Universidade Federal do Ceará
Departamento de Computação
Programa de Mestrado E Doutorado em Ciência da Computação

# Algoritmos para o Problema de Localização Simples Baseados nas Formulações Clássica e Canônica

**Autor**
**Fábio Carlos Sousa Dias**

Fortaleza – Ceará
Julho 2008

Universidade Federal do Ceará
Departamento de Computação
Programa de Mestrado E Doutorado em Ciência da Computação

# Algoritmos para o Problema de Localização Simples Baseados nas Formulações Clássica e Canônica

**Autor**
**Fábio Carlos Sousa Dias**

**Orientador**
Prof. Dr. Manoel Bezerra Campelo Neto

*Dissertação apresentada à Coordenação do Programa de Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de* ***Mestre em Ciência da Computação****.*

Fortaleza – Ceará
Julho 2008

# Agradecimentos

O autor Este trabalho é parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e pelo Projeto ALPHA-II.

# Resumo

Neste trabalho, estudamos o problema de localização simples (*SPLP - Simple Plant Location Problem*). Usando a formulação matemática clássica e uma outra formulação proposta recentemente, desenvolvemos vários algoritmos para encontrar limites inferiores e superiores, bem como algoritmos tipo *branch-and-bound*. Com a formulação clássica, tais limites são obtidos utilizando o método de correção de dados e critérios de dominância entre os custos fixos e de transporte. Propomos uma projeção dessa formulação, que se mostrou computacionalmente atrativa. Usando a nova formulação propomos e mostramos a corretude de vários procedimentos iterativos que procuram encontrar uma solução para o problema, resolvendo uma seqüência de subproblemas paramétricos obtidos com a remoção de variáveis e restrições da formulação original. Em cada iteração desse processo, podemos gerar limites inferiores e superiores. Aplicamos ainda relaxação lagrangeana a essa nova formulação para obter outros limites. Analisamos várias possibilidades de relaxação das restrições. Desenvolmento também algoritmos *branch-and-bound* baseados em ambas as formulações e nos limites obtidos. Avaliamos a eficiência computacional de todos os algoritmos com instâncias de teste difíceis, disponíveis na literatura. Resultados computacionais e comparações com outros algoritmos da literatura são reportados.

**PALAVARAS CHAVE. Problemas de localização de facilidades, Relaxação Lagrangeana, Redução do Tamanho do Problema.**

# Abstract

In this work, we study the Simple Plant Location Problem (SPLP). Using its classical mathematical programming formulation and another recently proposed formulation, we develop several algorithms to find lower and upper bounds for the problem as well as branch-and-bound algorithms. With the classical formulation, such bounds are obtained via the data correction method and dominance criteria between fixed and transportation costs. We propose a projection of this formulation that has shown to be computationally atractive. Using the new formulation, we propose and prove the correctness of several iterative procedures that attempt to find an optimal solution to the problem by solving a sequence of parametric sub-problems, each one obtained by removing some variables and constraints of the original formulation. At each iteration of this process, we can obtain lower and upper bounds. We also apply Lagrangean relaxation to this new formulation in order to get other bounds. We consider several possibilities of relaxing the constraints. In addition, we develop branch-and-bound algorithms based on both formulations and the obtained bounds. We evaluate the computational efficiency of all proposed algorithms with hard test instances from the literature. Computational results are reported and comparisons with other algorithms from the literature are carried out.

**KEYWORDS. Facility Location Problems, Lagrangean Relaxation, Problem Reduction.**

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Em vários contextos nos deparamos com situações em que se deseja decidir onde instalar entidades (usualmente denominadas de facilidades) fornecedoras de produtos e/ou serviços demandados por um conjunto de consumidores e como atendê-los a partir das facilidades instaladas. Comumente deseja-se que tal decisão leve a minimizar o custo total de instalação e distribuição. Esses problemas são chamados genericamente de problemas de localização de facilidades.

Neste trabalho vamos considerar as principais variações deste problema, concentrando a atenção no Problema de Localização Simples (*Simple Plant Location Problem - SPLP*), ou ainda, chamado Problema de Localização Não Capacitado por alguns autores. No Capítulo 2, vamos apresentar as definições e formulações matemáticas clássicas destes problemas. Realizamos uma breve revisão bibliográfica, destacando os trabalhos mais recentes e aqueles com maior interseção com o nosso desenvolvimento. Esta revisão não pretende ser exaustiva, dada a grande quantidade e variedade de trabalhos existentes na literatura, mas procura mostrar que o tema continua atual e desafiador. Possivelmente, o interesse por esta classe de problemas é motivado por sua dificuldade de solução e pela gama de aplicações que modela. Particularmente o *SPLP*, a versão sobre a qual trabalharemos, é um problema NP-Díficil, que se relaciona com vários outros problemas clássicos, como coloração, cobertura e particionamento de conjuntos.

Ainda no próximo capítulo, apresentamos um dos ingredientes mais frequentes em algoritmos para resolver problemas de localização: os testes *ADD/DROP*. Estes testes procuram definir, de forma ótima ou heurística, o satus final das facilidades. Esta estratégia relaciona-se ao fato de que o problema de localização de facilidades (*PLF*) pode ser decomposto em dois subproblemas inter-relacionados: o problema de localização propriamente, onde queremos identificar as facilidades a serem instaladas, e o problema de distribuição, que define como os consumidores serão atendidos a partir das facilidades abertas. Para cada "solução"do primeiro, uma "solução"para o segundo é determinada por um problema de transporte (ou mesmo de alocação, para o *SPLP*). Neste sentido, boas regras para determinação dos stati das facilidades pode ser de grande valia. Na Seção 2.4 descrevemos com detalhes os testes *ADD/DROP*, condições suficientes de otimalidade e como podem ser aplicados iterativamente.

Concluindo o capítulo, descrevemos três conjuntos de instâncias de teste, disponíveis na literatura, que serão usadas para avaliação dos algoritmos que desenvolvemos. Essas instâncias apresentam características variadas, com relação à esparsidade, à dimensão, à estrutura dos custos e a outros aspectos que afetam a dificuldade de solução.

A partir do Capítulo 3, restringiremos o estudo ao *SPLP*. Começamos explorando sua formulação clássica, que essencialmente ressalta a visão do problema como composição dos dois subproblemas interdependentes citados acima. Revisitamos a formulação, com o objetivo de reduzir a quantidade de variáveis e restrições utilizados. Apresentamos um modelo que corresponde a uma projeção do original. Particularizamos os testes *ADD/DROP* para o *SPLP* e, principalmente, introduzimos algoritmos para avaliação mais eficiente deles quando aplicados de forma iterativa. Apresentamos novas heurísticas *ADD/DROP*, que procuram estabeler decisões mais acertadas que as heurísticas originais, ou seja, procuram determinar melhores limites superiores, empregando esforço computacional similar. Utilizamos ainda estes testes *ADD/DROP* combinados com a técnica de *correção de dados*, usada por Goldegorin, Ghosh e Serksma[23], para obtenção de limites inferiores. Estes elementos são incorporados a algoritmos do tipo *branch-and-bound*.

Relatamos experimentos computacionais com os algoritmos propostos e comparamos os resultados com os da literatura.

No Capítulo 4, realizamos desenvolvimento similar, agora baseados em uma reformulação do *SPLP*, proposta por Labbé e Marin[34], como um problema de cobertura de conjuntos. Seguindo uma sequência análoga ao do capítulo anterior, apresentamos uma estratégia para redução do tamanho da formulação, sugerida por Labbé e Marin[34], propomos procedimentos para cálculo de limites inferiores e superiores e definimos algoritmos do tipo *branch-and-bound*. Entretanto, a metodologia empregada para a construção desses algoritmos é bastante diferente. Ela baseia-se na definição de um problema paramétrico, que introduzimos na Subseção 4.2, e na escolha da sequência de problemas paramétricos que serão resolvidos. Propomos e avaliamos algumas das possíveis opções através de experimentos computacionais, cujos resultados são apresentados e comparados.

Para definir uma terceira classe de algoritmos para o *SPLP*, aplicamos relaxação lagrangeana à formulação de Labbé e Marin[34]. Embora haja várias experiências neste campo com a formulação clássica, ao nosso conhecimento, este é o primeiro estudo com relaxação lagrangeana aplicada à nova formulação. Recapitulamos as principais idéias desta técnica no Capítulo 5 para então a usarmos no Capítulo 6. Consideramos alguns possíveis duais lagrangeanos, relaxando diferentes grupos de restrições e acrescetando ou não restrições válidas para fortalecer o limite inferior obtido. Propomos algumas heurísticas lagrangeanas simples, para estabelecer o gap de dualidade. Novamente, implementamos e avaliamos computacionalmente os procedimentos propostos.

Finalmente, fazemos uma análise geral dos resultados obtidos no Capítulo 7.

# Capítulo 2

# Problema de Localização de Facilidades

Neste capítulo, vamos apresentar o problema geral de localização de facilidades e os principais casos particulares, suas definições e formulações matemáticas clássicas. Realizaremos uma breve revisão bibliográfica, recapitulando um pouco do histórico dos estudos sobre o problema e, principalmente, destacando os trabalhos mais recentes, que demonstram a atualidade do tema, apesar de toda a extensão e variada gama de trabalhos registrados na literatura.

O interesse constante pelo problema pode ser explicado em duas vias: primeiro, por sua dificuldade de resolução (mesmo sua versão mais simples é NP-Difícil) e segundo, pela capacidade de modelar ou aparecer como subproblemas de vários outros problemas.

Ainda neste capítulo, apresentaremos técnicas de redução do tamanho do problema geral de localização, normalmente descritas na literatura para casos particulares. Aqui, estabeleceremos os conhecidos testes de redução e mostraremos sua corretude, segundo uma abordagem unificada.

Finalmente, descreveremos vários tipos de instâncias de testes encontradas na literatura, centralizando nossa atenção nos problemas chamados não-euclideanos. Estas instâncias serão usadas ao longo de todo o trabalho para avaliar o desempenho dos algoritmos

que vamos propor nos capítulos subseqüentes.

## 2.1 Descrição

Em vários contextos, aparecem problemas nos quais é preciso decidir qual a forma mais vantajosa de onde instalar entidades (usualmente denominadas de facilidades) fornecedoras de produtos e/ou serviços e de como distribuir esses itens, a partir das facilidades instaladas, para centros consumidores de uma determinada região. Esses problemas são chamados genericamente de problemas de localização de facilidades.

Mais precisamente, conforme Mateus e Carvalho[36], o problema de localização de facilidades pode ser definido como aquele no qual facilidades devem ser alocadas entre $n$ possíveis locais de instalação, com o objetivo de minimizar o custo total em satisfazer a demanda distribuída em $m$ locais de consumo. No custo total são computados os custos fixos de instalação das facilidades e os custos variáveis de atendimento da demanda pelas facilidades, também chamados custos de distribuição, alocação ou transporte.

Dentro dessa definição abrangente, podem-se enquadrar muitas variações do problema, cujas diferenças principais dizem respeito à capacidade e ao custo de instalação das facilidades, ao número máximo de facilidades que podem ser instaladas e à distribuição dos possíveis locais de instalação no espaço e em relação aos pontos de demanda.

Essas várias possibilidades ensejam classificações do problema de acordo com suas características. Assim, por exemplo, falamos de problema capacitado ou não-capacitado, conforme cada oferta seja suficiente ou não para atender toda a demanda. Taxionomias completas do problema de localização podem ser encontradas em Daskin[17] e Brandeau e Chiu[12].

Enquadrados nessa classe, encontram-se problemas específicos que descreveremos a seguir. São eles: o problema de localização capacitado, não-capacitado, $p$-medianas e de localização simples (*SPLP - Simple Plant Location Problem*).

## 2.2 Formulação Matemática

Nesta seção, descreveremos formalmente os quatro problemas de localização citados acima, apresentando suas formulações matemáticas clássicas. A partir de agora iremos utilizar os termos *aberta* para designar uma facilidade que foi escolhida para ser instalada e *fechada* para designar uma facilidade que não foi escolhida para ser instalada. Além disso, iremos utilizar o termo *melhor* facilidade para atender uma demanda $j$ para designar a facilidade com o menor custo de transporte para atender $j$.

Vamos denotar por $I = \{1, \ldots, n\}$ o conjunto de facilidades candidatas à instalação, por $J = \{1, \ldots, m\}$ o conjunto de centros de demanda e por $p$ o número máximo de facilidades que podem ser instaladas. A capacidade de oferta da facilidade $i \in I$ é denotada por $A_i$, enquanto $B_j$ representa a demanda do centro consumidor $j \in J$. Já $f_i$ expressa o custo fixo para instalar a facilidade $i \in I$ e $c_{ij}$, o custo para a facilidade $i \in I$ atender toda a demanda em $j \in J$. Assim, definindo as variáveis $x_{ij}$, para representar o percentual da demanda do consumidor $j \in J$ atendida pela facilidade $i \in I$, e $y_i$, uma variável binária igual a 1 se, e somente se, a facilidade $i$ é aberta, o *Problema de Localização de Facilidades* $(PLF)$ pode ser formulado como:

$$(PLF) \qquad \min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \tag{2.1}$$

$$\text{sujeito a:} \sum_{i \in I} x_{ij} = 1, \forall j \in J, \tag{2.2}$$

$$\sum_{j \in J} B_j x_{ij} \leq A_i y_i, \forall i \in I, \tag{2.3}$$

$$\sum_{i \in I} y_i \leq p, \tag{2.4}$$

$$x_{ij} \geq 0, \forall i \in I, \forall j \in J, \tag{2.5}$$

$$y_i \in \{0, 1\}, \forall i \in I. \tag{2.6}$$

A função objetivo (2.1) minimiza a soma dos custos fixos e variáveis. Em (2.2), garante-se que a demanda em cada centro $j \in J$ seja totalmente atendida. Por outro

lado, (2.3) garante que uma facilidade, se for aberta, poderá atender até o máximo de sua capacidade de oferta. A restrição (2.4) limita o número de facilidades abertas a $p$, enquanto (2.5)-(2.6) definem os domínios das variáveis.

O modelo acima define, para alguns autores, o *Problema de Localização Capacitado* ($PLC$). Mais freqüentemente, porém, a restrição (2.4) é desconsiderada na descrição deste problema ([1],[13],[35]), opção que faremos aqui sempre que nos referimos a $PLC$, que é então formulado como:

$$(PLC) \quad \min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \tag{2.7}$$

$$\text{sujeito a:} \sum_{i \in I} x_{ij} = 1, \forall j \in J, \tag{2.8}$$

$$\sum_{j \in J} B_j x_{ij} \leq A_i y_i, \forall i \in I, \tag{2.9}$$

$$x_{ij} \geq 0, \forall i \in I, \forall j \in J, \tag{2.10}$$

$$y_i \in \{0, 1\}, \forall i \in I. \tag{2.11}$$

Quando consideramos que a oferta de cada facilidade é suficiente para atender a demanda total, ou seja,

$$A_i \geq \sum_{j \in J} B_j, \forall i \in I \tag{2.12}$$

definimos o *Problema de Localização Não-Capacitado* ($PLNC$). Neste caso, as variáveis $x_{ij}$ podem ser consideradas binárias, uma vez que cada consumidor poderá ser sempre atendido integralmente pela melhor facilidade aberta. Assim, geramos o modelo:

$$(PLNC) \quad \min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \tag{2.13}$$

$$\text{sujeito a:} \sum_{i \in I} x_{ij} = 1, \forall j \in J, \tag{2.14}$$

$$x_{ij} \leq y_i, \forall i \in I, \forall j \in J, \tag{2.15}$$

$$\sum_{i \in I} y_i \leq p, \tag{2.16}$$

$$x_{ij} \in \{0,1\}, \forall i \in I, \forall j \in J, \tag{2.17}$$

$$y_i \in \{0,1\}, \forall i \in I. \tag{2.18}$$

Considerando que $p \geq |I|$ ou equivalentemente, eliminando a restrição que limita o número de facilidades instaladas, obtemos o modelo para *Simple Plant Location Problem* (*SPLP*):

$$(SPLP) \quad \min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \tag{2.19}$$

$$\text{sujeito a:} \sum_{i \in I} x_{ij} = 1, \forall j \in J, \tag{2.20}$$

$$x_{ij} \leq y_i, \forall i \in I, \forall j \in J, \tag{2.21}$$

$$x_{ij} \in \{0,1\}, \forall i \in I, \forall j \in J, \tag{2.22}$$

$$y_i \in \{0,1\}, \forall i \in I. \tag{2.23}$$

Vale observar que os problemas de localização definidos acima podem ser igualmente descritos sobre um grafo, onde os vértices representam os pontos de oferta e/ou consumo. Esta opção torna-se mais interessante quando há vários centros que são simultaneamente de oferta e demanda e no caso extremo em que $I = J$. Este é, por exemplo, o caso do problema de $p$-medianas (*P-MED*), onde se deseja determinar um subconjunto $S \subseteq I = J$ de exatamente $p$ facilidades que atendem as demandas em $I - S$ a custo de transporte mínimo. Formalmente, o modelo pode ser obtido de *PLNC* considerando $I = J$, $f_i = c_{ii} = 0$ para todo $i \in I$, e transformando (2.16) em uma igualdade, ou seja,

$$(P\text{-}MED) \quad \min \sum_{i \in I} \sum_{j \in I} c_{ij} x_{ij} \tag{2.24}$$

$$\text{sujeito a:} \sum_{i \in I} x_{ij} = 1, \forall j \in I, \tag{2.25}$$

$$x_{ij} \leq y_i, \forall i, j \in I, \tag{2.26}$$

$$\sum_{i \in I} y_i = p, \tag{2.27}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in I, \tag{2.28}$$

$$y_i \in \{0, 1\}, \forall i \in I. \tag{2.29}$$

Nas próximas seções, vamos apresentar ou derivar alguns resultados para o *PLF*. Entende-se neste momento que tais resultados também se aplicam aos casos específicos *PLC*, *PLNC*, *SPLP*. Em alguns pontos do texto, entretanto, podemos particularizar as propriedades para algum desses problemas, quando couber.

## 2.3 Revisão Bibliográfica

O problema de localização de facilidades (*PLF*), mesmo em sua versão mais simples (*SPLP*), é NP-Difícil, como pode ser visto em Garey e Johnson[21]. Além disso, o *PLF* (e particularmente o *SPLP*) relaciona-se a vários outros problemas clássicos como empacotamento, cobertura e particionamento. Estas características são suficientes para mostrar sua larga aplicabilidade e justificar o interesse pelo tema. Por isso mesmo, tem sido amplamente estudado desde o ínicio dos anos 60 (Balinski[4], Graciano Sá[25] e Manne[33]). Na literatura, podemos encontrar uma grande quantidade de algoritmos propostos, tanto de natureza heurística como exata.

A seguir, apresentaremos uma breve revisão destes trabalhos, que não tem a intenção de ser exaustiva ou detalhada. Para um histórico mais completo, sugerimos uma consulta a *surveys* ou livros sobre o tema, como por exemplo Krarup & Pruzan[31], Mateus e

Carvalho[36] e Daskin[17], que compilam trabalhos até meados dos anos 90. Aqui, vamos destacar os trabalhos mais recentes e aqueles com maior relação com o nosso. Apenas para efeito de apresentação, vamos tentar agrupar as referências, conforme o tipo de técnica usada para resolver o problema.

Uma grande parte dos algoritmos propostos para o *PLF* são de natureza heurística. Entre eles, destacamos as conhecidas heurísticas *ADD* e *DROP* propostas inicialmente para o caso não-capacitado, por Kuehn e Hamburger[30] e Feldman, Leher e Ray[19], respectivamente. A heurística *ADD*, inicia o processo com todas as facilidades fechadas e vai abrindo a que for mais econômica, ou seja, que resulte em um máximo decréscimo do custo total. O processo termina quando não for mais possível abrir alguma facilidade que possa melhorar o custo total. Ao contrário, a heurística *DROP* inicia com todas as facilidades abertas e vai fechando aquela que for menos econômica. Tais heurísticas foram generalizadas para o caso capacitado por Jacobsen[28], que também apresentou adaptações nas mesmas, usando formas aproximadas para cálculo das variações nos custos de atendimento ao se abrir ou fechar uma facilidade. Em geral, esses algoritmos estão acoplados a uma segunda fase, onde se procura trocar o satus de alguma facilidade, caso o custo seja reduzido.

Essas duas heurísticas básicas têm sido combinadas em procedimentos mais complexos, como aqueles propostos por Domschke e Drexl[18], Mateus e Bornstein[35] e Campêlo e Bornstein[13], ou incorporadas a algoritmos do tipo *branch-and-bound* para encontrar soluções viáveis e como regra de ramificação (Goldegorin, Ghosh e Sierksma[23]). Mais recentemente, heurísticas que usam a mesma idéia *ADD/DROP* mas critérios de dominância diferentes para abrir ou fechar uma facilidade foram proposta por Pereira[38] e Pinto Jr., Dias e Campêlo[39].

Ainda no campo das heurísiticas, encontramos trabalhos basedos em tabu search (Al-Sultan e Al-Fawzan[2], Michel e Hentenryck[37]), *simulated annealing* (Alves e Almeida[3], Bornstein e Azlan[11]), algoritmos genéticos (Kratica *et al.* [29]), GRASP (Resende e Werneck[40]) e busca em vizinhança ([22]).

É importante destacar também as heurísticas Lagrangianas, como aquelas apresentadas por Beasley[8], que também recapitula os trabalhos nessa linha até o início dos anos 90. Mais recentemente, citamos os trabalhos de Lorena e Senne[32] e Barahona e Chudak[6], que abordam ambos os casos capacitado e não-capacitado. Vale destacar que, neste último trabalho, diferentemente dos outros, que usam o método de subgradientes, os autores aplicam o algoritmo do volume, proposto por Barahona e Anbil[5], para resolver aproximadamente a relaxação lagrangeana. O método do volume é uma extensão do método do subgradiente que tem por objetivo acelerar a convergência e produzir boas soluções primais.

