RESULTADOS SOBRE O PROBLEMA DA FLORESTA RESTRITA DE PESO MÍNIMO

Sávio Soares Dias

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Luidi Gelabert Simonetti

Rio de Janeiro
Fevereiro de 2024

RESULTADOS SOBRE O PROBLEMA DA FLORESTA RESTRITA DE PESO MÍNIMO

Sávio Soares Dias

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Luidi Gelabert Simonetti

Aprovada por: Prof. Luidi Gelabert Simonetti
Prof. Nelson Maculan Filho
Prof. Laura Silvia Bahiense da Silva Leite
Prof. Yuri Abitbol de Menezes Frota
Prof. Rafael Augusto de Melo

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2024

Soares Dias, Sávio

Resultados sobre o Problema da Floresta Restrita de Peso Mínimo/Sávio Soares Dias. – Rio de Janeiro: UFRJ/COPPE, 2024.

X, 71 p.: il.; 29, 7cm.

Orientador: Luidi Gelabert Simonetti

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2024.

Referências Bibliográficas: p. 67 – 71.

1. Floresta Geradora Mínima. 2. Floresta Geradora Mínima Restrita. 3. Programação Inteira. 4. Heurísticas. I. Gelabert Simonetti, Luidi. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A todos que não desistiram de mim, mesmo quando eu havia desistido.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

RESULTADOS SOBRE O PROBLEMA DA FLORESTA RESTRITA DE PESO MÍNIMO

Sávio Soares Dias

Fevereiro/2024

Orientador: Luidi Gelabert Simonetti

Programa: Engenharia de Sistemas e Computação

Apresentam-se, nesta tese, abordagens e resultados teóricos sobre o Problema da Floresta Restrita de Custo Mínimo (PFR). No contexto do PFR, o principal objetivo é identificar uma floresta geradora do grafo de entrada, tal que a soma dos pesos de suas arestas seja mínimo, e cada componente conexa contenha pelo menos k vértices.

Na parte teórica do nosso trabalho realizamos um estudo de dominância com modelos existentes na literatura, bem como apresentamos um teste de redução para instâncias do problema. No escopo prático, propomos três novas formulações de programação linear inteira, onde uma delas, com modelagem direcionada, foi capaz de superar o atual estado-da-arte do problema. Além disso, propomos heurísticas para resolver mais eficientemente o problema de separação de desigualdades exponenciais presentes na literatura e nos modelos propostos.

Ademais, apresentamos quatro novas formulações baseadas em decomposições, sendo duas decomposições lagrangianas e duas decomposições do tipo Dantzig-Wolfe. As decomposições lagrangianas demonstraram potencial quando utilizadas como pré-processamento para nosso algoritmo de *Branch-and-Cut*. As reformulações do tipo Dantzig-Wolfe ainda apresentam subproblemas caracterizados como $\mathcal{NP}$-Díficil, e portanto sua motivação se limita à busca de melhores limites duais. Finalmente, apresentamos duas heurísticas primais baseadas nas metaheurísticas de Colônia de Formigas e *Greedy Randomized Adaptive Search Procedure*.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

RESULTS ON MINIMUM-WEIGHT CONSTRAINED FOREST PROBLEM

Sávio Soares Dias

February/2024

Advisor: Luidi Gelabert Simonetti

Department: Systems Engineering and Computer Science

This thesis presents approaches and theoretical results for the Constrained Forest Problem (CFP). In the CFP context, the main objective is to identify a spanning forest of the input graph, such that the cumulative weight of the selected edges is minimal, and that each connected component spans at least k vertices.

In the theoretical part of our work, we conducted a dominance study with existing models in the literature, as well as presented a reduction test for problem instances. In the practical scope, we proposed three new integer linear programming formulations, one of them uses arc variables and was able to outperform the current state-of-the-art for the problem.

In addition, we proposed efficient heuristics to solve the separation problem for exponential inequalities, which are present in the literature and in our proposed models. Also, we propose four new formulations that are decomposition-based, two of them are Lagrangian decompositions and the other two are Dantzig-Wolfe decompositions. The Lagrangian decompositions demonstrated potential when used as preprocessing for our Branch-and-Cut algorithm. The Dantzig-Wolfe reformulations still had $\mathcal{NP}$-hard subproblems, and therefore their motivation is limited to the search for better dual bounds. Finally, we present two primal heuristics based on the Ant Colony and Greedy Randomized Adaptive Search Procedure metaheuristics.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O estudo do Problema da Floresta Geradora Restrita de Custo Mínimo (PFR) se justifica por sua relevância em diversos campos de aplicação, como alocação de centros de comunicação e atribuição de redes [27]; grande variedade de problemas de transporte e logística [5]; problemas de microagregação de dados estatísticos [34, 51]; alocação de distritos políticos [41, 57]; e otimização de conexão de sensores para película tátil em robôs [3, 4]. Desta forma, a importância do PFR reside em sua capacidade de otimizar a infraestrutura de redes, minimizando custos e maximizando a eficiência.

Para definição do PFR, sejam $G = (V, E)$ um grafo não-direcionado, $w \in \mathbb{R}_+^{|E|}$ um vetor de pesos associado às arestas de G, e um inteiro positivo $2 \leq k \leq n$, onde $n = |V|$. O Problema da Floresta Geradora Restrita de Custo Mínimo tem como objetivo encontrar uma floresta geradora $F = (V, E_F)$, $E_F \subseteq E$, tal que cada componente conexa é uma árvore geradora contendo pelo menos k vértices, e o peso cumulativo de suas arestas seja mínimo. Na literatura, F pode também ser encontrado sob as nomenclaturas de floresta k-capacitada ou k-floresta [11, 15, 36]. O PFR é $\mathcal{NP}$-difícil para $3 \leq k \leq \lfloor \frac{n}{2} \rfloor$ [8, 29], o caso com $k = 2$ é reduzido ao problema de cobertura de arestas de peso mínimo, e se $k > \lceil \frac{n}{2} \rceil$, o problema é reduzido à Árvore Geradora Mínima (AGM).

A Figura 1.1 apresenta um exemplo de grafo de entrada para o PFR com 10 vértices (Figura 1.1.A.) e a solução ótima para $k = 3$ (Figura 1.1.B.) com custo igual a 13 e $k = 4$ (Figura 1.1.C.) com custo igual a 15.

Esta tese visa contribuir para a área de otimização combinatória, desenvolvendo algoritmos eficientes para o PFR. Os algoritmos propostos são avaliados em termos de tempo de execução, qualidade da solução e capacidade de lidar com diferentes restrições. Tais avaliações são feitas levando em consideração o atual estado-da-arte para o problema e os resultados desta pesquisa podem ser utilizados para otimizar a infraestrutura de redes em diversos campos de aplicação, gerando economia de recursos e aumento da eficiência.

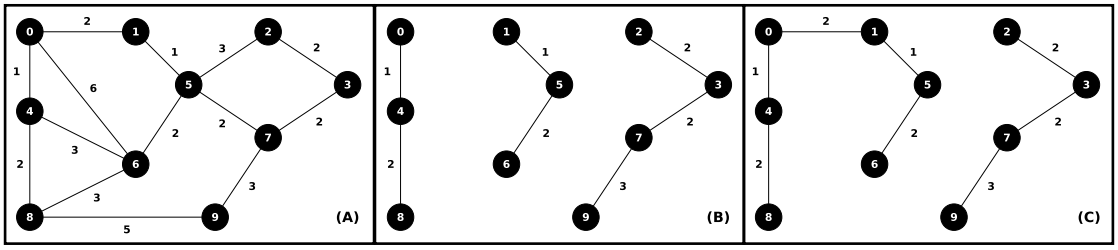

Figura 1.1: Solução para instância com $n = 10$ (A), $k = 3$ (B) e $k = 4$ (C)

O objetivo principal desta tese é desenvolver um conjunto de abordagens eficientes para resolver o PFR, bem como explorar o problema pelas diferentes perspectivas da otimização combinatória. Para isso, exploramos algoritmos exatos, heurísticos, e decomposições, de modo a redefinir o estado-da-arte do problema. Além disso, abordamos o problema em um escopo teórico ao estudar a dominância de dois modelos pré-existentes na literatura, de maneira a identificar o atual estado-da-arte.

Cada um dos demais capítulos dessa tese são dedicados a um tópico específico, com o objetivo de apresentar o tema de forma clara, concisa e organizada. A sequência dos capítulos segue uma lógica que facilita a compreensão do problema e do desenvolvimento da pesquisa. Sendo eles:

- **Estudo de Dominância e Teste de Redução:** Explora técnicas para reduzir o espaço de busca do problema, como dominância de modelos pré-existentes e testes de redução, a fim de aumentar a eficiência dos algoritmos.
- **Formulações:** Propõe e aborda diferentes formulações matemáticas para o PFR, considerando diferentes tipos de restrições e domínios de variáveis.
- **Heurísticas Primais:** Propõe e analisa heurísticas primais para gerar soluções viáveis de boa qualidade para o problema.
- **Decomposições:** Investiga técnicas de decomposição para abordar problemas de grande escala, dividindo-os em subproblemas menores e mais fáceis de resolver.
- **Resultados e Discussões:** Apresenta os resultados computacionais obtidos com os algoritmos desenvolvidos, comparando-os com outros métodos existentes na literatura.
- **Considerações Finais:** Resume as principais contribuições da tese, destaca as limitações do trabalho e sugere direções para pesquisas futuras.

Nas demais seções deste capítulo apresentamos uma revisão da literatura sobre o PFR e discutiremos alguns trabalhos relacionados.

1.1 Revisão da Literatura

A literatura apresenta diversos algoritmos para resolver o PFR. A maioria dos trabalhos focam em heurísticas aproximativas, como uma forma de baixo custo computacional. A primeira referência ao PFR na literatura, onde o problema foi apresentado, propõe uma heurística gulosa 2-aproximativa que constrói uma solução adicionando arestas de custo mínimo de uma AGM do grafo [29]. Essa heurística foi posteriormente melhorada para um fator $2 - \frac{1}{n}$ e estendida para uma classe de problemas de partição em grafos de custo mínimo [24]. LASZLO e MUKHERJEE [33, 35] apresentaram uma classe de heurísticas gulosas baseadas nos algoritmos de Prim e Kruskal para a AGM. Uma dessas heurísticas, chamada *Heaviest Edge First* (HEF), também possui um fator de aproximação igual a 2 e constrói uma solução removendo as arestas mais pesadas de um grafo enquanto mantém a viabilidade da floresta. Uma heurística gulosa $\frac{3}{2}$-aproximativa foi apresentada em COUËTOUX [13], e posteriormente estendida para uma classe de subproblemas que envolvem encontrar florestas de custo mínimo em COUËTOUX *et al.* [14]. Embora algumas dessas heurísticas possuam melhores fatores de aproximação, o HEF atualmente tem o melhor desempenho experimental. Para um estudo completo sobre a complexidade e aproximação do PFR, nós indicamos a leitura de BAZGAN *et al.* [8].

Após um levantamento bibliográfico com perspectiva em metaheurísticas, apenas dois algoritmos genéticos (AG) foram encontrados. O primeiro deles, apresentado em LASZLO e MUKHERJEE [36], usa AGMs de boa qualidade como cromossomos. Seu operador de mutação troca arestas que conectam as mesmas componentes conexas, enquanto o operador de cruzamento combina duas AGMs em duas novas. Por fim, uma heurística simples transforma as árvores em k-florestas. O segundo AG, apresentado em QUEERN [47], utiliza k-florestas como cromossomos, e seus operadores sempre mantém a viabilidade da solução. Vale notar que, uma vez que não existe conjunto de instâncias estabelecidos na literatura para o PFR, a comparação entre algoritmos não-determinísticos em diferentes ambientes é problemática, e portanto não existem tais experimentos na literatura. Apesar disso, os autores reportaram que os resultados obtidos foram significativamente melhores do que os obtidos pela HEF, onde os AGs foram capazes de obter o ótimo em instâncias consideradas pequenas.

A primeira formulação não-direcionada para o PFR foi apresentada em GOEMANS e WILLIAMSON [24], e utilizada somente como meio de definição formal para o problema. Tal formulação utiliza desigualdades baseadas nas clássicas *cut-sets*, mas com uma restrição no tamanho do conjunto de vértices S. Por esse motivo, não se conhece algoritmo polinomial para resolver o problema de separação dessas desigualdades. Outra formulação não-direcionada foi apresentada em JI [30]. Tal

formulação utiliza o mesmo domínio de variáveis, mas contempla dois conjuntos de desigualdades exponenciais diferentes. O primeiro conjunto é o das clássicas *Subcycle Elimination Constraints* (SECs), e o outro, chamado de *Flower Constraints*, também não possui algoritmo polinomial conhecido para sua separação. Os autores propuseram um algoritmo *Branch-and-Cut* (B&C) utilizando de duas heurísticas para resolução do problema de separação. Finalmente, BONATES *et al.* [11] propuseram um algoritmo *Branch-and-Bound* utilizando uma formulação baseada em fluxo com variáveis inteiras mistas. Os autores limitaram a entrada do algoritmo à AGM do grafo original, argumentando tal ação devido às simetrias inerentes e o alto número de variáveis e desigualdades explícitas da formulação. Portanto, por motivos computacionais, o algoritmo proposto não garante a solução ótima.

1.2 Problemas Relacionados

O PFR é intimamente relacionado a uma série de problemas de árvore e partição de grafos através da literatura. ALI e HUANG [2] estudaram uma variação onde o número de árvores é um parâmetro de entrada, e as árvores precisam ser balanceadas, i.e., a diferença entre o número de vértices entre as árvores não pode ser maior que um. Neste trabalho foi apresentado uma formulação inteira, resolvida através de relaxação lagrangiana e heurísticas primais. Foram utilizadas heurísticas *dual ascent* e heurísticas primais para obtenção de limites para o método.

Em GUTTMANN-BECK e HASSIN [27] foi apresentado outro problema similar, mas nesse caso, o número de árvores e vértices por árvore são dados. O método de resolução proposto foi um algoritmo aproximativo de ordem $2p-1$, com complexidade $O(p^2 4^p + n^2)$, sendo p o número de árvores.

Uma versão min-max do problema existe na literatura [57], onde não há limitação da quantidade de vértices por árvore, e o objetivo é minimizar a árvore de maior custo. O número de árvores é fixado atráves de um conjunto de vértices-raiz dado. Um *Branch-and-Bound* combinatório foi proposto por YAMADA *et al.* [57], junto de três limites inferiores que foram utilizados no método. Entretanto, o método só foi capaz de resolver otimamente problemas com até 2 raízes. Em 2015, DA CUNHA *et al.* [16] apresentou um algoritmo *Branch-and-Cut* que utiliza de heurísticas duais *multi-start* e novos cortes de otimalidade. Os resultados superaram as abordagens anteriores em tempo computacional, tamanho da árvore de enumeração e em quantidade de ótimos provados, sendo capaz de encontrar o ótimo em instâncias com 3 e 4 raízes.

O PFR aparece como uma formulação possível para o problema de posicionamento de sensores em película tátil de robôs, onde o custo das arestas é dinâmico. Nesse problema, o peso de cada aresta é dependente da solução, e só então é possível

calcular os custos de roteamento e atribuição entre os sensores presentes na película e os micro-controladores. O número de árvores é limitado pela quantidade de micro-controladores, portanto é um valor constante. A otimização atende três objetivos distintos: minimizar vértices isolados; balancear a carga em cada micro-controlador; e ampliar o alcance dos sensores na película. Esse problema foi abordado com uma heurística *Multi-Start* e outra baseada em Colônia de Formigas por ANGHINOLFI *et al.* [3]. Também foi apresentado uma formulação inteira mista para definição matemática do problema. Um estudo mais amplo nas técnicas de atualização dos feromônios dessa heurística foi feito em ANGHINOLFI *et al.* [4], apresentando cinco técnicas e testando-as com estudos de caso reais e teóricos.

Existem também versões *online* ou periódicas do PFR, com vetores de pesos diferentes para cada horizonte de planejamento [46], onde em cada período uma k-floresta precisa ser encontrada. Um dos problemas de agendamento de sub-árvores ótimas apresentado em ADASME [1] também se assemelha bastante ao PFR. Neste problema, o objetivo é encontrar uma única floresta geradora mínima, com cada árvore cobrindo exatamente $k = \frac{n}{|P|}$ vértices por período, onde $|P|$ é o número de períodos.

Finalmente, outro problema bastante similar ao PFR é o Problema da Árvore Geradora Mínima Capacitada [23]. Neste problema, o objetivo é encontrar uma árvore geradora mínima, tal que a capacidade de cada sub-árvore ligada à raiz não seja superior a Q (limitando superiormente a capacidade de cada sub-árvore), onde a demanda é associada a cada vértice e pode ser não-unitária. Desta forma, inspirado em tal problema, podemos reformular o PFR como um problema de árvore geradora mínima onde a capacidade seja um limite inferior e cada vértice possua demanda unitária. Para isso, basta adicionar um vértice artificial como raiz e ligá-lo a cada árvore.

Capítulo 2

Estudo de Dominância e Teste de Redução

Este capítulo aborda dois modelos matemáticos existentes na literatura no âmbito do estudo de dominância entre as duas formulações. O objetivo é demonstrar que um desses modelos é capaz de prover limites duais mais fortes, e portanto caracterizar o estado-da-arte em algoritmos exatos para o problema. Além disso, apresentamos um teste de redução para reduzir o espaço de busca do problema sem que a solução ótima seja eliminada.

Para as discussões apresentadas neste capítulo, considere o modelo de programação inteira apresentado por GOEMANS e WILLIAMSON [24] como definido em $\mathcal{P}_1$ e o modelo apresentado por JI [30] como definido em $\mathcal{P}_2$. Ambos os modelos utilizam o mesmo conjunto de variáveis de decisão $x \in \mathbb{B}^{|E|}$ para escolha de arestas do grafo de entrada. Logo, para o restante desse trabalho, sejam $S \subseteq V$ um subconjunto de vértices, $\delta(S) \subset E$ um subconjunto de arestas contendo exatamente um vértice em S (arestas de corte), $E(S) \subseteq E$ um subconjunto de arestas com ambos os vértices em S, e $x(E(\cdot)) = \sum_{e \in E(\cdot)} x_e$ o somatório das variáveis implicadas pelo subconjunto de arestas $E(\cdot) \subseteq E$.

$$(\mathcal{P}_1) \min \qquad w^T x \tag{2.1}$$

$$\text{s.t.:} \quad x(\delta(S)) \geq 1, \quad \forall S \subset V, 1 \leq |S| < k \tag{2.2}$$

O modelo de GOEMANS e WILLIAMSON [24] é uma representação exata do problema, onde a função objetivo (2.1) tem por finalidade minimizar o peso das arestas selecionadas. O conjunto exponencial de desigualdades (2.2) garante que cada árvore na solução tenha pelo menos k vértices, descartando componentes conexas de tamanho menor. Uma vez que $w > 0$, e a função objetivo será minimizada, as

soluções contendo ciclos ainda são válidas, mas não serão ótimas.

$$(\mathcal{P}_2)\ \min \quad w^T x \tag{2.3}$$

$$\text{s.t.:} \quad x(E(S)) \leq |S| - 1, \qquad \forall S \subseteq V \tag{2.4}$$

$$x(E(S)) + x(\delta(S)) \geq |S| - \left\lfloor \frac{|S|}{k} \right\rfloor, \quad \forall S \subseteq V \tag{2.5}$$

No modelo de JI [30], a função objetivo é idêntica à de GOEMANS e WILLIAMSON [24], enquanto os conjuntos de desigualdades são diferentes. As restrições de eliminação de subciclo foram utilizadas para garantir um grafo acíclico, ao invés das desigualdades estilo *cutset* de $\mathcal{P}_1$. Além disso, as desigualdades (2.5) impõem um limite inferior no número de arestas que uma k-floresta com $|S|$ vértices precisa ter.

$$x(E(V)) \leq |V| - 1 \tag{2.6}$$

$$x(\delta(v)) \geq 1, \qquad \forall v \in V \tag{2.7}$$

$$x(E(V)) \geq |V| - \left\lfloor \frac{|V|}{k} \right\rfloor \tag{2.8}$$

Finalmente, as desigualdades (2.6)-(2.8) também foram apresentadas por JI [30], e são casos específicos das desigualdades anteriores. O único próposito dessas desigualdades é servir como um limite inicial do algoritmo de planos de cortes.

Devido às similaridades entre os dois modelos, nós podemos obter uma relação de dominância entre eles. Uma vez que eles compartilham o mesmo espaço de variáveis, as suas restrições são automaticamente válidas em ambos os modelos. Logo, como as SECs estão implícitas em $\mathcal{P}_1$, o Lema 2.1 demonstra a dominância das desigualdades (2.2) pelas desigualdades em $\mathcal{P}_2$. Um conjunto de desigualdades domina outro se ele for mais restritivo, i.e., permite menos soluções fracionárias. Logo, no Teorema 2.2, concluímos que $\mathcal{P}_1$ contém $\mathcal{P}_2$.

Lema 2.1. *As desigualdades (2.4) e (2.5) dominam (2.2).*

Demonstração. Seja S um subconjunto de vértices com cardinalidade menor que k. Isso faria o lado direito das desigualdades (2.5) igual a $|S|$, já que $\left\lfloor \frac{|S|}{k} \right\rfloor$ é igual a zero. Se somarmos as desigualdades (2.4) e (2.5), para esse mesmo conjunto S, nós obtemos:

$$x(E(S)) + x(\delta(S)) - x(E(S)) \geq |S| - |S| + 1.$$

Depois de simplificar, obtemos $x(\delta(S)) \geq 1$, onde $|S| < k$, o que implica nas desi-

gualdades (2.2). Logo, para $|S| \geq k$, as desigualdades (2.4) e (2.5) eliminam soluções fracionárias do poliedro real e, portanto, dominam as desigualdades (2.2). □

Teorema 2.2. *O poliedro formado pela definição do modelo de JI [30] está contido no poliedro formado pelo modelo de GOEMANS e WILLIAMSON [24], i.e.* $\mathcal{P}_2 \subset \mathcal{P}_1$.

Demonstração. Primeiramente, note que as desigualdades (2.6) e (2.8) são casos específicos das desigualdades (2.4) e (2.5), onde $S = V$, portanto somente são listadas explicitamente no poliedro de modo a prover um limite inicial melhor no escopo do algoritmo de planos de cortes. Note também que (2.7) são casos específicos das desigualdades (2.2) e (2.5), onde $S = \{v\}, \forall v \in V$.
Uma vez que ambos os poliedros utilizam o mesmo domínio de variáveis $x \in \mathbb{B}^{|E|}$ e usando do Lema 2.1, temos que qualquer solução viável $\bar{x} \in \mathcal{P}_2$ também é viável para $\mathcal{P}_1$.
Logo, sem perda de generalidade, podemos construir uma solução $\bar{x}$, tal que:

- $\bar{x}(\delta(S)) \geq 1$, $\forall S \subset V$, $1 \leq |S| < k$;
- $\exists \bar{S} \subset V$, $k \leq |\bar{S}| < 2k$ onde $\bar{x}(\delta(\bar{S})) = 0$; e
- $\forall v \in \bar{S}$, $\bar{x}(\delta(v)) = 1$, e $\exists e \in \delta(v)$ tal que $\bar{x}(e) < 1$.

Note que $x(E(S)) = \frac{1}{2}\left[\sum_{v \in S} x(\delta(v)) - x(\delta(S))\right]$, portanto, temos que $\bar{x}(E(\bar{S})) = \frac{1}{2}\sum_{v \in \bar{S}} \bar{x}(\delta(v)) = \frac{|\bar{S}|}{2}$. Logo, a desigualdade (2.5) pode ser escrita para $\bar{S}$ e $\bar{x}$, resultando em:

$$\frac{|\bar{S}|}{2} + 0 \geq k - \left\lfloor \frac{k}{k} \right\rfloor \Rightarrow \frac{|\bar{S}|}{2} \geq k - 1.$$

Tal desigualdade está violada, uma vez que por definição, $k \leq |\bar{S}| < 2k$ e $\frac{|\bar{S}|}{2} \not\geq k - 1, \forall k > 2$. Portanto a solução $\bar{x}$ desenhada é viável em $\mathcal{P}_1$, mas inviável em $\mathcal{P}_2$. Concluindo, $\mathcal{P}_2 \subset \mathcal{P}_1$. □

Na Figura 2.1 temos um exemplo com $k = 3$ de solução fracionária com custo 14 que não viola nenhuma desigualdade (2.2), mas viola a desigualdade (2.5) com $S = \{4, 6, 8\}$.

A partir desse estudo de dominância, nós podemos observar uma simples relação na solução ótima do PFR que nos leva a um teste de redução para o problema. Tal teste pode ser usado como um mecanismo de pré-processamento do grafo de entrada para qualquer algoritmo que se proponha a resolver o PFR. O Teorema 2.3 implica em um procedimento para reduzir o grafo de entrada descartando arestas subótimas que não satisfaçam a desigualdade triangular, ou que a satisfaça somente

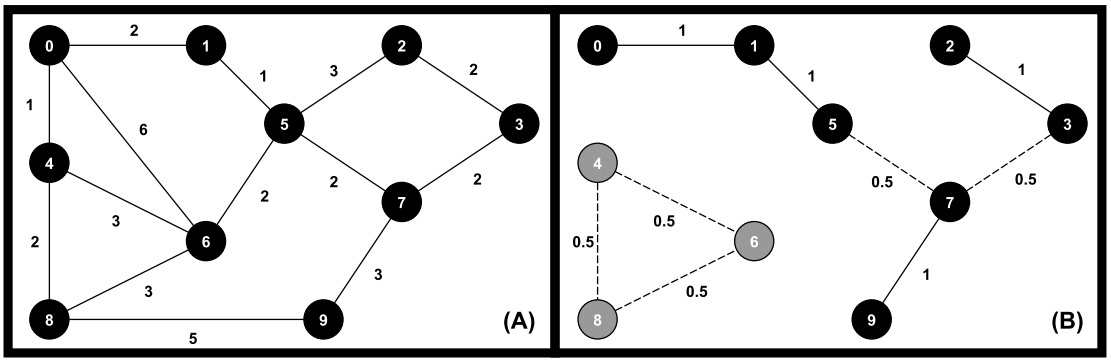

Figura 2.1: Exemplo de instância com $n = 10$ (A) e solução fracionária viável em $\mathcal{P}_1$ e inviável em $\mathcal{P}_2$ (B).

na igualdade. Por conta disso, esse teste é mais eficiente em grafos densos ou não-euclidianos, embora também possa ser efetivo em grafos euclidianos onde a igualdade seja satisfeita.

