PAGANINI BARCELLOS DE OLIVEIRA

PROBLEMA DE LOCALIZAÇÃO EM DOIS NÍVEIS DE FACILIDADES NÃO CAPACITADAS: ALGORITMOS EXATOS E HEURÍSTICOS

Belo Horizonte
09 de dezembro de 2014

Universidade Federal de Minas Gerais
Escola de Engenharia
Programa de Pós-Graduação em Engenharia de Produção

PROBLEMA DE LOCALIZAÇÃO EM DOIS NÍVEIS DE FACILIDADES NÃO CAPACITADAS: ALGORITMOS EXATOS E HEURÍSTICOS

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia de Produção da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Engenharia de Produção.

PAGANINI BARCELLOS DE OLIVEIRA

Belo Horizonte
09 de dezembro de 2014

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Problema de Localização em dois Níveis de Facilidades não Capacitadas: Algoritmos exatos e heurísticos

PAGANINI BARCELLOS DE OLIVEIRA

Dissertação defendida e aprovada pela banca examinadora constituída por:

Dr. Ricardo Saraiva de Camargo – Orientador
Universidade Federal de Minas Gerais

Dr. Alexandre Xavier Martins – Co-orientador
Universidade Federal de Ouro Preto

Dr. Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto

Dr. Gilberto de Miranda Junior
Universidade Federal de Minas Gerais

Belo Horizonte, 09 de dezembro de 2014

Resumo

Este trabalho investiga a aplicação de algoritmos exatos e heurísticos baseados no método de Decomposição de Benders e na metaheurística GRASP combinada com as técnicas de reconexão por caminhos (Path Relinking) e de perturbações orientadas das soluções, respectivamente, na resolução do Problema de Localização em dois Níveis de Facilidades não Capacitadas. Trata-se de um problema clássico de Otimização de Sistemas de Grande Porte que tem grande aplicabilidade nos mais variados modelos de sistemas logísticos existentes. O grande desafio de problemas dessa natureza é estabelecer quais facilidades serão instaladas de forma a garantir uma configuração de custo mínimo viável que implique na eficiência máxima no atendimento dos clientes de uma rede. O problema pode ser modelado e representado de inúmeras formas, sendo que em todas elas a ideia é estabelecer a quantidade de fluxo que sai de uma facilidade de primeiro nível, passa por outra de segundo nível e atende um cliente final. Para demonstrar a eficiência dos métodos propostos são feitas análises e comparações das técnicas em relação ao tempo computacional de resolução e a qualidade da solução.

Palavra-chaves: Problema de Localização em dois Níveis de Facilidades não Capacitadas; Método de Decomposição de Benders; Otimização de Sistemas de Grande Porte; GRASP; Reconexão por Caminhos.

Abstract

This paper investigates the application of exact and heuristic algorithms based on Benders decomposition method and on the GRASP metaheuristic combined with the techniques of reconnection by ways (Path Relinking) and targeted perturbations, respectively, to solve the Two Level Uncapacitated Facility Location Problem. This is a problem of the field of Optimization of Large Scale Systems that has wide applicability in several models of existing logistics systems. The challenge of such problems is to select which facilities will be installed that implies in the maximum efficiency to meet the demands of the customers through network. The problem can be modeled and represented in many forms, and in all of them the idea is to establish the amount of flow that leaves a facility on the first level, goes through another facility of second level, to supply the customer demands on the lower level. To demonstrate the efficiency of the proposed methods, analyzes and comparisons of the techniques are made regarding the computational time of resolution and solution quality.

Keywords: Two Level Uncapacitated Facility Location Problem; Benders decomposition; Optimization of Large Scale Systems; GRASP; Path Relinking.

Em especial ao meu pai e minha mãe;
Dedico também a toda minha família e amigos;

Agradecimentos

Primeiramente queria agradecer a Deus por tudo aquilo que ele me proporciona.

Ao meu pai Verdi Antonio de Oliveira, minha mãe Ana Lúcia Barcellos de Oliveira e a Paula Valamiel de Oliveira Vieira, pelo carinho, atenção e amor.

Ao meus orientadores Dr. Ricardo Saraiva de Camargo e Dr. Alexandre Xavier Martins, pela paciência, incentivo e confiança em meu trabalho.

Aos professores Dr. Ivan Contreras e Dr. Jean-Francois Cordeau, pela parceria e contribuição técnica.

À toda minha família e amigos por serem a base de meu sucesso.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

1.1 Contextualização

A estrutura hierárquica de muitos sistemas de grande porte comuns em nosso dia a dia tem como característica a existência de dois diferentes níveis de facilidades[1] para o atendimento de clientes em uma rede. A diferença básica entre as facilidades de primeiro e segundo nível está na função que cada uma exerce. A de primeiro nível pode ser vista como uma geradora de recursos, enquanto a de segundo nível funciona como um agente intermediário de distribuição física desses recursos.

Segundo Klose e Drexl (2005) a escolha do local de instalação de facilidades em um sistema de distribuição pode ser vista como uma estratégia que visa principalmente a redução de custos de transporte de muitos tipos de empresas. Nos casos em que a região geográfica que compreende o conjunto de clientes e facilidades é muito extensa (considerar todo um estado, um país, ou até mesmo um continente, por exemplo), o posicionamento das facilidades se torna um fator ainda mais determinante para o sucesso das empresas.

Exemplos reais de sistemas hierárquicos que possuem dois níveis de facilidades podem ser identificados em: serviços de atendimento bancário, no qual é necessário escolher onde instalar as agências bancárias (primeiro nível), postos de atendimento menores com apenas caixas eletrônicos (segundo nível) para atendimento de todos os municípios do país (clientes); serviços de atendimento a saúde, na localização estratégica dos centros de especialidade médicas (primeiro nível), hospitais regionais (segundo nível) e postos de saúde (clientes) de um estado; na produção e distribuição de produtos tais como veículos, alimentos e vestuário, onde é necessário escolher onde construir as fábricas ou usinas, que abastecerão os centros de distribuição e que posteriormente irão

[1] Uma definição para palavra facilidade, no contexto deste trabalho, seria um local físico que dispõe de uma quantidade suficiente de recursos (produtos ou serviços, por exemplo), para atendimento de todos os clientes.

repassar os produtos aos comerciantes varejistas espalhados ao redor do continente; e serviços de transmissão de sinais de TV e internet, na instalação de centrais de geração do sinal, que enviam dados para as torres de recepção/transmissão, que por sua vez, as repassam para as antenas locais dos clientes de uma grande cidade.

Para todos os exemplos citados, o problema consiste em estabelecer quais facilidades estarão ativas dentre todos os locais candidatos (primeiro e segundo níveis), de forma a atender toda a demanda dos clientes com o menor custo total possível (custos de instalação e fluxo de recursos).

Uma das principais variações do problema de localização de facilidades é relacionada ao fato dos mesmos serem problemas capacitados ou não capacitados. Em muitos modelos de problemas de localização de facilidades reais é levado em consideração a capacidade de geração/distribuição de recursos que a facilidade pode suportar, uma vez que nestes casos existe limitação de fluxo que as facilidades podem oferecer. Por outro lado, existem modelos nos quais todas as facilidades de primeiro e segundo nível conseguem atender a todos os clientes ao mesmo tempo, ao que chamamos de problemas não capacitados. Uma particularidade dos problemas não capacitados é que cada cliente será atendido pelo par de facilidades (uma de primeiro e outra de segundo nível) ativas que estiver mais próximo do mesmo, o que implica na configuração de custo mínimo.

No trabalho desenvolvido por Klose e Drexl (2005) é possível encontrar outras variantes para o problema de localização de facilidades, no qual aspectos como diferentes tipos de produtos a serem entregues aos clientes, múltiplos objetivos a serem alcançados, relação entre elementos da rede (facilidades e clientes) não sendo rigidamente hierarquizada, e questões probabilísticas e/ou temporais, podem ser levados em consideração. Neste sentido, nota-se o grande potencial de aplicação prática desses tipos de problemas, o que motiva o desenvolvimento de técnicas eficientes para resolução dos mesmos.

1.2 Motivação

Tendo em vista que Problemas de Grande Porte de Localização em dois Níveis de Facilidades costumam demandar um grande esforço computacional quando resolvidos por meio de resolvedores comerciais disponíveis no mercado, e sabendo do potencial de aplicação prática de problemas da área de telecomunicações, como descrito por Barros e Labbé (1994), pode-se dizer que o desenvolvimento de metodoslogias mais eficientes para solução de tais problemas é justificado.

Um fator ainda mais específico para realização deste trabalho é o fato de haver poucas pesquisas recentes sobre métodos exatos que abordem o Problema de Localização

em dois Níveis de Facilidades não capacitadas, corforme mostrado no levantamento bibliográfico sobre o assunto. O mesmo vale em relação ao desenvolvimento de métodos heurísticos, uma vez que, identificou-se a existência de alguns bons trabalhos sobre algoritmos aproximativos na literatura, porém o uso metaheurísticas ainda foi pouco explorado para o problema em questão.

1.3 Principais Contribuições deste Trabalho

Este trabalho propõe duas diferentes abordagens para a resolução do Problema de Localização em dois Níveis de Facilidades não capacitadas, sendo a primeira a aplicação de um método exato e a segunda, o desenvolvimento de metaheurísticas. As principais contribuições deste trabalho podem ser resumidas em:

- Para o método exato são apresentadas as seguintes técnicas:

  - Método de Decomposição de Benders;

  - Técnicas de aceleração do método de decomposição de Benders:

    ◇ Algoritmos especializados para resolução do Subproblema Dual;

    ◇ Pré-aquecimento (*Hot Start*);

    ◇ Inserção de múltiplos cortes (*Multicut*);

    ◇ Utilização de uma única árvore de *Branch-and-Bound*.

  - Inserção de quatro diferentes cortes: Fechamento de Facilidade, δ-Vizinhança, Pareto-ótimo por inspeção e Pareto-ótimo via subproblema de Papadakos (Papadakos (2008)).

- Para os métodos heurísticos são apresentadas as seguintes técnicas:

  - Algoritmo GPOr:

    ◇ Procedimento aleatório guloso com busca adaptativa (*Greedy Randomized Adaptive Search Procedures* - GRASP);

    ◇ Procedimento de descida em vizinhança variável (*Variable Neighborhood Descent* - VND);

    ◇ Método de reconexão por caminhos (*Path Relinking*).

  - Algoritmo GPR:

    ◇ GRASP;

    ◇ VND;

    ◇ Método de perturbações orientadas de soluções.

1.4 Organização do Trabalho

O restante do texto da dissertação está organizado da seguinte forma: No capítulo 2 é feita uma revisão sobre a literatura relacionada ao Problema de Localização em dois Níveis de Facilidades não capacitadas, bem como o método de decomposição de Benders tradicional e algumas técnicas de aceleração do método. É feita também uma revisão sobre as técnicas heurísticas utilizadas ao longo do trabalho. No capítulo 3 apresentam as definições e formulações adotadas para construção dos algoritmos exatos de resolução do problema, e os seus respectivos resultados computacionais, enquanto no capítulo 4 apresentam as definições e resultados de métodos heurísticos aplicados ao problema. E finalmente, no capítulo 5 são apresentados os comentários finais e as oportunidades de pesquisas futuras.

Capítulo 2

Revisão da Literatura

Este capítulo apresenta a revisão da literatura sobre as técnicas e métodos utilizados ao longo deste trabalho para a resolução do Problema de Localização em dois Níveis de Facilidades não capacitadas.

2.1 O Problema de Localização de Facilidades - FLP

O problema de localização de facilidades, ou também conhecido como FLP (*Facility Location Problem*), foi introduzido na literatura por Weber (1909) tendo como princípio básico o posicionamento de facilidades em um dado espaço geográfico de forma a minimizar a soma de todas as distâncias entre o ponto de demanda e as facilidades instaladas, ou seja, ao mesmo tempo é resolvido um problema de localização e outro de alocação.