Com respeito a algoritmos exatos, a maioria é do tipo *branch-and-bound*, mas recorrem a diferentes estratégias para definir regras de ramificação e para calcular os limites inferiores e obter soluções viáveis. Em outras palavras, usam diferentes testes de viabilidade/otimalidade para reduzir a árvore de busca. Neste grupo, um dos algoritmos precursores foi proposto por Akinc e Kumawala[1], que utilizam relaxação linear e testes baseados em critérios de dominância para reduzir o espaço de busca. Já Bartezzaghi, Colorni e Palermo[7] sugerem um algoritmo de busca em árvore, onde cada nó representa um subconjunto de facilidades abertas e o limitante inferior corresponde a uma solução dual viável do problema de transporte no nó. Mais recentemente, podemos encontrar os trabalhos de Goldegorin, Ghosh e Serksma[23] e Goldegorin *et at.* [24], que se baseiam em heurísticas *ADD/DROP*, para definir a ramificação, e correção de dados, para cálculo dos limites.

Em outra linha, como representante dos trabalhos que combinam vários estratégias usuais em programação inteira para compor o método de solução, podemos citar Galvão e Raggi[20], que propuseram um algoritmo em três fases para *PLNC*. A primeira fase é um algoritmo primal/dual; a segunda, um procedimento lagrangeano e a última, um processo enumerativo tipo *branch-and-bound*. Uma fase só é executada, caso a anterior não encontre a solução ótima. Este é um dos algoritmos da literatura mais eficientes para resolver o *PLNC* e o *P-MED*.

## 2.4 Testes de Redução

Todos os problemas apresentados na Seção 2.2 são NP - Difíceis. Assim, uma estratégia útil em um algoritmo de solução é procurar reduzir a dimensão do problema, fixando a priori algumas variáveis, particularmente as variáveis $y$.

Os testes de redução, propostos inicialmente por Akinc e Khumawala[1] para o caso capacitado, procuram determinar algumas facilidades que devem estar abertas ou fechadas na solução ótima. Estes testes são descritos a partir de uma função $\Delta_i()$ que estabelece critérios para abertura ou fechamento da facilidade $i$, comparando variações nos custos fixos e de transporte. Tais testes têm sido utilizados para construção de heurísticas e algoritmos que reduzem o tamanho do problema ([11],[13],[23],[28],[35],[38]). A partir destes testes, a solução ótima de algumas instâncias pode ser descoberta de forma iterativa. Sua definição é derivada de outras funções auxiliares, como se segue.

### 2.4.1 Funções Auxiliares

Iremos utilizar as algumas funções relacionadas ao conjunto viável e à função objetivo de *PLF*, quando tomamos um conjunto $K \subseteq I$ e fixamos as variáveis $y_i$ em 1 (se $i \in K$) ou em 0 (se $i \in I - K$). Precisamente, para um subconjunto $K \subseteq I$, sejam:

$$X(K) = \left\{ x \in \mathbb{R}_+^{|K|\times|J|} : \sum_{i\in K} x_{ij} = 1, \forall j \in J; \sum_{j\in J} B_j x_{ij} \leq A_i, i \in K \right\} \tag{2.30}$$

$$F(K) = \sum_{i\in K} f_i \tag{2.31}$$

$$W(K) = \inf \left\{ \sum_{i\in K}\sum_{j\in J} c_{ij} x_{ij} : x \in X(K) \right\} \tag{2.32}$$

$$Z(K) = F(K) + W(K) \tag{2.33}$$

O conjunto $X(K)$ descreve as possíveis formas de atender os consumidores em $J$ a partir das facilidades em $K$, enquanto as funções $F(K)$, $W(K)$ e $Z(K)$ fornecem, respectivamente, o custo fixo, o menor custo de transporte e o custo total mínimo correspondentes.

O ínfimo em (2.32) justifica-se pela possibilidade de acontecer $X(K) = \varnothing$, quando teremos $W(K) = Z(K) = +\infty$. Convencionamos ainda que $X(\varnothing) = \varnothing$ e $F(\varnothing) = 0$.

Com essas definições podemos reescrever *PLF* como:

$$\min \quad Z(K) \tag{2.34}$$

$$\text{sujeito a} \quad : \quad K \subseteq I, |K| \leq p \tag{2.35}$$

Essa formulação para *PLF* ressalta que uma solução é essencialmente definida por quais facilidades são instaladas, isto é, pela atribuição de valores a $y$. A partir da instanciação de $y$, os valores de $x$ são definidos facilmente pela solução do problema de transporte que aparece na definição de $W()$.

Destacamos agora duas importantes propriedades da função $W()$(Ver Wolsey [43]).

**Proposição 1** *A função $W()$ é não-crescente, isto é, $W(K) \leq W(K'), \forall K' \subseteq K$.*

**Proposição 2** *A função $W()$ é supermodular, quer dizer, $W(K) - W(K \cup i) \leq W(K') - W(K' \cup i), \forall K' \subseteq K, \forall i \in I - K$.*

Adicionalmente, para $i \in I - K$, consideramos as funções

$$\delta_i(K) = W(K) - W(K \cup i) \text{ e } \Delta_i(K) = f_i - \delta_i(K), \tag{2.36}$$

como definidas por Bornstein e Azlan ([11]) e Campêlo e Bornstein ([13]). Note que a função $\Delta_i()$ também pode ser definida como

$$\Delta_i(K) = Z(K \cup i) - Z(K) \tag{2.37}$$

Sendo assim, $\Delta_i(K)$ é a variação no custo total quando incluímos $i$ no conjunto $K$ de facilidades abertas ou, equivalentemente, quando retiramos $i$ do conjunto $K \cup i$ de facilidades abertas. Claramente, este valor é dado pelo balanço entre a variação $f_i$ nos custos fixos e a variação $\delta_i(K)$ nos custos de transporte, conforme expressão de $\Delta_i(K)$ em (2.36).

Pelas propriedades da função $W()$, apresentadas nas proposições 1 e 2, temos que:

**Proposição 3** $\delta_i(K) \geq 0, \forall K \subseteq I, \forall i \in I - K$.

**Proposição 4** $\Delta_i(K) \geq \Delta_i(K'), \forall K' \subseteq K, \forall i \in I - K$.

Esta propriedade da função $\Delta_i()$ garante a otimalidade das decisões decorrentes dos testes de redução definidos a seguir.

### 2.4.2 Solução Ótima Parcial

Do que foi dito na subseção anterior, temos que uma solução ótima para o *PLF* pode ser representada pela dupla $(K_0, K_1)$, onde $K_0$ é o conjunto das facilidades fechadas e $K_1$ é o conjunto das facilidades abertas. Além disso, se tal solução ótima é construída de forma iterativa, é útil definir o conjunto $K_2$, como o conjunto de facilidades cujos stati ainda estão indefinidos. Também iremos denotar por $K_0^c$ o conjunto $I - K_0 = K_1 \cup K_2$ das facilidades não-fechadas.

**Definição 5** *Uma solução parcial para PLF é uma dupla* $(K_0, K_1)$*, onde* $K_0 \subseteq I, K_1 \subseteq I, K_0 \cap K_1 = \varnothing$ *e* $|K_1| \leq p$.

**Definição 6** *Uma solução parcial* $(K_0, K_1)$ *é dita ótima parcial para PLF se existe solução ótima* $(K_0^*, K_1^*)$ *tal que* $K_0 \subseteq K_0^*, K_1 \subseteq K_1^*$.

Os testes abaixo mostram como uma solução ótima parcial pode ser melhorada com a abertura ou fechamento de outra facilidade([1],[35] e [13]).

**Proposição 7 (F-Teste)** *Sejam* $(K_0, K_1)$ *uma solução ótima parcial para PLF e* $i \in K_2$. *Se* $\Delta_i(K_1) \geq 0$ *então* $(K_0 \cup i, K_1)$ *é uma solução ótima parcial para PLF.*

**Prova.** Como $(K_0, K_1)$ é uma solução ótima parcial, então existe uma solução ótima $(K_0^*, K_1^*)$ tal que $(K_0, K_1) \subseteq (K_0^*, K_1^*)$. Vamos provar que existe uma outra solução ótima $(K_0^{**}, K_1^{**})$ tal que $(K_0 \cup i, K_1) \subseteq (K_0^{**}, K_1^{**})$. Dividiremos a prova em duas partes:

1. Suponha que $i \in K_0^*$. Já que $K_0 \cup i \subseteq K_0^*$, tomemos $K_0^{**} = K_0^*$ e $K_1^{**} = K_1^*$.

2. Suponha que $i \notin K_0^*$; logo $i \in K_1^*$. Vamos provar que $K_0^{**} = K_0^* \cup i$, $K_1^{**} = K_1^* - i$ também é uma solução ótima. Como $|K_1^* - i| < |K_1^*| \leq p$, resta mostrar que $Z(K_1^*) = Z(K_1^* - i)$. Trivialmente temos que $Z(K_1^*) \leq Z(K_1^* - i)$, pois $(K_0^*, K_1^*)$ é uma solução ótima. A outra desigualdade decorre de $\Delta_i(K_1) \geq 0$ e da Proposição 4. De fato, como $K_1 \subseteq K_1^* - i$ e usando (2.37), segue-se que $Z(K_1^*) - Z(K_1^* - i) = \Delta_i(K_1^* - i) \geq \Delta_i(K_1) \geq 0$, ou ainda, $Z(K_1^*) \geq Z(K_1^* - i)$.

■

Para os problemas onde não há a restrição limitando o número de facilidades a serem instaladas, obtemos um teste similar para abrir facilidades, estabelecido abaixo para o *PLC* e que vale conseqüentemente para o caso particular do *SPLP*.

**Proposição 8 (A-Teste)** *Sejam* $(K_0, K_1)$ *uma solução ótima parcial para PLC e* $i \in K_2$. *Se* $\Delta_i(K_0^c - i) \leq 0$ *então* $(K_0, K_1 \cup i)$ *é uma solução ótima parcial para PLC.*

**Prova.** Similarmente à demonstração da Proposição 7, a partir de uma solução ótima $(K_0^*, K_1^*) \supseteq (K_0, K_1)$, vamos exibir uma solução ótima $(K_0^{**}, K_1^{**})$ tal que $(K_0, K_1 \cup i) \subseteq (K_0^{**}, K_1^{**})$. Como antes, dividiremos a prova em duas partes:

1. Suponha que $i \in K_1^*$. Como $K_1 \cup i \subseteq K_1^*$, então $K_0^{**} = K_0^*$ e $K_1^{**} = K_1^*$.

2. Suponha que $i \notin K_1^*$, ou seja, $i \in K_0^*$. Vamos provar que $K_1^{**} = K_1^* \cup i$, $K_0^{**} = K_0^* - i$ também é uma solução ótima, ou seja, que $Z(K_1^*) = Z(K_1^* \cup i)$. Por um lado, $Z(K_1^*) \leq Z(K_1^* \cup i)$ pois $(K_0^*, K_1^*)$ é uma solução ótima. Por outro, dado que $\Delta_i(K_0^c - i) \leq 0$ e $K_1^* \subseteq K_0^c - i$ a Proposição 4 leva a $Z(K_1^* \cup i) - Z(K_1^*) = \Delta_i(K_1^*) \leq \Delta_i(K_0^c - i) \leq 0$, mostrando que $Z(K_1^* \cup i) \leq Z(K_1^*)$.

■

Note que um processo iterativo pode ser estabelecido com a aplicação destes dois testes da seguinte forma. Inicialmente, consideramos os stati de todas as facilidades como indefinidos, ou seja, $K_0 = K_1 = \varnothing$ e $K_2 = I$. Então, usamos o *A-teste* para

atribuir facilidades a $K_1$. Se ocorrer $X(K_1) \neq \varnothing$ (no caso não-capacitado é suficiente uma facilidade ser aberta), aplicamos o *F-teste* para atribuir facilidades a $K_0$. Caso alguma facilidade seja fechada, o processo é repetido. Assim, a aplicação sucessiva dos testes pode levar à determinação dos statis de algumas facilidades na solução ótima, reduzindo o tamanho do problema.

De acordo com a Proposição 4, temos que $\Delta_i(K_1) \leq \Delta_i(K_0^c - i)$, para todo $i \in K_2$. Além disso, a mesma proposição revela o deslocamento dos valores da função $\Delta_i()$, ao aplicarmos os testes de redução, como ilustrado pela Figura 2.1. Quando abrimos uma facilidade, aumentamos $\Delta_i(K_1)$, e quando fechamos uma facilidade, reduzimos $\Delta_i(K_0^c - i)$. Assim a resposta satisfatória de um dos testes potencializa a aplicação do outro justificando o processo iterativo acima descrito.


Figura 2.1: Deslocamento dos deltas com a aplicação dos testes de redução

Idealmente, tal processo poderia levar a $K_2 = \varnothing$, definindo os stati ótimos de todas as facilidades. Porém, em geral, essa situação raramente ocorre. Normalmente os testes conduzem a uma situação em que $\Delta_i(K_1) < 0 < \Delta_i(K_0^c - i)$, para todo $i \in K_2$.

## 2.5 Instâncias de Testes

Como dissemos, os problemas de localização descritos na Seção 2.2 são NP-Difíceis. Algoritmos propostos para esses problemas são comumente avaliados a partir da resolução de instâncias de teste.

Existem na literatura vários tipos de instâncias de testes para o *Problema de Localização de Facilidades* (*PLF*) e, particularmente, para o *Simple Plant Location Problem* (*SPLP*) que será o problema explorado nos próximos capítulos. Aqui descreveremos dois desses conjuntos de instâncias para o *SPLP*, que serão usados em nossos experimentos computacionais. Esta escolha foi motivada pelas características das instâncias ou por serem as mais usadas na literatura.

O primeiro conjunto de problemas está disponível na *OR-Library*[10], onde existem problemas-testes para várias áreas da pesquisa operacional. Para localização, eles são chamados de cap, pois foram originalmente projetados para o caso capacitado. Elegemos três grupos de instâncias, identificadas na Tabela 2.1, onde x $\in \{1, 2, 3, 4\}$ representa o número do problema no grupo. Estas são instâncias pequenas, com solução ótima conhecida e bastante usadas na literatura.

| Grupo Problema | Facilidades | Demandas |
|---|---|---|
| cap 7x | 16 | 50 |
| cap 9x | 25 | 50 |
| cap 13x | 50 | 50 |

Tabela 2.1: Descrição dos problemas-teste cap

Os dados que definem essas instâncias são originários de Kuehn e Hamburguer[30]. Os 50 centros consumidores representam cidades dos Estados Unidos, sendo que algumas delas também foram escolhidas como possíveis localização de facilidades. Os custos de transporte foram considerados proporcionais às distâncias entre as cidades $i$ e $j$ determinadas pela expressão

$$c_{ij} = 0.0125 d_{oi} + 0.0250 d_{ij}$$

onde

$d_{oi}$ = distância entre a fábrica e o armazém $i$

$d_{ij}$ = distância entre o armazém $i$ e o consumidor $j$.

Já os custos fixos foram escolhidos arbitrariamente conforme Tabela 2.2.

| Número Problema (x) | Custos Fixo |
|---|---|
| 1 | 7500 |
| 2 | 12500 |
| 3 | 17500 |
| 4 | 25000 |

Tabela 2.2: Custos Fixos para o Problema cap

O segundo grupo de instâncias foi obtido a partir de um gerador de problemas não-euclideanos desenvolvido por Pereira[38]. Uma instância do *SPLP* com $I = J$ é dita não-euclideana se os custos de transporte não satisfazem uma das seguintes propriedades:

*i*) $c_{ii} = 0$;

*ii*) $c_{ij} > 0$;

*iii*) $c_{ij} = c_{ji}$;

*iv*) $c_{ij} < c_{ik} + c_{kj}$;

No caso de Pereira, apenas a condição $(iv)$ não é satisfeita. Os problemas gerados são considerados bastante difíceis e para vários deles ainda não se conhece a solução ótima.

Tais problemas são divididos em cinco grupos, de acordo com o intervalo de variação dos custos fixos, havendo em cada grupo sete problemas com diferentes tamanhos. A descrição dos problemas é apresentada na tabela 2.3. Identificamos o problema do grupo x e tamanho y por x = y.

| Grupo | Intervalo dos Custos Fixos | Intervalo dos Custos de Transporte | Tamanhos ($n = m$) |
|---|---|---|---|
| 1 | 1 a 20 | 10 a 1000 | 50, 100, 150, 200, 400 600, 800 |
| 2 | 1 a 500 | | |
| 3 | 300 a 500 | | |
| 4 | 500 a 1000 | | |
| 5 | 1000 a 1020 | | |

Tabela 2.3: Descrição dos problemas-teste

# Capítulo 3

# Problema não Capacitado via Formulação Clássica

Dentre as muitas variações do problema de localização apresentadas no Capítulo 1, o nosso trabalho está voltado para o *Simple Plant Location Problem* (*SPLP*). Neste capítulo, vamos estudar o *SPLP* através de sua formulação clássica (2.19)-(2.22). Começamos propondo uma versão compacta dessa formulação, que trabalha com menos variáveis e restrições. Em seguida, particularizamos para o *SPLP* os testes de redução apresentados na Seção 2.4 para o *PFL*, evidenciando a simplicação nos cálculos. Usamos os critérios de dominância subjacentes a estes testes para definir novas heurísticas *ADD/DROP* e incorporá-las a algoritmos heurísticos. Apresentamos também procedimentos para cálculo de limites inferiores usando a técnica de *correção de dados* em conjunto com estes critérios de dominâncias. Estes são ingredientes utilizados para definir em seguida vários algoritmos tipo *branch-and-bound*. Experimentos computacionais com os procedimentos propostos permeiam as várias seções.

## 3.1 Formulação Clássica Revisitada

Motivados pela formulação proposta por Labbé e Marín[34] para o *SPLP*, que iremos mostrar no Capítulo 4, introduzimos uma modificação na formulação clássica (2.19)-(2.22) para reduzir o tamanho do modelo, usando uma única variável $x_{ij}$ para todas as facilidades $i$ com mesmo custo de transporte até um consumidor $j$.

Neste sentido, além da notação já definida na Seção 2.2, vamos utilizar a seguinte notação adicional. Denotamos por $K_j$ a quantidade de valores diferentes no conjunto $C_j = \{c_{ij} : i \in I\}$, de custos de transporte das várias facilidades para atender $j$, e por $d_{jr}$, $r = 1, \ldots, K_j$, o $r$-ésimo menor valor (desconsiderando igualdades) em $C_j$, ou seja,

$$\begin{aligned} d_{j1} &= \min\{c_{ij} : i \in I\} \text{ e} \\ d_{jr} &= \min\{c_{ij} : i \in I,\, c_{ij} > d_{jr-1}\},\ r = 2, 3, \ldots, K_j. \end{aligned}$$

Além disso, definimos uma nova variável $w_{jr}$, $j \in J$ e $r = 1, \ldots, K_j$, que será igual a 1 se, e somente se, o centro consumidor $j$ for atendido por facilidade com custo de transporte igual a $d_{jr}$, o $r$-ésimo melhor. Assim, o *SPLP* pode ser reformulado como:

$$(SPLPM) \quad \min \sum_{j \in J} \sum_{r=1}^{K_j} d_{jr} w_{jr} + \sum_{i \in I} f_i y_i \tag{3.1}$$

$$\text{sujeito a: } \sum_{r=1}^{K_j} w_{jr} = 1, \forall j \in J, \tag{3.2}$$

$$w_{jr} \leq \sum_{i \in I_{jr}} y_i, \forall j \in J,\ \forall r = 1, \ldots, K_j, \tag{3.3}$$

$$w_{jr} \in \{0, 1\}\ \forall j \in J,\ \forall r = 1, \ldots, K_j, \tag{3.4}$$

$$y_i \in \{0, 1\}\ i \in I. \tag{3.5}$$

onde $I_{jr} = \{i \in I : c_{ij} = d_{jr}\}$.

Mostraremos a seguir que $(SPLPM)$ é uma projeção de (2.19)-(2.22) dada por

$$w_{jr} = \sum_{i \in I_{jr}} x_{ij}.$$

Para isto, note que, para todo $j \in J$, $\{I_{jr} : r = 1, \ldots, K_j\}$ é uma partição de $I$. Denote por V[$P$] o valor ótimo de um problema $P$.

**Proposição 9** *V[SPLP] = V[SPLPM]*

**Prova.** Primeiro vamos provar que V[*SPLP*] $\geq$ V[*SPLPM*]. Para isso, vamos somar as restrições (2.21) em grupos, conforme partição $\{I_{jr}\}$, obtendo

$$\sum_{i \in I_{jr}} x_{ij} \leq \sum_{i \in I_{jr}} y_i \ \forall j \in J, \forall r = 1, \ldots, K_j$$

Então, tomando $w_{jr} = \sum_{i \in I_{jr}} x_{ij}$, em (2.19)-(2.22) chegamos à formulação (3.1)-(3.5), o que mostra a desigualdade enunciada.

Agora vamos provar que V[*SPLPM*] $\geq$ V[*SPLP*]. Seja $(\widehat{w}, \widehat{y})$ uma solução viável para *SPLPM.* Uma solução viável $(\overline{x}, \widehat{y})$ para *SPLP*, de mesmo valor, é construída da seguinte forma. Se $\widehat{w}_{jr} = 0$ então $\overline{x}_{ij} = 0$, $\forall i \in I_{jr}$. Se $\widehat{w}_{jr} = 1$, escolha $\widehat{\imath} \in I_{jr}$ tal que $\widehat{y}_{\widehat{\imath}} = 1$ e faça $\overline{x}_{\widehat{\imath}j} = 1$ e $\overline{x}_{ij} = 0$, $\forall i \in I_{jr} \backslash \widehat{\imath}$.

