
# PROBLEMAS DE ÁRVORES GERADORAS EM GRAFOS COM ÊNFASE NO NÚMERO DE FOLHAS

Pedro Henrique Pereira Vargas Liguori

Dissertação de Mestrado 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 Mestre em Engenharia de Sistemas e Computação.

Orientador: Abilio Pereira de Lucena Filho

Rio de Janeiro
Agosto de 2014

PROBLEMAS DE ÁRVORES GERADORAS EM GRAFOS COM ÊNFASE NO NÚMERO DE FOLHAS

Pedro Henrique Pereira Vargas Liguori

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

Examinada por:

_____
Prof. Abilio Pereira de Lucena Filho, Ph.D.

_____
Prof. Nelson Maculan Filho, D.Sc.

_____
Prof. Luidi Gelabert Simonetti, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2014

Liguori, Pedro Henrique Pereira Vargas

Problemas de árvores geradoras em grafos com ênfase no número de folhas/Pedro Henrique Pereira Vargas Liguori. – Rio de Janeiro: UFRJ/COPPE, 2014.

XI, 70 p.: il.; 29, 7cm.

Orientador: Abilio Pereira de Lucena Filho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2014.

Referências Bibliográficas: p. 65 – 70.

1. Árvores em grafos. 2. Branch and Cut. 3. Relax and Cut. I. Lucena Filho, Abilio Pereira de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*Dedico esse trabalho ao meu irmão e amigo, Caio. Para que você se sinta motivado a não parar nunca com os seus estudos.*

# Agradecimentos

Em primeiro lugar, gostaria de agradecer aos mestres Prof. Abilio Pereira de Lucena Filho pela amizade, pelos ensinamentos e pela dignidade e enorme respeito dirigidos à minha pessoa, na sua dedicação e interesses genuínos demonstrados a mim e ao meu trabalho; e Prof. Nelson Maculan Filho pelo enorme acolhimento e inestimáveis ensinamentos durante a minha estadia no PESC. Agradeço também aos demais professores e colegas do Programa de Engenharia de Sistemas e Otimização, que tiveram papel fundamental na minha formação.

Não poderia deixar de agradecer também àquelas pessoas do meu círculo social que contribuiram com o seu tempo, amizade e paciência em diversos momentos. Em especial agradeço ao amigo Samir Chammas, incansável incentivador, que teve papel fundamental na realização do sonho de me tornar Mestre. Ao amigo André Lima, pela enorme paciência e tempo, por ter me ouvido e me ajudado a construir boa parte daquilo que hoje se materializa no presente trabalho. Aos amigos Thaisa e Tomaz, Cristiane e Alexandre por transformarem a minha vida e pelo suporte emocional tantas vezes fundamental!

Agradeço à Gapso e aos gapsianos, pela convivência e pelos ensinamentos. Em especial ao Marcelo Reis e ao Alexandre Pigatti por me ajudarem a viabilizar esse sonho.

Não poderia deixar de agradecer àquelas pessoas do meu mais íntimo circulo de convivência, que foram afetadas de maneira mais direta, seja pela minha indisponibilidade em diversos momentos (fundamental à realização desse trabalho) como também pela ajuda e apoio nos momentos de angústia, dúvida e de cansaço, inevitáveis durante toda essa caminhada. Agradeço à minha mãe e irmã, Edna e Ana Paula, pela confiança e pelo enorme carinho; ao meu irmão e amigo Caio, grande incentivador e fonte de inspiração, em especial no momentos de dúvida e angústia; à minha esposa Clarisse pela convivência, pelo amadurecimento conjunto, pela paciência e pelo enorme suporte emocial.

Finalmente, agradeço a Deus, fonte maior de inspiração e força, que permitiu que eu pudesse concluir mais essa importante etapa da minha vida.

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

# PROBLEMAS DE ÁRVORES GERADORAS EM GRAFOS COM ÊNFASE NO NÚMERO DE FOLHAS

Pedro Henrique Pereira Vargas Liguori

Agosto/2014

Orientador: Abilio Pereira de Lucena Filho

Programa: Engenharia de Sistemas e Computação

Dado um grafo não orientado $G = (V, E)$ e um inteiro $l$, o problema da árvore geradora com restrição no número de folhas consiste em encontrar uma árvore geradora $T = (V, E')$, $E' \subset E$, com *pelo menos* $l$ folhas. A versão de otimização desse problema busca encontrar uma árvore geradora de custo mínimo para $G$, que contenha pelo menos $l$ folhas. Um outro problema muito próximo ao que acabamos de descrever é aquele de encontrar uma árvore geradora de $G$ com um número máximo de folhas. Formulações, desigualdades válidas e algoritmos de solução para esses dois problemas são os temas principais investigados nesse trabalho.

Entre as principais contribuições alcançadas, destacamos a introdução de uma nova família de desigualdades válidas para a formulação clássica baseada em grafos não orientados.

Ainda, como contribuições do trabalho, chamamos a atenção para o desenvolvimento de um algoritmo exato do tipo *branch-and-cut* baseado em uma nova formulação proposta para os problemas apresentados acima. Esse algoritmo se mostrou capaz de resolver à otimalidade as maiores instâncias propostas até o problema, apresentado performance comparável à de outras formulações existentes na literatura. Adicionalmente, sugerimos um conjunto de instâncias difíceis para o problema da árvore geradora de custo mínimo com limite inferior do número de folhas. Finalmente, desenvolvemos um algoritmo do tipo *relax-and-cut*, que contém, como parte integrante, uma heurística Lagrangeana que se mostrou eficaz na construção de soluções viáveis de boa qualidade para o problema.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

LEAF CONSTRAINED MINIMUM SPANNING TREE PROBLEM

Pedro Henrique Pereira Vargas Liguori

August/2014

Advisor: Abilio Pereira de Lucena Filho

Department: Systems Engineering and Computer Science

Given a non-directed graph $G = (V, E)$ and a number $l$, the leaf constrained minimum spanning tree problem (LCMSTP) can be formulated as the search of a spanning tree $T = (V, E')$, $E' \subset E$, with at least $l$ leaves. In its optimization version, one should look for a minimum cost spanning tree of $G$ with at least $l$ leaves. A close related problem is that of finding a spanning tree of $G$ with as many leaves as possible (known in literature as the maximum leaf spanning tree problem — MLSTP). Formulations, valid inequalities and solution algorithms are the main focus of this work.

Among the most importants contribuitions achieved we point out the introduction of a new family of valid inequalities to characterize the leaves in a tree.

Other contribuition of this work is the development of a branch-and-cut algorithm based on the formulation introduced earlier. This algorithm was able to solve optimally the biggest instances known. We also develop a Lagrangean based algorithm, relax-and-cut, for wich we use a Lagrangean heuristic to build good primal solutions to the problem. Finally, we introduce a set of difficult instances for LCMSTP.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Muitas situações práticas reais podem ser modeladas com o auxílio de estruturas matemáticas chamadas grafos. Introduzidos na primeira metade do século XVIII pelo matemático suiço Leonard Euller como artifício utilizado na modelagem do problema das Sete Pontes de Königsberg (BIGGS *et al.* (1986)), o estudo dessas estruturas particulares atraiu a atenção de muitos pesquisadores e, desde então, tem recebido vertinoso impulso, especialmente após a segunda metade do século XX, devido ao avanço tecnológico e a popularização dos computadores HARARY (1994).

Considere o problema de se desenhar uma rede qualquer na qual a localização dos seu nós (vértices) é conhecida. Para essa rede, existe um conjunto de potenciais ligações (arestas) conectando seus nós. De acordo com GRAHAM e HELL (1985), exemplos de situações nas quais se deseja conceber (ou projetar) uma tal rede, seriam entre outras: a instalação de uma rede de computadores, a construção de dutos entre poços de petróleos, a criação de galerias ligando minas de exploração de minério e a construção de redes de cobertura de dados (especialmente para telefonia celular). Todas essas aplicações compartilham uma questão em comum: *conhecidos os custos para realizar a ligação entre os nós da rede, quais conexões deveriam ser criadas de modo a garantir que sempre exista um caminho entre quaisquer pares de nós, assegurando que o custo de construção seja o menor possível*? Esse problema será formalmente introduzido a seguir.

### Árvores, folhas e árvore de custo mínimo

Seja dado um grafo conexo não direcionado $G = (V, E)$, onde $V$ é o conjunto de vértices e $E$ o conjunto de arestas. Uma aresta $e = \{i, j\} \in E$ representa uma ligação entre os vértices $i, j \in V$. Adicionalmente, associa-se um valor $c_e \in \mathbb{R}^+$ à aresta $e = \{i, j\}$, correspondente ao seu custo de criação. Uma árvore de $G$ definida é como qualquer subgrafo $T = (V', E')$, $V' \subseteq V$ e $E' \subseteq E$, acíclo e conexo. Diz-se

que uma árvore $T$ *gera* (do inglês *span*) o grafo $G$ quando $V' = V$ (PADBERG e WOLSEY (1983)).

Uma folha de $T$ é qualquer vértice cujo grau de incidência (*i.e.*, número de arestas incidentes) seja igual a 1, naquele grafo. Posto de outra maneira, um vértice será chamado folha quando ele estiver ligado a apenas um único outro vértice de $T$ (*i.e.* quando possuir um único vizinho).

O *Problema da Árvore Geradora Mínima* (*Minimum Spanning Tree Problem* - – MSTP) consiste em encontrar um árvore geradora $T^*$ de $G$ com a menor soma possível para os custos das arestas.

A história do problema da árvore geradora mínima é longa e interessante. Um levantamento bastante completo desse tema foi feito por GRAHAM e HELL (1985). A referência mais antiga a um algoritmo que resolva o MSTP foi proposto por BORUVKA (1926), um engenheiro e matemático tcheco interessado em definir uma cobertura elétrica eficiente para a região da Bohemia. Em meados da década de 1950, devido ao aumento da capacidade de processamento dos computadores da época, o problema da árvore geradora mínima novamente despertou o interesse dos pesquisadores. Entre eles estão KRUSKAL (1956) e PRIM (1957), que por vezes são apontados como os percurssores do estudo desse problema e responsáveis pelos primeiros algoritmos eficientes de solução – embora ambos os autores se refiram, em seus trabalhos, à BORUVKA (1926).

O estudo de problemas de árvores em grafos continua a despertar o interesse dos pesquisadores, embora esses problemas sejam classificados na literatura de otimização combinatória como bem resolvidos — *i.e.* são problemas para os quais são conhecidos algoritmos polinomias de solução. Boa parte do interesse que esse tipo de aplicação atrai, se dá pelo fato de existir uma grande gama de problemas reais que podem ser modelados como árvores, não sendo raro encontrá-las como subestruturas em uma grande variedade de outros problemas adicionais. Um exemplo clássico de como os algoritmos de resolução do MSTP podem servir como ponto de partida para a construção de soluções de outros problemas (mais complexos, inclusive) é a heurística proposta por CHRISTOFIDES (1976) para a resolução do problema do caixeiro viajante.

Além disso, a inclusão de algumas restrições adicionais ao problema original pode fazer com que a busca por uma árvore geradora de custo mínimo se torne especialmente difícil. São exemplos dessas restrições: limite superior para o grau dos vértices NARULA e HO (1980), limite superior para o diâmetro da árvore ACHUTAN e CACCETTA (1990), ou a imposição de que o número de folhas seja exatamente igual a dois, fazendo com que o problema se transforme naquele de se encontrar um caminho Hamiltoniano JULSTROM (2004).

## 1.1 Árvore geradora de custo mínimo com restrição no número de folhas

Considere um grafo $G = (V, E)$ ao qual estão associados custos $\{c_e : e \in E\}$ e um inteiro $l$ tal que $2 \leq l \leq |V| - 1$. O problema da árvore geradora de custo mínimo com restrição no número de folhas consiste em encontrar uma árvore geradora $T$, de custo mínimo, que possua pelo menos $l$ folhas. Na literatura, é conhecido como *Leaf Constrained Minimum Spanning Tree Problem* (LCMSTP), e coube a DEO e MICIKEVICIUS (1999) a tarefa de mostrar que LCMSTP pertence à classe $\mathcal{NP}$-Difícil.

Importante ressaltar que boa parte da dificuldade de resolução das instâncias do LCMSTP está relacionada com o parâmetro $l$. Vale a pena observar que um grafo qualquer $G$ pode possuir uma grande quantidade de árvores geradores mínimas (MSTs), irrestritas quanto ao número de folhas. Caso o valor do parâmetro $l$ seja suficientemente pequeno, é bem provável que algumas dessas MSTs (talvez todas) possuam uma quantidade de folhas superior a $l$. Nesses casos, bastaria aplicar algum dos algoritmos de solução do MSTP citados anteriormente para resolver essa instância específica do LCMSTP. Analisando a outra extremidade do espectro de valores de $l$, quando $l = |V|-1$, busca-se uma árvore de custo mínimo com topologia de estrela — que também pode ser obtida de maneira bastante eficiente. Conforme relatos de alguns autores (JULSTROM (2004), GOUVEIA e TELHADA (2011)), as instâncias mais difíceis do problema são, em geral, aquelas para as quais $0,75 \cdot |V| \leq l \leq 0,9 \cdot |V|$.

Assim como outros problemas relacionados à busca por árvores de custo mínimo, o LCMSTP possui uma ampla gama de aplicações práticas, desde os tradicionais problemas de localização de facilidades, aos mais sofisticados problemas de desenho de redes de circuitos e de telecomunicações JULSTROM (2004). Ele está bastante ligado ao problema das $p$-medianas para o qual são conhecidos $n$ pontos e se deseja escolher $p$ entre eles (chamados de medianas) capazes de minimizar a soma das distâncias de cada ponto à mediana mais próxima. De fato, HOELTING *et al.* (1995) descrevem o problema da árvore geradora com restrição no número de folhas como sendo uma variação do problema clássico de $p$-medianas para o qual existe uma restrição adicional de que as medianas estejam, elas mesmas, conectadas entre si.

Outra aplicação prática do LCMSTP, relacionada ao desenho de redes (de computadores, sensores ou dispositivos *wireless*), foi introduzida por FERNANDES e GOUVEIA (1998). Nessas aplicações, topologias de árvores geradoras são altamente desejadas pois garantem (i) conexidade entre todos os nós da rede (ii) evitando desperdícios. A motivação em usar uma restrição sobre número de folhas da árvore

se deve ao fato de que, para alguns casos reais, os requisitos de hardware e software necessários ao bom funcionamento da rede podem ser função da posição que o dispositivo ocupará na topologia especificada. Geralmente, softwares e hardwares associados aos vértices de "*grau* 1" (*i.e* as folhas) são mais baratos do que aqueles associados aos nós intermediários. Esses últimos geralmente desempenham função de roteamento das mensagens que trafegam na rede: um nó intermediário $j$ precisa identificar se uma determinada mensagem deve ou não ser repassada a outro nó subsequente. Imaginando que, algumas vezes, o número de roteadores disponíveis para a construção da rede já está definido a priori (constituindo-se em uma restrição para a criação da rede — por exemplo, considere que existam disponíveis $k$ desses dispositivos), deseja-se encontrar uma topologia de árvore para a qual o número de nós internos seja de no máximo $k$ elementos — ou alternativamente, deseja-se encontrar uma árvore que possua pelo menos $l = n - k$ folhas, onde $n$ é o número total de nós que a rede possui.

O LCMSTP tem também aplicações em clusterização de grafos. Considere um conjunto $N$ que contém $n$ elementos e seja $d : N \times N \to \mathbb{R}$ uma função de dissemelhança entre os pares de elementos. Seja $k$ o número de clusteres que se deseja criar no grafo. É desejável que cada elemento esteja inserido em um único cluster e, que os clusteres sejam escolhidos de modo a minimizar as somas das diferenças entre os elementos internos a cada cluster, acrescidas da distância ao centro do cluster (utilizando para tanto a função de dissemelhança $d$). Esse problema de clusterização pode ser modelado como um caso particular do LCMSTP onde no centro de cada cluster estará localizado um vértice interno da árvore. Esses vértices, além de se ligar ao centro de algum outro cluster, estarão também ligados aos demais elementos componentes do cluster em questão, que por sua vez serão representados como folhas da árvore que se deseja encontrar.

Podem ser encontradas na literatura do tema diversas estratégias para a resolução do LCMSTP, boa parte das quais está voltada ao desenvolvimento de heurísticas eficientes para o problema. Entre as principais técnicas heurísticas implementadas, ressalta-se o trabalho pioneiro de DEO e MICIKEVICIUS (1999), no qual é apresentada uma heurística gulosa $\mathcal{O}(n^4)$ que, a partir de uma MST irrestrita do grafo, busca construir uma solução viável através de trocas consecutivas de arestas da árvore por outras que estejam fora dela. Esse mecanismo é construido de tal forma que (i) a solução resultante continue sendo uma árvore, (ii) que haja um aumento do número de folhas e (iii) que o acréscimo à funçao objetivo seja "o menor possível". Esse processo iterativo se repete até que uma árvore com pelo menos $l$ folhas seja encontrada (ou até que nenhuma troca de arestas seja mais possível, situação na qual o algoritmo falha em encontrar uma solução para o LCMSTP).

Seguindo a mesma linha, JULSTROM (2004) também propõe uma heurística

$\mathcal{O}(n^4)$ que parte de uma MST irrestrita e busca, iterativamente, aumentar o número de folhas através de um mecanismo chamado MAIS-FOLHAS. Esse método é aplicado a cada vértice interno da árvore inicial com objetivo de transformá-lo em uma folha. Para isso, o vértice escolhido é retirado da árvore, criando duas componentes conexas separadas. Para que se forme novamente uma árvore, são identificados os vértices internos dessas componentes conexas e uma avaliação dos custos de ligação desses vértices internos (pertencentes a componentes conexas distintas) é conduzida. A ligação de menor custo é então reestabelecida, reconectando as duas componentes conexas em uma única. Nesse ponto do algoritmo a configuração do grafo é a seguinte: existirá uma grande componente conexa, contendo $|V| - 1$ vértices, e haverá um vértice desconectado (aquele vértice que havia sido inicialmente retirado da árvore). Finalmente, como último passo, o algoritmo avalia os custos para reconstruir a árvore, conectando o vértice solitário à compenente conexa através do menor custo de ligação desse vértice a um dos vértices internos da componente conexa. É fácil notar que a heurística proposta é sempre bem sucedida quando o grafo em questão é completo.

Abordagens mais sofisticadas, baseadas em meta-heurísticas, também podem ser encontradas na literatura, com especial ênfase para os algoritmos evolucionários, em especial o genético. Nesses casos, vale a pena citar os trabalhos de EDELSON e GARGANO (2000) e JULSTROM (2004), com especial destaque para a representação utilizada para as soluções viáveis (que se baseia em *strings* de Prüfer), possibilitando que o *output* das etapas de *cross-over* das soluções sejam sempre árvores viáveis para o LCMSTP. Tanto quanto sabemos, não foram propostos algoritmos aproximativos para o problema.

Em se tratando de algoritmos exatos, não são muitas as referências dedicadas ao LCMSTP, sendo muito mais fértil a busca por modelos e algoritmos exatos para um outro problema relacionado — o da Árvore Geradora com Número Máximo de Folhas (MLSTP - *Maximum Leaf Spanning Tree Problem*). Dentre os trabalhos dedicados ao LCMSTP, nosso ponto de partida é o trabalho de FERNANDES e GOUVEIA (1998) (talvez a primeira referência a abordar esse tema). Nesse trabalho, os autores introduziram duas formulações multifluxo distintas que permitem a resolução de instâncias relativamente pequenas, não excedendo 40 vértices.

Posteriormente, GOUVEIA e TELHADA (2011) fazem uma adaptação de uma formulação de arborescência proposta para o MLSTP, que tem como pedra angular a escolha de um dos vértices do grafo para raiz da arborescência em construção. Nesse método proposto pelos autores, ao invés de se utilizar uma única raiz, a formulação estendida baseia-se na "interseção" das formulações para cada raiz distinta. Devido ao grande número de variáveis e restrições criadas, os autores propõem um mecanismo que parte de um modelo inicial, criado com base em uma única raiz,

e acrescenta iterativamente novas formulações para raízes diferentes daquela considerada inicialmente. Com isso, obtém-se uma solução de compromisso, capaz de balancear a qualidade dos *bounds* obtidos com a dimensão da formulação a resolver. Ainda de acordo com esse último trabalho, os autores resolvem, para grafos esparsos, instâncias de tamanho até 60 vértices.

## 1.2 Árvore geradora com número máximo de folhas

Considere um grafo $G = (V, E)$, conexo e não direcionado. O MLSTP consiste em encontrar uma árvore geradora de $G$ com o maior número possível de folhas (FUJIE (2003), FUJIE (2004), LUCENA *et al.* (2010)).

Como observado por LUCENA *et al.* (2010), o MLSTP pode ser trivialmente resolvido para grafos completos e sua solução será qualquer estrela (para as quais o número de folhas é conhecido e sempre igual a $|V| - 1$). Entretanto, para o caso geral em que $G$ é um grafo esparso, foi demonstrado por GAREY e JOHNSON (1976) que o problema pertence à classe $\mathcal{NP}$-Difícil. Mesmo para grafos cúbicos (*i.e.* um grafo regular no qual todos os vértices tem grau igual a três), o problema continua pertencendo à classe $\mathcal{NP}$-Difícil (LEMKE (1988)). Conforme demonstrado por GALBIATI *et al.* (1994), o MLSTP também faz parte da classe dos problemas $\mathcal{MAX}$-$\mathcal{SNP}$-Difícil, isto é, existe $\epsilon > 0$ tal que o problema de resolver o MLSTP com um fator de aproximação de $(1 + \epsilon)$ pertence à classe $\mathcal{NP}$-Difícil. Algoritmos aproximativos para o problema, de fator 3 e 2, foram propostos respectivamente por LU e RAVI (1998) e SOLIS-OBA (1998).

Existem diversas aplicações práticas para o problema da árvore geradora com número máximo de folhas. Entre as principais podem ser citadas aquelas relacionadas ao desenho de circuitos eletrônicos e de redes de telecomunicação GUHA e KHULLER (1998), STORER (1981) e GOUVEIA e TELHADA (2011). Um trabalho mais recente, devido a CHEN *et al.* (2010) descreve uma aplicaçãodo do MLSTP ao projeto de redes de fibra ótica para as quais são necessárias a instalação de regeneradores óticos — tipo particular de fibra ótica contendo um revestimento interno especial capaz de intensificar os sinais luminosos degradados pelas perdas que ocorrem nas transmissões a longas distâncias. Nessas aplicações, os regeneradores óticos são tratados como vértices internos de um árvore definida sobre um grafo $G$, que se origina do grafo de definição do problema.

## 1.3 Problemas correlatos

### Problema restrito das $p$-medianas

Na área de otimização combinatória, os chamados problemas de localização são aqueles que têm como objetivo definir o melhor local de instalação de um determinado centro de serviços, seja ele um *shopping center*, um depósito, um hospital, entre outros. A estrutura matemática desses problemas depende da forma de seleção de uma localidade (dentre a opções disponíveis) e da estratégia de avaliação da qualidade das localizações — para uma discussão mais abrangente a respeito das classificações dos problemas de localização, indicamos o trabalho MINIEKA (1977).

O problema das $p$-medianas de um grafo é um problema clássico de localização em redes, pertencente à classe de problemas $\mathcal{NP}$-Difícil, conforme demonstrado por GAREY e JOHNSON (1976). Considere um grafo conexo e não direcionado $G = (V, E)$, com $n$ vértices, e um inteiro $p$, $p < n$. O problema das $p$-medianas consiste em encontrar um conjunto $P$, de cardinalidade $|P| = p$, de pontos (também chamados, medianas), no grafo $G$, de modo a minimizar a soma das distâncias dos vértices ao ponto mais próximo HOELTING *et al.* (1995). Dependendo da situação a ser modelada, esses pontos podem estar restritos aos vértices do grafo, mas, em casos mais gerais, podem ser aceitos até mesmo pontos sobre as arestas do grafo. Cada um dos pontos em $P$ servirá como um centro de serviços para os vértices que lhes são mais próximos no grafo.

