* ← P*_v
12         Insere v* no fim de P'
13     retorna P*;
```

ou seja, uma fixação de valores para o custo $c'_{ij} \in [l_{ij}, u_{ij}]$ para cada arco $(i, j) \in A$. FSH inicialmente calcula o caminho mais curto P' entre s e t no cenário r', utilizando algoritmo de Dijkstra. Em seguida, o custo robusto relativo de P' é calculado e P' é retornado como solução da heurística. A complexidade assintótica, no pior caso, desta heurística é $O(|V| \cdot \log |V|)$, assumindo-se que $|V| \cdot \log |V| \geq |A|$. Para cada cenário r' passado como parâmetro para FSH, esta pode retornar uma solução diferente para RSP-IRR. Poranto, ao contrário das heurísticas anteriores que exploram o espaço de caminhos, a heurística FSH explora o espaço de cenários. Vale salientar que nem todas as soluções são induzidas por algum cenário.

Para se explorar o espaço de cenários de forma eficiente, é proposta uma heurística baseada em Algoritmos Genéticos com Chaves Aleatórias (BRKGA, do inglês *Biased Random-Key Genetic Algorithm*) [Gonçalves & Resende, 2010]. A motivação para aplicar esta metaheurística para RSP-IRR é a sua aplicação bem sucedida para vários outros problemas de otimização combinatória como problemas de trânsito [Buriol et al., 2010; Reis et al., 2011], problemas de sequenciamento [Gonçalves et al., 2009, 2010], Problema do Carregamento de Contêineres [Gonçalves & Resende, 2012], Problema de Roteamento e Atribuição do Comprimento de Onda em Redes [Noronha et al., 2011].

Em BRKGAs, os cromossomos consistem em vetores de números reais com valores no intervalo $[0, 1]$. Cada elemento do vetor é chamado de *chave* e seu valor é concebido aleatoriamente na população inicial. A adaptabilidade do cromossomo é definida pelo custo da solução fornecida por uma heurística de decodificação. Além dos dados da

instância, esta última também recebe como um de seus dados de entrada o vetor de chaves do cromossomo e devolve uma solução viável para o problema. O custo desta solução é usado como o valor de adaptabilidade do cromossomo.

Na heurística BRKGA-FSH, cada cromossomo da população representa um possível cenário. Ou seja, um cromossomo p contém uma chave $k_{ij}^p \in [0,1]$ para cada arco $(i,j) \in A$. Já o custo de cada arco $(i,j) \in A$ no cenário r^p, onde r^p é induzido pelo cromossomo p, é fixado em $c_{ij}^p = l_{ij} + (u_{ij} - l_{ij}) \cdot k_{ij}^p$. A adaptabilidade de p é definida como o custo robusto relativo do caminho retornado pela heurística FSH quando esta é executada a partir do cenário r^p. Sendo assim, a complexidade assintótica no pior caso de avaliar um cromossomo é $O(|V| \cdot \log |V|)$.

O algoritmo de recombinação ou cruzamento utilizado neste algoritmo genético foi proposto em [Spears & DeJong, 1991]. Dados dois cromossomos c_1 e c_2, um novo cromossomo c_{new} é gerado da seguinte forma. O valor de cada uma das chaves do novo cromossomo c_{new} é herdado aleatoriamente da chave correspondente em c_1 com probabilidade p_{rec} e da chave correspondente em c_2 com probabilidade $1 - p_{rec}$.

A Figura 4.1 apresenta um exemplo do cruzamento de dois cromossomos c_1 e c_2. As chaves herdadas pelo cromossomo c_{new} estão em destaque. Para definir o valor de cada chave do cromossomo c_{new} é gerado um número aleatório. Neste exemplo, são gerados cinco números aleatórios $\theta = [0,72; 0,18; 0,93; 0,48; 0,04]$. Quando $\theta_i \leq 0,5$, a chave i é herdada do cromossomo c_1, caso contrário, a chave i é herdada do cromossomo c_2.

C_1= [0,36 **0,42** 0,12 **0,78** **0,04**]
C_2= [**0,27** 0,81 **0,99** 0,14 0,17]
C_{new}= [0,27 0,42 0,99 0,78 0,04]

Figura 4.1. Cruzamento de dois cromossomos no algoritmo genético.

Este algoritmo genético não faz uso de operadores tradicionais de mutação. Ao invés disso, a cada geração novos cromossomos são gerados aleatoriamente, tal como na população inicial.

A Figura 4.2 mostra a evolução da população do algoritmo genético entre duas gerações. A cada nova geração a população corrente é ordenada e particionada em três conjuntos: *TOP*, *MID* e *BOT*. Os cromossomos mais aptos são mantidos no conjunto *TOP*, os menos aptos no conjunto *BOT* e os restantes no conjunto *MID*. Os cromossomos no conjunto *TOP* são copiados para a próxima geração e aqueles em *BOT* são substituídos por novos cromossomos gerados aleatoriamente. Os cromossomos no

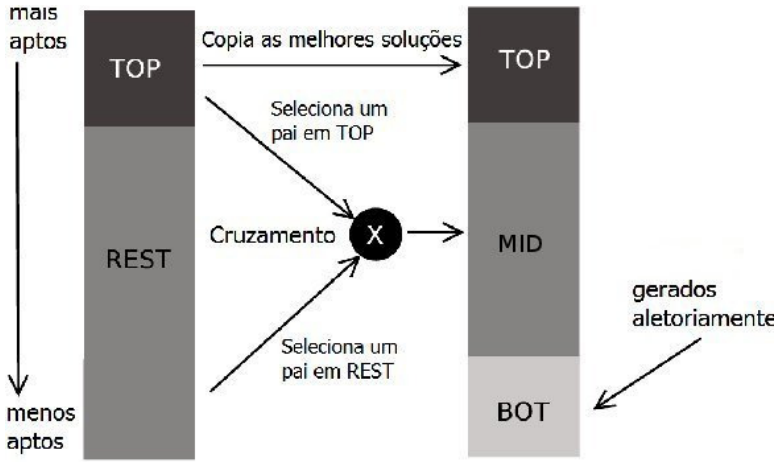

Figura 4.2. Transição entre gerações sucessivas no Algoritmo Genético com Chaves Aleatórias.