De fato, dado que $\widehat{w}_{jr} = \sum_{i \in I_{jr}} \overline{x}_{ij}$, temos que

$$\sum_{i \in I} \overline{x}_{ij} = \sum_{r=1}^{K_j} \sum_{i \in I_{jr}} \overline{x}_{ij} = \sum_{r=1}^{K_j} \widehat{w}_{jr} = 1, \forall j \in J.$$

Então, como $\overline{x}_{ij} = 1$ somente se $\widehat{y}_i = 1$ temos

$$\overline{x}_{ij} \leq \widehat{y}_i, \ \forall j \in J, \forall i \in I$$

e

$$\sum_{j \in J} \sum_{i \in I} c_{ij} \overline{x}_{ij} = \sum_{j \in J} \sum_{r=1}^{K_j} d_{jr} \sum_{i \in I_{jr}} \overline{x}_{ij} = \sum_{j \in J} \sum_{r=1}^{K_j} d_{jr} \widehat{w}_{jr}.$$

Logo, $(\overline{x}, \widehat{y})$ é viável para *SPLP* e possui mesmo valor que $(\widehat{w}, \widehat{y})$. O resultado vale particularmente quando $(\widehat{w}, \widehat{y})$ é ótimo para *SPLPM*, levando a desigualdade desejada. ■

Dessa forma provamos que as duas formulações são equivalentes. Além de possuírem o mesmo valor ótimo, a solução ótima de uma pode ser obtida a partir da solução ótima da

outra. Entretanto, a formulação (3.1)-(3.5) apresenta duas vantagens. Primeiro, possui menor dimensão, que será tão mais relevante quanto menor for $\sum_{j \in J} |K_j|$ em relação a $|J|\mathrm{x}|I|$. Segundo, elimina certas soluções equivalentes, uma vez que, uma solução $(w, y)$ é representada por várias soluções $(x, y)$, se $\sum_{i \in I_{jr}} y_i > 1$. Esta característica é particularmente interessante quando se trabalha com o método de planos de corte.

A Tabela 3.1 ilustra os aspectos acima mencionados, a partir da resolução dos problemas não-euclideanos do grupo 2 pelo software de programação matemática XPress. Para formulação clássica, registramos o número de restrições, que para os problemas não-euclideanos são iguais ao número de variáveis, e o tempo gasto, em segundos, pelo XPress. Apresentamos nas últimas colunas, a redução sofrida em cada um destes itens na formulação revisitada.

| | Formulação Clássica | | Formulação Clássica Revisitada | |
|---|---|---|---|---|
| Problema | Restrições | Tempo(s) | Restrições(%) | Tempo(%) |
| 2:50 | 2550 | 0,12 | 2,20 | 8,33 |
| 2:100 | 10100 | 0,98 | 4,46 | 38,78 |
| 2:150 | 22650 | 7,27 | 7,07 | 15,13 |
| 2:200 | 40200 | 25,04 | 9,10 | 46,96 |
| 2:400 | 160400 | 22,59 | 17,63 | 18,33 |
| 2:600 | 360600 | 69,59 | 24,91 | 40,65 |
| 2:800 | 640800 | 1595,82 | 31,36 | 43,96 |

Tabela 3.1: Comparação entre as formulações

## 3.2 Avaliação Iterativa dos Testes de Redução

Uma outra forma de reduzir o tamanho do problema é fixar, a priori, algumas variáveis $y$. Isto pode ser feito através dos testes de redução apresentados na Seção 2.4. Aqui particularizaremos tais testes para o *SPLP*. Na verdade vamos reescrever a função $\Delta_i(K)$

para este caso.

Para os problemas não-capacitados como o SPLP, o conjunto $X(K)$, definido em (2.30), torna-se

$$X\left(K\right)=\left\{x\in\mathbb{R}_{+}^{|K|\times|J|}:\sum_{i\in K}x_{ij}=1,\forall j\in J\right\}.$$

Com isso, a função $W()$ é simplificada para

$$W\left(K\right)=\sum_{j\in J}\min_{k\in K}c_{kj}, \tag{3.6}$$

indicando que cada consumidor é atendido pela melhor facilidade em $K$. Daí, decorre que

$$Z(K)=\sum_{k\in K}f_{k}+\sum_{j\in J}\min_{k\in K}c_{kj}$$

e

$$\Delta_{i}\left(K\right)=Z(K\cup i)-Z(K)=f_{i}+\sum_{j\in J}\min\left\{c_{ij}-\min_{k\in K}c_{kj},0\right\}. \tag{3.7}$$

Esta expressão simplificada de $\Delta_{i}\left(K\right)$ mostra que os cálculos necessários para avaliação dos testes de redução podem ser feitos em tempo polinomial.

Além disso, nos algoritmos iterativos que vamos propor será importante determinar $\Delta_{i}\left(K\cup s\right)$ e $\Delta_{i}\left(K-r\right)$, com facilidade, a partir de $\Delta_{i}\left(K\right)$. Em outros termos, queremos calcular de forma mais econômica $\Delta_{i}\left(K\right)$ quando modificarmos $K$.

Pela equação (3.7), derivamos as proposições abaixo para atualizar os valores de $\Delta_{i}\left(K\right)$, quando abrirmos ou fecharmos alguma facilidade.

**Proposição 10** *Sejam $K\subseteq I$, $\{s,i\}\subset I-K$ e $c_{m_{j}j}=\min_{k\in K}c_{kj}$, para todo $j\in J$. Então*

$$\Delta_{i}\left(K\cup s\right)=\Delta_{i}\left(K\right)+\sum_{j\in J}\max\left\{c_{m_{j}j}-\max\left\{c_{ij},c_{sj}\right\},0\right\}.$$

**Prova.** Por (3.7), temos que $\Delta_{i}\left(K\cup s\right)-\Delta_{i}\left(K\right)=\sum_{j\in J}\left(a_{j}-b_{j}\right)$, onde

$$a_{j}=\min\left\{c_{ij}-\min\left\{c_{m_{j}j},c_{sj}\right\},0\right\}\ \text{ e }\ b_{j}=\min\left\{c_{ij}-c_{m_{j}j},0\right\}.$$

Então, basta mostrar, para todo $j \in J$, que $a_j - b_j = \max\{d_j, 0\}$, com

$$d_j = c_{m_j j} - \max\{c_{ij}, c_{sj}\}.$$

Vamos considerar dois casos. Primeiro, suponha que $d_j \leq 0$. Se $c_{m_j j} \leq c_{sj}$, então $a_j = b_j$. Do contrário, $c_{sj} < c_{m_j j} \leq c_{ij}$ e, por conseguinte, $a_j = b_j = 0$. Então, neste primeiro caso, temos que $a_j - b_j = 0 = \max\{d_j, 0\}$. Agora vamos assumir que $d_j > 0$. Se $c_{ij} < c_{sj} \leq c_{m_j j}$, então $a_j = c_{ij} - c_{sj}$ e $b_j = c_{ij} - c_{m_j j}$, ou ainda, $a_j - b_j = c_{m_j j} - c_{sj} = d_j$. No subcaso complementar, $c_{sj} \leq c_{ij} \leq c_{m_j j}$, levando a $a_j = 0$ e $b_j = c_{ij} - c_{m_j j} = -d_j$. Nas duas situações, temos novamente $a_j - b_j = d_j = \max\{d_j, 0\}$. ∎

**Proposição 11** *Sejam $K \subseteq I$, $r \in K$, $i \in I - K$ e $c_{m_j j} = \min_{k \in K - r} c_{kj}$, para todo $j \in J$. Então*

$$\Delta_i(K - r) = \Delta_i(K) + \sum_{j \in J} \min\left\{\max\{c_{rj}, c_{ij}\} - c_{m_j j}, 0\right\}.$$

**Prova.** Por (3.7), temos que $\Delta_i(K - r) - \Delta_i(K) = \sum_{j \in J} (a_j - b_j)$, onde

$$a_j = \min\left\{c_{ij} - c_{m_j j}, 0\right\} \text{ e } b_j = \min\left\{c_{ij} - \min\left\{c_{m_j j}, c_{rj}\right\}, 0\right\}.$$

Similarmente à Proposição 10, vamos mostrar que, para todo $j \in J$, $a_j - b_j = \min\{d_j, 0\}$, onde agora

$$d_j = \max\{c_{rj}, c_{ij}\} - c_{m_j j}.$$

Primeiro, suponhamos $d_j \geq 0$ e consideramos dois subcasos. Sendo $c_{rj} \geq c_{m_j j}$, trivialmente $a_j = b_j$. Caso contrário, $c_{rj} < c_{m_j j} \leq c_{ij}$, implicando $a_j = b_j = 0$. Nos dois subcasos, $a_j = b_j = 0 = \min\{d_j, 0\}$. Agora, consideramos o caso $d_j < 0$ também em dois subcasos. Quando $c_{ij} < c_{rj} \leq c_{m_j j}$, segue-se que $a_j = c_{ij} - c_{m_j j}$, $b_j = c_{ij} - c_{rj}$ e $a_j - b_j = c_{rj} - c_{m_j j} = d_j$. Sendo $c_{rj} \leq c_{ij} \leq c_{m_j j}$, segue-se $a_j = c_{ij} - c_{m_j j} = d_j$ e $b_j = 0$. De novo, $a_j - b_j = d_j = \min\{d_j, 0\}$. ∎

## 3.3 Heurísticas ADD/DROP

Vários procedimentos que usam de forma heurística as condições dos testes de redução têm sido propostos na literatura (ver, por exemplo, Campelo e Bornstein[13], Jacobsen[28], Mateus e Bornstein[35] e Pereira[38]). Esses procedimentos escolhem de forma gulosa, a cada iteração, uma facilidade para ser aberta ou fechada, sendo, por conseguinte, chamados de heurísticas *ADD/DROP*. Apesar de não garantirem a otimalidade das decisões, determinam em geral soluções próximas à ótima. Além disso, caracterizam-se pela simplicidade de implementação, flexibilidade e baixo esforço computacional.

### 3.3.1 Heurísticas Existentes

As primeiras heurísticas *ADD/DROP* foram desenvolvidas por Kuhen e Hamburger[30] e Feldman, Lehner e Ray[19], para problemas não-capacitados. Depois, Jacoben[28] e Domschke e Drexe[18] estenderam essas heurísticas para o caso capacitado. Vários outros trabalhos se seguiram, apresentando algoritmos que usavam modificações e combinações variadas dessas heurísticas para encontrar boas soluções ou como ferramentas para compor algoritmos exatos ([35],[13],[11]).

Em essência, essas heurísticas apóiam-se diretamente nas condições que definem o *A-Teste* e o *F-Teste*. Tais condições sugerem que as facilidades associadas aos valores extremos de $\Delta_i(K_1)$ e $\Delta_i(K_0^c - i)$ apresentam uma maior tendência para serem abertas ou fechadas. Defina, conforme a Figura 3.1 as facilidades $a, b, c, d$ tais que:

$$\Delta_a(K_1) = \min_{i \in K_2} \Delta_i(K_1),$$

$$\Delta_b(K_1) = \max_{i \in K_2} \Delta_i(K_1),$$

$$\Delta_c(K_0^c - c) = \max_{i \in K_2} \Delta_i(K_0^c - i) \text{ e}$$

$$\Delta_d(K_0^c - d) = \min_{i \in K_2} \Delta_i(K_0^c - i).$$

Consideramos que toda facilidade $i \in K_2$ satisfaz $\Delta_i(K_1) < 0 < \Delta_i(K_0^c - i)$, do contrário já poderia ser transferida para $K_1$ ou $K_0$.

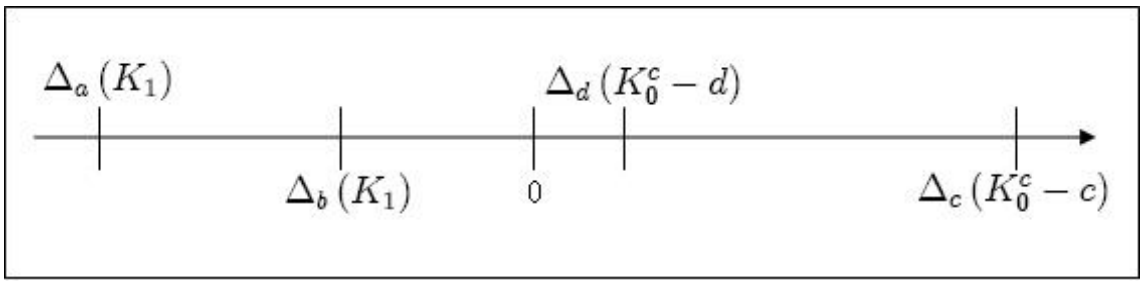


Figura 3.1: Facilidades com maior potencial para abrir ($a$, $d$) e fechar ($b$, $c$)

Pelo *F-Teste*, identificamos as facilidades $a$ e $b$ como as mais propensas para serem aberta e fechada, respectivamente. Similarmente, o *A-Teste* indica as facilidades $c$ e $d$ como tendo grande potencial para serem fechada e aberta, respectivamente.

A partir dessas idéias, vários autores sugeriram heurísticas *ADD/DROP*([13], [28], [30], [35]). Essas heurísticas usavam separadamente as funções $\Delta_i(K_1)$ e $\Delta_i(K_0^c - i)$ para tomar suas decisões. Observe, porém, que uma facilidade pode ser indicada para ser aberta por um desses valores e para ser fechada pelo outro.

Procurando resolver este conflito, uma nova heurística *ADD/DROP* foi proposta recentemente por Pereira[38], que a denominou Heurística de Soma de Deltas (*HSD*). A idéia principal da *HSD* reside no fato de que o módulo da soma de $\Delta_i(K_1)$ e $\Delta_i(K_0^c - i)$ mostra o quão díspares são os valores de $|\Delta_i(K_1)|$ e $|\Delta_i(K_0^c - i)|$, para cada facilidade $i$. Quanto maior a disparidade, ou seja, a assimetria em relação ao 0, menor parece ser o conflito entre a decisão de abrir ou fechar $i$. Assim, a facilidade relativa ao maior valor $|\Delta_i(K_1) + \Delta_i(K_0^c - i)|$, dentre as que estão em $K_2$, terá seu status definido pela *HSD*, conforme o sinal da soma $\Delta_i(K_1) + \Delta_i(K_0^c - i)$. Se for negativa, então abrimos a facilidade $i$; caso contrário, fechamos a facilidade $i$.

### 3.3.2 Heurísticas Propostas

Ainda com base nas condições que definem a abertura ou fechamento de uma facilidade pelos *A-Teste* e *F-Teste* e considerando diferentes critérios para a escolha de qual facilidade

terá seu status fixado, desenvolvemos novas heurísticas *ADD/DROP*. Considere aqui as facilidades $a, b, c, d$ como definidas na Subseção 3.3.1.

**Heurística Diferença de Deltas - HDD / HDDE:** A idéia da heurística é escolher uma entre as facilidades candidatas para ser aberta ou fechada, através do seguinte procedimento. Definimos o *índice de abertura* de uma facilidade $i$ como

$$|\Delta_i(K_0^c - i) - \Delta_d(K_0^c - d)| + |\Delta_i(K_1) - \Delta_a(K_1)|.$$

Similarmente, definimos o *índice de fechamento* de uma facilidade $i$, como

$$|\Delta_i(K_0^c - i) - \Delta_c(K_0^c - c)| + |\Delta_i(K_1) - \Delta_b(K_1)|.$$

A facilidade com o menor índice (de abertura ou fechamento) será a escolhida (para ser aberta ou fechada, respectivamente). Desta forma, a facilidade escolhida atende melhor aos dois critérios simultaneamente. Quando apenas as facilidades $a, b, c, d$ são consideradas candidatas, chamamos esta heurística de Diferença de Deltas(*HDD*). Se todas as facilidades em $K_2$ são candidatas, chamamo-na de Diferença de Deltas Estendida (*HDDE*).

**Heurística Melhor Decisão Relativa - HMDR:** Seja $a_2$ a segunda facilidade mais propensa para ser aberta segundo o *F-Teste*, ou seja, tal que $\Delta_{a_2}(K_1) = \min_{i \in K_2 - a} \Delta_i(K_1)$. Similarmente, considere as facilidades $b_2, c_2, d_2$, como as segundas candidatas, depois de $b$, $c$ e $d$, respectivamente. Definimos o *coeficiente relativo* da facilidade $i$ como $|\Delta_i(K) - \Delta_{i_2}(K)|$, onde $K = K_1$ para $i = a, b$ ou $K = K_0^c - i$ para $i = c, d$. A facilidade $i$ com maior coeficiente relativo será escolhida para ser aberta (se $a$ ou $c$) ou fechada (se $b$ ou $d$). Esta é a estratégia da Heurística Melhor Decisão Relativa.

**Heurística Região de Aceitação - HRA:** A heurística Região de Aceitação combina a idéia da Heurística Soma de Delta (*HSD*) com a observância dos valores extremos das funções $\Delta_i()$. Pereira[38] reporta que *HSD* supõe um deslocamento equilibrado dos referenciais $\Delta_i()$, $i \in K_2$ (Figura 2.1*)*. Então consideramos a distribuição desses valores

nos segmentos $(0, -\Delta_a(K_1)]$ e $(0, \Delta_c(K_0^c - c)]$, que definem as respectivas amplitudes. Precisamente, seja $p$ a facilidade indicada pela *HSD*. A decisão recomendada pela *HSD* será efetivada se $p$ estiver na região de aceitação, o que ocorre quando:

*i)* $\Delta_p(K_0^c - p) \in (0, \Delta_c(K_0^c - c)/|K_2|]$, se $p$ foi indicada para ser aberta;

*ii)* $-\Delta_p(K_1) \in (0, -\Delta_a(K_1)/|K_2|]$, se $p$ foi indicada para ser fechada.

Caso a decisão da *HSD* não seja efetivada devido ao caso $(i)$, optamos por fechar $c$, recorrendo assim ao critério guloso da heurística proposta por Jacobsen[28]. Nesse caso, escolhemos fechar uma facilidade, quando rejeitamos a decisão de abrir dada pela *HSD*, pois é sabido que, removendo facilidades de $K_0^c$, os valores dos $\Delta_i(K_0^c - i)$ descrescem, o que potencializaria a facilidade $p$ a entrar na região de aceitação. Caso a decisão da *HSD* seja negada pelo caso $(ii)$, decidimos então abrir $a$, por razões semelhantes a do caso anterior.

### 3.3.3 Algoritmos Heurísticos

Combinando cada uma das heurísticas (*HDD*, *HDDE*, *HMDR*, *HRA*) com os testes de redução, definimos quatro algoritmos para encontrar boas soluções para o *SPLP*. A estrutura de cada algoritmo heurístico está ilustrada na Figura 3.2. As setas mostram as iterações entre os módulos, enquanto os valores a elas associados representam a quantidade de facilidades cujo stati podem ser definidos em cada transição.

Basicamente, o algoritmo começa aplicando iterativamente o *A-Teste* e o *F-Teste* até que nenhuma facilidade seja aberta ou fechada. Em seguida, aplica-se uma das heurísticas, que deverá fechar ou abrir uma facilidade. No primeiro caso, retorna-se ao *A-Teste* e, no segundo, ao *F-Teste.* O processo é então repetido. Como a cada iteração testes-heurística o status de pelo menos uma facilidade é definido, o processo converge para uma solução viável.

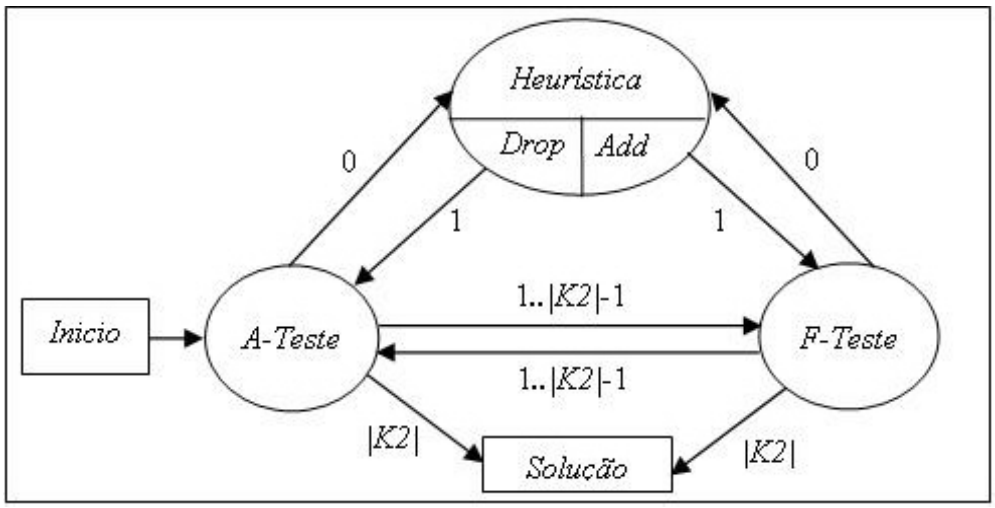


Figura 3.2: Estrutura do algoritmo heurístico

### 3.3.4 Resultados Computacionais

Para avaliarmos o desempenho desses algoritmos, fizemos uma implementação e a submetemos a vários testes computacionais com problemas não-euclideanos. Esses problemas mostram-se mais difíceis de serem resolvidos que os euclideanos. Em seguida, fizemos uma comparação com o algoritmo de Pereira[38], que detém excelente desempenho para este tipo de problema. Os resultados estão apresentados na Tabela 3.2, onde cada algoritmo está identificado pela heurística que o define. Para cada um deles, são apresentados o valor da solução viável encontrada e o tempo gasto em segundos. A implementação foi feita na linguagem C e executada em um Atlon XP 2000+.