Finalmente, pode-se desejar que as medianas sejam elas mesmas conectadas entre si. Esse tipo de restrição surge naturalmente em contextos de fluxos em redes (como é o caso de *supply chain*), no qual se está interessado em encontrar a localização ótima de centros de distribuição avançados para atendimento de mercados consumidores. Nesses casos, se deseja assegurar que a rede construída seja conexa, de modo a assegurar que sempre exista um caminho entre os centros de distribuição para o escoamento de bens e serviços. Essa imposição surge como uma alternativa à minimização dos riscos da indústria. Sua adição faz com que a solução para o problema das $p$-medianas seja uma árvore no subgrafo induzido pelas medianas, conectadas aos demais vértices do grafo (que necessariamente devem ser atendidos por um dos centros de serviços localizados em um dos $p$ pontos) a um custo mínimo.

Olhando sob a ótica dos vértices que serão servidos, o problema das $p$-medianas, com a adição da restrição de conexidade entre as medianas, é equivalente ao de se encontrar uma árvore geradora de custo mínimo no grafo, contendo pelo menos $n-p$ folhas. De fato, HOELTING *et al.* (1995) descrevem o problema da árvore geradora com restrição no número de folhas como sendo essa variação do problema clássico de $p$-medianas.

**Problema do conjunto dominante conexo**

Considere um grafo conexo não orientado $G = (V, E)$. Um conjunto $S$, $S \subseteq V$, é dito dominante se todo vértice de $V \backslash S$ for adjacente a pelo menos um vértice de $S$ — *i.e.*, se existir pelo menos uma aresta ligando cada vértice de $V \backslash S$ a um vértice de $S$. O conjunto dominante será conexo quando o subgrafo que $S$ induz em $G$, $G_S = (S, E(S))$, for conexo ou contiver apenas um único vértice. O Problema do Conjunto Dominante Conexo Mínimo (MCDSP — *Minimum Connected Dominating Set Problem*) busca encontrar o conjunto dominante conexo de $G$ com a menor cardinalidade possível GUHA e KHULLER (1998).

Esse problema de otimização combinatória está intimamente ligado ao problema da árvore geradora com número maximo de folhas. De fato, conforme apontado em LUCENA *et al.* (2010), dada uma solução ótima para o MLSTP é possível obter de maneira eficiente uma solução ótima para o MCDSP — sendo o inverso também verdadeiro. A partir de qualquer conjunto dominante conexo de um grafo $G$, digamos um conjunto $S \subseteq V$, é possível construir eficientemente uma árvore geradora $T$ para $G_S$. Essa árvore poderá então ser expandida para se tornar uma árvore geradora de $G$ fazendo com que todos os vértices dominados $V \backslash S$ sejam folhas (note que, pela definição de conjunto dominante, essa construção é sempre possível). Assim, para qualquer conjunto dominate conexo $S$, é possível construir eficientemente uma árvore geradora de $G$ com $(|V| - |S|)$ folhas. Em particular, quando $S$ for um cojunto dominante conexo mínimo (*i.e.* aquele para o qual $|S|$ é mínima), então a árvore geradora construída por esse mecanismo será necessariamente uma árvore geradora de $G$ com o número máximo de folhas. Na direção oposta, partindo-se da solução da árvore geradora com número máximo de folhas será possível identificar um cojunto dominante conexo mínimo de maneira eficiente tomando-se apenas os vértices internos da árvore.

## 1.4 Organização do trabalho

O presente trabalho está organizado da seguinte maneira: no Capítulo 2 são apresentadas as principais formulações encontradas na literatura para o MLSTP e o LCMSTP. Nesse mesmo capítulo são também introduzidas novas formulações para esses problemas.

No Capítulo 3 são inicialmente discutidas algumas bases teóricas para a construção de algortimos do tipo *branch-and-cut*. Seguindo-se à discussão teórica, são apresentadas algumas famílias de desigualdades válidas, comuns aos problemas aqui tratados, e, posteriormente, são introduzidos dois algoritmos de planos de corte, um deles para o MLSTP e o outro para o LCMSTP. Ao final do capítulo, são apresen-

tados resultados computacionais para os experimentos efetuados.

A seguir, no Capítulo 4, após uma introdução teórica relativa a algoritmos de Relaxação Lagrangeana, é apresentado um algoritmo do tipo *relax-and-cut* para o LCMSTP. Esse tipo de algoritmo pode ser entendido como um análogo Lagrangeano de um algoritmo de planos de corte. Essa apresentação será também seguida pela discussão a respeito dos resultados cumputacionais dos experimentos.

Finalizando o trabalho, estará, no Capítulo 5, a síntese de todo o trabalho feito, ressaltando algumas das principais contribuições e indicando a possibilidade de trabalhos futuros.

# Capítulo 2

# Formulações

Nesse capítulo são apresentadas algumas das formulações clássicas encontradas na literatura do MLSTP e que servirão tanto como base para formulações do LCMSTP, quanto como inspiração para a construção de uma nova formulação para o problema.

Embora exista uma literatura bastante fértil de trabalhos relacionados ao MLSTP, o estudo da cronologia dos problemas de árvores geradoras com ênfase na quantidade de folhas revela que FERNANDES e GOUVEIA (1998) foram os primeiros a modelar o problema. Especificamente, os autores introduziram uma formulação para uma árvore geradora com um número exato $k$ de folhas. Nesse trabalho, os autores propuseram uma formulações de fluxo baseada em grafos orientados, e outra formulação alternativa para grafos não-orientados. Sugeriram também mecanismos para adaptar essas ao MLSTP — embora não sejam apresentados resultados computacionais dessa adaptação.

Uma releitura bem sucedida da formulação de FERNANDES e GOUVEIA (1998) para o MLSTP foi proposta, na sequência, por FUJIE (2003) e FUJIE (2004). Além de fortalecer a formulação inicial, o autor propôs uma família de cortes que induzem facetas do politopo formado pela envoltória convexa das soluções viáveis para o problema.

Buscando aproveitar-se das facetas outrora identificadas, LUCENA *et al.* (2010) introduziram uma formulação análoga àquela de FUJIE (2003), diferenciando-se inicialmente pela orientação das arestas do grafo. Ao propor que essa reformulação fosse escrita em termos de um digrafo, os autores foram capazes de identificar uma nova família de desigualdades (de difícil caracterização para o caso não orientado), o que permitiu a construção de um algoritmo exato (do tipo *branch-and-cut*) bastante eficiente para o problema. Merece destaque o fato de que essa formulação demanda a escolha de um vértice $v$ para atuar como raiz da arborescência a ser obtida. Por sua vez esse fato faz com que os limitantes de relaxação linear resultantes passem a depender da escolha do vértice raiz — como efeito colateral da introdução das desigualdades aludidas acima.

Ainda em LUCENA *et al.* (2010), uma nova formulação alternativa, em termos de árvore de Steiner, foi proposta para o MLSTP. A partir dessa segunda abordagem, os autores obtêm limites de relaxação linear ainda mais fortes, embora computacionalmente mais caros. Esta reformulação de Steiner foi também independentemente proposta por CHEN *et al.* (2010), no contexto de modelagem do problema de regeneradores óticos — embora não tenha sido observado por esses últimos autores o fato de que o problema subjacente seria aquele de conjuntos dominantes conexos mínimos, cuja equivalência com o LCMSTP já foi abordada nesse texto.

Finalmente, o trabalho de GOUVEIA e TELHADA (2011) apresenta uma técnica alternativa de modelagem para o LCMSTP, baseada na primeira formulação apresentada por LUCENA *et al.* (2010) para o MLSTP. Nessa abordagem, são construídos limitantes de relaxação linear extremamente fortes numa reformulação do LCMSTP batizada pelos próprios autores como de interseção de modelos direcionados. Mais uma vez, os resultados computacionais indicam que esses limitantes são caros e de difícil obtenção.

Em resumo, naquilo que tange a resolução exata dos problemas de árvores geradoras com ênfase na quantidade de folhas, a maioria das formulações encontradas na literatura difere, principalmente, na forma pela qual são modeladas as folhas das árvores. Uma vez definidas as bases que permitem a sua caracterização (das folhas), as adaptações dos modelos para os diferentes problemas são, na maioria das vezes, diretas. Assim sendo, os modelos aqui apresentados terão como arcabouço o problema da árvore geradora com número máximo de folhas (MLSTP). Quando oportuno, serão indicados os mecanismos de adaptação das formulações apresentadas para o LCMSTP.

## 2.1 Formulação de Fujie

A primeira formulação aqui apresentada para o MLSTP foi introduzida por FUJIE (2003), e tem como origem uma das formulações de FERNANDES e GOUVEIA (1998) para o LCMSTP.

Considere um grafo conexo, não orientado, $G = (V, E)$, como definido anteriormente. Seja $\delta_i$ o conjunto das arestas incidentes em $i \in V$ e, $n_i = |\delta_i|$, o grau correspondente a esse vértice em $G$. Ainda, seja $E(S) = \{\{i, j\} \in E : i \in S, j \in S, S \subset V\}$ um subconjunto de arestas induzido pelo subconjunto de vértices $S$. São então definidas as variáveis de decisão $\{x_e \in \mathbb{R}_+^{|E|} : e \in E\}$, associadas às arestas do grafo, que indicam se uma dada aresta estará ou não presente na solução. Além dessas, são também definidas as variáveis $\{z_i \in \mathbb{R}_+^{|V|} : i \in V\}$, associados aos vértices do grafo, para caracterizar as folhas de uma árvore geradore de $G$. Considere então a região

poliedral $\mathcal{R}_1$, definida em relação aos conjuntos de variáveis introduzidos, como

$$\sum_{e \in E} x_e = |V| - 1 \tag{2.1}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \quad S \subset V,\ |S| \geq 2 \tag{2.2}$$

$$\sum_{e \in \delta_i} x_e + (n_i - 1) z_i \leq n_i, \qquad i \in V \tag{2.3}$$

$$0 \leq x_e \leq 1 \qquad e \in E \tag{2.4}$$

$$0 \leq z_i \leq 1 \qquad i \in V. \tag{2.5}$$

A formulação proposta por FUJIE (2003) para o MLSTP é dada por

$$max \left\{ \sum_{i \in V} z_i : (\mathbf{x}, \mathbf{z}) \in \mathcal{R}_1 \cap \left( \mathbb{B}^{|E|} \times \mathbb{B}^{|V|} \right) \right\}. \tag{2.6}$$

Diferentemente do modelo proposto por FERNANDES e GOUVEIA (1998), nessa formulação, FUJIE (2003) utiliza uma versão mais apertada das desigualdades de caracterização das folhas da árvore (2.3).

Conforme apontado por FUJIE (2003), nota-se que um vértice $i \in V$ de grau 1 numa solução viável $(\bar{x}, \bar{z})$ para (2.6) (ou seja, tal que $\sum_{e \in \delta_i} \overline{x}_e = 1$) não garante necessariamente que $z_i = 1$ seja satisfeito. Na realidade, o que garante a satisfação dessa condição para qualquer solução ótima de (2.6) é a estrutura da função objetivo. Para que se tenha, de fato, uma formulação para MLSTP é suficiente acrescentar à $\mathcal{R}_1$ a seguinte restrição:

$$\sum_{e \in \delta_i} x_e + z_i \geq 2,\ i \in V. \tag{2.7}$$

Em seu trabalho posterior, FUJIE (2004) mostrou que é possível fortalecer a formulação (2.6) pela inclusão da seguinte família de desigualdades válidas:

$$\sum_{e \in F} x_e + (|F| - 1) z_i \leq |F|,\ i \in V,\ F \subseteq \delta_i,\ |F| \geq 2. \tag{2.8}$$

Nessa oportunidade, o autor demonstrou que esses cortes induzem facetas para a envoltória convexa dos pontos inteiros de (2.6) — tomando $F = \delta_i$, fica evidente que a restrição acima se trata de uma generalização de (2.3).

### 2.1.1 Adaptação para o LCMSTP

A adaptação da formulação proposta por FUJIE (2003) para LCMSTP é bastante direta. Considere um grafo $G = (V, E)$ não orientado e uma função real de custos

$c : e \in E \rightarrow \mathbb{R}$. A imposição de um número mínimo $l$ de folhas para a árvore $T$ que se deseja obter pode ser feita diretamente através da adição da seguinte desigualdade:

$$\sum_{i \in V} z_i \geq l. \tag{2.9}$$

Denotando por $\mathcal{R}_2$ a região poliedral definida pela interseção de $\mathcal{R}_1$ com as restrições (2.7) e (2.9), o LCMSTP pode ser então formulado como

$$min \left\{ \sum_{e \in E} c_e x_e : (\mathbf{x}, \mathbf{z}) \in \mathcal{R}_2 \cap \left( \mathbb{B}^{|E|} \times \mathbb{B}^{|V|} \right) \right\}. \tag{2.10}$$

Vale ressaltar que a adaptação proposta configura uma versão não direcionada da formulação proposta por FERNANDES e GOUVEIA (1998) para o LCMSTP. Assim como para o MLSTP, a família de desigualdades (2.8) também é válida para para esse outro problema.

## 2.2 Formulação baseada em arborescências

Em seu trabalho, LUCENA *et al.* (2010) propuseram uma formulação alternativa para o problema da árvore geradora com número máximo de folhas, análoga àquela sugerida por FUJIE (2003). Aqui, no entanto, os autores trabalham sobre um grafo orientado, transformando o problema de maneira que o objetivo passe a ser o de encontrar uma arborescência geradora com um número máximo de folhas. Essa formulação pode ser interpretada como uma versão para o MLSTP da formulação (direcionada) de FERNANDES e GOUVEIA (1998) para o LCMSTP, que se utiliza do reforço proposto por FUJIE (2003) para as restrições (2.3), acrescida de novas desigualdades válidas, que serão introduzidas na sequência do texto.

Seja $r \in V$ o vértice definido como raiz da arborescência a ser construída para o digrafo $D = (V, A)$, que se origna de $G = (V, E)$. O conjunto $A$ de arcos é definido como:

$$A = \{(i,j),(j,i) : e = [i,j] \in E,\ i,j \in V \backslash \{r\}\} \ \cup \ \{(r,j) : e = (r,j) \in E\}. \tag{2.11}$$

Ainda, $A(S)$, $S \subset V$, sera utilizado para denotar o subconjunto de arcos com ambas as extremidades em $S$.

Adicionalmente, denotamos por $\delta_i^+ \subset A$ e $\delta_i^- \subset A$, respectivamente, o conjunto dos arcos que saem do vértice $i$, e o conjunto dos arcos que entram em $i$. Segue da construção do conjunto $A$ que $\delta_r^- = \emptyset$.

O grau de um vértice $i \in V \backslash \{r\}$ é definido pelo seu grau de entrada, $n_i = |\delta_i^-|$. Para o vértice raiz será adotado o grau de saída $n_r = |\delta_r^+|$. Observe que a definição

de um vértice folha continua a mesma: serão considerados folhas todos os vértices de grau 1 na arborescência. Posto de forma alternativa, o vértice $i \in V \backslash \{r\}$ será folha se e somente se não houver arcos apontados para fora de $i$. Por outro lado, a raiz $r$ será uma folha se e somente se houver um único arco dela saindo.

Diante do que acabamos de expor, definem-se as variáveis $\{y_a \in \mathbb{R}^{|A|} : a \in A\}$, que indicam se o arco $a$ está ou não na arborescência, e as variáveis $\{z_i \in \mathbb{R}^{|V|} : i \in V\}$, que indicam se o vértice $i$ é ou não uma folha.

A região poliedral $\mathcal{R}_3$, proposta por LUCENA *et al.* (2010), associada à arborescência geradora com um número máximo de folhas é definida como

$$\sum_{j \in V} y_{ji} = 1, \qquad i \in V \backslash \{r\} \tag{2.12}$$

$$\sum_{a \in A(S)} y_a \leq |S| - 1, \quad S \subset V,\ |S| \geq 2 \tag{2.13}$$

$$\sum_{j \in V} y_{ij} + (n_i - 1) z_i \leq (n_i - 1), \quad i \in V \backslash \{r\} \tag{2.14}$$

$$\sum_{j \in V} y_{rj} + (n_r - 1) z_r \leq n_r, \tag{2.15}$$

$$y_a \geq 0 \qquad a \in A \tag{2.16}$$

$$z_i \geq 0 \qquad i \in V. \tag{2.17}$$

Importante notar que (2.12) assegura que, para cada vértice da arborescência, existirá um único arco apontando para ele (à exceção do vértice raiz); (2.13) garante que a solução encontrada será acíclica. Juntas com (2.16) tais restrições descrevem o politopo da arborescência geradora (LUCENA *et al.* (2010)). Caso sejam adicionadas as restrições $x_e \geq 0$ e $x_e = y_{ij} + y_{ji}$, para $e = \{i, j\} \in E$, ter-se-á então uma descrição alternativa para o politopo da árvore geradora (MAGNANTI e WOLSEY (1995)).

As desigualdades (2.14) e (2.15) são utilizadas para indicar que, caso um vértice seja folha na solução, então não poderão existir arcos apontando para fora desse vértice. Entretanto, como não é possível identificar a priori se o vértice raiz $r$ será uma folha ou um vértice interno de uma arborescência ótima, é preciso que se faça um tratamento específico para o mesmo. Esse fato explica a diferença entre os valores à direita nas restrições (2.14) e (2.15).

Finalmente, tem-se a formulação proposta por LUCENA *et al.* (2010) para o problema da arborescência geradora com número máximo de folhas,

$$max \left\{ \sum_{i \in V} z_i : (\mathbf{y}, \mathbf{z}) \in \mathcal{R}_3 \cap \left( \mathbb{B}^{|A|} \times \mathbb{B}^{|V|} \right) \right\}, \tag{2.18}$$

cuja relaxação linear fornece o mesmo valor de limitante da formulação (2.6).

A grande novidade da reformulação proposta por LUCENA *et al.* (2010) está na caracterização de uma família de desigualdades que se mostrou bastante eficiente no fortalecimento do modelo, superando amplamente os limitantes fornecidos pelos modelos de FUJIE (2003) e FUJIE (2004):

$$y_a + z_i \leq 1,\ a \in \delta_i^+,\ i \in V \backslash \{r\}. \tag{2.19}$$

Conforme apontado pelo autores, caracterizar tal família, para o caso não direcionado, não é tarefa trivial. Temos ainda que, mesmo fortalecido por essas desigualdades, o *bound* de relaxação linear da formulação (2.18) continua sendo uma função da escolha da raiz. Foi essa relação de dependência que motivou a busca por uma formulação que pudesse, de alguma forma, se beneficiar da orientação das arestas do grafo, sem no entanto depender, para o limitante obtido, da escolha de uma raiz para a arborescência.

### 2.2.1 Adaptação para o LCMSTP

Mais uma vez, a adaptação da formulação de LUCENA *et al.* (2010) para o caso da arborescência geradora de custo mínimo com restrição no número de folhas é direta, bastando apenas acrescentar, àquela formulação, as restrições

$$\sum_{a \in \delta_i^+} y_a + z_i \geq 1,\ i \in V \backslash \{r\} \tag{2.20}$$

e

$$\sum_{a \in \delta_r^+} y_a + z_r \geq 2, \tag{2.21}$$

que são as correspontes orientadas de (2.7).

Tomando $\mathcal{R}_4$ como a interseção de (2.9), (2.20), (2.21) e $\mathcal{R}_3$, é possível formular o problema da arborescência geradora de custo mínimo com restrição no número de folhas como

$$min \left\{ \sum_{a \in A} c_a y_a : (\mathbf{y}, \mathbf{z}) \in \mathcal{R}_4 \cap \left( \mathbb{B}^{|A|} \times \mathbb{B}^{|V|} \right) \right\}, \tag{2.22}$$

onde $c_{ij} = c_{ji} = c_e$ para $e = \{i, j\} \in E$.

Como antes, essa formulação pode ser fortalecida pelas desigualdades (2.19). Da mesma forma, assim como ocorre para (2.18), a formulação pode também ser fortalecida pelas contrapartidas direcionadas de (2.8):

$$\sum_{a \in F} y_a + \sum_{\substack{a=(j,i) \in A: \\ (i,j) \in F}} y_a + (|F| - 1) z_i \leq |F|,$$
$$\forall i \in V,\ F \subseteq \delta_i^+ \cup \{(r,i)\},\ |F| \geq 2. \tag{2.23}$$

## 2.3 Reformulação por interseção

Tanto quanto sabemos, a formulação mais forte atualmente para o LCMSTP (no que tange a qualidade dos *bounds* de relaxação linear obtidos) é devida a GOUVEIA e TELHADA (2011). Nesse trabalho, os autores impõem que a formulação sugerida por LUCENA *et al.* (2010) seja escrita não em termos de uma única raiz, mas em termos de todas as possíveis raízes $r \in V$, dando origem a um modelo de interseção (conforme nomenclatura dos próprios autores). A motivação para essa reformulação do problema partiu da percepção de que os *bounds* de relaxação linear fornecidos pela formulação (2.18), acrescida da família de desigualdades (2.19), variavam enormemente dependendo da escolha da raiz.

Embora pareça bastante promissora, a reformulação por interseção tem sua aplicação limitada à resolução de problemas extremamente pequenos, devido ao elevado número de variáveis e restrições envolvidas — os testes realizados se limitaram a grafos com no máximo 50 vértices.

Buscando contornar essa limitação, os autores propuseram um algoritmo de resolução que se inicia com uma formulação para uma única raiz e, sequencialmente, com base em violações encontradas na solução de relaxação linear dessa formulação, adiciona outras formulações completas (para raízes diferentes), na expectativa de se obter um modelo de intermediário no qual apenas algumas raízes fossem consideradas na interseção.

## 2.4 Fortalecimento das formulações

Conforme observado por LUCENA *et al.* (2010), os *bounds* de relaxação linear obtidos para o MLSTP a partir das formulações de árvore, (2.6), e de arborescência, (2.18), são rigorosamente os mesmos — algo que, em uma análise superficial, poderia indicar não ser vantajosa a adoção da segunda formulação.

Entretanto, como mostram os autores, o direcionamento das arestas do grafo, permite a caracterização de uma família de desigualdades bastante simples, (2.19), capaz de melhorar consideravelmente a qualidade dos *bounds* fornecidos pela formulação de arborescência. Ainda, segundo os próprios autores, a caracterização desses cortes para o modelo subjacente de árvore não é trivial.

A partir dessa observação, e numa tentativa de fortalecer a formulação de árvore, foram realizados alguns estudos com objetivo de tentar projetar essa família no subespaço definido por $\mathcal{R}_1$. Considere:

- $r \in V$ - vértice tomado para raiz da arborescência.
- $y_{ij}$ - variável que indica a existência de um arco (na formulação de arborescência) que parte do vértice $i$ em direção ao vértice $j$.
- $x_{ij}$ - variável que indica a existência de uma aresta (na formulação de árvore) entre os pares de vértices $i$ e $j$ da árvore.
- $z_i$ - variável que indica se o vértice $i$ é uma folha.