O problema de Weber (1909) corresponde a otimização de $\min \sum_{k \in K} w_k d_k(x, y)$, onde $d_k(x, y)$ corresponde à distância do cliente k até a facilidade localizada na posição (x, y) (coordenada cartesiana em um dado espaço geográfico $\mathbb{R} \times \mathbb{R}$, onde $\mathbb{R}$ é o conjunto dos números reais), K é o conjunto de clientes, e w_k são pesos atribuídos a cada uma das distâncias $d_k(x, y)$. O cálculo de $d_k(x, y)$ é dado pela expressão $\sqrt{(x - a_k)^2 + (y - b_k)^2}$, no qual (a_k, b_k) são as coordenadas que representam a localização dos clientes. Segundo Klose e Drexl (2005), uma extensão do problema de Weber (1909) é um modelo de programação inteira mista não linear, que pode ser escrito como:

$$
\begin{aligned}
&\min \sum_{k \in K} \sum_{j\,=\,1}^{p} (w_k c_{kj}) z_{kj} \\
&\text{s.a.:} \sum_{j\,=\,1}^{p} z_{kj} = 1 && \forall k \in K && (2.1)\\
&z_{kj} \in \{0,1\} && k \in K, j = 1...p \\
&x, y \in \mathbb{R}^p
\end{aligned}
$$

No modelo anterior, z_{kj} são variáveis binárias que indicam se a facilidades j está ou não atendendo ao cliente k. Já c_{kj} são as distâncias entre os clientes k e as facilidades j, ponderadas pelos parâmetros de peso w_k. O parâmetro p indica a quantidade de facilidades k a serem abertas, enquanto o $\mathbb{R}^p$ representa o espaço p dimensional de coordenadas (x, y) pertencentes ao conjunto de números reais. As equações (2.1) garantem que cada um dos clientes k sejam atendidos por uma única facilidade j, entre todas as p facilidades ativas.

Um exemplo de um modelo muito citado na literatura sobre o FLP, disponível no trabalho de Klose e Drexl (2005), é o Problema de Localização de facilidades não capacitadas UFLP que pode ser visto nas equações de (2.2) a (2.4). Os nós i e j, respresentam respectivamente clientes e facilidades, enquanto as variáveis y_j indicam se a facilidade está ou não instalada e x_{ij} mostra o quantitativo de fluxo que sai de uma facilidade j para um cliente i. Os custos de alocação de facilidades são representados por f_j, assim como os custos de serviços dos m clientes são representados por c_{ij}.

$$
\begin{aligned}
&\min \sum_{j\,=\,1}^{n} f_j y_j + \sum_{i\,=\,1}^{m} \sum_{j\,=\,1}^{n} c_{ij} x_{ij} && && (2.2)\\
&\text{s.a.:} \sum_{j\,=\,1}^{n} x_{ij} = 1 && i = 1...m && (2.3)\\
&y_j - x_{ij} \geq 0 && i = 1...m;\ j = 1...n && (2.4)\\
&x_{ij} \geq 0 && i = 1...m;\ j = 1...n \\
&y_j \in \{0,1\} && j = 1...n
\end{aligned}
$$

Na função objetivo (2.2), o primeiro somatório é referente ao custo de instalação das facilidades, enquanto o restante da equação calcula o custo de atendimento dos clientes i, pelas facilidades j. As restrições (2.3) garantem que todos os clientes sejam atendidos. As restrições (2.4) eliminam a possibilidade de que uma facilidade não

instalada atenda a um cliente. Já as duas últimas restrições indicam o domínio das variáveis do problema. Em uma formulação mais compacta do FLP, as restrições (2.4) passam a ser modeladas como $\sum_{i=1}^{m} x_{ij} \leq |m|y_j$, onde $|m|$ é igual a quantidade de clientes i.

A Figura (2.1) ilustra uma configuração básica de um UFLP considerando todo o território brasileiro, no qual as estrelas representam os lugares escolhidos para localização das facilidades e os pontos são todos os clientes a serem atendidos na rede. A facilidade que estiver mais próxima do cliente será aquela que o atenderá.

Figura 2.1: Problema de localização de facilidades não capacitadas

Aikens (1985), Klose e Drexl (2005), Farahani et al. (2009) e Verter (2011) reuniram grande parte do estado da arte relacionada ao FLP, e apresentaram alguns modelos e aplicações para o problema. Ao longo dos anos, alguns trabalhos da literatura se destacaram, Kuehn e Hamburger (1963) propuseram a utilização de um método heurístico guloso para o UFLP. Já Efroymson e Ray (1966) fizeram a relaxação linear do UFLP e o resolveram de forma analítica. No trabalho de Krarup e Pruzan (1983) foi apresentada uma revisão sobre os principais trabalhos sobre o UFLP que existiam até então. Erlenkotter (1978) propôs um algoritmo de *Branch-and-Bound* para o UFLP baseado em um procedimento dual com direção de subida, enquanto Goldengorin et al. (2003) propuseram uma versão aprimorada do algoritmo de *Branch-and-Bound*.

Uma variação do FLP que apresenta muitas aplicações práticas é quando se considera facilidades capacitadas (CFLP). Para resolução dos CFLP muitos autores como

Akinc e Khumawala (1977), R.M. (1978) e Mateus e Thizy (1999), por exemplo, propõem o uso de um método de Relaxação Lagrangeana especializado.

No trabalho de Klose e Drexl (2005) é possível obeservar que existem ainda diversos modelos de programação inteira mista relacionados ao FLP, tais como: modelos multiproduto, modelos probabilísticos, modelos dinâmicos e modelos de dois ou mais níveis de facilidades.

Para este trabalho escolheu-se trabalhar como o Problema de Localização em dois Níveis de Facilidades não capacitadas, também conhecido como TLUFLP (*Two Level Uncapacitated Facility Location Problem*) que pode ser visto como uma extensão do problema em um único nível não capacitado (UFLP).

2.2 O Problema de Localização em dois Níveis de Facilidades não Capacitadas - TLUFLP

Em se tratando do problema clássico para o TLUFLP, em meados dos anos sessenta, os trabalhos desenvolvidos pelos pesquisadores Balinski (1964) e Manne (1964) citados por Kaufman et al. (1977), resultaram em um modelo básico para o problema de localização de facilidades dos quais não era considerada a restrição de capacidade de cada uma das facilidades. Nesta mesma época pesquisas relacionadas ao problema capacitado estavam sendo desenvolvidas em paralelo.

As características principais do TLUFLP são determinar pontos de abertura de facilidades de primeiro e segundo nível que resultem na mínima distância/custo de atendimento dos clientes.

A Figura (2.2) ilustra a configuração de uma solução para o problema de localização de facilidades em dois níveis no qual existem três candidatos a facilidade de primeiro nível, cinco candidatos de segundo nível e nove clientes a serem atendidos. Cada cliente é atendido por um par de facilidades ativas, no qual as setas representam fluxo de custo mínimo desta configuração. Por se tratar de um problema não capacitado, não faz sentido que um cliente seja atendido por mais de uma facilidade de primeiro ou segundo nível.

Segundo Aardal et al. (1996) pouco se sabe sobre as propriedades estruturais dos TLUFLP, exceto o fato de se tratar de um problema do tipo NP-difícil[2] e que existiam poucos trabalhos na literatura sobre o TLUFLP até o início da década de noventa.

Os principais trabalhos sobre o TLUFLP podem ser descritos a começar por Kaufman et al. (1977), que desenvolveram um algoritmo de *Branch-and-Bound* baseado em

[2]A medida que se aumenta a dimensão do problema ele pode deixar de ser resolvido em tempo hábil

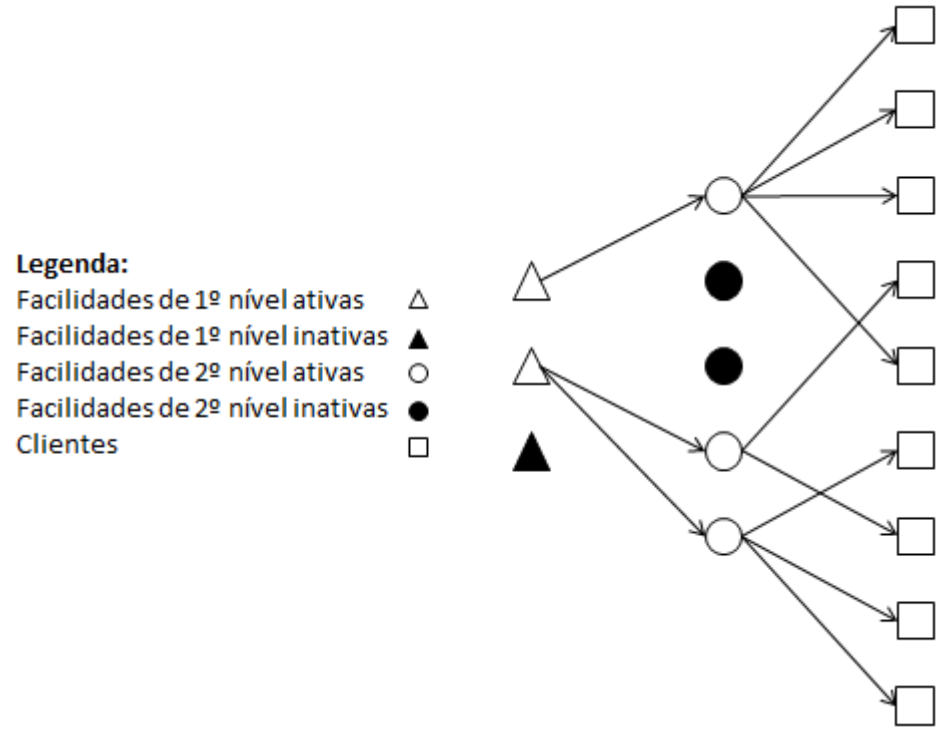

Figura 2.2: Representação esquemática de uma solução para o TLUFLP

um algoritmo para o UFLP proposto por Efroymson e Ray (1966), seguido pela elaboração de um método *Branch-and-Bound*, utilizando um procedimento dual com direção de subida (seguindo os mesmos princípios de Erlenkotter (1978)) e decrescimento do primal proposto por Ro e Tcha (1984). Já Barros e Labbé (1994) propuseram um método de relaxação lagrangeana combinado com uma heurística primal para gerar bons limites superior e inferior em um algoritmo de *Branch-and-Bound*.

Ao longo dos anos, diferentes formulações foram propostas para o problema, dentre as quais destacamos duas principais: uma com apenas uma variável de fluxo de três índices proposta por Balinski (1964) e Manne (1964) e outra com duas variáveis de fluxo de dois índices apresentada por Narula e Ogbu (1979).

2.2.1 Formulação proposta por Balinski e Manne - 1964

Por definição nos modelos de TLUFLP há apenas o deslocamento entre facilidades (cliente-facilidade ou facilidade-facilidade), com o qual são atribuídos seus respectivos custos, associado à demanda de cada um dos clientes.

Para descrever o modelo matemático do problema as seguintes definições foram adotadas: Consideremos K e J o conjunto de nós candidatos para se tornarem facilidades de nível 1 e nível 2, respectivamente, e I o conjunto de clientes. Consideremos também, c_{ijk} serem os custos de atendimento de um cliente $i \in I$ por uma facilidade

$k \in K$ e uma facilidade $j \in J$. Além disso, caso uma facilidade entre o conjunto de nós candidatos no nível 2 seja instalada, então haverá um custo de instalação a_j associado, o mesmo acontece para as facilidades de nível 1, com o parâmetro f_k.

As variáveis de decisão do problema $z_k \in \{0,1\}$ para todo $k \in K$ e $y_j \in \{0,1\}$ para todo $j \in J$, indicam se entre todos os nós candidatos, as facilidades k e j estão instaladas, sendo também definidas ao longo das atribuições o valor das variáveis $x_{ijk} = 1$, quando as facilidades $k \in K$ e $j \in J$ são alocadas ao cliente $i \in I$. Neste contexto o problema pode ser modelado como segue:

$$\min \sum_{k \in K} f_k z_k + \sum_{j \in J} a_j y_j + \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} c_{ijk} x_{ijk} \tag{2.5}$$

$$\text{s.a.:} \sum_{j \in J} \sum_{k \in K} x_{ijk} = 1 \qquad \forall\, i \in I \tag{2.6}$$

$$y_j - \sum_{k \in K} x_{ijk} \geq 0 \qquad \forall\, i \in I, j \in J \tag{2.7}$$

$$z_k - \sum_{j \in J} x_{ijk} \geq 0 \qquad \forall\, i \in I, k \in K \tag{2.8}$$

$$x_{ijk} \geq 0 \qquad \forall\, i \in I, j \in J, k \in K$$

$$z_k \in \{0,1\} \qquad \forall\, k \in K$$

$$y_j \in \{0,1\} \qquad \forall\, j \in J$$

As restrições (2.6) garantem o atendimento de todos os clientes, enquanto que as restrições (2.7) e (2.8) eliminam a possibilidade de haver atribuição de fluxo oriundas de facilidades que não se encontram instaladas. Neste contexto, a função objetivo (2.5) resulta na configuração de custo mínimo para as variáveis z_z, y_j e x_{ijk} que respeitam as restrições apresentadas.

2.2.2 Formulação proposta por Narula e Ogbu - 1979

Mantendo a mesma estrutura e conceitos da formulação proposta por Balinski (1964) e Manne (1964), Narula e Ogbu (1979) propuseram algumas alterações no modelo para o problema capacitado (que também se aplica ao caso não capacitado), de forma a transformar as variáveis de fluxo x_{ijk} em duas outras variáveis de fluxo: $p_{jk} \geq 0$ para as facilidades de primeiro e segundo nível ($k \in K$ e $j \in J$) e $q_{ij} \geq 0$ para as facilidades de segundo nível $j \in J$ e os clientes $i \in I$.