Teorema 2.3. *Seja P um caminho de G com vértices terminais em u e v, mas não incluindo a aresta (u, v). Então, se $w_{uv} \geq w(E(P))$ acontecer, existe uma solução ótima para o PFR que não usa (u, v).*

Demonstração. Seja $F = (V, E_F)$ uma Floresta Restrita ótima com $(u, v) \in E_F$. Logo, u e v pertencem à mesma árvore na solução abrangendo pelo menos $k-2$ outros vértices. Seja também $F' = (V, E_{F'})$ um grafo com $E_{F'} = E_F \cup E(P) - \{(u, v)\}$, onde $E(P)$ são arestas de P.
Note que F' pode ter arestas duplicadas se $E(P) \cap E_F \neq \emptyset$. Nesse caso, nós podemos naturalmente descartar as duplicatas, levando a uma solução com $w(E_{F'}) \leq w(E_F)$. Entretanto, F' pode ser uma k-floresta inviável. Isso ocorre devido ao potencial surgimento de ciclos com as arestas adicionadas. Além disso, F' não contém nenhuma componente conexa com menos de k vértices, já que adicionamos arestas em uma floresta que já respeitava as restrições de cardinalidade, embora a remoção da aresta (u, v) possa gerar duas novas componentes conexas. Contudo, já que garantimos que u e v permanecem na mesma árvore usando as arestas $E(P)$, isso nunca acontece. Da mesma forma, quaisquer outros vértices anteriormente conectados a u e v permanecem conectados em F'.
Finalmente, para o potencial surgimento de ciclos ao adicionar $E(P)$ na solução, nós podemos remover a aresta de maior peso em cada ciclo sem perder a viabilidade. Logo, F' é uma solução para o PFR com $(u, v) \notin E_{F'}$ e $w(E_{F'}) \leq w(E_F)$. □

Na Figura 2.2 temos um exemplo prático de onde o Teorema 2.3 pode ser aplicado. Nesse caso, a aresta (u, v) com custo 6 pode ser removida, pois dois caminhos

com custo menor ou igual ligam os mesmos vértices sem utilizá-la. O caminho (u, x, y, v) possui custo 6 e o caminho (u, y, v) possui custo 5. Pelo Teorema 2.2, se existe uma solução ótima usando a aresta (u, v), então existem vértices conectados a u e v, tal que a árvore que os contém respeita a cardinalidade k. Da mesma forma, isso é verdade para os vértices x e y (com suas respectivas árvores). Portanto, se substituirmos a aresta (u, v) nessa solução por qualquer um dos dois caminhos apontados, e removermos arestas excedentes (conforme apontado na demonstração), ainda manteremos a solução viável. O único efeito colateral está apenas em, potencialmente, reduzir a quantidade de árvores na solução, o que não impacta no custo nem na viabilidade.

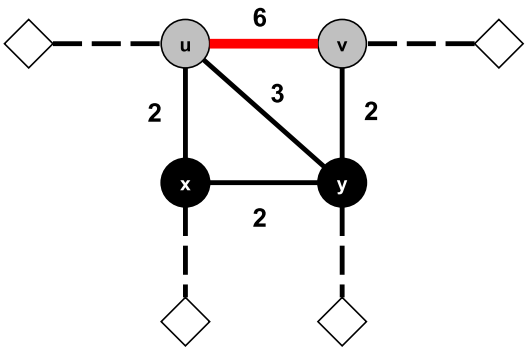

Figura 2.2: Exemplo de teste de redução usando o Teorema 2.3.

Para a aplicação desse teste como um passo de pré-processamento, é suficiente calcular o caminho mínimo entre cada par de vértices $u, v \in V$, excluindo $(u, v) \in E$. Para isso, foi utilizado o algoritmo de FLOYD [21]. Então, se o caminho mínimo encontrado cumpre o requisito do Teorema 2.3, nós descartamos a aresta (u, v) da solução.

Capítulo 3

Formulações Propostas

Sendo o PFR um problema $\mathcal{NP}$-Difícil, algoritmos de enumeração implícita, como *Branch-and-Bound* e *Branch-and-Cut*, são comumente utilizados para solucionar otimamente o problema. Esses algoritmos resolvem modelos lineares inteiros, onde os limites fornecidos cumprem um papel importante no desempenho. Os limites duais geralmente são obtidos através da relaxação linear do modelo em um determinado nó da enumeração. Desta forma, relaxações mais próximas da envoltória convexa do problema tornam a poda de ramificações da árvore de enumeração mais eficiente, o que tende a acelerar a convergência do algoritmo na direção do ótimo. Sabendo disso, neste capítulo são apresentadas e exploradas novas formulações para o problema.

3.1 Formulação Direcionada

Seja $D = (V', A')$ um grafo direcionado, onde $V' = V \cup \{r\}$, $A' = \{(r, j) \mid j \in V\} \cup A$, $A = \{(u, v), (v, u) \mid e = (u, v) \in E\}$, e $\hat{w} \in \mathbb{R}_+^{|A'|}$ com $\hat{w}_{rj} = 0, \forall j \in v$, $\hat{w}_{uv} = \hat{w}_{vu} = w_{uv}, \forall (u, v) \in E$. Logo, seja $\mathcal{P}_3$ uma formulação de Programação Inteira com o domínio de variáveis $z_a \in \mathbb{B}, \forall a \in A'$, assume-se 1 se o arco a está na solução, e 0 caso contrário. Da mesma forma que nas formulações não-direcionadas, seja $\delta^-(S) \subset A$ o subconjunto de arcos com origem em $V \setminus S$ e destino em S, $\delta^+(S) \subset A$ o subconjunto de arcos com origem em S e destino em $V \setminus S$, $\delta_r^-(S) \subset A'$ o subconjunto de arcos com origem em r e destino em S, $A(S) \subseteq A$ o subconjunto de arcos com origem e destino em S, e $z(A(\cdot)) = \sum_{a \in A(\cdot)} z_a$ a soma das variáveis implicadas pelo subconjunto de arcos $A(\cdot) \subseteq A'$.

$$(\mathcal{P}_3)\ \min \quad \hat{w}^T z \tag{3.1}$$

$$\text{s.t.:} \quad z_{ri} + z(\delta^-(i)) = 1, \qquad \forall i \in V \tag{3.2}$$

$$z(\delta_r^-(S)) + z(\delta^-(S)) \geq 1, \qquad \forall S \subseteq V \tag{3.3}$$

$$z(\delta_r^-(S)) - z(\delta^+(S)) \leq \left\lfloor \frac{|S|}{k} \right\rfloor, \quad \forall S \subseteq V \tag{3.4}$$

$$z(\delta^-(S)) + z(\delta^+(S)) \geq 1, \qquad \forall S \subseteq V, |S| < k \tag{3.5}$$

O problema reformulado tem como objetivo encontrar uma arborescência geradora enraizada em r de custo mínimo, onde cada subarborescência de r contém, pelo menos, k vértices. Note que, uma solução viável de $\mathcal{P}_3$ se torna também uma k-floresta com a remoção dos arcos $\delta^+(r)$, e ignorando a assimetria dos arcos remanescentes. A função objetivo (3.1) é a mesma de $\mathcal{P}_1$ e $\mathcal{P}_2$, minimizando a soma dos pesos dos arcos escolhidos. As desigualdades (3.2) garantem exatamente um arco com destino em cada vértice, portanto impedem vértices isolados.

As desigualdades (3.3) são as clássicas desigualdades *cutset* direcionadas, que servem para evitar componentes isoladas, e são válidas para a reformulação direcionada proposta. As desigualdades (3.4) e (3.5) servem para forçar a restrição de cardinalidade para cada árvore (ou subarborescência) na nossa reformulação. Essas desigualdades foram inspiradas nas apresentadas por SIMONETTI *et al.* [53], e são versões direcionadas das desigualdades (2.5) e (2.2), respectivamente. Logo, apenas um desses conjuntos de desigualdades seria necessário para garantir a viabilidade do modelo. Em nossos experimentos, testamos o impacto da relaxação linear usando diferentes combinações dessas desigualdades.

Para as desigualdades (3.4), considere $|S| = z(\delta_r^-(S)) + \sum_{j \in S} z(\delta^-(j))$. Isso segue direto da adição das desigualdades (3.2) para cada $j \in S$. Note também que $\sum_{j \in S} z(\delta^-(j)) = z(\delta^-(S)) + z(A(S))$. Logo, se adicionarmos $|S|$ no lado esquerdo de (3.4), e $z(\delta_r^-(S)) + z(\delta^-(S)) + z(A(S))$ no lado direito, obtemos $z(A(S)) + z(\delta^+(S)) + z(\delta^-(S)) \geq |S| - \left\lfloor \frac{|S|}{k} \right\rfloor$. Tal desigualdade é equivalente à (2.5), mas usando as variáveis de arco definidas para $\mathcal{P}_3$. Portanto, (3.4) são desigualdades válidas para nossa reformulação do PFR. De maneira mais direta, as desigualdades (3.5) são válidas simplesmente substituindo as variáveis de arco em $\mathcal{P}_3$ com suas respectivas variáveis de aresta em $\mathcal{P}_1$.

Devido às suas variáveis de arco adicionais, $\mathcal{P}_3$ tende a sofrer com problemas de simetria. Desta forma, tentamos quebrar tais simetrias usando as desigualdades (3.6). Além disso, outra forma de quebrar simetrias desse modelo é fixando as variáveis $z_{r1} = 1$, $z_{j1} = 0$, $\forall j \in V$, e $z_{ri} = 0$, $\forall i > n - k$.

$$z_{ij} + z_{ji} + z_{rj} \leq 1, \quad \forall (i,j) \in E, i < j \tag{3.6}$$

Tais restrições impõem que qualquer subarborescência deve ter apenas vértices cujos índices são menores que a sua raiz. Vale notar que tais restrições não são capazes de melhorar a relaxação linear do problema.

Assim como foi feito em $\mathcal{P}_2$, temos as seguintes desigualdades para impor um limite inicial na convergência do algoritmo de planos de cortes. Note que essas desigualdades são casos específicos das desigualdades (3.3)-(3.5).

$$|V| - \left\lfloor \frac{|V|}{k} \right\rfloor \leq z(A(V)) \leq |V| - 1 \tag{3.7}$$

$$1 \leq z(\delta^+(r)) \leq \left\lfloor \frac{|V|}{k} \right\rfloor \tag{3.8}$$

3.2 Modelo com Árvores Explícitas

O modelo proposto a seguir tem como ideia enumerar cada árvore individualmente na solução. Para tal, seja o conjunto $T = \left\{1, 2, ..., \left\lfloor \frac{|V|}{k} \right\rfloor \right\}$ de árvores possíveis. Para essa formulação, considere também três conjuntos de variáveis, são elas: $y^t \in \mathbb{B}$, que assume valor 1 caso a árvore t pertença à solução, 0 caso contrário; $z_v^t \in \mathbb{B}$, que assume valor 1 caso o vértice v pertença à árvore t, 0 caso contrário; e $x_e^t \in \mathbb{B}$, que assume valor 1 caso a aresta e pertença à árvore t, 0 caso contrário. Desta forma, a nova formulação é apresentada a seguir.

$$(\mathcal{P}_4)\ \min \quad \sum_{e \in E} w_e \sum_{t \in T} x_e^t \tag{3.9}$$

$$\text{s.t.:} \quad \sum_{t \in T} z_v^t = 1, \qquad \forall v \in V \setminus \{1\} \tag{3.10}$$

$$\sum_{t \in T} x_e^t \leq 1, \qquad \forall e \in E \tag{3.11}$$

$$\sum_{v \in V} z_v^t \geq k y^t, \qquad \forall t \in T \tag{3.12}$$

$$x_e^t \leq \begin{cases} z_u^t \\ z_v^t \end{cases}, \qquad \forall t \in T, \forall e \in E, e = (u, v) \tag{3.13}$$

$$\sum_{e \in E} x_e^t = \sum_{v \in V} z_v^t - y^t, \qquad \forall t \in T \tag{3.14}$$

$$\sum_{e \in \delta(v)} x_e^t \geq z_v^t, \qquad \forall t \in T, \forall v \in V \tag{3.15}$$

$$\sum_{e \in E(S)} x_e^t \leq \sum_{v \in S} z_v^t - z_u^t, \qquad \forall t \in T, \forall S \subseteq V, \exists u \in S \tag{3.16}$$

$$z_1^1 = y^1 = 1 \tag{3.17}$$

$$y^{t-1} \geq y^t, \qquad \forall t \in T \setminus \{1\} \tag{3.18}$$

$$z_v^t \geq 1 - \sum_{u=1}^{v-1} z_u^t - \sum_{j=1}^{t-1} z_v^j, \quad \forall t \in T \setminus \{1\}, \forall v \in V \setminus \{1\} \tag{3.19}$$

$$x \in \mathbb{B}_{|E|}^{|T|}, y \in \mathbb{B}^{|T|}, z \in \mathbb{B}_{|V|}^{|T|} \tag{3.20}$$

A função objetivo (3.9) visa minimizar o custo das arestas utilizadas por cada árvore. As restrições (3.10) garantem que todo vértice esteja exatamente em uma árvore, para garantir uma floresta geradora. Da mesma forma, as restrições (3.11), asseguram que uma aresta pertença à no máximo uma árvore. As restrições (3.12) forçam cada árvore ativa na solução a ter pelo menos k vértices. As restrições (3.13) garantem que uma aresta só pode ser utilizada por uma árvore que contenha seus dois vértices incidentes. Nas restrições (3.14) são atribuídos o número de arestas de cada árvore, enquanto as restrições (3.15) garantem a conectividade de cada vértice. Os subciclos são eliminados da solução através das restrições (3.16). Para quebra de simetria do modelo, as restrições (3.17)-(3.19) foram incorporadas. Por fim, as restrições (3.20) são as de binaridade das variáveis.

As restrições (3.16) são incorporadas ao modelo sob demanda, assim como foi feito com as restrições de ordem exponencial nos modelos até aqui apresentados. Para a separação de tal restrição, aplicamos o algoritmo de corte mínimo em cada árvore ativa do grafo suporte separadamente, seja a solução inteira ou fracionária. Logo, se em qualquer árvore, os cortes encontrados partindo de um vértice para todos os demais forem inferiores a z_u^t, sabemos que há um subciclo em algum dos conjuntos induzidos pelo corte, e então adicionamos a restrição violada ao modelo. Essa separação é uma adaptação da utilizada para desigualdades do tipo *cutset* [56].

3.3 Modelo de q-Arborescência

Uma formulação com variáveis indicando capacidade para o PFR é apresentada a seguir. Tal formulação é baseada na proposta por GOUVEIA [25] para o problema de árvore geradora mínima capacitada. Seja $D = (V', A')$ o grafo direcionado como definido na Seção 3.1. O modelo apresentado a seguir utiliza um conjunto de variáveis com indexação de capacidade e representação de arborescência em D: x_{ij}^q assume 1 se o arco (i, j) está na solução com arborescência em j e capacidade q; e 0 caso contrário.

$$(\mathcal{P}_5)\ \min \quad \sum_{(i,j)\in A} w_{ij} \sum_{q=1}^{|V|} x_{ij}^q \tag{3.21}$$

$$\text{s.t.:} \quad \sum_{q=k}^{|V|} \sum_{j\in V} x_{rj}^q \geq 1, \tag{3.22}$$

$$\sum_{q=k}^{|V|} x_{rj}^q + \sum_{q=1}^{|V|-1} \sum_{(i,j)\in\delta^-(j)} x_{ij}^q = 1, \qquad \forall j \in V \tag{3.23}$$

$$\sum_{q=k}^{|V|} q x_{rj}^q + \sum_{q=1}^{|V|-1} q \sum_{(i,j)\in\delta^-(j)} x_{ij}^q = 1 + \sum_{q=1}^{|V|-1} q \sum_{(j,i)\in\delta^+(j)} x_{ji}^q, \quad \forall j \in V \tag{3.24}$$

$$\sum_{q=1}^{|V|-1} \sum_{(i,j)\in A(S)} x_{ij}^q + \sum_{q=1}^{|V|-1} \sum_{(i,j)\in\delta^+(S)\cup\delta^-(S)} x_{ij}^q \geq |S| - \left\lfloor \frac{|S|}{k} \right\rfloor, \quad \forall S \subseteq V, |S| \geq 2 \tag{3.25}$$

$$x \in \mathbb{B}_{|A'|}^{|V|} \tag{3.26}$$

A função objetivo (3.21) visa minimizar o peso de todos os arcos utilizados, independente de capacidade. A restrição (3.22) garante que, no mínimo, um arco saia do vértice artificial r com capacidade pelo menos k, gerando assim uma q-arborescência. As restrições (3.23) garantem que exatamente um arco deve chegar em cada vértice original. Para evitar ciclos e subarborescências as restrições (3.24) são aplicadas. Tais restrições são conhecidas como restrições de equilíbrio de capacidade [54], e funcionam garantindo que a capacidade de chegada em um vértice seja suficiente pra cumprir a sua demanda (unitária) e a demanda da subarborescência enraizada nesse vértice. As restrições (3.25) são as desigualdades de cortes adaptadas de JI [30] para a formulação, de maneira a tentar tornar o poliedro definido por $\mathcal{P}_5$ mais próximo da envoltória convexa. E finalmente, as restrições (3.26) garantem a integralidade das variáveis.

3.4 Separação de Desigualdades Exponenciais

Considere o seguinte modelo para admitir a separação exata de desigualdades do tipo *Cutset* com cardinalidade máxima de conjuntos (2.2), as quais assumem a forma $x(\delta(S)) \geq 1, \forall S \subset V, 1 \leq |S| < k$.

$$\begin{aligned}
(\mathcal{I}_1) \min_{s,t\in V, s\neq t} \quad & \sum_{a\in A} \bar{x}_a w_a & \\
\text{s.t.:} \quad & w_{ij} \geq y_i - y_j, & \forall (i,j) \in A \\
& w_{ij} \leq y_i, & \forall (i,j) \in A \\
& w_{ij} \leq 1 - y_j, & \forall (i,j) \in A \\
& y(V) \leq k - 1 & \\
& y_s = 1; y_t = 0 &
\end{aligned}$$

A formulação $\mathcal{I}_1$ modela o problema de corte mínimo, onde um dos conjuntos de vértices particionados precisa ter menos do que k vértices. Para tal, a formulação utiliza de variáveis de decisão de vértices $y \in \mathbb{B}^{|V|}$, que indica que um vértice pertence

ou não ao conjunto. As variáveis de arcos $w \in \mathbb{B}^{|A|}$ indicam quais arcos formam o corte. Tais variáveis só podem estar ativas caso exatamente um de seus vértices pertence ao conjunto formulado pelas variáveis y. Para separação das desigualdades (2.2), é suficiente encontrar o corte mínimo entre todos os pares distintos de vértice, usando o grafo suporte construído da solução fracionária de $\mathcal{P}_1$. Caso o custo seja menor que 1, temos uma violação das desigualdades (2.2). Tal formulação modela o problema de corte mínimo com cardinalidade de uma das partições limitada. Esse problema é provado $\mathcal{NP}$-Difícil para classes gerais de grafos com pesos nas arestas [12].

Igualmente, considere o seguinte modelo para prover a separação exata das desigualdades do tipo *Flower* (2.5), as quais assumem a forma $x(E(S)) + x(\delta(S)) \geq |S| - \left\lfloor \frac{|S|}{k} \right\rfloor, \forall S \subseteq V$.

$$
\begin{aligned}
(\mathcal{I}_2) \min_{i=0,1,...,\lfloor \frac{|V|}{k} \rfloor} \quad & \left\{ \min i + \sum_{e \in E} \bar{x}_e w_e - \sum_{v \in V} y_v \right\} & \\
\text{s.t.:} \quad & w_{uv} \leq y_u + y_v, & \forall (u,v) \in E \\
& w_{uv} \geq y_u, & \forall (u,v) \in E \\
& w_{uv} \geq y_v, & \forall (u,v) \in E \\
& \max\{2, ki\} \leq y(V) \leq \min\{k(i+1) - 1, n-1\} &
\end{aligned}
$$

Assim como anteriormente, a formulação $\mathcal{I}_2$ modela o problema de encontrar um conjunto de vértices, tal que a soma das arestas incidentes a todos os vértices desse conjunto seja mínima, enquanto a quantidade de vértices do conjunto seja máxima. Para tal, a formulação utiliza de variáveis de decisão de vértices $y \in \mathbb{B}^{|V|}$ e arestas $w \in \mathbb{B}^{|E|}$. De modo a usar tal problema para separação das desigualdades (2.5), linearizarizamos a porção $\left\lfloor \frac{|S|}{k} \right\rfloor$ do lado direito da desigualdade, discretizando o problema em etapas $i = 0, 1, ..., \lfloor \frac{|V|}{k} \rfloor$. Portanto, para cada i, resolvemos o problema limitando a cardinalidade do conjunto S, representado pelas variáveis y, a $\max\{2, ki\} \leq y(V) \leq \min\{k(i+1) - 1, n-1\}$. Logo, resolvendo qualquer desses problemas com custo de solução negativo leva a um conjunto $y(V)$ violado. Para tal problema, não se conhece na literatura resultados teóricos acerca da $\mathcal{NP}$-Completude.

Apesar de praticamente inviáveis como forma de separação das desigualdades em um esquema de algoritmo de *Branch-and-Cut*, tais formulações nos ajudam a resolver exatamente as relaxações lineares de $\mathcal{P}_1$ e $\mathcal{P}_2$, nos dando assim, uma informação a respeito do limite teórico que podemos atingir ao usar tais formulações. Desta forma, heurísticas surgem como boas alternativas para resolver tais problemas, de maneira que se aproxime o máximo possível do limite teórico sem comprometer o tempo

gasto com a parte de enumeração do algoritmo.

Desta forma, seja $\hat{G}$ o grafo suporte obtido ao se resolver uma relaxação linear de um modelo de programação linear com a solução $\hat{x}$. Para separação das desigualdades (2.2), propomos a heurística apresentada no Algoritmo 1, baseado nos algoritmos de corte mínimo (ou fluxo máximo) utilizados na separação de desigualdades *cutset* sem restrição de cardinalidade (`MinCut`). O algoritmo calcula o corte mínimo, entre cada par de vértices do grafo, e adiciona as desigualdades violadas advindas dos conjuntos induzidos por esse corte, cuja cardinalidade seja menor que k.

Algoritmo 1: separacao_P1($\hat{G}$, V, E)

```
1  início
2      para cada i, j ∈ V, i ≠ j faça
3          Seja Ĉ o corte mínimo entre i e j em Ĝ
4          Sejam S ⊂ V e S̄ ⊂ V os conjuntos induzido por Ĉ
5          se |S| < k e w(Ĉ) < 1 então
6              adicione desigualdade (2.2) para S
7          fim
8          se |S̄| < k e w(Ĉ) < 1 então
9              adicione desigualdade (2.2) para S̄
10         fim
11     fim
12 fim
```

Para separação das desigualdades (2.4)-(2.5), temos a heurística apresentada no Algoritmo 2. Tal algoritmo prioriza as desigualdades do tipo *Flower*, uma vez que elas impactam mais no valor da relaxação linear. O algoritmo também utiliza dos conjuntos induzidos pelos cortes mínimos do grafo residual. Como a solução do corte mínimo entre cada par de vértices em um grafo costuma ser custosa em termos computacionais, limitamos sua aplicação apenas ao nó raiz da árvore de enumeração. Além disso, apresentamos uma heurística gulosa, que é aplicada até o quinto nível da árvore de enumeração. Para a separação nos demais nós, buscamos apenas violações em soluções inteiras.

Nossa heurística gulosa do Algoritmo 2 (linhas 14-42) começa verificando todas as componentes conexas do grafo residual, utilizando *Breadth-First Search* (BFS). Note que nesse caso, já que estamos lidando com um grafo cujos pesos nas arestas são contínuos no intervalo $(0, 1)$, consideramos como inexistentes todas as arestas cujo peso seja menor que um $\varepsilon = 10^{-5}$. Para cada conjunto S induzido por essas componentes conexas, verificamos se ele viola alguma das desigualdades e aplicamos a devida restrição. Caso ele não encontre nenhuma violação com as desigualdades (2.5), o algoritmo procede com uma heurística gulosa baseada em circuitos elementares. Tal heurística verifica se o conjunto induzido por cada circuito elementar do grafo residual viola a desigualdade (2.5), caso contrário, nosso algoritmo tenta aumentar esse conjunto com vértices de $V \setminus S$ que maximizam a violação.

Algoritmo 2: separacao_P2($\hat{G}$, V, E)

início
  se *profundidade == 0* **então**
    para *cada* $i, j \in V, i \neq j$ **faça**
      Seja $\hat{C}$ o corte mínimo entre i e j em $\hat{G}$
      Seja $S \subset V$ e $\bar{S} \subset V$ os conjuntos induzidos por $\hat{C}$
      se $\hat{x}(E(S)) + \hat{x}(\delta(S)) < |S| - \left\lfloor \frac{|S|}{k} \right\rfloor$ **então**
        adicione desigualdade (2.5) para S
      fim
      se $\hat{x}(E(\bar{S})) + \hat{x}(\delta(\bar{S})) < |\bar{S}| - \left\lfloor \frac{|\bar{S}|}{k} \right\rfloor$ **então**
        adicione desigualdade (2.5) para $\bar{S}$
      fim
    fim
  fim
  se *profundidade* < 5 **então**
    Seja $\mathfrak{S}$ o conjunto de componentes conexas de $\hat{G}$
    flag $\leftarrow$ true
    para *cada* $S \in \mathfrak{S}$ **faça**
      se $\hat{x}(E(S)) > |S| - 1$ **então**
        adicione desigualdade (2.4) para S
      fim
      se $\hat{x}(E(S)) + \hat{x}(\delta(S)) < |S| - \left\lfloor \frac{|S|}{k} \right\rfloor$ **então**
        adicione desigualdade (2.5) para S
        flag $\leftarrow$ false
      fim
    fim
    se *flag* **então**
      Seja $\mathfrak{C}$ o conjunto de circuitos elementares de $\hat{G}$
      para *cada* $S \in \mathfrak{C}$ **faça**
        se $\hat{x}(E(S)) + \hat{x}(\delta(S)) < |S| - \left\lfloor \frac{|S|}{k} \right\rfloor$ **então**
          adicione desigualdade (2.5) para S
          continue
        fim
        enquanto *flag* **faça**
          S $\leftarrow S \cup v$, $v \in V \setminus S$, tal que $S \cup v$ possui maxima violação
          se $\hat{x}(E(S)) + \hat{x}(\delta(S)) < |S| - \left\lfloor \frac{|S|}{k} \right\rfloor$ **então**
            adicione desigualdade (2.5) para S
            flag $\leftarrow$ false
          fim
        fim
      fim
    fim
  fim
fim

Para encontrar o conjunto de circuitos elementares do grafo residual, utilizamos o algoritmo de Johnson para circuitos não-triviais [31], que tem uma complexidade $\mathcal{O}((|V|+|E|)(c+1))$, onde c é a quantidade de circuitos elementares do grafo. Note que esse número é exponencial e pode facilmente extrapolar um valor aceitável à medida que o grafo vai ficando mais denso. Logo, sua aplicação é limitada aos primeiros níveis da árvore de enumeração para não ter muito peso computacional no algoritmo. Além disso, cortes aplicados próximos da raiz impactam em mais nós.

Com a formulação $\mathcal{P}_3$, utilizamos as heurísticas BFS e `MinCut` para encontrar violações das desigualdades (3.4)-(3.5). A heurística BFS somente é aplicada em soluções inteiras. Além disso, utilizamos uma heurística gulosa baseada em estruturas chamadas de blocos (`BlAug`), proposta por SIMONETTI *et al.* [53] para o

Problema da Árvore k-Capacitada (kCTP, da sigla em inglês). Um bloco é um caminho direcionado cujos arcos de uma mesma direção compartilham o mesmo peso. Na Figura 3.1, por exemplo, o bloco da direita possui tamanho dois, uma vez que em um sentido os arcos possuem peso igual a 1.0, e no outro possuem peso igual a 0.0. Por outro lado, os vértices da esquerda não formam um mesmo bloco, já que em um sentido possuem peso igual a 0.75, mas no outro possuem pesos diferentes. Logo, são dois blocos separados.