### Aplicação do método de Fourier-Motzkin

Para obter uma projeção dos cortes (2.19) foi aplicado o método de eliminação de Fourier-Motzkin. Seja $P \subseteq \mathbb{R}^5$ o subespaço definido pelas seguintes desigualdades lineares:

$$P = \begin{cases} y_{ij} + y_{ji} - x_{ij} = 0 \\ y_{ij} + z_i \leq 1 \\ y_{ji} + z_j \leq 1 \\ y_{ij} \leq 1 \\ y_{ji} \leq 1 \\ x_{ij} \leq 1. \end{cases} \tag{2.24}$$

Isolando $y_{ij}$ no sistema acima, obtemos:

$$\begin{cases} y_{ij} = x_{ij} - y_{ji} \\ y_{ij} \leq 1 - z_i \\ y_{ij} \leq 1 \\ y_{ji} + z_j \leq 1 \\ y_{ji} \leq 1 \\ x_{ij} \leq 1. \end{cases} \tag{2.25}$$

Com base no sistema acima, podemos eliminar a variável $y_{ij}$ e escrever:

$$x_{ij} - y_{ji} = yij \leq min\{1, 1 - z_i\} \Rightarrow \begin{cases} x_{ij} - y_{ji} \leq 1 - z_i \\ x_{ij} - y_{ji} \leq 1. \end{cases} \tag{2.26}$$

Incorporando ao sistema original as desigualdades obtidas pela eliminação de $y_{ij}$ e aplicando raciocínio análogo, isolaremos agora $y_{ji}$:

$$\left\{\begin{array}{rcrcrcl} x_{ij} & - & y_{ji} & & \leq & 1 & - \; z_i \\ x_{ij} & - & y_{ji} & & \leq & 1 & \\ & & y_{ji} & + \; z_j & \leq & 1 & \\ x_{ij} & & & & \leq & 1 & \\ & & y_{ji} & & \leq & 1 & \end{array}\right. \Rightarrow \left\{\begin{array}{rcrcrcl} y_{ji} & & \geq & x_{ij} & + & z_i & - \; 1 \\ y_{ji} & & \geq & x_{ij} & & & - \; 1 \\ y_{ji} & & \leq & & - & z_i & + \; 1 \\ y_{ji} & & \leq & & & & 1 \\ & x_{ij} & \leq & & & & 1. \end{array}\right. \tag{2.27}$$

Da maneira como está apresentado o sistema, é fácil observar que

$$max\{x_{ij} + z_i - 1, x_{ij} - 1\} \leq y_{ji} \leq min\{1, 1 - z_j\}. \tag{2.28}$$

Isso permite que seja construido um último sistema de desigualdades, agora apenas nas variáveis $x$ e $z$, ambas utilizadas na caracterização da formulação de árvore (2.6):

$$\left\{\begin{array}{rcrcrcl} x_{ij} & + & z_i & - & 1 & \leq & 1 \\ x_{ij} & + & z_i & - & 1 & \leq & 1 \; - \; z_j \\ x_{ij} & - & & & 1 & \leq & 1 \\ x_{ij} & - & & & 1 & \leq & 1 \; - \; z_j. \end{array}\right. \tag{2.29}$$

Rearranjando os termos do sistema acima e acrescentando a restrição em $x_{ij}$ existente em (2.27), obtém-se a projeção $P'$ de $P$ no espaço definido pelas variáveis $x_{ij}, z_i$ e $z_j$:

$$P' = \left\{\begin{array}{rcrcrcl} x_{ij} & + & z_i & & & \leq & 2 \\ x_{ij} & + & z_i & + & z_j & \leq & 2 \\ x_{ij} & + & & & z_j & \leq & 2 \\ x_{ij} & & & & & \leq & 2 \\ x_{ij} & & & & & \leq & 1. \end{array}\right. \tag{2.30}$$

A partir da avaliação da projeção acima e, comparando as desigualdades obtidas com aquelas utilizadas por FUJIE (2003) na sua formulação de árvore, percebe-se a possibilidade de fortalecimento dos *bounds* a partir da inclusão da seguinte família de desigualdades:

$$x_{ij} + z_i + z_j \leq 2, \qquad \forall \; e = \{i, j\} \in E. \tag{2.31}$$

É importante esclarecer que, o resultado da aplicação do método de Fourier-Motzkin foi a projeção dos cortes (2.19) no espaço das variáveis não-direcionadas da formulação de FUJIE (2003) — não caracterizando portanto a projeção de toda a formulação direcionada, mas apenas de uma família de cortes.

As desigualdades apresentadas, que podem ser obtidas simplesmente a partir da

relaxação *surrogate* dos pares de restrições (2.19), também podem ser adaptadas para o caso direcionado, tomando especial interesse aquelas escritas em função da raiz da arborescência:

$$y_{rj} + z_r + z_j \leq 2, \qquad \forall\, (r,i) \in \delta_r^+. \tag{2.32}$$

### 2.4.1 Fortalecimento formulação não direcionada

A partir das desigualdades (2.31), derivadas anteriormente, somos capazes de fortalecer a formulação não direcionada para os problemas aqui tratados. Para isso, considere a região poliedral $\mathcal{R}_1^F$ obtida a partir de $\mathcal{R}_1$ através da substituição das restrições de caracterização das folhas (2.3) pelas novas desigualdades (2.31) introduzidas na seção anterior:

$$\sum_{e \in E} x_e \;=\; |V| - 1 \tag{2.33}$$

$$\sum_{e \in E(S)} x_e \;\leq\; |S| - 1, \quad S \subset V,\ |S| \geq 2 \tag{2.34}$$

$$\sum_{e \in \delta_i} x_e + (n_i - 1) z_i \;\leq\; n_i, \qquad i \in V \tag{2.35}$$

$$x_e + z_i + z_j \;\leq\; 2, \qquad e = \{i,j\} \in E \tag{2.36}$$

$$0 \leq\; x_e \;\leq 1 \qquad e \in E \tag{2.37}$$

$$0 \leq\; z_i \;\leq 1 \qquad i \in V. \tag{2.38}$$

Dessa forma, a formulação não direcionada fortalecida para o MLSTP será dada por:

$$max \left\{ \sum_{i \in V} z_i : (\mathbf{x}, \mathbf{z}) \in \mathcal{R}_1^F \cap \left( \mathbb{B}^{|E|} \times \mathbb{B}^{|V|} \right) \right\}. \tag{2.39}$$

Importante observar que, pela inclusão da desigualdade (2.36), a formulação fortalecida não contempla mais o caso trivial de grafos formados por apenas dois vértices ($i$ e $j$) e uma única aresta ($e = \{i,j\}$) entre eles. Instâncias com essa característica podem ser facilmente verificadas e a sua solução calculada de forma direta, de modo que não se justifica um tratamento da formulação para esses casos — o que fatalmente enfraqueceria os limitantes encontrados.

As adaptações necessárias para o LCMSTP são diretas, e não serão aqui demonstradas.

## 2.5 Nova formulação: escolha automática da raiz

Conforme apontado anteriormente, a incapacidade de precisar a priori qual o papel desempenhado pelo vértice raiz na solução do problema (se folha ou vértice interno da árvore) desencadeia uma série de tratamentos especiais às restrições da formulação (2.22). A formulação que será apresentada aqui busca automatizar o processo de seleção do vértice raiz da arborescência, que se torna, a partir de agora, mais uma decisão deixada a cargo do modelo.

Sejam definidas novas variáveis $\{r_i \in \mathbb{R} : i \in V\}$, responsáveis por indicar se o vértice $i$ será a raiz da arborescência. É possível adaptar a formulação (2.18) e definir uma nova região poliedral $\mathcal{R}_5$ como:

$$\sum_{j \in V} y_{ji} + r_i = 1, \qquad i \in V \tag{2.40}$$

$$\sum_{a \in A(S)} y_a \leq |S| - 1, \qquad S \subset V,\ |S| \geq 2 \tag{2.41}$$

$$\sum_{j \in V} y_{ij} + (n_i - 1)z_i \leq (n_i - 1) + r_i, \quad i \in V \tag{2.42}$$

$$\sum_{j \in V} y_{ij} + z_i \geq r_i + 1, \qquad i \in V \tag{2.43}$$

$$y_{ij} + z_i \leq 1 + r_i, \qquad (i,j) \in A, i \in V \tag{2.44}$$

$$\sum_{i \in V} r_i = 1 \tag{2.45}$$

$$y_a \geq 0, \qquad a \in A \tag{2.46}$$

$$z_i \geq 0, \qquad i \in V \tag{2.47}$$

$$r_i \geq 0, \qquad i \in V. \tag{2.48}$$

A partir desse momento, é possível reformular o MLSTP como:

$$\max \left\{ \sum_{i \in V} z_i : (\mathbf{r}, \mathbf{y}, \mathbf{z}) \in \mathcal{R}_5 \cap \left( \mathbb{B}^{|V|} \times \mathbb{B}^{|A|} \times \mathbb{B}^{|V|} \right) \right\}. \tag{2.49}$$

Essa reformulação pode ser também interpretada a partir da expansão do grafo $G = (V, E)$ através da inclusão de um vértice artificial, de índice $n + 1$. Dessa maneira é possível construir o grafo $G' = (V', E')$, em que $V' = V \cup \{n + 1\}$ e $E' = E \cup \{(i, n+1) : i \in V\}$ (quando necessário, considerar que os custos associados às novas arestas será $c_{i,n+1} = 0$).

A figura 2.1 ilustra esse processo.

A transformação do grafo expandido $G'$ no digrafo associado $D'$ é direta e segue os passos indicados na Seção 2.2.

Uma análise mais detalhada da formulação (2.49) revela que o vértice artifical

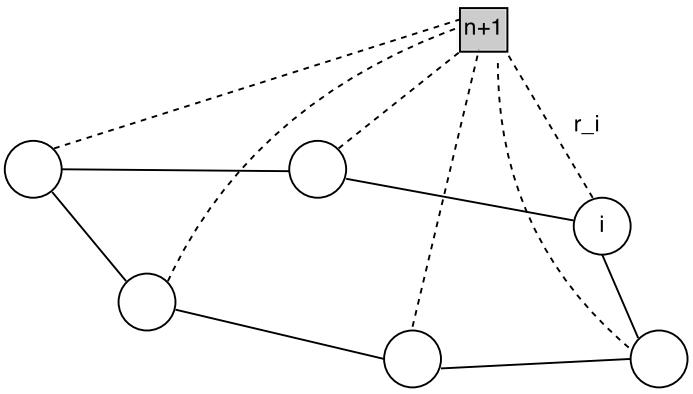


Figura 2.1: Grafo expandido

$n+1$ será a raiz da arborescência no digrafo expandido $D'$; que as variáveis $r_i$ inseridas indicarão a qual vértice $i \in V$ o vértice artifical estará ligado, e; finalmente, impor-se-á ao vértice artifical a condição de que ele seja uma folha na arborescência geradora de $D'$ — condição essa imposta pela restrição (2.45).

Como antes, é fácil observar que a adaptação de (2.49) para o LCMSTP é imediata.

### 2.5.1 Fortalecimento do modelo

A inclusão das variáveis $r_i$, embora elimine a dependência entre os *bounds* da relaxação linear e as raízes escolhidas como parâmetros, traz consigo a inconveniência de enfraquecer os cortes (2.19). Com objetivo de contornar (ou de pelo menos minimizar o referido enfraquecimento) foram conduzidos testes computacionais para verificar o potencial de fortalecimento da formulação (2.49) com a adição das restrições:

$$y_{ij} + y_{ji} + z_i + z_j \leq 2,\ a = (i,j) \in A. \tag{2.50}$$

Perceba que, mesmo com a inclusão das variáveis de escolha do vértice raiz ($\mathbf{r}$), dados dois vértices quaisquer $i$ e $j$ da arborescência, é sempre possível afirmar que ou ambos serão folhas na solução (nesse caso as respectivas variáveis $z_i$ e $z_j$ serão iguais a 1) e não haverá arco entre eles, ou então no máximo um delas será folha, sendo possível a existência de um arco entre eles. Essa observação permanece verdadeira mesmo que um dos vértices em questão seja uma raiz da arborescência (como por exemplo o vértice $i$): nesse caso, embora seja possível que $i$ seja folha, obrigando a existência de um único arco saindo em direção a algum vértice $j$, então esse vértice, sabidamente, não poderá também exercer o papel de folha da arborescência na solução — isso violaria a condição de conexidade da solução, uma vez que não

haveriam mais arcos saindo de $j$.

# Capítulo 3

# Algoritmos *branch-and-cut*

Esse capítulo é dedicado à apresentação dos algoritmos *branch-and-cut* desenvolvidos para os problemas aqui tratados. Todavia, antes de entrar nas questões mais práticas a respeito da aplicação desse método, será feita uma breve introdução para apresentar alguns dos principais conceitos relacionados à técnica empregada.

## 3.1 Introdução ao algoritmo de *branch-and-cut*

Em conformidade com os conceitos apresentados por MITCHEL (2002), os algoritmos *branch-and-cut* são um método exato de resolução de problemas de programação inteira (e inteira mista) baseados na combinação de duas outras técnicas: algoritmo *branch-and-bound* (ou de enumeração implícita), para a resolução do problema inteiro, aliado à técnica de planos de cortes, cujo objetivo é fortalecer a relaxação de programação linear do problema através da identificação de cortes que aproximem a formulação da sua envoltória convexa dos pontos inteiros.

Os algoritmos *branch-and-bound* tem em sua essência o paradigma de divisão e conquista: quebrar um problema em uma série de problemas menores que possam ser facilmente (e independentemente) resolvidos (geralmente a partir de algoritmos recursivos) para, posteriormente, serem recombinados, fornecendo a solução do problema original (CORMEN *et al.* (1990)).

Quando aplicado em um contexto de otimização combinatória, especialmente se utilizado na resolução de problemas do tipo $\mathcal{NP}$-difícil, o paradigma da divisão e conquista leva à pesquisa de todas as soluções viáveis para o problema em questão. Infelizmente, na ausência de mecanismos que evitem essa enumeração explícita, a aplicação de tal método seria amplamente reduzida e limitada, como indicado em WOLSEY (1998).

Diante dessa limitação surge a ideia de enumeração *implícita*, possível a partir da avaliação de limitantes (*bounds*) aplicados às soluções pesquisadas. Os percusores na utilização dessa técnica para a resolução de problemas de otimização inteira e

combinatória foram LAND e DOIG (1960). No entanto, foram LITTLE *et al.* (1963) quem cunharam o termo *branch-and-bound* ao descrever uma aplicação de sucesso do método na resolução de problema do caixeiro viajante. Nessa aplicação, a relaxação utilizada para o problema foi o chamado problema da designação (*assignment problem*). Finalmente, a maioria dos pacotes computacionais de otimização possuem uma implementação do método *branch-and-bound* com base na relaxação linear do problema inteiro associado, utilizando um esquema de *branching* (*i.e*, partição do espaço de soluções) semelhante àquele proposto por DAKIN (1965).

O estado do algoritmo de *branch-and-bound*, em relação à busca no espaço de soluções, pode ser representado através de um subconjunto de soluções ainda não exploradas, além do valor da melhor solução encontrada até o momento (MITCHEL (2002)). No início do algoritmo, apenas o subconjunto de soluções não exploradas existe, e a melhor solução encontrada até o momento tem valor de $\infty$ (para problemas de minimização, esse valor será $-\infty$).

Ainda segundo MITCHEL (2002), o subconjunto de soluções não exploradas pode ser representado como nós em uma árvore de busca, que inicialmente contém apenas o nó raiz. A cada iteração, o algoritmo processa um único nó dessa árvore. Essas iterações são divididas em duas partes principais (que dão nome ao algoritmo): uma etapa de *branching* e a outra de *bounding*.

A etapa de *branching* consiste em escolher um nó (*i.e*, um subproblema) ainda não explorado da árvore e dividi-lo em outros nós (geralmente dois), que representarão subespaços (desejavelmente disjuntos) da região de viabilidade associada ao problema original.

Para cada novo nó criado, a etapa de *branching* segue através da determinação da estimativa de um limitante (o mais justo possível) para o valor da melhor solução que se poderia encontrar caso fosse conduzida uma pesquisa exaustiva no subespaço representado pelo nó em questão. Tais valores são chamados limitantes duais. As técnicas mais utilizadas para obtenção dos *bounds* duais são aquelas de relaxação, mais notadamente, a de relaxação linear — podendo ser empregadas outras técnicas como relaxação e decomposição Lagrangeana, por exemplo.

Ainda durante essa etapa, são empregadas heurísticas que buscam encontrar soluções viáveis para o problema — os chamados limitantes primais. Essas heurísticas podem ser construídas de maneira *ad hoc*, especificamente para o problema sendo tratado, ou podem ser utilizados *frameworks* mais genéricos de construção de soluções, como por exemplo a aplicação de metaheurísticas (MITCHEL (2002), e WOLSEY (1998)).

Uma vez finalizado o cálculo dessas estimativas, tem início a etapa de *bounding*. Nela, serão conduzidas comparações para tentar encontrar subespaçoes (ou nós da árvore de busca) que sabidamente não levarão a soluções melhores do que a melhor

já encontrada.

Segundo WOLSEY (1998), as principais regras de *bound* empregadas são :

i **Poda por dominância ou qualidade**: ocorre quando o limitante dual encontrado para um nó da árvore for pior do que o valor de uma solução viável (limitante primal) que já se conheça.

ii **Poda por otimalidade**: ocorre quando o limitante dual encontrado para o ramo for igual ao valor do limitante primal fornecido por aquele ramo.

iii **Poda por inviabilidade**: ocorre quando o subespaço pesquisado não possui soluções viáveis para o problema em questão – não existe limitante dual ou ele é ilimitado.

Quando alguma das três condições acima é verificada, o subespaço de soluções (representado pelo nó em questão da árvore) pode ser descartado (ou podado), pois todas as suas soluções viáveis foram implicitamente enumeradas.

Importante ressaltar o papel fundamental da obtenção de bons valores para os limitantes, tanto duais quanto primais: quanto melhor os limitantes obtidos, mais efetiva será a etapa de *bounding*, acelerando a convergência do algoritmo.

Em boa parte das vezes, a obtenção de bons limitantes primais para o problema segue diretamente do emprego de métodos construtivos de soluções viáveis, aliado a técnicas de busca local nessas soluções. Esses algoritmos são necessariamente polinomiais no tamanho das entradas e observa-se, portanto, uma relação direta entre a qualidade dos limitantes obtidos e o tempo dado à busca local (algumas vezes traduzido como número de iterações).

Indo na direção contrária, porém, a derivação de bons limitantes duais nem sempre dependem da disponibilidade de tempo que se tem para sua obtenção. Esses valores estão intimamente relacionados à capacidade de se obter formulações mais justas para o problema tratado, o que depende da maneira como está caracterizada a região de viabilidade. Idealmente, quanto mais próxima a formulação for da descrição da envoltória convexa dos pontos inteiros, melhores serão os *bounds* duais observados. Infelizmente, para problemas do tipo $\mathcal{NP}$-difícil é praticamente nula a chance de caracterização da envoltória convexa (WOLSEY (1998)). Entretanto, a aplicação da técnica de planos de cortes possibilita encontrar uma aproximação dessa envoltória no entorno da solução ótima do problema. Em geral esse método preconiza a introdução, em cada iteração, de uma ou mais desigualdades válidas à relaxação linear do problema em questão (MACULAN (2002)).

A primeira referência à utilização dessa técnica é devido à DANTZIG *et al.* (1954) na resolução de uma instância de 49 cidades para o problema do caixeiro viajante. Na sequência, vieram os trabalho de GOMORY (1958), que propôs um

mecanismo de geração de cortes baseados em arredondamentos tendo, posteriormente (GOMORY (1960)), proposto um dispositivo mais completo para a geração de cortes algébricos para problemas de programação inteira. Nesse último trabalho, o autor demonstrou ainda que o método possui convergência finita se os custos da função objetivo forem inteiros. Sucedendo esses dois trabalhos, CHVÁTAL (1973) generalizou o procedimento proposto por Gomory e demonstrou que qualquer desigualdade válida para um problema de programação inteira pode ser obtida através da aplicação de um número finito de vezes do procedimento de geração dos cortes algébricos.

Entretanto, as dificuldades computacionais relacionadas à obtenção dos planos de cortes fez com que a utilização da técnica perdesse espaço para outras que pareciam mais promissoras — inclusive para a própria técnica de *branch-and-bound*, que tornou-se bastante popular nos anos 70 (MITCHEL (2002)). Somente nos anos 80 o interesse pelo método ressurgiu, em boa parte devido a avanços na teoria poliedral. Isso permitiu a geração de planos de cortes poliedrais, especificos para algumas famílias de problemas, em geral muito mais fortes do que aqueles algébricos (MITCHEL (2002)).

De acordo com MITCHEL (2002), seria possível, na teoria, que a utilização única do algoritmo de planos de cortes fosse suficiente para a resolução de problemas de programação inteira. Na prática, observa-se que esse método sofre de um efeito análogo ao de uma saturação, denominado *tailing-off*, no qual se verifica que o acréscimo de um grande número de cortes à formulação do problema relaxado não é mais suficiente para que se obtenha melhorias no *bound*. Nesse contexto, o emprego dessa técnica em conjunto com o *branch-and-bound* se tornou uma alternativa bastante interessante à resolução desses problemas. Alguns dos exemplos mais bem sucedidos da aplicação do algoritmo de *branch-and-cut* para a resolução de prolemas de programação inteira talvez sejam aqueles ligados ao problema do caixeiro viajante (PADBERG e RINALDI (1987), (PADBERG e RINALDI (1991)) e (GRÖTSCHEL e HOLLAND (1991), APPLEGATE *et al.* (2006)).

## 3.2 Um algoritmo *Branch-and-cut* para o MLSTP

Algumas vezes, uma formulação para um problema de otimização inteira envolve um número exponencialmente grande de restrições, inviabilizando o tratamento direto do modelo por qualquer um dos pacotes computacionais existentes.

Considere a seguinte formulação ($P$) para um problema geral de programação inteira:

$$\text{(P):} \quad \max \quad x_0 = \mathbf{c}^T\mathbf{x} \tag{3.1}$$

sujeito a:

$$A\mathbf{x} \leq \mathbf{b} \tag{3.2}$$

$$E\mathbf{x} \leq \mathbf{f} \tag{3.3}$$

$$\mathbf{x} \geq 0 \tag{3.4}$$

$$x_j \in \mathbb{Z}, \qquad \forall\, j \in S \subseteq \{1, 2, \ldots, n\}, \tag{3.5}$$

onde (3.3) pode conter um número exponencial de restrições. Para a obtenção da relaxação linear dessa formulação, se impõem então, nesses casos, a necessidade de desenvolver dispositivos que permitam o tratamento implícito dessas restrições.

Em linhas gerais, o que se busca é a possibilidade de identificar de maneira eficiente (*i.e.* através da aplicação de um algoritmo polinomial) se um dado ponto de $\mathbb{R}^n_+$ viola alguma das restrições (3.3). Caso se conheça esse algoritmo, torna-se atraente trabalhar com uma formulação ($\bar{P}$), relaxada pela retirada das restrições complicadoras, e que será iterativamente reforçada pela adição de restrições violadas encontradas *on-the-fly*.

Esse raciocínio só é possível pois, a completa caracterização da região de viabilidade, através da listagem exaustiva de todas as suas restrições, é a *priori* desnecessária. Para a maioria das aplicações de otimização, basta que se tenha uma caracterização bastante precisa da vizinhança de pelo menos uma das soluções ótimas.

O mecanismo descrito anteriormente remonta aos princípios preconizados pelo algoritmo de planos de cortes, com a diferença de que, agora, esses cortes não serão utilizados apenas para aproximar a formulação da envoltória convexa dos seus pontos inteiros, mas sim, para garantir que a relaxação linear de uma dada formulação seja obtida.

Um dos trabalhos pioneiros na resolução de um problema inteiro a partir de uma relaxação obtida pela exclusão de toda uma família de restrições é o de LITTLE *et al.* (1963). Nesse trabalho, os autores partem de uma relaxação de *assignment* para o problema do caixeiro viajante, obtida a partir da remoção das restrições de eliminação de subrotas. A medida em que se identificam a existência de circuitos ilegais nas soluções de *assignment* encontradas, são então inseridas, explicitamente na formulação, as restrições de eliminação de subrotas a eles correspondentes. Isso é feito na expectativa de que, progressivamente, as soluções encontradas se tornem livres desses circuitos.