A qualidade das soluções encontradas pelos vários algoritmos é bastante similar. Entretanto, pode-se verificar que o algoritmo usando *HRA* encontra com maior freqüência as melhores soluções, o que ocorreu em 10 problemas. Com relação ao esforço computacional, cada um dos novos algoritmos consegue reduzir pela metade o tempo médio para resolver as instâncias testadas. Esse bom desempenho é ainda mais evidente, e em particular no *HMDR*, para os problemas dos grupos 3, 4 e 5, onde os custos fixos são maiores. No primeiro grupo, onde os custos fixos são muito reduzidos, o algoritmo original foi supe-

| | HSD | | HDD | | HDDE | | HMDR | | HRA | |
|---|---|---|---|---|---|---|---|---|---|---|
| Prob | Sol | T | Sol | T | Sol | T | Sol | T | Sol | T |
| 1:200 | **1860** | 0.01 | **1860** | 0.34 | 1866 | 0.20 | 1863 | 0.33 | **1860** | 0.34 |
| 1:400 | 3505 | 0.06 | **3504** | 3.19 | 3518 | 2.20 | **3504** | 3.59 | **3504** | 3.14 |
| 1:600 | **5055** | 0.11 | 5056 | 11.03 | 5060 | 7.42 | 5056 | 10.92 | **5055** | 10.92 |
| 1:800 | **6621** | 0.21 | 6623 | 26.26 | 6628 | 17.95 | 6623 | 25.77 | **6621** | 25.81 |
| 2:200 | 6516 | 0.42 | 6522 | 0.38 | 6576 | 0.19 | 6511 | 0.26 | **6495** | 0.36 |
| 2:400 | 10039 | 3.28 | 10033 | 2.66 | 10086 | 1.80 | 10044 | 2.44 | **10025** | 2.92 |
| 2:600 | 12427 | 7.86 | 12433 | 10.55 | 12484 | 6.30 | 12428 | 12.08 | **12426** | 9.78 |
| 2:800 | **15632** | 43.76 | 15633 | 23.64 | 15706 | 15.14 | 15642 | 23.84 | 15646 | 23.61 |
| 3:200 | **13073** | 0.36 | 13289 | 0.42 | 13377 | 0.17 | 13221 | 0.30 | 13258 | 0.38 |
| 3:400 | 22328 | 4.64 | 22059 | 3.16 | 22140 | 1.81 | 21866 | 2.53 | **21765** | 3.39 |
| 3:600 | 27762 | 16.54 | **27719** | 8.97 | 27924 | 6.22 | 28059 | 9.05 | 27817 | 10.64 |
| 3:800 | 33757 | 52.43 | 33982 | 20.53 | 34124 | 14.88 | 33970 | 20.44 | **33470** | 23.81 |
| 4:200 | **16111** | 0.35 | 16340 | 0.28 | 16515 | 0.17 | 16306 | 0.25 | 16221 | 0.36 |
| 4:400 | 26821 | 4.46 | 27073 | 2.92 | 26745 | 1.78 | **26533** | 2.45 | 26833 | 3.13 |
| 4:600 | 34021 | 26.09 | 33870 | 9.27 | 34126 | 6.20 | 34305 | 8.47 | **33818** | 10.31 |
| 4:800 | 40840 | 67.92 | 40716 | 21.03 | 41602 | 14.89 | **40704** | 20.13 | 41026 | 22.34 |
| 5:200 | 21995 | 0.45 | **21048** | 0.31 | 22242 | 0.19 | 22022 | 0.27 | 22035 | 0.33 |
| 5:400 | 34023 | 4.73 | **33397** | 2.95 | 34221 | 1.78 | 34260 | 2.77 | 33535 | 2.52 |
| 5:600 | **43177** | 25.29 | **43177** | 9.28 | 43823 | 6.19 | 43369 | 10.66 | 43789 | 10.03 |
| 5:800 | 51327 | 67.29 | 51137 | 22.50 | **51010** | 14.88 | 51056 | 18.83 | 51742 | 22.05 |

Tabela 3.2: Resultados Computacionais com problemas não-euclideanos de grande porte

rior. A justificativa para a superioridade em termos de tempo está na forma iterativa de calcular $\Delta_i()$ proposta na Seção 3.2.

Vale mencionar ainda que os problemas cap, que tem dimensão pequena, são fácilmente resolvidos, e de forma exata, por todas as heurísticas.

## 3.4 Limites Inferiores via Correção de Dados

Com a finalidade de desenvolver um algoritmo exato do tipo *branch-and-bound,* baseado essencialmente nas funções $\Delta_i()$, derivamos, nesta seção, limites inferiores a partir das aplicações do *A-Teste* e *F-Teste* e da observação de que esses testes podem gerar uma solução ótima do problema ou fornecem uma solução ótima de um problema modificado, gerado a partir do problema original. Essa é a idéia da técnica *correção de dados* usada em [23].

A técnica de *correção de dados* consiste em gerar um problema de fácil resolução, para o qual conhecemos uma solução ótima, a partir do problema original que desejamos resolver, pela modificação ou correção dos seus dados, de tal forma que a solução ótima do problema gerado forneça um limite inferior para o problema original.

Para o problema de localização, essa correção pode ser feita modificando os custos fixos ou os custos de transporte ou ambos, conforme desenvolvimento genérico abaixo.

Seja $(F, C)$ uma instância de *SPLP*, onde $F = [f_i]$ é o vetor dos custos fixos e $C = [c_{ij}]$ é a matriz dos custos de transporte, para a qual não se conhece a solução ótima $(K_0^*, K_1^*)$. Considere ainda uma instância modificada $(F', C')$, onde $\pi_i = f_i - f'_i$ e $\theta_{ij} = c_{ij} - c'_{ij}$, cujo valor ótimo é conhecido.

Considerando $V[F, C]$ o valor ótimo de uma instância qualquer $(F, C)$, temos:

$$\begin{aligned} V[F,C] &= \sum_{j\in J} \min_{i\in K_1^*} c_{ij} + \sum_{i\in K_1^*} f_i \\ &= \sum_{j\in J} \min_{i\in K_1^*} c'_{ij} + \sum_{i\in K_1^*} f'_i + \sum_{j\in J} \left[ \min_{i\in K_1^*} c_{ij} - \min_{i\in K_1^*} c'_{ij} \right] + \sum_{i\in K_1^*} \pi_i \end{aligned}$$

Então, com os dois primeiros somatórios fornecem o valor da solução viável $(K_0^*, K_1^*)$ para $(F', C')$, segue-se que

$$V[F,C] \geq V[F',C'] + \sum_{j\in J}\left[\min_{i\in K_1^*} c_{ij} - \min_{i\in K_1^*} c'_{ij}\right] + \sum_{i\in K_1^*}\pi_i \tag{3.8}$$

Consideremos agora $K_1^< \subseteq K_1^*$ e $K_1^> \supseteq K_1^*$. Por exemplo, conhecendo uma solução ótima parcial $(K_0, K_1)$ para $(F, C)$, podemos tomar $K_1^< = K_1$ e $K_1^> = I - K_0 = K_0^c$. Definindo ainda $I^+ = \{i \in I : \pi_i > 0\}$ e $I^- = \{i \in I : \pi_i < 0\}$, obtemos o seguinte limite inferior:

$$V[F,C] \geq V[F',C'] + \sum_{j\in J}\left[\min_{i\in K_1^>} c_{ij} - \min_{i\in K_1^<} c'_{ij}\right] + \sum_{i\in K_1^<\cap I^+}\pi_i + \sum_{i\in K_1^>\cap I^-}\pi_i. \tag{3.9}$$

Outro limite inferior pode ser obtido a partir de (3.8) juntamente com o fato de que

$$\min_{i\in K_1^*}\{c_{ij}\} = \min_{i\in K_1^*}\left\{c'_{ij} + \theta_{ij}\right\} \geq \min_{i\in K_1^*}\left\{c'_{ij}\right\} + \min_{i\in K_1^*}\{\theta_{ij}\}.$$

Com isso, chegamos a

$$\begin{aligned} V[F,C] &\geq V[F',C'] + \sum_{j\in J}\min_{i\in K_1^*}\theta_{ij} + \sum_{i\in K_1^*}\pi_i \\ &\geq V[F',C'] + \sum_{j\in J}\min_{i\in K_1^>}\theta_{ij} + \sum_{i\in K_1^<\cap I^+}\pi_i + \sum_{i\in K_1^>\cap I^-}\pi_i. \end{aligned} \tag{3.10}$$

Iremos ilustrar agora como a instância $(F', C')$ pode ser gerada, alterando apenas os custos fixos (ou seja, mantendo $C' = C$), a partir da aplicação dos testes de redução. O desenvolvimento para os outros casos é similar. Relembre que a instância $(F', C')$ deve ser tal que $V[F', C']$ é conhecido.

Considere uma solução ótima parcial $(K_0, K_1)$, tal que $\Delta_i(K_1) < 0 < \Delta_i(K_0^c - i)$ $\forall i \in K_2$. Quando obtemos $\Delta_i(K_0^c - i) > 0$ para uma facilidade $i \in I$ qualquer, podemos gerar um novo custo fixo $f'_i = f_i - \Delta_i(K_0^c - i)$, para que o novo valor de $\Delta_i(K_0^c - i)$ seja igual a 0. Assim, pelo *A-Teste*, essa facilidade $i$ estará aberta na solução ótima desse novo problema. Similarmente, quando obtemos $\Delta_i(K_1) < 0$ para uma facilidade $i \in I$ qualquer, podemos gerar um novo custo fixo $f'_i = f_i - \Delta_i(K_1)$, para que o novo valor de

$\Delta_i(K_1)$ seja igual a 0. Então, o *F-Teste* garante que $i$ estará fechada na solução ótima desse novo problema.

Desta forma, usando a estratégia acima acoplada a qualquer um dos algoritmos heurísticos propostos na Seção 3.3.2, obtemos uma solução ótima para o problema modificado $(F', C)$. Uma ilustração desse procedimento pode ser vista na Figura 3.3.

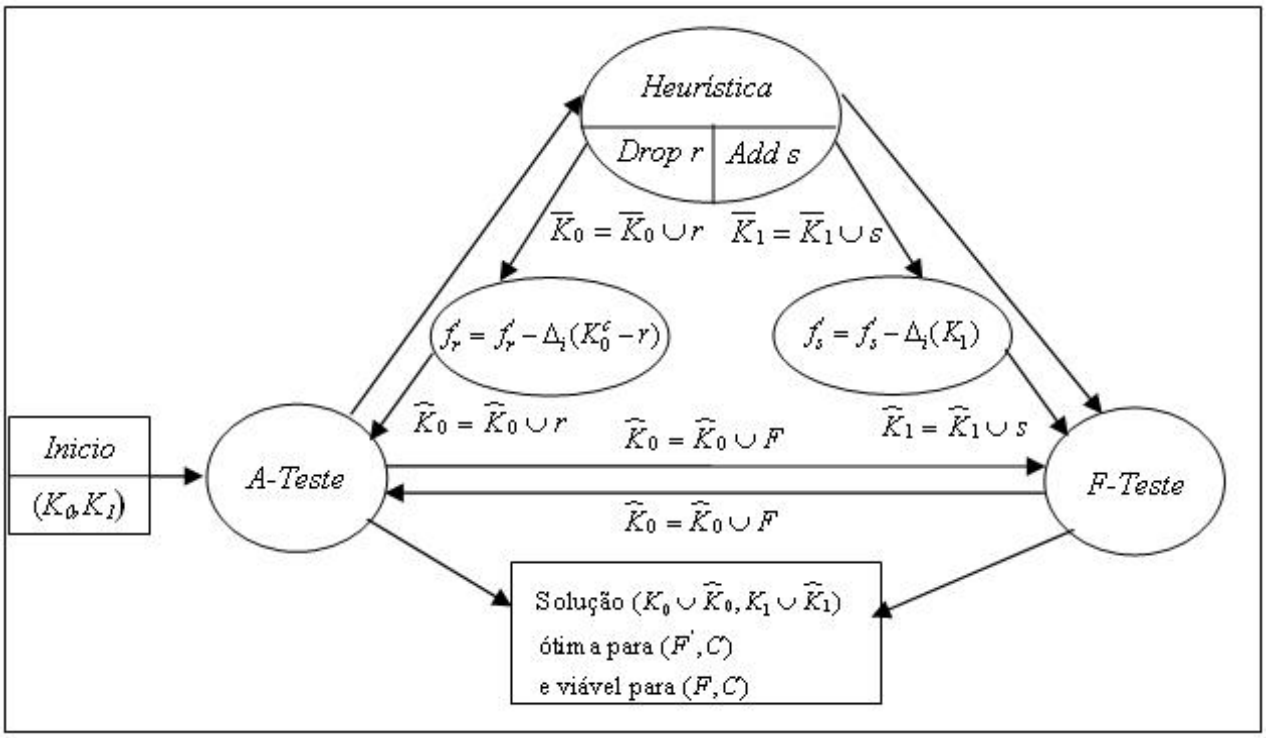


Figura 3.3: Estrutura do Procedimento de Correção de Dados

Chamemos $\widehat{K}_1/\widehat{K}_0$ as facilidades abertas/fechadas pelo algoritmo e de $\overline{K}_1 \subseteq \widehat{K}_1$ e $\overline{K}_0 \subseteq \widehat{K}_0$ aquelas facilidades abertas/fechadas através da heurística (que tiveram seu custo fixo modificado). Então, fazendo $K_1^> = K_0^c$, $K_1^< = K_1$, $I^- = K_0$ em (3.10), temos

$K_1^< \cap I^+ = \varnothing$ e $K_1^> \cap I^- = \overline{K}_0$ levando a:

$$\begin{aligned} V[F,C] &\geq V[F',C] + \sum_{i\in\overline{K}_0}(f_i - f_i') && (3.11)\\ &= \sum_{i\in K_1\cup\widehat{K}_1} f_i' + \sum_{j\in J}\min_{i\in K_1\cup\widehat{K}_1} c_{ij} + \sum_{i\in\overline{K}_0}(f_i - f_i') \\ &= \sum_{i\in K_1\cup\widehat{K}_1} f_i + \sum_{j\in J}\min_{i\in K_1\cup\widehat{K}_1} c_{ij} + \sum_{i\in\overline{K}_1}(f_i' - f_i) + \sum_{i\in\overline{K}_0}(f_i - f_i') \\ &= Z\left(K_1\cup\widehat{K}_1\right) + \sum_{i\in\overline{K}_1}(f_i' - f_i) + \sum_{i\in\overline{K}_0}(f_i - f_i') && (3.12) \end{aligned}$$

Por (3.11), o limite inferior é obtido pelo valor ótimo do problema modificado decrescido dos aumentos nos custos fixos das facilidades ou, equivalentemente, segundo (3.12), pelo valor da solução heurística somado às variações negativas sofridas pelos custos fixos. A expressão (3.12) do limite inferior fornece também uma estimativa do gap de dualidade dos algoritmos heurísticos dados por

$$Z\left(K_1\cup\widehat{K}_1\right) - V[F,C] \leq \sum_{i\in\overline{K}_1}(f_i' - f_i) + \sum_{i\in\overline{K}_0}(f_i - f_i').$$

Pelo exposto, podemos obter limites inferiores a partir dos testes de redução, modificando o problema original, tanto nos custos fixos como nos custos variáveis. Entretanto, em nossos experimentos computacionais, os limites inferiores obtidos pela modificação dos custos variáveis e pela modificação de ambos os custos não mostraram melhores do que os limites gerados pela modificação dos custos fixos apenas.

## 3.5 Branch-and-Bound com Testes de Redução

### 3.5.1 Estrutura Geral

O primeiro algoritmo exato do tipo *branch-and-bound* que propomos baseia-se nos mesmos critérios de dominância entre os custos fixos e variáveis que definem os testes de redução. Usamos estes critérios para guiar a busca na árvore de enumeração, incorporando-

os à estrutura do algoritmo proposto por Goldengorin[23]. Os elementos que definem nosso algoritmo são descritos a seguir.

**Limite Superior:** O limite superior inicial é dado pela solução de um algoritmo heurístico. Em cada nó onde os stati de todas as facilidades são fixados, o limite superior pode ser atualizado, se o valor da solução viável encontrada for melhor.

**Pré-Processamento:** Em cada nó, aplicamos os testes de redução de forma iterativa, tentando fixar os stati de algumas facilidades, até que não sejam mais satisfeitas as condições que garantem a otimalidade das decisões.

**Ramificação:** A ramificação é feita sobre o domínio de uma variável $y_i \in \{0, 1\}$, que é escolhida utilizando uma das heurísticas citadas, ou seja, *HSD*, *HDD*, *HDDE, HMDR, HRA*. Obtemos, assim, cinco regras de ramificação. Dos dois subproblemas gerados, o primeiro a ser avaliado relaciona-se à decisão indicada pela heurística. A idéia é que, se primeiro escolhemos o subproblema favorável à decisão da heurística, então chegaremos mais rapidamente a limites superiores melhores. Com isso iremos podar mais cedo a árvore. Experimentalmente, verificamos que a aplicação dessa ordem de fato gerou árvores com menos subproblemas.

**Limite Inferior:** Utilizamos a idéia de correção de dados, descrita na seção anterior, acoplado ao algoritmo heurístico (Ver Figura 3.2). A partir dos conjuntos $K_0$ e $K_1$ de facilidades já fixadas, procuramos definir outras facilidades pelos testes de redução, utilizamos uma heurística para escolher uma facilidade quando os testes não podem mais ser aplicados, modificamos os custos fixos e retornamos aos testes. Com isso temos vários limites inferiores, um para cada heurística. Os resultados computacionais mostraram que o limite inferior utilizando a heurística *HDDE* é melhor do que os demais. Esse resultado parece ir de encontro à constatação de que *HDDE* mostrou-se ser a heurística que retorna as piores soluções. Entretanto, ele pode ser explicado pelo fato de que *HDDE* define

menos facilidades de forma heurística (e mais pelos testes de redução) que as demais.

O procedimento aplicado ao nó corrente é ilustrado na Figura 3.4, onde são descritas as iterações através de setas, às quais está associado o número de facilidades fixadas em cada transição.

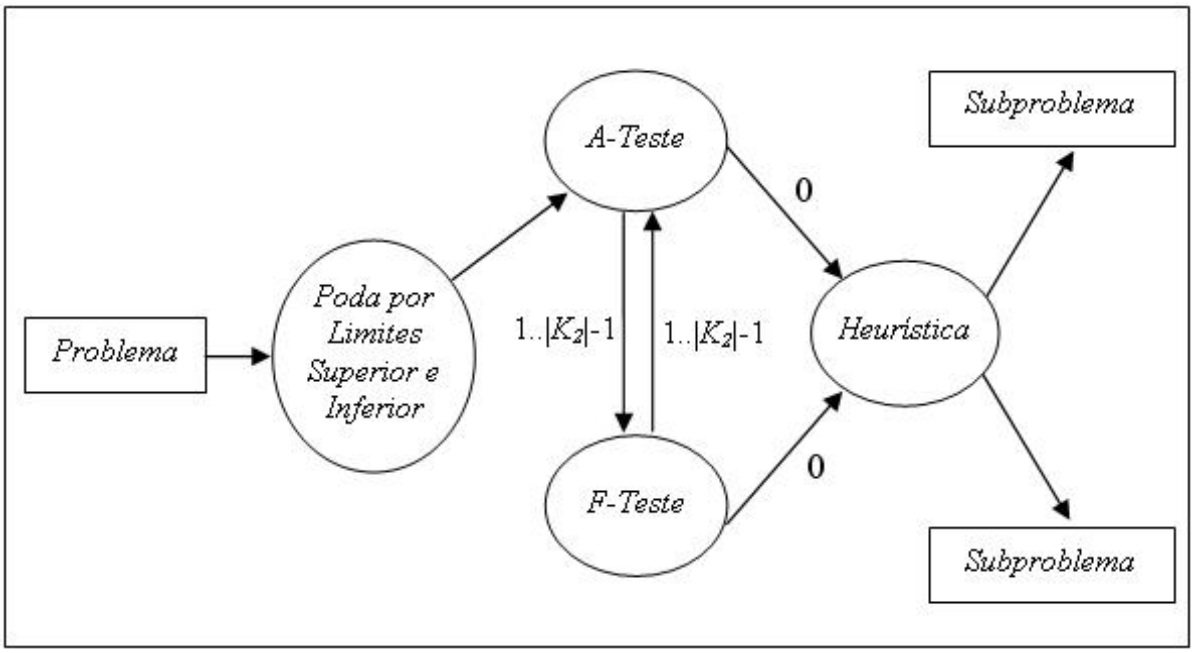


Figura 3.4: Estrutura do algoritmo *branch-and-bound* em cada nó

### 3.5.2 Resultados Computacionais

Para cada regra de ramificação definida, geramos um algoritmo diferente, que será identificado pela sigla da heurística associada. Os algoritmos propostos foram implementados na linguagem C e testados, preliminarmente, com a mesma bateria de problemas testes usados por Goldengorin[23], quais sejam, as instâncias cap. A Tabela 3.3 apresenta uma comparação dos resultados obtidos em termos do número de subproblemas gerados.

Quando comparados com o algoritmo de Goldengorin[23], os novos algoritmos apresentaram resultados significativamente melhores nos seguintes casos. No grupo de problemas

71-74, todos eles obtiveram as soluções exatas gerando um número menor de subproblemas. O mesmo aconteceu com todos os problemas grupo 101-104, para os algoritmos baseados em *HSD, HDD*, *HDDE* e HRA, e para dois desses quatro problemas, quando consideramos *HMDR*. Finalmente, o algoritmo que usa a heurística *HDDE* foi notadamente melhor que todos os outros, quando aplicado aos problemas do grupo 131-134. Na verdade, em todos os problemas testados, podemos observar a superioridade desse algoritmo com relação aos demais. Vale mencionar, porém, que a heurística *HDDE*, quando usada para compor um algoritmo heurístico, não obteve, em geral, melhores soluções viáveis, como podemos constatar na Tabela 3.2.

| | Número de Subproblemas | | | | | |
|---|---|---|---|---|---|---|
| Problemas | HSD | HDD | HRA | HMDR | HDDE | Goldengorin |
| cap71 | **2** | **2** | 3 | 6 | **2** | 19 |
| cap72 | **4** | **4** | **4** | 9 | **4** | 21 |
| cap73 | 6 | 6 | 5 | 12 | **3** | 63 |
| cap74 | **1** | **1** | **1** | **1** | **1** | 11 |
| cap101 | **5** | 9 | 8 | 15 | 6 | 92 |
| cap102 | 21 | **17** | 20 | 174 | 10 | 138 |
| cap103 | **13** | 14 | 35 | 37 | **13** | 71 |
| cap104 | 5 | 4 | 14 | 42 | **3** | 38 |
| cap131 | 3050 | 7115 | 3171 | 15677 | **509** | 2167 |
| cap132 | 1369 | 545 | 318 | 5561 | **161** | 1226 |
| cap133 | 167 | 165 | 283 | 575 | **78** | 503 |
| cap134 | 20 | 29 | 76 | 38 | **11** | 125 |

Tabela 3.3: Resultados obtidos pelos algoritmos Branch-and-Bound aplicados aos problemas da OR-Library

Finalmente, procuramos investigar o comportamento das regras de particionamento quando aplicadas aos problemas não-euclideanos gerados por Pereira[38]. As instâncias de

tamanho 50x50 de todos os grupos foram resolvidas exatamente por todos os algoritmos, assim como as instâncias de tamanho 100x100, 150x150, 200x200, 400x400 do grupo 1 e de tamanho 100x100 e 150x150 do grupo 2 como podemos obsevar na Tabela 3.4.