O custo que antes era agregado no parâmetro c_{ijk} passou também a ser dividido em dois outros parâmetros c_{jk} e g_{ij} que representam, respectivamente, os custos de atribuição de uma facilidade $k \in K$ a uma facilidade $j \in J$, e uma facilidade $j \in J$ ser

atribuída a um cliente $i \in I$. Assim, é possível montar uma formulação baseada em duas variáveis de fluxo, como segue:

$$\min \sum_{k \in K} f_k z_k + \sum_{j \in J} a_j y_j + \sum_{j \in J} \sum_{k \in K} c_{jk} p_{jk} + \sum_{i \in I} \sum_{j \in J} g_{ij} q_{ij} \tag{2.9}$$

$$\text{s.a.:} \sum_{j \in J} q_{ij} = 1 \qquad \forall\, i \in I \tag{2.10}$$

$$\sum_{k \in K} p_{jk} = y_j \qquad \forall\, j \in J \tag{2.11}$$

$$y_j - q_{ij} \geq 0 \qquad \forall\, i \in I, j \in J \tag{2.12}$$

$$z_k - p_{jk} \geq 0 \qquad \forall\, j \in J, k \in K \tag{2.13}$$

$$p_{jk} \geq 0 \qquad \forall\, j \in J, k \in K$$

$$q_{ij} \geq 0 \qquad \forall\, i \in I, j \in J$$

$$z_k \in \{0, 1\} \qquad \forall\, k \in K$$

$$y_j \in \{0, 1\} \qquad \forall\, j \in J$$

As restrições (2.10) garantem o atendimento de todos os clientes, enquanto que as restrições (2.11) garantem o atendimento da facilidade $j \in J$, pela soma dos fluxos que saem da facilidade $k \in K$ e chegam à facilidade $j \in J$. As restrições (2.12) e (2.13) eliminam a possibilidade de haver fluxo oriundo de facilidades que não estão instaladas.

2.2.3 Trabalhos recentes sobre o TLUFLP

Em se tratando de trabalhos mais recentes relacionados ao TLUFLP, percebe-se uma tendência para o desenvolvimento de algoritmos heurísticos aproximativos para problemas de localização de facilidades em dois níveis e o multinível (dois ou mais).

Aardal et al. (1999), por exemplo, desenvolveram um algoritmo 3-aproximativo para resolução do problema de k-níveis de facilidades não capacitadas, que utiliza um procedimento de arredondamentos aleatórios das soluções ótimas da relaxação linear do problema primal e do seu dual, para escolher quais facilidades seriam abertas. Já Peng Zhang (2007), propõe um $2 + \sqrt{3} + \epsilon$-algoritmo eficiente de resolução em tempo polinomial do problema de k-níveis de facilidades não capacitadas, que utiliza buscas locais e um algoritmo Guloso proposto por Jain et al. (2003). Gabor e van Ommeren (2010), por sua vez, apresentaram outra alternativa de resolução do problema UFLP multinível, por meio de um algoritmo 3-aproximativo baseado em soluções lineares arredondadas (*LP-rounding*).

Em trabalhos ainda mais recentes abordando o TLUFLP, destaque para os tra-

balhos de Gendron et al. (2013) e Baïou e Barahona (2014). O primeiro propos a resolução do TLUFLP com uma restrição de atribuição simples por meio de um algoritmo Lagrangeano associado a um *Branch-and-Bound* para resolução de instâncias industriais reais e outras artificiais. Já o segundo identificou uma classe de problemas do TLUFLP que possui resolução em tempo polinomial, bem como apresentou algumas inequações e facetas associadas ao politopo do problema.

O presente trabalho utiliza o modelo proposto por Balinski (1964) e Manne (1964), com a única diferença de acrescentar a informação da demanda de cada um dos clientes. Escolheu-se trabalhar com duas abordagens, a primeira delas um método exato baseado no método de Decomposição de Benders (Benders (1962))e a outra duas heurísticas construídas através da estrutura de associação de GRASP com Path Relinking e com outra técnica de perturbação das soluções.

2.3 O Método de Decomposição de Benders para o UFLP

O chamado método de Decomposição de Benders proposto por Benders (1962) é uma dentre várias alternativas para a solução de problemas de natureza inteira mista e não-linear. O algoritmo consiste na decomposição do problema original em dois outros problemas, o problema mestre e o subproblema, que, por sua vez, costuma ser resolvido em seu formato dual. Para explicar o método, consideremos o problema genérico que segue:

$$\begin{aligned} \min\ & f^\top z + c^\top x \\ \text{s.a.:}\ & Ax + Bz \geq b && (2.14)\\ & Dz \geq d && (2.15)\\ & x \geq 0 \\ & z \in \mathbb{Z}^+ \end{aligned}$$

No problema anterior, A, B e D representam as matrizes de coeficientes das restrições (2.14) e (2.15), enquanto c, f, d e b são os parâmetros do modelo. As variáveis de decisão do problema são representadas por x e z. Esse mesmo problema pode ser representado da seguinte forma:

$$\min\ f^\top z + v(z) \tag{2.16}$$
$$\text{s.a.: } Dz \geq d \tag{2.17}$$
$$z \in \mathbb{Z}^+ \tag{2.18}$$

No qual $v(z)$ pode ser definida através da seguinte função:

$$\begin{aligned} v(z) = \min\ & c^\top x \\ \text{s.a.: } & Ax \geq b - Bz \\ & x \geq 0 \end{aligned} \tag{2.19}$$

A função $v(z)$, quando resolvida em seu formato dual, é chamada de subproblema dual de Benders. Para montar o dual do subproblema, associaremos a variável u à restrição (2.19), conforme mostrado em sequência.

$$\begin{aligned} \max\ & (b - Bz)^\top u \\ \text{s.a.: } & A^\top u \leq c \\ & u \geq 0 \end{aligned} \tag{2.20}$$

A resolução do subproblema pelo primal ou dual implicará sempre no mesmo valor de $v(z)$, no entanto, o problema dual não depende de variáveis inteiras, o que o torna mais atrativo. Neste sentido, podemos montar um conjunto de soluções viáveis para o subproblema dual de Benders $\Omega = \{u \geq 0 : A^\top u \leq c\}$, que, por sua vez, pode ser criado a partir de um número finito de pontos extremos e de raios extremos. Chamaremos de H o conjunto de pontos extremos, assim como G será o conjunto de raios extremos de Ω.

É valido destacar ainda que a utilização de soluções ilimitadas para o subproblema dual de Benders não é benéfica, pois estas representam soluções primais inviáveis. As restrições apresentadas a seguir foram criadas para garantir que apenas valores primais viáveis de z sejam usados durante a resolução de $v(z)$.

$$0 \geq (b - Bz)^\top u_h \quad \forall h \in H \tag{2.21}$$

Assim, podemos reformular o problema descrito pelas equações de (2.16) à (2.18), da seguinte forma:

$$\begin{aligned} &\min\ f^\top z + \max\{(b - Bz)^\top u : u \in \Omega\} && && (2.22)\\ &\text{s.a: } Dz \geq d \\ &\quad 0 \geq (b - Bz)^\top u_h && \forall u_h \in H \\ &\quad z \in \mathbb{Z}^+ \end{aligned}$$

A solução do problema de maximização, presente na função objetivo (2.22), resultará sempre em um ponto extremo $u_g \in G$. Se substituírmos o problema de maximização por uma variável contínua η chegaremos ao chamado problema mestre de Benders, conforme mostrado em sequência.

$$\begin{aligned} &\min\ f^\top z + \eta \\ &\text{s.a: } Dz \geq d \\ &\quad \eta \geq (b - Bz)^\top u_g && \forall u_g \in G && (2.23)\\ &\quad 0 \geq (b - Bz)^\top u_h && \forall u_h \in H && (2.24)\\ &\quad z \in \mathbb{Z}^+ \\ &\quad \eta \geq 0 \end{aligned}$$

O problema mestre de Benders possui menos variáveis e mais restrições que o problema genérico incial. Se considerarmos que nem todas estas restrições estarão ativas na solução ótima, uma boa alternativa seria a inserção gradativa (iterações de Benders) das restrições (2.23) e (2.24), conhecidas como cortes de otimalidade e de viabilidade, respectivamente, no problema mestre de Benders.

Nos últimos anos o método de decomposição de Benders tem sido empregado com sucesso em diferentes problemas. Cordeau et al. (2000) resolveram o problema de atribuição de carros e locomotivas por meio do método de decomposição de Benders. Costa (2005) utilizou o método de Benders para resolução do problema de desenho de redes com custo fixo, enquanto Cordeau et al. (2001) aplicaram o método no problema de roteamento simultâneo de aviões e agendamento de tripulação. O método de Benders se aplica também em algumas variações do problema de localização de *Hubs*, como mostrado nos trabalhos de Camargo et al. (2008), Camargo et al. (2009), Contreras et al. (2011) e Martins de Sá et al. (2013).

Apresentaremos agora um exemplo do método de decomposição de Benders aplicado ao modelo básico de UFLP (que pode ser encontrado no trabalho de Klose e Drexl (2005)) que segue:

$$
\begin{aligned}
\min \quad & \sum_{j\,=\,1}^{n} f_j y_j + \sum_{i\,=\,1}^{m} \sum_{j\,=\,1}^{n} c_{ij} x_{ij} & \\
\text{s.a.:} \quad & \sum_{j\,=\,1}^{n} x_{ij} = 1 & i = 1...m \\
& y_j - x_{ij} \geq 0 & i = 1...m;\ j = 1...n \\
& x_{ij} \geq 0 & i = 1...m;\ j = 1...n \\
& y_j \in \{0, 1\} & j = 1...n
\end{aligned}
$$

Consideremos que y_j é uma variável complicante, ou seja, ao subdividirmos o problema mestre e subproblema, as variáveis y_j pertencerão apenas ao Problema Mestre, e para o Subproblema serão apenas parâmetros $\bar{y}_j$. Assim o Subproblema de Benders pode ser escrito da seguinte forma:

$$
\begin{aligned}
\min \quad & \sum_{i\,=\,1}^{m} \sum_{j\,=\,1}^{n} c_{ij} x_{ij} & & \\
\text{s.a.:} \quad & \sum_{j\,=\,1}^{n} x_{ij} = 1 & i = 1...m & \quad (2.25) \\
& -x_{ij} \geq -\bar{y}_j & i = 1...m;\ j = 1...n & \quad (2.26) \\
& x_{ij} \geq 0 & i = 1...m;\ j = 1...n &
\end{aligned}
$$

Sabendo que o método trabalha com o Subproblema em sua forma dual, devemos encontrar o dual do problema anterior. Associando as variáveis duais $\lambda_i \in \mathbb{R}$ e $\pi_{ij} \geq 0$ às restrições (2.25) e (2.26) respectivamente, obtemos o seguinte subproblema dual:

$$
\begin{aligned}
\zeta(\bar{y}_j) = \max \quad & \sum_{i\,=\,1}^{m} \lambda_i - \sum_{j\,=\,1}^{n} (\sum_{i\,=\,1}^{m} \pi_{ij}) \bar{y}_j & & \quad (2.27) \\
\text{s.a.:} \quad & \lambda_i - \pi_{ij} \leq c_{ij} & i = 1...m;\ j = 1...n & \quad (2.28) \\
& \pi_{ij} \geq 0 & i = 1...m;\ j = 1...n & \\
& \lambda_i \in \mathbb{R}^+ & i = 1...m &
\end{aligned}
$$

Para esse Subproblema Dual em específico, Magnanti e Wong (1990) propõem ainda uma solução natural para o problema, baseada nos seguintes princípios: Consideremos $O = \{j \mid y_j = 1\}$ como facilidades ativas ($O = Open$) e $C = \{j \mid y_j = 0\}$ como

facilidades inativas ($C = Close$) numa iteração qualquer s. Consideremos também que $c_{ij(i)}$ representa o menor custo de atendimento do cliente i, pela facilidade j nesta mesma iteração s do algoritmo de Benders. Isto permite fazer as seguintes afirmações: Para todo $i = 1...m$, as equações de (2.29) a (2.31) resolvem o Subproblema Dual.

$$\lambda_i^s = c_{ij(i)} \qquad i = 1...m \tag{2.29}$$

$$\pi_{ij}^s = 0 \qquad se\ j \in O,\ j = 1...n \tag{2.30}$$

$$\pi_{ij}^s = \max(0,\ \lambda_i - c_{ij}) \qquad se\ j \in C,\ j = 1...n \tag{2.31}$$

O valor ótimo das variáveis duais λ_i^s e π_{ij}^s tem uma interpretação bem interessante do ponto de vista econômico. Pode-se considerar que λ_i^s é o custo de serviço dos clientes $i \in I$ na iteração s para uma dada configuração do parâmetro $\bar{y}_j$. Já o π_{ij}^s é a redução do custo de serviço do cliente $i \in I$, quando a facilidade $j \in J$ está aberta.