O pseudo-código de um algoritmo simplificado de *branch-and-cut* é apresentado

a seguir:

BRANCH-AND-CUT($P$)

```
1  lb = 0
2  ub = ∞
3  bb_tree = P̄
4  x̂ = 0
5  while lb < ub and bb_tree.notEmpty()
6        SOLVE(P̄, x̂, lb)
7        C = SEPARATION-ROUTINE(x̂)
8        if |C| > 0
9              ADD-CUTS(P̄, C)
10       else
11             BRANCHING(bb_tree, x̂)
12 if ⌈lb⌉ == ub
13       return x̂
14 else
15       return ∅
```

### 3.2.1 Desigualdades de eliminação de subrotas - SECs

Embora existam exceções, tipicamente, formulações para árvores em grafos apresentam um número exponencialmente grande de restrições, e o caso não é diferente para o MLSTP: repare que a família de desigualdades (2.41), conhecidas na literatura como *desigualdades de eliminação de subrotas* (do inglês *subtour elimination constraints* – SECs) possui $\mathcal{O}\left(2^{|V|}\right)$ restrições, uma vez que a propriedade deve ser verificada para cada um dos subconjuntos $S \subset V$. Essa família de desigualdades é bastante conhecida e remonta ao trabalho seminal de DANTZIG *et al.* (1954) na resolução do problema do caixeiro viajante. Conceitualmente, essas desigualdades estão relacionadas à eliminação de ciclos ilegais, isso é, ciclos que não envolvam todos os vértices do grafo. Note que, para um problema que tem uma árvore geradora como subestrutura básica, até mesmo um ciclo envolvendo todos os vértices de $G$ é ilegal.

Considere um grafo conexo direcionado $D = (V, A)$, onde $V$ é o conjunto de vértices e $A$ é o conjunto de arcos. As desigualdades de eliminação de subrotas podem ser matematicamente modeladas como:

$$\sum_{a \in A(S)} y_a \leq |S| - 1, \forall S \subseteq V \tag{3.6}$$

onde $A(S)$ é o subconjunto de arcos com ambas as extremidades em $S$.

**Algoritmo de separação das SECs**

No contexto de otimização combinatória, para qualquer problema $P$, formulado genericamente como $max\{\mathbf{cx} : \mathbf{x} \in X \subseteq \mathbb{R}^n\}$, é possível associar um outro problema, chamado problema da separação (WOLSEY (1998)):

*Dado* $\hat{\mathbf{x}} \in \mathbb{R}^n$*, é verdade que* $\hat{\mathbf{x}} \in conv(X)$*? Senão, encontre uma desigualdade* $\pi\mathbf{x} \leq \pi_0$ *que seja satisfeita por todos os pontos de* $X$*, e violada por* $\hat{\mathbf{x}}$.

Denominam-se *algoritmos de separação*, aqueles empregados na identificação das desigualdades $\pi\mathbf{x} \leq \pi_0$ violadas por $\hat{\mathbf{x}}$.

Para assegurar que as soluções encontradas para a relaxação linear de (2.49), retiradas as SECs, sejam acíclicas, utilizamos o algoritmo de separação proposto por PADBERG e WOLSEY (1983). Nesse trabalho os autores demonstraram um resultado não trivial de que o problema da separação das desigualdades (2.41) poderia ser eficientemente resolvido a partir do cálculo de no máximo $(|V| - 2)$ fluxos máximos em um grafo definido apropriadamente.

Seja $\hat{\mathbf{x}} = (\hat{\mathbf{r}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$ uma solução para a relaxação linear da formulação (2.49), excluídas as SECs — frisando que $\mathbf{y}$ é utilizado para indicar se um determinado arco do grafo está na solução. A partir do grafo de suporte de $\hat{\mathbf{x}}$, é construída uma rede capacitada $H = (V', A)$ da seguinte maneira:

i Para cada arco $a = (i, j)$ são atribuídas capacidades $u_{ij} = \hat{y}_{ij}$.

ii Para cada vértice $v \in V$, calcula-se as quantidades $b_v = \sum_{j \in V} (y_{vj} + y_{jv})$.

iii Acrescente-se a $H$ um vértice artifical, $s$, denominado origem, de onde saem arcos $(s, v)$, $v \in V$, com capacidades $u_{sv} = \max\left\{\frac{1}{2}b_v - 1; 0\right\}$.

iv Acrescente-se a $H$ um outro vértice aritfical, $t$, denominado sorvedouro, para onde apontaram arcos $(v, t)$, $v \in V$, com capacidades $u_{vt} = \max\left\{1 - \frac{1}{2}b_v; 0\right\}$.

Sobre o grafo $H$ construído deverão ser resolvidos iterativamente $|V| - 2$ problemas do fluxo máximo entre os pares de vértices $s$ e $t$, sendo que as capacidades associadas aos arcos de $H$ deverão ser atualizadas a cada iteração, segundo a seguinte regra: para o $k$-ésimo problema, ajusta-se temporariamente a capacidade $u_{sk} = \infty$, e, se $k \geq 2$, então $u_{v,t} = \infty$, para todo $v = 1, 2, \ldots, k - 1$.

Para cada problema de fluxo máximo resolvido no passo anterior, associa-se um corte $A(S, V \backslash S)$. Sempre que $|S| \geq 2$ e $\sum_{a \in A(S)} \hat{y}_a > |S| - 1$, então uma desigualdade de eliminação de subrotas foi encontrada e deverá ser incorporada à relaxação linear corrente. Caso não se verifique a violação de nenhuma dessas desigualdades ao término da resolução dos $(|V| - 2)$ problemas de fluxo máximo, então é possível concluir que não mais existem desigualdades de eliminação de subrotas violadas. Se, ainda assim, $(\hat{\mathbf{r}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$ for fracionário, deve-se então partir para a etapa de *branching*.

Para encontrar o fluxo máximo em um grafo direcionado existem diversas referências na literatura. Umas das primeiras nesse sentido é o algoritmo de Ford-Fulkerson (FORD e FULKERSON (1962)). Embora não seja o algoritmo mais eficiente para a resolução do problema, esse algoritmo foi de extrema importância para o desenvolvimento da área de fluxos em redes. Em particular, introduziu três dos principais conceitos posteriormente utilizados para a construção de novos algoritmos: fluxos (ou redes) residuais, caminhos de aumento e cortes (CORMEN *et al.* (1990)).

Para encontrar fluxos máximos, utilizou-se o método de *preflow-push*, desenvolvido inicialmente por GOLDBERG (1982) e posteriormente refinado por GOLDBERG e TARJAN (1986). Uma descrição bastante detalhada do algoritmo, incluindo seu pseudo-código, pode ser encontrada em CORMEN *et al.* (1990).

### 3.2.2 *Cut-sets*

Existe uma outra família de desigualdades, conhecidas na literatura como *cut sets*, que podem ser utilizadas como alternativas às SECs apresentadas anteriormente. Tais desigualdades, que foram concebidas conceitualmente para garantir a conexidade da solução, têm como efeito prático a eliminação das ciclos que são incompatíveis com uma topologia de árvore.

Dado um grafo $G = (V, E)$, define-se um corte $K = E(S, \bar{S}) \subseteq E$, onde $\bar{S} = V \backslash S$, como qualquer partição do conjunto $V$ nos conjuntos $S$ e $\bar{S}$. Diz-se que uma aresta $(u, v) \in E$ atravessa o corte $C$ se uma das suas extremidades estiver em $S$ e a outra, em $\bar{S}$ (CORMEN *et al.* (1990)). Ao conjunto de arestas que atravessam o corte, dá-se o nome de *cut-set*.

Para uma árvore $T \subseteq G$ e para todo corte $E(S, \bar{S})$ existe pelo menos uma aresta que atravessa esse corte. Essa observação dá origem à desigualdades de *cut-sets* apresentadas à seguir:

$$\sum_{(u,v):\, e \in S,\, v \in \bar{S}} x_{u,v} \geq 1, \qquad \forall\, E(S, \bar{S}),\ S \subset V,\ S \neq \emptyset. \tag{3.7}$$

Em termos práticos, a utilização das desigualdades do tipo SEC ou *cut-sets* leverão às relaxações lineares de nossas formulações para MLSTP/LCMSTP. A opção por uma ou outra dependerá primordialmente dos aspectos numéricos de Programação Linear (em especial de características do Método Simplex). Em geral, para problemas em que a estrutura dominante é aquela de árvores, as *cut-sets* parecem estar relacionadas a matrizes menos densas e, por sua vez, resultam em algoritmos *branch-and-cut* mais rápidos.

### Algoritmo de separação dos *cut-sets*

Diferentemente do que ocorria para a separação das SECs violadas, o algoritmo de separação de *cut-sets* é mais intuitivo e segue diretamente da análise da desigualdade (3.7). Considere novamente $\hat{\mathbf{x}} = (\hat{\mathbf{r}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$, a solução de relaxação linear da formulação (2.49) relaxada pela remoção das desigualdades de eliminação de subrotas (2.41). Novamente, será construída uma rede capacitada $H = (V, A)$ da seguinte maneira:

i Para cada $\hat{y}_{ij} > 0$, inserir em $H$ o arco $a = (i, j) \in A$, com capacidade $f_{ij} = \hat{y}_{ij}$.

Utilizando o algoritmo de fluxo máximo entre todos os pares de vértices do grafo, com base no resultado do Teorema do Fluxo Máximo-Corte Mínimo (introduzido independentemente por FORD e FULKERSON (1956) e ELIAS *et al.* (1956) – que assegura que o fluxo máximo entre dois vértices de um grafo é igual ao valor do corte mínimo que os separam), um *cut-set* violado será encontrado sempre que o valor do fluxo máximo for inferior a uma unidade.

A prova da corretude do algortimo de separação acima pode ser feita por contradição: imagine que exista pelo menos um corte $A(S, \bar{S})$ de $H$ que viole (3.7), isso é, o somatório dos arcos que atravessam o corte é inferior a uma unidade. A existência de tal corte garante que há no grafo suporte pelo menos dois vértices $i \in S$ e $j \in \bar{S}$ posicionados em lados opostos do corte. Na $k$-ésima iteração do algortimo de separação, será calculado o fluxo máximo entre os pares de vértices $i$ e $j$. Caso o algoritmo seja capaz de encontrar um fluxo entre esses vértices com valor superior a uma unidade, então, essa seria claramente uma contradição da hipótese inicial de existência de um *cut-set* violado, haja visto que, pelo teorema do fluxo máximo-corte mínimo, essas duas quantidades — fluxo e corte — deveriam ser iguais.

Sendo $n = |V|$ e $m = |A|$, a análise de complexidade do pior caso para o algoritmo delineado acima revela que serão necessárias $\mathcal{O}(n^2)$ chamadas ao algoritmo de fluxo máximo, que na sua implementação mais eficiente, para grafos esparsos, tem complexidade $\mathcal{O}\left(nm \log\left(\frac{n^2}{m}\right)\right)$ (GOLDBERG e TARJAN (1986)).

Como o desempenho desse algoritmo de separação não é encorajador, uma alternativa à sua utilização seria a construção de uma heurística de separação baseada nos princípio apontados anteriormente. Entretanto, com a utilização da heurística é possível que cortes violados não sejam encontrados, mesmo que eles existam.

Para evitar a utilização de um procedimento heurístico, e como alternativa ao primeiro algoritmo de separação apontado, utilizou-se nesse trabalho um algortimo incrivelmente simples proposto por STOER e WAGNER (1997) para encontrar o corte mínimo em um grafo não-orientado. Diferentemente dos algoritmos clássicos de resolução do problema (FORD e FULKERSON (1956), HAO e ORLIN (1992), NAGAMOCHI e IBARAKI (1992)), o algorimo utilizado tem foco em encontrar de

maneira eficiente os vértices $s$ e $t$, candidatos à fonte e sorvedouro do fluxo, e em como calcular o valor do corte mínimo separando esses dois vértices. O algoritmo proposto está fundamentado no seguinte teorema:

**Teorema 3.2.1** *Sejam $s$ e $t$ dois vértices de um grafo $G$. Será chamado $G\backslash\{s,t\}$ o grafo obtido a partir da aglutinação (*merge*) dos vértices $s$ e $t$. Então, um corte mínimo de $G$ poderá ser obtido através do menor entre (i) o valor do corte mínimo separando $s$ e $t$ e (ii) o corte mínimo de $G\backslash\{s,t\}$.*

**Prova** O teorema se mantém válido desde que exista um corte mínimo de $G$ que separe os vertices $s$ e $t$, situação para a qual o $s$-$t$-corte de $G$ é o menor corte possível em $G$. Em caso negativo, então o corte mínimo do novo grafo $G\backslash\{s,t\}$, será o corte mínimo de $G$. E o teorema segue verdadeiro. ∎

Com base no teorema enunciado acima, é possível perceber que um procedimento para encontrar, arbitrariamente, o menor $s$-$t$-corte do grafo poderá ser utilizado de maneira recursiva para encontrar o corte mínimo desse grafo. O algoritmo abaixo, conhecido na literatura como *maximum adjacency search* conduz à criação de um $s$-$t$-corte mínimo no grafo:

```
MinCutPhase(G)
1  w = ArbitraryVertex(G)
2  W = {w}
3  while W ≠ V
4        w = Tightly-Connected(V, W)        // Retorna o vértice de V
                                            // mais fortemente conectado
                                            // aos vértices de W
5        W = W ∪ {w}                        // Adiciona o vértice w
                                            // ao conjunto W
   // os vértices s e t serão os dois últimos vértices na
   // ordem em que foram adicionados ao conjunto W
6  return Cut(W − {t}, t)
```

O método Tightly-Connected, utilizado na construção do corte mínimo (no laço principal do algoritmo MinCutPhase), retorna o vértice do conjunto $V$ que ainda não foi adicionado a $W$, cuja soma dos pesos das arestas que chegam aos demais vértices de $W$ seja a maior possível.

Com base nos procedimentos apresentados, segue o algoritmo proposto por STOER e WAGNER (1997) para calcular o corte mínimo do grafo:

STOER-WAGNERSEPARATION($\hat{y}$)

1 $min\_cut\_val = \infty$
2 $S^* = \emptyset$
3 $H$ = BUILDSUPPORTGRAPH($\hat{y}$) // Construção do grafo suporte
4 **while** $|V| > 1$
5 $(S,T)$ = MINCUTPHASE($H$) // Encontra o menor $(S,T)$ corte da iteração
6 $cut\_val$ = EVALUATECUT($S,T$) // Avalia o valor do corte
7 **if** $cut\_val < min\_cut\_val$ // Atualização
8 $S^* = S$
9 $min\_cut\_val = cut\_val$
10 MERGE($H,s,t$) // Combina os vértices $s$ e $t$
11 **return** $S^*$ // Retorna o corte mínimo

O precedimento MERGE combina os dois vértices $s$ e $t$ recebidos como entrada em um único vértice $\{s,t\}$ e, qualquer aresta que possuia um desses dois vértices como ponto extremo é substituída por uma nova aresta cujo peso será igual à soma dos pesos das duas arestas originais em $G$.

MERGE($G,s,t$)

1 $G = G \backslash \{s,t\}$ // Remove os vértices $s$ e $t$ do grafo
2 $G = G \cup \{st\}$ // Adiciona um novo vértice $st$
3 **for** $v \in V : v \neq \{st\}$
4 $c_{st,v} = c_{s,v} + c_{t,v}$ // Atualiza as capacidades das arestas

A implementação eficiente do método TIGHTLY-CONNECTED (através da utilização de *heaps*) permite que a complexidade do pior caso para o algoritmo STOER-WAGNERSEPARATION seja $\mathcal{O}(nm + n^2 \log n)$.

Tal como apresentado, a utilização do algoritmo STOER-WAGNERSEPARATION assegura que, caso haja algum *cut-set* violado pela solução $\hat{\mathbf{x}}$, então será encontrado aquele corte de maior violação global. Entretanto, sob o aspecto algorítmico do *branch-and-cut*, a identificação de um único corte violado por iteração pode não ser benéfica, fazendo com que o processo de *tailing-off* seja acelerado. Dessa maneira, na busca por outros *cut-sets* violados, foi proposta uma ligeira alteração da verificação efetuada na linha 7 do algoritmo: sempre que o valor do *cut-set* da iteração for inferior a uma unidade, isso significa que uma desigualdade violada foi encontrada e deveria ser armazenada.

### 3.2.3 Desigualdades *F-cuts*

Visando reforçar as relaxações lineares de nossas formulações, foi utilizada uma adaptação da separação proposta por LUCENA *et al.* (2010) para os cortes in-

troduzidos por FUJIE (2004). Essas desigualdades, apresentadas em (2.23), serão doravante denominadas *F-cuts*. A sua utilização é bastante atraente, tendo sido provado por FUJIE (2004) que elas induzem facetas para o politopo do MLSTP. Vale ressaltar que as *F-cuts* foram utilizadas com sucesso em LUCENA *et al.* (2010).

Assim como ocorre para as restrições de eliminação de subciclos, o conjunto das *F-cuts* contém um número exponencial de cortes. Dessa maneira, a abordagem mais indicada para a sua utilização é aplicação do *framework* de planos de cortes, identificando *on-the-fly* desigualdades que sejam violados pela solução corrente, adicionando-as à relaxação em mãos.

### Algoritmo de separação dos *F-cuts*

A separação das desigualdades do tipo *F-cuts* violadas será conduzida a partir da adaptação do procedimento proposto por LUCENA *et al.* (2010).

Para $(\hat{\mathbf{r}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}) \in \mathcal{R}_5$ definimos um vetor auxiliar $\hat{\mathbf{x}}$ cujas componentes serão $\hat{x}_{ij} = \hat{y}_{ij} + \hat{y}_{ji}$.

Para todo $i \in V$, tome o subconjunto de elementos $\hat{x}_i = \{\hat{x}_e : e = (i,k), \forall k \neq i, k \in V\}$, cuja cardinalidade é denotada por $n_i$. Ordene seus elementos em ordem decrescente dos valores, obtendo $\hat{x}_i = \{\hat{x}_{e_1}, \hat{x}_{e_2}, \ldots, \hat{x}_{e_{n_i}}\}$. Em seguida, para cada $i \in V$ e $l \in \{2, 3, \ldots, n_i\}$, calcule $val_i = \left(\sum_{k=1}^{l} \hat{x}_{e_k} + (l-1)z_i\right)$. Dessa maneira, obter-se-á o conjunto $F$ de cardinalidade $l$ que maximiza o valor do lado esquerdo da desigualdade (2.23). Sempre que $val_i$ exceder o valor de $l$, então uma desigualdade (2.23), violada pela solução corrente $(\hat{\mathbf{r}}, \hat{\mathbf{y}}, \hat{\mathbf{x}})$, terá sido encontrada. Caso contrário, não existirá nenhuma combinação de $i$ e $l$ capaz de violar (2.23). O pseudo-código do algoritmo é dado abaixo:

F-CUT-SEPARATION$(\hat{y}, \hat{z})$

```
1   x_i.clear                                  // vetor contendo
                                               // com uma das extremidades em i
2   for i ∈ V
3       x_i = BUILD(ŷ, i)                      // ∀j ∈ δ_i : x_ij = ŷ_ij + ŷ_ji
4       SORT(x_i)
5       val_i = x_i.first
6       for l = 2 to n_i
7           val_i = val_i + x_i.at(l)
8           if val_i + (l − 1) · z_i ≤ l
9               BREAK                          // não existirá mais nenhum corte violado
10          else
11              GENERATE-CUT(x_i, l)           // encontrado corte de cardinalidade l
```

É fácil verificar que, para cada $i \in V$, o algoritmo de separação delineado anteriormente tem complexidade do pior caso dada por $\mathcal{O}(n_i \log n_i)$.

## 3.3 Experimentos computacionais

Todos os experimentos conduzidos foram executados em uma máquina Intel Core i7 2620M, de 2.70 GHz e 8 GB de RAM. Os modelos desenvolvidos foram implementados em linguagem C++, utilizando as bibliotecas *Callable Library* IBM (2013) e LEMON (*Library of Efficient Models and Optimization in Networks*, COIN-OR (2003-2013)) e resolvidos através da utilização do IBM ILOG CPLEX *Studio Academic* 12.4.

Em todos os experimentos realizados, e nos quais foram necessários a utilização do CPLEX, tomou-se o devido cuidado de desabilitar a geração automática de cortes (funcionalidade presente no pacote) para que se pudesse verificar de maneira mais consistente o impacto das familias de desigualdades utilizadas. A gestão da árvore de procura (nos algoritmos *branch-and-cut*) foi deixada à cargo do próprio CPLEX, sendo que nenhuma estratégia especial de *branching* fosse explicitamente imposta. Além disso, foram desabilitadas todas as heurísticas do CPLEX, e sistema foi configurado para executar com uma única *thread* (não sendo permitido paralelismos).

### 3.3.1 Experimentos com o MLSTP

Para testar o potencial da formulação (2.49) proposta nesse trabalho optou-se por conduzir uma análise comparativa dos *bounds* fornecidos por essa nova formulção e aqueles associados ao trabalho de LUCENA *et al.* (2010). Adicionalmente, foi decidido que seria utilizado o mesmo conjunto de instâncias proposto pelos autores, cujo mecanismo gerador encontra-se descrito em LUCENA *et al.* (2010).

A Tabela 3.1 sintetiza o resultado dos experimentos conduzidos com as instâncias do *max leaf*. As duas primeiras colunas, **n** e **d** representam, respectivamente, o número de vértices no grafo e a densidade de arestas. Na sequência, tem-se a coluna **opt** que traz o número máximo de folhas para uma árvore geradora no grafo (*i.e.* a solução ótima do problema MLSTP). A quarta e quinta colunas da tabela trazem um comparativo dos valores da relaxação linear da formulação (2.18) (fortalecida pelo acréscimo das restrições (2.19) e (2.23)) com aqueles da formulação (2.49). As duas últimas colunas da tabela apresentam uma medida da distância entre os limitantes encontrados pelas formulações e a solução ótima do problema.

Importante ressaltar que os valores informados da solução ótima das instâncias e de relaxação linear para a formulação (2.18) foram apresentados por LUCENA *et al.* (2010), tendo sidos tão somente reproduzidos nesse trabalho.

| | | | lp | | gap (%) | |
|---|---|---|---|---|---|---|
| n | d | opt | dir | automatic | dir | automatic |
| 50 | 5 | 19 | 19 | 19,27 | 0,00 | 1,42 |
| 50 | 20 | 43 | 45,12 | 45,78 | 4,93 | 5,30 |
| 50 | 50 | 47 | 48,18 | 48,33 | 2,51 | 4,53 |
| 70 | 5 | 43 | 44,41 | 44,95 | 3,28 | 3,60 |
| 70 | 20 | 63 | 64,88 | 65,79 | 2,98 | 4,59 |
| 70 | 50 | 67 | 68,05 | 68,85 | 1,57 | 3,21 |
| 120 | 5 | 95 | 97,52 | 97,53 | 2,65 | 2,98 |
| 120 | 20 | 112 | 114,93 | 116,28 | 2,62 | 3,82 |
| 120 | 50 | 116 | 118,12 | 119,99 | 1,83 | 4,04 |
| | | | | média | 2,49 | 3,72 |

Tabela 3.1: Comparação dos *bounds* de relaxação linear – MLSTP

Pelo que se é possível concluir da análise da Tabela 3.1, a formulação (2.18) foi capaz de fornecer os melhores *bounds* para as instâncias testadas (apresentando um *gap* de otimalidade médio em torno de $2,5\%$), embora o desempenho da formulação (2.49) (que forneceu *bounds* a uma distância média de $3,7\%$ das soluções ótimas testadas) a coloque como uma alternativa promissora para a resolução do *max leaf*.