conjunto *MID* são substituídos por novos cromossomos obtidos pelo cruzamento de dois cromossomos, sendo que o primeiro é escolhido aleatoriamente do conjunto *TOP* e o segundo do conjunto $REST = MID \cup BOT$. O tamanho da população e a cardinalidade dos conjuntos *TOP*, *MID* e *BOT* são parâmetros que devem ser ajustados.

O BRKGA-FSH é descrito no Algoritmo 2. Na Linha 2, a população inicial é gerada utilizando-se vetores com chaves aleatórias. O laço das linhas 3-14 é executado enquanto a condição de parada não é satisfeita. Na Linha 4, ordena-se a população de acordo com o custo robusto relativo. Na Linha 5, divide-se a população em três grupos, de acordo com o custo robusto relativo, chamados de *TOP*, *MID* e *BOT*. No laço das linhas 6 e 7, copia-se o grupo *TOP* para a próxima geração. No laço das linhas 8-11, realiza-se o cruzamento, onde os cromossomos da porção *MID* da próxima geração são gerados. No laço das linhas 12-14 são gerados, de forma aleatória, os novos elementos da porção *BOT* da próxima geração. Por fim, na linha 15, retorna-se a solução com o melhor custo robusto relativo.

Algoritmo 2: Random Key Genetic Algorithm Fixed Scenario Heuristic (BRKGA-FSH).

Entrada: $G = (V, A)$, l_{ij}, u_{ij}, s, t
Saída: P^*, Melhor solução encontrada

```
1  início
2      Gerar uma população inicial de vetores de chaves aleatórias;
3      enquanto (Condição de Parada não satisfeita) faça
4          Ordenar a população em ordem crescente de custo robusto relativo;
5          Dividir a população nos grupos TOP, MID e BOT;
6          para cada elemento da População TOP faça
7              Copiar para a próxima geração;
8          para i de 1 até |MID| faça
9              Selecione aleatoriamente um cromossomo c^t de TOP e outro c^r de
               REST = MID ∪ BOT;
10             c^f ← Cruzamento (c^t, c^r)
11             Copiar c^f para a próxima geração
12         para i de 1 até |BOT| faça
13             Gerar um novo cromossomo c^a aleatoriamente
14             Copiar c^a para a próxima geração
15     retorna melhor solução de TOP
```

Capítulo 5

Experimentos Computacionais

Neste capítulo são descritos os experimentos computacionais que comparam as heurísticas desenvolvidas na seção anterior entre si e com as principais heurísticas encontradas na literatura [Kasperski & Zieliński, 2006b, 2007; Pérez et al., 2012]. As heurísticas M-Kasperski, U-Kasperski, SA-RSP, D-KSP, Y-KSP, PM e BRKGA-FSH foram implementadas em C++ e compiladas a versão 4.6.3 do Compilador Linux/GNU. Os parâmetros do BRKGA-FSH foram definidos de forma semelhante a [Goulart et al., 2011], onde (i) o tamanho da população p foi fixado em 100 cromossomos, (ii) os conjuntos $|TOP|$ e $|BOT|$ foram definidos como $|TOP| = 0,2 \times p$ e $|BOT| = 0,1 \times p$, (iii) o valor de p_{rec} foi fixado em $0,5$ e, (iv) a execução do BRKGA foi encerrada após $|V|$ gerações. A formulação proposta na Seção 3.2 dada pela função objetivo (3.13) e as restrições (3.8)-(3.12), (3.14) e (3.16)-(3.20) foi implementada e executada no ILOG CPLEX versão 12.4 com parâmetros *default*. A máquina de testes possui processador Intel Core i7 2.67 GHz e 4GB de memória RAM. Dois conjuntos de instâncias foram utilizados nos experimentos computacionais: grafos de Karasan [Karasan et al., 2001] e instâncias *grid*, propostos neste trabalho.

Grafos de Karasan foram propostos em [Karasan et al., 2001] e utilizados nos experimentos computacionais de [Montemanni et al., 2004; Montemanni & Gambardella, 2005a,b; Pérez et al., 2012]. Eles são grafos acíclicos [Bondy & Murty, 1976] e em camadas [Sugiyama et al., 1981]. Nos grafos de Karasan existem L camadas e cada camada possui W vértices. Em grafos de Karasan existe um arco de cada nó de uma camada l para todos os nós da camada $l+1$, para todo $l \in \{0, 1, ..., L-2\}$. Além disto, existe um arco do vértice s para cada vértice da camada 0 e um arco de cada vértice da camada $L-1$ para o vértice t. Na Figura 5.1 é dado um exemplo de um grafo de Karasan com 8 vértices e largura igual a 2.

Os grafos de Karasan que foram utilizados nos experimentos computacionais

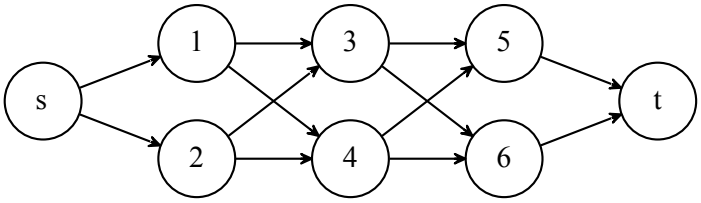

Figura 5.1. Grafo de Karasan com 8 vértices e largura igual a 2.

foram gerados tal como sugerido em [Karasan et al., 2001]. Os grafos de Karasan possuem 1002 vértices (incluindo s e t) e largura w, onde $w \in \{5, 10, 25, 50, 100\}$. O intervalo $[l_{ij}, u_{ij}]$ de custo dos arcos (i, j) foi gerado tal como descrito em [Karasan et al., 2001; Montemanni et al., 2004]. Para cada arco é gerado um número aleatório $c_{ij} \in [1, 200]$ e seu intervalo é definido pela seleção aleatória de l_{ij} em $U[(1-d)\cdot c_{ij}, (1+d)\cdot c_{ij}]$ e u_{ij} em $U[l_{ij}+1, (1+d)\cdot c_{ij}]$, onde $d = 0.9$ para todas as instâncias deste grupo. Por fim, vale salientar, que o nome de cada instância de Karasan é dado por K-1000-200-0.9-w.