A montagem do Problema Mestre de Benders não é uma formação estática, ou seja, todas as vezes que o Subproblema Dual de Benders for resolvido, o Problema Mestre receberá uma nova restrição (corte) que carregará as informações referentes a uma iteração do Subproblema Dual de Benders. É necessário também que seja criada uma nova variável η, que faz parte da montagem do chamado corte de Benders. Assim podemos chegar ao problema mestre de Benders como segue:

$$\phi(y_j, \eta) = \min \sum_{j\,=\,1}^{n} y_j + \eta$$

$$\text{s.a.: } \eta \geq \sum_{i\,=\,1}^{m} \bar{\lambda}_i - \sum_{j\,=\,1}^{n} (\sum_{i\,=\,1}^{m} \bar{\pi}_{ij}) y_j \tag{2.32}$$

$$\sum_{j\,=\,1}^{n} y_j \geq 1 \tag{2.33}$$

$$y_j \in \{0, 1\} \qquad j = 1...n$$

$$\eta \leq 0$$

A equação (2.32) representa o chamado corte de Benders, e sua montagem é feita utilizando a expressão da função objetivo do Subproblema Dual (2.27) e a variável η. A restrição (2.33) é adicionada ao Problema Mestre para garantir que pelo menos uma das n facilidade seja instalada durante a execução do método de Benders.

Em linhas gerais, a ideia principal do método é ir acrescentando cortes ao Problema Mestre a cada resolução do Subproblema Dual, até que a diferença entre os valores do

limite inferior - LB (*lower bound*) e o superior - UB (*upper bound*) seja zero. Os passos lógicos do método de Benders clássico aplicado no UFLP pode ser observado no Algoritmo (1).

Algoritmo 1: Método de Benders Tradicional

```
início
    Pare ← falso
    CT ← 0
    Gap ← 100
    enquanto (Pare = falso) faça
        Resolva o PM
        LB ← φ(y_j, η)
        Atualize os parâmetros do SD
        Resolva o SD
        Acrescente o corte de Benders ao PM
        CT ← ζ(ȳ_j) + FOM
        UB ← min(UB, CT)
        Gap ← 100(UB − LB)/UB
        se (Gap < z) então
            Pare ← verdadeiro
        fim
    fim
fim
```

No Algoritmo (1), PM representa o Problema Mestre, SD o Subproblema Dual, CT é o custo total da iteração, z é valor que podemos considerar como 0 ($1x10^{-9}$, por exemplo) e FOM é $\phi(y_j, \eta)$ desconsiderando o valor de η.

A escolha desta técnica para resolução do TLUFLP se deu principalmente pelas suas características básicas de apresentar variáveis complicantes (relativas à instalação ou não das facilidades) e os subproblemas resultantes poderem ser resolvidos sem a utilização do SIMPLEX[3].

2.3.1 Técnicas de Aceleração do Método de Benders para o UFLP

De maneira geral, a utilização do método de Benders em sua forma clássica pode não ser muito atrativa do ponto de vista do tempo computacional de resolução dos problemas, tendo em vista que os resolvedores disponíveis no mercado costumam apresentar maior eficiência quando comparado ao algoritmo de Benders.

[3]Técnica básica de resolução de problemas lineares proposta por George Bernard Dantzig em 1947.

Neste sentido, ao longo dos anos pesquisadores buscaram aperfeiçoar e incrementar o método de Benders de forma a torná-lo competitivo frente a outras técnicas de resolução.

Neste trabalho em específico escolheu-se trabalhar com algumas dessas técnicas de aceleração do algoritmo de Benders que são: a utilização da técnica de Pré-aquecimento (*Hot Start*), o uso de uma única árvore de *Branch-and-Bound* pelas funções *Callback* do CPLEX[4], e a geração de cortes múltiplos (*Multicut*).

2.3.1.1 Pré-aquecimento

O Pré-aquecimento *Hot Start* visa a obtenção de uma solução inicial de forma rápida e eficiente para acelerar a convergência dos métodos de resolução de problemas lineares. Existem diferentes formas de se implementar a técnica de Pré-aquecimento, uma delas foi proposta por McDaniel e Devine (1977) que sugere a relaxação linear das variáveis inteiras do Problema Mestre nas primeiras iterações do algoritmo de Benders. Assim, é possível gerar em poucas iterações (pré-definidas) alguns cortes ao Problema Mestre.

O número de iterações de Pré-aquecimento é um parâmetro que precisa ser testado. Normalmente o número de iterações é um valor bem pequeno, uma vez que um número grande pode implicar no aumento do tempo computacional de resolução do problema.

2.3.1.2 Utilização de uma única árvore de *Branch-and-Bound*

As funções *Callback* do CPLEX são métodos recursivos de chamada de funções, que podem ser utilizados na função principal de algoritmos de otimização. Existem diferentes tipos de funções *Callback*, dentre as quais podemos citar como as mais comuns a *CutCallback* e a *LazyConstraintCallback*.

Um exemplo de aplicação do método de Benders combinado com as funções *Callback* do CPLEX é apresentado no trabalho de Fortz e Poss (2009), que utiliza a função *CutCallback* no seu algoritmo de *Branch-and-Cut*.

A utilização das funções *Callback* combinadas com o método de Benders pode ser considerada uma boa opção pelo fato de sua utilização resultar em uma única árvore de *Branch-and-Bound* durante a resolução do Problema Mestre de Benders.

2.3.1.3 Múltiplos cortes

Segundo You e Grossmann (2013), pelo fato do algoritmo de Decomposição Benders tradicional retornar apenas um corte por iteração para o Problema Mestre, em problemas de grande porte, a sua convergência pode ser lenta, sendo que o algoritmo pode levar muitas iterações para chegar ao limite de tolerância predefinido.

[4]Resolvedor comercial da empresa IBM (*International Business Machines*).

Para acelerar o algoritmo de Benders os trabalhos de Birge e Louveaux (1988) e You e Grossmann (2013) propõem a decomposição da variável η em cenários de forma a retornar múltiplos cortes s (equivalente ao número de cenários) em cada iteração. Vale ressaltar que a adição de múltiplos cortes somente é possível em casos específicos que o modelo utilizado pode ser desagregado em relação a algum índice.

Para exemplificar esse método, vamos dar continuidade ao exemplo do método de Benders aplicado ao UFLP utilizado neste capítulo. É possível que o Subproblema Dual de Benders seja desagregado para cada i que representa o conjunto de clientes, conforme ilustrado nas equações que seguem.

$$\begin{aligned} &\max \lambda_i - \sum_{j\,=\,1}^{n} \pi_{ij}\bar{y}_j \\ &\text{s.a.: } \lambda_i - \pi_{ij} \leq c_{ij} && j = 1...n \\ &\pi_{ij} \geq 0 && j = 1...n \\ &\lambda_i \in \mathbb{R}^+ \end{aligned}$$

Dessa forma, o corte de Benders (2.32) pode ser transformado em outros i cortes, ou seja, a cada resolução do Subproblema será gerado um corte para cada cliente i. As equações a seguir apresentam a versão múltiplos cortes de Benders aplicada ao UFLP.

$$\begin{aligned} &\min \sum_{j\,=\,1}^{n} y_j + \sum_{i\,=\,1}^{m} \eta_i \\ &\text{s.a.: } \eta_i \geq \bar{\lambda}_i - \sum_{j\,=\,1}^{n} \bar{\pi}_{ij} y_j && i = 1...m \\ &\sum_{j\,=\,1}^{n} \geq 1 \\ &y_j \in \{0,1\} && j = 1...n \\ &\eta_i \leq 0 \end{aligned} \tag{2.34}$$

Existe também a possibilidade que seja montada uma versão de múltiplos cortes de Benders para o UFLP, no qual o Subproblema Dual é resolvido de maneira agregada e sejam gerados i cortes sob a mesma estrutura das equações apresentadas anteriormente. Contudo, não é possível afirmar se a abordagem agregada ou a desagregada é mais eficiente, uma vez que o valor das variáveis duais do Subproblema de Benders podem assumir valores diferentes para cada uma das duas versões, mantendo sempre o mesmo

valor de função objetivo.

2.3.2 Quatro diferentes tipos de cortes de Benders

Visando a tornar o método de Benders ainda mais competitivo para o caso do problema de localização de facilidades, Magnanti e Wong (1990) apresentaram três diferentes tipos de cortes que podem vir a ser utilizados como procedimentos que limitam o algoritmo de *branch-and-bound*, ou até mesmo no desenvolvimento de algoritmos heurísticos para o problema. São eles, o Fechamento de Facilidades, o Pareto-Ótimo via inspeção e o δ-Vizinhança. Papadakos (2008) por sua vez, apresentou outra abordagem para o corte Pareto-Ótimo, o qual iremos chamar de Pareto-Ótimo/Papadakos.

2.3.2.1 Fechamento de Facilidades - CF

O chamado corte Fechamento de Facilidades ou CF (*Closing Facility*) é um corte derivado do corte de Benders tradicional quando aplicado ao problema de localização de facilidades. Segundo Magnanti e Wong (1990) no corte tradicional são consideradas apenas as economias provenientes de abertura de novas facilidades, enquanto no CF é considerada também a adição de economias relacionadas ao fechamento de facilidades. Para apresentar esse novo corte, iremos dar prosseguimento ao exemplo abordado ao longo deste capítulo.

Primeiramente vamos fazer algumas definições: Utilizaremos os conceitos apresentados nas equações de (2.29) a (2.31). Consideremos que uma facilidade j que está atendendo o cliente i na iteração de Benders atual é representada por $j(i)$, e que caso ela seja fechada, o cliente i deve ser atendido por outra facilidade $k = 1...n$, tal que $k \neq j(i)$, que será representada por $k(i)$. Desse modo, o novo custo de atendimento de todos os $i = 1...m$ deve ser representado por $c_{ik(i)}$, calculado conforme a expressão que segue:

$$c_{ik(i)} = \min\{c_{iq} : \ 1 \leq q \leq n \wedge q \neq j(i)\}$$

Para fazer o cálculo referente ao custo de fechar uma facilidade aberta na iteração atual de Benders, vamos criar duas novas variáveis σ_i e υ_j, que são regidas pelas seguintes equações:

$$\sigma_i = \max\{c_{ik(i)} - c_{ij(i)},\ 0\}$$
$$\upsilon_j = \sum_{i\ =\ 1}^{m} \{\sigma_i :\ 1 \leq i \leq m \wedge j = j(i)\}$$

Assim, é possível escrever um novo corte de Benders, o chamado CF em uma estrutura muito parecida com a do corte de Benders tradicional (2.32).

$$\eta \geq \sum_{i\ =\ 1}^{m} \bar{\lambda}_i + \sum_{j \in O} (1\ -\ y_j)\upsilon_j - \sum_{j \in C} (\sum_{i\ =\ 1}^{m} \bar{\pi}_{ij}) y_j \tag{2.35}$$

Uma diferença notória entre as equações (2.32) e (2.35) é que os somatórios em j passam a ser associados a situação da variável y_j, que pode estar ativa ou não em cada iteração do algoritmo de Benders. A variável υ_j indica o custo mínimo total de serviço incorrido sobre todos os clientes i atendidos pela facilidade j.

Magnanti e Wong (1990) ainda destacam que quando algum $\upsilon_j \neq 0$ e ao mesmo tempo algum $y_j \neq 0$ em uma solução corrente, o CF domina o corte tradicional.

2.3.2.2 Pareto-Ótimo via inspeção

No trabalho desenvolvido por Magnanti e Wong (1981), eles perceberam que o subproblema dual de Benders possui múltiplas soluções ótimas, o que possibilita gerar vários cortes diferentes. Eles então desenvolveram uma técnica que garantisse a geração de apenas um corte único que dominasse todos os outros cortes, o chamado corte Pareto-Ótimo.