Na nossa heurística, começamos usando um bloco como conjunto S potencialmente violado para desigualdades (3.4). Caso esse conjunto falhe, aumentamos S com um bloco adjacente a ele, tal que a violação seja maximizada. Isso é feito iterativamente até que uma violação seja encontrada, ou até que não seja mais possível aumentar o conjunto. Essa heurística também é utilizada para as desigualdades (3.5), com a observância da cardinalidade do conjunto S ser limitada. O conjunto de blocos de um grafo direcionado pode ser facilmente encontrado utilizando um algoritmo de *Depth-First Search* iniciando de todos os vértices do grafo. Além disso, blocos adjacentes são aqueles que contêm intersecção de vértices.

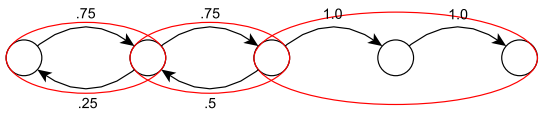

Figura 3.1: Exemplo de grafo com três blocos (elipses vermelhas)

Algoritmo 3: heuristica_blocos($\hat{D}$, V', A')

```
1  início
2      Seja 𝔅 o conjunto de blocos de D̂
3      para cada S ∈ 𝔅 faça
4          violado ← false
5          B ← 𝔅 \ S
6          enquanto não violado e B ≠ ∅ faça
7              se ẑ(δ⁻_r(S)) − ẑ(δ⁺(S)) > ⌊|S|/k⌋ então
8                  adicione desigualdade (3.4) para S
9                  violado ← true
10             senão
11                 S ← S ∪ b, b ∈ B, tal que b é adjacente a S e S ∪ b possua maxima violação
12                 B ← B \ b
13             fim
14         fim
15     fim
16 fim
```

3.4.1 Estratégias de *Branch-and-Cut* para $\mathcal{P}_3$

Considerando que estamos lidando com heurísticas para a separação das desigualdades (3.4)-(3.5), em qualquer nó da árvore de enumeração (e iteração do algoritmo

de planos de cortes), não temos garantia de encontrar alguma violação, portanto a relaxação linear pode ser impactada com a ordem de execução dessas heurísticas.

Por este motivo, tentamos duas diferentes abordagens utilizando as heurísticas propostas na Seção 3.4. Na primeira abordagem, denominada hierárquica, uma determinada heurística é aplicada somente quando a outra falha em encontrar alguma violação. Na segunda abordagem, ambas as heurísticas são aplicadas e quaisquer violações são adicionadas. Para a abordagem hierárquica, testes empíricos iniciais mostraram que é melhor começar com a heurística `MinCut`, já que ela tende a gerar mais desigualdades por iteração de planos de corte. As desigualdades *cutset* (3.3) são separadas de maneira exata se nenhuma violação for encontrada utilizando as heurísticas anteriores para separação de (3.4)-(3.5). Finalmente, se nenhuma violação dessas desigualdades for encontrada, nós tentamos separar desigualdades violadas (3.6), reiniciando assim o *loop* do algoritmo de planos de cortes no nó corrente.

Capítulo 4

Heurísticas Primais

Uma vez que não há algoritmo determinístico conhecido que resolva o problema em tempo polinomial, garantindo a otimalidade da solução encontrada, propomos dois métodos heurísticos. Grande parte da motivação em se explorar novas heurísticas, é devido ao fato de que métodos exatos requerem um custo computacional elevado, à medida que o tamanho das instâncias aumentam. Desta forma, heurísticas primais surgem como alternativas encontrando bons limites a um custo computacional significativamente menor. Tal comportamento pode ser observado em alguns experimentos do Capítulo 6. Para tal, neste capítulo propomos o desenvolvimento de duas heurísticas, uma baseada na metaheurística de Colônia de Formigas [18] e outra na metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP) [20].

4.1 Colônia de Formigas

Sendo apresentada como uma das primeiras metaheurísticas populacionais, e usada amplamente nos mais diversos problemas de caminho e roteamento em grafos [10], parece natural o emprego dessa técnica ao PFR. As primeiras abordagens de heurística primal apresentadas nesse trabalho são duas variações da heurística de Colônia de Formigas. A primeira, e mais simples, é conhecida originalmente na literatura como *Ant System* (AS), e representa a versão mais clássica do algoritmo, onde todas as m formigas da colônia contribuem a cada iteração para a atualização da lista de feromônios. A segunda, é uma versão mais robusta, conhecida como *Ant Colony System* (ACS), e introduz o conceito de atualização de feromônio local. Neste caso, durante a construção da solução de cada formiga, as arestas escolhidas têm seu feromônio atualizado, de modo a evitar que formigas subsequentes formem soluções semelhantes. Desta forma, é menos provável que formigas de uma mesma iteração produzam soluções idênticas [18].

O Algoritmo 4 apresenta o funcionamento geral dos nossos algoritmos baseados em colônia de formigas. Para tal, ele recebe os dados de entrada do problema (G

Algoritmo 4: ant_colony(G, w, N, α, β, q_0, ρ, Q, φ)

```
1  início
2      (i, x*, w(x*)) ← (0, ∅, +∞)
3      m ← min(|V|, 100)
4      τ_0 ← init_pheromone()
5      τ_e ← τ_0, ∀e ∈ {1, .., |E|}
6      enquanto i < N faça
7          X ← ∅
8          para k in {1, .., m} faça
9              v ← random(1, |V|)
10             x ← generate_ant(G, w, v, α, q_0, τ, φ)
11             x ← hef_heuristic(G, w, x)
12             X ← X ∪ x
13         fim
14         x̄ ← refine(G, w, argmin_{x∈X}(w(x)))
15         se w(x*) > w(x̄) então
16             x* ← x̄
17         fim
18         update_pheromones(τ, X, ρ, Q)
19         i ← i + 1
20     fim
21     retorne (x*, w(x*))
22 fim
```

e w) e alguns parâmetros. Tais parâmetros, por ordem de listagem, são: número de iterações do algoritmo (N); controle relativo das informações de ferômonio (α) e heurística (β) na escolha de uma aresta; limiar da regra proporcional pseudo-aleatória (q_0); taxa de evaporação de ferômonio global (ρ); constante de proporção de ferômonio em cada arestas (Q); taxa de evaporação de ferômonio local (φ).

Os pontos de divergência entre ambas abordagens serão destacados a seguir. Primeiro, as variáveis de controle do algoritmo são inicializadas (linhas 2-5), onde m representa o número de formigas e τ_0 representa a quantidade de feromônio inicial com que cada aresta começa, e τ o vetor de feromônios associado a cada aresta. Seguindo as recomendações de DORIGO *et al.* [18], para a versão AS, o valor do feromônio inicial é dado pela expressão $\tau_0 = \frac{1}{|V| \times w(\hat{x})}$. Para a versão ACS, esse valor é dado por $\tau_0 = \frac{m}{w(\hat{x})}$, sendo $w(\hat{x})$ o custo de uma solução viável conhecida para o problema. No nosso caso, calculamos a solução inicial dada pela heurística HEF.

Depois, dado um número N de iterações do algoritmo, começamos cada iteração gerando m formigas (linhas 7-13). Tal etapa começa selecionando aleatoriamente, com igual probabilidade, um vértice v do grafo por onde o processo de geração irá começar. Para geração de soluções viáveis, utilizamos um algoritmo baseado no algoritmo proposto por PRIM [45] para a construção da floresta. Neste algoritmo, uma aresta e incidente ao vértice corrente da formiga, tal que sua adição à solução não forme um ciclo, tem probabilidade $p_e = \tau_e^\alpha \eta_e^\beta$ de ser escolhida, onde $\eta_e = \frac{1}{w_e}$ é a informação heurística. A aresta de maior probabilidade será escolhida, se um valor $0 \leq q \leq 1$ gerado aleatoriamente satisfizer $q \leq q_0$. Caso contrário, selecionamos aleatoriamente uma aresta incidente usando a distribuição de probabilidades calculada. Este critério é conhecido como regra proporcional pseudo-aleatória [18], e é

principalmente empregado em variantes ACS do algoritmo. No nosso caso, ambas as versões empregam essa regra de seleção. No caso da versão ACS, ainda temos uma etapa extra de atualização local do feromônio para a aresta selecionada, dada através da expressão $\tau_e \leftarrow (1-\varphi)\tau_e + \varphi\tau_0$, onde φ é conhecido como coeficiente de declínio do feromônio.

É importante notar que, caso o grafo de entrada não seja completo, não necessariamente podemos gerar uma solução viável para o problema. Por esse motivo, parametrizamos a execução de uma heurística simples de viabilidade, que consiste em adicionar a aresta de menor custo à solução para conectar árvores com menos de k vértices.

Após a geração da floresta por cada formiga, aplicamos a heurística HEF para refinar a solução antes dela ser integrada à colônia. Ao final da geração da colônia, uma heurística de refino é aplicada à solução de menor custo gerada naquela iteração. A heurística empregada visa uma melhora local contida, uma vez que é esperado que o algoritmo de geração seja capaz de prover diversidade e melhoria global através dos mecanismo de feromônios e probabilidades heurísticas. Para isso, pegamos as partições dadas pela solução escolhida e calculamos a arvore geradora mínima de cada componente conexa.

Nas linhas 15-17, verificamos se a melhor solução gerada naquela iteração é capaz de atualizar a melhor solução global e fazemos a atualização de acordo. Por fim, atualizamos os feromônios para a iteração seguinte. Na versão ACS, isso é feito através do método chamado de atualização de ferômonio *offline*. Para a versão AS do algoritmo, o feromônio associado a uma aresta e qualquer é atualizado pela expressão

$$\tau_e \leftarrow (1-\rho)\tau_e + \sum_{k=1}^{m} \Delta_e^k \tag{4.1}$$

onde $\Delta_e^k = \frac{Q}{w(x^k)}$ se $e \in x^k$, 0 caso contrário. Já para a versão ACS do algoritmo, considere que x^* é a melhor solução obtida até agora pelo algoritmo. Logo, temos que o feromônio associado a uma aresta e é atualizado através da expressão

$$\tau_e \leftarrow \begin{cases} (1-\rho)\tau_e + \frac{1}{w(x^*)}, & \text{se } e \in x^* \\ \tau_e, & \text{caso contrário.} \end{cases} \tag{4.2}$$

Na literatura, também é comum encontrar versões em que a atualização do feromônio para a versão ACS é feita com a melhor solução da iteração ($\bar{x}$ no Algoritmo 4). Entretanto, em experimentos preliminares, nosso algoritmo obteve melhor desempenho usando x^*.

Outro uso importante para o algoritmo de colônia de formigas, é como um *warm-*

start em algoritmos de geração de colunas [42], provendo um *pool* de soluções iniciais de boa qualidade, e assim, evitando custosas iterações na solução do subproblema. No nosso caso, armazenamos as colunas geradas a cada iteração como mecanismo de pré-processamento para o algoritmo descrito na Seção 5.2.

4.2 GRASP

Outra heurística proposta para resolver o PFR utiliza uma abordagem *Multi-Start*, baseada na metaheurística GRASP [20]. Estas metaheurísticas têm sido utilizadas com sucesso na literatura em diversos problemas de árvore, devido à sua simplicidade de implementação e eficiência na obtenção de bons limites primais [6, 40, 48]. O funcionamento geral do método está descrito no Algoritmo 5.

Algoritmo 5: GRASP(G, w, N, α, m, κ_0)

```
1  início
2      x* ← random_kruskal(G, w, 0)
3      x* ← hef_heuristic(G, w, x*)
4      (i, X) ← (0, {x*})
5      enquanto i < N faça
6          x̄ ← random_kruskal(G, w, α)
7          x̄ ← hef_heuristic(G, w, x̄)
8          x̄ ← local_search(G, w, x̄)
9          update_pool(X, m, κ0, x̄)
10         x̄ ← recombination(X, x̄)
11         update_pool(X, m, κ0, x̄)
12         se w(x*) > w(x̄) então
13             x* ← x̄
14             i ← 0
15         senão
16             i ← i + 1
17         fim
18     fim
19     x* ← recombination(X)
20     retorne (x*, w(x*))
21 fim
```

No Algoritmo 5 proposto neste trabalho, começamos com uma solução inicial puramente gulosa através do algoritmo de KRUSKAL [32], e refinada pela heurística HEF (linhas 2-3). Essa solução é assumida como a melhor solução corrente do algoritmo e adicionada ao *pool* de soluções (linha 4). Como critério de execução, adotamos um número N de iterações sem melhorias da solução global.

Para cada iteração do algoritmo, uma solução viável utilizando o critério guloso-aleatório é construída (linha 6). Para tal construção, utilizamos uma variação do algoritmo de KRUSKAL [32], em que construímos iterativamente uma floresta adicionando aleatoriamente uma aresta dentre as presentes na Lista Restrita de Candidatos (LRC). O conjunto de arestas disponíveis nessa lista é selecionado da seguinte forma

$$\text{LRC} = \left\{ e \in LC \mid w_e \leq \min_{e' \in E}(w_{e'}) + \alpha \left(\max_{e' \in E}(w_{e'}) - \min_{e' \in E}(w_{e'}) \right) \right\} \tag{4.3}$$

onde LC é conhecido na literatura como Lista de Candidatos, e é o conjunto de arestas viáveis não selecionadas para a solução. Nesse caso, LC é composto por todas as arestas que ainda não foram incorporadas à solução (parcialmente) construída e que não formam ciclos no grafo.

Após a construção, aplicamos a heurística HEF (linha 7) na tentativa de refinar a solução, i.e., remover arestas mantendo a restrição de cardinalidade entre as componentes conexas. Depois, uma nova etapa de refinamento é aplicada através de uma heurística de busca local (linha 8). Como busca local, utilizamos três abordagens diferentes (ou vizinhanças) em uma estratégia *Variable Neighborhood Descent* (VND) com *best improvement* [28]. As abordagens são, por ordem de aplicação dentro do *framework* VND:

1. `mst-hef` - Tomamos todas as árvores da atual solução que possuam tamanho estritamente maior que k, e calculamos a MST desse conjunto de vértices. Depois, refinamos essa árvore utilizando a heurística HEF para remover arestas custosas de modo a reduzir o custo parcial.

2. `shift-leaves` - Tomamos as folhas de uma árvore T qualquer e tentamos inserir suas folhas em uma outra árvore T'. Tal operação envolve a remoção de uma aresta e a adição de outra na solução, sempre que isso impacte na melhoria do custo parcial. Essa operação só é feita em árvores T que possuam pelo menos $k+1$ vértices, de modo a manter a viabilidade da solução.

3. `swap-leaves` - Tomamos as folhas de uma árvore T qualquer e tentamos trocar suas folhas com as de outra árvore T'. Tal operação envolve a remoção de duas arestas e a adição de outras duas na solução, sempre que isso impacte na melhoria do custo parcial.

Tendo $\bar{x}$ como a melhor solução gerada na iteração corrente do algoritmo, fazemos a atualização do nosso *pool* de soluções X (linhas 9 e 11). Uma solução $\bar{x}$ pode ser incorporada a X, se

1. $|X| < m$ e $\max\limits_{x \in X}(\kappa(\bar{x}, x)) \leq \kappa_0$; ou

2. $|X| = m$ e $\exists x \in X \mid \kappa(\bar{x}, x) \geq \kappa_0$ e $w(x) > w(\bar{x})$. Nesse caso, substituímos x por $\bar{x}$ em X.

Onde $\kappa_0 \in [0, 1]$ é um parâmetro para fator de similaridade de aceitação e $\kappa : \mathbb{B}^{|E|} \times \mathbb{B}^{|E|} \rightarrow \mathbb{R}$ é uma função que mapeia dois vetores binários num valor real indicando a similaridade entre eles. No nosso caso, considere $\kappa(x', x'') = \frac{|x' \cap x''|}{\max(|x'|, |x''|)}$.

O último mecanismo do nosso algoritmo, tenta fazer bom uso do conhecimento de boas soluções acumuladas ao longo das iterações. Desta forma, implantamos

um mecanismo de recombinação de soluções, aplicado de duas maneiras. A primeira, visa recombinar uma solução dada com todas as outras presentes do *pool* de soluções, e é empregada na linha 10 do nosso algoritmo. A segunda, visa realizar uma recombinação entre todas as soluções dentro do *pool*, e é empregada como pós-processamento, ao final do algoritmo (linha 19). Um passo de recombinação é realizado sempre sobre dois vetores solução. Para isso, construímos um grafo tal que o conjunto de arestas é composto pela união entre as arestas dos grafos das duas soluções, e então utilizamos a heurística HEF para transformar esse grafo em uma k-floresta.

Por fim, tentamos mais uma vez atualizar o *pool* de soluções com a nova solução recombinada, e tentamos atualizar a melhor solução adquirida até o momento pelo algoritmo, reiniciando o contador sempre que uma melhoria foi possível (linhas 11-17). Uma outra técnica de recombinação bastante conhecida, e amplamente utilizada em heurísticas GRASP, é a de Reconexão de Caminhos [49]. Diferentemente da empregada no nosso trabalho, essa técnica visa construir passo-a-passo uma solução em outra, e.g. adicionando uma aresta por vez. Nossa abordagem emprega um caminho direto (e mais rápido), simplesmente unindo as duas soluções e obtendo uma solução viável a partir do grafo união.

Capítulo 5

Decomposições

Dentre as técnicas de decomposição passíveis de aplicação sobre o PFR, neste capítulo, exploramos duas abordagens baseadas no algoritmo *Non-Delayed Relax and Cut* [39] (R&C), um método de relaxação lagrangiana adaptado para lidar com a dualização de conjuntos de restrições exponenciais. Tais reformulações foram baseadas no modelo de JI [30], apresentado no Capítulo 2. Além disso, também apresentamos duas decomposições DANTZIG e WOLFE [17], que foram baseadas em modelos propostos no Capítulo 3. Tais decomposições foram resolvidas através do método de geração de colunas.

5.1 Algoritmo *Relax-and-Cut*

Considere o problema de dualizar as desigualdades exponenciais (2.5) da formulação $\mathcal{P}_2$ através de relaxação lagrangiana. Considerando $\mu \in \mathbb{R}_+^{|\mathcal{S}|}$ as variáveis duais associadas a essas desigualdades, também chamados de multiplicadores lagrangianos, onde $\mathcal{S} = \{S \subseteq V, |S| \geq 2\}$ e seja $\varphi(S) = E(S) \cup \delta(S)$. O Problema de Relaxação Lagrangiana (PRL) que surge dessa dualização representa um limite inferior válido para o PFR. Entretanto, na literatura, não existem algoritmos de complexidade polinomial que levem à solução ótima desse PRL. Por esse motivo, nós consideramos dois PRLs derivados.

O primeiro PRL é obtido quando descartamos a desigualdade (2.8) de $\mathcal{P}_2$, o que ainda nos leva a um limite inferior válido. Vale notar que consideramos também dualizar explicitamente tal restrição, entretanto isso não trouxe nenhum impacto significativo na convergência do algoritmo, uma vez que essa restrição é um caso particular das desigualdades (2.5). Logo,

$$\mathcal{L}_1(\mu) = \min \sum_{e \in E} \left(w_e - \sum_{S \in \mathcal{S}: e \in \varphi(S)} \mu_S \right) x_e$$

$$+\sum_{S\in\mathcal{S}}\mu_S\left(|S|-\left\lfloor\frac{|S|}{k}\right\rfloor\right) \tag{5.1}$$

$$\text{s.t.:} \qquad x(E(S)) \leq |S|-1, \qquad \forall S\subseteq V \tag{5.2}$$

$$x(\delta(v)) \geq 1, \qquad \forall v\in V \tag{5.3}$$

$$x\in\mathbb{B}^{|E|} \tag{5.4}$$

onde $\mathcal{L}_1(\mu)$ é conhecido na literatura como problema de cobertura em floresta com custo mínimo. Tal problema é resolvível otimamente em tempo polinomial usando o algoritmo proposto por WHITE [55] e descrito no Algoritmo 6. O melhor limite teórico alcançável a partir de $\mathcal{L}_1(\mu)$ é dado pela resolução do problema de otimização $\mathcal{L}_1^*(\mu) = \max_{\mu\in\mathbb{R}_+^{|\mathcal{S}|}}\{\mathcal{L}_1(\mu)\}$.

Algoritmo 6: cobertura_em_floresta(V, E, $\hat{w}$)

```
1  início
2      Seja E_- ← {e ∈ E | ŵ(e) ∈ ℝ*_-}
3      Seja V_- ← {v ∈ V | δ(v) ∩ E_- ≠ ∅}
4      Seja V_+ ← V \ V_-
5      Seja V_± ← {v ∈ V_- | ∃e ∈ δ(v) ∉ E_-}
6      Seja Ĝ_+ ← (V_+ ∪ V_±, E[V_+] ∪ δ(V_+))
7      Seja F' ← EmparelhamentoPerfeitoMaximo(Ĝ_+)
8      Remova arestas e ∈ F' com custo ŵ(e) = 0 que não isolem vértices
9      Seja F'' a floresta de custo mínimo de (V_-, E_-)
10     retorne F' ∪ F''
11 fim
```

O Algoritmo 6 para resolução do problema de cobertura em floresta de custo mínimo recebe como entrada o grafo original e os custos lagrangianos associados a cada aresta ($\hat{w}$). O primeiro passo do algoritmo consiste em particionar o grafo de acordo com os seus custos lagrangianos, onde a parte negativa é composta por vértices tendo, pelo menos, uma aresta incidente com custo negativo (linhas 2-3). A parte positiva é composta exclusivamente por vértices que não possuem arestas incidentes de custos negativos (linha 4). Os vértices de corte, são aqueles que possuem arestas com ambos os sinais (linha 5).

Após o particionamento, o algoritmo resolve um problema de emparelhamento máximo perfeito [19] no subgrafo formado pelas arestas positivas e de corte, descartando arestas de custo nulo que ainda mantenham a propriedade de cobertura (linhas 6-8). Por fim, o algoritmo resolve o problema de floresta de custo mínimo na partição negativa do grafo. A união dos conjuntos dessas arestas resolvem o problema de cobertura em floresta de custo mínimo. GAMBLE e PULLEYBLANK [22] e SCHRIJVER *et al.* [52] discorrem em mais detalhes as demonstrações e algoritmos que levaram à solução do problema.

O segundo PRL proposto nesse trabalho é obtido quando também dualizamos as desigualdades (2.7). Sejam $\pi\in\mathbb{R}_+^{|V|}$ os multiplicadores lagrangianos associados a tais desigualdades. Portanto,

$$\mathcal{L}_2(\mu,\pi) = \min \sum_{e=(u,v)\in E}\left(w_e - \sum_{S\in\mathcal{S}:e\in\varphi(S)}\mu_S - \pi_u - \pi_v\right)x_e + \sum_{S\in\mathcal{S}}\mu_S\left(|S| - \left\lfloor\frac{|S|}{k}\right\rfloor\right) + \sum_{v\in V}\pi_v \quad (5.5)$$

$$\text{s.t.:} \quad x(E(S)) \leq |S| - 1, \qquad \forall S \subseteq V \quad (5.6)$$

$$x(E(V)) \geq |V| - \left\lfloor\frac{|V|}{k}\right\rfloor \quad (5.7)$$

$$x \in \mathbb{B}^{|E|} \quad (5.8)$$

onde $\mathcal{L}_2(\mu,\pi)$ modela o problema da floresta geradora mínima, resolvível em tempo polinomial através dos algoritmos de KRUSKAL [32] ou PRIM [45]. Além disso, o melhor limite teórico alcançável é dado pela resolução do problema $\mathcal{L}_2^*(\mu,\pi) = \max_{\mu\in\mathbb{R}_+^{|\mathcal{S}|},\pi\in\mathbb{R}_+^{|V|}}\{\mathcal{L}_2(\mu,\pi)\}$.

Ambos os problemas $\mathcal{L}_1^*(\mu)$ e $\mathcal{L}_2^*(\mu,\pi)$ são resolvíveis (aproximados) utilizando o Método do Subgradiente (MS) com algumas adaptações propostas por LUCENA [39] para lidar com os multiplicadores lagrangianos de ordem exponencial. O funcionamento do algoritmo, em linhas gerais é apresentado no Algoritmo 7.

O Método do Subgradiente adaptado (Algoritmo 7) recebe como entrada o grafo e os pesos associados a cada aresta. Além disso, recebe os parâmetros de número máximo de iterações (N), número de iterações sem melhora no *lower bound* global (Γ), a proporção de refino do tamanho de passo, $\alpha \in (0,1)$, e número de iterações (β) para permanência de cortes inativos em Δ. Nas linhas 2-6 as variáveis de controle do algoritmo são inicializadas, e na linha 7 a condição de parada do algoritmo é checada. Para cada iteração, o MS inicia resolvendo o PRL dados os custos lagrangianos atuais (linha 8). Os limites globais do algoritmo são atualizados nas linhas 9-15.

Na linha 11 aplicamos uma heurística primal sempre que o melhor *lower bound* é atualizado. Essa heurística é aplicada visando encontrar boas soluções viáveis para o problema original utilizando dos custos lagrangianos como informação. Esse comportamento é conhecido na literatura como heurística lagrangiana, e é de substancial importância que tais heurísticas sejam rápidas e capazes de encontrar soluções de boa qualidade. Para tal, utilizamos o algoritmo de KRUSKAL [32] como método de construção de uma solução viável, e a heurística HEF para refino dessa solução. Essas heurísticas foram utilizadas de três maneiras até se formar a melhor solução viável possível para o PFR:

1. usando as arestas da solução do PRL como solução inicial e os custos originais de cada aresta como objetivo de otimização;

Algoritmo 7: MS_adaptado(V, E, w, N, Γ, α, β)

```
1  início
2      UB ← +∞, LB ← −∞
3      k ← 0, i ← 0, w^k ← w, α^k ← 2
4      inicialize multiplicadores λ^k ← 0
5      x* ← x̄* ← ∅
6      X ← Δ ← {}
7      enquanto UB - LB > 1 e k < N e α^k > 10^−10 faça
8          (LB^k, x^k) ← PRL_solver(V, E, w^k)
9          se LB^k > LB então
10             (LB, x*) ← (LB^k, x^k)
11             (UB^k, x̄^k) ← heuristica_primal(x^k, w^k)
12             se UB^k < UB então
13                 (UB, x̄*) ← (UB^k, x̄^k)
14             fim
15             i ← 0
16         senão
17             i ← i + 1
18         fim
19         X ← X ∪ fixar_variaveis(x^k, w^k)
20         se i = Γ então
21             α^k ← α^k * α
22             (LB^k, x^k) ← (LB, x*)
23             i ← 0
24         fim
25         g^k ← (b − Ax^k)
26         buscar_cortes(x^k, Δ, β)
27         θ^k ← α^k(UB−LB^k) / ||g^k||
28         λ_j^{k+1} ← max{0, λ_j^k − θ^k g_j^k}, ∀j = 1, ..., m
29         w^{k+1} ← w − λ^{k+1}
30         k ← k + 1
31     fim
32     retorne (UB, LB, x̄*, X, Δ)
33 fim
```

2. a partir de uma solução inicial vazia, construímos e refinamos a solução tomando os custos lagrangianos associados a cada aresta como objetivo de otimização;

3. a partir de uma solução inicial vazia, construímos uma solução viável com o algoritmo de KRUSKAL [32] tomando os custos lagrangianos associados a cada aresta. A etapa de refinamento da heurística HEF é feito utilizando os custos originais do problema como objetivo.