Grafos *grid* são baseados em uma matriz $m \times n$, onde m é o número de linhas e n é o número de colunas. Cada posição da matriz representa um vértice do grafo e existem dois arcos bidirecionados entre cada par de vértices cujas respectivas células são adjacentes na matriz. O vértice s é definido como o vértice superior esquerdo e o vértice t é definido como o vértice inferior direito. Estas instâncias foram escolhidas, pois são semelhantes a intersecção de ruas em cidades. Na Figura 5.2, um exemplo de um *grid* de tamanho 3×5 é dado.

Nos experimentos deste trabalho, foram utilizados *grids* de tamanhos 6×60 (360 vértices e 1308 arcos), 7×70 (490 vértices e 1806 arcos), 8×80 (640 vértices e 2384 arcos), 9×90 (810 vértices e 3042 arcos) e 10×100 (1000 vértices e 3780 arcos). Os valores dos intervalos $[l_{ij}, u_{ij}]$, para todo $(i, j) \in A$ dos grafos *grid* foram gerados da mesma forma que os intervalos dos grafos de Karasan.

No experimento a seguir, a formulação proposta na Seção 3.2 foi implementada em C++ e executada no *solver* CPLEX[1] com o objetivo de obter limites inferiores e superiores para o RSP-IRR. O resultado do experimento é mostrado na Tabela 5.1. Vale a pena salientar que estes resultados são apresentados apenas para as instâncias de Karasan, porque a formulação proposta na Seção 3.2 só funciona se o grafo é acíclico,

[1] `http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/`

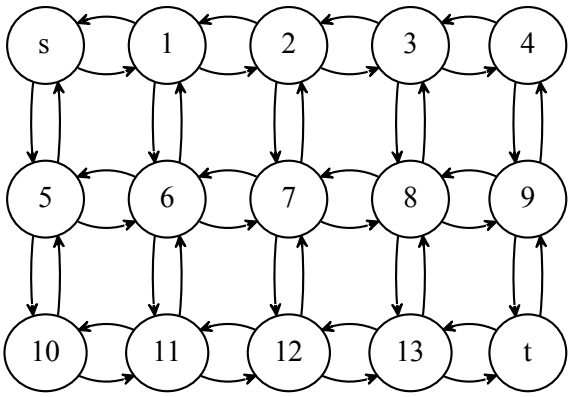

Figura 5.2. Exemplo de grafo grid 3×5.

o que não é o caso das instâncias *grid*. Para cada instância, o tempo de execução foi fixado em 2 horas (7200 segundos). A primeira coluna mostra o nome de cada instância. A segunda coluna mostra o valor do limite inferior (LB, do inglês *Lower Bound*) e a terceira exibe o valor do limite superior (UB, do inglês *Upper Bound*) obtido pelo CPLEX. A coluna 4, apresenta o número de nós da árvore de *Branch-and-Bound (BB)* avaliados. Por fim, a última coluna mostra o *gap* de cada instância, dado por $\frac{(LB-UB)}{LB}$. Pode-se observar que a linearização da formulação de [Averbakh, 2005] é bastante fraca.

Instância	LB	UB	Nós	GAP (%)
K-1000-200-0.9-a-5	-99,91%	70,37%	55.700	241,99
K-1000-200-0.9-b-5	-99,91%	83,43%	68.400	219,76
K-1000-200-0.9-a-10	-100,00%	100,97%	43.300	199,04
K-1000-200-0.9-b-10	-100,00%	79,30%	32.600	226,10
K-1000-200-0.9-a-25	-97,39%	68,67%	91.700	241,83
K-1000-200-0.9-b-25	-100,00%	98,65%	30.300	201,37
K-1000-200-0.9-a-50	-100,00%	122,45%	46.200	181,67
K-1000-200-0.9-b-50	-100,00%	261,36%	2.200	138,26
K-1000-200-0.9-a-100	-100,00%	505,00%	10.000	119,80
K-1000-200-0.9-b-100	-100,00%	275,00%	500	136,36
Média	-99,72%	166,52%	38.090	191,00

Tabela 5.1. Implementação da formulação proposta na Seção 3.2 no *solver* CPLEX para grafos de Karasan.

No experimento a seguir é realizada uma comparação entre as heurísticas M-Kasperski [Kasperski & Zieliński, 2006b] e U-Kasperski [Kasperski & Zieliński, 2007] com o objetivo de verificar qual das duas heurísticas obtêm melhores resultados para o

RSP-IRR. Os resultados dos experimentos são mostrados nas Tabelas 5.2 para grafos de Karasan e 5.3 para grafos *grid*. A primeira coluna mostra o nome de cada instância. A segunda e a terceira colunas mostram, respectivamente, o custo e o tempo de processamento da heurística M-Kasperski. Já a quarta e a quinta colunas mostram, respectivamente, o custo e o tempo de processamento da heurística U-Kasperski. A menor solução encontrada para cada instância está destacada em negrito. Para grafos de Karasan, pode-se observar que M-Kasperski obteve os melhores resultados em 9 das 10 instâncias testadas. Na média, o custo robusto relativo de M-Kasperski é 78,54% e o de U-Kasperski é de 86.42%. Já para grafos *grid*, cada heurística apresentou o melhor resultado em 5 das 10 instâncias, porém na média o custo robusto relativo de M-Kasperski foi 54.82%, enquanto o de U-Kasperski é 56.31%.

	M-Kasperski		U-Kasperski	
Instância	Custo	T(s)	Custo	T(s)
K-1000-200-0.9-a-5	**64,07%**	0,01	64,69%	0,00
K-1000-200-0.9-b-5	78,12%	0,01	**74,13%**	0,00
K-1000-200-0.9-a-10	**96,94%**	0,01	102,76%	0,01
K-1000-200-0.9-b-10	**82,56%**	0,01	83,72%	0,00
K-1000-200-0.9-a-25	**66,90%**	0,01	67,36%	0,02
K-1000-200-0.9-b-25	**78,57%**	0,01	86,99%	0,02
K-1000-200-0.9-a-50	**68,09%**	0,04	86,05%	0,04
K-1000-200-0.9-b-50	**65,91%**	0,03	81,82%	0,04
K-1000-200-0.9-a-100	**84,21%**	0,05	105,56%	0,05
K-1000-200-0.9-b-100	**100,00%**	0,05	111,11%	0,06
Média	78,54%	0,02	86,42%	0,02

Tabela 5.2. Comparação entre as heurísticas construtivas M-Kasperski e U-Kasperski para o RSP-IRR em grafos de Karasan.