Vamos dar continuidade ao exemplo do UFLP deste capítulo, também abordado por Magnanti e Wong (1990). Em uma iteração de Benders qualquer, temos uma configuração $\bar{y}_j$ das variáveis do Problema Mestre. Sabendo que o Subproblema Dual pode ser decomposto para cada cliente i (como apresentado na seção sobre múltiplos cortes), e ainda utilizando as equações de (2.29) a (2.31), encontramos que a solução ótima de todos os i Subproblemas Duais é dada por $\gamma_i(\bar{y}_j) = \bar{\lambda}_i$ e que $(\lambda_i - \sum_{j=1}^{n} \bar{y}_j \pi_{ij}) \leq \gamma_i(\bar{y}_j)$ é uma solução factível para quaisquer valores de (λ, π). Este fato nos leva a seguinte conclusão:

$$\sum_{i\ =\ 1}^{m} (\lambda_i - \sum_{j\ =\ 1}^{n} \bar{y}_j \pi_{ij}) = \sum_{i\ =\ 1}^{m} \gamma_i(\bar{y}_j)$$

Essa condição anteriormente descrita é valida se e somente se $(\lambda_i - \sum_{j=1}^{n} y_j \pi_{ij}) = \gamma_i(y_j)$. Neste sentido, Magnanti e Wong (1990) afirmam que é possível fazer uma nova decomposição do Subproblema Dual para cada cliente i, que resultará em um vetor de variáveis λ_i e π_{ij}, para todo $i = 1...m$ e $j = 1...n$, do tipo Pareto-Ótimo. Mas isso somente será válido se as seguintes definições forem respeitadas:

Definição 1. *Um corte Pareto-ótimo é aquele que não pode ser dominado por nenhum outro corte. Seja* $\Omega = \{(\lambda_i, \pi_{ij}) : \lambda_i \in \mathbb{R},\ \pi_{ij} \geq 0,\ \lambda_i - \pi_{ij} \leq c_{ij}\ \forall i, j\}$*, o conjunto de valores viáveis para as variáveis duais* λ_i *e* π_{ij}*. Então o corte de Benders associado as variáveis* λ_i^* *e* $\pi_{ij}^* \in \Omega$ *domina todos os outros cortes associados aos* λ_i *e* $\pi_{ij} \in \Omega$*, se:*

$$\sum_{i\,=\,1}^{m} \lambda_i^* - \sum_{j\,=\,1}^{n} (\sum_{i\,=\,1}^{m} \pi_{ij}^*)\bar{y}_j \geq \sum_{i\,=\,1}^{m} \lambda_i - \sum_{j\,=\,1}^{n} (\sum_{i\,=\,1}^{m} \pi_{ij})\bar{y}_j$$

Além disso, para construção dos cortes Pareto-ótimo, Magnanti e Wong (1990) utilizam os chamados *core-points* que são definidos em sequência.

Definição 2. *Um ponto* y_j^0*, para todo cliente* i *é um core point se ele pertence ao interior relativo da casca convexa, ou seja, se* $y_j^0 \in \boldsymbol{ir}(Y^c)$*, onde* $\boldsymbol{ir}(Y^c)$ *é o interior relativo da casca convexa* Y^c *de* Y *(conjunto de soluções viáveis).*

Assim, podemos montar o seguinte Subproblema Dual para cada cliente i = 1..m:

$$\max\ \lambda_i - \sum_{j\,=\,1}^{n} y_j^0 \pi_{ij} \tag{2.36}$$

$$\text{s.a.:}\ \lambda_i - \sum_{j\,=\,1}^{n} \bar{y}_j \pi_{ij} = \bar{\lambda}_i \tag{2.37}$$

$$\lambda_i - \pi_{ij} \leq c_{ij} \qquad j = 1...n$$

$$\pi_{ij} \geq 0 \qquad j = 1...n$$

$$\lambda_i \in \mathbb{R}^+$$

Agora devemos mostrar que para cada cliente i o valor ótimo do Subproblema Dual apresentado nas equações anteriores é uma função linear de λ_i por partes. Para isso, ela deve atender a seguinte condição:

$$\lambda_i - \sum_{j \in O} \pi_{ij} = \bar{\lambda}_i = c_{ij(i)}$$

Além disso, $\lambda_i - \pi_{ij(i)} \leq c_{ij(i)}$ e $\pi_{ij} \geq 0$ devem ser verdadeiros para toda facilidade j, bem como $\pi_{ij} = 0$ para toda facilidade $j \in O$, $j \neq j(i)$, e $\pi_{ij(i)} = \lambda_i - c_{ij(i)} = \lambda_i - \bar{\lambda}_i$. Atendidas todas essas condições, se resolvermos substituir o λ_i na função objetivo do novo Subproblema Dual (2.36), chegaremos à seguinte equação:

$$\max \bar{\lambda}_i - \sum_{j\,=\,1}^{n} (\bar{y}_j - y_j^0)\pi_{ij}$$

Em complementaridade, devemos lembrar de associar para cada facilidade $j \in C$, como sendo $\bar{y}_j = 0$, e o coeficiente $\varepsilon_j \equiv (\bar{y}_j - y_j^0)$ de π_{ij} sendo menor ou igual a zero. Chamaremos de $\varepsilon_{j(i)}$ o coeficiente associado um cliente $i \in I$ atendido pela facilidade $j \in O$. Portanto, o valor ótimo das variáveis π_{ij} deve ser obtido através da expressão $\pi_{ij} = \max(\lambda_i - c_{ij})$. Reunindo todas essas informações podemos montar uma função da variável λ_i, $f(\lambda_i)$ conforme a equação (2.38).

$$f(\lambda_i) = \bar{\lambda}_i + \varepsilon_{j(i)}(\lambda_i - \bar{\lambda}_i) + \sum_{j \in C} \varepsilon_j \max(0, \lambda_i - c_{ij}) \tag{2.38}$$

Magnanti e Wong (1990) ainda propõem associar um limite superior L_i e um limite inferior $\bar{\lambda}_i$ para λ_i, de forma a auxiliar na otimização do Subproblema Dual utilizando a ideia do *core point*. Segundo eles, o valor de L_i pode ser calculado pela equação (2.39).

$$L_i = \min\{c_{ij} : \; j \in O \wedge j \neq j(i)\} \tag{2.39}$$

Assim, considerando que a função linear (2.38) por partes é côncava em λ_i, é possível minimizar essa função dentro dos segmentos lineares no intervalo $\bar{\lambda}_i \leq \lambda_i \leq L_i$, de forma crescente de maneira a haver uma evolução dos segmentos até que a inclinação de algum segmento se torne negativa.

Para explicar como seria feito o cálculo das variáveis $\bar{\lambda}_i$ na construção do corte Pareto-Ótimo, Magnanti e Wong (1990) desenvolveram um algoritmo que mostra o passo a passo da técnica para uma iteração qualquer de Benders, conforme ilustrado no Algoritmo (2).

No Algoritmo (2), B é um subconjunto de facilidades $j \in C$, enquanto θ_i é um parâmetro de custo e λ_i^* representa o valor ótimo da variável λ_i para todo $i = 1...m$ na iteração atual de Benders.

Em resumo, uma vez que o valor de λ_i, para cada um dos clientes i é calculado,

Algoritmo 2: Cálculo das variáveis duais $\bar{\lambda}_i$ para o corte Pareto-Ótimo

início
  para todo $(i = 1...m \wedge j = 1...n)$ **faça**
    $\varepsilon_j = (\bar{y}_j - y_j^0)$
    $c_{ij(i)} = \min\{c_{ij} : \ j \in O\}$
    $L_i = \min\{c_{ij} : \ j \in O \wedge j \neq j(i)\}$
  fim
  para todo $(i = 1...m)$ **faça**
    $\lambda_i \leftarrow \bar{\lambda}_i$
    enquanto $(\lambda_i \neq \lambda_i^*)$ **faça**
      $B \leftarrow \{j \in C : c_{ij} \leq \lambda_i\}$
      $s \leftarrow \varepsilon_{j(i)} + \sum_{j \in B} \varepsilon_j$
      se $(s \leq 0)$ **então** $\lambda_i^* \leftarrow \lambda_i$
      se $(s > 0 \wedge B = C)$ **então** $\lambda_i^* \leftarrow \lambda_i = L_i$
      $\theta_i \leftarrow \min\{c_{ij} : j \in C \wedge j \notin B\}$
      se $(\theta_i > L_i)$ **então** $(\lambda_i \leftarrow L_i) \wedge (\lambda_i^* \leftarrow L_i)$ **senão** $\lambda_i \leftarrow \theta_i$
    fim
  fim
fim

podemos encontrar o valor das variáveis π_{ij}, através da expressão $\pi_{ij} = \max(\lambda_i - c_{ij}, 0)$. Assim, podemos montar o chamado corte Pareto-Ótimo utilizando a mesma estrutura do corte de Benders Tradicional (2.32), conforme ilustrado a seguir:

$$\eta \geq \sum_{i\,=\,1}^{m} \lambda_i - \sum_{j\,=\,1}^{n} (\sum_{i\,=\,1}^{m} \pi_{ij}) y_j$$

Como o cálculo das variáveis λ_i e π_{ij} é feito utilizando os *core points*, o corte gerado domina o corte tradicional.

2.3.2.3 Pareto-Ótimo via subproblema de Papadakos

Segundo Papadakos (2008), apesar de ser uma boa ideia o uso dos cortes Pareto-ótimo proposta por Magnanti e Wong (1981), em alguns casos a utilização dessa técnica pode vir a provocar instabilidade numérica no Subproblema Dual devido às restrições (2.37), e além disso, a atualização do *core point* também pode ser bem complicada.

Visando aperfeiçoar a técnica de Pareto-Ótimo tradicional, Papadakos (2008) desenvolveu um Subproblema que excluía as restrições que causavam instabilidade. Vamos utilizar o exemplo do Problema de Localização de Facilidades para ilustrar a estrutura do novo Subproblema Dual proposta por Papadakos (2008). As equações a seguir apresentam o Subproblema Dual segundo estes novos conceitos, no qual y_j^0 são as variáveis

de *core points* associadas as facilidades $j = 1...n$.

$$\begin{aligned}
\max \sum_{i=1}^{m} \lambda_i - \sum_{j=1}^{n} (\sum_{i=1}^{m} \pi_{ij}) \bar{y}_j^0 & \\
\text{s.a.: } \lambda_i - \pi_{ij} \leq c_{ij} & \qquad i = 1...m;\ j = 1...n \\
\pi_{ij} \geq 0 & \qquad i = 1...m;\ j = 1...n \\
\lambda_i \in \mathbb{R}^+ & \qquad i = 1...m
\end{aligned}$$

Essa nova técnica geração dos *core points* passou a ser calculada de forma independente do Subproblema Dual. O cálculo dos *core points* é feito por meio da combinação convexa do valor da variável y_j, $j = 1...n$, extraída do Problema Mestre na iteração t, associada a informação do valor da variável de *core point* y_j^0, $j = 1...n$, da iteração anterior $(t-1)$, conforme mostrado na equação seguinte:

$$y_j^{0(t)} = \tau y_j^{0(t-1)} + (1-\tau) y_j^{(t)} \qquad j = 1...n;\ 0 \leq \tau \leq 1$$

Papadakos (2008) sugere a utilização de $\tau = \frac{1}{2}$ para a atualização dos *core points*, valor do qual ele obteve os melhores resultados. Neste contexto, podemos montar o corte Pareto-Ótimo/Papadakos como segue.

$$\eta \geq \sum_{i=1}^{m} \bar{\lambda}_i - \sum_{j=1}^{n} (\sum_{i=1}^{m} \bar{\pi}_{ij}) y_j^0$$

Desta forma, a cada iteração do método de Benders um novo corte é gerado e os valores dos *core points* y_j^0 são atualizados.

2.3.2.4 δ-Vizinhança

O corte δ-Vizinhança (em inglês δ-*Neighborhood*) pode ser construído de duas formas: a primeira delas é seguindo alguns procedimentos utilizados no corte de Pareto-Ótimo; a segunda é utilizando conceitos que o tornam sempre igual ou superior ao CF. As duas abordagens utilizam um novo conceito de vizinhança $N_i(\delta)$ para todos os clientes i, proposto por Magnanti e Wong (1990) e com o qual eles julgam ser muito útil para construir um corte de Benders para o Problema de Localização de Facilidades. Vamos novamente dar sequência no exemplo utilizado neste capítulo.

O conceito de $N_i(\delta)$ é dado pelo conjunto de facilidades $j = 1...n$ que respeitam a

equação $\lambda_i = \bar{\lambda}_i + \delta$. Vamos definir também o conceito de vizinhanças internas $N_i^0(\delta)$ de $i = 1...m$, que é dada pela equação $N_i^0(\delta) = \{j = 1...n : c_{ij} < \bar{\lambda}_i + \delta\}$.