Também implementamos um método de fixação de variáveis (linha 19) com base nos custos duais do lagrangiano e de uma estimativa de solução viável. De acordo com LUCENA [39], uma aresta subótima e pode ser descartada ($x_e = 0$) da solução em qualquer iteração k se $LB^k - \max\{x^k w^k\} + w_e^k \geq UB$. Onde $\max\{x^k w^k\}$ representa o maior custo dentre as arestas escolhidas como solução do PRL na iteração k.

Ainda segundo LUCENA [39], há uma forma de fixar variáveis na solução ($x_e = 1$) em qualquer iteração do MS. Para tal, seja e uma aresta qualquer do problema que desejamos fixar na iteração k, e seja LB_e^k o custo ótimo do PRL com a variável $x_e^k = 1$. Se $LB_e^k > UB$, temos que a aresta e pertence à solução

ótima do problema original. Esse teste é significativamente mais caro que o anterior, uma vez que requer a solução de um PRL para cada aresta. Por esse motivo, ele é geralmente empregado após o término das iterações do MS, onde espera-se que uma grande quantidade de variáveis já tenham sido descartadas e melhores limites estejam disponíveis. Entretanto, nos nossos experimentos iniciais com o PFR, o custo não se mostrou justificável em relação à quantidade de variáveis fixadas dessa maneira.

Nas linhas 20-24 temos a atualização do parâmetro de tamanho de passo do método, sempre a cada Γ iterações em que não há melhora no limite dual do algoritmo. Após essa atualização, reiniciamos a iteração corrente para a melhor solução global, de modo a aproveitar uma solução de boa estrutura.

Os subgradientes são computados nas linhas 25 e 26. Especificamente, as adaptações ao MS para lidar com o conjunto de desigualdades exponenciais dualizados passam por essas estapas. Seguindo sugestões de BEASLEY [9] para a boa convergência do MS original, LUCENA [39] sugere que sejam computados apenas os subgradientes (desigualdades) ativos na iteração corrente, i.e. $g_i^k > 0$. Além disso, qualquer desigualdade que possua multiplicador nulo $\lambda_i^k = 0$, deve ter seu subgradiente anulado também. Essas estratégias permitem lidar adequadamente com o grande conjunto de desigualdades dualizadas.

A etapa de adição de novos cortes, ou dualização de novas desigualdades violadas (linha 26), se resume a resolver um problema de separação polinomial, uma vez que o problema possui apenas variáveis inteiras. Para tal, basta realizar uma *Breadth-First Search* para identificar as componentes conexas no grafo e checar a violação de cada conjunto nas desigualdades. Ainda para evitar problemas de convergência e gerenciamento de muitas desigualdades, que também impactam nos custos lagrangianos, adicionamos apenas cortes considerados maximais em cardinalidade do conjunto de vértices. Desta forma, se forem encontradas desigualdades violadas com os conjuntos $S_1 = \{0, 1, 3\}$ e $S_2 = \{0, 1, 2, 3\}$, apenas a desigualdade associada a S_2 será adicionada. Além disso, qualquer desigualdade sob demanda previamente adicionada só permanece no *pool* de desigualdades Δ, se nas últimas $\beta \in \mathbb{N}$ iterações contribuiu com o custo lagrangiano, ou seja, esteve ativa ($g_i^k > 0$) e com multiplicador não-nulo ($\lambda_i^k \neq 0$).

Finalmente, nas linhas 27-30, é calculado o tamanho do passo da iteração corrente, bem como atualizados os multiplicadores e custos lagrangianos para a próxima iteração do método. Se algum dos critérios de parada for satisfeito, o algoritmo retorna as informações encontradas, seja para uma solução satisfatória do problema, ou para utilização como *warm-start* em outros métodos (linha 32).

Mesmo com essas adaptações, por vezes a convergência do MS pode ser limitada em termos numéricos. Por isso, testamos o uso das informações de boa qualidade

obtidas até um determinado número de iterações do MS como um mecanismo de *warm-start* para um algoritmo de B&C. Tais informações são particularmente úteis nos escopos de: limitação do problema, através da fixação de variáveis com os custos duais; boa seleção de cortes iniciais para as etapas de planos de corte; e limites primais e duais iniciais para a etapa de *branching* e *pruning*.

Com o objetivo de aproveitar o máximo possível de informação dual lagrangiana na população de um modelo de programação linear inteira, LUCENA [39] também apresentou um método para transformar soluções de problemas baseados em árvores em cortes do tipo SECs ou *cutsets*. Desta forma, além dos cortes presentes em Δ ao final da execução do MS, populamos o modelo com cortes de eliminação de ciclos construídos a partir da melhor solução dual obtida.

Ainda na tentativa de melhorar a performance de algoritmos B&C com informações do R&C, podemos aplicar testes de redução baseado nos custos reduzidos da relaxação linear obtida, de maneira semelhante ao que fizemos no lagrangiano. Isso pode se tornar particularmente mais efetivo com o *pool* de cortes e limites primais obtidos ao final da execução do R&C. Para tal, segue a Proposição 5.1.

Proposição 5.1. *Dado um limite superior (UB) válido de uma solução ótima para o problema, um limite inferior (LB) obtido através de uma relaxação linear deste problema, e o vetor de custos reduzidos associado à essa relaxação linear ($\bar{w}$). Se $\bar{w}_e + LB \geq UB$ for verdade, onde a coluna associada à variável x_e não pertence à base ótima, então existe uma solução ótima que não usa a aresta e, portanto $x_e = 0$.*

Demonstração. Trivial. A prova segue diretamente da definição de custos reduzidos em Programação Linear. □

5.2 Geração de Colunas

Considere $\mathcal{D}_1$ uma reformulação de $\mathcal{P}_3$, ao estilo apresentado por DANTZIG e WOLFE [17] para problemas de programação linear, onde as desigualdades (3.3) e (3.5) sofrem convexificação. Para isso, seja $\overrightarrow{\varphi}(S) = A(S) \cup \delta^+(S) \cup \delta^-(S)$ e considere um problema mestre onde existe exatamente um arco com destino para cada vértice (5.10). Neste problema, o número de colunas ativas deve ser limitado superiormente pelo número de árvores possíveis no PFR (5.12), e o conjunto de arcos selecionados em todas as colunas deve obedecer a restrição de cardinalidade mínima para cada componente conexa do grafo (5.11). Logo, seja a variável $\lambda^j \in \mathbb{R}$ que indica se a coluna j pertence à solução, $\forall j = 1, ..., p$, sendo que p representa o número de possíveis arborescências enraizadas em r contendo pelo menos $k+1$ vértices e k arcos. Finalmente, podemos reescrever implicitamente as variáveis $z_a = \sum_{j=1}^{p} d_a^j \lambda^j$, onde $d^j \in \mathbb{B}^{|A'|}$ é um vetor suporte que indica quais arcos pertecem à coluna j.

$$(\mathcal{D}_1)\ \min \quad \sum_{j=1}^{p}\left(\sum_{a\in A} w_a d_a^j\right)\lambda^j \tag{5.9}$$

$$\text{s.t.} \quad \sum_{j=1}^{p}\left(d_{ri}^j + \sum_{a\in\delta^-(i)} d_a^j\right)\lambda^j = 1, \qquad \forall i \in V \tag{5.10}$$

$$\sum_{j=1}^{p}\left(\sum_{a\in\overrightarrow{\varphi}(S)} d_a^j\right)\lambda^j \geq |S| - \left\lfloor\frac{|S|}{k}\right\rfloor, \quad \forall S \subseteq V, |S| \geq 2 \tag{5.11}$$

$$\sum_{j=1}^{p}\lambda^j \leq \left\lfloor\frac{|V|}{k}\right\rfloor \tag{5.12}$$

$$0 \leq \lambda^j \leq 1, \qquad \forall j = 1, ..., p \tag{5.13}$$

Para o subproblema $\mathcal{S}_1$, nós formulamos um problema de arborescência de custo mínimo contendo, pelo menos, k vértices mais a raiz. O custo de *pricing* para uma variável associada ao arco $a = (i, j)$ é dado por $\bar{w}_a = w_a - \pi_j - \sum_{S|a\in\overrightarrow{\varphi}(S)} \gamma_S$, $\bar{w}_{rj} = -\pi_j$, onde π, ρ e γ são as variáveis duais associadas às desigualdades (5.10), (5.12) e (5.11), respectivamente. A variável $x_a \in \mathbb{B}$ indica se o arco a pertence à arborescência, e a variável $y_i \in \mathbb{B}$ indica se um vértice i pertence à arborescência enraizada em r. Assim como definido na Seção 3.1, seja $x(\bar{A}) = \sum_{a\in\bar{A}} x_a$ e $y(S) = \sum_{i\in S} y_i$.

$$(\mathcal{S}_1)\ \min \quad \bar{w}^T x - \rho \tag{5.14}$$

$$\text{s.t.} \quad y(V) \geq k \tag{5.15}$$

$$x(\delta^+(r)) = 1 \tag{5.16}$$

$$x_{ri} + x(\delta^-(i)) = y_i, \qquad \forall i \in V \tag{5.17}$$

$$x_a \leq y_i, \qquad \forall a = (i, j) \in A \tag{5.18}$$

$$x(\delta_r^-(S)) + x(\delta^-(S)) \geq y_i, \qquad \forall S \subseteq V, \forall i \in S \tag{5.19}$$

$$x(\delta^+(S)) \geq x(\delta_r^-(S)), \qquad \forall S \subseteq V, 2 \leq |S| < k \tag{5.20}$$

$$kx(\delta_r^-(S)) + x(\delta^-(S)) \leq (k-1)x(\delta^+(S)) + y(S), \quad \forall S \subseteq V, |S| \geq 2 \tag{5.21}$$

$$x \in \mathbb{B}^{|A'|}, y \in \mathbb{B}^{|V|} \tag{5.22}$$

O objetivo (5.14) do subproblema $\mathcal{S}_1$ é encontrar colunas com o menor custo reduzido de arcos $\bar{w}$. Claramente, para que uma coluna seja considerada benéfica ao problema mestre $\mathcal{D}_1$, basta que seu custo seja negativo, não sendo necessário (em muitos casos) a solução ótima do subproblema $\mathcal{S}_1$. A desigualdade (5.15) garante um

número mínimo de vértices escolhidos na arborescência de modo a atender a restrição de capacidade, enquanto a restrição (5.16) garante que apenas uma arborescência enraizada exista na solução. As restrições (5.17) garantem que vértices ativos na solução possuam exatamente um arco chegando. Arcos que tenham como origem um vértice $i \in V$ qualquer só podem estar ativos caso este vértice esteja na solução (5.18). Finalmente, para a completude da formulação, as desigualdades de corte do tipo *cutset* generalizadas (5.19) são adicionadas sob demanda em cada relaxação linear.

Visando fortalecer ainda mais a relaxação linear do problema $\mathcal{S}_1$, as desigualdades válidas (5.20)-(5.21) foram implementadas. As desigualdades (5.20)-(5.21) são as mesmas apresentadas por SIMONETTI *et al.* [53] para o problema de árvore mínima de cardinalidade k, e visam limitar inferiormente a quantidade de arcos e vértices de cada arborescência. Ademais, as desigualdades (5.21) são conhecidas como desigualdades *Enhanced Reverse Multistar* (ERMS), e foram independentemente introduzidas por GOUVEIA e SALAZAR-GONZÁLEZ [26] para o Problema de Roteamento de Veículos.

$\mathcal{D}_2$ apresenta outra reformulação para o PFR, baseado em $\mathcal{P}_5$, e convexificando as restrições (3.22) e (3.24). Desta forma, reescrevemos implicitamente as variáveis $x_a^q = \sum_{j=1}^{p} d_a^{qj} \lambda^j$, onde $d^j \in \mathbb{B}^{|A'| \times |V|}$ é um vetor suporte que indica quais arcos capacitados fazem parte da coluna j.

$$(\mathcal{D}_2) \min \quad \sum_{j=1}^{p} \left(\sum_{a \in A} w_a \sum_{q=1}^{|V|} d_a^{qj} \right) \lambda^j \tag{5.23}$$

$$\text{s.t.:} \quad \sum_{j=1}^{p} \left(\sum_{q=k}^{|V|} d_{ri}^{qj} + \sum_{q=1}^{|V|-1} \sum_{a \in \delta^-(i)} d_a^{qj} \right) \lambda^j = 1, \quad \forall i \in V \tag{5.24}$$

$$\sum_{j=1}^{p} \left(\sum_{q=1}^{|V|-1} \sum_{a \in \overrightarrow{\varphi}(S)} d_a^{qj} \right) \lambda^j \geq |S| - \left\lfloor \frac{|S|}{k} \right\rfloor, \quad \forall S \subseteq V, |S| \geq 2 \tag{5.25}$$

$$0 \leq \lambda^j \leq 1, \qquad \forall j = 1, ..., p \tag{5.26}$$

Na literatura, o problema de *pricing* que surge da reformulação $\mathcal{D}_2$ é conhecido como Arborescência Capacitada de Custo Mínimo, ou *q-arborescence*, e trata-se de um problema também $\mathcal{NP}$-Díficil [43]. UCHOA *et al.* [54] resolvem uma relaxação do problema de *q-arborescence*, onde é permitido que vértices e arcos apareçam mais de uma vez na mesma arborescência. Desta forma, o subproblema pode ser resolvido em tempo pseudo-polinomial usando Programação Dinâmica. Tal abordagem foi utilizada para resolver o problema de Árvore Geradora Mínima Capacitada em instâncias de até 200 vértices.

Capítulo 6

Resultados e Discussões

Nesse capítulo, serão apresentados os resultados computacionais propostos para resolver o PFR. Para esses experimentos, nós utilizamos a linguagem de programação C++ (compilada com g++ 7.5.0). Como *solver* de Programação Linear e Inteira, foi utilizado o IBM ILOG CPLEX 12.8. Os resultados foram obtidos com o desabilitando todas as diretivas de preprocessamento, cortes automáticos e heurísticas primais do solver, a fim de garantir a máxima fidelidade às abordagens originais. A arquitetura utilizada para execução dos testes foi um Intel Core i7-7700HQ @ 2.80GHz com 8GB de RAM rodando no sistema operacional Ubuntu 18.04 64-bit. Todos os testes foram executados em uma única *thread*, e a menos que seja apresentado de outra forma em algum resultado específico, foi imposto um limite de tempo de 3600 segundos para cada experimento.

Pelo nosso conhecimento, até a data desse trabalho, não há conjunto de instâncias usadas como *benchmark* para comparar abordagens do PFR na literatura. Todos os trabalhos prévios usaram de instâncias arbitrárias geradas aleatoriamente, as quais não estão publicamente disponíveis, nem via contato com os autores. Isso torna a tarefa de comparar diferentes abordagens ainda mais difícil e imprecisa. Com isso em mente, utilizamos instâncias selecionadas de conjuntos disponíveis na OR-Library[1]. Esses conjuntos de instâncias foram retirados dos seguintes problemas de otimização clássicos, que compartilham similaridades com o PFR:

- **SHRD** - Árvore Geradora Mínima com Restrição de Grau. Grafos completos com $n \in \{30, 50, 70, 100\}$.
- **CAPMST** - Árvore Geradora Mínima Capacitada. Grafos completos com custos altamente simétricos e $n \in \{40, 50, 80, 100, 160, 200\}$.
- **TSPLIB** - Caixeiro Viajante. Instâncias clássicas com grafos completos de dimensões $n \in \{127, 150, 229, 280, 318\}$.

[1] `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`

- **kCTP** - Árvore de Cardinalidade k. Enorme variedade de grafos não-completos gerados com dimensões $n \in \{225, 400, 450, 1000, 1089\}$, enquanto também possui instâncias utilizadas pelo Problema da Árvore de Steiner. Para uma explicação detalhada sobre o processo de geração e escolha dessas instâncias, veja SIMONETTI *et al.* [53].

Esse capítulo se subdivide em quatro seções. Na Seção 6.1, serão apresentados os resultados computacionais acerca do teste de redução nas instâncias apresentadas, um estudo sobre as relaxações lineares, que demonstram na prática o estudo de dominância apresentados no Capítulo 2, e um estudo do impacto do valor de k no custo e dificuldade em resolver o problema nas diferentes instâncias. Continuando com os resultados de abordagens exatas, na Seção 6.2 serão apresentados os dados dos modelos de programação inteira (literatura e propostos) e suas decomposições, realizando comparações entre eles, quando aplicável. Os resultados heurísticos estão apresentados na Seção 6.3. Por fim, na Seção 6.4 apresentamos os resultados relativos aos experimentos envolvendo as decomposições propostas.

6.1 Resultados Preliminares

Esta seção apresenta os resultados obtidos por meio de uma série de estudos preliminares conduzidos como parte da pesquisa. Tais estudos desempenham um papel fundamental no delineamento e validação das abordagens adotadas, bem como na identificação de tendências e desafios sobre o PFR. Nestes experimentos, avaliamos o impacto do parâmetro de entrada k na solução do PFR, o impacto dos testes de redução propostos nas instâncias testadas, e um estudo acerca dos limites duais obtidos pelos modelos de programação linear apresentados.

6.1.1 Impacto do Parâmetro k

Como um experimento inicial, tentamos identificar o impacto da restrição de tamanho mínimo de cada árvore do PFR nos diferentes tipos de instância testados, tanto em termos de custo de solução, quanto em dificuldade. Para isso, usamos a variação `CutL1` do nosso algoritmo (ver Seção 6.2 para referência), variando $k \in \{3, 4, ...\lfloor n/2 \rfloor\}$. Selecionamos uma instância por conjunto de teste. Os resultados de tempo computacional e custo estão na Figura 6.1.

Na Figura 6.1 cada instância contém dois gráficos. O primeiro é o gráfico que mostra a evolução do custo da solução, à medida que o parâmetro k cresce. Nestes gráficos mostramos o limite superior (`UB`) obtido pelo algoritmo `CutL1`, e o valor do limite inferior obtido pela relaxação linear do nosso modelo (`LR`) na raiz da árvore de enumeração. O segundo gráfico de cada instância apresenta o tempo gasto pelo

nosso algoritmo para processamento da raiz da árvore de enumeração (`Root`), e o tempo total do algoritmo (`BnC`).

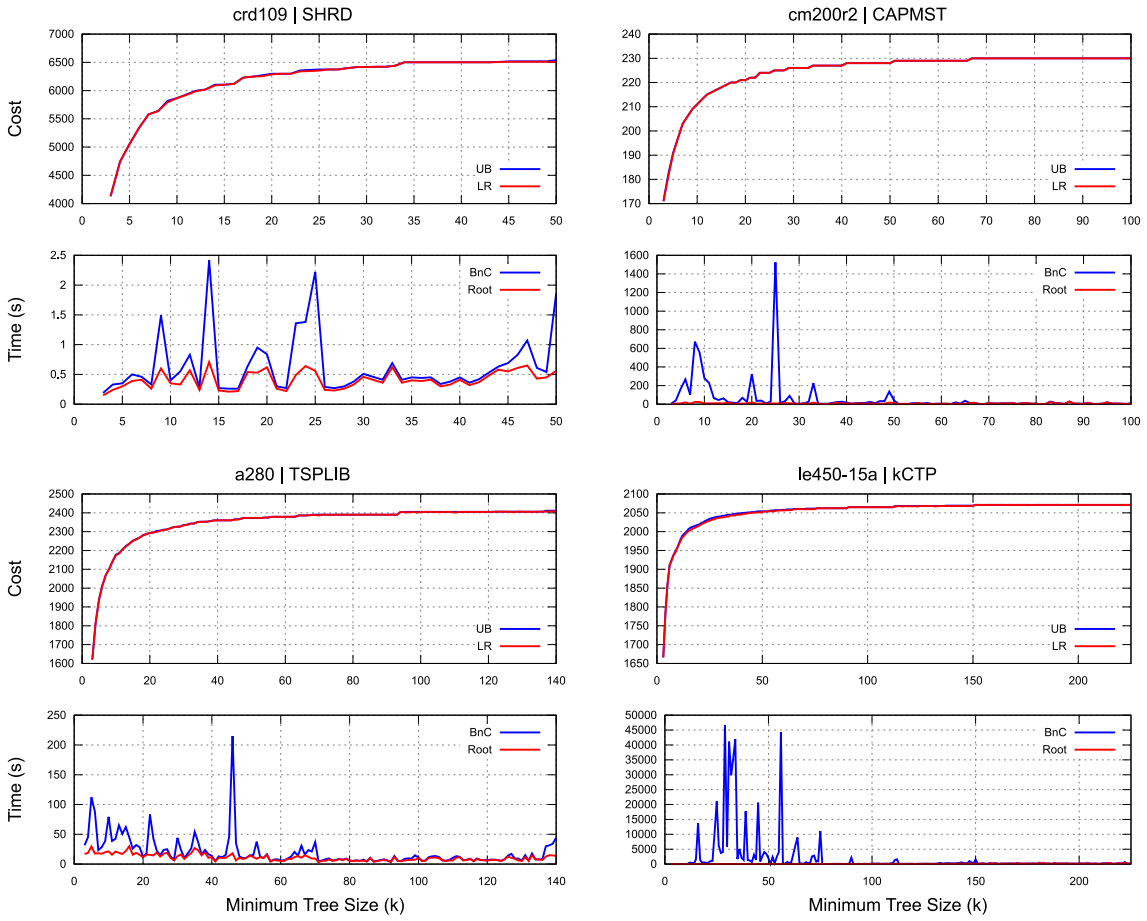

Figura 6.1: Custos de Solução & dificuldade (em tempo) ao variar k para diferentes instâncias.

Como podemos observar, a tendência geral de crescimento de custos segue uma escala logarítica, com a inclinação da curva dependendo do número de arestas utilizadas e seus pesos, o que está diretamente relacionado a n e k. Vale notar também que o custo da relaxação linear no nó raiz (LR, na figura) nesses casos é bem próximo ao limite primal ou custo ótimo, independente do valor de k. No geral, é interessante notar que nosso algoritmo tem maior dificuldade em resolver instâncias com valores de $k \leq \lfloor n/4 \rfloor$, especialmente para valores pequenos de k. Uma possível explicação para tal fenômeno é o elevado número de soluções viáveis usando árvores com tamanhos menores. Outra correlação é que, à medida que k aumenta, o número de arestas utilizadas tende a permanecer o mesmo. Portanto, as soluções podem compartilhar o mesmo grau de dificuldade que seus vizinhos com valores de k adjacentes, em alguns casos sendo exatamente a mesma solução.

Portanto, para os próximos experimentos, nós consideramos diversos valores de k para tentar cobrir os diferentes cenários apresentados nessa seção.

De maneira geral, escolhemos variar o prâmetro de entrada na sequência $k \in \{3, 4, 5, ..., 15, 20, 25, 30, 40, 50, 100, 125, 150, 175, 200, 250, ..., \lfloor n/4 \rfloor\}$.

6.1.2 Teste de Redução

Os resultados do teste de redução exposto no Teorema 2.3, são apresentados na Tabela 6.1. Tais resultados são agrupados por conjunto de instância, com as três últimas colunas indicando o número de arestas fixadas com o teste, a relação desse número com o total $|E|$, e o tempo gasto no algoritmo utilizado para realizar as verificações do teste. Os resultados são apresentados com três medidas estatísticas de média, mínimo e máximo para cada conjunto de instância.

Tabela 6.1: Resultados do Teste de Redução

Conjunto	**Instâncias**		**Arestas eliminadas**	**% de $\|E\|$**	**t(s)**
SHRD	104	Med	1265.87	47.22	0.09
		Min	126	28.97	0.02
		Max	3142	63.49	0.30
CAPMST	117	Med	6027.11	80.19	0.25
		Min	469	60.24	0.02
		Max	19072	95.84	0.68
TSPLIB	65	Med	22222.80	76.91	2.80
		Min	3411	42.64	0.54
		Max	37963	95.40	8.48
kCTP	131	Med	1362.33	19.77	2.65
		Min	1	0.05	0.05
		Max	6759	82.76	8.03

Como explicado antes, esse teste de redução é mais efetivo em grafos maiores e densos, ou em grafos cujas arestas tenham pesos altamente simétricos, como os exibidos nos resultados das instâncias do CAPMST e TSPLIB. Por outro lado, no conjunto kCTP, o teste de redução tem um impacto menor em geral, apresentando piores resultados nos grafos altamente esparsos baseados em grid. Em muitos casos, o uso do teste de redução diminui a simetria de alguns modelos e permite que os algoritmos *Branch-and-Cut* fechem seu *gap* de dualidade mais rapidamente, compensando a desvantagem do aumento no tempo de execução do solver. Se considerarmos os resultados com as heurísticas primais, o teste de redução permite uma redução considerável no espaço de busca, uma vez que estamos diminuindo o número de arestas a serem consideradas durante as etapas de construção e busca local (resultados na Seção 6.3).

Um experimento mais geral considera o impacto real de usar o teste de redução dentro de um framework de *Branch-and-Cut* em dois dos modelos apresentados neste trabalho. Nós fizemos testes em instâncias dos conjuntos SHRD, CAPMST e TSPLIB, cujas soluções ótimas foram alcançadas, portanto nos permite comparar os limites obtidos e performance com o teste de redução. Os resultados apresentados

na Tabela 6.2 estão agrupados por conjunto de instância, quanto ao uso do teste de redução (RT, na tabela), e o valor médio e máximo de cada coluna. As colunas nessa tabela são: # 0% **LR** representa o número de instâncias onde a Relaxação Linear na raiz equivale ao ótimo; **T (s)** representa o tempo total do algoritmo de B&C; **LR** (%) representa o *gap* relativo entre a relaxação linear na raiz da árvore de enumeração e o valor ótimo; **LR (s)** representa o tempo necessário para processar o nó raíz, i.e. resolver a relaxação linear. Nessa tabela apresentamos o impacto de performance da formulação $\mathcal{P}_2$, usando nossa heurística de separação da Seção 3.4, e a formulação $\mathcal{P}_3$ usando a heurística de separação hierárquica.

Tabela 6.2: Melhorias do Teste de Redução em algoritmos B&C

			$\mathcal{P}_2$ **c/ Nossa Heurística**				$\mathcal{P}_3$ **c/ Hierárquica**			
Conjunto	**RT?**		# 0% **LR**	**T (s)**	**LR** (%)	**LR (s)**	# 0% **LR**	**T (s)**	**LR** (%)	**LR (s)**
SHRD	No	Med	-	0.30	0.21	0.14	-	0.32	0.09	0.23
		Max	53	2.51	1.05	0.72	72	2.34	1.10	1.44
	Yes	Med	-	0.25	0.20	0.10	-	0.38	0.09	0.22
		Max	52	2.82	1.05	0.43	71	2.38	1.45	0.96
CAPMST	No	Med	-	23.17	0.19	5.73	-	51.03	0.11	1.91
		Max	67	490.00	1.12	130.05	75	2081.11	0.66	29.39
	Yes	Med	-	7.81	0.20	2.31	-	4.72	0.10	1.10
		Max	69	146.18	1.05	39.17	74	88.06	0.66	8.37
TSPLIB	No	Med	-	235.66	0.37	15.99	-	102.99	0.16	12.08
		Max	5	3600.00	1.31	88.98	14	3600.00	0.81	39.23
	Yes	Med	-	140.22	0.36	8.90	-	61.10	0.15	6.08
		Max	3	2431.38	1.09	130.13	15	1683.73	0.72	18.17

Os resultados sugerem que, na média, nosso teste de redução não é capaz de melhorar a qualidade dos limites duais, especialmente utilizando heurísticas para encontrar violações de desigualdades. Entretanto, ele é capaz de reduzir simetrias nos modelos, diminuir o tempo necessário para separar desigualdades violadas e também o número de iterações do algoritmo de Plano de Cortes. Isso é particularmente visível nas instâncias altamente simétricas do conjunto CAPMST e nos grafos completos do TSPLIB. Além disso, a formulação direcionada é a que mais se beneficia do teste de redução, uma vez que ela naturalmente tende a sofrer de problemas de simetrias. Entretanto, tais simetrias, quando bem tratadas, podem levar a identificação de violações mais fáceis.