	M-Kasperski		U-Kasperski	
Instância	Custo	T(s)	Custo	T(s)
G_6x60_a	**44,24%**	0,01	49,45%	0,01
G_6x60_b	56,08%	0,01	**54,09%**	0,01
G_7x70_a	65,76%	0,01	**61,00%**	0,01
G_7x70_b	**54,95%**	0,01	57,42%	0,01
G_8x80_a	53,69%	0,01	**49,46%**	0,01
G_8x80_b	56,26%	0,01	**54,31%**	0,01
G_9x90_a	**51,38%**	0,01	57,83%	0,01
G_9x90_b	57,07%	0,01	**56,04%**	0,01
G_10x100_a	**59,22%**	0,01	69,39%	0,01
G_10x100_b	**49,50%**	0,01	54,10%	0,01
Média	54,82%	0,01	56,31%	0,01

Tabela 5.3. Comparação entre as heurísticas construtivas M-Kasperski e U-Kasperski para o RSP-IRR em grafos *grid*.

No experimento a seguir é realizada uma comparação entre as heurísticas D-KSP e Y-KSP, além da comparação de Y-KSP para diferentes valores de k com o objetivo de verificar o valor de k no qual Y-KSP apresenta o melhor custo beneficio. O resultado

destes experimentos é apresentado nas Tabelas 5.4 para grafos de Karasan e 5.5 para grafos *grid*. Os valores k para Y-KSP foram fixados em 10, 100 e $|V|$, tanto para grafos de Karasan quanto para grafos *grid*. Cada heurística foi executada uma única vez, pois todas são determinísticas. A primeira coluna mostra o nome de cada instância. Já na segunda e terceira colunas são mostrados o custo e o tempo necessário para execução de D-KSP, respectivamente. Nas colunas 4 e 5 são aprestados o custo e o tempo de processamento (em segundos) de Y-KSP com $k = 10$, respectivamente. Nas colunas 6 e 7 são mostrados, respectivamente, o custo e o tempo de processamento (em segundos) de Y-KSP com $k = 100$. Por fim, nas duas últimas colunas, são apresentados o custo e o tempo de processamento (em segundos) de Y-KSP com $k = |V|$, respectivamente. A menor solução encontrada para cada instância está destacada em negrito. Pode-se observar que o tempo de processamento do KSP aumenta proporcionalmente com o valor de k. Também pode ser observado que a diferença entre D-KSP e Y-KSP com $k = 10$ é aproximadamente 5% para grafos de Karasan e 1% para grafos *grid*. Já em relação ao aumento do valor de k no Y-KSP, pode-se observar que o aumento do valor de k de 10 para 100 resulta em uma diminuição de aproximadamente 1%, em média, no custo robusto relativo, tanto para grafos de Karasan, quanto para grafos *grid*. Já quando o valor de k é alterado de 100 para $|V|$, o valor do custo robusto relativo médio diminui de 1% em ambos conjuntos de instâncias.

	D-KSP		Y-KSP					
			k = 10		k = 100		k = $\|V\|$	
Instância	Custo	T(s)	Custo	T(s)	Custo	T(s)	Custo	T(s)
K-1000-200-0.9-a-5	64,69%	0,04	63,63%	0,52	62,38%	4,44	**61,82%**	41,34
K-1000-200-0.9-b-5	74,13%	0,05	73,15%	0,54	71,79%	3,58	**70,84%**	30,12
K-1000-200-0.9-a-10	102,76%	0,15	100,44%	0,35	96,35%	2,29	**95,57%**	20,67
K-1000-200-0.9-b-10	83,72%	0,15	77,85%	0,38	77,36%	2,62	**77,23%**	23,52
K-1000-200-0.9-a-25	67,36%	0,98	67,12%	0,60	**65,07%**	5,30	**65,07%**	47,93
K-1000-200-0.9-b-25	86,99%	0,98	81,51%	0,51	79,45%	4,33	**77,93%**	43,50
K-1000-200-0.9-a-50	86,05%	3,79	**68,09%**	0,62	**68,09%**	4,70	**68,09%**	48,41
K-1000-200-0.9-b-50	81,82%	3,78	**65,91%**	0,62	**65,91%**	5,26	**65,91%**	54,92
K-1000-200-0.9-a-100	**84,21%**	15,57	**84,21%**	1,22	**84,21%**	9,96	**84,21%**	98,39
K-1000-200-0.9-b-100	95,00%	15,62	**90,00%**	1,14	**90,00%**	9,36	**90,00%**	89,56
Média	82,67%	4,11	77,19%	0,65	76,06%	5,18	75,67%	49,84

Tabela 5.4. Ajuste do valor de K para Y-KSP e comparação com D-KSP para grafos de Karasan.