Já considerando as instâncias de dimensão 100x100 ou maior dos outros grupos, um tempo de 2,5 horas foi insuficiente para que os algoritmos comprovassem ter encontrado a solução ótima, mesmo já a tendo determinado em alguns casos. O limite inferior utilizado não foi capaz de definir uma poda eficiente, e, por conseguinte, de terminar o processo enumerativo em um tempo aceitável.

| Problema | Números de Subproblemas | Tempo |
|---|---|---|
| 1:50 | 1 | 0.03 |
| 1:100 | 1 | 0.11 |
| 1:150 | 6 | 0.39 |
| 1:200 | 39 | 1.17 |
| 1:400 | 1316 | 84.80 |
| 2:50 | 124 | 0.16 |
| 2:100 | 12290 | 40.52 |
| 2:150 | 46316 | 419.23 |
| 3:50 | 16104 | 14.06 |
| 4:50 | 71995 | 58.75 |
| 5:50 | 135385 | 107.13 |

Tabela 3.4: Resultados obtidos pelos algoritmos *branch-and-bound* aplicados aos problemas não-euclideanos

# Capítulo 4

# Problema não Capacitado via Formulação Canônica

Neste capítulo, apresentamos uma reformulação do *SPLP* segundo um modelo de cobertura de conjuntos. Estudamos a corretude da formulação e definimos um subproblema paramétrico, que será usado em procedimentos iterativos para cálculo de limites inferiores e superiores. Propomos, implementamos e testamos a eficiência computacional de vários desses procedimentos. Depois os usamos para definir algoritmos tipo *branch-and-bound*. Os resultados computacionais são mostrados ao longo das seções.

## 4.1 Descrição da Formulação

Uma nova formulação de programação inteira para o *Simple Plant Location Problem* (*SPLP*) foi proposta recentemente por Labbé e Marín[34], gerando um modelo de cobertura de conjuntos. Os autores apresentam essa nova formulação usando a descrição do problema sobre um grafo completo, com conjunto de vértices $I = J$. A seguir adaptamos a formulação proposta por Labbé e Marín[34] à nossa descrição do *SPLP*.

Vamos usar a mesma notação introduzida na Seção 2.2 e estendida na Seção 3.1. De novo usamos a váriavel binária $y_i$, $i \in I$, que será igual 1 se, e somente se, a facilidade $i$

for aberta. Adicionalmente, definimos $z_{jr} \in \{0,1\}$, $j \in J$ e $r = 2, \ldots, K_j$, que será igual a 1 se, e somente se, toda facilidade com custo de transporte para $j$ menor que o $r$-ésimo menor estiver fechada. Formalmente, para $j \in J$ e $r = 2, \ldots, K_j$, temos

$$z_{jr} = 1 \Leftrightarrow y_i = 0, \forall i : c_{ij} < d_{jr} \Leftrightarrow \sum_{i:c_{ij}<d_{jr}} y_i = 0. \tag{4.1}$$

Desta relação entre $y$ e $z$, obtemos as seguintes expressões equivalentes para determinar o custo de atendimento de uma demanda.

**Proposição 12** *Se $y$ e $z$ satisfazem (4.1) e $y \neq 0$ então*

$$\min\{c_{ij} : i \in I, y_i = 1\} = d_{j1} + \sum_{r=2,\ldots,K_j} (d_{jr} - d_{jr-1}) z_{jr},$$

*para todo $j \in J$.*

**Prova.** Seja $j \in J$. Como $y \neq 0$, o mínimo à esquerda da igualdade está bem definido. Suponha que ele ocorra em $c_{i^*j}$ e que $c_{i^*j} = d_{jr^*}$, para determinados $i^* \in I$ e $r^* \in \{1, 2, ..., K_j\}$. Então, $y_{i^*} = 1$ e $y_i = 0$, para todo $i \in I$ tal que $c_{ij} < c_{i^*j} = d_{jr^*}$. Consequentemente, por (4.1), $z_{jr} = 1$, para todo $r = 2, 3, ..., r^*$, e $z_{jr} = 0$, para todo $r = r^* + 1, ..., K_j$. Logo,

$$c_{i^*j} = d_{jr^*} = d_{j1} + \sum_{r=2,\ldots,r^*} (d_{jr} - d_{jr-1}) = d_{j1} + \sum_{r=2,\ldots,K_j} (d_{jr} - d_{jr-1}) z_{jr}.$$

■

A Proposição 12 fundamenta a formulação apresentada por Labbé e Marin[34], que pode ser escrita como:

$$(PC) \qquad \min \sum_{i \in I} f_i y_i + \sum_{j \in J} \sum_{r=2..K_j} (d_{jr} - d_{jr-1})\, z_{jr} + \sum_{j \in J} d_{j1} \tag{4.2}$$

$$\text{sujeito a: } z_{jr} + \sum_{i:\ c_{ij}<d_{jr}} y_i \geq 1, \forall j \in J, \forall r = 2, \ldots, K_j, \tag{4.3}$$

$$y_i \in \{0,1\}, \forall i \in I, \tag{4.4}$$

$$z_{jr} \in \{0,1\}, \forall j \in J, \forall r = 2, \ldots, K_j. \tag{4.5}$$

Note que a implicação inversa em (4.1) é diretamente garantida pelas restrições (4.3) e (4.5). Por outro lado, se a última expressão em (4.1) não é verificada, o melhor valor para $z_{jr}$ é zero, tendo em vista $d_{jr} - d_{jr-1} > 0$ e o problema é de minimização. Assim, pela Proposição 12, vemos que a função objetivo (4.2) é igual a (2.19), fornecendo o custo total de uma solução onde $y \neq 0$. Destacamos, porém, que a condição $y \neq 0$, que assegura a boa definição do mínimo na Proposição 12, pode não ser verificada em uma solução ótima de (4.2)-(4.5). De fato, isto ocorre quando $\min_{i \in I} f_i > \sum_{j \in J} d_{jK_j}$. Neste caso, a solução ótima é $y = 0$ e $z = 1$, cujo valor é $\sum_{j \in J} d_{jK_j}$. Pelo exposto, a formulação apresentada por Labbé e Marin[34] não está completamente correta. Para obtermos sua corretude, acrescentamos ainda a restrição

$$\sum_{i \in I} y_i \geq 1. \tag{4.6}$$

Para exemplificar a formulação, considere uma instância dada por $I = \{1, 2, 3\}$, $J = \{a, b\}$ e custos de transporte $c_{1a} < c_{3a} < c_{2a}$, $c_{3b} = c_{2b} \leq c_{1b}$. Dessa forma, temos

$$\min f_1 y_1 + f_2 y_2 + f_3 y_3 + (c_{3a} - c_{1a})\, z_{a2} + (c_{2a} - c_{3a})\, z_{a3} + (c_{1b} - c_{2b})\, z_{b2}$$

sujeito a: $z_{a2} + y_1 \geq 1$,

$z_{a3} + y_1 + y_3 \geq 1$,

$z_{b2} + y_3 + y_2 \geq 1$,

$y_1 + y_2 + y_3 \geq 1$,

$y_i \in \{0, 1\}, \forall i \in I$,

$z_{jr} \in \{0, 1\}, \forall j \in J, \forall r = 2, 3.$

Vale dizer que uma formulação equivalente a (4.2)-(4.6) foi proposta por Cornuejols, Nemhauser e Wolsey[16] e depois explorada por Simão e Thizy[42]. Entretanto, a definição das variáveis correspondentes às variáveis $z$ não é feita diretamente sobre a matriz de custos de transporte, mas a partir da construção de uma *matriz canônica*, conforme definida pelos autores. Esta matriz, de ordem $p \times n$, onde $p = \sum_{j \in J} K_j$, é gerada a partir de uma série

de operações sobre a matriz $[c_{ij}]$ e define a submatriz das restrições (4.5) correspondente às variáveis $y$. A descrição de Labbé e Marin[34], feita de forma independente, é mais direta, levando a uma prova de corretude mais sucinta. De qualquer modo, chamaremos essa formulação de canônica, seguindo denominação original de Cornuejols, Nemhauser e Wolsey[16].

## 4.2 Subproblema Paramétrico

Uma estratégia usual para tratar grandes instâncias do *SPLP* e outros problemas difíceis é procurar reduzir o tamanho da instância pela determinação *a priori* de parte da solução ou, do ponto de vista da formulação associada, pela fixação do valor de algumas variáveis e possível eliminação de restrições. No caso da formulação canônica (4.2)-(4.6), podemos ver duas possibilidades para este fim. A primeira, explícita na própria formulação, está no fato de que $K_j$ pode ser bem menor que $|I|$, reduzindo a quantidade de variáveis e restrições, em relação à formulação clássica original (2.19)-(2.22), como fizemos em (3.1)-(3.5). A segunda advém da seguinte relação de ordem entre as variáveis $z_{jr}$.

**Proposição 13** *As desigualdades $z_{jr} \leq z_{jr-1}$, para $j \in J$ e $r = 3, ..., K_j$, são satisfesitas no ótimo da formulação (PC).*

**Prova.** Por (4.1), se $z_{jr} = 1$ então $\sum_{i:c_{ij}<d_{jr}} y_i \geq \sum_{i:c_{ij}<d_{jr-1}} y_i$ e, consequentemente, $z_{jr-1} = 1$. ■

Desta forma, descobrindo-se que certa variável $z_{jr-1}$ é nula, pode-se concluir o mesmo sobre $z_{jr}$ e, conseqüentemente, pode-se eliminar esta variável e a restrição correspondente em (4.3). Em outras palavras, para cada $j$, existe um $K_j^* \leq K_j$ que poderia ser usado em (4.2)-(4.6), em substituição a $K_j$, gerando um modelo equivalente mais compacto. Esta constatação sugere o desenvolvimento de procedimentos iterativos para determinar o modelo reduzido.

Nesse sentido, definimos um subproblema paramétrico $(PC^t)$, $t \geq 0$, a partir de valores $K_j^t$, $j \in J$, obtido de $(PC)$ com a retirada das restrições em (4.3) associadas a $j \in J$ e

$r > K_j^t$, e a conseqüente eliminação da função objetivo das variáveis $z_{jr}$ correspondentes. Note que, após a retirada destas restrições, tais variáveis recebem valor zero no ótimo. Formalmente, o subproblema é dado por:

$$(PC^t) \qquad \min \sum_{i \in I} f_i y_i + \sum_{j \in J} \sum_{r=2}^{K_j^t} (d_{jr} - d_{jr-1})\, z_{jr} + \sum_{j \in J} d_{j1} \tag{4.7}$$

$$\text{sujeito a: } z_{jr} + \sum_{i:\, c_{ij} < d_{jr}} y_i \geq 1, \forall j \in J, \forall r = 2, \ldots, K_j^t, \tag{4.8}$$

$$\sum_{i \in I} y_i \geq 1, \tag{4.9}$$

$$y_i \in \{0,1\}, \forall i \in I, \tag{4.10}$$

$$z_{jr} \in \{0,1\}, \forall j \in J, \forall r = 2, \ldots, K_j^t. \tag{4.11}$$

Assim, obtemos diretamente a seguinte relação:

**Proposição 14** *($PC^t$) é uma relaxação para (PC).*

Temos, portanto, que o valor ótimo de $(PC^t)$, $t \geq 0$, fornece um limite inferior para $(PC)$. A qualidade desde limite vai depender de quão reduzido seja o subproblema $(PC^t)$, ou seja, da quantidade de restrições e variáveis existentes no modelo. Essa propriedade do subproblema será amplamente utilizada nos algoritmos do tipo *branch-and-bound* para a nova formulação que apresentaremos na Seção 4. Mais precisamente, será usada para nortear o cálculo de limites inferiores nos nós da árvore de *branch-and-bound*.

Claramente, o limite inferior fornecido por $(PC^t)$ é ótimo quando $K_j^t \geq K_j^*$, para todo $j \in J$. Apresentamos a seguir uma condição suficiente para chegar a esta condição.

**Proposição 15** *Suponha que em uma solução ótima de ($PC^t$) ocorra, para todo $j \in J$, que $z_{jK_j^t} = 0$ ou $y_{i_j} = 1$, para $i_j$ tal que $c_{i_j j} = d_{jK_j^t}$. Então uma solução ótima para (PC) pode ser obtida acrescentando $z_{jr} = 0$, para todo $j \in J$ e todo $r > K_j^t$.*

**Prova.** Seja $(y, z)$ a solução construída conforme a proposição e considere $j \in J$. Para $r = 2, ..., K_j^t$, a restrição (4.3) associada está diretamente satisfeita por estar no subproblema

$(PC^t)$. E para $r = K_j^t + 1, \ldots, K_j$, onde se entende $K_j^t < K_j$, segue-se que $z_{jK_j^t} = 0$ ou $y_{i_j} = 1$, implicando $\sum_{i:c_{ij}<d_{jr}} y_i \geq 1$ ou $y_{i_j} = 1$. Então

$$\sum_{i:c_{ij}<d_{jr}} y_i \geq y_{i_j} + \sum_{i:c_{ij}<d_{jK_j^t}} y_i \geq 1,$$

mostrando que a restrição (4.3) correspondente também é verificada. Logo, $(y, z)$ é viável para $(PC)$. Além disso, o valor de $(y, z)$ é o mesmo valor da solução de $(PC^t)$, o que mostra sua otimalidade para $(PC)$, pela Proposição 14. ∎

Vale destacar que, se o subproblema $(PC^t)$ inclui todas as restrições (4.3) para um certo $j \in J$, ou seja, $K_j^t = K_j$, então a condição da Proposição 15 é satisfeita para este $j$. De fato, a restrição (4.6) garante $y_{i_j} = 1$ se $z_{jK_j} = 1$.

## 4.3 Procedimentos Iterativos

### 4.3.1 Limites Inferiores

As proposições 14 e 15 sugerem o desenvolvimento de procedimentos iterativos, que resolvam uma seqüência de subproblemas $(PC^t)$, $t \geq 0$. A cada iteração obtém-se um limite inferior para *SPLP* e, eventualmente, se o processo chegar a satisfazer a condição da Proposição 15, chega-se a uma solução ótima. Em particular, a convergência é trivialmente garantida se tomamos $K_j^{t+1} \geq K_j^t$ para todo $j$ e $K_j^{t+1} > K_j^t$ para algum $j$. Mais ainda, está condição assegura que $V\,[PC] \geq V\,[PC^{t+1}] \geq V\,[PC^t]$.

A esperança com esse processo iterativo é que o esforço gasto com a resolução do conjunto de subproblemas, de menor dimensão, seja inferior àquele demandado para resolver a formulação (4.2)-(4.6) completa.

A seguir iremos descrever alguns possíveis procedimentos que implementam esta idéia. Basicamente, eles se diferenciam pela seqüência de subproblemas formados, ou melhor, em como $K_j^0$ é escolhido e como $K_j^{t+1}$ é atualizado a partir de $K_j^t$. Todos terminam devido à Proposição 15. Observe que a condição para convergência apontada acima sempre é

verificada.

**Fronteira Constante:** Neste procedimento, a cada iteração, aumentamos de um valor fixo, para cada $j \in J$, o número de restrições e variáveis do subproblema, até que a condição da Proposição 15 seja satisfeita. Mais precisamente, para todo $j \in J$, tomamos $K_j^0 = 2$ e $K_j^{t+1} = \min\{K_j, K_j^t + M\}$, com $M > 0$. Com isso mantemos a mesma fronteira $K_j^t$ para todo $j$. Os subproblemas $(PC^t)$ assim definidos são resolvidos exatamente. Este processo foi sugerido por Labbé e Marin[34], que, entretanto, não haviam mostrado sua corretude, agora uma decorrência da Proposição 15.

**Fronteira Variável:** Primeiro, encontramos uma solução viável $(z', y')$ para $(PC)$, usando uma das heurísticas propostas na Seção 3.3.2, e definimos o subproblema inicial $(PC^0)$ fazendo $K_j^0 = \min\{K_j, \max\{r : z'_{jr} = 1\} + 1\}$. Note que o processo iterativo já começa com todas as variáveis $z_{jr}$ que foram fixadas em 1 na solução heurística e, possivelmente, a primeira fixada em zero, para cada $j \in J$. Então, iterativamente, resolvemos de forma exata o subproblema $(PC^t)$, encontrando solução ótima $(z^t, y^t)$, e geramos o subproblema $(PC^{t+1})$ fazendo $K_j^{t+1} = \min\{K_j, K_j^t + M\}$, com $M > 0$, se $z^t_{jK_j^t} = 1$, ou mantendo $K_j^{t+1} = K_j^t$, caso contrário.

**Fronteira Variável Baseada na Relaxação Linear:** Este procedimento mistura idéias dos dois anteriores e procura reduzir o esforço computacional, resolvendo apenas a relaxação linear da maioria dos subproblemas gerados. Começamos com $K_j^0 = 2$ como no procedimento Fronteira Constante, e atualizamos $K_j^{t+1}$ de forma similar ao procedimento Fronteira Variável. Na verdade, em uma primeira fase, fazemos $K_j^{t+1} = \min\{K_j, K_j^t + M\}$, se $z^t_{jK_j^t} > 0$, ou $K_j^{t+1} = K_j^t$, caso contrário, mas agora baseados na solução $(z^t, y^t)$ da relaxação linear de $(PC^t)$. Em outras palavras, não resolvemos o subproblema considerando as variáveis inteiras, mas contínuas. Quando em uma iteração ocorrer $K_j^{t+1} = K_j^t$, para todo $j$, passamos a resolver de forma exata os novos subproblemas (com as restrições de integralidade) e atualizamos $K_j^{t+1}$ exatamente como no procedimento Fronteira Variável.

Algumas observações merecem ser feitas sobre esses procedimentos. Primeiro, o valor de $M$ é escolhido de acordo com o tamanho do problema original e a partir da análise de vários testes realizados com instâncias da literatura. Poderíamos, na verdade, usar valores diferentes de $M$ para cada $j$. Segundo, pode-se pensar em outros critérios de convergência que permitam $K_j^{t+1} < K_j^t$ para algum $j$, em alguma iteração $t$, levando a reduzir o tamanho do subproblema de uma iteração para a outra, quando há indícios de que $K_j^t > K_j^*$. Sob esta perspectiva, de considerar a cada iteração o menor subproblema possível, avaliamos o seguinte procedimento.

**Fronteira Reduzida:** Tomamos $K_j^0$ como no procedimento Fronteira Variável. Então, encontrada uma solução ótima $(z^t, y^t)$ para $(PC^t)$, definimos o subproblema $(PC^{t+1})$ para $K_j^{t+1} = \min\left\{K_j, \max\left\{z_{jr}^t = 1\right\} + M\right\}$. Ou seja, introduzimos, para cada $j$, apenas $M > 0$ variáveis $z_{jr}$ a mais do que aquelas para os quais o valor 1 foi atribuído na solução de $(PC^t)$. Verificamos que o procedimento acima pode não convergir. É possível que o subproblema $(PC^t)$ seja igual a um subproblema subseqüente $(PC^{t+\delta})$, $\delta > 0$, levando o algoritmo a ciclar.

Por outro lado, é importante ressaltar que, a cada iteração de qualquer dos algoritmos descritos acima, obtemos um limite inferior para $(PC)$, que é dado pelo valor da solução $(z^t, y^t)$ do subproblema $(PC^t)$ ou sua relaxação linear, conforme Proposição 14. Sendo assim, mesmo que o processo não convirja, ou seja, pare antes de convergir, obtemos ao final um limite inferior para o valor ótimo.

### 4.3.2 Limites Superiores

Da mesma forma que limite inferior, podemos gerar também limites superiores através de algoritmos iterativos similares àqueles apresentados na Subseção 4.3.1. Particularmente, podemos estender qualquer dos procedimentos apresentados na Seção 4.3.1 para calcular também, iterativamente, limites superiores e, conseqüentemente, um gap de dualidade, o que permite parar o processo antes do ótimo com uma aproximação conhecida

deste. O limite inferior é dado diretamente pela solução $(z^t, y^t)$ do subproblema $(PC^t)$ ou sua relaxação linear, conforme Proposição 14. Já o limite superior pode ser obtido a partir de heurísticas primais, que podem ser desenvolvidas usando $(z^t, y^t)$. Em nossos experimentos computacionais, implementamos a seguinte heurística.

**Heurística Primal** Começamos considerando abertas as facilidades indicadas em $y^t$, ou seja, aquelas em $K = \{i \in I : y_i = 1\}$. Então, para cada $j \in J$ que não satisfaz a condição da Proposição 15, determinamos $i^* \in \arg\min\{f_i : c_{ij} = d_{jK_j^t}\}$ e fazemos $K = K \cup \{i^*\}$ se $Z(K \cup \{i^*\}) < Z(K)$, isto é, se a abertura da facilidade $i^*$ melhorar a solução. A definição de $i^*$ relaciona-se à expectativa de que as variáveis $z_{jr}$, para $r > K_j^t$, assumiriam valor 0 se presentes no subproblema.