Primeiramente devemos considerar que $\delta_i = 0$ para as facilidades $j = 1...n$ fechadas e $\delta_i = c_{ij(i)}$ para as facilidades $j = 1...n$ abertas. Neste sentido, vamos criar uma nova variável α_j, que representa a distância/custo de se alcançar a segunda facilidade vizinha j mais próxima do cliente i, como mostrado a seguir:

$$\alpha_j = \sum_{i\,=\,1}^{m} \{\delta_i : \ 1 \leq i \leq m \wedge j = j(i)\}$$

Assim, podemos montar um corte baseado na estrutura do CF como mostrado na equação que segue:

$$\eta \geq \sum_{i\,=\,1}^{m} \bar{\lambda}_i + \sum_{j \in O} (1 \ - \ y_j)\alpha_j - \sum_{j \in C} (\sum_{i\,=\,1}^{m} \bar{\pi}_{ij}) y_j$$

Segundo Magnanti e Wong (1990), para montar um corte que domine o CF tradicional devemos expandir todas as facilidades $j = 1...n$ abertas em um único valor $\bar{\delta}$, ou seja, vamos somar um valor de $\bar{\delta}$ ao α_j da equação anterior. O valor de $\bar{\delta}$ deve ser o maior valor possível que respeite as seguintes restrições:

- Toda facilidade $j = 1...n$ fechada pode pertencer a somente uma única vizinhança N_i^0 das facilidades $j = 1...n$ abertas;
- Somente a própria facilidade $j = 1...n$ aberta pode estar contida na sua vizinhança N_i^0, ou seja, o valor de $\bar{\delta}$ deve respeitar a seguinte equação:

$$\bar{\lambda}_i + \bar{\delta} = \lambda_i \leq L_i$$

  onde L_i é o mesmo valor limite (2.39) calculado no método de Pareto-Ótimo.

Outra consideração importante é que devido ao incremento de $\bar{\delta}$, torna-se necessário a criação da variável Δ_j como fator de compensação para corrigir o valor das variáveis π_{ij} (calculadas conforme a equação (2.31)). Vale ressaltar que $\Delta_j \leq \bar{\delta}$ para toda facilidade $j = 1...n$ fechada e o seu cálculo é baseado na diferença entre π_{ij} e π_{ij}^{δ}, conforme mostrado abaixo:

$$\pi_{ij}^{\delta} = \max(\bar{\lambda}_i + \bar{\delta} - c_{ij}, \ 0)$$
$$\Delta_j = \pi_{ij} - \pi_{ij}^{\delta}$$

Por fim, podemos montar o corte δ-Vizinhança como mostrado na equação (2.40).

$$\eta \geq \sum_{i=1}^{m} \bar{\lambda}_i + \sum_{j \in O}(1 - y_j)(\alpha_j + \bar{\delta}) - \sum_{j \in C}((\sum_{i=1}^{m} \bar{\pi}_{ij}) + \Delta_j)y_j \tag{2.40}$$

A diferença principal entre (2.35) e (2.40) está na presença de dois novos parâmetros, $\bar{\delta}$ e Δ_j, que tornam o corte δ-Vizinhança mais forte quando esses parâmetros são maiores do que zero.

2.4 Métodos heurísticos

Técnicas heurísticas para resolução de problemas lineares costumam ser boas alternativas para problemas de grande porte, principalmente por apresentarem boas soluções em um tempo inferior aos métodos exatos, na maioria dos casos. Por outro lado, as técnicas heurísticas não garantem que a solução encontrada seja ótima. Portanto chamamos de heurísticas, os algoritmos aproximativos que oferecem soluções viáveis para os problemas, mas que não apresentam comprovação matemática de convergência para a solução ótima.

O termo Meta-heurística é empregado sempre que se utiliza uma técnica heurística genérica disponível na literatura que pode ser extendida vários problemas distintos. Alguns exemplos de Meta-heurísticas clássicas são: Algoritmo Genético, *Simulated annealing*, *Greedy Randomized Adaptive Search Procedures* (GRASP), Busca tabu e Colônia de formigas. Nas seções de revisão que seguem serão apresentados alguns conceitos de Meta-heurísticas utilizadas ao longo deste trabalho.

2.4.1 Procedimento aleatório guloso com busca adaptativa - GRASP

O termo GRASP vem da expressão em inglês *"Greedy Randomized Adaptive Search Procedures"* e surgiu de um trabalho proposto por Feo e Resende (1989), e pode ser entendida como um processo iterativo de geração de múltiplas soluções iniciais que normalmente é empregado para a otimização de problemas de natureza combinatória. Resende e Ribeiro (2003) reuniram algumas aplicações de heurísticas baseadas em GRASP que obtiveram ótimos resultados.

Alguns exemplos de aplicação do GRASP em problemas combinatórios podem ser vistos nos trabalhos de Klincewicz (1992) que combinaram o método de Busca Tabu com GRASP para resolução do problema de localização de *p-hub*, Binato et al. (2001)

que utilizaram o método para resolução de problemas de expansão de redes de transmissão, e Martins et al. (2009) que desenvolveram um algoritmo híbrido dos métodos de Simulated Annealing e GRASP para o planejamento de aulas de um departamento.

Segundo Feo e Resende (1995) e Resende e Ribeiro (2003) o GRASP é constituída de duas principais fases: na primeira delas é construída uma solução inicial aleatória parcialmente gulosa; e na segunda, essa solução é refinada por buscas locais em diferentes vizinhanças a partir da solução inicial, visando encontrar melhores soluções para o problema. Essas duas fases são repetidas até que um critério de parada seja atingido.

2.4.2 Procedimento de descida em vizinhança variável - VND

A criação de variados tipos de vizinhanças[5] possibilita explorar diferentes soluções para o problema estudado. Esse tipo de manobra é necessário quando o método utilizado fica preso em uma área de busca restrita, que somente pode ser ampliada mediante a perturbação da solução corrente (ou conjunto de soluções).

Mladenović e Hansen (1997) propuseram um método chamado VND (Variable Neighborhood Descent) que permite a integração de todas as buscas elaboradas para o problema. O método consiste na exploração sistemática das vizinhanças de uma solução, por meio de uma busca local em forma de descida que evita que o algoritmo fique preso em um ótimo local. Durante a exploração de uma vizinhança, todo movimento de melhora é aceito. Além disso, sempre que houver melhoria em uma das vizinhanças, o algoritmo refaz todas as buscas do VND a partir da primeira.

2.4.3 Método de reconexão por caminhos - Path Relinking

O método denominado Path Relinking (PR) surgiu de um trabalho desenvolvido por Glover (1996) sobre Busca Tabu ou *Tabu Search* (TS), como uma estratégia de intensificação e diversificação de buscas. Em resumo, o algoritmo PR é inicado com a montagem de conjunto elite C que reúne as melhores soluções encontradas para o problema até o momento. Feito isso, a ideia do método é explorar caminhos no espaço de vizinhança de uma solução corrente qualquer, que permitam conectar parte das soluções de um dos elementos do conjunto elite C.

A maioria dos trabalhos da literatura mostram algoritmos que combinam o PR com outra técnica heurística. Reeves e Yamada (1998) desenvolveram um algoritmo PR com Algoritmo Genético para resolução do problema de sequênciamento de fluxo de produção. Laguna e Martí (1999), por sua vez, desenvolveram um algoritmo que combinava o PR com GRASP, o qual obtiveram grande sucesso. Já Resende e Werneck

[5] Diferentes soluções para um problema específico que somente são alcançadas por meio de alterações ou perturbações na área de varredura do algoritmo.

(2006) propuseram a resolução do UFLP (baseado no problema das p-medianas) por meio de um algoritmo que gera múltiplas soluções iniciais, combinado com um método de busca local e o PR.

Capítulo 3

Método de Decomposição de Benders aplicado ao TLUFLP

Para este trabalho escolheu-se utilizar o modelo TLUFLP proposto por Balinski (1964) e Manne (1964) (apresentado no capítulo 2) tendo como a única diferença, o acréscimo da informação da demanda d_i de cada um dos clientes $i \in I$. Neste sentido, os conceitos e definições que foram adotados anteriormente serão aproveitados nesta seção.

A informação da demanda do cliente i será incorporada ao parâmetro de custo c_{ijk} por meio da expressão $c_{ijk} = d_i(c_{ij} + c_{jk})$, no qual c_{ij} e c_{jk} representam respectivamente os custos entre os clientes $i \in I$ e as facilidades $j \in J$, e os custos entre as facilidades $j \in J$ e $k \in K$. O modelo matemático escolhido para este trabalho foi re-escrito conforme mostrado abaixo.

$$
\begin{aligned}
\min \quad & \sum_{k \in K} f_k z_k + \sum_{j \in J} a_j y_j + \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} c_{ijk} x_{ijk} & \\
\text{s.a.:} \quad & \sum_{j \in J} \sum_{k \in K} x_{ijk} = 1 & \forall\ i \in I \\
& y_j - \sum_{k \in K} x_{ijk} \geq 0 & \forall\ i \in I, j \in J \\
& z_k - \sum_{j \in J} x_{ijk} \geq 0 & \forall\ i \in I, k \in K \\
& x_{ijk} \geq 0 & \forall\ i \in I, j \in J, k \in K \\
& z_k \in \{0, 1\} & \forall\ k \in K \\
& y_j \in \{0, 1\} & \forall\ j \in J
\end{aligned}
$$

A escolha do método de Decomposição de Benders (Benders (1962)) na resolução do TLUFLP se deu principalmente pelas características básicas das variáveis e res-

trições presentes na formulação de Balinski (1964) e Manne (1964), que são: possuir duas variáveis complicantes z_k e y_j, para todo $k \in K$ e $j \in J$ respectivamente, que quando fixadas resultam em um subproblema que consiste em atribuir a cada $i \in I$ a configuração de menor custo entre as facilidades $j \in J$ e $k \in K$ ativas.

A aplicação do método de Decomposição de Benders no TLUFLP pode ser alcançada por meio da relaxação do Problema Mestre responsável pela fixação dos valores das variáveis z_k e y_j, referentes a decisão de instalar ou não uma facilidade, associada aos incrementos de cortes gerados por meio do Subproblema Dual, para cada um dos clientes $i \in I$.

Nesse sentido, vamos apresentar primeiramente o subproblema gerado a partir da fixação das variáveis z_k e y_j, conforme ilustrado abaixo:

$$\begin{aligned}
\min \quad & \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} c_{ijk}x_{ijk} & & \\
\text{s.a.:} \quad & \sum_{j\in J}\sum_{k\in K} x_{ijk} = 1 & \forall\, i \in I & \quad (3.1)\\
& -\sum_{k\in K} x_{ijk} \geq -\bar{y}_j & \forall\, i \in I, j \in J & \quad (3.2)\\
& -\sum_{j\in J} x_{ijk} \geq -\bar{z}_k & \forall\, i \in I, k \in K & \quad (3.3)\\
& x_{ijk} \geq 0 & \forall\, i \in I, j \in J, k \in K &
\end{aligned}$$

Como se trata de um método que trabalha com o subproblema em sua forma dual, associaremos as variáveis v_i, w_{ij} e u_{ik} às restrições de (3.1) à (3.3), respectivamente, para montagem do seguinte subproblema:

$$\begin{aligned}
\max \quad & \sum_{i\in I} v_i - \sum_{i\in I}\sum_{j\in J} \bar{y}_j w_{ij} - \sum_{i\in I}\sum_{k\in K} \bar{z}_k u_{ik} & & \quad (3.4)\\
\text{s.a.:} \quad & v_i - w_{ij} - u_{ik} \leq c_{ijk} & \forall\, i \in I, j \in J, k \in K & \quad (3.5)\\
& w_{ij} \geq 0 & \forall\, i \in I, j \in J & \\
& u_{ik} \geq 0 & \forall\, i \in I, k \in K & \\
& v_i \in \mathbb{R} & \forall\, i \in I &
\end{aligned}$$

Assim, é possível montar o Problema Mestre (PM) de Benders, baseado no Subproblema Dual (SD) e na criação de uma variável η que a cada iteração do algoritmo irá ajudar a compor um novo corte para o problema.

$$\min \sum_{k \in K} f_k z_k + \sum_{j \in J} a_j y_j + \eta \tag{3.6}$$

$$\text{s.a.: } \eta \geq \sum_{i \in I} \bar{v}_i - \sum_{i \in I} \sum_{j \in J} y_j \bar{w}_{ij} - \sum_{i \in I} \sum_{k \in K} z_k \bar{u}_{ik} \tag{3.7}$$

$$\sum_{j \in J} y_j \geq 1 \tag{3.8}$$

$$\sum_{k \in K} z_k \geq 1 \tag{3.9}$$

$$\eta \geq 0$$

$$z_k \in \{0, 1\} \qquad \forall\ k \in K$$

$$y_j \in \{0, 1\} \qquad \forall\ j \in J$$

As restrições (3.8) e (3.9) foram incorporadas ao PM para garantir que quando aplicado o algoritmo de Benders, haja sempre pelo menos uma facilidade instalada em cada nível. Essas duas restrições somente são necessárias no início da execução do algoritmo, quando o PM ainda não possui cortes de Benders.

A aplicação do método de Benders tradicional no TLUFLP não se mostra uma técnica muito eficiente quando aplicada na íntegra, porém a medida que algumas técnicas de aceleração do método são acrescentadas, espera-se que o tempo computacional de resolução dos problema diminua. Neste sentido a próxima seção apresentará algumas técnicas de aceleração do método de Benders implementadas para o TLUFLP.