As Tabelas 3.2 e 3.3 tem formatos idênticos e apresentam a performance dos testes conduzidos utilizando a formulação (2.18), fortalecida pelas desigualdades (2.19). A diferença entre elas se dá simplesmente na regra utilizada para escolha dos vértices raizes das arborescências (indicados na coluna **r**). A Tabela 3.2 foi construída a partir dos resultados obtidos pelos vértices capazes de fornecer a pior relaxação linear para o modelo em cada instância (valores esses são apresentados na coluna **root relax.**). Na direção oposta, a Tabela 3.3 foi construída a partir dos resultados fornecidos pelos vértices mais atraentes, isso é, aqueles capazes de fornecer a melhor relaxação linear para o modelo. Sintetizando, buscamos contrapor o melhor e o pior caso respectivamente para a formulação (2.18).

Ademais, as duas primeiras colunas das tabelas descrevem as características das instâncias (a quantidade de vértices e a densidade do grafo). A coluna **opt** apresenta o número máximo de folhas, solução ótima da instância. As colunas que se seguem, resumem a performance propriamente dita do algoritmo: **lb** e **ub** apresentam respectivamente os melhores *lower bound* e o *upper bound* encontrados; a coluna **nodes** apresenta o número de nós da árvore B&B pesquisado. As colunas **F-cuts** e **cut sets** apresentam as quantidades dos respectivos cortes identificados e, a última coluna, **time** mostra o tempo em segundos gasto pelo algoritmo. Todas os

| n | d | r | opt | root relax. | lb | ub | nodes | F-cuts | cut sets | time (s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | 5 | 8 | 19 | 20,11 | 19 | 19,00 | 0 | 68 | 268 | 0,12 | ** |
| 50 | 20 | 15 | 43 | 45,36 | 43 | 43,00 | 134 | 3564 | 1338 | 27,08 | ** |
| 50 | 50 | 15 | 47 | 48,17 | 47 | 47,00 | 9 | 612 | 414 | 5,07 | ** |
| 70 | 5 | 65 | 43 | 45,25 | 43 | 43,00 | 96 | 556 | 1781 | 13,04 | ** |
| 70 | 20 | 40 | 63 | 65,02 | 63 | 63,00 | 959 | 22659 | 16541 | 2682,33 | ** |
| 70 | 50 | 33 | 67 | 68,13 | 67 | 67,00 | 43 | 1191 | 604 | 15,12 | ** |
| 120 | 5 | 92 | 95 | 98,77 | 34 | 98,13 | 1439 | 5786 | 42813 | 3600,30 | |
| 120 | 20 | 24 | 112 | 115,03 | 111 | 114,39 | 300 | 37419 | 14372 | 3600,58 | |
| 120 | 50 | 23 | 116 | 118,12 | 116 | 118,03 | 127 | 37091 | 8983 | 3600,10 | * |

Tabela 3.2: Performance da pior raiz para a formulação direcionada nas instâncias do MLSTP

resultados apresentados correspondem aos melhores valores encontrados dentro de um limite de tempo de 3600 segundos. Para facilitar a visualização, após a última coluna de cada tabela utilizamos uma marcação para indicar quando o algoritmo fora capaz de resolver a instância em sua otimalidade dentro da restrição de tempo imposta (**) ou quando o modelo encontrou a solução ótima, não tendo sido capaz de provar a sua otimalidade (*).

Fica evidente, pela análise da tabela, que a escolha da raiz afeta de maneira determinante a performance do algoritmo, tanto na sua capacidade de resolução das instâncias maiores (e mais difíceis) do problema, quanto no esforço empreendido na resolução (vide as diferenças do número de nós pesquisados, bem como da quantidade de desigualdades violadas introduzidas ao modelo, e o respectivo acréscimo no tempo de resolução dessas instâncias). Como regral geral, quanto melhor o limite dual encontrado na resolução do nó raiz da árvore B&B, melhor o desempenho geral do algoritmo, embora essa diferença apresente tendência de redução em função do aumento da densidade do grafo associado à instância — nota-se que a medida que o grafo se torna mais denso, as diferenças de performance entre o pior e o melhor caso tornam-se menos acentuadas.

A Tabela 3.4 apresenta a performance da formulação (2.49) nas instâncias do MLSTP. Essa tabela tem exatamente a mesma configuração das Tabelas 3.2 e 3.3, exceto pela ausência da coluna **r**, utilizada para indicar os vértices escolhidos como raizes.

A análise comparativa das tabelas apresentadas permite concluir que os resultados obtidos pela utilização da formulação (2.49) foram bastante encorajadores, uma vez que a sua performance foi capaz de rivalizar com a performance da formulação

| n | d | r | opt | root relax. | lb | ub | nodes | F-cuts | cut sets | time (s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | 5 | 20 | 19 | 19,50 | 19 | 19,00 | 0 | 25 | 55 | 0,06 | ** |
| 50 | 20 | 1 | 43 | 44,80 | 43 | 43,00 | 55 | 1157 | 367 | 3,32 | ** |
| 50 | 50 | 27 | 47 | 47,98 | 47 | 47,00 | 3 | 234 | 232 | 0,97 | ** |
| 70 | 5 | 54 | 43 | 44,15 | 43 | 43,00 | 5 | 164 | 112 | 0,59 | ** |
| 70 | 20 | 7 | 63 | 64,76 | 63 | 63,00 | 497 | 8828 | 4026 | 327,87 | ** |
| 70 | 50 | 59 | 67 | 67,97 | 67 | 67,00 | 7 | 1031 | 360 | 4,27 | ** |
| 120 | 5 | 105 | 95 | 97,33 | 95 | 95,00 | 138 | 1823 | 605 | 17,75 | ** |
| 120 | 20 | 67 | 112 | 114,75 | 111 | 114,17 | 513 | 17929 | 8458 | 3600,14 | |
| 120 | 50 | 36 | 116 | 118,02 | 116 | 117,97 | 186 | 21379 | 4197 | 3600,02 | * |

Tabela 3.3: Performance da melhor raiz para a formulação direcionada nas instâncias do MLSTP

(2.18), no seu melhor caso. Dentro das restrições de tempo impostas ao algoritmo de B&B, ela se mostrou capaz de provar a otimalidade das mesmas instâncias resolvidas pela utilização da formulação (2.18).

Importante ainda destacar a performance dessa formulação para as instâncias mais difíceis testadas (aquelas com 120 vértices). Embora nenhuma das formulações implementadas tivesse sido capaz de provar a otimalidade das soluções encontradas para essas instâncias, ao término do tempo, os gaps de otimalidade fornecidos pela formulação (2.49) foram ligeiramente melhores do que aqueles encontrados pela formulação (2.18), indicando que a nova formulação foi capaz de recuperar a ligeira desvantagem inicial (limitante dual fornecido no nó zero da árvore), reforçando a ideia de que a formulação proposta seja realmente promissora.

### 3.3.2 Experimentos com o LCMSTP

Para os experimentos com o LCMSTP utilizamos por um conjunto de 16 instâncias geradas aleatoriamente através de uma rotina especificamente concebida neste trabalho. Essas instâncias variam de densidade e tamanho, e foram escolhidas na tentativa de capturar diferentes nuanças do problema. Para sua criação foram utilizados pontos espalhados no plano Euclidiano, contidos em um quadrado com lados de comprimento fixados a priori. A esses pontos associam-se os vértices do grafo $G$. Posteriormente, com base na densidade de arestas desejada para o grafo, são sorteados pares de pontos, cada qual correspondendo a uma aresta de $G$. Para os custos de ativação dessas arestas, adotou-se a distância Euclidiana entre os pontos. Além dos grafos, o parâmetro quantidade mínima de folhas ($l$) foi definido igual a 80% do total de vértices do grafo (esse percentual escolhido tem respaldo nos trabalhos de

| n | d | opt | root relax. | lb | ub | nodes | F-cuts | cut sets | time (s) | |
|---|---|---|---|---|---|---|---|---|---|---|
| 50 | 5 | 19 | 20,071 | 19 | 19,000 | 13 | 101 | 220 | 0,502 | ** |
| 50 | 20 | 43 | 45,068 | 43 | 43,000 | 90 | 1559 | 373 | 3,676 | ** |
| 50 | 50 | 47 | 48,018 | 47 | 47,000 | 0 | 0 | 0 | 0,528 | ** |
| 70 | 5 | 43 | 45,267 | 43 | 43,000 | 128 | 537 | 1878 | 18,512 | ** |
| 70 | 20 | 63 | 64,910 | 63 | 63,000 | 169 | 4251 | 1150 | 27,595 | ** |
| 70 | 50 | 67 | 67,950 | 67 | 67,000 | 183 | 2324 | 621 | 17,036 | ** |
| 120 | 5 | 95 | 98,330 | 95 | 95,000 | 1332 | 3953 | 14992 | 2238,58 | ** |
| 120 | 20 | 112 | 114,827 | 112 | 114,006 | 799 | 22486 | 4651 | 3602,53 | * |
| 120 | 50 | 116 | 118,006 | 116 | 117,886 | 923 | 50571 | 5260 | 3583,67 | * |

Tabela 3.4: Performance da formulação (2.49) nas instâncias do MLSTP

JULSTROM (2004) e GOUVEIA e TELHADA (2011)).

A Tabela 3.5 resume o desempenho dos algoritmos testados nas instâncias geradas. As três primeiras colunas da tabela descrevem as características gerais da instância, como a quantidade de vértitices no grafo, $n$, o número mínimo de folhas que solução deverá conter, $l$, e a densidade de arestas no grafo, $d$ (dado em valores percentuais). A coluna *opt* exibe o valor da ávore de custo mínimo contendo pelo menos $l$ folhas — para certificar a corretude dos custos das soluções ótimas apresentados, foram implementadas duas diferentes versões do algoritmo *branch-and-cut*, tomando como base as formulações (2.22) e adaptação da formulação (2.49) para o LCMSTP (essa adaptação é trivial e segue procedimento idêntico àquele explicado na seção 2.2 para a obteção de (2.22) a partir de (2.18)).

As demais colunas que se seguem apresentam *bounds* de relaxação linear fornecidos por cada uma das formulações estudadas nesse trabalho. Para facilitar a compreensão dos dados, foram destacadas em **negrito** as melhores ocorrências dos *bounds* (*i.e.* aquele limitante que estiver mais próximo do valor da solução ótima). Melhor do que uma comparação envolvendo puramente a performance dos algoritmos medida em tempo de execução (que é função direta do código implementado, do *hardware* no qual ele será executado e, mais importante, de flutuações erráticas do tempo devido a questões aleatórias exclusivamente ligadas à estrutura da árvore de busca associada a cada uma das formulações — ver o trabalho de FISCHETTI e MONACI (2014) a respeito de como explorar essas flutuações para obter benefícios na resolução de MIPs), acreditamos que a comparação dos limitantes seja um direcionador bastante relevante e confiável para a comparação da força de cada uma das formulações. Todos esses resultados apresentados correspondem ao melhor valor de limitante encontrado no nó zero da árvore de pesquisa.

| Instância | | | | Não dir. | | Direcionado | | | |
|---|---|---|---|---|---|---|---|---|---|
| n | l | d | opt | fuj | $\text{fuj}_{\text{for}}$ | $\text{dir}_{\text{min}}$ | $\text{dir}_{\text{avg}}$ | $\text{dir}_{\text{max}}$ | automatic |
| 30 | 24 | 30 | 923 | 855 | 918 | 897 | 921 | **923** | **923** |
| 40 | 32 | 30 | 1146 | 1051 | 1116 | 1135 | 1143 | **1146** | **1146** |
| 50 | 40 | 30 | 1039 | 981 | 1007 | 1011 | 1029 | **1038** | 1036 |
| 60 | 48 | 30 | 1762 | 1626 | 1654 | 1650 | 1685 | **1699** | 1693 |
| 70 | 56 | 30 | 2018 | 1910 | 1940 | 1934 | 1959 | **1977** | 1966 |
| 80 | 64 | 30 | 2200 | 1970 | 2104 | 2114 | 2135 | **2149** | **2149** |
| 90 | 72 | 30 | 2233 | 2026 | 2134 | 2157 | 2177 | **2189** | **2189** |
| 100 | 80 | 30 | 2662 | 2415 | 2582 | 2607 | 2623 | **2632** | 2629 |
| 30 | 24 | 70 | 620 | 522 | 582 | 566 | 581 | **588** | **588** |
| 40 | 32 | 70 | 691 | 653 | 653 | 639 | 653 | 659 | **660** |
| 50 | 40 | 70 | 722 | 645 | 674 | 684 | 694 | **699** | **699** |
| 60 | 48 | 70 | 1214 | 1097 | 1195 | 1168 | 1177 | **1182** | **1182** |
| 70 | 56 | 70 | 1359 | 1214 | 1267 | 1291 | 1314 | **1326** | 1320 |
| 80 | 64 | 70 | 1412 | 1208 | 1342 | 1335 | 1353 | **1360** | 1355 |
| 90 | 72 | 70 | 1452 | 1345 | 1376 | 1400 | 1412 | **1420** | 1416 |
| 100 | 80 | 70 | 1676 | 1511 | 1562 | 1609 | 1621 | **1629** | 1626 |

Tabela 3.5: Comparação dos *bounds* de relaxação linear - LCMSTP

Para o caso das formulações não direcionadas, a coluna **fuj** fornece a relaxação linear da formulação (2.10) acrescida das desigualdades válidas (2.8). Ao lado dessa coluna, e ainda se tratando das formulações não direcionadas, é exibida a coluna $\mathbf{fuj_{for}}$ que apresenta o *bound* de relaxação linear obtido pela nova formulação não direcionada fortalecida (2.39). Há que se ressaltar que essa formulação fortalecida foi capaz de melhorar os limitantes duais para todas as instâncias testadas. Em termos do ganho real representado por esse aumento, o fechamento percentual do *gap* de otimalidade (calculado a partir do módulo da diferença entre os dois *bounds*, dividido pela diferença entre o menor deles e a solução ótima calculada) foi de aproximadamente 50% em média, o que se caracteriza como um ganho bastante expressivo, destacando a eficiência das desigualdades propostas no fortalecimento da relaxação linear desse modelo.

As três colunas que se seguem na tabela, com os cabeçalhos $\mathbf{dir_{min}}$, $\mathbf{dir_{avg}}$ e $\mathbf{dir_{max}}$ representam respectivamente o pior *bound* obtido (relativo a raiz de pior desempenho dentre todas as raízes possíveis), o *bound* médio e o melhor deles fornecidos pela relaxação linear de (2.22), adaptada para o LCMSTP, acrescidos os cortes (2.23). A variação dos valores da relaxação linear para diferentes escolhas do vértice raíz já era esperado, conforme observado por GOUVEIA e TELHADA (2011), e é devida a adição dos cortes (2.19). Os valores de mínimo, médio e máximo foram obtidos a partir da resolução de um total de 1040 relaxações lineares (para um grafo de tamanho $n$ foram testadas todas as $n$ formulações possíveis obtidas a partir da escolha de um dos vértices como raiz da arborescência).

Importante salientar que, embora tenha sido estabelecida uma relação de dependência entre o *bound* fornecido pela formulação e a escolha do vértice raiz, justificada pela adição das restrições (2.19), na sua situação mais favorável (na coluna $\mathbf{dir_{max}}$), a formulação direcionada dominou amplamente as demais, tendo sido responsável pelo fornecimento do melhor bound na maioria dos experimentos conduzidos, se mostrando uma alternativa bastante atraente para a resolução das instâncias do LCMSTP. Mesmo na sua situação mais desfavorável (cujos valores são apresentados na coluna $\mathbf{dir_{min}}$), esse modelo parece suplantar os resultados fornecidos pela formulação não direcionada, mesmo após o seu fortalecimento ($\mathbf{fuj_{for}}$).

Finalmente, na décima e última coluna da tabela, denominada **automatic** (em referência ao fato de que a escolha do vértice raiz da arborescência é delegada ao modelo), são apresentados os resultados obtidos pela relaxação linear da formulação (2.49), adaptada ao LCMSTP. Os resultados da tabela indicam que essa formulação é capaz de produzir *bounds* pelo menos tão bons quanto a média dos valores fornecidos pela formulação direcionada. De fato, uma análise mais minusciosa mostra que, para esse conjunto de instâncias, os *bounds* advindos dessa nova formulação estão situados acima do terceiro quartil dos *bounds* do caso direcionado, significando que

o resultado fornecido pelo modelo, para essas instâncias, é maior do que 75% dos *bounds* fornecidos pelo modelo direcionado.

Para facilitar essa anáise, foram construídos diagramas de caixa (também conhecidos como *box-plot*) para cada uma das instâncias testadas. Esses diagramas são ferramentas de estatística descritiva utilizados em estudos de dispersão de amostras. As Figuras 3.1, 3.2, 3.3 e 3.4 são diagramas de caixa criados a partir dos *bounds* de relaxação linear fornecidos pelo modelo direcionado (2.22). Neles, analisando da parte inferior para a superior, os pequenos círculos vazados representam os *outliers* da amostra, *i.e.* correspondem a dados que sejam pelo menos 1,5 vezes menores ou maiores do que os quartis inferior e superior. As pequenas barras horizontais (ligadas à caixa por uma linha tracejada) representam os valores mínimo e máximo observados, excluindo-se os *outliers*. Os quartis inferior (i.e. a linha que separa os 25% dos valores mais baixos) e superior (*i.e.* a linha que separa os 25% dos valores mais altos) das medições são representados pelas bordas horizontais da caixa. Finalmente, a linha horizontal mais grossa, situada sempre no interior da caixa, localiza a mediana dos dados, *i.e.* o valor central observado, aquele que separam as metades inferior e superior dos dados.

Com objetivo de salientar a qualidade das soluções propostas pelo novo modelo, foi acrescentado aos diagramas uma linha vermelha que posiciona o *bound* de relaxação linear devido à formulação (2.49), deixando bastante evidente o grande potencial dessa nova abordagem para o problema.

As Tabelas 3.6, 3.7 e 3.8 apresentadas a seguir sintetizam a performance do algoritmo de *branch-and-cut* construído a partir da formulação (2.18), para três diferentes estratégias de seleção do vértice raiz, quais seriam, respectivamente, a escolha da pior raiz (sob o ponto de vista da qualidade do limitante de relaxação linear fornecido), raiz de maior grau (*i.e.*, o vértice de maior grau do grafo) e, a melhor raiz, também sob a ótica da qualidade do limitante de relaxação linear fornecido. As quatro primeiras colunas das tabelas apresentam as características estruturais das instâncias (tamanho de vértices do grafo, número mínimo de folhas que a arborescência deverá conter, densidade do grafo e o índice do vértice escolhido como raiz da arborescência). Na sequência, a quinta, sexta e sétima colunas apresentam o valor de relaxação linear fornecido pela raiz escolhida (no nó zero da árvore), o valor da melhor solução dual encontrada e o valor da melhor solução primal, ao término da execução do algoritmo (ou ao término do tempo máximo de execução, limitado em 3600 segundos). Finalizando, as colunas oito, nove, dez e onze apresentam a performance do algoritmo de *branch-and-cut* em função da quantidade de nós explorados na árvore de buscar (B&B), o número de desigualdades do tipo *F-cuts* e *cut-sets* encontradas, e o tempo de execução do algoritmo.