Outra forma de obter limites superiores, usando procedimentos iterativos baseados na formulação canônica, é apresentado a seguir. Considere o subproblema $(PC^t)$ acrescido das seguintes restrições,

$$\sum_{i:\, c_{ij} < d_{j(K_j^t+1)}} y_i \geq 1, \forall j \in J : K_j^t < K_j. \tag{4.12}$$

Note que este problema, que denotaremos por $(\overline{PC}^t)$, é equivalente a $(PC)$ acrescido das restrições:

$$z_{j\left(K_j^t+1\right)} = 0 \ \forall j \in J : K_j^t < K_j.$$

De fato, pela Proposição 13, a fixação dessas variáveis em zero leva à eliminação das variáveis $z_{jr}$ seguintes $\left(j \in J,\, r > K_j^t + 1\right)$ e à obtenção da restrição (4.12). Sendo assim, concluímos diretamente que o valor ótimo de $(\overline{PC}^t)$ fornece um limite superior para $(PC)$.

Sugerimos duas maneiras de usar o subproblema restrito $(\overline{PC}^t)$, como segue:

**Procedimento Combinado:** Aplicamos algum dos procedimentos para cálculo de limite inferior até certa iteração $t$. Então, resolvemos o subproblema $(\overline{PC}^t)$, obtendo um limite superior e, conseqüentemente, um gap de dualidade.

**Problema Restrito Iterativo:** Resolvemos o subproblema inicial $(\overline{PC}^0)$ com $K^t$ como no procedimento Fronteira Variável. A partir da solução ótima $(z^t, y^t)$ de $(\overline{PC}^t)$, definimos o subproblema $(\overline{PC}^{t+1})$, fazendo

$$K_j^{t+1} = \begin{cases} \min\left\{K_j,\, K_j^t + M\right\}, & \text{se } z^t_{jK_j^t} = 1 \\ K_j^t, & \text{caso contrario} \end{cases}$$

Duas observações merecem ser destacadas. O procedimento combinado faz-se conveniente quando desejamos parar o processo interativo para cálculo do limite inferior antes de sua convergência, por exemplo, quando impomos um limite de tempo de execução. E sobre o segundo procedimento é importante frisar que a solução obtida ao final, quando teríamos $K_j^{t+1} = K_j^t$, para todo $j \in J$, não é necessariamente ótima. Em outras palavras, mesmo que as desigualdades (4.12) sejam satisfeitas de forma estrita por uma solução, ela pode não ser ótima. É o que podemos ver com o exemplo abaixo, onde $I = \{1, 2, 3\}$, $J = \{a, b\}$, $c_{1a} = c_{1b} = c_{2a} - 1 = c_{2b} - 1 = c_{3a} - 2 = c_{3b} - 2$:

$$\begin{aligned} \min \quad & 6y_1 + \infty y_2 + y_3 + z_{a2} + z_{a3} + z_{b2} + z_{b3} \\ \text{sujeito a:} \quad & z_{a2} + y_1 \geq 1, \\ & z_{a3} + y_1 + y_2 \geq 1, \\ & z_{b2} + y_1 \geq 1, \\ & z_{b3} + y_1 + y_2 \geq 1, \\ & y_1 + y_2 + y_3 \geq 1, \\ & y_i \in \{0, 1\}, \forall i \in I, \\ & z_{jr} \in \{0, 1\}, \forall j \in J, \forall r = 2, 3. \end{aligned}$$

De fato, quando fixamos $z_{a3} = z_{b3} = 0$ no problema acima, obtemos a solução $y = (1, 0, 0)$ e $z_{a2} = z_{b2} = 0$ de valor 6. Como as variáveis $z$ são nulas, chegamos ao final do procedimento. Entretanto, a solução ótima é $y = (0, 0, 1)$ e $z = 1$, com valor 5.

### 4.3.3 Resultados Computacionais

Implementamos os vários procedimentos iterativos descritos na Subseção 4.3, utilizando a linguagem Mosel do software de programação matemática XPressIV, que foi usado para resolver os problemas lineares e inteiros. Limitamos o tempo de execução de cada algoritmo em uma hora. Caso este tempo limite seja ultrapassado, o algoritmo é abortado e uma nova execução é feita. Desta vez, após a resolução de cada subproblema ($PC^t$), aplicamos a heurística primal da Subseção 4.3.2. Com isso teremos a cada iteração uma solução viável, que pode ser melhorada toda vez que um novo subproblema é resolvido. Isso aumenta o tempo de execução dos algoritmos, devido ao fato de que estamos aplicando a heurística primal seguidamente, mas nos permite avaliar o limite inferior obtido a cada iteração.

Em qualquer caso, ao final do algoritmo, obtemos um limite inferior (LI). Caso a convergência se dê no tempo limite, LI é o próprio valor da solução ótima. Do contrário, quando o processo é interrompido, a heurística primal fornece um limite superior (LS).

Na tabela 4.1, registramos os resultados obtidos com cada algoritmo aplicado a problemas não-euclideanos, descritos na Seção 2.5. Além de LI e LS, apresentamos o tempo de execução em segundos e o gap dado por 100*(LS - LI)/LI. Quando o algoritmo converje, LS é o próprio valor da solução ótima encontrada.

Podemos notar pela Tabela 4.1 que o Procedimento Fronteira Fixa encontrou a solução ótima, no tempo limite, de todos os problemas dos grupos 1 e 2, onde os custos fixos são pequenos, e os problemas menores dos grupos 3, 4 e 5, onde os custos fixos são grandes. O mesmo aconteceu com os Procedimentos Fronteira Variável e Fronteira Variável Baseada na Relaxação Linear, à exceção do problema 3:150. Vemos ainda que os problemas com custos fixos altos e dimensão $n = m > 150$ tornam-se proibidos para estes procedimentos.

Com respeito ao tempo, não houve supremacia clara de um sobre outro, embora aparentemente o Procedimento Fronteira Variável seja mais rápido. Em comparação ao tempo requerido para resolver a formulação completa, pudemos constatar que este é sempre maior que aquele gasto por qualquer dos procedimentos, em todos os problemas.

| | Fronteira Constante | | Fronteira Variável | | Fronteira Variável RL | | | |
|---|---|---|---|---|---|---|---|---|
| Prob | LS | Tempo | LS | Tempo | LS | Tempo | LI | gap% |
| 1:50 | 488 | 0.05 | 488 | 0.06 | 488 | 0.06 | 488 | 0 |
| 1:100 | 914 | 0.19 | 914 | 0.20 | 914 | 0.19 | 914 | 0 |
| 1:150 | 1438 | 0.42 | 1438 | 0.43 | 1438 | 0.43 | 1438 | 0 |
| 1:200 | 1860 | 0.79 | 1860 | 0.81 | 1860 | 0.81 | 1860 | 0 |
| 1:400 | 3504 | 3.60 | 3504 | 3.65 | 3504 | 3.69 | 3504 | 0 |
| 1:600 | 5054 | 9.36 | 5054 | 9.44 | 5054 | 9.51 | 5054 | 0 |
| 1:800 | 6621 | 18.57 | 6621 | 18.62 | 6621 | 18.69 | 6621 | 0 |
| 2:50 | 2864 | 0.18 | 2864 | 0.11 | 2864 | 0.09 | 2864 | 0 |
| 2:100 | 4618 | 0.38 | 4618 | 0.38 | 4618 | 0.47 | 4618 | 0 |
| 2:150 | 5536 | 1.68 | 5536 | 1.59 | 5536 | 1.81 | 5536 | 0 |
| 2:200 | 6495 | 7.08 | 6495 | 3.53 | 6495 | 3.42 | 6495 | 0 |
| 2:400 | 10025 | 7.40 | 10025 | 8.04 | 10025 | 9.30 | 10025 | 0 |
| 2:600 | 12418 | 28.62 | 12418 | 26.54 | 12418 | 39.86 | 12418 | 0 |
| 2:800 | 15627 | 500.36 | 15627 | 340.49 | 15627 | 198.98 | 15627 | 0 |
| 3:50 | 5701 | 0.57 | 5701 | 0.46 | 5701 | 0.55 | 5701 | 0 |
| 3:100 | 9243 | 378.51 | 9243 | 261.32 | 9243 | 310.19 | 9243 | 0 |
| 3:150 | 11314 | 3600 | 11314 | 2471.53 | 11314 | 3659.46 | 11314 | 0 |
| 4:50 | 7290 | 3.47 | 7290 | 4.16 | 7290 | 4.73 | 7290 | 0 |
| 4:100 | 11314 | 1839.11 | 11314 | 1275.35 | 11310 | 3650.92 | 11314 | 0 |
| 4:150 | 13785 | 3600 | 13785 | 4207.7 | 13785 | 4884.23 | 13756 | 0.0021 |
| 5:50 | 9214 | 7.14 | 9214 | 6.64 | 9214 | 8.00 | 9214 | 0 |
| 5:100 | 13685 | 3803.66 | 13685 | 1368.21 | 13685 | 2979.52 | 13685 | 0 |
| 5:150 | 19984 | 3600 | 21469 | 3600 | - | - | 16997 | 0 |

Tabela 4.1: Resultados computacionais com procedimentos iterativos

## 4.4 Branch-and-Bound

### 4.4.1 Estrutura Geral

Desenvolvemos algoritmos do tipo *branch-and-bound* que utilizam os processos iterativos de redução do tamanho do problema descritos na Subseção 4.3.1. Mais precisamente, utilizaremos tais procedimentos para obter limites inferiores em cada nó da árvore e para a ramificação.

A idéia principal destes algoritmos é associar a cada nó da árvore um subproblema paramétrico ($PC^t$), cuja solução fornecerá um limite inferior e um limite superior, conforme descrito nas subseções 4.3.1 e 4.3.2. Tal associação é feita de modo que, ao descermos na árvore, o subproblema é acrescido de mais variáveis do tipo $z$ e suas restrições associadas. Desta forma, restringimos cada vez mais os subproblemas, fazendo com que os limites inferiores cresçam ao longo de qualquer ramo. Note que, à medida que descemos na árvore de *branch-and-bound*, embora a dimensão do subproblema cresça, mais variáveis são fixadas. Isto tende a equilibrar o custo de encontrar tais limites.

Desenvolvemos vários algoritmos *branch-and-bound* baseados nesta estratégia. Para apresentação, vamos dividi-los em dois grupos, conforme a regra de ramificação utilizada. No primeiro grupo, a ramificação é feita sobre a variável $y$ e, no segundo, sobre a variável $z$. Nas duas subseções seguintes iremos detalhar cada algoritmo de cada grupo. Agora, vamos descrever algumas características comuns a todos eles.

**Limite Inferior:** O limite inferior em cada nó da árvore de *branch-and-bound* é dado pelo valor da solução do subproblema ($PC^t$) a ele associado. Dependendo do processo iterativo usado para resolver o subproblema, obtemos um algoritmo diferente, cuja eficiência é fortemente influenciada pelo processo escolhido.

**Limite Superior:** O limite superior ou solução viável para cada nó da árvore de *branch-and-bound* é calculado a partir da solução do subproblema ($PC^t$). Quando a Proposição

15 é satisfeita, a própria solução ótima do subproblema é viável para $(PC)$. Além disso, se os stati de todas as facilidades forem fixados pela ramificação, obtemos uma solução viável a partir das facilidades abertas. Em qualquer dos dois casos, chegamos à folha deste ramo. Para os outros casos, iremos obter uma solução viável desta forma:

1. Calculamos o valor da solução obtida considerando os stati das facilidades na solução do subproblema $(PC^t)$.

2. Calculamos uma outra solução considerando as facilidades abertas na solução do subproblema $(PC^t)$ e abrindo as facilidades com stati fechada na solução do subproblema mas que, para algum $j$ que não satisfaz a Proposição 15, esta facilidade é a $K_j$-ésima melhor para atender $j$.

3. Escolhemos a melhor das duas soluções.

**Pré-Processamento:** Decidimos não aplicar nenhuma forma de pré-processamento, pois constatamos, a partir de alguns testes computacionais, que o pré-processamento utilizando os testes de redução para fixar os stati das facilidades não traz nenhuma melhoria considerável.

### 4.4.2 Ramificação na Variável $y$

Para estabelecermos a regra de ramificação deste grupo de algoritmos, lembremos que, em cada nó da árvore, é resolvido um subproblema $(PC^t)$, ou melhor, um subproblema $(PC^t)$ com algumas variáveis fixadas em 0 ou 1. Dito isto, definimos a prioridade de uma variável $y_i$ em um nó $q$ da árvore como o número de vezes que esta variável recebeu 1 nos ótimos dos subproblemas associados aos nós do ramo que vão da raiz até $q$. A variável com maior prioridade em $q$ é aquela escolhida para ramificar este nó.

O subproblema associado a um nó é obtido a partir do subproblema do nó pai, com uma variável $y_i$ fixada pela ramificação e acrescido de algumas novas variáveis $z_{jr}$ e restrições associadas, escolhidas de acordo com uma das regras adotadas nos procedimentos

iterativos propostos da Subseção 4.3.1 (Ver Figura 4.1). Desta forma, vários algoritmos são gerados, cada um associado a um procedimento de redução de tamanho, conforme descrição abaixo.

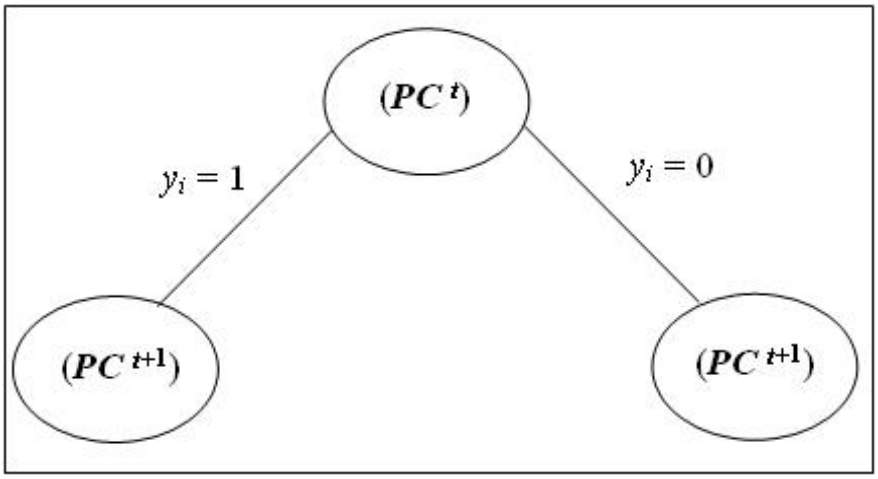


Figura 4.1: Ramificação na Variável $y$

**Fronteira Constante:** Esse algoritmo baseia-se no procedimento Fronteira Constante de redução do tamanho do problema. No nó raiz associamos um subproblema definido por uma certa quantidade inicial de variáveis $z_{jr}$ e suas restrições associadas, que depende do tamanho do problema. Essa quantidade foi determinada através da observação de testes realizados. O subproblema correspondente a qualquer outro nó é obtido a partir daquele associado ao seu pai, acrescido de mais variáveis e restrições, em uma quantidade fixa $M$, para todo $j \in J$, conforme procedimento Fronteira Constante. O valor de $M$ também foi estabelecido através da observação de testes realizados. Em cada nó, o subproblema é resolvido considerando suas variáveis inteiras. Observamos que não se estabelece um processo iterativo em um nó, mas ao longo de um ramo.

**Fronteira Variável:** Aqui tomamos como base o procedimento Fronteira Variável da Subseção 4.3.1. Isto significa dizer que esse algoritmo B&B é similar ao anterior, porém

variáveis $z_{jr}$ são acrescidas em um nó filho apenas para $j \in J$ tal que na solução do nó pai $z_{jK_j} = 1$. A quantidade de variáveis e restrições adicionadas é igual para todos esses $j$'s.

**Fronteira Baseada na RL - I:** Neste algoritmo, aplicamos em cada nó o procedimento Fronteira Variável Baseada na Relaxação Linear da Subseção 4.3.1, até que o primeiro subproblema com restrições de integralidade seja resolvido. Mais exatamente, resolvemos iterativamente subproblemas paramétricos lineares, acrescentando novas variáveis e restrições a cada iteração, até que a solução (relaxada) obtida satisfaça a Proposição 15. Neste momento, resolvemos novamente o último subproblema, agora com restrições de integralidade.

**Fronteira Baseada na RL - II:** Este algoritmo B&B é igual ao anterior, exceto pela forma como calculamos uma solução viável a partir da solução do subproblema. Nesse caso, substituímos a heurística descrita na Subseção 4.3.2 pela seguinte. Ordenamos as facilidades conforme a ordem não-crescente do valor das variáveis $y$ na solução do subproblema relaxado. Seguindo essa ordem, abrimos facilidades, enquanto isto melhorar a solução. Depois aplicamos o método de troca de Pereira[38].

### 4.4.3 Ramificação na Variável $z$

Como nos algoritmos da Subseção 4.4.2, aqui também associamos a cada nó da árvore de B&B um subproblema paramétrico $(PC^t)$, cuja solução fornecerá o limite inferior. Esta solução, embora inteira, não é geralmente viável, uma vez que algumas restrições foram desconsideradas. Nos algoritmos apresentados nesta seção, definimos uma regra de ramificação para cujos subproblemas gerados tal solução seja inviável.

Pela Proposição 15, se a solução $(z^t, y^t)$ do subproblema paramétrico $(PC^t)$ associado a um nó não satisfaz a condição de poda, então existe $j \in J$ tal que

***i)*** $z^t_{jK^t_j} = 1$ e

***ii)*** $y^t_{i_j} = 0$, para $j \in J$ tal que $c_{ij} = d_{jK^t_j}$

Sendo assim para tal $j \in J$ e $r = K^t_j + 1$, temos que

$$\sum_{i:c_{ij}<d_{jr}} y^t_i = 0. \tag{4.13}$$

Esta constatação nos indica a variável $z_{jr}$, com $j$ satisfazendo $(i)$ e $(ii)$ e $r = K^t_j + 1$, para ramificar o subproblema $(PC^t)$. Note que a solução $(z, y)$ obtida de $(z^t, y^t)$ por

$$y = y^t \text{ e } z_{jr} = \begin{cases} z^t_{jr}, \text{ se } j \in J,\ r \leq K^t_j \\ 0, \text{ se } j \in J,\ r \geq K^t_j + 1 \end{cases}$$

não é viável para qualquer dos dois subproblemas gerados com a ramificação. Trivialmente, ela é inviável para $z_{jK^t_j} = 1$. E o mesmo ocorre por (4.13) para $z_{jK^t_j+1} = 0$, pois a restrição $z_{jr} + \sum_{i:c_{ij}<d_{jr}} y_i \geq 1$ é violada.

Quando há mais de um consumidor $j$ satisfazendo $(i)$ e $(ii)$, sugerimos duas regras alternativas de desempate:

- Escolher o valor de $j$ tal que $K^t_j + 1$ seja o menor possível.
- Escolher o valor de $j$ tal que $K^t_j + 1$ seja o maior possível.

Nos nossos algoritmos optamos pelo segundo critério devido ao fato que primeiro analisamos o ramo que fixa em 1.

Pela Proposição 13, quando usamos a primeira regra, estamos beneficiando o ramo que fixa a variável em 0, pois dessa forma fixamos $z_{jr} = 0, \forall r > K^t_j + 1$.

De outro modo, a segunda regra beneficia o ramo que fixa a variável em 1, pois com isso fixamos $z_{jr} = 1, \forall r < K^t_j + 1$.

### 4.4.4 Resultados Computacionais

Aqui apresentamos os resultados computacionais obtidos com a implementação dos algoritmos *branch-and-bound*. Para cada um dos três procedimentos iterativos descritos

na Subseção 4.4.2, codificamos um algoritmo com o auxílio do software de programação matemática XPress.

Na tabela 4.2, mostramos os resultados obtidos pelos vários algoritmos *branch-and-bound* implementados. As células em branco correspondem a instâncias que não foram resolvidas no tempo de 1 hora.

Podemos observar pelos resultados na tabela que não houve uma supremacia de um algoritmo sobre os outros, havendo uma alternância de melhor tempo entre os algoritmos. Isso pode ser explicado pela natureza e dimensão das instâncias e pelo método aplicado no algoritmo que, em conjunto, podem ter facilitado a resolução de alguns problemas e dificultado a de outros.

| | Ramificação em $y$ | | | | | | Ramificação em $z$ | |
|---|---|---|---|---|---|---|---|---|
| | Fronteira Variável | | Fronteira RL-I | | Fronteira Constante | | Fronteira RL-I | |
| Prob | LS | Tempo | LS | Tempo | LS | Tempo | LS | Tempo |
| 1:50 | 488 | 0.18 | 488 | 0.18 | 488 | 0.11 | 488 | 0.18 |
| 1:100 | 914 | 0.24 | 914 | 0.24 | 914 | 0.62 | 914 | 0.24 |
| 1:150 | 1438 | 0.84 | 1438 | 0.84 | 1438 | 1.27 | 1438 | 0.84 |
| 1:200 | 1860 | 0.95 | 1860 | 0.947 | 1860 | 2.27 | 1860 | 0.947 |
| 1:400 | 3504 | 4.17 | 3504 | 4.17 | 3504 | 11.47 | 3504 | 4.17 |
| 1:600 | 5054 | 11.42 | 5054 | 11.42 | 5054 | 33.66 | 5054 | 11.42 |
| 1:800 | 6621 | 22.87 | 6621 | 22.87 | 6621 | 70.84 | 6621 | 22.87 |
| 2:50 | 2864 | 0.14 | 2864 | 0.13 | 2864 | 0.45 | 2864 | 0.18 |
| 2:100 | 4618 | 0.41 | 4618 | 0.48 | 4618 | 1.74 | 4618 | 0.42 |
| 2:150 | 5536 | 1.89 | 5536 | 1.96 | 5536 | 10.39 | 5536 | 1.71 |
| 2:200 | 6495 | 9.97 | 6495 | 5.43 | 6495 | 35.95 | 6495 | 5.42 |
| 2:400 | 10025 | 8.98 | 10025 | 9.63 | 10025 | 101.56 | 10025 | 8.69 |
| 2:600 | 12418 | 30.09 | 12418 | 30.81 | 12418 | 845.65 | 12418 | 27.50 |
| 2:800 | 15627 | 564.49 | 15627 | 336.09 | - | - | 15627 | 357.98 |
| 3:50 | 5701 | 0.56 | 5701 | 0.78 | 5701 | 1.53 | 5701 | 1.69 |
| 3:100 | 9243 | 278.14 | 9243 | 326.80 | 9243 | 604.66 | 9243 | 255.14 |
| 3:150 | - | - | 11314 | 3496.53 | - | - | - | - |
| 4:50 | 7290 | 4.79 | 7290 | 6.99 | 7290 | 7.75 | 7290 | 13.88 |
| 4:100 | 11314 | 854.11 | 11314 | 506.73 | 11314 | 1143.98 | 11314 | 3732.67 |
| 5:50 | 9214 | 6.77 | 9214 | 17.88 | 9214 | 17.17 | 9214 | 15.16 |
| 5:100 | 13685 | 674.07 | 13685 | 491.54 | - | - | - | - |

Tabela 4.2: Resultados computacionais dos Branch-and-Bound

# Capítulo 5

# Relaxação Lagrangeana

Em otimização combinatória, observa-se que muitos problemas considerados difíceis ou mesmo problemas cuja relaxação linear e limites inferiores são de difícil resolução ou obtenção, podem ser vistos como problemas fáceis que são complicados por um conjunto de restrições, isto é, quando se retira esse conjunto de restrições, a resolução do problema se torna fácil e até mesmo trivial.