O experimento a seguir apresenta uma comparação entre as heurísticas SA-RSP, proposta em [Pérez et al., 2012], BRKGA-FSH e PM. A tabela 5.6 apresenta os resultados para a grafos de Karasan enquanto a tabela 5.7 apresenta os resultados para grafos *grid*. O valor dos parâmetros da heurística SA-RSP utilizados neste experimento foram os mesmos de [Pérez et al., 2012]. A primeira coluna mostra o nome de cada instância. As colunas 2, 3 e 4 apresentam, respectivamente, o custo médio (após

	D-KSP		Y-KSP					
			k = 10		k = 100		k = $\|V\|$	
Instância	Custo	T(s)	Custo	T(s)	Custo	T(s)	Custo	T(s)
G_6x60_a	44,24%	0,01	44,08%	0,04	**43,39%**	0,35	**43,39%**	1,18
G_6x60_b	56,08%	0,01	55,06%	0,02	52,82%	0,26	**51,85%**	0,91
G_7x70_a	65,76%	0,01	61,83%	0,12	61,14%	0,93	**60,50%**	4,40
G_7x70_b	54,95%	0,01	54,60%	0,09	54,28%	0,84	**53,97%**	3,92
G_8x80_a	53,69%	0,01	53,01%	0,08	52,86%	0,68	**47,49%**	4,19
G_8x80_b	56,26%	0,01	54,71%	0,10	54,42%	0,70	**52,75%**	4,09
G_9x90_a	51,38%	0,01	51,22%	0,19	**50,90%**	1,70	**50,90%**	12,94
G_9x90_b	57,07%	0,01	57,07%	0,13	54,08%	1,06	**53,60%**	7,80
G_10x100_a	59,22%	0,01	59,16%	0,21	58,85%	1,47	**57,66%**	12,21
G_10x100_b	49,50%	0,01	49,26%	0,26	48,98%	2,20	**48,26%**	20,13
Média	54,82%	0,01	54,00%	0,12	53,17%	1,02	52,04%	7,18

Tabela 5.5. Ajuste do valor de Y-KSP e comparação com D-KSP para grafos *grid*.

20 execuções), o desvio padrão (em %) e o tempo de processamento (em segundos) de SA-RSP. Já as colunas 5, 6 e 7 mostram o custo médio (após 20 execuções), o desvio padrão (em %) e o tempo de processamento (em segundos) de BRKGA-FSH, respectivamente. Tanto em SA-RSP quanto em BRKGA-FSH o valor das sementes foi definido entre 1 e 20. Por fim, nas duas últimas colunas, são mostrados respectivamente o custo e o tempo de processamento (em segundos) de F-PM. Pode ser observado que BRKGA-FSH encontrou soluções melhores que as duas outras heurísticas para todas as instâncias, exceto K-1000-200-0.9-a-5, onde o melhor resultado foi encontrado por PM. Contudo, os tempos de processamento de BRKGA-FSH foram muito maiores que os tempos de processamento das outras heurísticas. Além disso, pode-se observar que F-PM encontrou resultados melhores que SA-RSP na média, com tempos de execução similares para grafos de Karasan e tempos de execução menores em grafos *grid*.

	SA-RSP [Pérez et al., 2012]			BRKGA-FSH			PM	
Instância	Custo	Dev(%)	T(s)	Custo	Dev(%)	T(s)	Custo	T(s)
K-1000-200-0.9-a-5	64,07%	0,00	2,14	61,49%	0,00	281,74	61,36%	2,80
K-1000-200-0.9-b-5	78,12%	0,00	2,12	67,58%	0,00	281,83	68,86%	2,83
K-1000-200-0.9-a-10	96,94%	0,00	2,73	93,38%	0,00	380,48	99,78%	3,50
K-1000-200-0.9-b-10	82,56%	0,00	2,98	76,74%	0,00	380,81	77,23%	3,46
K-1000-200-0.9-a-25	66,90%	0,00	6,18	65,07%	0,00	711,62	67,36%	5,64
K-1000-200-0.9-b-25	78,54%	0,00	5,37	77,93%	0,00	711,75	80,14%	5,64
K-1000-200-0.9-a-50	68,09%	0,00	8,39	68,09%	0,00	1.339,12	68,09%	8,19
K-1000-200-0.9-b-50	65,91%	0,00	8,33	65,91%	0,00	1.337,78	65,91%	8,17
K-1000-200-0.9-a-100	84,21%	0,00	11,21	84,21%	0,00	2.593,59	84,21%	13,15
K-1000-200-0.9-b-100	91,50%	0,02	11,22	90,00%	0,00	2.553,54	90,00%	12,98
Média	77,68%	0,00	6,07	75,04%	0,00	1.057,23	76,29%	6,64

Tabela 5.6. Comparação entre as heurísticas SA-RSP [Pérez et al., 2012], BRKGA-FSH e PM para grafos de Karasan.

Nas figuras (5.3)-(5.6), são mostrados os *TTT-Plots* para as instâncias K-1000-200-0.9-b-25, K-1000-200-0.9-b-100, G_6x60_b e G_7x70_b. Cada instância foi exe-

	SA-RSP [Pérez et al., 2012]			BRKGA-FSH			PM	
Instância	Custo	Dev(%)	T(s)	Custo	Dev(%)	T(s)	Custo	T(s)
G_6x60_a	49,45%	0,00	2,84	42,47%	0,00	20,53	43,03%	0,11
G_6x60_b	54,09%	0,00	2,90	50,05%	0,00	20,35	53,83%	0,10
G_7x70_a	61,00%	0,00	5,45	58,78%	0,00	37,67	59,68%	0,18
G_7x70_b	57,42%	0,00	5,31	53,51%	0,00	37,66	53,96%	0,16
G_8x80_a	49,46%	0,00	9,38	46,66%	0,00	69,24	48,02%	0,32
G_8x80_b	54,31%	0,00	9,26	52,75%	0,00	65,50	52,75%	0,26
G_9x90_a	57,83%	0,00	17,23	50,53%	0,00	111,81	50,77%	0,47
G_9x90_b	56,04%	0,00	15,32	51,86%	0,00	110,10	52,32%	0,45
G_10x100_a	69,39%	0,00	26,35	57,09%	0,00	182,42	57,30%	0,73
G_10x100_b	54,10%	0,00	29,74	47,44%	0,00	176,70	47,50%	0,66
Média	56,31%	0,00	12,38	51,11%	0,00	83,20	51,92%	0,34

Tabela 5.7. Comparação entre as heurísticas SA-RSP [Pérez et al., 2012], BRKGA-FSH e PM para grafos *grid*.

cutada 200 vezes para cada um dos algoritmos, variando-se a semente aleatória entre 1 e 200. Nas instâncias das figuras 5.3, 5.5 e 5.6 o tempo para o BRKGA-FSH atingir o alvo é sempre menor ou igual ao SA-RSP. Já na instância da figura 5.4 em 20% das execuções o SA-RSP foi mais rápido e nos outros 80% o BRKGA-FSH chegou no alvo em um menor tempo de processamento.