A análise dessas tabelas nos permite concluir que tanto melhor será a perfor-

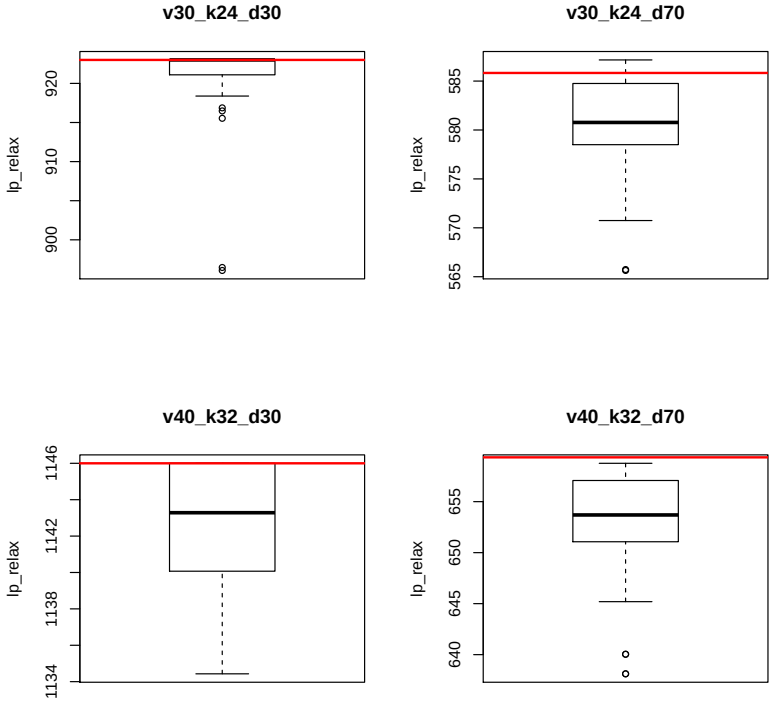


Figura 3.1: Box plot: instâncias de tamanho $n$=30 e $n$=40

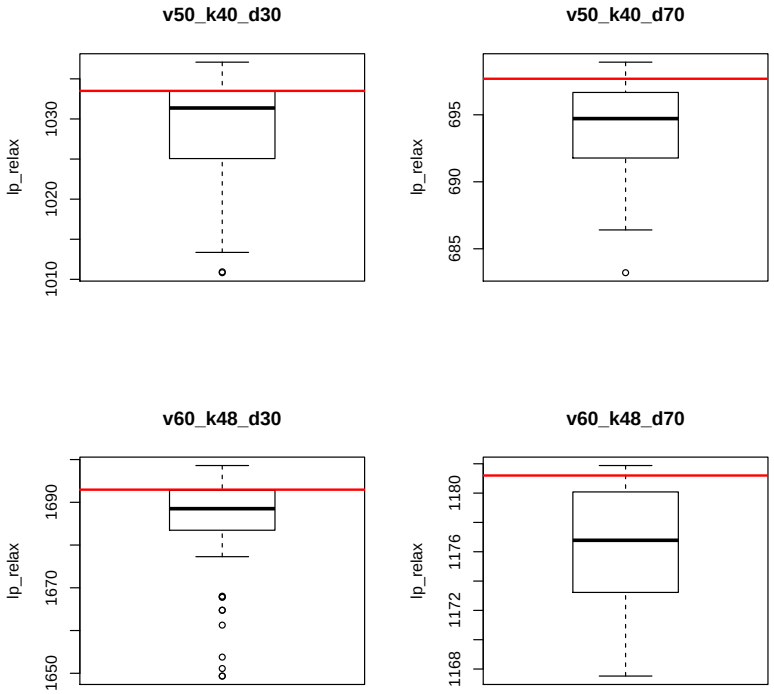


Figura 3.2: Box plot: instâncias de tamanho $n$=50 e $n$=60

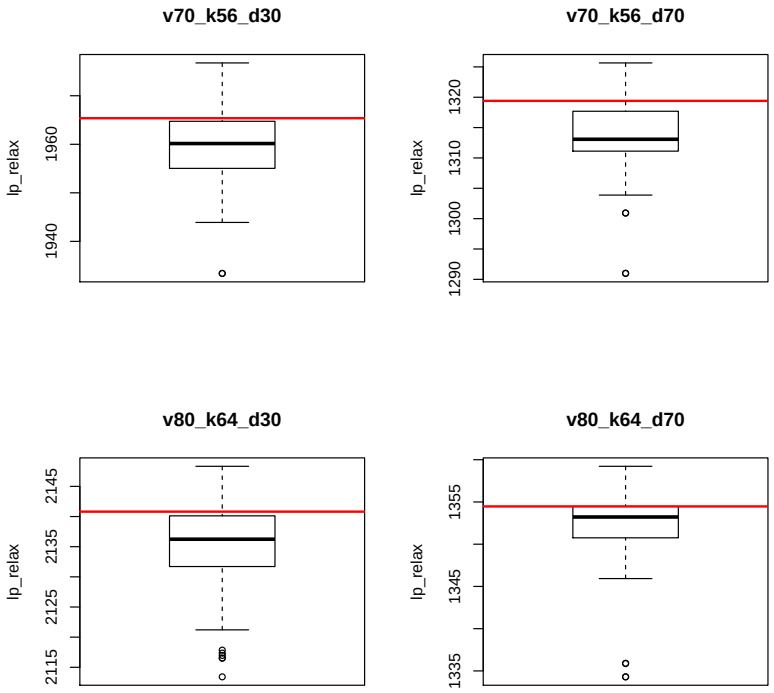


Figura 3.3: Box plot: instâncias de tamanho $n$=70 e $n$=80

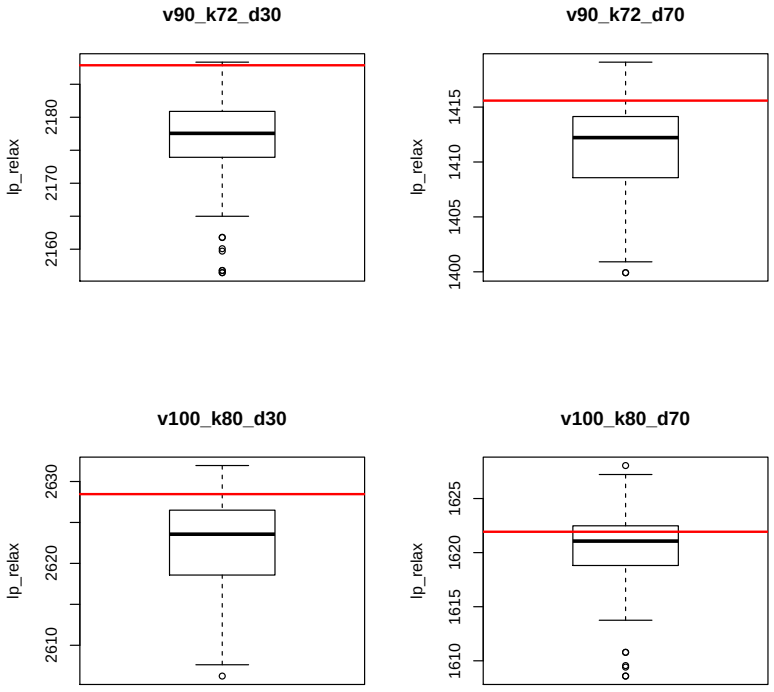


Figura 3.4: Box plot: instâncias de tamanho $n$=90 e $n$=100

| n | l | d | r | $lr_0$ | lb | ub | nodes | f-cuts | cut-set | t(s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 30 | 24 | 30 | 26 | 896,08 | 923,0 | 923 | 12 | 268 | 50 | 1,3 | * |
| 40 | 32 | 30 | 27 | 1134,43 | 1146,0 | 1146 | 6 | 196 | 67 | 2,2 | * |
| 50 | 40 | 30 | 40 | 1010,83 | 1039,0 | 1039 | 34 | 526 | 316 | 2,8 | * |
| 60 | 48 | 30 | 4 | 1649,35 | 1762,0 | 1762 | 437 | 2022 | 2046 | 88,6 | * |
| 70 | 56 | 30 | 19 | 1933,39 | 2018,0 | 2018 | 294 | 1654 | 1949 | 71,4 | * |
| 80 | 64 | 30 | 20 | 2113,42 | 2200,0 | 2200 | 217 | 2137 | 2492 | 109,4 | * |
| 90 | 72 | 30 | 20 | 2156,44 | 2233,0 | 2233 | 867 | 3118 | 2446 | 451,4 | * |
| 100 | 80 | 30 | 59 | 2606,24 | 2662,0 | 2662 | 132 | 1559 | 1191 | 37,4 | * |
| 30 | 24 | 70 | 14 | 565,64 | 620,0 | 620 | 108 | 601 | 233 | 1,5 | * |
| 40 | 32 | 70 | 17 | 638,12 | 691,0 | 691 | 418 | 953 | 546 | 10,1 | * |
| 50 | 40 | 70 | 21 | 683,22 | 722,0 | 722 | 272 | 1152 | 933 | 19,5 | * |
| 60 | 48 | 70 | 51 | 1167,52 | 1214,0 | 1214 | 295 | 1530 | 960 | 29,8 | * |
| 70 | 56 | 70 | 62 | 1290,97 | 1359,0 | 1359 | 897 | 2165 | 2698 | 362,9 | * |
| 80 | 64 | 70 | 2 | 1334,30 | 1393,7 | 1419 | 2686 | 3125 | 11962 | 3600,0 | |
| 90 | 72 | 70 | 20 | 1399,92 | 1452,0 | 1452 | 368 | 1582 | 1309 | 118,6 | * |
| 100 | 80 | 70 | 19 | 1608,59 | 1659,3 | 1694 | 1387 | 3035 | 8917 | 3600,0 | |

Tabela 3.6: Performance formulação direcionada LCMSTP - estratégia pior raiz

| n | l | d | r | $lr_0$ | lb | ub | nodes | f-cuts | cut-set | t(s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 30 | 24 | 30 | 14 | 916,48 | 923,0 | 923 | 12 | 251 | 33 | 0,4 | * |
| 40 | 32 | 30 | 8 | 1143,51 | 1146,0 | 1146 | 4 | 171 | 43 | 2,0 | * |
| 50 | 40 | 30 | 2 | 1025,06 | 1039,0 | 1039 | 8 | 369 | 94 | 1,8 | * |
| 60 | 48 | 30 | 47 | 1694,29 | 1762,0 | 1762 | 114 | 749 | 422 | 3,9 | * |
| 70 | 56 | 30 | 48 | 1945,62 | 2018,0 | 2018 | 93 | 822 | 507 | 6,3 | * |
| 80 | 64 | 30 | 40 | 2132,22 | 2200,0 | 2200 | 68 | 973 | 515 | 8,3 | * |
| 90 | 72 | 30 | 13 | 2188,35 | 2233,0 | 2233 | 90 | 699 | 140 | 3,0 | * |
| 100 | 80 | 30 | 89 | 2617,63 | 2662,0 | 2662 | 101 | 1247 | 658 | 14,9 | * |
| 30 | 24 | 70 | 14 | 565,64 | 620,0 | 620 | 108 | 601 | 233 | 2,3 | * |
| 40 | 32 | 70 | 14 | 658,61 | 691,0 | 691 | 187 | 569 | 252 | 2,0 | * |
| 50 | 40 | 70 | 5 | 694,60 | 722,0 | 722 | 156 | 633 | 392 | 4,4 | * |
| 60 | 48 | 70 | 47 | 1180,42 | 1214,0 | 1214 | 113 | 732 | 221 | 3,8 | * |
| 70 | 56 | 70 | 15 | 1323,57 | 1359,0 | 1359 | 252 | 1172 | 773 | 12,1 | * |
| 80 | 64 | 70 | 73 | 1334,30 | 1394,0 | 1427 | 2661 | 2801 | 12023 | 3600,0 | |
| 90 | 72 | 70 | 34 | 1415,61 | 1452,0 | 1452 | 122 | 864 | 391 | 15,2 | * |
| 100 | 80 | 70 | 21 | 1619,06 | 1667,0 | 1685 | 2404 | 2620 | 9705 | 3600,0 | |

Tabela 3.7: Performance formulação direcionada LCMSTP - estratégia raiz de maior grau

| n | l | d | r | $lr_0$ | lb | ub | nodes | f-cuts | cut-set | t(s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 30 | 24 | 30 | 1 | 923,00 | 923,0 | 923 | 0 | 37 | 10 | 0,0 | * |
| 40 | 32 | 30 | 3 | 1146,00 | 1146,0 | 1146 | 2 | 129 | 24 | 0,5 | * |
| 50 | 40 | 30 | 13 | 1037,08 | 1039,0 | 1039 | 6 | 289 | 108 | 0,8 | * |
| 60 | 48 | 30 | 18 | 1698,60 | 1762,0 | 1762 | 69 | 632 | 230 | 1,4 | * |
| 70 | 56 | 30 | 58 | 1976,76 | 2018,0 | 2018 | 72 | 763 | 404 | 3,5 | * |
| 80 | 64 | 30 | 29 | 2148,33 | 2200,0 | 2200 | 121 | 1087 | 690 | 8,8 | * |
| 90 | 72 | 30 | 13 | 2188,35 | 2233,0 | 2233 | 90 | 699 | 140 | 2,5 | * |
| 100 | 80 | 30 | 53 | 2631,95 | 2662,0 | 2662 | 53 | 804 | 290 | 6,2 | * |
| 30 | 24 | 70 | 4 | 587,16 | 620,0 | 620 | 28 | 179 | 46 | 0,5 | * |
| 40 | 32 | 70 | 22 | 658,76 | 691,0 | 691 | 81 | 434 | 136 | 1,0 | * |
| 50 | 40 | 70 | 4 | 698,93 | 722,0 | 722 | 38 | 373 | 87 | 0,8 | * |
| 60 | 48 | 70 | 32 | 1181,88 | 1214,0 | 1214 | 123 | 860 | 418 | 7,1 | * |
| 70 | 56 | 70 | 24 | 1325,65 | 1359,0 | 1359 | 127 | 813 | 477 | 7,0 | * |
| 80 | 64 | 70 | 77 | 1359,22 | 1412,0 | 1412 | 659 | 1456 | 1747 | 64,4 | * |
| 90 | 72 | 70 | 65 | 1419,09 | 1452,0 | 1452 | 92 | 886 | 505 | 12,7 | * |
| 100 | 80 | 70 | 80 | 1628,05 | 1676,0 | 1676 | 1048 | 1672 | 2851 | 145,6 | * |

Tabela 3.8: Performance formulação direcionada LCMSTP - estratégia melhor raiz

| n | l | d | $lr_0$ | lb | ub | nodes | f-cuts | cut-set | t(s) | |
|---|---|---|---|---|---|---|---|---|---|---|
| 30 | 24 | 30 | 923,00 | 923 | 923 | 0 | 152 | 19 | 0,1 | * |
| 40 | 32 | 30 | 1146,00 | 1146 | 1146 | 2 | 147 | 56 | 2,2 | * |
| 50 | 40 | 30 | 1033,50 | 1039 | 1039 | 19 | 449 | 198 | 3,7 | * |
| 60 | 48 | 30 | 1692,96 | 1762 | 1762 | 266 | 1673 | 1148 | 15,1 | * |
| 70 | 56 | 30 | 1965,37 | 2018 | 2018 | 190 | 1291 | 803 | 14,0 | * |
| 80 | 64 | 30 | 2140,81 | 2200 | 2200 | 107 | 1304 | 902 | 19,6 | * |
| 90 | 72 | 30 | 2187,88 | 2233 | 2233 | 525 | 2266 | 1717 | 70,1 | * |
| 100 | 80 | 30 | 2628,46 | 2662 | 2662 | 63 | 929 | 457 | 8,8 | * |
| 30 | 24 | 70 | 585,83 | 620 | 620 | 65 | 528 | 260 | 2,3 | * |
| 40 | 32 | 70 | 659,34 | 691 | 691 | 269 | 878 | 391 | 5,2 | * |
| 50 | 40 | 70 | 697,69 | 722 | 722 | 466 | 1191 | 966 | 26,0 | * |
| 60 | 48 | 70 | 1181,20 | 1214 | 1214 | 482 | 1968 | 1893 | 76,3 | * |
| 70 | 56 | 70 | 1319,40 | 1359 | 1359 | 1238 | 2478 | 3595 | 292,3 | * |
| 80 | 64 | 70 | 1354,47 | 1412 | 1412 | 2878 | 2441 | 9988 | 3032,3 | * |
| 90 | 72 | 70 | 1415,59 | 1452 | 1452 | 1246 | 2475 | 3387 | 447,4 | * |
| 100 | 80 | 70 | 1621,93 | 1655,54 | 1694 | 2479 | 3501 | 13066 | 3600,0 | |

Tabela 3.9: Performance formulação automática LCMSTP

mance do algoritmo *branch-and-cut* quanto mais justo for o limitante dual encontrado no nó inicial da árvore de busca. Perceba que a estratégia de utilização da melhor raiz (que fornece os *bounds* mais apertados) resultou na melhor performance obtida, seja sob o ponto do vista do total de nós explorados, ou do tempo de execução do algoritmo, que foi capaz de resolver à otimalidade todas as instâncias listadas, dentro do limite de tempo estipulado.

Seguindo a mesma formatação das tabelas anteriores, a Tabela 3.9 apresenta a performance da implementação do algoritmo de *branch-and-cut* para a formulação (2.49), no conjunto de instâncias utilizado. Da análise da tabela podemos concluir que a formulação proposta é de fato uma abordagem promissora para o tratamento do problema, tendo sido capaz de resolver à otimalidade a maioria das instâncias testadas (à exceção da instância com 100 vértices e densidade 70%).

Para finalizar os experimentos, foi utilizado um segundo conjunto de instâncias, maiores do que as primeiras, com objetivo de identificar o potencial das formulações em termos do tamanho máximo dos grafos para os quais o problema poderia ser resolvido. As tabelas 3.10 e 3.11 apresentam os resultados dos experimentos computacionais sobre esse novo conjunto de instâncias, utilizando os algoritmos implementados a partir das formulações (2.18) e (2.49) respectivamente. Os nomes

| n | l | r | $lr_0$ | lb | ub | nodes | f-cuts | cut-set | sel. t(s) | t(s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 125 | 100 | 11 | 2700,28 | 2763 | 2763 | 1488 | 8056 | 5194 | 36,6 | 179,2 | * |
| 125 | 113 | 12 | 3686,37 | 3926 | 3926 | 1148 | 10323 | 13906 | 33,4 | 2352,3 | * |
| 150 | 120 | 95 | 2894,52 | 2970,45 | 3005 | 4249 | 13689 | 18427 | 73,9 | 7200,0 | |
| 150 | 135 | 45 | 4020,29 | 4208,78 | 4431 | 2013 | 17258 | 18354 | 58,5 | 7200,0 | |
| 175 | 140 | 161 | 2934,14 | 2985,61 | 3034 | 3703 | 18024 | 17714 | 97,9 | 7200,1 | |
| 200 | 160 | 136 | 3026,46 | 3102,79 | 3120 | 9957 | 20010 | 18193 | 173,4 | 7200,0 | |

Tabela 3.10: Performance formulação direcionada LCMSTP - instâncias grandes

| n | l | $lr_0$ | lb | ub | nodes | f-cuts | cut-set | lr t(s) | t(s) | |
|---|---|---|---|---|---|---|---|---|---|---|
| 125 | 100 | 2685,5 | 2743,7 | 2785 | 2260 | 4004 | 14721 | 22,7 | 7200,1 | |
| 125 | 113 | 3689,1 | 3926,0 | 3926 | 1795 | 5439 | 14781 | 7,6 | 7200,0 | * |
| 150 | 120 | 2882,4 | 2942,0 | 3029 | 2547 | 4708 | 15559 | 19,1 | 7200,7 | |
| 150 | 135 | 4021,2 | 4186,4 | 4440 | 2672 | 14038 | 11083 | 46,9 | 7200,1 | |
| 175 | 140 | 2924,7 | 2961,9 | 3062 | 1552 | 5037 | 12882 | 118,9 | 7200,1 | |
| 200 | 160 | 3013,0 | 3041,5 | 3165 | 891 | 4988 | 15704 | 88,9 | 7200,3 | |

Tabela 3.11: Performance formulação automática LCMSTP - instâncias grandes

das colunas das tabelas seguiram o mesmo padrão adotado nas demais tabelas e não serão novamente explicados. Faz-se necessário apenas comentar o significado da penúltima coluna de cada tabela, que estaria relacionada com o tempo total gasto na resolução da relaxação linear no nó zero. Para a formulação (2.18), utilizou-se uma pequena lista de vértices (contendo 10% dos vértices do grafo), escolhidos aleatoriamente, para os quais foram resolvidas as relaxações lineares dos modelos quando considerados esses vértices como raiz da arborescência — o vértice capaz de fornecer o melhor *bound* de relaxação linear foi escolhido para a busca na árvore.

Pelos resultados apresentados, nenhuma das implementações *branch-and-cut* se mostrou capaz de resolver as instâncias para grafos com mais do que 150 vértices, sugerindo que o limite dessas formulações estivessem em torno de 125 vértices.

# Capítulo 4

# Algoritmos *relax-and-cut*

Uma idéia computacional extremamente importante é a observação de que muitos problemas difíceis de otimização podem ser interpretados como problemas fáceis que foram complicados pela adição de um ou mais conjuntos de restrições complicadoras FISCHER (1981). A Relaxação Lagrangeana é uma técnica que permite lidar com essa particularidade: associando um vetor de multiplicadores a essas restrições, somos capazes de inserir essa informação na função objetivo num processo denominado dualização. O novo problema obtido – chamado Problema Lagrangeano – poderá então ser resolvido de maneira iterativa: encontra-se uma solução para o Problema Lagrangeano, atualiza-se o vetor de multiplicadores de Lagrange, e esse processo é repetido até que um critério de parada específico seja atingido. Assim procedendo, um limite inferior (ou superior, caso o problema original seja de maximização) é então obtido para o problema original.

Acredita-se que o trabalho de LORIE e SAVAGE (1973) tenha sido a primeira publicação em que aparece a noção de dualidade em programação inteira, através da utilização do método de Relaxação Lagrangeana. Outro trabalho percussor dessa teoria foi o de EVERETT (1963), mas foram os trabalhos de HELD e KARP (1970) e HELD e KARP (1971) que provocaram uma intensa investigação nessa área do conhecimento científico. O termo Relaxação Lagrangeana só apareceu em 1974, tendo sido cunhado por GEOFFRION (1974).

## 4.1 Relaxação Lagrangeana

Considere o seguinte problema de otimização combinatória, $\mathcal{NP}$-difícil, formulado como um problema de programação linear inteira:

$$\begin{aligned} \text{Minimizar} \quad & z = \mathbf{cx} \\ \text{sujeito a:} \quad & A\mathbf{x} \geq \mathbf{b} \\ & E\mathbf{x} \leq \mathbf{f} \\ & \mathbf{x} \in \mathbb{B}^n \end{aligned} \tag{4.1}$$

onde $\mathbf{c}$ é um vetor de custos $n \times 1$, $\mathbf{b}$ é um vetor $m \times 1$, $\mathbf{f}$ um outro vetor $k \times 1$ e as demais matrizes, $A$ e $E$, tendo dimensões apropriadas.

Assumiremos sem perda de generalidade que o conjunto de restrições $E\mathbf{x} \leq f$ seja um conjunto de restrições complicadoras, sem as quais o problema de minimizar $z$ (ou qualquer outra função linear de $\mathbf{x}$) sujeito ao conjunto de restrições $A\mathbf{x} \geq b$ seria um problema fácil de ser resolvido (*i.e.* um problema para o qual se conheça um algoritmo polinomial). Seguindo essa ideia, definiremos o Problema Lagrangeano

$$\begin{aligned} \text{Minimizar} \quad & z_D(\lambda) = \mathbf{cx} + \lambda(E\mathbf{x} - f) \\ \text{sujeito a:} \quad & \mathbf{x} \in X \end{aligned} \tag{4.2}$$

onde $\lambda \in \mathbb{R}^k_+$ será chamado vetor de multiplicadores de Lagrange e a região de viabilidade do Problema Lagrangeano será dada por um conjunto finito $X = \{\mathbf{x} : A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \in \mathbb{B}^n\} \neq \emptyset$.

Por conveniência, assumiremos que o problema original (4.1) seja viável. Então, $z_D(\lambda)$ será finito para qualquer vetor de multiplicadores $\lambda$.

Pode-se facilmente verificar que $z_D(\lambda) \leq z$ para qualquer vetor de multiplicadores $\lambda \in \mathbb{R}^k_+$ – a solução do Problema Lagrangeano é um limitante inferior para a solução do problema original. Para provar esse fato, tome uma solução ótima $\mathbf{x}^*$ para (4.1). Dessa forma, é fácil verificar que

$$z_D(\lambda) \leq \mathbf{cx}^* + \lambda(E\mathbf{x}^* - \mathbf{f}) \leq \mathbf{cx}^* = z.$$

A primeira desigualdade segue diretamente da definição de otimalidade para o Problema Lagrangeano. Como $\lambda \geq 0$ e, para qualquer solução viável $\bar{\mathbf{x}}$ de (4.1), $(E\bar{\mathbf{c}} - \mathbf{f}) \leq 0$ se aplica, segue-se então a segunda desigualdade. Finalmente, a igualdade resulta diretamente da definição de $z$ em (4.1).

Para que a abordagem Lagrangeana tenha sucesso, é preciso que se encontre o limitante mais apertado, ou seja, aquele que esteja o mais próximo possível da solução do problema original. Isso significa que é preciso encontrar o vetor ótimo de multiplicadores de Lagrange – aquele que resolva o *Problema Dual Lagrangeano*,

definido como:

$$z_D = max_\lambda z_D(\lambda).$$

Esse problema tem uma série de propriedades estruturais interessantes que tornam possível a sua resolução. Como assumimos que o conjunto viável $X$ é finito, uma abordagem possível para a resolução do problema poderia considerar a enumeração de todos os $T = |X|$ elementos. Partindo desse pressuposto, uma representação alternativa para $X$ é dada por $X = \{\mathbf{x}^t, t = 1, \ldots, T\}$. Isso permite reformular o Problema Dual Lagrangeano da seguinte maneira:

$$\begin{aligned} \text{Maximizar} \quad & z_D = w \\ \text{sujeito a:} \quad & w \leq \mathbf{c}\mathbf{x}^t + \lambda(E\mathbf{x} - \mathbf{f}) \\ & w \in \mathbb{R}. \end{aligned} \tag{4.3}$$

Essa nova representação para o Problema Dual Lagrangeano deixa claro que $z_D(\lambda)$ é o envelope inferior de uma familia (finita) de funções lineares — $z_D(\lambda)$ é uma função côncava e linear por partes. Apesar de apresentar as características de concavidade e continuidade desejáveis para a construção de bons algoritmos de busca em linha, essa função é não-diferenciável em qualquer um dos pontos de interesse e, portanto, será preciso aplicar alguma técnica específica para a solução desse problema de otimização. Foi proposto por DANTZIG e WOLFE (1960) uma técnica de decomposição, também conhecida como geração de colunas, que resolve o problema de maneira exata à otimalidade. Entretanto, a complexidade do algoritmo faz com que o método seja, por vezes, preterido em relação a outros. Entre eles, um dos mais utilizados e que merecem grande destaque é o Método do Subgradiente, que teria sido originalmente proposto na união soviética por Shor em 1970 (BOYD e VANDENBERGHE (2003)), e independentemente desenvolvido por HELD *et al.* (1974). Sua simplicidade, facilidade de implementação e os bons resultados proporcionados garantiram a esse método enorme popularidade.

Métodos mais sofisticados, como o método dos feixes (LAMARÉCHAL (1989)), cuja origem remete à resolução de problemas de otimização não-linear, e o método do volume (BARAHONA e ANBIL (2000)), uma extensão do algoritmo do Subgradiente capaz de produzir ambas soluções, primal e dual, podendo ser interpretado como uma maneira rápida de se obter uma aproximação da decomposição de Dantzig-Wolfe, também merecem destaque.