Nosso teste de redução ajudou o modelo direcionado a facilmente superar o modelo não-direcionado da literatura em instâncias maiores, tanto em qualidade de limites, quanto em tempo total (mais resultados na Seção 6.2). Portanto, a menos que esteja explicíto nos resultados daqui pra frente, o teste de redução será aplicado em todos os algoritmos propostos neste trabalho.

6.1.3 Limites Duais

Computacionalmente falando, conduzimos experimentos para checar a relação apontada pelo nosso estudo de dominância (Teorema 2.2). Para isso, resolvemos as relaxações lineares de $\mathcal{P}_1$ e $\mathcal{P}_2$ usando programação inteira para separação exata de desigualdades exponenciais violadas. O conjunto SHRD foi selecionado para os testes em decorrência da natureza da separação implementada, por serem estas as instâncias as de menor complexidade dentre os conjuntos apresentados. Na Tabela 6.3, o *gap* de otimalidade está agrupado por modelo, ou conjunto de desigualdade. Os resultados são apresentados através de quatro medidas estatísticas de média, mínimo, máximo e desvio padrão (σ).

Tabela 6.3: Limites exatos de modelos da literatura

	$\mathcal{P}_1$	$\mathcal{P}_2$	(2.5)	(2.2) + (2.5)	$\mathcal{P}_1 \cup \mathcal{P}_2$
Med	6.43	0.12	0.62	0.17	0.12
Min	3.94	0.00	0.00	0.00	0.00
Max	10.51	0.93	2.28	1.12	0.93
σ	0.017	0.002	0.008	0.003	0.002

Como sugerido pelo nosso teorema, os limites obtidos por $\mathcal{P}_2$ são os melhores possíveis entre as duas formulações. Dessa forma, adicionar as desigualdades (2.2) em $\mathcal{P}_2$ não melhora a qualidade da relaxação linear do modelo, uma vez que tais desigualdades são dominadas por (2.4)-(2.5). Logo, nossos resultados computacionais reforçam os resultados teóricos. Por fim, vale notar que, uma vez que estamos lidando com um algoritmo exponencial, resolver otimamente o problema de separação para essas desigualdades em instâncias maiores se torna proibitivo.

6.2 Formulações

Nesta seção apresentamos os resultados decorrentes da aplicação, e as análises das formulações propostas neste trabalho. Os resultados obtidos mostram a eficácia e desempenho das nossas abordagens, frente às já existentes na literatura para a resolução do PFR com garantia de otimalidade.

6.2.1 Árvores Explícitas e q-Arborescência

Devido ao grande número de variáveis e simetrias das formulações $\mathcal{P}_4$ e $\mathcal{P}_5$, tivemos que limitar os resultados a um número bastante reduzido de instâncias do conjunto SHRD, uma vez que a capacidade computacional e o tempo limite foram facilmente extrapoladas nos experimentos das demais instâncias. De fato, dentre as duas formulações, somente $\mathcal{P}_4$ foi capaz de resolver na otimalidade instâncias com, no máximo,

30 vértices e $k = 6$. Desta forma, considere os resultados apresentados na Tabela 6.4 para a formulação $\mathcal{P}_4$ em comparação com $\mathcal{P}_2$.

Na Tabela 6.4, a primeira e a segunda colunas apresentam a quantidade de vértices e o tamanho mínimo de cada árvore na floresta, respectivamente. As duas colunas seguintes apresentam o *gap* de otimalidade e o tempo de execução de $\mathcal{P}_2$. Os dados da execução do algoritmo de enumeração aparecem nas próximas seis colunas, sendo elas: **UB**, limite superior; **LB**, o limite inferior; **Gap**, a diferença relativa entre esses dois limites; **T(s)**, o tempo de execução necessário; **Sols**, o número de soluções viáveis encontradas ao longo da execução; e **Nodes**, a quantidade de nós da árvore de *branching*. As últimas três colunas indicam os dados relativos à relaxação linear do modelo, com **Bnd** sendo o valor da relaxação, **T(s)** o tempo necessário para resolução do modelo relaxado, e **Gap** a diferença relativa entre o valor da relaxação e a solução ótima.

Tabela 6.4: Resultados para Formulação $\mathcal{P}_4$

		$\mathcal{P}_2$		***Branch-and-Cut***						**Relaxação Linear**		
n	k	**Gap**	**T(s)**	**UB**	**LB**	**Gap**	**T(s)**	**Sols**	**Nodes**	**Bnd**	**T(s)**	**Gap**
30	3	0.00	0.62	2462	2462.00	0.00	55.63	5	1619	2124.00	0.00	13.73
	4	0.00	0.54	2779	2779.00	0.00	24.51	4	985	2409.00	0.00	13.31
	5	0.00	0.74	3024	3024.00	0.00	67.31	5	2511	2521.00	0.00	16.63
	6	0.00	0.51	3123	3123.00	0.00	16.99	10	730	2627.00	0.00	15.88
30	3	0.00	0.60	2246	2246.00	0.00	18.94	4	905	1927.00	0.00	14.20
	4	0.00	0.32	2764	2764.00	0.00	40.97	1	1881	2217.00	0.00	19.79
	5	0.00	0.46	3070	3070.00	0.00	149.41	9	5791	2350.00	0.00	23.45
	6	0.00	0.52	3144	3144.00	0.00	32.89	8	1492	2485.00	0.00	20.96
Média		0.00	0.54	2826.50	2826.50	0.00	50.83	6	1989	2332.50	0.00	17.25

Para uma melhor exploração, apesar do péssimo desempenho desses modelos, na Tabela 6.5 apresentamos uma comparação entre as relaxações lineares dos modelos $\mathcal{P}_4$ e $\mathcal{P}_5$, em relação à solução ótima de cada instância. Para fins de comparação, também colocamos a relaxação linear obtida pelo atual estado-da-arte para o problema $\mathcal{P}_2$. Nesta tabela, a coluna **OPT** apresenta o valor ótimo da respectiva instância. As entradas da tabela com hífen significam que na resolução da relaxação linear os limites de memória ou tempo de processamento foram excedidos.

Podemos observar que o modelo $\mathcal{P}_4$ é capaz de obter melhores limites em comparação com $\mathcal{P}_5$, embora a demanda computacional torne a memória insuficiente para instâncias com 70 e 100 vértices. Isso impossibilita a utilização desses modelos como soluções de *Branch-and-Cut* em instâncias onde os demais modelos conseguem resolver facilmente. Além disso, o gap da relaxação linear dos modelos propostos é bem elevado, quando comparado com $\mathcal{P}_2$.

Ademais, formulações como $\mathcal{P}_5$ são bastante utilizadas em algumas soluções de *Branch-and-Cut-and-Price*, uma vez que a sua decomposição costuma levar a sub-

Tabela 6.5: Relaxações Lineares de $\mathcal{P}_4$ e $\mathcal{P}_5$

			$\mathcal{P}_2$			$\mathcal{P}_4$			$\mathcal{P}_5$		
n	k	**OPT**	**Bnd**	**T(s)**	**Gap(%)**	**Bnd**	**T(s)**	**Gap(%)**	**Bnd**	**T(s)**	**Gap(%)**
30	3	2462	2390.00	0.41	2.92	2124.00	0.00	13.73	1551.97	1.48	36.96
	4	2779	2751.00	0.41	1.01	2409.00	0.00	13.31	1883.29	1.21	32.23
	5	3024	2874.00	0.38	4.96	2521.00	0.00	16.63	2098.20	1.45	30.62
	6	3123	3056.00	0.38	2.15	2627.00	0.00	15.88	2256.61	1.11	27.74
30	3	2246	2121.00	0.41	5.57	1927.00	0.00	14.20	1397.90	1.34	37.76
	4	2764	2764.00	0.10	0.00	2217.00	0.00	19.79	1663.27	1.00	39.82
	5	3070	3040.00	0.35	0.98	2350.00	0.00	23.45	1850.22	1.56	39.73
	6	3144	3104.00	0.41	1.27	2485.00	0.00	20.96	1983.51	1.72	36.91
50	3	3146	3146.00	0.12	0.00	2631.60	1.80	16.35	1809.22	5.37	42.49
	4	3679	3585.00	0.43	2.56	2987.37	1.07	18.80	2196.23	4.52	40.30
	5	3843	3761.00	0.38	2.13	2961.00	0.63	22.95	2455.55	5.78	36.10
	6	4050	3984.00	0.43	1.63	3192.50	0.62	21.17	2642.92	3.89	34.74
50	3	2925	2917.00	0.42	0.27	2600.04	1.63	11.11	1682.28	3.53	42.49
	4	3433	3390.00	0.39	1.25	3339.00	0.38	2.74	2053.85	5.37	40.17
	5	3693	3557.00	0.29	3.68	3160.75	0.61	14.41	2311.74	5.64	37.40
	6	3856	3724.00	0.43	3.42	3409.14	0.48	11.59	2497.07	4.90	35.24
70	3	3801	3687.00	0.44	3.00	3135.51	10.58	17.51	2127.63	7.74	44.02
	4	4143	4091.00	0.41	1.26	3529.44	5.74	14.81	2593.59	7.63	37.40
	5	4558	4516.00	0.44	0.92	3751.38	3.83	17.70	2916.09	6.54	36.02
	6	4830	4765.00	0.45	1.35	3992.00	1.71	17.35	3147.87	8.08	34.83
70	3	3469	3377.00	0.36	2.65	3064.83	8.21	11.65	2208.90	7.16	36.32
	4	3976	3963.00	0.49	0.33	-	-	-	2667.22	8.30	32.92
	5	4249	4223.00	0.51	0.61	-	-	-	2963.85	6.84	30.25
	6	4498	4470.00	0.50	0.62	4001.00	1.94	11.05	3174.81	8.35	29.42
100	3	3682	3592.00	0.48	2.44	3227.86	92.37	12.33	2273.82	17.02	38.24
	4	4490	4454.00	0.50	0.80	-	-	-	2749.09	22.09	38.77
	5	4795	4729.00	0.43	1.38	-	-	-	3052.33	18.30	36.34
	6	5033	4947.00	0.54	1.71	-	-	-	3267.30	17.94	35.08
100	3	4131	4054.00	0.43	1.86	-	-	-	2358.50	14.86	42.91
	4	4747	4646.00	0.50	2.13	-	-	-	2863.33	18.39	39.68
	5	5054	5003.00	0.54	1.01	-	-	-	3182.17	19.34	37.04
	6	5340	5259.00	0.50	1.52	-	-	-	3416.16	18.74	36.03
Média			3748.13	0.41	1.79	2941.02	5.72	15.63	2304.48	4.90	36.49

problemas que podem ser resolvidos através de Programação Dinâmica. Tal reformulação foi proposta em $\mathcal{D}_2$, mas o subproblema resultante não se encaixou nessas características. Apesar disso, reescrevemos esse modelo no *framework* VRPSolver [44], na tentativa de gerar colunas para o problema, mas nenhum resultado digno de menção foi obtido a partir de tal estudo.

6.2.2 Heurísticas de Separação

Uma vez que estamos lidando com problemas de separação dos quais não se conhece algoritmo polinomial capaz de resolvê-los otimamente, o uso de heurísticas para tal problema merece atenção especial. Dessa forma, é essencial que o impacto do seu uso seja verificado na tarefa de achar cortes violados de boa qualidade em tempo aceitável. Portanto, a Figura 6.2 apresenta um estudo com resultados da relaxação linear de seis configurações diferentes das heurísticas de separação aplicadas a $\mathcal{P}_3$. Tais heurísticas são as mesmas apresentadas na Seção 3.4.

Os experimentos da Figura 6.2.A. são relativos aos conjuntos de instâncias

SHRD, CAPMST e TSPLIB. As seis combinações de heurísticas e cortes apresentadas são, em ordem: somente heurística `MinCut`; somente heurística `BlAug`; heurística `BlAug` aplicada depois que a heurística `MinCut` falhar na identificação de cortes (`Hierarchical`); ambas as heurísticas são aplicadas em cada iteração (`All In`); ambas as heurísticas são aplicadas a cada iteração, separando somente as desigualdades (3.3)-(3.4) (`CutL1`); ambas heurísticas são aplicadas a cada iteração, separando somente as desigualdades (3.3) e (3.5) (`CutL2`). Na Figura 6.2.B., os experimentos foram feitos com as instâncias kCTP com as três melhores combinações anteriores. Para os casos em que o *boxplot* está achatado, adicionamos gráficos escalados para melhor visualização.

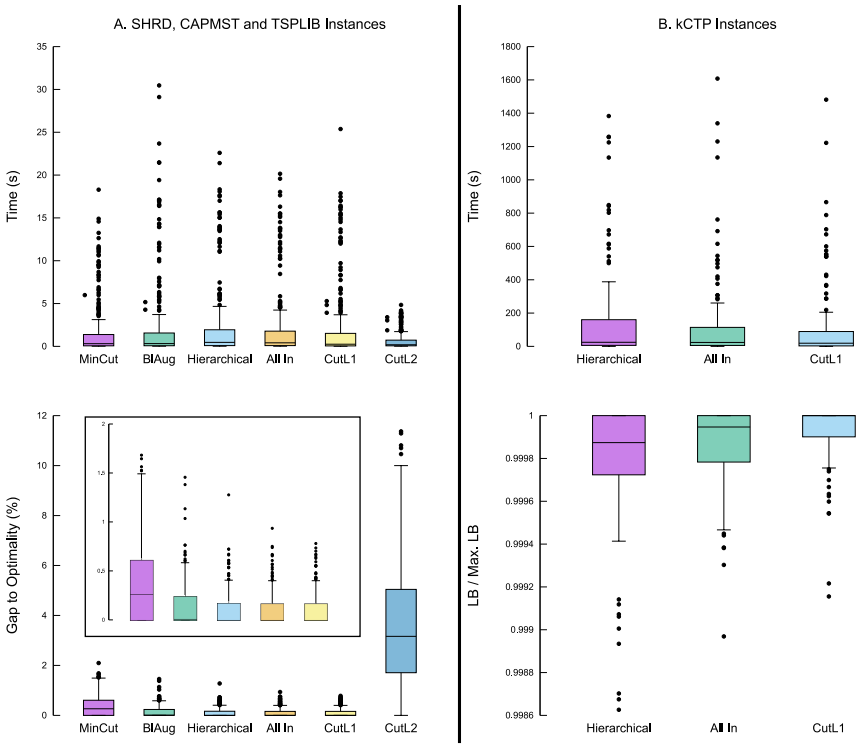

Figura 6.2: Comparação entre diferentes configurações de heurísticas de separação (tempo e limite) em conjuntos de instâncias diferentes.

A comparação entre as combinações de algoritmos e cortes levou em consideração duas métricas: valor da relaxação linear, e tempo de solução do nó raiz. Na Figura 6.2.A., a qualidade do limite é medida em termos de gap de otimalidade, portanto quanto mais próximo de zero, melhor. Para as instâncias kCTP (Figura 6.2.B.), o valor ótimo de algumas instâncias não é conhecido, então comparamos o limite dual de cada algoritmo relativo ao melhor limite dual obtido naquela instância. Portanto, quanto mais próximo de um, melhor o limite do algoritmo em comparação com os demais.

Os resultados mostram ligeira diferença em termos de tempo e qualidade dos limites entre as versões `Hierarchical`, `All In`, ou `CutL1` para os primeiros expe-

rimentos, com `CutL1` alcançando os melhores limites. Seguindo para as instâncias kCTP, os resultados são favoráveis à `CutL1` em tempo e qualidade do limite. Por fim, notamos que realizar a separação das desigualdades (3.5) não levou a uma melhoria de performance ou de qualidade do limite, no caso de uso de heurísticas para separação. Isso correlaciona diretamente com os resultados da Tabela 6.3, uma vez que (3.5) é uma versão direcionada das desigualdades (2.2).

6.2.3 *Branch-and-Cut*

Como primeiro estudo de comparação das formulações apresentadas aqui, com as existentes na literatura, relacionamos a relaxação linear do nó raiz da árvore de enumeração entre cada um dos nossos algoritmos com $\mathcal{P}_2$. Sendo que $\mathcal{P}_2$ é a formulação da literatura que leva aos melhores limites, de acordo com o Teorema 2.2, e com os resultados da Tabela 6.3. Desta forma, checamos a qualidade dos limites e número de cortes aplicados na raiz em todas as 417 instâncias testadas.

Os resultados desse estudo de relaxações lineares estão apresentados na Tabela 6.6, onde: # 0% **LR** indica a quantidade de instâncias cuja relaxação linear é igual à solução ótima; **Gap (%)** indica a diferença relativa entre o valor da relaxação linear obtida e o valor ótimo, ou melhor limite primal conhecido de cada instância; e as colunas referentes às desigualdades indica a quantidade de cada corte daquele tipo que foi incorporado ao modelo.

Para o modelo $\mathcal{P}_2$, comparamos duas versões. A primeira utiliza a heurística de separação da literatura, e a segunda utiliza nossa heurística de separação proposta para a formulação. Para $\mathcal{P}_3$, comparamos as três melhores combinações de separações heurísticas do nosso estudo da Seção 6.2.2.

Tabela 6.6: Estatísticas da Relaxação Linear: gap e número de cortes aplicados

		Gap (%)		**(2.4) or (3.3)**		**(2.5) or (3.4)**		**(3.5)**		**(3.6)**	
Algoritmo	# 0% **LR**	**Méd**	**Máx**	**Méd**	**Máx**	**Méd**	**Máx**	**Méd**	**Máx**	**Méd**	**Máx**
$\mathcal{P}_2$ c/ Lit. Heur.	76	1.13	15.12	28.56	237	384.53	4831	-	-	-	-
$\mathcal{P}_2$ c/ Nossa Heur.	128	0.41	3.66	19.51	808	5094.25	52730	-	-	-	-
$\mathcal{P}_3$ c/ Hier.	168	0.22	2.94	224.71	1169	2302.41	20409	1094.63	8078	219.04	1259
$\mathcal{P}_3$ c/ All In	172	0.21	2.95	389.12	2237	2825.16	21605	842.44	3974	188.94	1182
$\mathcal{P}_3$ c/ CutL1	172	0.21	2.95	419.40	2853	2842.25	23129	-	-	189.10	1176

Os resultados do nó raiz demonstram que desprender mais tempo identificando desigualdades violadas, podem melhorar ainda mais uma relaxação linear já bastante próxima do ótimo, e.g., diminuindo a média de 1.13 para 0.21 e, no pior caso, de 15.12 para 2.94. Além disso, uma relaxação linear mais próxima do ótimo pode ajudar imensamente na convergência da parte de *branching* do algoritmo. Nesse quesito, o modelo direcionado $\mathcal{P}_3$ apresenta uma melhoria considerável comparado a $\mathcal{P}_2$.

Para a performance geral e comparação de cada algoritmo B&C, nós compilamos os resultados dos experimentos em gráficos do tipo *boxplot*, os quais permitem uma melhor visualização das distribuições. Os algoritmos na Figura 6.3 seguem a mesma ordem dos apresentados na Tabela 6.6, onde os resultados estão agrupados por conjuntos de instâncias SHRD (A), CAPMST (B), e TSPLIB (C). Para cada conjunto de instância, nós comparamos os algoritmos em duas métricas: tempo total de execução do algoritmo em cada instância; e tempo do algoritmo dividido pelo menor tempo que um algoritmo levou para resolver cada instância. Portanto, na última métrica, quanto mais próximo de um, melhor desempenho o algoritmo apresenta dentre os comparados.

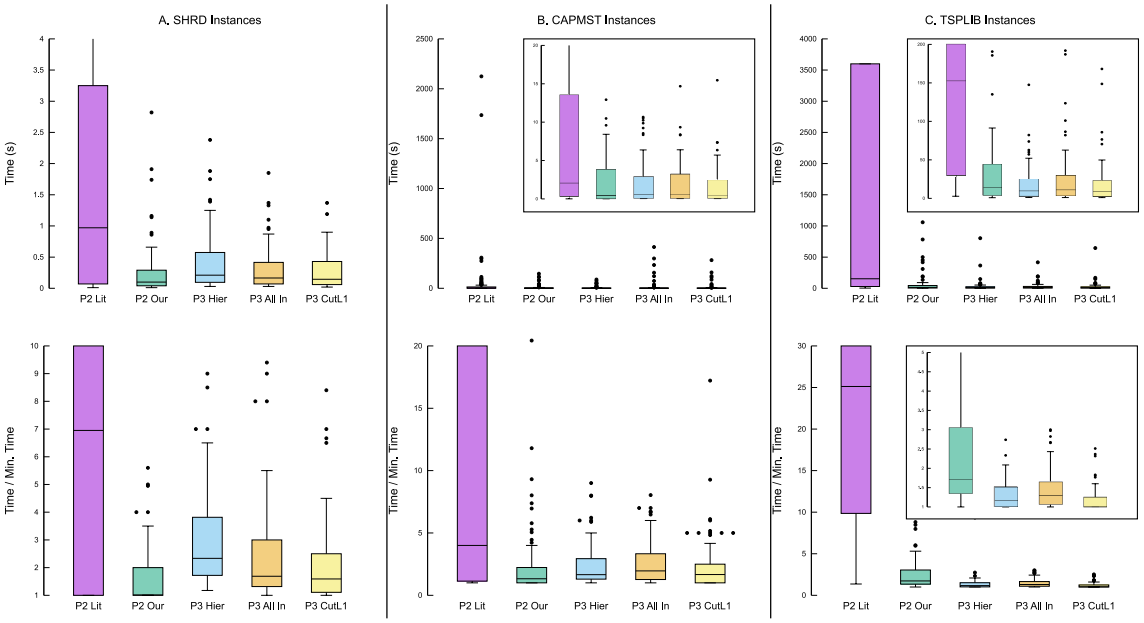

Figura 6.3: Performance dos diferentes algoritmos de B&C nos conjuntos SHRD (A), CAPMST (B), e TSPLIB (C).

No geral, os resultados dos algoritmos B&C confirmaram as indicações preliminares de melhor performance do nosso modelo direcionado $\mathcal{P}_3$. Além disso, nosso algoritmo de separação proposto para as desigualdades da formulação $\mathcal{P}_2$ obteve melhor desempenho, quando comparado ao da literatura. Nos conjuntos de instâncias SHRD e CAPMST, os resultados são inconclusivos entre a formulação $\mathcal{P}_2$ utilizando nossa separação, e $\mathcal{P}_3$ `CutL1`. As instâncias SHRD são relativamente fáceis de serem resolvidas por todos algoritmos, portanto pequenas flutuações ditam o comportamento médio, enquanto que nas instâncias CAPMST a enorme simetria pode causar problemas de desempenho, em alguns casos, para modelos direcionados. É somente nas instâncias do TSPLIB que os melhores limites obtidos pela formulação $\mathcal{P}_3$ prevalecem sobre os limites de $\mathcal{P}_2$, especialmente `CutL1`. Vale notar também que a formulação não-direcionada, com separação da literatura, foi a única incapaz

de encontrar o ótimo em todas as instâncias do TSPLIB, como mostrado no gráfico *boxplot* de tempo.

Para as instâncias kCTP, os resultados são apresentados na Figura 6.4. Uma vez que nossa versão de $\mathcal{P}_2$ foi capaz de superar em todos os aspectos a da literatura, limitamos nossas comparações para esse conjunto de instâncias somente aos algoritmos que tiveram a melhor performance nos conjuntos anteriores. O primeiro gráfico (Figura 6.4.A.) mostra o gap final dos algoritmos, usando o UB e LB individual. Em algumas dessas instâncias, os algoritmos não foram capazes de encontrar um limite primal dentro do tempo limite dado, portanto tais entradas não foram consideras na estatística. A quantidade desses casos é descrita explicitamente dentro de parênteses para cada algoritmo. Por exemplo, $\mathcal{P}_2$ `Our (18)` representa o estudo do gap final do algoritmo $\mathcal{P}_2$, usando nossa heurística de separação, para todas as instâncias do conjunto kCTP, exceto em 18 delas, em que nenhuma solução viável foi encontrada dentro do tempo limite de 3600 segundos.

Entretanto, esse estudo por si só pode ser ilusório, uma vez que alguns modelos podem justamente descartar as instâncias mais difíceis de serem resolvidas. Desta forma, na Figura 6.4.B., nós calculamos o gap obtido pelo algoritmo dado um UB válido fixado para cada instância. Tal limite pode ser o valor ótimo ou o melhor UB obtido até o momento. Desta forma, se o algoritmo não encontrar uma solução viável melhor que a dada, nós ainda temos uma referência da performance desse algoritmo. Entre parênteses, após o nome de cada algoritmo, está explicitado o número de ótimos encontrados dentro do conjunto. Finalmente, na Figura 6.4.C., nós temos a comparação do tempo total de execução dos algoritmos nos mesmos moldes da Figura 6.3.

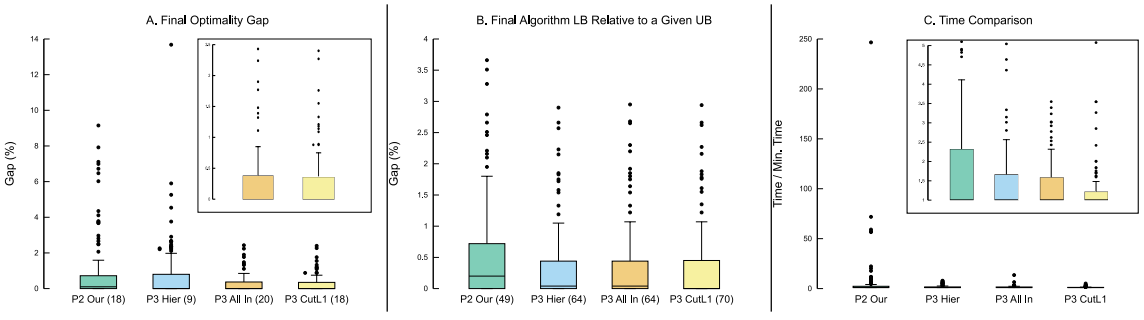

Figura 6.4: Comparação dos diferentes algoritmos no conjunto de instâncias kCTP: Gap final de otimalidade (A), Gap final dado um UB (B), e comparação de tempos de execução (C). Em (A), o valor entre parênteses nos nomes representa a quantidade de casos onde nenhuma solução viável foi encontrada pelo algoritmo. Em (B), o valor entre parênteses representa o número de casos onde o limite dual ótimo foi encontrado.

Conforme aconteceu nos experimentos anteriores, as margens entre as diferentes

separações do modelo direcionado são bem tênues, mas $\mathcal{P}_3$ `CutL1` tem a vantagem de ser mais resiliente. Considerando que é capaz de, consistentemente, encontrar mais soluções ótimas e de necessitar de menos tempo de execução em comparação aos demais métodos. Concluindo, nosso modelo direcionado foi capaz de, mais uma vez, superar sua versão não-direcionada em experimentos práticos.