A dualização deste conjunto de restrições complicadoras, isto é, a transferência destas para a função objetivo multiplicadas por um vetor de multiplicadores, chamado de multiplicadores de Lagrange, produz um problema de fácil resolução, cujo valor da solução ótima é um limite inferior (para problemas de minimização) para o valor ótimo do problema original. Este problema é chamado de Problema Lagrangeano, e esta idéia é chamada de Relaxação Lagrangeana.

O Problema Lagrangeano pode, portanto, ser usado no lugar da relaxação linear para produzir limites inferiores em algoritmos do tipo *branch-and-bound*. Além disso, com base nesse limite inferior, é possível estimar quão próxima está uma solução viável disponível da solução ótima.

A Relaxação Lagrangeana foi desenvolvida em meados dos anos 70, no trabalho pioneiro de Held e Karp[26, 27] para o problema do caixeiro viajante, e hoje é uma técnica considerada indispensável para gerar limites inferiores para serem usadas em algoritmos

de otimização combinatória.

## 5.1 Problema Lagrangeano

Considere um problema de otimização combinatória, formulado como problema de programação inteira, da seguinte forma:

$$(PI) \quad \min f(x) \tag{5.1}$$

$$\text{sujeito a: } g_i(x) \leq 0 \ \forall i = 1, \ldots, m \tag{5.2}$$

$$Ax \leq b \tag{5.3}$$

$$x = (x_1, x_2, \ldots, x_n)^T \in Z_+^n \tag{5.4}$$

onde $f(x)$ e $g_i(x) \leq 0$, $\forall i = 1, \ldots, m$, são funções arbitrárias, com $g$ sendo difícil de tratar (representando aqui as restrições complicadoras, ou seja, aquelas que dificultam o problema). Muitos problemas combinatóriais difíceis de resolver encaixam-se na estrutura do problema $(PI)$. Isso acontece, por exemplo, com as duas formulações apresentadas para o *SPLP*.

Pelas características descritas, o problema $(PI')$ $\min\{f(x) : x \in X\}$, onde $X = \{x \in Z_+^n : Ax \leq b\}$ é de fácil resolução. Note que o problema $(PI')$ fornece um limite inferior de $(PI)$, porém sua solução ótima em geral não satisfaz $g(x) \leq 0$.

Procurando penalizar a inviabilidade da solução obtida, para cada restrição (5.2), associa-se um número real $u_i \geq 0$, chamado *multiplicador de Lagrange*, e define-se a *função Lagrangeana* como sendo:

$$FL(x, u) = f(x) + ug(x)$$

onde $u = (u_1, u_2, ..., u_m)$ e $g(x) = [g_1(x), g_2(x), ..., g_m(x)]^T$ . Já o *problema lagrangeano* é dado por

$$L(u) = \min_{x \in X}\{FL(x, u)\} \tag{5.5}$$

Uma observação importante é que se as restrições complicadoras, dadas pela função $g(x)$, forem de igualdade, o multiplicador $u$ torna-se irrestrito em sinal.

Podemos mostrar que (5.5) é uma relaxação de $(PI)$, como segue:

**Proposição 16** *Para qualquer $u \in R_+^m$ e para toda solução viável $x = (x_1, x_2, ..., x_n)^T \in Z_+^n$ de (PI), tem-se $L(u) \leq f(x)$. Em particular, $L(u) \leq f(x^*)$, onde $x^*$ é uma solução ótima de (PI).*

**Prova.** Por definição, $L(u) \leq FL(x,u) = f(x) + ug(x)$, $\forall x \in X$. Como $u_i \geq 0$ e $g_i(x) \leq 0$, pois $x$ é viável para $(PI)$, tem-se $L(u) \leq f(x)$. Em particular, para $x^*$ ótimo de $(PI)$, tem-se que $L(u) \leq L(u^*) \leq f(x^*)$. ■

Uma conseqüência imediata desta propriedade é que, se o problema dual $(DL)$ tem um ótimo ilimitado $(\infty)$, então o primal $(PI)$ não tem solução viável.

Note que a idéia da relaxação é levar as restrições complicadoras para a função objetiva, penalizando-as com os multiplicadores $u$. Vale ressaltar que devemos construir o problema lagrangeano de forma que, para todo $u \in R_+^m$, exista um bom algoritmo para computar $L(u)$. Por bom algoritmo, entende-se um algoritmo rápido ou até mesmo trivial.

Mas uma questão importante aqui é saber qual é o conjunto de multiplicadores $u$ que corresponde ao melhor limitante $L(u)$ para $(PI)$. Neste sentido, definimos o problema *dual lagrangeano* de $(PI)$ como:

$$(DL) \quad \max_{u \in R_+^m} L(u)$$

ou

$$(DL) \quad \max_{u \in R_+^m} \min_{x \in X} \{FL(x,u)\}$$

A questão que se apresenta agora é como resolver o dual lagrangeano. Note que $L(u)$ é não-diferenciável em geral. Além disso, pode ser uma função não-côncava. A seguir vamos considerar um caso particular onde se enquadra o *SPLP*, qual seja, as funções $f()$ e $g_i()$, $i = 1, \ldots, m$, são lineares. Neste caso, $L()$ passa a ser côncava linear por partes.

## 5.2 Método de Subgradientes

Considere que as funções $f()$ e $g_i() \leq 0$, $\forall i = 1, \ldots, m$, em (5.1)-(5.2), são da forma

$$f(x) = cx$$
$$g(x) = d - Dx$$

onde $c = (c_1, c_2, \ldots, c_n)$, $d = (d_1, d_2, \ldots, d_m)^T$ e $D$ é uma matriz real de ordem $m \times n$. Assim, tem-se que

$$L(u) = \min_{x \in X} cx + u(d - Dx) = \min_{x \in X}(c - uD)x + ud \tag{5.6}$$

Como $L()$ é o ínfimo de uma família de funções lineares, pode-se mostrar que $L()$ é uma função côncava[41]. Por outro lado, $L()$ é não-diferenciável, ou seja, o gradiente de $L()$ pode não ser definido em todos os pontos do seu domínio. A generalização do conceito de gradiente para estas funções é dada a seguir.

**Definição 17** *Seja $h : \mathbb{R}^m \to \mathbb{R}$ uma função côncava. Um subgradiente de $h$ em $\overline{u} \in \mathbb{R}^{1 \times m}$ é um vetor $s \in \mathbb{R}^m$ tal que $h(u) \leq h(\overline{u}) + (u - \overline{u})s$, para todo $u \in \mathbb{R}^m$.*

No caso da função $h()$, um subgradiente em $\overline{u}$ é dado diretamente pela solução do problema linear que define $L(\overline{u})$, como podemos observar abaixo.

**Proposição 18** *Sejam $\overline{u} \in \mathbb{R}^m$ e $\overline{x} \in X$ tais que $L(\overline{u}) = c\overline{x} + \overline{u}(d - D\overline{x})$. Então, um subgradiente de $L$ em $\overline{u}$ é $s = d - D\overline{x}$.*

**Prova.** Seja $u \in \mathbb{R}^m$. Então

$$L(u) \leq c\overline{x} + u(d - D\overline{x}) = c\overline{x} + \overline{u}(d - D\overline{x}) + (u - \overline{u})(d - D\overline{x}) = L(\overline{u}) + (u - \overline{u})s$$

■

A propriedade acima é a base de um dos métodos mais tradicionais para encontrar o máximo de $L()$, ou seja, resolver o dual lagrangeano($DL$). Este método é conhecido

na literatura como método de subgradientes. Ele consiste basicamente em gerar uma seqüência $\{u^k\} \subseteq \mathbb{R}^m_+$, que deve convergir para uma solução ótima de $(DL)$, conforme algoritmo abaixo.

**P1.** Escolha $u^0 \in \mathbb{R}^m_+$ e faça $k = 0$.

**P2.** Dado $u^k \in \mathbb{R}^m_+$, calcule $L(u^k) = \min_{x \in X} \{(c + uD)x - ud\}$.

**P3.** Encontre $u^{k+1} = u^k + \theta^k s^k$, onde $s^k$ é um subgradiente de $L$ em $u^k$ e $\theta^k$ é o tamanho do passo. Se $u^{k+1} \notin \mathbb{R}^m_+$ então projete $u^{k+1}$ sobre $\mathbb{R}^m_+$. Faça $k = k+1$ e retorne para P2.

A convergência deste algoritmo depende da escolha dos passos. Condições suficientes podem ser encontradas em Wolsey[44]. Na implementação deste algoritmo sugerida por Beasley[9], as seguintes escolhas são adotadas:

$$\begin{aligned} u^{k+1} &= \max\{0, u^k + \theta^k \overline{s}^k\}, \\ \overline{s}^k_j &= \begin{cases} s^k_j & \text{, se } u^k_j \neq 0 \text{ ou } u^k_j + \theta^k s^k_j \geq 0, \\ 0 & \text{, caso contrário} \end{cases} \\ \theta^k &= \pi(1,05LS - L(u^k))/\left\|\overline{s}^k\right\|^2_2, \end{aligned}$$

sendo $LS$ o melhor limite superior conhecido e $\pi \in [0, 2]$ um parâmetro que decresce à medida que se passam várias iterações sem aumentar o valor do limite inferior. O processo termina quando a norma $\left\|\overline{s}^k\right\|^2_2$, o parâmetro $\pi$ ou o gap $LS - L(u^k)$ são pequenos. Para maiores detalhes sugerimos uma consulta a Beasley[9].

Simultaneamente ao procedimento de subgradientes, podemos utilizar heurísticas, com o objetivo de encontrar soluções viáveis melhoradas, ou seja, limites superiores menores, que serão aplicados no algoritmo acima, para determinar o tamanho do passo a ser utilizado na busca de um novo multiplicador lagrangeano[9].

## 5.3 Limites via Método de Subgradientes

Como pôde ser visto na seção anterior, o método de subgradientes é bastante simples, sendo cada iteração essencialmente definida pela solução do subproblema lagrangeano $L(u^k)$. Dependendo da escolha dos passos, a convergência pode ser muita lenta ou mesmo não ocorrer. De qualquer modo, a cada iteração, um limite inferior é obtido.

Dentro dessa perspectiva, muitas implementações do método de subgradientes não possuem o compromisso de resolver o dual lagrangeano, mas têm a finalidade de maximizar o limite inferior obtido pelo problema relaxado, através do ajuste de multiplicadores. Geralmente, as heurísticas de relaxação com otimização por subgradientes são caracterizadas pelos seguintes passos:

a) utilizam o problema lagrangeano para definir um limite inferior $f_{inf}$ para o $(PI)$;

b) tentam maximizar o limite inferior via otimização de subgradientes;

c) produzem soluções viáveis a partir das soluções do problema lagrangeano, o que define um limite superior $f_{sup}$ para o $(PI)$;

O passo $(c)$ é fundamental para se conseguir um limite superior para o problema original. Nesse ponto, pode-se utilizar um algoritmo heurístico conveniente para acelerar a convergência da Relaxação Lagrangeana. Quanto ao limite inferior, pode-se modificar vários parâmetros do procedimento de subgradientes padrão para conseguir melhorá-lo.

Entre os diversos testes de parada do algoritmo, pode-se usar os seguintes:

1. subgradiente nulo;

2. $f_{sup} = f_{inf}$;

3. $f_{sup} - f_{inf} < \epsilon$;

4. o tamanho de passo é muito pequeno para que haja uma alteração significativa no valor de $f_{inf}$ de uma iteração para outra;

5. o valor de $f_{inf}$ não apresenta melhorias significativas durante um certo número de iterações consecutivas;

6. um número máximo de iterações é atingido.

Ao final da heurística, $f_{sup}$ é o valor da melhor solução viável encontrada e $f_{inf}$ é o valor do melhor limite inferior encontrado para o valor ótimo do problema original. A qualidade da solução final obtida pode ser avaliada pela expressão:

$$\frac{f_{sup} - f_{inf}}{f_{inf}}$$

# Capítulo 6

# Relaxação Lagrangeana para a Formulação Canônica

A relaxação lagrangeana é um método bastante conhecido para tratar problemas de programação inteira e, particularmente, há vários trabalhos na literatura que aplicam este método à formulação clássica do *SPLP*. Entretanto, ao nosso conhecimento, ainda não existem experiências similares com a formulação canônica. Nesta seção, apresentamos algumas possibilidades de uso da relaxação lagrangeana para esta formulação.

## 6.1 Dual Lagrangeano

Considere o problema dado por (4.2)-(4.5) e, para $j \in J$, denote por $R_j \subseteq \{2, \ldots, K_j\}$ o subconjunto de restrições em (4.3) que serão relaxadas através de multiplicadores lagrangeanos $u_{jr} \geq 0$, $j \in J$, $r \in R_j$. Seja ainda $\bar{R}_j = \{2, \ldots, K_j\} \setminus R_j$. A função

lagrangeana assim obtida é:

$$\begin{aligned}\psi p(z,y,u) &= \sum_{i\in I} f_i y_i + \sum_{j\in J}\sum_{r=2}^{K_j} (d_{jr}-d_{jr-1})\, z_{jr} + \sum_{j\in J}\sum_{r\in R_j} u_{jr}\left(1 - z_{jr} - \sum_{i:c_{ij}<d_{jr}} y_i\right)\\ &= \sum_{j\in J}\sum_{r\in R_j} u_{jr} + \sum_{i\in I} f_i y_i - \sum_{j\in J}\sum_{r\in R_j}\sum_{i:c_{ij}<d_{jr}} u_{jr} y_i\\ &\quad + \sum_{j\in J}\sum_{r\in R_j} (d_{jr}-d_{jr-1}-u_{jr})\, z_{jr} + \sum_{j\in J}\sum_{r\in \overline{R}_j} (d_{jr}-d_{jr-1})\, z_{jr}.\end{aligned}$$

Claramente vemos que

$$\sum_{j\in J}\sum_{r\in R_j}\sum_{i:c_{ij}<d_{jr}} u_{jr} y_i = \sum_{i\in I}\left(\sum_{j\in J}\sum_{r\in R_j:c_{ij}<d_{jr}} u_{jr}\right) y_i.$$

Então, temos que

$$\begin{aligned}\psi p(z,y,u) &= \sum_{j\in J}\sum_{r\in R_j} u_{jr} + \sum_{i\in I}\left(f_i - \sum_{j\in J}\sum_{r\in R_j:c_{ij}<d_{jr}} u_{jr}\right) y_i\\ &\quad + \sum_{j\in J}\sum_{r\in R_j} (d_{jr}-d_{jr-1}-u_{jr})\, z_{jr} + \sum_{j\in J}\sum_{r\in \overline{R}_j} (d_{jr}-d_{jr-1})\, z_{jr}.\end{aligned}$$

Dado $u \geq 0$, definimos a seguinte relaxação lagrangeana da formulação canônica:

$$(RL) \quad LP(u) = \min \ \psi p(z,y,u) \tag{6.1}$$

$$\text{sujeito a: } z_{jr} + \sum_{i:\ c_{ij}<d_{jr}} y_i \geq 1, \forall j \in J, \forall r \in \bar{R}_j, \tag{6.2}$$

$$z_{jr} \geq z_{j(r+1)}, \forall j \in J, \forall r \in S_j, \tag{6.3}$$

$$y_i \in \{0,1\}, \forall i \in I, \tag{6.4}$$

$$z_{jr} \in \{0,1\}, \forall j \in J, \forall r = 2,\ldots,K_j. \tag{6.5}$$

onde acrescentamos algumas das desigualdades válidas anunciadas na Proposição 13, aquelas indexadas por $j \in J$ e $r \in S_j \subseteq \{2,\ldots,K_j-1\}$.

Sabemos que $LP(u)$, para qualquer $u \geq 0$, é um limite inferior para o *SPLP*. O vetor $u^*$ que fornece o melhor limite inferior $LP(u^*)$ é obtido pela resolução do seguinte dual lagrangeano:

$$(DL) \quad LP(u^*) = \max_{u\geq 0} LP(u)$$

Utilizando o método de subgradientes, como mostrado na Seção 5.2, podemos observar que a cada iteração $k$ do método, dado o vetor $u^k \geq 0$, encontramos uma solução $(z^k, y^k)$ para $L(u^k)$ e atualizamos

$$u_{jr}^{k+1} = \max\{0, u_{jr}^k + \theta^k s_{jr}^k\},$$

onde

$$s_{jr}^k = 1 - z_{jr}^k - \sum_{i \in I: c_{ij} < d_{jr}} y_i^k,$$

a menos que esse valor seja negativo e $u_{jr}^k = 0$, quando fazemos diretamente $s_{jr}^k = 0$, e

$$\theta^k = \pi(1,05LS - L(u^k))/\left\|s^k\right\|_2^2.$$

## 6.2 Diferentes Relaxações Lagrangeanas

Observamos que diferentes relaxações podem ser obtidas em (6.1)-(6.5), dependendo de como os conjuntos $R_j$ e $S_j$ são escolhidos. Estas escolhas interferem, por um lado, na complexidade do problema $(RL)$ e, por outro, na qualidade do limite $L(u^*)$ obtido ou mesmo no número de iterações necessárias para encontrar $u^*$. Estas questões são, portanto, fundamentais. A seguir, enumeramos as alternativas que destacamos neste trabalho, dentre várias outras possíveis.

**Relaxação lagrangeana completa:** Neste caso, relaxamos todas as restrições em (4.3) e não acrescentamos as restrições de ordem entre as varíaveis $z$. Em outros termos, tomamos $R_j = \{2, \ldots, K_j\}$, e $\overline{R_j} = S_j = \varnothing$, para todo $j \in J$. Aqui, $L(u)$ é facilmente calculado como:

$$L(u) = \sum_{j \in J} \sum_{r \in R_j} u_{jr} + \sum_{i \in I} \left( f_i - \sum_{j \in J} \sum_{r \in R_j: c_{ij} < d_{jr}} u_{jr} \right)^- + \sum_{j \in J} \sum_{r \in R_j} (d_{jr} - d_{jr-1} - u_{jr})^-$$

onde $(a)^- = \min\{0, a\}$.

**Relaxação lagrangeana completa com restrições de ordem:** Novamente, para cada $j \in J$, tomamos $R_j = \{2, \ldots, K_j\}$ e $\overline{R}_j = \varnothing$; mas agora tomamos $S_j = \{2, \ldots, K_j - 1\}$. Mesmo com o acréscimo das restrições de ordem, o valor de $L(u)$ ainda pode ser calculado em tempo polinomial como:

$$L(u) = \sum_{j \in J} \sum_{r \in R_j} u_{jr} + \sum_{i \in I} \left( f_i - \sum_{j \in J} \sum_{r : c_{ij} < d_{jr}} u_{jr} \right)^{-} + \sum_{j \in J} \min_{s \in R_j} \sum_{r=2..s} (d_{jr} - d_{jr-1} - u_{jr})$$

**Relaxação lagrangeana parcial baseada na relaxação linear:** Para definirmos os conjuntos $R_j$, $j \in J$, primeiro resolvemos a relaxação linear de *(PC)*, usando o procedimento Fronteira Variável Baseado na *RL*, descrito na Subseção 4.3.1. Ao final do processo, algumas restrições de (4.3) foram introduzidas no modelo linear e outras ficaram de fora. Estão últimas vão constituir $R_j$, $j \in J$, enquanto as primeiras formam $\overline{R}_j$, $j \in J$. Neste caso, consideramos ainda $S_j = \varnothing$.

## 6.3 Heurísticas Lagrangeanas

A partir de uma solução ótima $(y^u, z^u)$ para $L(u)$, podemos procurar construir soluções viáveis para o *SPLP* de várias formas. Particularmente, podemos elaborar tais estratégias considerando as heurísticas *ADD/DROP* apresentadas na Seção 3.3.2. Estes procedimentos podem ser incorporados ao algoritmo de subgradientes, atualizando a fórmula de cálculo dos passos e, possivelmente, acelerando a convergência.

Dentre as várias possibilidades para elaboração de uma heurística lagrangeana, destacamos as duas abaixo:

**Heurística Cobertura de Conjuntos:** Conforme sugerido em [44], dada a solução $(y^u, z^u)$, retiramos todas as restrições do problema já satisfeitas, ou seja, tais que

$$z^u_{jr} + \sum_{i : c_{ij} < d_{jr}} y^u_i \geq 1,$$

e eliminamos todas as variáveis $(z, y)$ com valor 1 em $(y^u, z^u)$. O subproblema definido pelas variáveis e restrições remanescentes é resolvido por uma heurística qualquer, gerando solução $(y^*, z^*)$. A solução viável para o problema completo será dada pela composição de $(y^u, z^u)$ com $(y^*, z^*)$. Podemos ainda tentar melhorar esta solução, anulando alguma variável fixada em 1 em $(y^u, z^u)$.