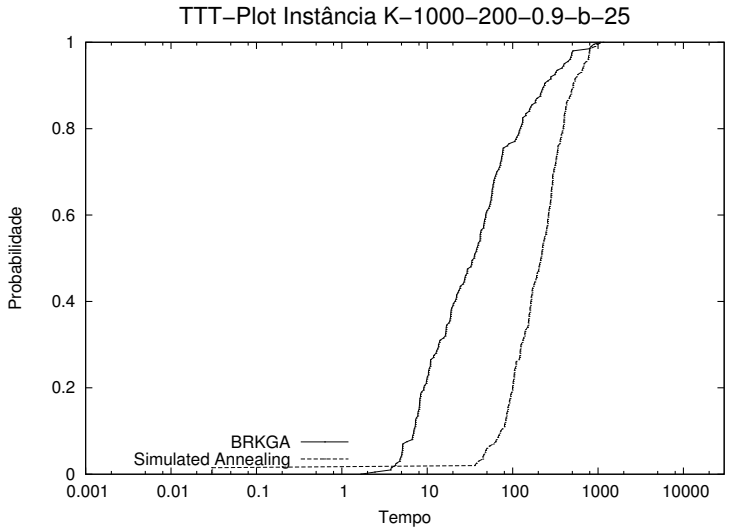

Figura 5.3. TTT-Plot ilustrando o desempenho das heurísticas BRKGA-FSH e SA-RSP relativo a instância K-1000-200-0.9-b-25. Alvo igual a: 78,5%

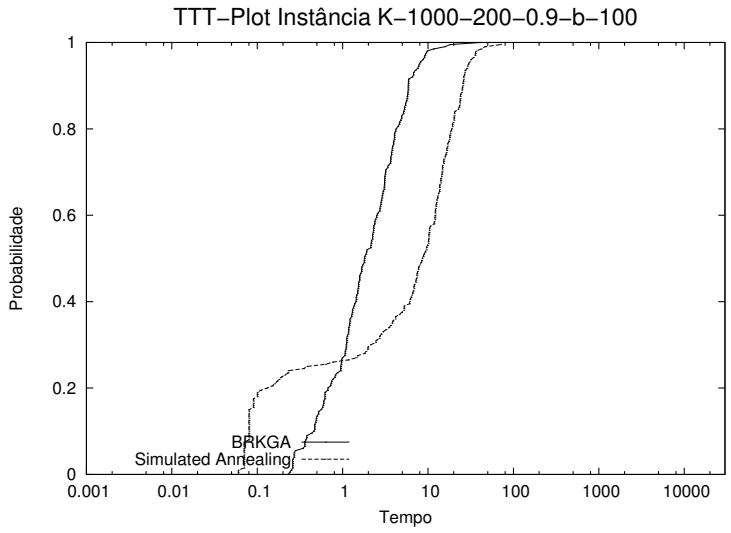

Figura 5.4. TTT-Plot ilustrando o desempenho das heurísticas BRKGA-FSH e SA-RSP relativo a instância K-1000-200-0.9-b-100. Alvo igual a 91,4%

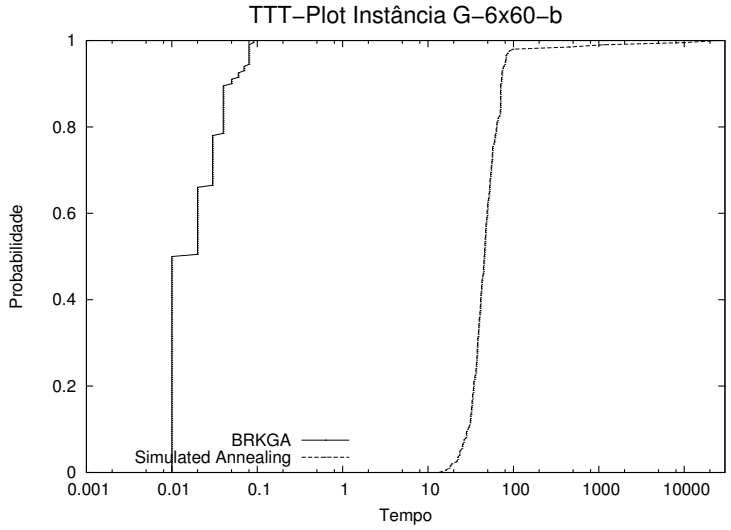

Figura 5.5. TTT-Plot ilustrando o desempenho das heurísticas BRKGA-FSH e SA-RSP relativo a instância G_6x60_b. Alvo igual a 54,0%

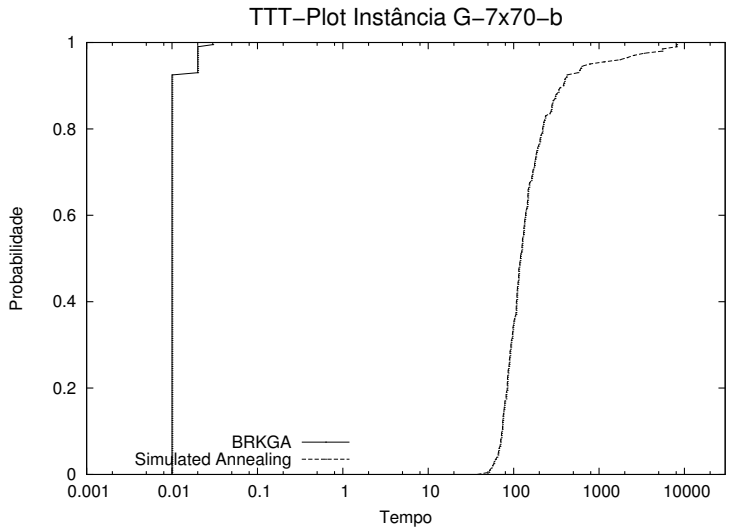

Figura 5.6. TTT-Plot ilustrando o desempenho das heurísticas BRKGA-FSH e SA-RSP relativo a instância G_7x70_b. Alvo igual a 57,4%

Capítulo 6

Conclusões e Trabalhos Futuros

Nesta dissertação, foi abordado o Problema de Caminho Mais Curto Robusto (RSP) com enfoque na versão intervalar com arrependimento relativo (RSP-IRR). Este problema consiste em encontrar o caminho com menor custo robusto relativo entre os vértices s e t em um grafo $G = (V, A)$. O problema foi proposto em [Kouvelis & Yu, 1997]. Averbakh [2005] propôs uma formulação não linear para o problema. Neste trabalho, foi proposta uma linearização da formulação de [Averbakh, 2005] e foram desenvolvidas diversas heurísticas construtivas baseadas em k caminhos mínimos (D-KSP e Y-KSP) [Yen, 1971] e *Pilot Method* [Duin & Voss, 1999], além de uma metaheurística baseada em algoritmos genéticos (BRKGA-FSH) [Bean, 1994].