6.3 Heurísticas Primais

Esta seção apresenta os resultados das abordagens heurísticas propostas para resolver o PFR. Tais resultados envolvem as heurísticas de colônia de formigas e GRASP.

Para a calibração dos hiperparâmetros das metaheurísticas aqui apresentadas, foi usado o *irace package* [37]. O software foi executado com um *budget* de 10^5, usando instâncias diferentes das usadas para teste e *benchmark*. O objetivo do *iterated racing* foi definido para minimização do custo da solução, ao mesmo tempo em que o tempo computacional foi considerado na escolha dos melhores candidatos elite. Para os testes heurísticos, o algoritmo foi executado 30 vezes para cada instância, com sementes aleatórias diferentes em cada execução.

Para o caso da heurística *Multi-Start*, o número de iterações sem melhora ms_{max} foi limitado a 30 para as instâncias kCTP, e 50 para os demais conjuntos de instâncias. Os demais parâmetros foram calibrados nos intervalos: $\alpha \in [0.0, 1.0]$, $\beta \in \{5, 10, 15, 20\}$, e $\gamma \in [0.7, 1.0]$. O melhor conjunto de parâmetros selecionado foi $\alpha = 0.4547$, $\beta = 15$, e $\gamma = 0.9262$.

Para o caso das heurísticas de Colônia de Formigas, o número máximo de iterações foi limitado a 50 para as instâncias testadas. O parâmetro $\alpha = 1.0$ foi definido para a versão *Ant System*, que é o valor mais indicado na literatura. No caso da versão ACS, o valor $\alpha = 1.317$ foi definido após calibração do *irace* no intervalo $[1.0, 5.0]$. Os parâmetros $Q = 1.0$ e $\varphi = 0.1$ também foram definidos de acordo com a literatura citada. Os demais parâmetros foram calibrados pelo *irace* nos intervalos $\beta \in [2.0, 10.0]$, $\rho \in [0.01, 1.0]$ e $q_0 \in [0.0, 1.0]$, retornando os valores: $\beta = 2.2456$, $\rho = 0.8774$ e $q_0 = 0.5887$ para a versão AS; e $\beta = 2.1777$, $\rho = 0.0205$ e $q_0 = 0.3556$ para a versão ACS.

Para as tabelas de resultados desta seção, considere as colunas: **B-Gap** representam o gap calculado usando a melhor solução obtida em todas as 30 execuções; **A-Gap** representam o gap calculado usando a solução média dessas execuções; **T (s)** representam o tempo médio dessas execuções em segundos. Finalmente, em cada linha de média nas colunas de gap, adicionamos o número de instâncias no qual o algoritmo foi capaz de alcançar a solução ótima (ou ultrapassou a melhor solução obtida pelos algoritmos B&C).

Os resultados dos algoritmos de Colônia de Formigas estão compilados na Fi-

gura 6.5. Como experimento comparativo, rodamos os algoritmos em um número reduzido das instâncias do conjunto SHRD, e os comparamos com os resultados da heurística HEF e GRASP propostas. Testamos as versões dos algoritmos com e sem a heurística de viabilidade apresentada.

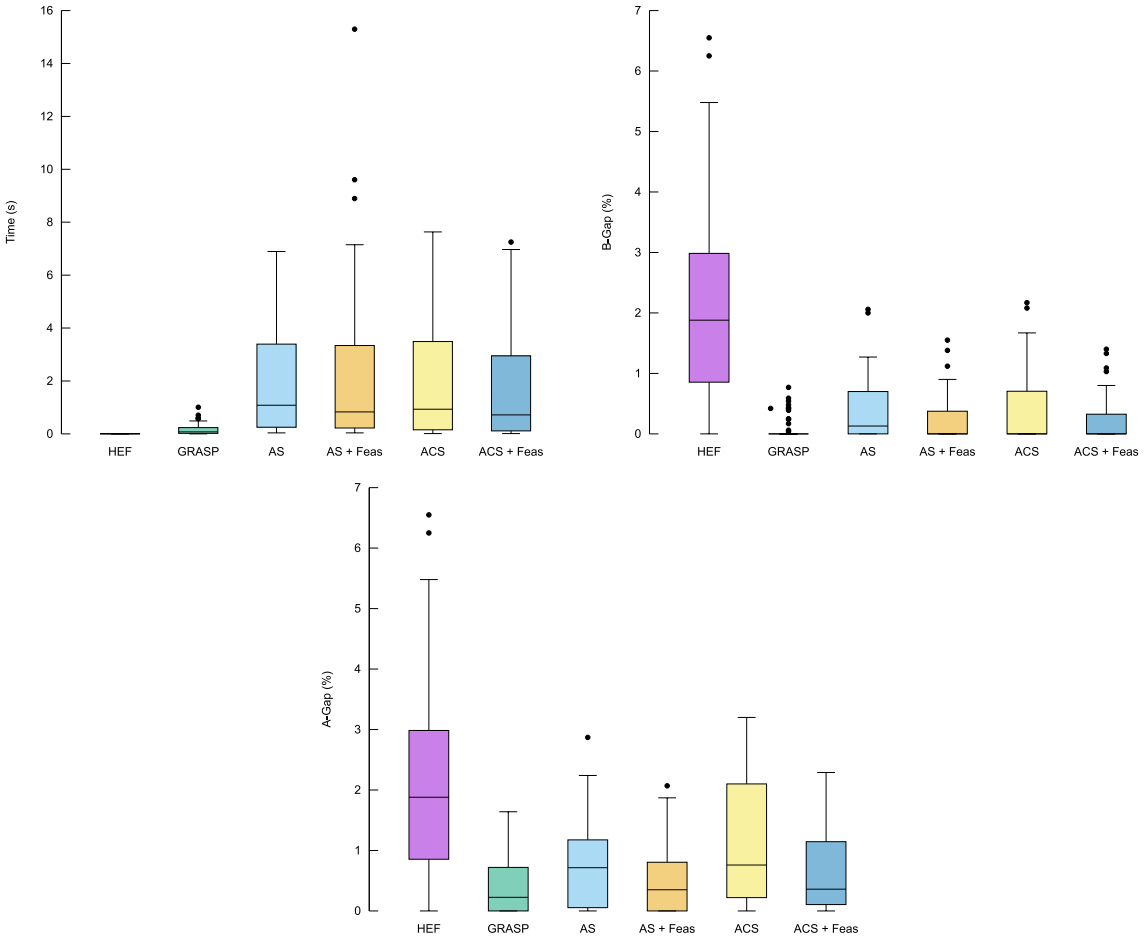

Figura 6.5: Comparação entre heurísticas primais: tempo de execução, gap da melhor solução e gap médio

Pela Figura 6.5, das configurações testadas dos algoritmos, `ACS + Feas` foi o melhor dentre as combinações de colônia de formigas, apresentando as melhores soluções, enquanto mantém o mesmo desempenho médio. Entretanto, vale notar que mesmo se tratando de uma heurística populacional, os algoritmos de colônia de formigas ainda não foram capazes de obter resultados igualmente satisfatórios aos do GRASP. Embora fosse esperado que heurísticas GRASP fossem mais rápidas, a qualidade da solução das heurísticas AS e ACS ficou aquém do esperado. Isso pode ser explicado pela busca local mais simplificada em comparação ao GRASP, uma vez que os algoritmos AS/ACS, por si só, já são bastante robustos e deveriam explorar melhor o espaço de busca.

Em decorrência desses resultados preliminares, decidimos prosseguir explorando

a heurística GRASP para todas as instâncias propostas. Uma vez que ela apresenta um bom balanço entre tempo de processamento e qualidade de solução, mesmo quando comparado com a heurística HEF pura. Portanto, considere os resultados para nossa heurística GRASP compilados na Tabela 6.7. Visto que não há *benchmark* de resultados ou instâncias, previamente disponíveis para o PFR, usamos a heurística HEF como comparação. Além disso, executamos experimentos alternando o uso dos nossos testes de redução para verificar o real impacto em ambas as heurísticas.

Ainda na Tabela 6.7, todos os valores são apresentados em três medidas estatísticas de média, máximo e desvio padrão (σ). Os *gaps* são calculados utilizando os valores ótimos para os três primeiros conjuntos de instâncias, e usando o melhor limite superior obtido pelos métodos exatos para o conjunto kCTP. Nos resultados em que o teste de redução foi utilizado, o tempo gasto em pre-processamento foi considerado nos resultados.

Tabela 6.7: Resultados da heurística GRASP

			HEF		**GRASP**		
Conjunto	**RT?**		**Gap (%)**	**T (s)**	**B-Gap (%)**	**A-Gap (%)**	**T (s)**
SHRD	Não	Méd	$1.81^{\dagger 15}$	0.005	$0.06^{\dagger 88}$	$0.36^{\dagger 24}$	0.157
		Máx	6.55	0.010	0.77	1.64	1.008
		σ	1.55	0.01	0.16	0.39	0.19
	Sim	Méd	$1.81^{\dagger 15}$	0.053	$0.08^{\dagger 86}$	$0.34^{\dagger 24}$	0.169
		Máx	6.55	0.180	0.98	1.75	0.665
		σ	1.55	0.06	0.21	0.39	0.18
CAPMST	Não	Méd	$2.56^{\dagger 17}$	0.015	$0.56^{\dagger 52}$	$1.05^{\dagger 18}$	0.898
		Máx	10.71	0.060	3.66	5.40	4.814
		σ	1.91	0.02	0.77	0.97	1.20
	Sim	Méd	$2.55^{\dagger 16}$	0.177	$0.57^{\dagger 50}$	$1.03^{\dagger 19}$	0.543
		Máx	6.59	0.640	3.55	4.51	2.350
		σ	1.65	0.20	0.70	0.93	0.60
TSPLIB	Não	Méd	$2.46^{\dagger 0}$	0.072	$0.96^{\dagger 5}$	$1.33^{\dagger 0}$	14.466
		Máx	9.93	0.250	4.87	5.54	59.707
		σ	1.99	0.05	1.04	1.20	13.08
	Sim	Méd	$2.43^{\dagger 0}$	2.253	$0.93^{\dagger 7}$	$1.30^{\dagger 0}$	6.858
		Máx	8.50	9.090	3.44	4.72	36.738
		σ	1.86	2.65	0.92	1.12	7.88
kCTP	Não	Méd	$1.27^{\dagger 8}$	0.271	$0.85^{\ddagger 20}$	$0.96^{\ddagger 18}$	18.691
		Máx	6.84	0.830	3.80	4.11	127.92
		σ	1.27	0.23	0.91	0.96	30.25
	Sim	Méd	$1.26^{\dagger 14}$	2.53	$0.82^{\ddagger 25}$	$0.95^{\ddagger 21}$	20.844
		Máx	6.38	8.250	3.54	4.03	168.624
		σ	1.27	2.23	0.87	0.94	32.11

† número de instâncias com Gap = 0 ‡ número de instâncias com Gap $\leq$ 0

Conforme esperado, nossa heurística GRASP é capaz de obter soluções melhores que a heurística HEF usando de um tempo computacional significativamente maior, uma vez que ela é mais robusta. Apesar disso, ela é capaz de retornar soluções com

um gap médio abaixo de 1% em todos os conjuntos de instâncias testados. Essa característica se torna especialmente interessante nas instâncias kCTP, visto que a maior parte delas ainda não possui soluções ótimas. Em 25 das 131 instâncias desse conjunto, nossa heurística GRASP foi capaz de igualar ou ultrapassar a melhor solução obtida pelos algoritmos B&C apresentados aqui.

Do ponto de vista do nosso teste de redução, os resultados em termos de qualidade de solução são pouco claros, já que estamos lidando com métodos sem garantia de otimalidade. Entretanto, esse teste de redução é útil quando o tempo gasto preprocessando o grafo compensa em qualidade geral da solução e tempo total de execução. No caso das instâncias do kCTP, a qualidade das soluções melhoraram uma pequena margem, mas o esforço computacional aumentou devido ao tempo de preprocessamento ineficaz.

Notavelmente, esses experimentos demonstraram mais uma vez a qualidade das soluções dadas pela heurística HEF, com pouco esforço para uma grande variedade de topologias de grafos. Isso também beneficia nossa heurística, promovendo uma boa solução inicial e um mecanismo de refino para as soluções exploradas. Da mesma forma, graças aos nossos métodos exatos, nós temos uma boa visão geral da qualidade das soluções obtidas por todas as heurísticas, com ou sem o teste de redução. Logo, podemos afirmar que nosso GRASP é uma alternativa viável para prover soluções satisfatórias, em um tempo razoavelmente menor que os métodos exponenciais, especialmente nos casos em que tais métodos têm dificuldade.

6.4 Decomposições

Nesta seção apresentamos os resultados decorrentes da aplicação e análise das decomposições propostas neste trabalho. Os resultados obtidos para as técnicas de decomposição permitem que problemas maiores sejam tratados com menos consumo computacional, devido aos novos limites duais e ao tratamento explícito dos subproblemas relaxados.

6.4.1 *Relax-and-Cut*

Visando buscar melhores limites duais e primais para o PFR, realizamos experimentos com a decomposição baseada em Relaxação Lagrangiana proposta na Seção 5.1. Através de testes preliminares em um subconjunto das instâncias, variamos o parâmetro de refino do tamanho de passo $\alpha \in \{0.33, 0.4, 0.5, 0.67, 0.8\}$, e decidimos por fixar o valor em $\alpha = 0.8$. Da mesma forma, o número de máximo de iterações do algoritmo foi fixado em $N = 1000$.

Para a variação de PRL $\mathcal{L}_1$, considere os resultados apresentados na Tabela 6.8.

Tais resultados estão divididos por grupo de instâncias com medidas estatísticas de média, máximo e desvio padrão. Testamos também a variação dos parâmetros de iterações sem melhora do limite, para atualização do tamanho de passo $\Gamma \in \{20, 50\}$, e o número de iterações de permanência de cortes inativos em Δ, $\beta \in \{1, 3\}$. Na Tabela 6.9, temos os resultados para os testes utilizando a variação do algoritmo do Método do Subgradiente adaptado com $\mathcal{L}_2$ como PRL.

Tabela 6.8: Resultados *Relax-and-Cut* com $\mathcal{L}_1$

			$\beta = 1$						$\beta = 3$					
Γ	**Conjunto**		**UB**	**LB**	**Gap**	**T (s)**	**% de $\lvert E\rvert$**	**$\lvert\Delta\rvert$**	**UB**	**LB**	**Gap**	**T (s)**	**% de $\lvert E\rvert$**	**$\lvert\Delta\rvert$**
20	SHRD	Méd	4399.89	4265.43	3.26	0.34	84.60	129.63	4400.91	4260.49	3.39	0.46	84.35	149.05
		Máx	6104.00	6102.62	17.56	1.45	96.61	285.00	6104.00	6101.82	19.06	2.83	96.61	403.00
		σ	940.673	921.084	3.999	0.357	9.516	69.081	942.131	919.842	4.138	0.587	9.644	89.257
	CAPMST	Méd	517.09	499.94	4.00	0.58	92.84	185.95	516.94	499.65	4.04	0.64	92.78	199.44
		Máx	1060.00	1054.90	22.17	1.99	99.18	365.00	1062.00	1054.51	21.72	2.27	99.36	393.00
		σ	258.100	252.103	4.294	0.547	5.073	98.483	258.235	252.273	4.287	0.594	4.928	106.464
	TSPLIB	Méd	24551.29	22786.22	9.94	7.04	80.39	414.69	24566.25	22620.75	9.25	9.61	79.75	481.11
		Máx	91327.00	88962.94	89.06	30.88	95.80	729.00	91186.00	88366.50	30.02	36.43	95.81	903.00
		σ	32094.144	30135.353	11.904	8.207	16.037	141.894	32118.454	29844.281	6.786	11.258	16.842	161.848
	kCTP	Méd	14962.60	13898.38	12.61	34.78	20.49	1120.12	14952.47	14154.05	7.54	35.24	20.47	1222.30
		Máx	30418.00	30354.75	136.67	365.13	86.51	1836.00	30418.00	30349.15	40.28	345.23	88.23	1871.00
		σ	10104.514	9796.160	21.703	59.799	28.368	434.105	10106.299	9804.058	8.002	57.002	28.366	441.053
50	SHRD	Méd	4399.11	4278.54	2.89	0.27	85.46	106.46	4399.69	4274.68	3.01	0.39	85.41	131.20
		Máx	6104.00	6103.00	16.25	0.96	96.73	198.00	6104.00	6101.00	18.29	2.67	96.65	308.00
		σ	941.525	921.574	3.683	0.233	9.161	46.806	939.887	919.364	3.968	0.477	9.362	71.609
	CAPMST	Méd	517.33	500.34	4.46	0.48	93.07	158.87	517.56	500.67	3.86	0.62	92.94	215.24
		Máx	1060.00	1055.74	59.26	1.34	99.36	378.00	1060.00	1055.46	20.44	2.66	99.18	536.00
		σ	258.079	253.038	7.756	0.407	4.717	82.567	258.509	252.405	4.075	0.610	4.727	132.815
	TSPLIB	Méd	24550.88	22873.30	10.59	4.85	80.16	316.43	24538.40	22804.69	8.77	7.61	80.30	456.66
		Máx	91327.00	88925.97	89.06	20.83	95.81	543.00	90977.00	88270.49	30.34	27.37	95.80	773.00
		σ	32101.976	30235.367	13.935	5.364	16.093	103.378	32079.645	30136.607	6.469	8.265	16.043	144.293
	kCTP	Méd	14971.82	13723.00	13.82	23.77	20.71	933.04	14956.89	14172.32	7.16	24.86	20.63	1270.82
		Máx	30418.00	30356.24	136.67	313.42	88.09	1732.00	30418.00	30356.04	30.18	323.76	88.23	1889.00
		σ	10111.380	9688.433	22.286	46.387	28.468	413.342	10106.781	9800.259	7.159	44.407	28.476	443.026

Para ambas as tabelas, 6.8 e 6.9, cada coluna representa: **UB**, o limite primal; **LB**, o limite dual; **Gap**, a diferença relativa entre os limites; **T (s)**, o tempo de execução total do algoritmo; **% of $|E|$**, a proporção de arestas que foram descartadas da solução atráves dos custos duais lagrangianos; e $|\Delta|$, a quantidade de cortes armazenados no *pool* ao final da execução, i.e., cortes ainda ativos.

Como podemos observar nas Tabelas 6.8 e 6.9, como método único para solução do PFR, nosso R&C proposto, embora demande menos computacionalmente do que métodos exatos, não foi capaz de obter uma boa aproximação da solução ótima. Sendo que os melhores gaps de otimalidade vieram da versão $\mathcal{L}_2 \mid \beta = 1 \mid \Gamma = 50$, que também é uma das versões testadas que mais demandam tempo computacional. Isso ocorre, em grande parte, devido à maior complexidade de resolução de $\mathcal{L}_2$ como PRL. Entretanto, com os dados das colunas de variáveis fixadas e número de cortes, surge a possibilidade de se aproveitar essas informações como *warm-start* para um algoritmo B&C. Logo, tomando a melhor configuração de B&C apresentada para solução do PFR como *baseline*, considere os experimentos da Tabela 6.10.

Tomando $\mathcal{P}_3$ `CutL1` como *baseline* do experimento, decidimos aplicar as informações obtidas após o final da execução dos algoritmos R&C propostos em um *framework* B&C. Desta forma, começamos o algoritmo B&C com limites primal

Tabela 6.9: Resultados *Relax-and-Cut* com PRL $\mathcal{L}_2$

			$\beta = 1$						$\beta = 3$					
Γ	**Conjunto**		**UB**	**LB**	**Gap**	**T (s)**	**% de $\|E\|$**	**$\|\Delta\|$**	**UB**	**LB**	**Gap**	**T (s)**	**% de $\|E\|$**	**$\|\Delta\|$**
20	SHRD	Méd	4407.83	4281.67	3.06	0.59	80.94	110.77	4407.11	4262.00	3.46	1.43	79.64	134.05
		Máx	6112.00	6075.80	17.69	2.23	92.80	236.00	6183.00	6042.81	17.50	6.25	92.88	282.00
		σ	944.552	925.796	3.819	0.574	8.420	57.939	946.941	915.488	3.788	1.683	8.460	74.405
	CAPMST	Méd	516.71	499.91	3.61	1.09	91.10	150.55	517.40	498.29	4.05	2.12	90.47	171.21
		Máx	1066.00	1052.00	18.37	3.30	105.00	302.00	1070.00	1050.02	19.44	9.51	105.00	366.00
		σ	259.696	251.120	4.912	0.962	5.943	80.846	259.222	249.449	4.333	1.937	5.741	90.188
	TSPLIB	Méd	24521.20	23255.47	6.10	12.40	78.76	364.00	24565.83	23056.61	7.82	18.42	78.22	434.74
		Máx	90913.00	88960.22	34.66	45.06	95.86	572.00	90987.00	88540.11	34.38	65.77	95.82	673.00
		σ	32053.332	30877.982	5.445	12.027	18.035	111.223	32133.404	30581.643	5.788	14.619	18.355	125.188
	kCTP	Méd	14959.68	14214.20	6.16	61.93	19.81	920.02	14963.53	14295.70	5.30	46.78	19.81	1001.23
		Máx	30418.00	30356.28	59.72	233.20	82.77	1371.00	30418.00	30345.54	24.00	177.91	82.77	1530.00
		σ	10110.835	9794.196	7.549	58.648	28.092	366.798	10112.944	9798.611	4.702	43.585	28.093	391.276
50	SHRD	Méd	4402.07	4312.74	2.04	0.50	83.57	94.47	4407.03	4289.92	2.71	0.82	81.61	118.37
		Máx	6110.00	6091.08	16.75	1.73	94.08	173.00	6183.00	6090.58	15.75	3.34	93.29	276.00
		σ	945.685	916.784	2.733	0.477	6.220	40.720	947.074	912.581	3.217	0.884	7.227	60.835
	CAPMST	Méd	517.40	502.06	3.24	1.03	91.61	141.60	517.22	500.06	3.64	1.75	91.07	205.46
		Máx	1074.00	1053.91	16.12	5.03	105.00	334.00	1070.00	1053.99	18.50	6.07	105.00	515.00
		σ	259.441	251.613	4.128	1.018	5.207	77.609	259.073	250.515	4.215	1.678	5.646	125.961
	TSPLIB	Méd	24502.00	23408.10	5.58	8.67	79.47	322.68	24544.78	23172.41	7.31	15.93	78.50	450.18
		Máx	90913.00	89642.50	33.35	26.29	95.86	575.00	90987.00	89260.15	33.24	55.33	95.83	846.00
		σ	32012.307	31127.946	5.348	7.568	17.623	116.630	32082.087	30772.373	5.586	14.627	18.077	177.931
	kCTP	Méd	14962.24	14284.41	5.48	43.06	19.82	870.27	14966.17	14302.48	5.22	50.32	19.83	1087.76
		Máx	30418.00	30357.30	34.53	152.44	82.79	1367.00	30418.00	30358.72	24.46	168.42	82.79	1711.00
		σ	10113.411	9819.964	5.849	38.924	28.090	360.999	10114.960	9798.597	4.722	43.594	28.089	437.095

e dual válidos, e um conjunto de cortes potencialmente de boa qualidade, dado a natureza do algoritmo R&C. Além disso, o domínio das variáveis criadas é significativamente menor, uma vez que todas as arestas descartadas pelo algoritmo R&C não são populadas no modelo.

Note que como $\mathcal{P}_3$ é um modelo com variáveis direcionadas, precisamos fazer a tradução das desigualdades dualizadas e das variáveis fixadas de $\mathcal{L}_1$ ou $\mathcal{L}_2$ que serão aplicadas à formulação. Tal tradução é automática no caso das variáveis de arestas para arco. Já para o caso das desigualdades SEC e *Flower*, essa relação com as do modelo $\mathcal{P}_2$ foi demonstrada na Seção 3.1. Por fim, uma vez que partimos de um UB potencialmente bom, em um espaço de busca já bastante reduzido e com desigualdades de qualidade, podemos aplicar testes de redução aos custos reduzidos do problema linear do modelo. Tal teste foi apresentado na Proposição 5.1.

Assim sendo, na Tabela 6.10 temos os resultados médios da relaxação linear obtida na raiz do modelo $\mathcal{P}_3$ com algoritmo de separação `CutL1`. Tal *baseline* é incorporado pela etapa de pré-processamento dos algoritmos R&C propostos e tem seus resultados apresentados em todos os conjuntos de instâncias do PFR. Logo, na coluna: **Algoritmo** temos a descrição do algoritmo utilizado e seus parâmetros; O **Conjunto** de instâncias que a linha representa; **LR** representa o valor da relaxação linear média pra cada algoritmo no respectivo conjunto de instâncias; **Gap** representa o gap médio da relaxação linear em relação ao ótimo ou melhor limite primal conhecido para a instância; # 0% **LR** o número de instâncias, no conjunto, em que o valor ótimo foi encontrado ao resolvermos a relaxação linear; **T (s)** o tempo médio de solução da relaxação linear e separação de cortes acrescido do custo do respec-

Tabela 6.10: *Relax-and-Cut* como *warm-start* da relaxação linear de $\mathcal{P}_3$ `CutL1` (média)

Algoritmo	**Conjunto**	**LR**	**Gap**	**# 0% LR**	**T (s)**	**% de $\|A'\|$**	**(3.3)**	**(3.4)**	**(3.6)**
Baseline	SHRD	4372.96	0.08	73	0.13	46.41	99.24	52.99	7.72
	CAPMST	512.25	0.10	75	1.08	79.21	532.05	82.70	27.63
	TSPLIB	24193.87	0.14	14	5.16	76.55	1051.32	236.68	78.29
	kCTP	14745.46	0.45	9	122.51	17.80	7971.84	1101.66	532.28
$\mathcal{L}_1 \mid \beta = 1 \mid \Gamma = 20$	SHRD	4373.35	0.07	76	0.67	91.96	132.65	63.69	7.88
	CAPMST	512.28	0.09	75	1.94	93.24	363.73	130.08	14.87
	TSPLIB	24194.54	0.14	16	18.96	91.70	1006.60	295.42	39.08
	kCTP	14743.60	0.47	9	101.10	21.24	3014.29	944.11	112.79
$\mathcal{L}_1 \mid \beta = 1 \mid \Gamma = 50$	SHRD	4373.83	0.06	77	0.60	92.21	108.59	64.57	7.87
	CAPMST	512.27	0.09	76	1.62	93.19	336.28	131.77	13.71
	TSPLIB	24193.80	0.15	17	15.89	91.50	923.62	298.11	39.75
	kCTP	14744.34	0.46	8	84.75	21.30	2794.08	938.15	108.29
$\mathcal{L}_1 \mid \beta = 3 \mid \Gamma = 20$	SHRD	4373.48	0.07	77	0.87	92.00	151.53	63.39	7.84
	CAPMST	512.26	0.10	75	1.76	93.39	380.46	129.13	14.09
	TSPLIB	24192.80	0.14	16	20.82	91.40	1068.37	291.78	39.69
	kCTP	14743.00	0.47	9	100.77	21.23	3088.47	934.18	113.75
$\mathcal{L}_1 \mid \beta = 3 \mid \Gamma = 50$	SHRD	4373.65	0.06	77	0.85	92.09	133.63	63.81	8.22
	CAPMST	512.25	0.10	75	2.01	93.06	394.62	129.50	14.93
	TSPLIB	24195.51	0.14	13	20.71	91.76	1029.92	291.48	39.57
	kCTP	14743.69	0.47	8	106.71	21.22	3148.33	931.37	111.39
$\mathcal{L}_2 \mid \beta = 1 \mid \Gamma = 20$	SHRD	4373.64	0.06	77	0.68	91.40	103.41	63.45	8.19
	CAPMST	512.30	0.09	76	2.08	92.77	310.66	127.35	14.19
	TSPLIB	24194.56	0.14	15	17.16	91.96	908.29	294.25	36.48
	kCTP	14745.96	0.45	9	97.14	21.19	2775.46	953.50	108.32
$\mathcal{L}_2 \mid \beta = 1 \mid \Gamma = 50$	SHRD	4373.37	0.07	75	0.59	91.92	85.11	62.76	7.89
	CAPMST	512.26	0.10	75	1.88	92.75	299.95	128.26	14.14
	TSPLIB	24195.38	0.14	17	13.07	92.34	857.03	290.55	38.65
	kCTP	14745.87	0.44	9	72.79	21.38	2715.34	951.42	107.75
$\mathcal{L}_2 \mid \beta = 3 \mid \Gamma = 20$	SHRD	4373.49	0.07	77	1.60	91.47	130.56	63.71	7.44
	CAPMST	512.26	0.10	75	3.39	92.67	330.50	127.26	13.94
	TSPLIB	24194.44	0.14	14	24.34	91.41	969.37	289.97	40.02
	kCTP	14745.39	0.45	9	72.64	21.34	2818.17	957.82	107.86
$\mathcal{L}_2 \mid \beta = 3 \mid \Gamma = 50$	SHRD	4373.46	0.06	78	0.92	91.55	111.83	63.21	7.73
	CAPMST	512.27	0.10	76	2.64	92.70	356.47	126.62	13.82
	TSPLIB	24194.95	0.15	15	20.75	91.56	974.31	289.55	38.77
	kCTP	14745.64	0.45	9	89.34	21.17	2920.13	956.68	105.88

tivo algoritmo R&C, se houver; e **% de $|A'|$** representa a proporção, em média, das variáveis de $\mathcal{P}_3$ que foram descartadas da solução. Por fim, as últimas três colunas representam a quantidade de cada tipo de corte adicionado.