3.1 Técnicas de Aceleração do Método de Benders

Dentre todas as técnicas de aceleração do método de Benders disponíveis na literatura o presente trabalho utilizou as seguintes abordagens para o TLUFLP:

- Utilizar uma técnica de Pré-aquecimento;
- Desenvolver uma versão de múltiplos cortes de Benders para TLUFLP;
- Utilização de uma única árvore de *Branch-and-Bound*.

3.1.1 Pré-aquecimento do método de Benders

A técnica de Pré-aquecimento do método de Benders aplicada ao TLUFLP foi feita através da relaxação linear das variáveis binárias y_j e z_k que passam a ser $\mathbb{R}^+$ para todo $j \in J$ e $k \in K$, respectivamente, nas quatro primeiras iterações do método de

Benders. A quantidade de iterações foi escolhida de forma empirica, a partir de alguns testes preliminares realizados.

A estrutura do algoritmo de Pré-aquecimento segue os mesmos princípios utilizados no algoritmo de Benders padrão, inclusive em relação a inserção de diferentes tipos de cortes (que serão apresentados posteriormente). Contudo vale ressaltar que o uso de tais cortes nas iterações de Pré-aquecimento não poderão ser feitos por meio da utilização das técnicas especializadas de resolução do Subproblema Dual (que também serão apresentadas posteriormente), uma vez que os algoritmos especializados levam em conta que as variáveis y_j e z_k sejam do tipo inteiras ($\mathbb{Z}^+ \in \{0, 1\}$). O método de Pré-aquecimento pode ser resumido pelo Algoritmo (3).

Algoritmo 3: Pré-aquecimento do método de Benders

```
Procedimento Pré-aquecimento(y_j, z_k, Iter)
início
    (y_j ∧ z_k) ∈ ℝ+
    enquanto (Iter ≥ 0) faça
        Resolva o PM
        Atualize o LB e os parâmetros do SD
        Resolva o SD
        Acrescente o corte de Benders ao PM
        Acrescente outro tipo de corte ao PM
        Iter ← Iter − 1
    fim
    (y_j ∧ z_k) ∈ 𝔹
fim
```

No Algoritmo (3) o parâmetro $Iter$ é o número de iterações de Pré-aquecimento, enquanto o LB representa a atualização do limite mínimo para o TLUFLP. O acréscimo dos outros tipos de cortes, que não o de Benders tradicional, serão apresentados ao longo deste capítulo. Por se tratar de um algoritmo ilustrativo considerou-se que a quantidade de iterações de Pré-aquecimento do método de Benders ($Iter$) não é suficiente para resolver o TLUFLP na otimalidade.

3.1.2 Múltiplos cortes de Benders para o TLUFLP

A ideia de geração de cortes múltiplos (*Multicuts*) para o TLUFLP se deu devido a estrutura do problema permitir que o SD seja decomposto para todos os clientes $i \in I$. Duas abordagens de múltiplos cortes serão apresentadas, a primeira delas corresponde a resolver o SD desagregado para todo $i \in I$ e a outra consiste em resolver o SD agregado. Em ambos a variável η será transformada em um somatório ($\sum_{i \in I} \eta_i$) na construção do PM.

Para montagem do corte desagregado para cada cliente $i \in I$, devemos transformar o SD em i subproblemas, conforme mostrado nas equações que seguem:

$$\begin{aligned}
&\max\ v_i - \sum_{j\in J} \bar{y}_j w_{ij} - \sum_{k\in K} \bar{z}_k u_{ik} && \\
&\text{s.a.: } v_i - w_{ij} - u_{ik} \leq c_{ijk} && \forall j \in J, k \in K \\
&\quad w_{ij} \geq 0 && \forall j \in J \\
&\quad u_{ik} \geq 0 && \forall k \in K \\
&\quad v_i \in \mathbb{R} &&
\end{aligned}$$

Assim, é possível montar um PM utilizando o SD desagregado para cada i como mostrado a seguir:

$$\min \sum_{k\in K} f_k z_k + \sum_{j\in J} a_j y_j + \sum_{i\in I} \eta_i \tag{3.10}$$

$$\text{s.a.: } \eta_i \geq \bar{v}_i - \sum_{j\in J} y_j \bar{w}_{ij} - \sum_{k\in K} z_k \bar{u}_{ik} \qquad \forall i \in I \tag{3.11}$$

$$\sum_{j\in J} y_j \geq 1$$

$$\sum_{k\in K} z_k \geq 1$$

$$\eta_i \geq 0$$

$$z_k \in \{0,1\} \qquad \forall\ k \in K$$

$$y_j \in \{0,1\} \qquad \forall\ j \in J$$

A montagem da versão de múltiplos cortes para o problema agregado utiliza a mesma estrutura apresentada no PM anterior, porém o SD é resolvido por inteiro e não mais desagregado. Por definição, resolver o SD agregado ou desagregado implica na mesma solução ótima, porém as variáveis u_{ik}, $k \in K$, e w_{ij}, $j \in J$, do SD podem assumir valores diferentes, o que pode gerar cortes mais fracos ou mais fortes.

É de conhecimento que a inserção de i cortes a cada iteração do algoritmo de Benders, pode não ser uma boa alternativa quando empregada em problemas de grande porte, uma vez que a convergência do método pode ficar mais lenta. Neste caso, outras formas de geração de múltiplos cortes podem ser utilizadas, como no caso do estabelecimento de regras (baseado na demanda dos clientes, ou em um número de cortes pré-definidos, ou até mesmo em uma combinação entre os dois) que permitam a

agregação parcial dos i cortes. Contudo, somente por meio de testes computacionais é possível descobrir se alguma técnica de múltiplos cortes é vantajosa quando aplicada na resolução do TLUFLP.

3.1.3 Uma única árvore de *Branch-and-Bound* no método de Benders

Neste trabalho optou-se pela utilização das funções *CutCallback* e a *LazyConstraint-Callback* (ambas do resolvedor CPLEX) combinadas com o método de Benders para conseguir montar uma única árvore de *Branch-and-Bound* para o PM.

A ideia de executar a função *CutCallback* parte do princípio que seja feita a expansão sobre todos os nós da árvore de *Branch-and-Bound* no qual o valor das variáveis y_j e z_k do PM são viáveis na região de viabilidade do TLUFLP, porém não inteiras viáveis. Já a *LazyConstraintCallback* terá o mesmo papel, mas somente para as variáveis inteiras, e que desta forma são candidatas a serem uma solução incumbente. Como se trata de um método recursivo, se um nó de *Branch-and-Bound* for explorado, ele não precisará ser explorado novamente durante as iterações de Benders.

Para este trabalho estabeleceu-se que o uso da função *CutCallback* seria controlado, ou seja, apenas as cinco primeiras soluções fracionárias do TLUFLP seriam exploradas. Essa quantidade, assim como no método de Pré-aquecimento, foi definida após a realização de testes computacionais preliminares.

3.2 Diferentes tipos de cortes aplicados ao TLUFLP

Como visto na literatura, o corte de Benders tradicional é dominado por outros cortes mais robustos que quando aplicados ao problema podem vir a trazer ganhos significativos em relação ao tempo computacional de resolução. Para este trabalho explorou-se quatro diferentes tipos, a saber: o Fechamento de Facilidades, o Pareto-Ótimo, o Pareto-Ótimo/Papadakos e o δ-Vizinhança. Em complementariedade é proposto também a resolução do SD utilizando dois algoritmos especializados diferentes.

3.2.1 Fechamento de Facilidades

Para montagem do corte Fechamento de Facilidades, ao qual chamaremos de CF (*Closing Facility*), para o TLUFLP serão adotadas notações semelhantes às apresentadas no capítulo de revisão da literatura, a começar pela utilização de C_z e C_y para representar as facilidades fechadas de primeiro ($k \in K$) e segundo ($j \in J$) nível, respectivamente, assim como O_z e O_y representarão as facilidades abertas. Neste contexto, duas novas

variáveis α_j e β_k foram criadas no intuito de representarem o custo mínimo total de serviço incorrido sobre todos os clientes $i \in I$ atendidos pelas facilidades $j \in O_y$ e $k \in O_z$.

Outras notações utilizadas para montagem do CF são descritas a seguir:

- $j(i)$: facilidade $j \in J$ que atende o cliente $i \in I$ na iteração atual de Benders;
- $k(i)$: facilidade $k \in K$ que atende o cliente $i \in I$ na iteração atual de Benders;
- $c_{(i)}$: custo de atendimento de $i \in I$ para a solução corrente do SD;
- $c^j_{(i)}$: melhor custo de atendimento de $i \in I$, desconsiderando $j = j(i)$;
- $c^k_{(i)}$: melhor custo de atendimento de $i \in I$, desconsiderando $k = k(i)$;
- n^i: quantidade de clientes $i \in I$;
- n^j: quantidade de facilidades $j \in J$;
- n^k: quantidade de facilidades $k \in K$.

Primeiramente, consideremos que se uma facilidade $j \in O_y$, passa a ser do tipo C_y, os clientes i que eram atendidos por j, precisarão ser atendidos por outra facilidade $j \in J$ com custo $c^j_{(i)}$ (ilustrado na Figura (3.1)). O mesmo vale nos casos em que uma facilidade $k \in O_z$, passa a ser do tipo C_z, conforme mostrado nas seguintes equações:

$$c^j_{(i)} = \min\{c_{iqk} : 1 \leq q \leq n^j \wedge q \neq j(i)\}$$
$$c^k_{(i)} = \min\{c_{ijq} : 1 \leq q \leq n^k \wedge q \neq k(i)\}$$

Encontrar os valores de $c^j_{(i)}$ e $c^k_{(i)}$ através das equações anteriores pode ser trivial em relação aos conceitos, porém na montagem do algoritmo de Benders essa etapa pode ser custosa se considerarmos que a cada iteração do método esses valores precisam ser recalculados.

Neste sentido, uma alternativa proposta foi a criação de um procedimento que evitasse esse recalculo do ponto de vista computacional. Utilizou-se uma matriz m de inteiros positivos $\mathbb{Z}^+$ de dimensão n^i por 6. Para cada linha i (referente a cada $i \in I$) da matriz m, as duas primeiras posições representariam o par de facilidades jk $j \in J$ e $k \in K$ de menor custo c_{ijk}. Já terceira e quarta posição representariam o segundo par jk de menor custo c_{ijk}. As duas últimas posições da matriz m somente seriam necessárias nos casos em que os dois pares jk de menor custo tivessem em sua formação a repetição de uma das facilidades j ou k, ao qual seria armazenado o par

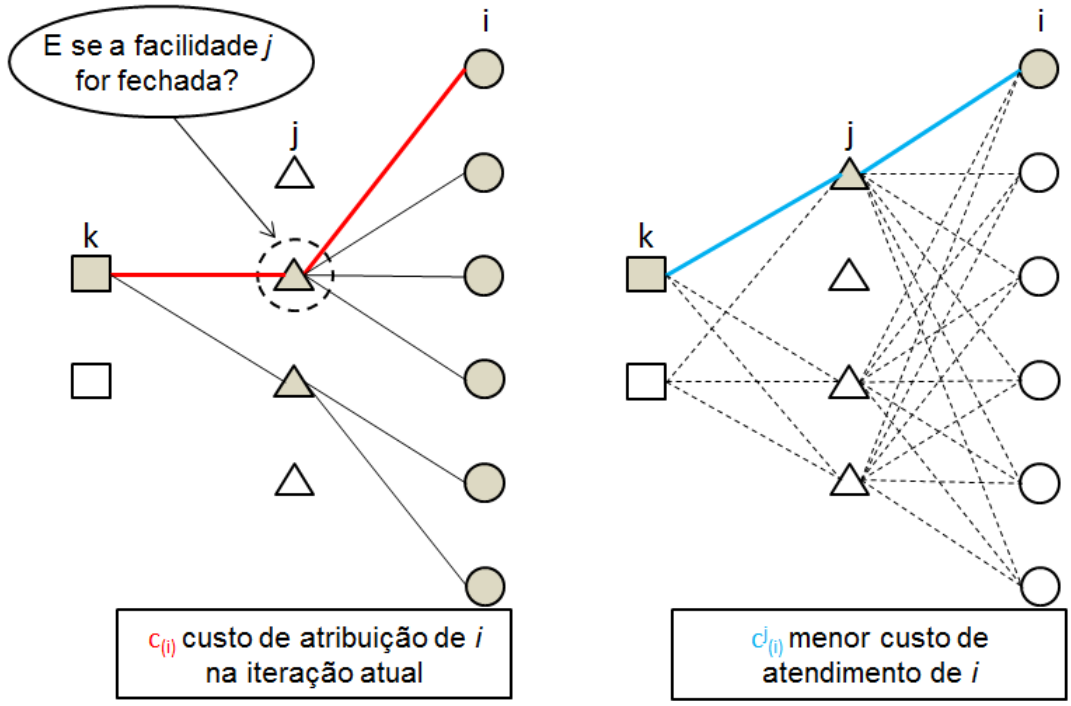

Figura 3.1: Ideia para montagem do corte CF

jk de menor custo c_{ijk} desconsiderando a facilidade (j ou k) que se repete nos dois primeiros pares jk.