Até onde se sabe, este é o primeiro trabalho a propor heurísticas dedicadas a solucionar o RSP-IRR e a utilizar um *benchmark* de instâncias *grid*. Para fins de comparação, as heurísticas de [Kasperski & Zieliński, 2007] e [Pérez et al., 2012] para RSP-IR foram adaptadas para o RSP-IRR. Dentre as heurísticas propostas neste trabalho, a heurística baseada em Pilot Method foi aquela que obteve os melhores resultados. Vale salientar, também, que esta mesma heurística obteve resultados melhores que as heurísticas da literatura [Kasperski & Zieliński, 2007; Pérez et al., 2012]. Já em relação as Metaheurísticas, BRKGA-FSH obteve resultados melhores que a heurística SA-RSP [Pérez et al., 2012].

Como trabalhos futuros, sugere-se melhorar a formulação proposta e propor novas formulações lineares inteiras, o estudo de algoritmos exatos e híbridos para esta versão do problema, a adaptação e aplicação das heurísticas propostas em outras versões do problema, além do estudo destas heurísticas para outros problemas de otimização robusta.

Referências Bibliográficas

Adjiashvili, D. & Zenklusen, R. (2011). An s-t connection problem with adaptability. *Discrete Applied Mathematics*, 159:695--705.

Ahuja, R. K.; Magnanti, T. L. & Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ.

Aissi, H.; Bazgan, C. & Vanderpooten, D. (2005). Approximation complexity of min-max (regret) versions of shortest path, spanning tree, and knapsack. Em *ESA'05 Proceedings of the 13th annual European conference on Algorithms*, pp. 862--873.

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

Aron, I. D. & Hentenryck, P. V. (2004). On the complexity of the robust spanning tree problem with interval data. *Operations Reserach Letters*, 32:36--40.

Averbakh, I. (2001). On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, 90:263--272.

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

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

Bean, J. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal On Computing*, 2:154--160.

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

Ben-Tal, A. & Nemirovski, A. (2002). Robust optimization – methodology and applications. *Mathematical Programming*, 92:453--480.

Bertsekas, D. P. & Castanon, D. A. (1999). Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5:89--108.

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

Bertsimas, D. & Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, 98:49--71.

Birge, J. R. & Louveaux, F. (1997). *Introduction to Stochastic Programming*. Springer.

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

Bondy, J. A. & Murty, U. S. R. (1976). *Graph theory with applications*. Elsevier Science Ltd.

Buriol, L. S.; Hirsch, M. J.; Querido, T.; Pardalos, P. M.; Resende, M. G. C. & Ritt, M. (2010). A biased random-key genetic algorithm for road congestion minimization. *Optimization Letters*, 4:619--633.

Büsing, C. (2009). The exact subgraph recoverable robust shortest path problem. Em Ahuja, R. K.; Möhring, R. H. & Zaroliagis, C. D., editores, *Robust and Online Large-Scale Optimization*, capítulo The Exact Subgraph Recoverable Robust Shortest Path Problem, pp. 231--248. Springer-Verlag.

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

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

Chen, X.; Hu, J. & Hu, X. (2009). A new model for path planning with interval data. *Computers & Operations Research*, 36:1893--1899.

Conde, E. (2004). An improved algorithm for selecting p items with uncertain returns according to the minimax-regret criterion. *Mathematical Programming*, 100:345--353.

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

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. (2009). *Introduction to Algorithms*. The MIT Press, 3rd edição. ISBN 978-0-262-03384-8.

Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton University Press.

Dhamdhere, K.; Goyal, V.; Ravi, R. & Singh, M. (2005). How to pay, come what may: Approximation algorithms for demand-robust covering problems. Em *In Symposium on Foundations of Computer Science*, pp. 367--378.

Dias, L. C. & Climaco, J. N. (2000). Shortest path problems with partial information: Models and algorithms for detecting dominance. *European Journal Of Operational Research*, 121:16--31.

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

Duin, C. & Voss, S. (1994). Steiner tree heuristics - A survey. Em Dyckhoff, H.; Derigs, U.; Salomon, M. & Tijms, H. C., editores, *Operations Research Proceedings*, pp. 485--496.

Duin, C. & Voss, S. (1999). The Pilot Method: A strategy for heuristic repetition with application to the Steiner problem in Graphs. *Networks*, 34:181--191.

Eley, M. (2002). Solving container loading problems by block arrangement. *European Journal of Operational Research*, 141:393--409.

Eppstein, D. (1992). Parallel recognition of series-parallel graphs. *Information and Computation*, 98:41--55.

Escoffier, B.; Monnot, J. & Spanjaard, O. (2008). Some tractable instances of interval data minmax regret problems: bounded disctance from triviality. Em *Proceedings of the 34th conference on Current trends in theory and practice of computer science*, pp. 280--291, Smokovec, Slovakia.

Fink, A. & Voss, S. (2003). Solving the continuous flow-shop scheduling problem by metaheuristics. *European Journal of Operational Research*, 151:400--414.

Gabrel, V.; Murat, C. & Wu, L. (2011). New models for the robust shortest path problem: complexity, resolution and generalization. *Annals of Operations Research*, pp. 1--24.

Gallo, G. & Pallottino, S. (1986). Shorthest path methods: A unifying approach. *Mathematical Progamming Study*, 26:38--64.

Golovin, D.; Goyal, V. & Ravi, R. (2006). Pay today for a rainy day: Improved approximation algorithms for demand-robust min-cut and shortest path. Em *Proceedings of the 23rd Annual conference on Theoretical Aspects of Computer Science*, pp. 206--217.