**Heurística ADD:** Consideramos abertas as facilidades em $K = \{i \in I : y_i^u = 1\}$ e, a partir desse ponto, procuramos melhorar a solução, aplicando iterativamente a seguinte heurística *ADD*: enquanto $Z(K \cup \{i^*\}) = \min_{i \in I-K} Z(K \cup \{i\}) < Z(K)$ faça $K = K \cup \{i^*\}$. Dentre as várias heurísticas que testamos, esta foi a que obteve melhor resultado, quando comparamos esforço computacional e qualidade da solução encontrada.

## 6.4 Fixação de Variável

Logo que um limite superior $LS$ para $SPLP$ for conhecido (por exemplo, o valor de uma solução viável), também é possível utilizar a relaxação lagrangeana para fixar variáveis. Uma solução viável $(y, z)$ melhor deve satisfazer

$$L(u) \leq Z(y, z) < LS,$$

onde $Z()$ fornece o valor da solução.

Defina então

$$\overline{f_i} = f_i - \sum_{j \in J} \sum_{r \in R_j : c_{ij} < d_{jr}} u_{jr},$$

$$g_{jr} = d_{jr} - d_{jr-1} - u_{jr}$$

e considere os seguintes conjuntos

$$\begin{aligned} I_1 &= \left\{i \in I : \overline{f_i} > 0\right\}, \\ I_0 &= \left\{i \in I : \overline{f_i} < 0\right\}, \\ JR_1 &= \{(j, r) : j \in J, r \in R_j, g_{jr} > 0\}, \\ JR_0 &= \{(j, r) : j \in J, r \in R_j, g_{jr} < 0\}. \end{aligned}$$

Obtemos as seguintes formas para fixar variáveis:

**Proposição 19** *: As seguintes fixações de variáveis ocorrem em uma solução viável de valor menor que $LS$.*

- *Se* $t \in I_1$ *e*

$$\sum_{j \in J} \sum_{r \in R_j} u_{jr} + \sum_{i \in I_0} \left(\overline{f_i}\right) + \sum_{(j,r) \in JR_0} (g_{jr}) + \left(\overline{f_i}\right) \geq LS$$

*então* $y_t = 0$.

- *Se* $t \in I_0$ *e*

$$\sum_{j \in J} \sum_{r \in R_j} u_{jr} + \sum_{i \in I_0 - \{t\}} \left(\overline{f_i}\right) \geq LS$$

*então* $y_t = 1$ *e, conseguentemente,* $z_{jr} = 0$ *para todo* $j \in J$ *e* $r \in R_j$ *tais que* $c_{tj} < d_{jr}$.

- *Se* $(j', r') \in JR_1$ *e*

$$\sum_{j \in J} \sum_{r \in R_j} u_{jr} + \sum_{i \in I_0} \left(\overline{f_i}\right) + \sum_{(j,r) \in JR_0} (g_{jr}) + (g_{j'r'}) \geq LS$$

*então* $z_{j'r'} = 0$.

- *Se* $(j', r') \in JR_0$ *e*

$$\sum_{j \in J} \sum_{r \in R_j} u_{jr} + \sum_{i \in I_0} \left(\overline{f_i}\right) + \sum_{(j,r) \in JR_0 - (j',r')} (g_{jr}) \geq LS$$

*então* $z_{j'r'} = 1$ *e, por conseguinte,* $y_i = 0$ *para todo* $i \in I$ *tal que* $c_{ij'} < d_{j'r'}$.

Observamos que essa forma de fixação não mostrou resultados significativos para os problemas-teste não-euclideanos. Não descartamos, porém, sua eficácia para outros tipos de instâncias.

Destacamos ainda que, para as relaxações completa com restrições de ordem e parcial, podemos derivar condições suficientes para fixação de variáveis mais forte que aquelas apresentadas na Proposição 19, entretanto de verificação computacional mais difícil.

## 6.5 Resultados Computacionais

Aqui apresentamos os resultados computacionais com as relaxações e heurísticas lagrangeanas propostas. Implementamos o método de subgradiente para resolver cada uma das três relaxações apresentadas na Seção 6.2, acoplado a heurística ADD da Seção 6.3. A codificação foi feita em Mosel, a linguagem do software XPress, que foi utilizado para resolver os problemas lineares.

Para cada algoritmo limitamos o tempo de execução em 1 hora, ao final do qual ele é interrompido, caso ainda não tenha satisfeito seu critério de parada. Em qualquer caso, ele devolve o melhor limite inferior ($LI$) e superior ($LS$) encontrados. Calculamos, então, o *gap* como 100*($LS$-$LI$)/$LI$. Registramos ainda o tempo gasto em segundos.

Na Tabela 6.1, mostramos os resultados obtidos apenas pela relaxação completa com restrições de ordem ($RC/RO$) e pela relaxação parcial baseada na relaxação linear ($RP/RL$), uma vez que a relaxação completa foi sempre inferior à $RC/RO$ com respeito à qualidade do limite encontrado. Como critérios de parada do método de subgradientes, usamos, além do limite de tempo, $\left\|s^k\right\|_2 \leq 0.00001$, $LS - L(u^k) \leq 0.005$ ou $\pi \leq 0.005$, sendo que $\pi$ é dividido pela metada sempre que se passam 10 iterações sem aumento do melhor limite inferior.

Pelos números da tabela, observamos que os dois algoritmos conseguiram fechar o gap para todas as instâncias do grupo 1 e algumas do grupo 2. Nas demais instâncias podemos observar um comportamento distinto das duas relaxações. O algoritmo RC/RL obteve melhores limites inferiores e superiores em todas estas outras instâncias. Além disso, como conhecemos o ótimo, podemos perceber que a heurística lagrangeana o determina na maioria dos casos, mesmo com gap diferente de zero. Quando o gap é superior a 1%, normalmente o algoritmo pára por exceder o tempo. Por outro lado, o algoritmo $RP/RO$ termina, em geral, por atingir um valor pequeno para $\pi$, sem conseguir reduzir o gap de forma tão efetiva. Isso se deve à qualidade inferior do limite gerado por essa relaxação lagrangeana.

| Prob | Opt | RC/RL | | | | RP/RO | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Tempo | LS | LI | gap% | Tempo | LS | LI | gap% |
| 1:50 | 488 | 0.16 | 488 | 488 | 0 | 0.16 | 488 | 488 | 0 |
| 1:100 | 914 | 1.10 | 914 | 914 | 0 | 1.07 | 914 | 914 | 0 |
| 1:150 | 1438 | 3.64 | 1438 | 1438 | 0 | 3.56 | 1438 | 1438 | 0 |
| 1:200 | 1860 | 8.41 | 1860 | 1860 | 0 | 8.36 | 1860 | 1860 | 0 |
| 1:400 | 3504 | 64.76 | 3504 | 3504 | 0 | 64.43 | 3504 | 3504 | 0 |
| 1:600 | 5054 | 216.84 | 5054 | 5054 | 0 | 215.87 | 5054 | 5054 | 0 |
| 1:800 | 6621 | 538.00 | 6621 | 6621 | 0 | 517.48 | 6621 | 6621 | 0 |
| 2:50 | 2864 | 0.16 | 2864 | 2864 | 0 | 0.19 | 2864 | 2864 | 0 |
| 2:100 | 4618 | 1.08 | 4618 | 4618 | 0 | 1.10 | 4618 | 4618 | 0 |
| 2:150 | 5536 | 77.35 | 5536 | 5531.75 | 0.08 | 16.73 | 5542 | 5531.75 | 0.18 |
| 2:200 | 6495 | 498.38 | 6501 | 6494 | 0.11 | 37.90 | 6501 | 6494 | 0.11 |
| 3:50 | 5701 | 3.91 | 5701 | 5673.34 | 0.49 | 46.41 | 5701 | 5650.33 | 0.89 |
| 3:100 | 9243 | 3700.22 | 9243 | 9184.25 | 0.64 | 377.43 | 9312 | 8658.25 | 7.02 |
| 3:150 | 11314 | 5558.39 | 11314 | 11173.7 | 1.24 | 1238.78 | 11721 | 10657.6 | 9.07 |
| 3:200 | 13043 | 5504.70 | 13043 | 12875 | 1.29 | 2804.11 | 13043 | 12355.4 | 5.27 |
| 4:50 | 7290 | 48.23 | 7299 | 7273.16 | 0.35 | 49.71 | 7299 | 7273.16 | 0.35 |
| 4:100 | 11314 | 3794.81 | 11331 | 11135.3 | 1.73 | 415.36 | 11337 | 10430.3 | 8.00 |
| 4:150 | 13785 | 4970.16 | 13785 | 13675.6 | 0.79 | 1231.20 | 14021 | 12876.8 | 8.16 |
| 5:50 | 9214 | 937.66 | 9214 | 9118.49 | 1.04 | 47.23 | 9214 | 8667.16 | 5.93 |
| 5:100 | 13685 | 3608.22 | 13689 | 13520.1 | 1.23 | 437.60 | 13788 | 12674.5 | 8.08 |
| 5:150 | 17713 | 6085.04 | 17790 | 16549.6 | 6.97 | 1318.15 | 17919 | 15975.7 | 10.84 |

Tabela 6.1: Resultados computacionais com procedimentos baseados em relaxação lagrangeana

# Capítulo 7

# Conclusão

Nesta dissertação trabalhamos com o problema de localização, onde concentramos nossa atenção no Problema de Localização Simples (*Simple Plant Problem Location - SPLP*), uma das principais variações do problema geral. Exploramos sua formulação clássica e canônica para desenvolver algoritmos de vários tipos, com a finalidade de obter boas soluções ou diminuir o tempo computacional de resolução de algumas instâncias existentes na literatura.

Revisitamos a formulação clássica, para a qual propomos uma projeção do seu conjunto viável. Com isso, conseguimos reduzir a quantidade de variáveis e restrições utilizada na definição do modelo. Além de possuir dimensão menor do que a original, esta formulação elimina certas soluções equivalentes, o que é interessante quando trabalhamos com o método de planos de corte. Com a utilização do software de programação matemática XPress mostramos que tal formulação revisitada é mais eficiente que a clássica original: o XPress resolveu as instâncias testadas em menos tempo.

Considerando a formulação clássica, desenvolvemos algoritmos heurísticos e *branch-and-bound* baseados essencialmente nos testes de redução. Os resultados computacionais obtidos mostram que a qualidade das soluções encontradas pelas nossas heurísticas é bastante similar ao algoritmo de Pereira[38], detentor de excelente desempenho para problemas não-euclideanos. Particularmente a nossa heurística *HRA* encontra com maior

freqüência as melhores soluções, o que ocorreu em 10 problemas testados. Quando comparamos o esforço computacional, cada um das nossas heurísticas consegue reduzir o tempo médio pela metade, sendo que a heurística *HMDR* mostra uma maior eficiência principalmente nos problemas dos grupos 3, 4 e 5 onde os custos fixos são elevados. Em cada um dos algoritmos heurísticos usamos uma forma iterativa de recalcular os valores de $\Delta_i()$, que constituiu um elemento importante para fornecer a superioridade em termos de tempo dos nossos algoritmos, além das boas decisões que tais heurísticas tomam.

Estudando a formulação canônica proposta recentemente por Labbé e Marín[34], verificamos uma imprecisão no modelo e propomos o acréscimo de uma restrição para obter sua corretude, que foi formalmente demonstrada.

Quando, para cada demanda, a quantidade de facilidades com valores de custo de transporte distintos é pequena, a formulação é bastante compacta. De outra forma, dada a sua estrutura, podemos reduzir sua dimensão aplicando procedimentos iterativos. Para descrevê-los, introduzimos a definição de um subproblema paramétrico. A idéia desses procedimentos iterativos foi sugerida por Labbé e Marín[34], que não mostraram sua corretude. Apresentamos condições suficientes para a corretude de tais procedimentos e mostramos como usá-los para obter limites inferiores e superiores. Fazendo uma análise dos resultados computacionais obtidos, podemos observar que tais procedimentos não são apropriados para instâncias de custos fixos altos e dimensão $n = m > 150$, mas conseguem tratar adequadamente outros casos.

Melhoramos ainda mais o tempo computacional quando desenvolvemos um algoritmo *branch-and-bound* que utiliza os procedimentos iterativos. A análise mostra que os tempos obtidos pelos algoritmos B&B são melhores em comparação aos obtidos apenas pela aplicação dos procedimentos, mas que para as instâncias de custos fixos altos e dimensão $n = m > 150$ ainda continuam proibitivos. Parte dos experiências com os procedimentos iterativos e estes algoritmos B&B fazem parte do trabalho aceito no *XIV Congresso Latino Ibero Americano de Investigación de Operaciones*, 2008[14].

Com a expectativa de melhorar tais tempos e a finalidade de avaliar uma nova abor-

dagem ainda não existente na literatura para a formulação canônica, fizemos um estudo com a relaxação lagrangeana aplicada à nova formulação. Os resultados compõem um artigo aceito no *XL Simpósio Brasileiro de Pesquisa Operacional*, 2008[15]. Nos testes computacionais realizados, pudemos observar que a aplicação de um algoritmo utilizando apenas a relaxação lagrangeana não é aconselhável, pelo menos para os problemas-teste usados. Todavia, essa abordagem pode ser utilizada em um algoritmo do tipo *branch-and-bound*. Nossa expectativa é que a qualidade dos limites inferiores obtidos pela relaxação lagrangeana sejam similares aos obtidos pelos procedimentos iterativos, mas que sejam obtidos em melhores tempos computacionais. Os procedimentos iterativos, como forma de calcular limites inferiores, obtiveram maior eficiência nas instâncias mais difíceis, quando embutidos em algoritmos B&B. O mesmo talvez ocorra com a relaxação lagrangeana. Este é um estudo que merece atenção.

Como comentário final, devemos mencionar que os algoritmos testados encontram maior dificuldade à medida que os custos fixos crescem. Acreditamos que isto pode ser explicado se observarmos que, quanto maior os custos fixos, mais difícil é decidir qual dentre duas facilidades deve ser aberta na solução ótima. Quando os custos fixos são pequenos, comparados aos de transporte, a decisão pode ser tomada em favor de uma ou outra facilidade sem gerar grande impacto no valor do custo final. Especializações dos algoritmos para custos fixos altos podem ser pensadas.

# Referências Bibliográficas


[1] Akinc, U., Khumawala, B. M. “An efficient branch and bound algorithm for the capacited warehouse location problem”. *Management Science*, v.23, n.6, pp. 585-594, 1977.

[2] Al-Sultan, K. S., Al-Fawzan, M. A. "A tabu search approach to the uncapacited facility location problem". *Annals of Operations Research*, v. 86, pp. 91-103, 1999.

[3] Alves, M. L., Almeida, M. T. "Simulated Annealing Algorithm for the Simple Plant Location Problem: A Computational Study". *Revista Investigaćão Operacional*, v. 12, 1992.

[4] Balinski, M. L. "On finding integer solutions to linear programs". *Procedings of IBM Scientific Symposium on Combinatorial Problems*, pp 225-248, 1966.

[5] Barahona, F., Anbil, R. "The volume algorithm: producing primal solutions with a subgradient method". *Mathematical Programming*, Vol. 87, pp. 385-399, 2000.

[6] Barahona F., Chudak F. "Near-optimal solutions to large scale facility location problems". *Discrete Optimization*, vol. 2, pp 35-50, 2005.

[7] Bartezzaghi, E., Colorni, A., Palermo, P. C. "A tree search algorithm for plant location problems". *European Journal of Operational Research*. Vol. 7, pp 371-379, 1981.

[8] Beasley, J. E. "Lagrangian heuristics for location problems". *European Journal of Operational Research*, Vol. 65, pp. 383-399, 1993.

[9] Beasley, J. E. Lagrangean relaxation, em Reeves, C. R. (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Sci. Publications, 243-303, 1993.

[10] Beasley, J. E. "OR-Library: Distribution test problems by eletronic mail". *Journal of Operational Research Society*, v. 41, pp. 1069-1072, 1990.

[11] Bornstein, C.T. & Azlan, H.B. (1998). The Use of Reduction Tests and Simulated Annealing for the Capacitated Plant Location Problem. *Location Sci.*, 6, 67-81.

[12] Brandeau, M. L., Chiu, S. S. "Overview of Representative Problems in Location Research". *Management Science*, v. 35, n. 6, pp. 645 674, 1989.

[13] Campêlo, M. B., Bornstein, C. T. "ADD/DROP procedures for the capacited plant location problem". *Pesquisa Operacional*, v. 24, n. 1, p. 151-162, 2004.

[14] Campelo, M. B., Dias, Fabio C. S., Martine Labbé. "Algoritmo para problema de localização não capacitado baseados na formulação canônica". *XIV Congresso Latino Ibero Americano de Investigación de Operaciones*, 2008.

[15] Campelo, M. B., Dias, Fabio C. S., Martine Labbé. "Relaxações para a formulação canônica do problema de localização de facilidades não capacitado". *XL Simpósio Brasileiro de Pesquisa Operacional*, 2008.

[16] Cornuejols, G., Nemhauser, G., Wolsey, L. A. A canonical representation of simple plant location problems and its applications, *SIAM J. Alg. Disc. Meth*, 1(3): 261-272, 1980.

[17] Daskin, M. S. *Network and Discrete Location.* John Wiley and Sons, New York, 1995.

[18] Domschke, W., Drexl, A. "Add-heuristics starting procedures for capacitated plant location models". *European Journal of Operational Research*, Vol. 21, pp. 47-53, 1985.

[19] Feldman, E., Lehrer, F. A., Ray, T. L. "Warehouse location under continuous economies of scale". *Management Science*, v. 12, pp. 670-684, 1966.

[20] Galvão, R. D., Raggi, L. A. "Method for solving to opttimality uncapacitated location problems". *Annals of Operations Research*, v. 18, pp. 225-244, 1989.

[21] Garey, M. R., Johnson, D. S. *Computers and Intractibility: a guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[22] Ghosh, D. "Neighborhood search heuristics for the uncapacitated facility location problem". *European J. Operational Research*, vol. 150, pp. 150-162, 2003.

[23] Goldengorin, B., Ghosh, D. e Sierksma, G. "Branch and peg algorithms for the simple plant location problem". *Computers & Operations Research*, v.30, pp. 967-981, 2003.

[24] Goldengorin, B., Tijssen, G., Ghosh, D. e Sierksma, G. "Solving the simple plant location problem using a data correction approach". *J. Global Optimization*, vol. 25, pp. 377-406, 2003.

[25] Graciano Sá. "Branch-and-bound and approximate solutions to the capacitated plant-location problem". *Operations Research*, vol. 17, No. 6, pp. 1005-1016, 1969.

[26] Held, M., Karp, R. M. "The traveling salesman problema and minimum spanning trees". *Mathematical Programming*, 18: 1138-1162, 1970.

[27] Held, M., Karp, R. M. "The traveling salesman problema and minimum spanning trees: part II". *Mathematical Programming*, 1: 6-25, 1970.

[28] Jacobsen, S. K. "Heuristics for the capacited plant location model". *European J. of Operational Research*, v. 12, pp. 253-261, 1983.

[29] Kratica, J., Tosie, D., Filipović, V., Ljubić, I. "Solving the Simple Plant Location Problem by Generic Algorithm". *RAIRO Operations Research*, v. 35, pp. 127-142, 2001.

[30] Kuehn, A. A., Hamburguer, M. J. "A heuristic program for locating warehouses". *Management Science*, v. 9, n. 4, pp. 643-666, 1963.

[31] Krarup, J., Pruzan, P. M. "The simple plant location problem: survey and synthesis". *European Journal of Operational Research*, v. 12, pp. 36-81, 1983.

[32] Lorena, L. A. N., Senne E. L. F. "Improving traditional subgradient scheme for Lagrangean relaxation: an application to location problems". *International Journal of Mathematical Algorithms*, Vol. 1, pp.133-151,1999.

[33] Manne, A. S. "Plant location under economics of scale decentralization and computation". *Management Science*, v. 12, n. 2, pp.213-235, 1964.

[34] Labbé, Martine, Marín, Alfredo. "Old and new formulations uncapacitated network location problems". International Network Optimization Conference - INOC, 2005.

[35] Mateus, G. R., Bornstein, C.T. "Dominance criteria for the capacited warehouse location problem." *Journal of the Operational Research Society*, n. 42, pp. 145-149, 1991.

[36] Mateus, G. R., Carvalho, J. C. P. "O Problema de Localização não-capacitado: modelos e algoritmos". Investigación Operativa, v. 2 n. 3, pp.297-317, 1992.

[37] Michel, L., Hentenryck, P. Van. *A Simple Tabu Search for Warehouse Location.* Technical Report CS-02-05, Brown University, April 2002.

[38] Pereira, A. B. "Um algoritmo para o Problema de Localização Não Capacitado baseado em Testes de Redução e Heurísticas ADD/DROP", Tese de mestrado, 2002.

[39] Pinto Júnior, J., Dias, F., Campêlo, M. Um estudo de critérios add/drop para o problema de localização de facilidades não-capacitado. *Simpósio Brasileiro de Pesquisa Operacional*, 2005.

[40] Resende, M. e Werneck, R. “A Hybrid Multistart Heuristic for the Uncapacitated Facility Location Problem“. *Optimization Online Digest*, 2003.

[41] Rockafellar, R. *Convex Analysis*, Princetm University Press, 1997.

[42] Simão, H. P., Thizy, J. M. "A dual simplex algorithm for the canonical representation of the uncapacitated facility location problem". *Operations Research North-Holland*, v. 8, pp. 279-286, 1989.

[43] Wolsey, L.A. (1983). Fundamental Properties of Certain Discrete Location Problems. **In**: *Locational Analysis of Public Facilities* [edited by J.-F. Thisse & H.G. Zoller],North-Holland, 331-355.

[44] Wolsey, L. A. *Integer Programming*, John Wiley & Sons, 1998.