### 4.1.1 Algoritmo do Subgradiente

Diz-se que um vetor $\mathbf{g} \in \mathbb{R}^n$ é um subgradiente de $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ em um ponto $\mathbf{x} \in \mathbf{dom}f$ se, para todo $\mathbf{y} \in \mathbf{dom}f$, verifica-se

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}(\mathbf{y} - \mathbf{x}).$$

Caso a função $f$ seja diferenciável, então o subgradiente no ponto $\mathbf{x}$ coincide com o gradiente da função nesse mesmo ponto. Entretanto, o subgradiente pode existir mesmo que a função $f$ seja não-diferenciável em $\mathbf{x}$ (veja BOYD e VANDENBERGHE (2003)).

O Algoritmo do Subgradiente é uma adaptação direta do método do gradiente no qual o gradiente da função $f$ no ponto $\mathbf{x}$ é substituído por seu subgradiente. Infelizmente, a direção apontada pelo subgradiente não é sempre uma direção de acréscimo como acontecia antes — essa direção pode nem mesmo ser uma direção viável. Por causa disso, é comum que se mantenha uma cópia da melhor solução encontrada ao transcorrer do método, i.e. aquela que minimiza o valor da função objetivo (para os métodos de descida, não há necessidade de utilização desse artifício, uma vez que a melhor solução encontrada será sempre a solução corrente).

Numa iteração $p$ do Método do Subgradiente, seja $\lambda^p$ um vetor viável de multiplicadores de Lagrange. Sejam ainda $\bar{\mathbf{x}}^p$ a solução ótima de $z_D(\lambda^p)$ e $z_{ub}$ um limite superior para a função objetivo quando avaliada em $\bar{\mathbf{x}}^p$ — esse limitante pode ser dado, por exemplo, por qualquer heurística primal para o problema. Pode-se calcular o subgradiente associado às restrições dualizadas da seguinte maneira:

$$\mathbf{g}^p = \mathbf{f} - \mathbf{e}\bar{\mathbf{x}}^p \tag{4.4}$$

Observe que, em uma dada iteração do Método do Subgradiente, a componente $g_i^p$ associada à $i$-ésima restrição indica se aquela desigualdade foi violada ($g_i^p < 0$) por $\bar{\mathbf{x}}^p$, ou não ($g_i^p \geq 0$).

Da mesma maneira como é feito para o método do gradiente, dado inicialmente um vetor válido de multiplicadores (geralmente o vetor nulo é utilizado), uma sequência $\{\lambda^p\}$ de multiplicadores poderá ser gerada a cada iteração, a partir da seguinte regra:

$$\lambda^{p+1} = max\{0,\ \lambda^p - \theta^p \mathbf{g}^p\} \tag{4.5}$$

onde $\mathbf{g}^p$ é o subgradiente calculado na iteração $p$ e $\theta^p$ é um escalar positivo responsável por determinar o tamanho do passo na direção apontada pelo subgradiente.

Um dos resultados teóricos mais importantes a respeito do Método do Subgradiente é que o método converge (i.e. $z_D(\lambda^p) \rightarrow z_D$) quando garantidas algumas

condições acerca do comportamento da sequência $\{\theta^p\}$, do tamanho do passo. Segundo FISCHER (1981), garantidas as condições:

i $\theta^p \to 0$

ii $\sum_{j=0}^{p} \theta^p \to \infty$

o método convergirá em um número finito de iterações.

Em BEALSEY (1993), o autor sugere uma regra para atualização do tamanho do passo. Na $p$-ésima iteração do algoritmo, tome:

$$\theta^p = \frac{\alpha^p \cdot (z_{ub}^p - z_D(\lambda^p))}{\|\mathbf{g}^p\|^2} \tag{4.6}$$

onde $\alpha^p$ é um escalar que satisfaz $0 < \alpha^p \leq 2$. Geralmente a sequência $\{\alpha^p\}$ é obtida tomando $\alpha^0 = 2$ e, sempre que não houver melhoria no valor $z_D(\lambda^p)$ por um número determinado de iterações, esse valor é dividido por 2.

Outras provas de convergência do método, para diferentes sequências de tamanho do passo, poderão ser encontradas em BERTSEKAS (1999) e BOYD e VANDENBERGHE (2003).

## 4.2 Algoritmos *Relax-and-Cut*

O algoritmo NDRC (*non-delayed relax-and-cut*) foi proposto em LUCENA (1992) e é baseado na utilização do Método do Subgradiente para a resolução de problemas combinatórios difíceis. Esse algoritmo propõe um esquema dinâmico de dualização que possibilita a aplicação da abordagem de Relaxação Lagrangeana a problemas em que o número de restrições a dualizar é exponencialmente grande.

Suponha a utilização tradicional da abordagem Lagrangeana em um problema desse tipo. O número exponencial de restrições dualizadas implicaria um esforço computacional proibitivo para cada iteração do método subgradiente. Mesmo que as iterações se dessem em um tempo "aceitável", é de se esperar que ocorram problemas de convergência, uma vez que valores muito pequenos de $\theta$ resultariam em cada iteração (lembre-se que para o cálculo de $\theta$ é usada informação da norma do subgradiente, que nesse caso, possuiria um número exponencial de componentes).

A alternativa do *Relax-and-Cut* se baseia na classificação das restrições a serem dualizadas em cada iteração. Na iteração $k$ do subgradiente, seja $\bar{x}^k$ a solução ótima do Subproblema Lagrangeano para um vetor de multiplicadores $\lambda^k$. Então, LUCENA (1992) propõe uma divisão das restrições dualizadas em 3 diferentes grupos: (i) o grupo 1 conterá todas as restrições que foram violadas por $\bar{x}^k$; (ii) o grupo 2 conterá aquelas restrições que possuem multiplicadores não-nulos associados a elas,

e; (iii) o grupo 3 conterá todas as demais restrições, abrangendo a vasta maioria das desigualdades envolvidas. Vale destacar que uma desigualdade poderá pertencer simultaneamente aos grupos 1 e 2.

Ainda segundo LUCENA (1992), essa classificação das restrições faz sentido uma vez que, primeiro, todos os multiplicadores associados às desigualdades pertencentes ao grupo 3 manterão o seu valor nulo ao final da iteração $k$ do método e não contribuirão diretamente para o cálculo dos custos lagrangeanos – essas desigualdades são chamadas de inativas. Por último, e não menos importante, a quantidade de componentes do vetor subgradiente associadas às restrições do grupo 3, e cujos valores sejam estritamente positivos, tende a ser muito grande. Assim, caso essas componentes sejam consideradas em (4.6), espera-se que o valor de $\theta^p$ calculado seja extremamente pequeno, o que poderia levar a problemas de convergência do método.

De acordo com essa nova abordagem, as únicas desigualdades utilizadas para o cálculo dos custos lagrangeanos são aquelas pertencentes aos grupos 1 e 2. Perceba que numa dada iteração $k$, existirá um conjunto de restrições, idexadas por $j$, às quais não se tinha um multiplicador $\lambda_j > 0$ associado, mas que se tornaram violadas em $\bar{x}^k$. Assim, um ponto central do algoritmo NDRC é, portanto, a necessidade de criação de dispositivos capazes de identificar desigualdades válidas que sejam violadas por $\bar{x}^k$. Isso equivale ao problema de separação (comum nos algoritmos do tipo *branch-and-cut*), que deverá ser resolvido a cada iteração do método subgradiente. De uma certa forma, um Algoritmo *Relax-and-Cut* poderá ser considerado um análogo Lagrangeano de um algoritmo de planos de corte.

## 4.3 Um algoritmo *relax-and-cut* para o LCMSTP

Importante notar que, para grafos esparsos, é possível que não existam soluções viáveis para o LCMSTP, dada uma quantidade mínima $l$ de folhas. Sob esse aspecto, não é trivial que se obtenham heurísticas primais capazes de encontrar boas soluções para o problema. Em cenários como esse, heurísticas Lagrangeanas, que se aproveitam da informação dual na tentativa de construir uma solução primal viável, se tornam bastante atraentes. Nesse contexto surgiu o interesse em identificar o potencial de utilização desses mecanismos, e para isso, desenvolveu-se um algoritmo *relax-and-cut* simples para o LCMSTP, na expectativa de que boas soluções primais pudessem ser construídas.

O algoritmo foi desenvolvido com base na formulação não-direcionada para o LCMSTP, adaptada diretamente de FUJIE (2004). Definir-se-á aqui uma nova região poliedral, $\mathcal{R}_6$, com objetivo exclusivamente didático de facilitar a compreensão e centralizar as referências às desigualdades utilizadas na formulação:

$$\sum_{e \in E} x_e = |V| - 1 \tag{4.7}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \quad S \subset V,\ |S| \geq 2 \tag{4.8}$$

$$\sum_{e \in \delta_i} x_e + (n_i - 1) z_i \leq n_i, \qquad i \in V \tag{4.9}$$

$$\sum_{e \in F} x_e + (|F| - 1) z_i \leq |F|, \qquad i \in V,\ F \subseteq \delta_i,\ |F| \geq 2 \tag{4.10}$$

$$x_e + z_i + z_j \leq 2, \qquad e \in E \tag{4.11}$$

$$\sum_{i \in V} z_i \geq l \tag{4.12}$$

$$0 \leq x_e \leq 1 \qquad e \in E \tag{4.13}$$

$$0 \leq z_i \leq 1 \qquad i \in V \tag{4.14}$$

As restrições (4.7) - (4.9) são aquelas propostas por FUJIE (2003) e têm como função descrever a árvore no grafo $G$ e identificar os vértices folhas. A restrição (4.10), que também será chamada de corte, foi proposta em FUJIE (2004), define as facetas *F-cuts* e, por isso, é importante para o fortalecimento do *bound*. A restrição (4.11), por ter se mostrado bastante eficiente no fortalecimento dos *bounds* de relaxação linear desse modelo (conforme evidenciado nos experimentos conduzidos no presente trabalho), serão introduzidas à formulação. Finalmente, a restrição (4.12) é utilizada para assegurar que a árvore obtida contenha o número mínimo exigido de folhas.

Nesse caso, a formulação de programação inteira utilizada como base para o algoritmo de *relax-and-cut* será dada por:

$$\min \left\{ \sum_{e \in E} c_e x_e : (\mathbf{x}, \mathbf{z}) \in \mathcal{R}_6 \cap \left( \mathbb{B}^{|E|} \times \mathbb{B}^{|V|} \right) \right\} \tag{4.15}$$

Para se obter um problema relaxação Lagrangeana associado a (4.15), são definidos vetores de multiplicadores de Lagrange $\lambda \geq 0$ e $\mu \geq 0$ respectivamente às restrições (4.9) e (4.11). Estas serão mantidas dualizadas durante todo o processo, nos moldes preconizados pela técnica. Tais restrições serão por vezes chamadas de complicantes, pois são elas as responsáveis por relacionar as variáveis $x_e$ e $z_i$. As restrições (4.10 ) estarão associadas ao vetor de multiplicadores ($\gamma \geq 0$), mas diferentemente das anteriores, serão dualizadas *on-the-fly*, *i.e.*, apenas quando a violação de alguma delas for constatada. Dessa forma, denotar-se-á a região $\mathcal{R}_6^L$ a região poliedral resultante da relaxação Lagrangeana de $\mathcal{R}_6$:

$$\sum_{e \in E} x_e \;=\; |V| - 1 \tag{4.16}$$

$$\sum_{e \in E(S)} x_e \;\leq\; |S| - 1, \quad S \subset V,\; |S| \geq 2 \tag{4.17}$$

$$\sum_{i \in V} z_i \;\geq\; l \tag{4.18}$$

$$0 \leq x_e \leq 1 \qquad e \in E \tag{4.19}$$

$$0 \leq z_i \leq 1 \qquad i \in V \tag{4.20}$$

O problema de relaxação Lagrangeana associado a (4.15) é então dado por:

$$\min \left\{ \bar{\mathbf{c}}\mathbf{x} + \bar{\mathbf{d}}\mathbf{z} - \mathcal{K} : (\mathbf{x}, \mathbf{z}) \in \mathcal{R}_6^L \cap \left( \mathbb{B}^{|E|} \times \mathbb{B}^{|V|} \right) \right\} \tag{4.21}$$

onde os custos Lagrangianos da função objetivo $\bar{\mathbf{c}}$ e $\bar{\mathbf{d}}$ e a constante $\mathcal{K}$ são calculados respectivamente pelas expressões (4.22), (4.23) e (4.24), que serão apresentadas posteriormente.

Avaliando a estrutura do subproblema Lagrangeano (SL) resultante da relaxação, é importante notar dois aspectos principais:

i Não há no subproblema nenhuma restrição que acople (relacione) as decisões representadas pelos vetores $\mathbf{x}$ e $\mathbf{z}$.

ii A função objetivo apresenta-se naturalmente decomposta em duas parcelas, uma associada às variáveis de decisão $\mathbf{x}$ e a outra associadas às variávies $\mathbf{z}$. Por ser uma função linear nessas variáveis, a sua minimização poderá ser efetuada através da minimização de cada uma dessas parcelas separadamente (i.e. para esse caso específico, existe a garantia que a soma dos mínimos locais resulte no mínimo global da função).

Aliadas, as duas características descritas acima permitem que a solução do SL seja obtida por decomposição através da resolução de outros dois subproblemas, ainda menores, no qual um será descrito exclusivamente nas variáveis $\mathbf{x}$, e o outro, exclusivamente em $\mathbf{z}$.

### Resolução do subproblema em z

Para solucionar o subproblema em $\mathbf{z}$, primeiramente, ordene as variáveis $z_i$ em ordem decrescente dos valores de $\bar{d}_i$, obtendo-se como resultado a sequência ordenada

$$\{z_{i_1}, z_{i_2}, \ldots, z_{i_{k-1}}, z_{i_k}, z_{i_{k+1}}, \ldots, z_{i_n}\}$$

onde $k$ é o índice da menor variável (*i.e*, aquela situada mais à esquerda na sequência) com um coeficiente $\bar{d}_{i_k} \geq 0$.

A partir de agora, subproblema em $\mathbf{z}$ poderá ser resolvido eficientemente através de simples inspeção, fazendo com que as variáveis $z_i$ que aparecerem à esquerda de $k$ tenham valor igual a 1, e aquelas à direita de $k$, inclusive, sejam igualadas à zero.

O único cuidado que se deve tomar é que, para garantir a viabilidade da solução proposta (*i.e.*, o respeito à restrição (4.18)), é importante observar se a condição $k \geq l$ é satisfeita. Em caso afirmativo, a solução construída pelo mecanismo descrito anteriormente será uma solução viável para o problema. Em caso negativo, será preciso fazer com que as $(l - k)$ variáveis que se encontrarem imediatamente à direita de $k$, na sequência, sejam igualadas a 1 (*i.e.*, serão tomadas somente as $l - k$ menores variáveis com custos positivos).

### Resolução do subproblema em x

Para as variáveis $\mathbf{x}$, a solução do problema é ainda mais interessante. Perceba que a interseção do subespaço definido pelo conjunto das restrições (4.16), (4.17) e (4.19), com o subespaço $\mathbb{B}^{|E|}$, descreve o politopo da árvore geradora no grafo $G = (V, E)$. Dessa forma, a cada iteração do *relax-and-cut* deverá ser resolvido um problema da árvore geradora mínima com custos Lagrangeanos dados por $\bar{c}_e$. Esse problema pode ser eficientemente resolvido em $\mathcal{O}(|E| \log |V|)$ através da aplicação do algoritmo de Prim, PRIM (1957), utilizando estruturas de *heap* e lista de adjacências para a representação do grafo.

### Identificação e dualização das desigualdades violadas

Assim como ocorria no algoritmo de planos de corte, a identificação de desigualdades violadas desempenha um papel central nos algoritmos *relax-and-cut*. A quantidade exponencial de desigualdades componentes da família (4.10) exige um tratamento diferenciado, adequado à técnica proposta.

De posse do vetor $(\hat{\mathbf{x}}, \hat{\mathbf{z}})$, solução do SL, a identificação das desigualdades violadas será efetuada diretamente pela aplicação do algoritmo de separação descrito na Subseção 3.2.3. Seja $\mathcal{F}$ o conjunto contendo as desigualdades de (4.10) violadas pela solução do SL ao longo do algoritmo *relax-and-cut* — uma solução de SL violará uma desigualdade sempre que o subgradiente a ela associado for positivo. Inicialmente, $\mathcal{F} = \emptyset$ e, sempre que uma nova desigualdade for encontrada pelo mecanismo de separação, deverá ser adicionada ao conjunto $\mathcal{F}$, se lá ainda não estiver.

Sempre que uma restrição pertencente a $\mathcal{F}$ se tornar inativa (*i.e.*, quando o subgradiente a ela associado for negativo e o seu multiplicador cair a zero) é importante que a restrição seja então removida do conjunto $\mathcal{F}$, podendo ser armazenada,

por exemplo, em um *pool* de desigualdades. A remoção das desigualdades inativas desempenha um papel importante para o método *relax-and-cut* em dois aspectos. Primeiramente porque favorece a performance do método, uma vez que o tamanho do conjunto de desiguldades violadas não crescerá indefinidamente. Por outro lado, e mais importante porque a parcela relativa ao subgradiente dessas restrições não será considerada quando do cálculo da norma utilizada na determinação do tamanho do passo do algoritmo do subgradiente (ver equação (4.6)).

No que se segue do algoritmo *relax-and-cut*, as restrições ativas contidas em $\mathcal{F}$ funcionarão exatamente como as demais restrições dualizadas.

### Cálculo dos custos Lagrangeanos

Através da manipulação algébrica das restrições dualizadas é possível descrever analíticamente os coeficientes Lagrangeanos. Considere que, a cada iteração, as restrições em $\mathcal{F}$ sejam indexadas por $k$. Os custos Lagrangeanos podem então ser expressos como:

$$\bar{c}_{(i,j)} = c_{(i,j)} + \sum_{j \in \delta_i} \lambda_j + \sum_{j \in \delta_i} \sum_{k} \gamma_j^k + \mu_{(i,j)} \tag{4.22}$$

$$\bar{d}_i = \lambda_i + \sum_{j \in \delta_i} \mu_{ij} + \sum_{k} (|F_k^i| - 1). \tag{4.23}$$

Deverá ser subtraída da função objetivo a seguinte constante:

$$\mathcal{K} = \sum_{i \in V} \lambda_i n_i + 2 \cdot \sum_{e \in E} \mu_e + \sum_{k} |F^k|. \tag{4.24}$$

### Heurística Lagrangeana

Um dos principais benefícios da utilização de uma abordagem de relaxação Lagrangeana para a resolução de problemas combinatórios difíceis reside na possibilidade de utilizar a informação dual disponível para a construção de soluções viáveis para o problema LUCENA (1992).

Nesse trabalho, adaptamos a heurística proposta por JULSTROM (2004). O algoritmo se inicia pela construção de uma árvore geradora mínima (MST) em termos dos custos Lagrangeanos $\bar{\mathbf{c}}$. A partir daí, verifica-se o número de folhas contidas na árvore e caso esse número seja inferior a $l$, aplica-se repetidamente o procedimento MAIS-FOLHAS. Esse procedimento é aplicado aos vértices internos da árvore, e tem como objetivo transformá-los em folhas através do remanejamento das arestas. Seja $j$ o vértice interno escolhido. Considere ainda que $pred(j)$ seja o predecessor do vértice $j$ na árvore e $suc(j)$ a lista de vértices sucessores a ele. Assim, o método

MAIS-FOLHAS retira o vértice $j$ da árvore fazendo a ligação direta entre $pred(j)$ a um dos $suc(j)$ (preferivelmente o mais barato deles). Nessa etapa, tem-se uma árvore de $G$ contendo $n-1$ vértices. Então, para que se recomponha a árvore geradora, o vértice selecionado é religado a um dos vértices internos restantes (novamente, privilegiando a inclusão da aresta mais barata). Ao final desse passo, o número de folhas da árvore foi acrescido em uma unidade. A utilização desse método é capaz de aumentar a quantidade de folhas da árvore de modo transformar seu número maior ou igual a $l$.

A heurística descrita acima é executada a cada iteração do Método Subgradiente.

### Esboço do algoritmo *relax-and-cut*

```
RELAX-AND-CUT()
    // Inicialização dos parâmetros do relax-and-cut
 1  α = 2
 2  λ = 0
 3  max_iter = 2000
 4  best-lb = z-lb = 0
 5  z-ub = ∞
 6  counter = 0
 7  update-step-size = 200
 8  c̄ = c
 9  d̄ = 0
    // Laço principal do algoritmo
10  for i = 0 → max_iter
11      z-lb = SOLVE-SL(x̂, ẑ)                    // Soluciona o Subproblema Lagrangeano
12      z-ub = LAGRANGEAN-HEURISTIC(c̄, d̄)        // Busca solução primal viável
13      F = F ∪ SEPARATING-F-CUTS(x̂, ẑ)          // Identifica desigualdades violadas
14      S = CALCULATE-SUGRANDIENT(x̂, ẑ)
15      θ = CALCULATE-STEP-SIZE(S, z-lb, z-ub, α)
16      UPDATE-LAGRANGEAN-MULTIPLIERS(S, θ, λ, μ, γ)
17      PURGE-INACTIVE-CUTS(F, S, γ)             // Retira os cortes inativos
18      if z-lb > best-lb
19          z-lb = best-lb
20          counter = 0
21      else
22          counter = counter + 1
23          if (counter MOD update-step-size) = 0
24              α = α/2
```

## 4.4 Experimentos com o *relax-and-cut*

Os experimentos aqui apresentados foram executados em uma máquina Intel Core i7 2620M, de 2.70 GHz e 8 GB de RAM. O código foi desenvolvido em linguagem C++ e compilado utilizando *Microsoft Visual Studio 2010 – Ultimate Edition*®. Os valores dos principais parâmetros do algoritmo estão listados abaixo:

- $\alpha$: utilizado no cálculo do tamanho do passo. Valor inicial igual a 2.
- $\lambda$: vetor inicial de multiplicadores de Lagrange, inicialmente fixado em zero.
- max_iter: número máximo de iterações do laço principal do algoritmo foi fixado em 2000.
- atualização do tamanho do passo (*update-step-size*): fixada em 200 iterações consecutivas sem melhorias no valor do melhor *lower bound* encontrado.

Esses parâmetros foram calibrados a partir de testes realizados em um outro conjunto de instâncias, criadas a partir do mesmo mecanismo gerador descrito na Seção 3.3.

| instância | | | MST | |
|---|---|---|---|---|
| $n$ | $l$ | $d$ | opt | $l_{\text{mst}}$ |
| 30 | 25 | 15 | 864 | 14 |
| 40 | 50 | 10 | 1395 | 19 |
| 60 | 50 | 15 | 1283 | 26 |
| 80 | 70 | 10 | 1869 | 32 |

Tabela 4.1: Características das instâncias de calibração

A Tabela 4.1 traz as principais características das instâncias utilizadas na calibração do algoritmo. As três primeiras colunas da tabela apresentam as informações básicas a respeito das instâncias de teste: quantidade de vértices no grafo, número mínimo de folhas e densidade de arestas. As duas colunas que se seguem trazem informações resumidas a respeito do custo da árvore geradora mínima (MST) e do número de folhas presentes na sua solução. Essa informação se torna pertinente nesse contexto haja vista que a MST pode ser interpretada como uma relaxação do LCMSTP e, logo, o valor da sua solução ótima será um *lower bound* natural para o problema.