Gonçalves, J. F.; Mendes, J. J. M. & Resende, M. G. C. (2009). A random key based genetic algorithm for the resource constrained project scheduling problems. *Computers & Operations Research*, 36:92--109.

Gonçalves, J. F. & Resende, M. G. C. (2010). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17:487--525.

Gonçalves, J. F. & Resende, M. G. C. (2012). A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers & Operations Research*, 29:179--190.

Gonçalves, J. F.; Resende, M. G. C. & Mendes, J. J. M. (2010). A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. *Journal of Heuristics*, 5:467--486.

Goulart, N.; Dias, L. G. S.; Souza, S. R. & Noronha, T. F. (2011). Biased random-key genetic algorithm for fiber installation in optical network optimization. Em *roceedings of 2011 IEEE Congress on Evolutionary Computation*, pp. 2267–2271.

Gupta, S. K. & Rosenhead, J. (1968). Robustness in sequential investment decisions. *Management science*, 15:B18--B29.

Hasuike, T. (2011). On a robust cost-importance ratio shortest path problem. *International Journal of Innovative*, 2:93--98.

Höller, H.; Mélian, B. & Voss, S. (2008). Applying the pilot method to improve VNS and GRASP metaheuristics for the design of SDH/WDM networks. *European Journal of Operational Research*, 191:691--704.

Kalai, R.; Lamboray, C. & Vanderpooten, D. (2012). Lexicographic α-robustness: An alternative to min-max criteria C. *European Journal of Operational Research*, 220:722--728.

Karasan, O. E.; Yaman, H. & Ç. Pinar, M. (2001). The robust shortest path problem with interval data. Relatório técnico, Bilkent University, Department of Industrial Engineering.

Kasperski, A.; Kobylański, P.; Kulej, M. & Zieliński, P. (2005). *Minimizing maximal regret in discrete optimization problems with interval data*, pp. 193--208. Akademicka Oficyna Wydawnicza EXIT, Warszawa.

Kasperski, A. & Zieliński, P. (2006a). An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97:177--180.

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

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

Kasperski, A. & Zieliński, P. (2008). On the approximability of minmax (regret) network optimization problems. *Information Processing Letters*, 109:262--266. ISSN 0020-0190.

Kasperski, A. & Zieliński, P. (2010). Minmax regret approach and optimality evaluation in combinatorial optimization problems with interval and fuzzy weights. *European Journal of Operational Research*, 200:680--687.

Kasperski, A. & Zieliński, P. (2011). On the approximability of robust spanning tree problems. *Theoretical Computer Science*, 412:365--374.

Kirkpatrick, S.; Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671--680.

Kouvelis, P. & Yu, G. (1997). *Robust discrete optimization and its applications*. Kluver Academic Publishers.

Libura, M. (2009). On the robustness of optimal solutions for combinatorial optimization problems. *Control and Cybernetics*, 36:671--685.

Martins, E. Q. V. & Pascoal, M. M. B. (2003). A new implementation of Yen's ranking loopless paths algorithm. *4OR – Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1:121--133.

Martins, P. (2007). Enhanced second order algorithm applied to the capacitated minimum spanning tree problem. *Computers & Operations Research*, 34:2495--2513.

Meloni, C.; Pacciarelli, D. & Pranzo, M. (2004). A rollout metaheuristic for job shop scheduling problems. *Annals of Operations Research*, 131:215--235.

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

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

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

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

Noronha, T. F.; Resende, M. G. C. & Ribeiro, C. C. (2011). A biased random-key genetic algorithm for routing and wavelength assignment. *Journal of Global Optimization*, 50:503--518.

Orgyczak, W. (1997). On the lexicographic minimax approach to location problems. *European Journal of Operational Research*, 100:566--585.

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

Puhl, C. (2009). Recoverable robust shortest path problems. Em *Special Issue of the International Network Optimization Conference*, volume 59, pp. 181--189.

Reis, R.; Ritt, M.; Buriol, L. S. & Resende, M. G. C. (2011). A biased random-key genetic algorithm for OSPF and DEFT routing to minimize network congestion. *International Transactions in Operation Reserach*, 18:401--423.

Rosenhead, M. J.; Elton, M. & Gupta, S. K. (1972). Robustness and optimality as criteria for strategic decisions. *Operational Research Quarterly*, 23:413--430.

Roy, B. (2010). Robustness in operational research and decision aiding: A multi-faceted issue. *European Journal of Operational Research*, 200:629--638.

Salazar-Neumann, M. (2010). The robust shortest path and the single-source shortest path problems : Interval data. *Annales du LAMSADE , Paris*, 7:237--251.

Seshadri, R. & Srinivasan, K. K. (2010). Algorithm for determining most reliable travel time path on network with normally distributed and correlated link travel times. *Transportation Research Record*, 2196:83--92.

Shimbel, A. (1955). Structure in communication nets. Em *Proceedings of the symposium on information Networks, New York.*, pp. 199--203.

Spall, J. C. (2003). *Introduction to stochastic search and optimization*. Wiley.

Spears, W. & DeJong, K. (1991). On the virtues of parameterized uniform crossover. Em Belew, R. & Booker, L., editores, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 230--236, San Mateo, Italy.

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

Voss, S.; Fink, A. & Duin, C. (2005). Looking ahead with the pilot method. *Annals of Operations Research*, 136:285--302.

Wu, X. & Nie, Y. (2010). Implementation issues for the reliable a priori shortest path problem. *Transportation Research Record*, 2081:51--60.

Xiong, Y.; Golden, B. & Wasil, E. (2006). Improved heuristics for the minimum label spanning tree problem. *Ieee Transactions on Evolutionary Computation*, 10:700--703.

Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *Management Science*, 17:712--716.

Yu, G. & Yang, J. (1997). On the robust shortest path problem. *Computers & Operations Research*, 25:457--468.

Zieliński, P. (2004). The computational complexity of the relative robust shortest path problem with interval data. *European Journal of Operational Research*, 158:570--576.