Em um escopo geral, o uso como *warm-start* do R&C nos diferentes parâmetros testados foi positivo pra performance do *baseline* aplicado. Especialmente na quantidade de variáveis descartadas nos três primeiros conjuntos de intâncias, levando a um aumento do número de instâncias resolvidas otimamente na raiz. Em tais casos, entretanto, o desempenho computacional ficou abaixo devido ao custo associado do algoritmo R&C. Ressaltamos que isso pode ser benéfico nos casos em que a exploração da etapa de *branching* seja necessária. Para o caso das instâncias kCTP, que são as mais difíceis para o algoritmo resolver, a melhoria de performance foi evidente. Nessas instâncias, nas melhores configurações de $\mathcal{L}_1$ e $\mathcal{L}_2$, o tempo médio gasto reduziu de 122.51 para abaixo dos 90 segundos. Além disso, chamamos a atenção para o número médio de desigualdades adicionadas, que foi bastante reduzida em

relação ao *baseline*. Isso demonstra a qualidade das desigualdades dualizadas que foram separadas pelo algoritmo R&C.

Tabela 6.11: Comparação entre performance média dos algoritmos B&C em subgrupo de instâncias

	Raiz			**B&C**				
Algoritmo	**Gap**	**T (s)**	**% de $\|A'\|$**	**Gap**	**# 0% Gap**	**T (s)**	**# de nós**	**# de cortes**
$\mathcal{P}_2$	2.26	0.9676	-	0.1187	77	857.10	920	13865
Baseline	0.44	9.2164	85.1161	0.0090	89	183.08	1105	1271
$\mathcal{L}_1 \mid \beta = 1 \mid \Gamma = 50$	0.47	8.7002	91.8096	0.0049	89	173.24	1121	3408
$\mathcal{L}_2 \mid \beta = 3 \mid \Gamma = 50$	0.44	9.6727	92.8875	0.0045	90	160.08	1063	3597

De modo a avaliar a performance completa do B&C com *warm-start*, comparamos as melhores configurações de parâmetros de cada R&C com seu *baseline* e com o atual estado-da-arte em métodos exatos da literatura. Para tais testes, selecionamos um subconjunto de 91 instâncias onde os algoritmos testados na Seção 6.2 tivessem dificuldade em termos de custo computacional e relaxação linear. O motivo para isso é que em instâncias mais fáceis, só o custo do R&C torna seu uso proibitivo em relação ao B&C. Desta forma, considere o resultado médio desses testes compilado na Tabela 6.11.

Na Tabela 6.11, comparamos os seguintes cenários: $\mathcal{P}_2$ sem etapa pré-processamento, i.e., sem fixação de variáveis; o *Baseline* é o nosso algoritmo $\mathcal{P}_3$ `CutL1` com a etapa de pré-processamento descrita no Capítulo 2; e as duas configurações de R&C, que têm a etapa de pré-processamento proposta, e a etapa de fixação de variáveis pela heurística lagrangiana.

Como podemos observar, mesmo nas instâncias selecionadas, $\mathcal{P}_3$ `CutL1` consegue desempenhar bem, só falhando em encontrar o ótimo em duas instâncias. De toda forma, o uso das abordagens R&C foi capaz de melhorar o *gap* médio nas instâncias onde o ótimo não pôde ser provado, bem como diminuir o tempo de processamento médio. No caso do algoritmo $\mathcal{L}_2 \mid \beta = 3 \mid \Gamma = 50$, somente uma instância não teve seu ótimo provado, restando um *gap* de otimalidade de 0.41%.

As Figuras 6.6, 6.7 e 6.8 apresentam um estudo na tentativa de entender melhor o impacto dos parâmetros N e Γ na convergência de R&C com $\mathcal{L}_2 \mid \beta = 3$. Tal estudo se faz necessário na tentativa de diminuir tais parâmetros, e consequentemente, acelerar a convergência do algoritmo de subgradiente sem comprometer os resultados. Tal estudo é particularmente complicado, uma vez que algoritmos lagrangianos tendem a ser de difícil convergência e bem sensíveis aos parâmetros estudados. Logo, pegamos três instâncias de conjuntos diferentes com alguns valores de k e monitoramos os dados de limite dual e primal, *gap* e tamanho do passo. Tais experimentos foram feitos com $N = 250$ e $\Gamma = 5$, em uma tentativa de monitorar melhor a convergência antecipada do algoritmo.

Por esses dados das Figuras 6.6, 6.7 e 6.8, podemos observar algumas tendências

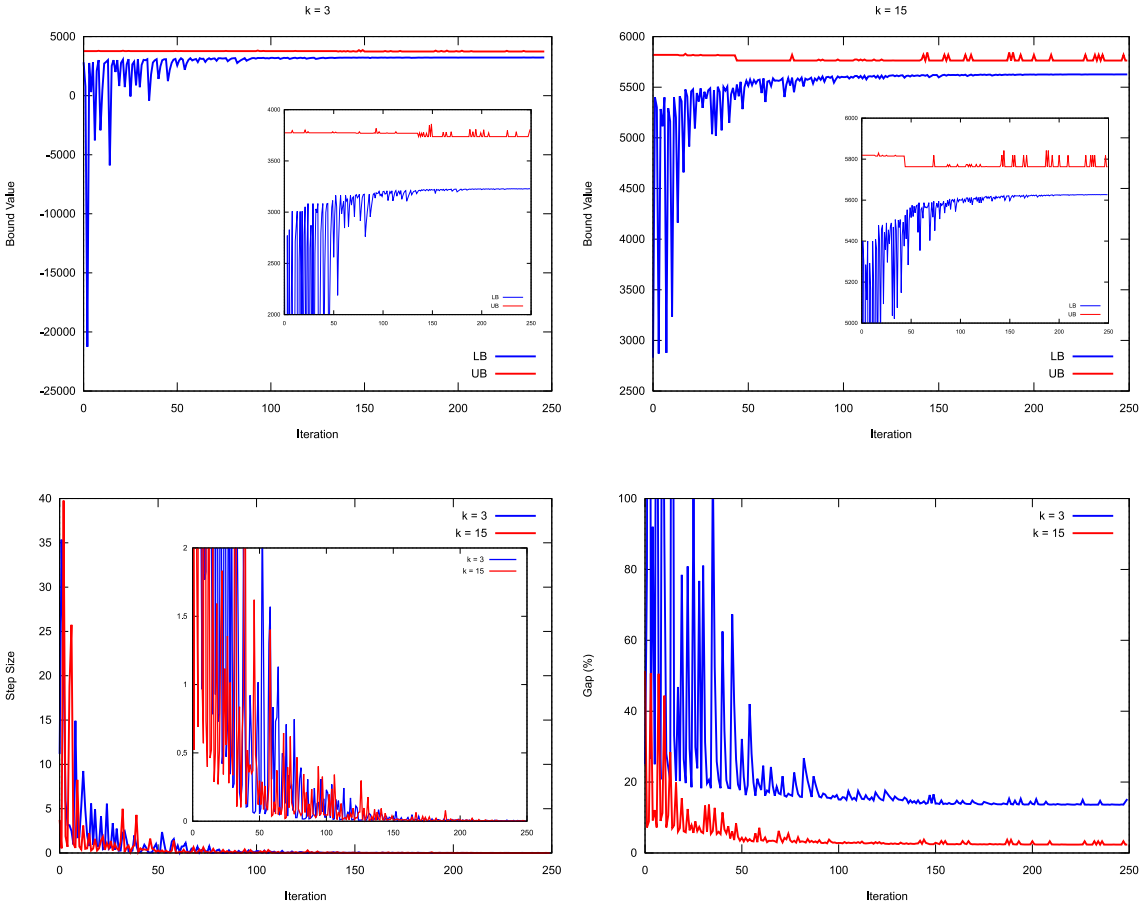

Figura 6.6: $\mathcal{L}_2 \mid \beta = 3 \mid \Gamma = 5 \mid N = 250$ (crd100)

da nossa implementação. Como é de praxe no MS para resolução de problemas de relaxação lagrangiana, os limites duais podem sempre oscilar entre iterações, especialmente nas primeiras iterações do método. Para a instância `le450_15a`, o gap estabilizou no seu menor valor possível próximo de 150 iterações, o que também coincide com o tamanho do passo se aproximando de zero. Para as instâncias `crd100` e `lin318` isso aconteceu em torno de 100 iterações. Esse comportamento se dá independente do valor de k, mas depende claramente da instância e dos parâmetros escolhidos pro MS. Logo, uma parametrização mais robusta é necessária para uma melhor eficácia do método R&C como *solver* independente em instâncias mais difíceis, ou como etapa de pré-processamento para um algoritmo B&C.

No caso de tal parametrização, ela deve ser feita de acordo com o objetivo, uma vez que melhores limites duais lagrangianos não necessariamente se traduzem em melhores cortes, ou fixação de variáveis que sirvam de bom começo para um algoritmo B&C. No limite, dado a convergência numérica ideal, o R&C deveria ser capaz de produzir bons resultados para ambos os objetivos. Entretanto, isso é impossível em termos práticos, dado a instabilidade do método. Logo, a primeira limitação que se deve ter em mente é em relação ao número máximo de iterações

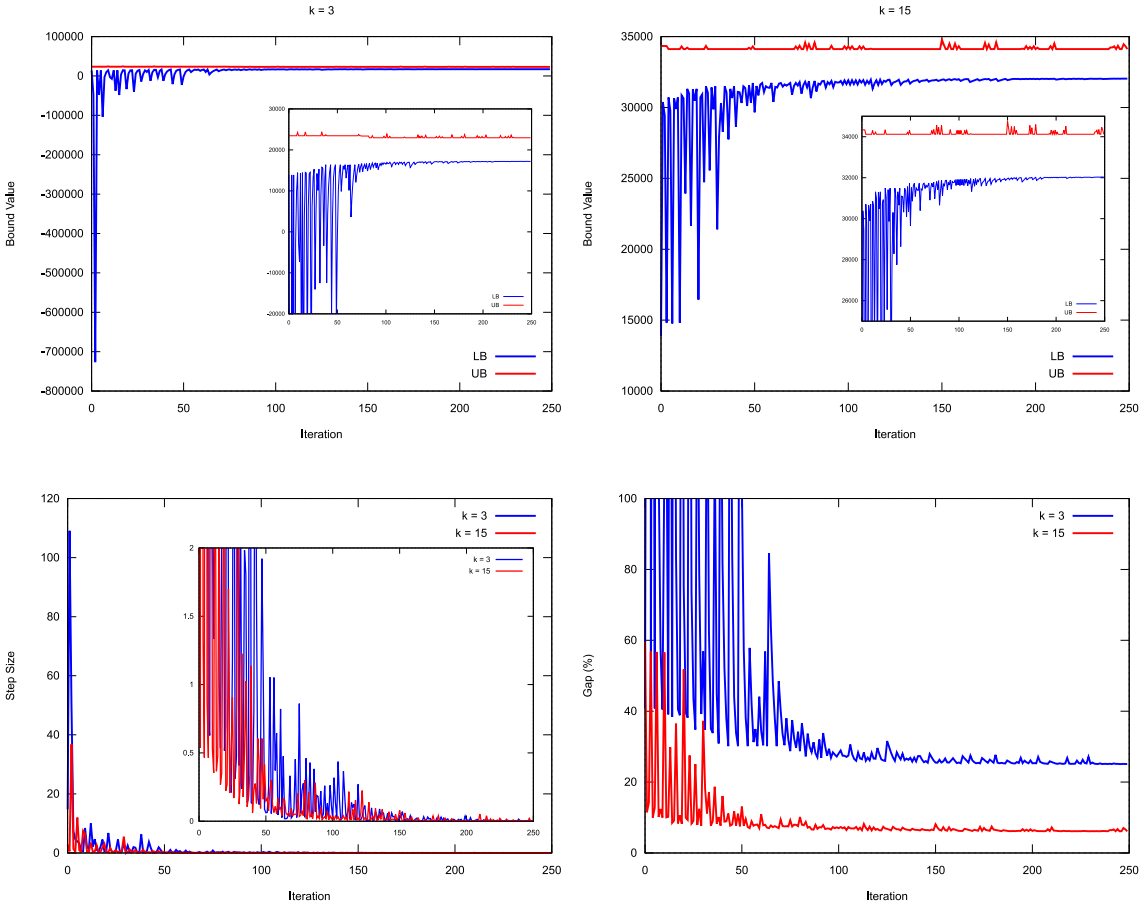

Figura 6.7: $\mathcal{L}_2 \mid \beta = 3 \mid \Gamma = 5 \mid N = 250$ (lin318)

que se pretende executar. No caso dos nossos experimentos das Tabelas 6.8 até 6.11, conseguimos demonstrar que o uso do R&C traz resultados ao B&C, mas deve ser utilizado tal que seu custo não ultrapasse um certo limiar aceitável e com foco em instâncias ainda não resolvidas.

6.4.2 Geração de Colunas

Conforme vimos na Seção 6.2, o *gap* médio da relaxação linear nas instâncias testadas pelas nossas formulações costuma ser bastante próximo ao valor ótimo. Isso, por si só, já indica uma certa aversão ao uso de geração de colunas para estas instâncias, uma vez que tal técnica tem um custo extra associado à resolução do problema de *pricing*. Desta forma, filtramos algumas instâncias em que a relaxação linear do nosso modelo, com as separações heurísticas, não atingisse o valor ótimo. Também vale notar que resolveremos o subproblema utilizando *Branch-and-Cut*, logo tivemos que limitar os testes a instâncias de até 30 vértices.

Como primeiro experimento, tentamos avaliar a qualidade dos limites duais gerados pela nossa formulação $\mathcal{D}_1$ frente ao limite dual ótimo de $\mathcal{P}_3$, utilizando separação

exata. Também experimentamos a utilização do nosso algoritmo ACS na geração de colunas iniciais de modo a popular as colunas da formulação. Tal *warm-start* tende a acelerar a convergência da relaxação linear da geração de colunas ao diminuir o número de iterações necessárias. Consequentemente, o número de vezes em que será necessário resolver o problema de *pricing* será menor. Os resultados de tais experimentos são apresentados na Tabela 6.12. Ademais, é válido mencionar que somente utilizamos o algoritmo de Planos de Corte no nosso algoritmo quando não é mais possível gerar uma nova coluna com custo reduzido negativo, i.e., a solução ótima do subproblema é maior ou igual a zero.

Tabela 6.12: Geração de Colunas - limite & tempo computacional

			$\mathcal{P}_3$ **Exato**			$\mathcal{D}_1$			**ACS** $\mathcal{D}_1$		
Instância	k	**OPT**	**LR**	**T (s)**	**Gap (%)**	**LR**	**T (s)**	**Gap (%)**	**LR**	**T (s)**	**Gap (%)**
bays29	8	1434	1432.50	0.08	0.10	1432.50	323.32	0.10	1432.50	38.05	0.10
crd300	3	2462	2462.00	0.00	0.00	2462.00	15.38	0.00	2462.00	1.75	0.00
	4	2779	2779.00	0.00	0.00	2779.00	39.82	0.00	2779.00	3.52	0.00
	5	3024	3024.00	0.00	0.00	3024.00	120.09	0.00	3024.00	7.10	0.00
	6	3123	3112.00	0.29	0.35	3112.00	161.32	0.35	3112.00	6.82	0.35
	7	3164	3164.00	0.00	0.00	3164.00	176.85	0.00	3164.00	21.53	0.00
	8	3293	3293.00	0.00	0.00	3293.00	147.90	0.00	3293.00	22.55	0.00
	9	3293	3293.00	0.00	0.00	3293.00	212.70	0.00	3293.00	25.22	0.00
	10	3465	3429.67	0.19	1.02	3440.67	212.23	0.70	3440.67	45.65	0.70
crd301	5	2617	2601.00	0.18	0.61	2601.00	76.89	0.61	2601.00	7.42	0.61
	9	2923	2921.00	0.09	0.07	2923.00	207.97	0.00	2923.00	79.98	0.00
	10	2972	2936.33	0.15	1.20	2942.50	259.33	0.99	2942.50	44.03	0.99
crd302	5	3261	3240.67	0.23	0.62	3259.00	84.73	0.06	3259.00	8.60	0.06
	6	3438	3423.50	0.53	0.42	3434.50	114.22	0.10	3434.50	18.29	0.10
	10	3779	3695.00	0.32	2.22	3746.20	189.35	0.87	3746.20	49.78	0.87
crd303	3	2744	2736.00	0.12	0.29	2744.00	49.21	0.00	2744.00	10.80	0.00
	4	3040	3008.20	0.51	1.05	3036.00	139.44	0.13	3036.00	7.63	0.13
	9	3578	3565.00	0.34	0.36	3578.00	497.99	0.00	3578.00	95.86	0.00
	10	3645	3554.50	0.15	2.48	3584.00	512.02	1.67	3584.00	53.25	1.67
crd304	5	3131	3128.00	0.14	0.10	3128.00	86.60	0.10	3128.00	14.40	0.10
	10	3736	3669.33	0.36	1.78	3681.67	342.71	1.45	3681.67	61.04	1.45
crd305	5	3009	3008.00	0.09	0.03	3008.00	51.06	0.03	3008.00	7.54	0.03
	9	3531	3494.17	0.41	1.04	3507.00	241.45	0.68	3507.00	16.96	0.68
	10	3533	3516.50	0.18	0.47	3533.00	234.68	0.00	3533.00	43.25	0.00
crd306	10	3615	3592.33	0.35	0.63	3603.25	384.62	0.33	3603.25	59.11	0.33
crd307	4	3054	3012.00	0.08	1.38	3012.00	46.16	1.38	3012.00	5.91	1.38
crd308	4	2730	2728.00	0.14	0.07	2728.00	30.07	0.07	2728.00	3.01	0.07
	6	2988	2987.50	0.26	0.02	2988.00	55.46	0.00	2988.00	6.67	0.00
	9	3389	3373.33	0.17	0.46	3389.00	122.69	0.00	3389.00	25.72	0.00
	10	3428	3404.50	0.15	0.69	3421.00	187.76	0.20	3421.00	40.39	0.20
crd309	3	2246	2225.00	0.24	0.93	2225.00	8.67	0.93	2225.00	4.16	0.93
	4	2764	2764.00	0.00	0.00	2764.00	42.62	0.00	2764.00	8.59	0.00
	5	3070	3070.00	0.00	0.00	3070.00	174.51	0.00	3070.00	10.05	0.00
	6	3144	3141.33	0.88	0.08	3144.00	171.52	0.00	3144.00	5.03	0.00
	7	3203	3203.00	0.00	0.00	3203.00	172.00	0.00	3203.00	14.02	0.00
	8	3350	3350.00	0.00	0.00	3350.00	198.51	0.00	3350.00	30.64	0.00
	9	3413	3413.00	0.00	0.00	3413.00	305.36	0.00	3413.00	22.57	0.00
	10	3470	3449.80	0.56	0.58	3470.00	260.45	0.00	3470.00	41.40	0.00
		Méd	3110.53	0.19	0.50	3118.06	175.20	0.28	3118.06	25.48	0.28
		σ	442.201	0.198	0.639	447.842	123.621	0.466	447.842	23.048	0.466

Conforme pudemos observar da Tabela 6.12, nosso algoritmo de Geração de Colunas foi capaz de encontrar melhores limites duais nas instâncias testadas, em relação ao ótimo da relaxação linear de $\mathcal{P}_3$. O que levou nosso algoritmo de GC a

baixar o *gap* médio nessas instâncias de 0.5% para 0.28%. Se considerarmos apenas as instâncias em que o *gap* era originalmente maior que zero, esse valor passa de 0.71% em $\mathcal{P}_3$ para 0.39% em $\mathcal{D}_1$, com limite dual ótimo em oito novas instâncias. Por outro lado, como já previsto, o custo computacional é significativamente maior, devido à complexidade do problema de *pricing* formulado. Ademais, a versão com *warm-start* do algoritmo ACS, foi capaz de reduzir o tempo de processamento em média sete vezes.

Na Tabela 6.13, fizemos um estudo acerca das diferentes configurações de cortes e algoritmos de separação propostos para resolver o problema de *pricing*, $\mathcal{S}_1$. Para isto, considere as colunas, onde: # **it** representa o número de iterações do algoritmo de GC, e consequentemente o número de problemas *pricing* resolvidos; # **Cuts** representa o número de desigualdades (5.11) adicionadas sob demanda ao modelo $\mathcal{D}_1$; **CP T (s)** representa o tempo gasto para identificação de violações dessas desigualdades; e **LR T (s)** apresenta o tempo gasto para resolução da relaxação linear do subproblema $\mathcal{S}_1$. As versões testadas foram cinco, com a formulação $\mathcal{S}_1$ contemplando as desigualdades (5.15)-(5.18) e acrescida de:

- **(5.19)** - Nesse caso, somente as desigualdades necessárias para a correta modelagem do problema de arborescência enraízada de custo mínimo com número mínimo de vértices. Para resolução do problema de separação de tal formulação, é suficiente utilizar o algoritmo de corte mínimo/fluxo máximo no grafo residual.

- **(5.19)-(5.20)** - Para separação de (5.19) usamos o algoritmo de corte mínimo. Para identificação das desigualdades (5.20) violadas no grafo residual, utilizamos o mesmo conjunto de heurísticas apresentadas em SIMONETTI *et al.* [53].

- **(5.19) & (5.21)** - Para resolução desse problema de separação, além do algoritmo de corte mínimo, utilizamos o algoritmo proposto por GOUVEIA e SALAZAR-GONZÁLEZ [26] para a separação dos conjuntos violados de desigualdades ERMS.

- **(5.19)-(5.21)** - União das duas versões anteriores.

- **(5.19)-(5.21) w/ BFS** - Nesse caso, utilizamos um algoritmo de separação menos robusto, optando por abrir mão da otimalidade de violação em troca de maior rapidez na resolução do problema de separação. Nesta versão, em todos os nós do algoritmo de B&C, utilizamos somente de uma heurística BFS com arrendondamento dos custos residuais para identificação de componentes conexas. Para cada uma dessas componentes conexas, nós tentamos encontrar violações das desigualdades por inspeção.

Tabela 6.13: Impactos na Geração de Colunas em diferentes versões do subproblema

		ACS $\mathcal{D}_1$				$\mathcal{S}_1$	
Versão de $\mathcal{S}_1$		**T (s)**	**# it**	**# Cuts**	**CP T (s)**	**T (s)**	**LR T (s)**
(5.19)	**Méd**	17.53	22.29	3.66	0.16	16.67	6.98
	Máx	44.99	102	16	0.67	43.95	24.91
	σ	10.258	18.985	4.276	0.149	10.199	5.211
(5.19)-(5.20)	**Méd**	11.91	33.84	3.66	0.18	11.01	9.33
	Máx	36.18	161	16	0.78	35.16	31.87
	σ	9.649	31.905	4.276	0.187	9.625	8.473
(5.19) & (5.21)	**Méd**	15.76	21.47	3.66	0.16	14.89	5.56
	Máx	35.42	85	16	0.76	34.54	18.60
	σ	9.274	16.841	4.276	0.155	9.233	4.592
(5.19)-(5.21)	**Méd**	12.94	33.50	3.66	0.16	12.04	10.23
	Máx	48.88	156	16	0.71	47.73	43.87
	σ	11.236	31.883	4.276	0.152	11.185	9.932
(5.19)-(5.21) w/ BFS	**Méd**	12.07	32.71	3.66	0.16	11.17	9.49
	Máx	42.11	136	16	0.69	41.32	39.74
	σ	9.617	28.742	4.276	0.150	9.596	8.812

À primeira vista, os resultados da Tabela 6.13, sugerem que usar somente as desigualdades (5.19) é bem desvantajoso para o algoritmo de *Branch-and-Cut*. Mais uma vez, isso é esperado considerando que muitas simetrias ainda permanecem no modelo. Da mesma forma, a adição das desigualdades ERMS traz uma pequena melhoria, entretanto, nada tão significativo quanto na versão utilizando as desigualdades (5.19)-(5.20). Tal esquema de separação aumenta o tempo de solução da relaxação linear de $\mathcal{S}_1$, mas isso acaba por recompensar o algoritmo de B&C. Por fim, tentamos experimentar o uso de todos os cortes, o que à primeira vista não traz nenhum ganho em relação à versão anterior, tanto com uma heurística de separação robusta ou *lazy* baseada em BFS.

Para finalizar os experimentos e tentar uma conclusão mais apropriada sobre qual versão de $\mathcal{S}_1$ empregar, a Figura 6.9 apresenta os resultados dos algoritmos de B&C na resolução do subproblema de arborescência mínima enraizada com cardinalidade mínima. Tais resultados foram obtidos resolvendo otimamente todas as instâncias geradas por meio dos custos duais durante a etapa de resolução de $\mathcal{D}_1$. Para isso, os dados apresentados para cada versão do algoritmo envolvem: tempo de processamento, número de nós da árvore de *branching*, número de cortes sob demanda aplicados, valor da relaxação linear, e tempo para resolução da relaxação linear. Vale reforçar que todas as instâncias foram resolvidas até a otimalidade por todas as versões propostas.