Outro ponto que desperta bastante atenção está relacionado com a heurística Lagrangeana concebida para o problema (adaptação da heurísitca proposta por JULSTROM (2004) e já apresentada anteriormente), que busca construir uma solução

| | MIP | | | NDRC | | | $\text{NDRC}_{\text{ext}}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| $n\_l\_d$ | lp | opt | gap (%) | $\text{lb}^*$ | $\text{ub}^*$ | gap (%) | $\text{lb}^*$ | $\text{ub}^*$ | gap (%) |
| 30_25_15 | 1192 | 1218 | 2,13 | 1157 | 1218 | 5,01 | 1193 | 1218 | 2,05 |
| 40_50_10 | 1667 | 1693 | 1,54 | 1656 | 1693 | 2,19 | 1663 | 1693 | 1,77 |
| 60_50_15 | 1782 | 1828 | 2,52 | 1765 | 1831 | 3,60 | 1772 | 1831 | 3,22 |
| 80_70_10 | 2984 | 3084 | 3,24 | 2962 | 3099 | 4,42 | 2974 | 3088 | 3,69 |

Tabela 4.2: Instâncias de calibração do algoritmo NDRC

para o LCMSTP a partir de uma árvore geradora para o grafo $G$ — utilizando para isso o procedimento MAIS-FOLHAS. Nessa situação, é natural esperar que, quanto maior a quantidade de folhas contidas na solução da MST, mais fácil seria para a heurística construir uma solução viável para o LCMSTP. Da mesma forma melhor seria o valor da solução encontrada. Perceba que, pela análise da Tabela 4.1, a quantidade de folhas presente na MST é muito inferior à *quantidade mínima de folhas* exigidas para as instâncias do LCMSTP consideradas.

Na Tabela 4.2 estão sumarizados os principais testes de calibragem e adequação do algoritmo NDRC. A primeira coluna descreve as características gerais dos grafos de definição das instâncias. As colunas **lp**, **opt** e **gap**, que aparecem na sequência, apresentam os valores respectivamente da relaxação linear da formulação (4.15), utilizada como base para o algoritmo *relax-and-cut* implementado. Esse valor tem uma importância teórica grande, uma vez que, para problemas que apresentem a propriedade de integralidade, o *bound* teórico de relaxação Lagrangeana deverá ser igual àquele de relaxação linear (BEALSEY (1993) e WOLSEY (1998)). Nesse caso, quanto menor a diferença entre o *lower bound* fornecido pelo NDRC e o valor da relaxação linear, maior será a qualidade do algoritmo NDRC.

As próximas três colunas da tabela, $\mathbf{lb}^*$, $\mathbf{ub}^*$ e **gap** (que aparecem logo abaixo do cabeçalho **NDRC**) indicam, respectivamente, o melhor limitante inferior encontrado, o valor da melhor solução viável encontrada pela heurística Lagrangeana, e o *gap* entre essas duas grandezas. Esses valores foram obtidos pelos experimentos com a implementação direta do algoritmo *relax-and-cut* a partir da formulação (4.15). As três últimas colunas, finalizando a tabela, apresentam os resultados da aplicação do mesmo algoritmo sobre uma formulação ligeiramente diferente (aqui chamada de formulação estendida), obtida a partir da formulação original, acrescida da seguinte restrição:

$$\sum_{e \in \delta(i)} x_e + z_i \geq 2 \qquad \forall i \in V. \tag{4.25}$$

Conforme apontado por LUCENA *et al.* (2010), a família de desigualdades (4.25) é utilizada para explicitar a ativação de uma folha $i$ caso exista no máximo uma aresta partindo desse vértice. Entretanto, embora essa condição seja satisfeita por qualquer solução ótima de (4.15), a sua inclusão no modelo subjacente teve um impacto bastante positivo na melhoria da performance do algoritmo, como se pode observar da análise da tabela 4.2. Esse impacto foi percebido tanto em termos no *lower bound* quanto do *upper bound* encontrado.

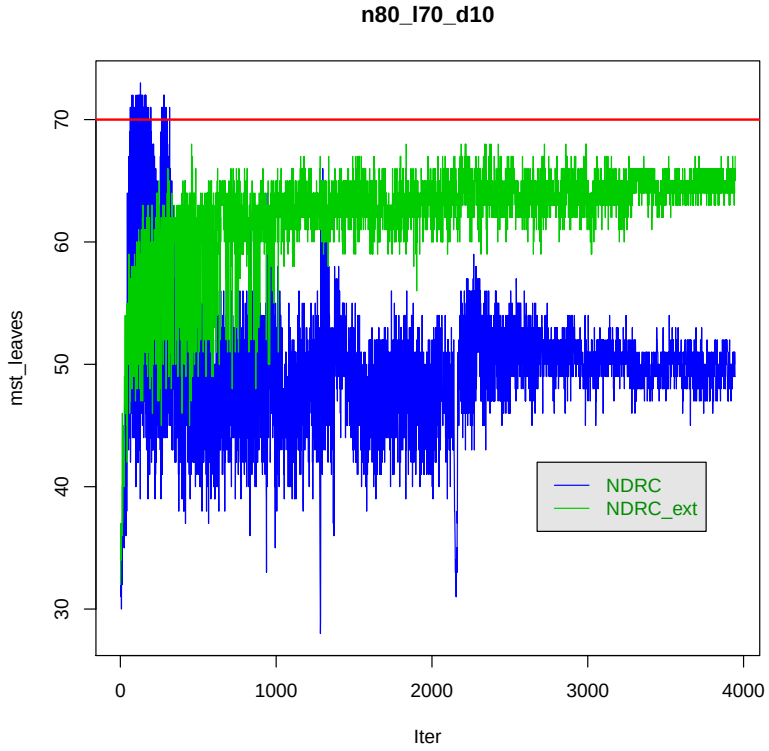


Figura 4.1: Evolução do número de folhas na MST

Sob o aspecto da quantidade de folhas presentes na solução da MST, a Figura 4.1 busca revelar o impacto, no desempenho do NDRC, causado pela inclusão da família de desigualdades (4.25). Isso é feito para a instância contendo 80 vértices, utilizada na calibragem do algoritmo. Importante lembrar que, a cada iteração do NDRC, uma nova árvore geradora de custo Lagrangeano mínimo deverá ser calculada (veja a passagem a respeito da *resolução do subproblema em* $x$, descrita na Seção 4.3), e o número de folhas presentes nessa árvore é então contado. Na figura, a linha em vermelho marca a quantidade mínima de folhas exigida para uma solução do LCMSTP, e foi incluida no gráfico apenas para orientar a análise. A série indicada em azul mostra a evolução do número de folhas presentes na solução da árvore de custo mínimo calculada ao longo das interações do NDRC. Em verde, é exibida a série indicando a evolução da contagem do número de folhas da MST resultando da formu-

| instância | | | | $NDRC_{ext}$ | | gap (%) | |
|---|---|---|---|---|---|---|---|
| n | l | d (%) | opt | lb | ub | lb/ub | (ub/opt)-1 |
| 30 | 24 | 30 | 923 | 923 | **923** | 0,00 | 0,00 |
| 40 | 32 | 30 | 1146 | 1140 | **1146** | 0,52 | 0,00 |
| 50 | 40 | 30 | 1039 | 1005 | 1052 | 4,47 | 1,25 |
| 60 | 48 | 30 | 1762 | 1632 | **1762** | 7,38 | 0,00 |
| 70 | 56 | 30 | 2018 | 1923 | **2018** | 4,71 | 0,00 |
| 80 | 64 | 30 | 2200 | 2106 | 2217 | 5,01 | 0,77 |
| 90 | 72 | 30 | 2233 | 2127 | 2280 | 6,71 | 2,10 |
| 100 | 80 | 30 | 2662 | 2579 | 2676 | 3,62 | 0,53 |
| 30 | 24 | 70 | 620 | 560 | **620** | 9,68 | 0,00 |
| 40 | 32 | 70 | 691 | 646 | **691** | 6,51 | 0,00 |
| 50 | 40 | 70 | 722 | 679 | 731 | 7,11 | 1,25 |
| 60 | 48 | 70 | 1214 | 1190 | **1214** | 1,98 | 0,00 |
| 70 | 56 | 70 | 1359 | 1238 | 1384 | 10,55 | 1,84 |
| 80 | 64 | 70 | 1412 | 1377 | 1493 | 10,45 | 5,74 |
| 90 | 72 | 70 | 1452 | 1329 | 1455 | 8,66 | 0,21 |
| 100 | 80 | 70 | 1676 | 1509 | 1707 | 11,95 | 1,85 |
| | | | | | Média | 6,21 | 0,97 |

Tabela 4.3: Resultados da aplicação do $NDRC_{ext}$

lação (4.15) fortalecida pela inclusão da família de desigualdades (4.25) (**$NDRC_{ext}$**). Fica evidente, da análise da figura que, embora se trate de uma família de restrições redundantes (em termos da solução ótima do problema), a sua utilização é capaz de melhorar a performance do algoritmo, medida em termos da aproximação das soluções da árvore geradora mínima obitdas e do LCMSTP (quando comparadas as quantidades de folhas obtidas na árvore construida e aquela exigida pelo LCMSTP). O mesmo comportamento foi observado para outras instâncias testadas.

Finalizando os experimentos com os algoritmos *relax-and-cut*, a Tabela 4.3 sintetiza os principais resultados obtidos ao submetermos à versão estendida do NDRC o mesmo conjunto de instâncias utilizado para testar os algoritmos *branch-and-cut*. Nesse caso, o destaque deve ser dado à qualidade das soluções primais construídas a partir de nossa heurística Lagrangeana, que é extremamente simples. As soluções por ela geradas obtiveram gap inferior a 1%, quando comparadas às soluções ótimas.

# Capítulo 5

# Conclusões

Esse trabalho teve como linha mestre principal o estudo de diferentes mecanismos de modelagem de problemas de árvores em grafos com algum tipo de restrição na quantidade de folhas, tendo sido inspirado e motivado pelos trabalhos de FUJIE (2003), LUCENA *et al.* (2010) e GOUVEIA e TELHADA (2011). Torna-se evidente, como resultado desse trabalho, que o principal ingrediente para a construção de bons algoritmos exatos para essa família de problemas está relacionado com a forma de se modelar as folhas de uma árvore sobre as quais são impostas algum tipo de condição específica.

Para a família de problemas objeto de estudo desse trabalho, o interesse maior voltou-se ao estudo de dois casos particulares, justificado pela importância atribuída aos mesmos (medida em termos do interesse que despertam na literatura). São eles, o problema da árvore geradora com número máximo de folhas e o problema da árvore geradora mínima com um número mínimo de folhas.

Para esses dois problemas, foram apresentados e discutidos três modelos bastante conhecidos na literatura que apresentam forte relação entre si.

Como resultado de um breve estudo poliedral das formulações, fomos capazes de introduzir um conjunto de desigualdades válidas (2.31) (e a sua contrapartida para o caso direcionado (2.32)), capazes de fortalecer os limitantes de relaxação linear associados às formulações, em especial aqueles relativos à formulação de FUJIE (2003), modelada sobre um grafo não orientado.

Poderíamos citar, ainda como contribuição desse trabalho, a introdução de uma nova formulação para os problemas, diferentes daquelas já conhecidas na literatura, e para a qual acreditamos que ainda exista possibilidade enorme de evolução e de melhoria.

Ao todo foram implementados quatro algoritmos de *branch-and-cut*, e um algortimo de Relaxação Lagrangeana para o LCMSTP baseado na formulação não direcionada de FUJIE (2003). Mais uma vez, tanto quanto sabemos, acreditamos que essa tenha sido a primeira tentativa de tratamento dessa classe de problemas

através de um algoritmo *relax-and-cut*, cuja performance foi mais uma vez verificada. Como destaque para essa parte da investigação, mencionamos a capacidade de combinar informações duais e primais na construção de boas soluções viáveis para o problema, mesmo quando utilizando heurísticas primais bastantes simples.

Em relação ao algoritmo Lagrangeano que implementamos, a avaliação do impacto da adição da família de desigualdades (4.25) no comportamento do NDRC (Figura 4.1) nos leva a conjecturar que, para algoritmos do tipo *relax-and-cut*, a caracterização de famílias de desigualdades (mesmo que redundantes) pode ser benéfica para o desempenho do algoritmo. Talvez seja possível descrever mais algumas famílias de desigualdades que, embora redundantes para o problema, possam contribuir positivamente para os resultados obtidos pela aplicação do algoritmo.

Finalmente, a análise dos experimentos com a nova formulação aqui proposta nos permitiu concluir que se trata de uma abordagem promissora com um enorme potencial de melhoria, especialmente a partir da possibilidade de investigação de novas famílias de desigualdades válidas que poderiam fortalecer ainda mais os *bounds* de relaxação linear, fazendo com que a modelagem se torne ainda mais competitiva. A investigação dessas famílias de desigualdades poderia ser conduzida em um trabalho futuro, juntamente com o desenvolvimento de um algoritmo do tipo *relax-and-cut* baseado nessa formulação.

Mencionamos ainda, como proposta para trabalhos futuros, a adaptação da nossa formulação a problemas como o Problema da Árvore Geradora Mínima com Mínima Restrição de Grau. Da mesma forma, nos parece interessante investigar procedimentos *relax-and-cut* para esses problemas, ao estilo daquele que aqui introduzimos.

# Referências Bibliográficas


BIGGS, N., LLOYD, E. K., WILSON, R. J. *Graph Theory 1736-1936.* Oxford University Press, 1986.

HARARY, F. *Graph Theory.* Addison-Wesley, 1994.

GRAHAM, R. L., HELL, P. "On the History of the Minimum Spanning Tree Problem", *Annals of the History of Computing*, v. 7, n. 1, pp. 43–57, 1985.

PADBERG, M. W., WOLSEY, L. A. "Trees and cuts", *Annals of Discrete MAthematics*, v. 17, pp. 511–517, 1983.

BORUVKA, O. "O jistém problému minimálním (about a certain minimal problem)", *Práca Moravské Přirodovědecké Společnosti*, v. 3, pp. 37–58, 1926. (In czech).

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

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

CHRISTOFIDES, N. *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem.* Relatório técnico, Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.

NARULA, S. C., HO, C. A. "Degree-constrained minimum spanning tree", *Computers and OR*, v. 7, n. 4, pp. 239–249, 1980.

ACHUTAN, N. R., CACCETTA, L. "Minimum Weight Spanning Trees with Bounded Diameter", *Australasion Journal of Combinatorics*, v. 5, pp. 261–276, 1990.

JULSTROM, B. A. "Codings and Operators in Two Genetic Algorithms for the leaf-constrained minimum spanning tree problem", *International Journal*

*of Applied Mathematics and Computer Science*, v. 14, n. 3, pp. 385–396, 2004.

DEO, N., MICIKEVICIUS, P. "A Heuristic for a Leaf Constrained Minimum Spanning Tree Problem", *Congressus Numerantium*, v. 141, pp. 61–72, 1999.

GOUVEIA, L., TELHADA, J. "Reformulation by IntersectionMethod on the MST Problem with Lower Bound on the Number of Leaves", *Lecture Notes in Computer Science*, v. 6701, pp. 83–91, 2011.

HOELTING, C. J., SCHOENEFELD, D. A., WAINWRIGHT, R. L. "Approximation Techniques for Variations of the $p$-Median Problem". In: *Proceedings ACM Symposium on Applied Computing*, pp. 293–299, 1995.

FERNANDES, L. M., GOUVEIA, L. "Minimal Spanning Trees with a constraint on the number of leaves", *European Journal of Operational Research*, v. 104, pp. 250–261, 1998.

EDELSON, W., GARGANO, M. L. "Feasible Encodings for GA solutions of Constrained Minimum Spanning Tree Problems". In: *Late-Braking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pp. 82–89, 2000.

FUJIE, T. "An exact algorithm for the maximum spanning tree problem", *Computers & Operations Research*, v. 30, pp. 1391–1944, 2003.

FUJIE, T. "The Maximum Leaf Spanning Tree Problem: Formulations and Facets", *Networks*, v. 43, n. 4, pp. 212–223, 2004.

LUCENA, A., MACULAN, N., SIMONETTI, L. "Reformulations and solution algorithms for the maximum spanning tree problem", *Computational Management Science*, v. 7, pp. 289–311, 2010.

GAREY, M., JOHNSON, D. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, 1976.

LEMKE, P. "The maximum leaf spanning tree problem for cubic graphs is $\mathcal{NP}$-Complete", *IMA preprint Series*, 1988.

GALBIATI, G., MAFFIOLI, F., MORZENTI, A. "A short note on the approximability of the maximum leaves spanning tree problem", *Information Processing Letters*, v. 52, n. 1, pp. 45–49, 1994.

LU, H.-I., RAVI, R. "Approximating Maximum Leaf Spanning Trees in Almost Linear Time", *Journal of Algorithms*, v. 29, pp. 132–131, 1998.

SOLIS-OBA, R. "2-Approximation Algorithm for Finding Spanning Tree with Maximum Number of Leaves", *Lecture Notes in Cmputer Science*, v. 1461, pp. 441–452, 1998.

GUHA, S., KHULLER, S. "Approximation Algorithms for Connected Dominating Sets", *Algorithmica*, v. 20, n. 4, pp. 374–387, 1998.

STORER, J. A. "Constructing Full Spanning Trees for Cubic Graphs", *Information Processing Letters*, v. 13, n. 1, pp. 8–11, 1981.

CHEN, S., LJUBIC, I., RAGHAVAN, S. "The regenerator location problem", *Networks*, v. 55, n. 3, pp. 205–220, 2010.

MINIEKA, E. "The Centers and Medians of a Graph", *Operations Research*, v. 25, n. 4, pp. 641–650, 1977.

MAGNANTI, T. L., WOLSEY, L. A. "Optimal Trees". In: Ball, M., Magnanti, T., Monma, C., et al. (Eds.), *Handbooks In Operations Research and Management Science*, v. 7, Elsevier, cap. 9, pp. 503–616, 1995.

MITCHEL, J. E. "Branch-and-Cut algorithms for combinatorial optimization problems". In: Pardalos, P. M., Resende, M. G. C. (Eds.), *Handbook of applied optimization*, cap. 3.3, pp. 65–77, Oxford University Press, Inc., 2002.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. *Introduction to Algorithms*. The MIT Press, 1990.

WOLSEY, L. A. *Integer Programming*. John Wiley & Sons, 1998.

LAND, A., DOIG, A. "An automatic method for solving discrete programing problems", *Econometrica*, v. 28, pp. 497–520, 1960.

LITTLE, J., MURTY, K., SWEENEY, D., et al. "An algorithm for the traveling salesman problem", *Operations Research*, v. 11, n. 5, pp. 972–989, 1963.

DAKIN, R. "A tree search algorithm for the mixed integer programming problemns", *Computer Journal*, v. 8, pp. 250–255, 1965.

MACULAN, N. "Integer Programming". In: Pardalos, P. M., Resende, M. G. C. (Eds.), *Handbook of applied optimization*, pp. 431–445, Oxford University Press, Inc., 2002.

DANTZIG, G. B., FULKERSON, D. R., JONHSON, S. M. "Solution of large scale traveling salesman problem", *Operations Research*, v. 2, pp. 393–410, 1954.

GOMORY, R. E. "Outline of an algorithm for integer solutions to linear programs", *Bulletin of American Mathematical Society*, v. 64, pp. 275–278, 1958.

GOMORY, R. E. *An algorithm for the mixed integer problem.* Relatório técnico, The Rand Corporation, 1960.

CHVÁTAL, V. "Edmonds Polytopes and a Hierarchy of Combinatorial Problems", *Discrete Mathematics*, v. 4, pp. 305–337, 1973.

PADBERG, M. W., RINALDI, G. "Optimization of a 532-city traveling salesman problem by branch and cut", *Operations Research Letters*, v. 6, pp. 1–8, 1987.

PADBERG, M. W., RINALDI, G. "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems", *SIAM Review*, v. 33, n. 1, pp. 60–100, 1991.

GRÖTSCHEL, M., HOLLAND, O. "Solution of large-scale travelling salesman problems", *Mathematical Programming*, v. 51, n. 2, pp. 141–202, 1991.

APPLEGATE, D. L., BIXBY, R. E., CHVÁTAL, V., et al. *The traveling Salesman Problem: a computational study.* Princeton: Princeton University Press, 2006.

FORD, L. R., FULKERSON, D. R. *Flows in Network.* Princeton University Press, 1962.

GOLDBERG, A. V. *Efficient Graph Algorithms for Sequential and Parallel Computers.* Tese de Doutorado, Department of Electrical Engineering and Computer Science, MIT, 1982.

GOLDBERG, A., TARJAN, R. E. "A new approach to the maximum flow problem". In: *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pp. 136–146, 1986.

FORD, L. R., FULKERSON, D. R. "Maximal flow through a network", *Canadian Journal of Mathematics*, v. 8, pp. 399–404, 1956.

ELIAS, P., FEINSTEIN, A., SHANNON, C. "A note on the maximum flow through a network", *Information Theory, IEEE Transactions on*, v. 2, n. 4, pp. 117–119, 1956.

STOER, M., WAGNER, F. "A simple min-cut algorithm", *Journal of the ACM*, v. 44, n. 4, pp. 585–591, 1997.

HAO, J., ORLIN, J. B. "A faster algorithm for finding the minimum cut in a graph". In: *3rd ACM-SIAM Syposuim on Discrete Algorithms*, pp. 921–940, 1992.

NAGAMOCHI, H., IBARAKI, T. "Computing Edge-Connectivity in mutigraphs and capacitated graphs", *SIAM Journal on Applied Mathematics*, v. 5, pp. 54–66, 1992.

IBM. "Callable Library". `http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r2/index.jsp?topic=%2Filog.odms.cplex.help%2FContent%2FOptimization%2FDocumentation%2FCPLEX%2F_pubskel%2FCPLEX297.html`, 2013. [Online; último acesso em 16 de fevereiro de 2014].

COIN-OR. "Lemon Graph Library". `http://lemon.cs.elte.hu/trac/lemon`, 2003-2013. [Online; último acesso em 16 de fevereiro de 2014].

FISCHETTI, M., MONACI, M. "Exploiting Erraticism in Search", *Operations Research - Articles in Advance*, v. 1461, pp. 1–9, 2014.

FISCHER, M. L. "The lagrangian relaxation method for solving integer programming problems", *Management Science*, v. 27, pp. 1–18, 1981.

LORIE, J., SAVAGE, L. J. "Three problems in capital rationing", *Journal of Business*, v. 28, n. 6, pp. 229–239, 1973.

EVERETT, H. "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources", *Operations Research*, v. 11, pp. 399–417, 1963.

HELD, M., KARP, R. M. "The traveling salesman problem and the minimum spanning trees", *Operations Research*, v. 18, pp. 1138–1162, 1970.

HELD, M., KARP, R. M. "The traveling salesman problem and the minimum spanning trees: part II", *Mathematical Programming*, v. 1, pp. 06–25, 1971.

GEOFFRION, A. M. "Lagrangean Relaxation for integer programming", *Mathematical Programming Study*, v. 2, pp. 82–114, 1974.

DANTZIG, G. B., WOLFE, P. "The decomposition algorithm for linear programming", *Operations Research*, v. 8, pp. 101–111, 1960.

BOYD, S., VANDENBERGHE, L. *Convex Optimization*. Cambridge University Press, 2003.

HELD, M., WOLFE, P., CROWDER, H. D. "Validation of subgradient optimization", *Mathematical Programming*, v. 6, pp. 62–88, 1974.

LAMARÉCHAL, C. "Nondifferentiable optimization". In: Nemhauser, G. L., Kan, A., Todd, M. (Eds.), *Optimization, Handbooks In Operations Research*, North Holland, pp. 529–572, 1989.

BARAHONA, F., ANBIL, R. "The volume algorithm: producing primal solutions with a subgradient method", *Mathematical Programming*, v. 87, pp. 385–399, 2000.

BEALSEY, J. E. "Lagrangian Relaxation". In: Reeves, C. (Ed.), *Modern heuristic techniques for combinatorial problems*, Oxford: Blackwell Scientific Press, pp. 243–303, 1993.

BERTSEKAS, D. P. *Non Linear Programming*. Athena Scientific, 1999.

LUCENA, A. "Steiner Problem in Graphs: Lagrangian Relaxation and Cutting Planes", *COAL Bulletin*, v. 21, pp. 2–8, 1992.