Conforme podemos observar pela Figura 6.9, das versões apresentadas, **(5.19)-(5.21) w/ BFS** é a que obtém os melhores limites duais com o menor número de cortes aplicados e necessitando explorar menos nós da árvore de *branch-and-bound*. Isso enquanto possui o menor tempo de processamento para encontrar o ótimo e a

relaxação linear no nó raiz. Vale notar que, como essas instâncias foram geradas utilizando custos duais do problema mestre $\mathcal{D}_1$, o valor ótimo médio das instâncias é de -11.88. Portanto, um gráfico *boxplot* com valor da relaxação linear mais próximo desse valor é o ideal.

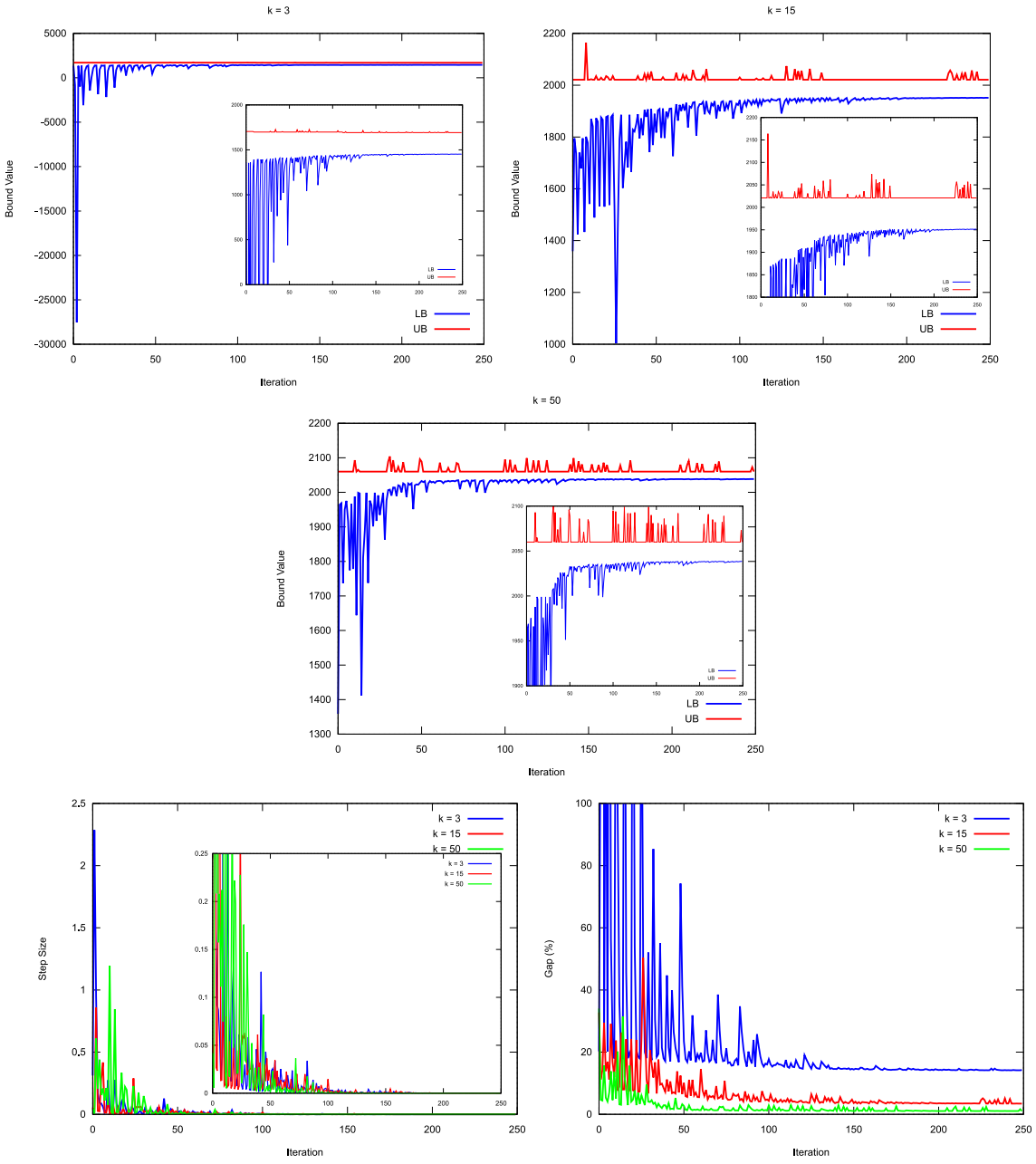

Figura 6.8: $\mathcal{L}_2 \mid \beta = 3 \mid \Gamma = 5 \mid N = 250$ (le450_15a)

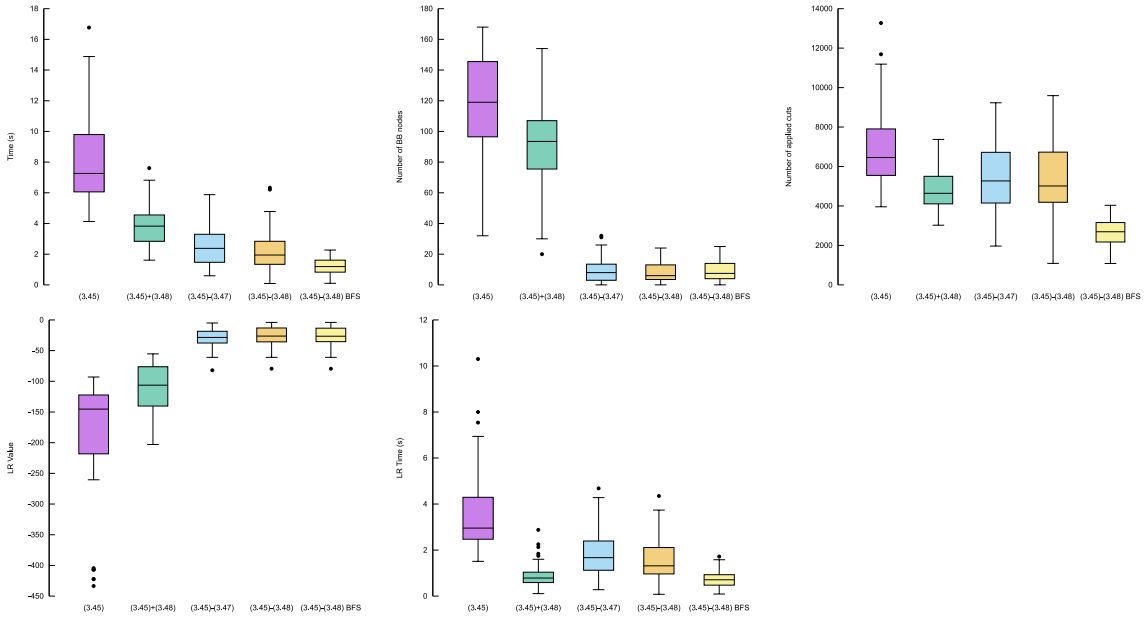

Figura 6.9: $\mathcal{S}_1$ - resultados do B&C nas instâncias de pricing nas diferentes versões

Capítulo 7

Considerações Finais

Este trabalho abordou o Problema da Floresta Geradora Mínima Restrita (PFR) sob diversos aspectos e metodologias de Otimização Combinatória. Começamos apresentado uma revisão da literatura contendo todas as abordagens atuais que resolvem o problema, bem como apresentando problemas relacionados e aplicações práticas do problema em questão.

Em termos de contribuições teóricas para a literatura do PFR, apresentamos um estudo de dominância relacionando os dois modelos existentes, e através desse estudo, concluímos o estado-da-arte para o problema. Tal estudo levou à formulação de um teste de redução para instâncias do problema, útil como pré-processamento, para se reduzir o espaço de busca. Tal teste tem grande influência em grafos de entrada altamente simétricos ou densos. Os métodos propostos nesse trabalho, bem como os da literatura, se beneficiaram de tal pré-processamento.

No escopo prático, propusemos quatro conjuntos de instâncias padronizadas baseadas em instâncias de problemas clássicos ou relacionados, que são facilmente encontradas na *OR-Library*. Até o presente trabalho, as instâncias de testes eram geradas aleatoriamente por cada artigo na literatura, e nenhuma delas está disponível. Tal padronização facilita o teste e a comparação das técnicas que propomos com as da literatura, seja ela existente ou com futuras abordagens, especialmente no caso de heurísticas. Além disso, fizemos um estudo considerando o impacto do parâmetro de entrada k nos custos da função objetivo e na dificuldade de solução. Tal estudo sugere que uma instância é geralmente mais difícil de ser resolvida para valores de $k \leq \lfloor \frac{n}{4} \rfloor$, uma vez que o número de soluções viáveis tende a ser maior.

Em termos de métodos exatos, propusemos três novas formulações de programação linear inteira. A primeira provou ser experimentalmente capaz de superar o atual estado-da-arte, que foi demonstrado através do estudo de dominância. Tal modelo é uma reformulação do problema e utiliza de variáveis binárias em um grafo direcionado, tal que generaliza o PFR para problemas com custos assimétricos. Para esse modelo, foram propostos novos conjuntos de desigualdades exponenciais, que

reduzem as simetrias de sua relaxação linear, tornando a solução do problema de separação por heurísticas mais efetiva.

Também propusemos uma formulação baseada na enumeração explícita das árvores que compõe a solução do PFR. Entretanto, tal modelo não se mostrou útil para instâncias com número de vértices maiores do que 100. Tal fenômeno pode ter sido influenciado pela alta explosão combinatória das variáveis, o que acarreta em uma relaxação linear fraca e muitas simetrias a serem quebradas via enumeração. A última formulação de programação linear inteira que propusemos foi baseada em capacidade de fluxo por subárvore, chamada de q-arborescência. Tal formulação é baseada na formulação clássica de GOUVEIA [25] para o Problema de Roteamento de Veículos, mas quando utilizada no método *Branch-and-Cut* também não se provou efetiva devido ao enorme número de variáveis e simetrias.

Em relação às desigualdades exponenciais presentes na literatura e nas nossas formulações, algumas dessas não possuem algoritmo de separação em tempo polinomial conhecido. Portanto, dedicamos um tempo desse trabalho em propor e explorar algoritmos que pudessem achar violações de boa qualidade para essas desigualdades. Tal estudo melhorou significativamente a performance dos algoritmos baseados nas formulações que exploramos.

Visando buscar novos limites duais, apresentamos duas decomposições lagrangianas para o PFR. A exploração das formulações lagrangianas não conseguiram ter uma boa performance como abordagem única para resolver o PFR através do Método do Subgradiente. Entretanto, tais reformulações demonstraram potencial quando utilizadas como *warm-start* do nosso algoritmo de B&C, inclusive sendo capaz de obter melhores soluções em menor tempo para instâncias mais difíceis. Entretanto, por ser um método númerico bastante instável e sensível aos parâmetros definidos, um estudo paramétrico mais robusto pode torná-lo significativamente melhor.

Outras duas reformulações apresentadas foram baseadas nas decomposições de DANTZIG e WOLFE [17], onde uma delas foi fundamentada na formulação q-arborescência, e não foi explorada experimentalmente neste trabalho. A outra reformulação foi baseada no nosso modelo direcionado, onde cada coluna do problema principal representa uma arborescência enraizada de custo mínimo contendo pelo menos k vértices. Tal subproblema ainda é $\mathcal{NP}$-Difícil, e propusemos uma formulação de programação linear inteira para resolvê-lo. Pelo nosso conhecimento, essa foi a primeira vez que tal problema foi estudado desta forma. Além disso, utilizamos de uma heurística baseada em colônia de formigas para popular inicialmente as colunas do algoritmo, de modo a tentar realizar menos iterações e consequentemente ser necessário recorrer menos vezes à resolução do subproblema. Devido ao elevado custo computacional, os resultados práticos foram obtidos apenas em um subconjunto de

instâncias com até 30 vértices, e provaram que nossa reformulação é capaz de obter limites duais melhores para o PFR.

Para heurísticas primais, implementamos uma baseada em Colônia de Formigas. Essa heurística foi capaz de obter bons resultados frente à heurística HEF, embora seu custo computacional seja significativamente maior, devido à necessidade de lidar com várias soluções a cada iteração. Por esse motivo, ela teve seu uso melhor aproveitado como população inicial para o algoritmo de geração de colunas. Também apresentamos uma heurística baseada em GRASP, cujo custo computacional é significativamente menor. Tal heurística também superou a heurística HEF e provou ser capaz de obter boas soluções em um tempo significativamente menor que as abordagens exatas, em alguns casos até melhorando o melhor limite primal de instâncias com ótimo ainda não provado.

É digno de nota que experimentamos outras abordagens não listadas neste trabalho como forma de solucionar o PFR. Tais abordagens não foram mencionadas antes pois seus resultados foram inferiores aos demais apresentados. A primeira dessas abordagens envolveu o uso de heurísticas *Dual Ascent*, baseadas em $\mathcal{P}_1$ e $\mathcal{P}_2$, dentro de um *framework* de *Branch-and-Bound* (B&B) combinatório. Outras tentativas com B&B combinatório envolveram o uso de problemas relaxados do PFR como método de *bounding*, tais como: problema da floresta geradora mínima com número mínimo de arestas; e problema de cobertura em floresta. Embora tais limites duais sejam válidos e polinomialmente resolvíveis, eles não se provaram próximos o suficiente do ótimo para evitar uma explosão combinatória da árvore de *branching*, e portanto levar a uma solução ótima mais rápido.

Existe bastante margem para exploração de novos horizontes da nossa pesquisa, especialmente em instâncias maiores e mais difíceis. Podendo começar pelos casos em que o algoritmo de planos de corte demanda bastante tempo para resolver a relaxação linear. Para esses casos, imaginamos que uma melhor parametrização dos nossos algoritmos R&C será capaz de trazer bastante benefícios, uma vez que pode reduzir significativamente o espaço de busca e iniciar a relaxação linear com desigualdades de boa qualidade. Outras opções de melhoria do R&C envolvem diferentes técnicas para a solução do problema dos multiplicadores lagrangianos, como o Algoritmo de Volume [7] ou até mesmo considerar outros vetores direção e métodos de atualização de passo do MS. O uso de novas heurísticas primais em conjunto com o R&C também pode ser interessante, uma vez que pode levar a mais fixações de variáveis.

Em relação à abordagem de Geração de Colunas para instâncias maiores, vemos bastante campo de estudo. Especialmente se o problema for reformulado de tal forma que seu subproblema possua complexidade resolvida em tempo polinomial ou pseudo-polinomial. Além disso, também vemos margem para exploração no subpro-

blema que propusemos através de nossa reformulação, o problema de arborescência enraizada com número mínimo de vértices. Tal subproblema herda características e aplicações semelhantes ao PFR, e pode ser facilmente explorado da mesma forma, enquanto se beneficia de características de outros problemas cuja solução não precisa ser geradora.

No campo heurístico, nossas abordagens metaheurísticas foram bastante limitadas na sua aplicação imediata, em conjunto com as demais metodologias propostas. Portanto, acreditamos que uma metaheurística mais robusta e com um devido estudo mais amplo de vizinhanças e paramêtros pode ser bastante útil a partir de instâncias que nosso B&C começa a sofrer computacionalmente. Uma abordagem heurística visando uma melhora na qualidade da solução é o uso da Reconexão de Caminhos [49] como mecanismo de pós-otimização. A extensão do critério de aceitação no *pool* de soluções abaixo do grau de similaridade para soluções com bom custo também pode apresentar vantagens. Outra opção envolve a implementação de heurísticas focadas em busca local extensiva, como *Iterated Local Search* [38] ou *Adaptive Large Neighborhood Search* [50]. Desta forma, utilizando as vizinhanças aqui propostas e outras novas que sejam mais robustas. O uso de uma heurística estocástica destrutiva baseada na heurística HEF também surge como alternativa aos métodos construtivos propostos.

Finalmente, acreditamos que as metodologias apresentadas podem ser expandidas e aplicadas com êxito a problemas que compartilham bastante semelhanças com o PFR. Principalmente a heurística primal GRASP, o B&C em sua versão direcionada, as heurísticas de separação, e os algoritmos R&C condicionados a uma melhor parametrização. Alguns destes problemas relacionados foram citados na introdução deste trabalho.

Referências Bibliográficas

[1] ADASME, P., 2019, "Optimal sub-tree scheduling for wireless sensor networks with partial coverage", *Computer Standards & Interfaces*, v. 61, pp. 20–35.

[2] ALI, A. I., HUANG, C.-H., 1991, "Balanced spanning forests and trees", *Networks*, v. 21, n. 6, pp. 667–687.

[3] ANGHINOLFI, D., CANNATA, G., MASTROGIOVANNI, F., et al., 2012, "Heuristic approaches for the optimal wiring in large scale robotic skin design", *Computers & Operations Research*, v. 39, n. 11, pp. 2715–2724.

[4] ANGHINOLFI, D., CANNATA, G., MASTROGIOVANNI, F., et al., 2014, "Application and Experimental Validation of Pheromone Design in Ant Colony Optimization: the Problem of Robot Skin Wiring", *Applied Artificial Intelligence*, v. 28, n. 3, pp. 292–321.

[5] ARAQUE, J. R., HALL, L. A., MAGNANTI, T. L., 1990, "Capacitated trees, capacitated routing, and associated polyhedra", .

[6] ARROYO, J. E. C., VIEIRA, P. S., VIANNA, D. S., 2008, "A GRASP algorithm for the multi-criteria minimum spanning tree problem", *Annals of Operations Research*, v. 159, n. 1, pp. 125–133.

[7] BAHIENSE, L., MACULAN, N., SAGASTIZÁBAL, C., 2002, "The volume algorithm revisited: relation with bundle methods", *Mathematical Programming*, v. 94, pp. 41–69.

[8] BAZGAN, C., COUËTOUX, B., TUZA, Z., 2011, "Complexity and approximation of the Constrained Forest problem", *Theoretical Computer Science*, v. 412, n. 32, pp. 4081–4091.

[9] BEASLEY, J. E., 1993, "Lagrangean heuristics for location problems", *European journal of operational research*, v. 65, n. 3, pp. 383–399.

[10] BLUM, C., 2002, "Ant colony optimization for the edge-weighted k-cardinality tree problem". In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pp. 27–34.

[11] BONATES, T. O., SANTIAGO, C. P., SILVA, J. B., 2011, "Novas Abordagens Exatas para o Problema de Partiçao de um Grafo em Arvores k-Capacitadas", *XLIII Simposio Brasileiro de Pesquisa Operacional*.

[12] BRUGLIERI, M., MAFFIOLI, F., EHRGOTT, M., 2004, "Cardinality constrained minimum cut problems: complexity and algorithms", *Discrete Applied Mathematics*, v. 137, n. 3, pp. 311–341.

[13] COUËTOUX, B., 2011, "A 3/2-approximation for a Constrained Forest Problem". In: Demetrescu, C., Halldórsson, M. M. (Eds.), *Algorithms – ESA 2011*, pp. 652–663, Berlin, Heidelberg. Springer Berlin Heidelberg. ISBN: 978-3-642-23719-5.

[14] COUËTOUX, B., DAVIS, J. M., WILLIAMSON, D. P., 2015, "A 3/2-approximation algorithm for some minimum-cost graph problems", *Mathematical Programming*, v. 150, n. 1, pp. 19–34.

[15] CRESCENZI, P., KANN, V., 1997, "Approximation on the web: A compendium of NP optimization problems". In: *International Workshop on Randomization and Approximation Techniques in Computer Science*, pp. 111–118. Springer.

[16] DA CUNHA, A. S., SIMONETTI, L., LUCENA, A., 2015, "Optimality cuts and a branch-and-cut algorithm for the K-rooted mini-max spanning forest problem", *European Journal of Operational Research*, v. 246, n. 2, pp. 392–399.

[17] DANTZIG, G. B., WOLFE, P., 1960, "Decomposition principle for linear programs", *Operations research*, v. 8, n. 1, pp. 101–111.

[18] DORIGO, M., BIRATTARI, M., STUTZLE, T., 2006, "Ant colony optimization", *IEEE computational intelligence magazine*, v. 1, n. 4, pp. 28–39.

[19] EDMONDS, J., 1965, "Paths, trees, and flowers", *Canadian Journal of mathematics*, v. 17, pp. 449–467.

[20] FEO, T. A., RESENDE, M. G., 1995, "Greedy randomized adaptive search procedures", *Journal of global optimization*, v. 6, n. 2, pp. 109–133.

[21] FLOYD, R. W., 1962, "Algorithm 97: Shortest Path", *Commun. ACM*, v. 5, n. 6 (jun.), pp. 345. ISSN: 0001-0782. doi: 10.1145/367766.368168. Disponível em: <`https://doi.org/10.1145/367766.368168`>.

[22] GAMBLE, A. B., PULLEYBLANK, W. R., 1989, "Forest covers and a polyhedral intersection theorem", *Mathematical Programming*, v. 45, n. 1, pp. 49–58.

[23] GAVISH, B., 1983, "Formulations and algorithms for the capacitated minimal directed tree problem", *Journal of the ACM (JACM)*, v. 30, n. 1, pp. 118–132.

[24] GOEMANS, M. X., WILLIAMSON, D. P., 1994, "Approximating minimum-cost graph problems with spanning tree edges", *Operations Research Letters*, v. 16, n. 4, pp. 183–189.

[25] GOUVEIA, L., 1995, "A 2n constraint formulation for the capacitated minimal spanning tree problem", *Operations Research*, v. 43, n. 1, pp. 130–141.

[26] GOUVEIA, L., SALAZAR-GONZÁLEZ, J.-J., 2013, "Polynomial-time separation of enhanced reverse multistar inequalities", *Operations Research Letters*, v. 41, n. 3, pp. 294–297.

[27] GUTTMANN-BECK, N., HASSIN, R., 1998, "Approximation algorithms for minimum tree partition", *Discrete Applied Mathematics*, v. 87, n. 1-3, pp. 117–137.

[28] HANSEN, P., MLADENOVIĆ, N., BRIMBERG, J., et al., 2019, "Variable neighborhood search". In: *Handbook of metaheuristics*, Springer, pp. 57–97.

[29] IMIELIŃSKA, C., KALANTARI, B., KHACHIYAN, L., 1993, "A greedy heuristic for a minimum-weight forest problem", *Operations Research Letters*, v. 14, n. 2, pp. 65–71.

[30] JI, X., 2004, "Graph partition problems with minimum size constraints", *Rensselaer Polytechnic Institute, Troy, New York*.

[31] JOHNSON, D. B., 1975, "Finding all the elementary circuits of a directed graph", *SIAM Journal on Computing*, v. 4, n. 1, pp. 77–84.

[32] KRUSKAL, J. B., 1956, "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proceedings of the American Mathematical society*, v. 7, n. 1, pp. 48–50.

[33] LASZLO, M., MUKHERJEE, S., 2005, "Another greedy heuristic for the constrained forest problem", *Operations Research Letters*, v. 33, n. 6, pp. 629–633.

[34] LASZLO, M., MUKHERJEE, S., 2005, "Minimum spanning tree partitioning algorithm for microaggregation", *IEEE Transactions on Knowledge & Data Engineering*, v. 17, n. 7, pp. 902–911.

[35] LASZLO, M., MUKHERJEE, S., 2006, "A class of heuristics for the constrained forest problem", *Discrete Applied Mathematics*, v. 154, n. 1, pp. 6–14.

[36] LASZLO, M., MUKHERJEE, S., 2011, "A Genetic Algorithm for the Constrained Forest Problem", *ACEEE International Journal on Information Technology*, v. 1, n. 3, pp. 47.

[37] LÓPEZ-IBÁÑEZ, M., DUBOIS-LACOSTE, J., PÉREZ CÁCERES, L., et al., 2016, "The irace Package: Iterated Racing for Automatic Algorithm Configuration", *Operations Research Perspectives*, v. 3, pp. 43–58. doi: 10.1016/j.orp.2016.09.002.

[38] LOURENÇO, H. R., MARTIN, O. C., STÜTZLE, T., 2003, "Iterated local search". In: *Handbook of metaheuristics*, Springer, pp. 320–353.

[39] LUCENA, A., 2005, "Non delayed relax-and-cut algorithms", *Annals of Operations Research*, v. 140, n. 1, pp. 375–410.

[40] MARTINS, S. L., RESENDE, M. G., RIBEIRO, C. C., et al., 2000, "A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy", *Journal of Global Optimization*, v. 17, n. 1-4, pp. 267–283.

[41] MEHROTRA, A., JOHNSON, E. L., NEMHAUSER, G. L., 1998, "An optimization based heuristic for political districting", *Management Science*, v. 44, n. 8, pp. 1100–1114.

[42] MEREL, A., GANDIBLEUX, X., DEMASSEY, S., 2011, "A collaborative combination between column generation and ant colony optimization for solving set packing problems". In: *9th metaheuristics international conference (MIC 2011), Udine, Italy.*

[43] PAPADIMITRIOU, C. H., 1978, "The complexity of the capacitated tree problem", *Networks*, v. 8, n. 3, pp. 217–230.

[44] PESSOA, A., SADYKOV, R., UCHOA, E., et al., 2020, "A generic exact solver for vehicle routing and related problems", *Mathematical Programming*, v. 183, pp. 483–523.

[45] PRIM, R. C., 1957, "Shortest connection networks and some generalizations", *The Bell System Technical Journal*, v. 36, n. 6, pp. 1389–1401.

[46] QIAN, J., WILLIAMSON, D. P., 2011, "An o (logn)-competitive algorithm for online constrained forest problems". In: *International Colloquium on Automata, Languages, and Programming*, pp. 37–48. Springer.

[47] QUEERN, J., 2013, "An evolutionary algorithm for the constrained forest problem", *Nova Southeastern University ProQuest Dissertations Publishing.*

[48] RESENDE, M. G., RIBEIRO, C. C., 2005, "GRASP with path-relinking: Recent advances and applications". In: *Metaheuristics: progress as real problem solvers*, Springer, pp. 29–63.

[49] RIBEIRO, C. C., RESENDE, M. G., 2012, "Path-relinking intensification methods for stochastic local search algorithms", *Journal of heuristics*, v. 18, pp. 193–214.

[50] ROPKE, S., PISINGER, D., 2006, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows", *Transportation science*, v. 40, n. 4, pp. 455–472.

[51] SANDE, G., 2002, "Exact and approximate methods for data directed microaggregation in one or more dimensions", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, v. 10, n. 05, pp. 459–476.

[52] SCHRIJVER, A., OTHERS, 2003, *Combinatorial optimization: polyhedra and efficiency*, v. 24. Springer.

[53] SIMONETTI, L., DA CUNHA, A. S., LUCENA, A., 2013, "Polyhedral results and a Branch-and-cut algorithm for the k-cardinality tree problem", *Mathematical Programming*, v. 142, n. 1, pp. 511–538.

[54] UCHOA, E., FUKASAWA, R., LYSGAARD, J., et al., 2008, "Robust branch-cut-and-price for the capacitated minimum spanning tree problem over a large extended formulation", *Mathematical Programming*, v. 112, pp. 443–472.

[55] WHITE, L. J., 1971, "Minimum covers of fixed cardinality in weighted graphs", *SIAM Journal on Applied Mathematics*, v. 21, n. 1, pp. 104–113.

[56] WOLSEY, L. A., NEMHAUSER, G. L., 2014, *Integer and combinatorial optimization.* John Wiley & Sons.

[57] YAMADA, T., TAKAHASHI, H., KATAOKA, S., 1997, "A branch-and-bound algorithm for the mini-max spanning forest problem", *European Journal of Operational Research*, v. 101, n. 1, pp. 93–103.