Para exemplificar a necessidade das últimas duas posições de m, consideremos os dois pares de menor custo, $j = 1 \wedge k = 2$ e $j = 1 \wedge k = 3$, para o cliente $i = 4$ ($n^i \geq n^j \geq n^k \geq 6$). A facilidade $j = 1$ faz parte dos dois pares iniciais, portanto desde que em alguma iteração qualquer de Benders a facilidade $j = 1$ esteja atendendo o cliente $i = 4$, então nenhum dos dois pares respeita as equações para o cálculo de $c^j_{(4)}$. Então o par jk de menor custo seria aquele que desconsiderasse $j = 1$, como por exemplo, $j = 5 \wedge k = 6$.

Voltando ao processo de construção do corte CF, apresentaremos agora as variáveis σ^j_i e σ^k_i referentes às variações nos custos de serviço de todos os clientes $i \in I$, e que respeitam as seguintes equações:

$$\sigma^j_i = \max\{c^j_{(i)} - c_{(i)},\ 0\}$$
$$\sigma^k_i = \max\{c^k_{(i)} - c_{(i)},\ 0\}$$

Sempre que $\sigma^j_i > 0$, o cliente $i \in I$ sofrerá um aumento no custo de serviço de pelo menos σ^j_i, caso a facilidade $j(i)$ seja fechada. Neste contexto, se somarmos todos os σ^j_i referentes a todos os clientes $i \in I$ atendidos por uma mesma facilidade $j \in O_y$

poderemos chegar ao valor de α_j. O mesmo vale para os $\sigma_i^k > 0$ em relação aos $k \in O_z$ e a β_k, conforme mostrado nas equações a seguir:

$$\alpha_j = \sum_{i \in I} \{\sigma_i^j : 1 \leq i \leq n^i \wedge j = j(i)\}$$
$$\beta_k = \sum_{i \in I} \{\sigma_i^k : 1 \leq i \leq n^i \wedge k = k(i)\}$$

Reunindo as equações apresentadas ao longo desta subseção é possível montar o corte CF como apresentado na equação (3.12).

$$\eta \geq \sum_{i \in I} \bar{v}_i + \sum_{j \in O_y} (1 - y_j)\alpha_j - \sum_{i \in I} \sum_{j \in C_y} y_j \bar{w}_{ij} + \sum_{k \in O_z} (1 - z_k)\beta_k - \sum_{i \in I} \sum_{k \in C_z} z_k \bar{u}_{ik} \quad (3.12)$$

É possível notar que a estrutura do corte tradicional (3.7) é bem parecida com o CF (3.12), com a diferença do acréscimo das informações do fechamento de facilidades. Neste sentido, espera-se que seja possível a montagem de uma versão de múltiplos cortes CF, seguindo os mesmos passos apresentados na seção sobre múltiplos cortes deste capítulo. A equação (3.13) ilustra a estrutura dos múltiplos cortes de CF para cada cliente $i \in I$.

$$\eta_i \geq \bar{v}_i + (1 - y_{j(i)})\sigma_i^j - \sum_{j \in C_y} y_j \bar{w}_{ij} + (1 - z_{k(i)})\sigma_i^k - \sum_{k \in C_z} z_k \bar{u}_{ik} \quad \forall i \in I \quad (3.13)$$

Os parâmetros σ_i^j e σ_i^k passam a fazer parte do corte CF em substituição aos parâmetros α_j e β_k, respectivamente. Os somatórios referentes às facilidades abertas (O_y e O_z) deixam de existir, uma vez que apenas um único par de facilidades (uma de primeiro e outra de segundo nível) deverão atender a cada um dos clientes $i \in I$. Assim, a variável $y_{j(i)}$ representa a facilidade $j \in O_y$ que atende ao cliente i, assim como $z_{k(i)}$ indica a facilidade $k \in O_z$ que atende ao mesmo cliente i.

3.2.2 δ-Vizinhança

A estrutura do corte δ-Vizinhança é bem semelhante ao CF, com a diferença que são acrescidas variáveis que estão diretamente relacionadas ao que chamaremos de vizinhanças interiores $N_i^0(\delta_j)$ e $N_i^0(\theta_k)$ dos clientes $i \in I$. O conceito de $N_i^0(\delta_j)$ e $N_i^0(\theta_k)$ é dado pelo conjunto de facilidades $j \in J$ e $k \in K$, respectivamente, que respeitam as equações que seguem:

$$N_i^0(\delta_j) = \{j : c_{ijk} < \bar{v}_i + \delta_j\}$$
$$N_i^0(\theta_k) = \{k : c_{ijk} < \bar{v}_i + \theta_k\}$$

O valor de $\bar{v}_i$ é igual ao menor custo para se atender ao cliente $i \in I$ e δ_j é o incremento máximo que cada uma das facilidades $j \in J$ pode alcançar para pertencer a vizinhança do cliente $i \in I$ na iteração atual, o mesmo vale para θ_k em relação as facilidades $k \in K$.

Levando em consideração que os valores de δ_j e θ_k sejam todos iguais a zero, é possível montar um corte baseado no conceito de vizinhança, com uma estrutura muito parecida com o CF e que utilize dois parâmetros de penalização ρ_j e ϱ_k que são calculados por meio das seguintes equações:

$$\sum_{i \in I} \bar{v}_i = \rho_j \qquad \forall j \in O_y$$
$$\sum_{i \in I} \bar{v}_i = \varrho_k \qquad \forall k \in O_z$$

Os parâmetros ρ_j e ϱ_k representam os somatórios de todos os custos de serviço oferecidos pelas facilidades $j \in O_y$ e $k \in O_z$, respectivamente. Assim, podemos construir um corte δ-Vizinhança conforme mostrado em sequência.

$$\eta \geq \sum_{i \in I} \bar{v}_i + \sum_{j \in O_y} (1 - y_j)\rho_j - \sum_{j \in C_y} \sum_{i \in I} \bar{w}_{ij} + \sum_{k \in O_z} (1 - z_k)\varrho_k - \sum_{k \in C_z} \sum_{i \in I} \bar{u}_{ik}$$

Para construir um corte que domine o CF serão acrescentadas algumas notações e considerações referentes a estrutura do corte. Em primeiro lugar, devemos ter em mente que a expansão das vizinhanças para cada cliente $i \in I$ deve ser feita apenas sobre as facilidades do tipo $j \in O_y$ e $z \in O_z$, ou seja $\delta_j = 0$ e $\theta_k = 0$ para todas a facilidades $j \in C_y$ e $z \in C_z$. Em se tratando dos casos em que os valores de δ_j e θ_k podem ser maiores ou iguais a zero, duas restrições precisam ser atendidas: Toda facilidade fechada ($j \in C_y$ e $z \in C_z$) pode estar em no máximo uma única vizinhança de facilidade aberta ($j \in O_y$ e $z \in O_z$); O interior de toda vizinhança pode conter apenas uma única facilidade aberta, que em linguagem matemática, pode ser explicada pelas equações de (3.14) e (3.15).

$$\bar{v}_i + \delta_j = v_i \leq \min\{c_{ijk} : j \in O_y \wedge j \neq j(i)\} \qquad \forall\ i \in I \qquad (3.14)$$

$$\bar{v}_i + \theta_k = v_i \leq \min\{c_{ijk} : k \in O_z \wedge k \neq k(i)\} \qquad \forall\ i \in I \qquad (3.15)$$

Pode-se dizer que o limite de expansão da variável v_i é dado pela diferença de distância/custo entre as duas facilidades abertas mais próximas do cliente $i \in I$, em cada um dos dois níveis na iteração de Benders atual. Para a construção de um corte único que respeite as equações (3.14) e (3.15) vamos substituir os parâmetros δ_j e θ_k pelos parâmetros $\bar{\delta}$ e $\bar{\theta}$ que assumem valores fixos de expansão sobre todas as vizinhanças dos clientes $i \in I$. Os valores de $\bar{\delta}$ e $\bar{\theta}$ são obtidos ao selecionar os menores valores δ_j e θ_k, respectivamente, o que impede que duas facilidades abertas estejam em mais de uma vizinhança.

Por definição, pelo fato de estarmos incorporando os valores de $\bar{\delta}$ e $\bar{\theta}$ ao corte, é necessário que se faça um ajuste em relação às variáveis do SD w_{ij} e u_{ik} (parâmetros $\bar{w}_{ij}$ e $\bar{u}_{ik}$ no PM). Esse ajuste ou fator de compensação será representado pelas variáveis Δ_j e Θ_k, que por sua vez devem sempre serem menores ou iguais aos valores $\bar{\delta}$ e $\bar{\theta}$, respectivamente, para todo $j \in C_y$ e $k \in C_z$. Os valores dessas novas variáveis podem ser obtidos pelas equações que seguem:

$$\Delta_j = (\bar{w}_{ij}) - (\bar{w}^{\delta}_{ij}) \qquad \forall\ j \in C_y$$

$$\Theta_k = (\bar{u}_{ik}) - (\bar{u}^{\theta}_{ik}) \qquad \forall\ k \in C_z$$

onde

$$\bar{w}^{\delta}_{ij} \geq \max_{k \in O_z} \{\max\{0, v^t_i + \bar{\delta} - c_{ijk}\}\} \qquad \forall\ j \in C_y$$

$$\bar{u}^{\theta}_{ik} \geq \max_{j \in O_y} \{\max\{0, v^t_i + \bar{\theta} - c_{ijk}\}\} \qquad \forall\ k \in C_z$$

O cálculo das variáveis $\bar{w}^{\delta}_{ij}$ e $\bar{u}^{\theta}_{ik}$ pode ser feito por meio de um algoritmo especializado (descrito na seção 3.3), ou por meio de um resolvedor comercial. Por fim, é possível chegar ao corte δ-Vizinhança, descrito pela equação (3.16):

$$\eta \geq \sum_{i \in I} \bar{v}_i + \sum_{j \in O_y} (1 - y_j)(\rho_j + \bar{\delta}) - \sum_{j \in C_y} ((\sum_{i \in I} \bar{w}_{ij}) + \Delta_j) y_j + \sum_{k \in O_z} (1 - z_k)(\varrho_k + \bar{\theta})$$
$$- \sum_{k \in C_z} ((\sum_{i \in I} \bar{u}_{ik}) + \Theta_k) z_k \tag{3.16}$$

Espera-se que, assim como o corte CF, seja possível a montagem de um corte δ-Vizinhança desagregado para cada clientes $i \in I$, até mesmo porque a expansão das vizinhanças é feita para cada cliente. Infelizmente, apenas a teoria relacionada ao corte δ-Vizinhança foi desenvolvida a tempo de ser mostrada neste trabalho, sendo a sua implementação e validação deixada para trabalhos futuros.

3.2.3 Pareto-Ótimo

Em cada iteração do método de Benders, o PM fornece uma configuração específica das variáveis y_j para todo $j \in J$ e z_k para todo $k \in K$, que por sua vez, possibilita resolver o SD e encontrar múltiplas soluções ótimas. Dentre o conjunto de soluções ótimas para o SD, o corte Pareto-Ótimo (POt) é aquele que pertence ao conjunto de soluções e domina todos os outros cortes existentes. Levando em consideração que as variáveis v_i ($\forall\ i \in I$), w_{ij} ($\forall\ i \in I \wedge j \in J$) e u_{ik} ($\forall\ i \in I \wedge k \in K$) referentes às restrições (3.5) do problema do SD podem assumir diferentes valores sem que o valor da função objetivo (3.4) seja alterado, temos a seguinte condição:

$$\sum_{i \in I} v_i^* - \sum_{i \in I} \sum_{j \in J} \bar{y}_j w_{ij}^* - \sum_{i \in I} \sum_{k \in K} \bar{z}_k u_{ik}^* \geq \sum_{i \in I} v_i - \sum_{i \in I} \sum_{j \in J} \bar{y}_j w_{ij} - \sum_{i \in I} \sum_{k \in K} \bar{z}_k u_{ik}$$

As variáveis v_i^*, w_{ij}^* e u_{ik}^* indicam a melhor configuração possível que as variáveis do SD podem assumir para montagem do corte POt na iteração atual de Benders. Além disso, outra condição importante para construção do corte POt é a definição dos chamados *core points* y_j^0 e z_k^0, que precisam pertencer ao interior relativo da casca convexa $\mathbf{ir}(Y^c)$ e $\mathbf{ir}(Z^c)$, respectivamente, do TLUFLP, ou seja Y^c e Z^c pertencem ao conjunto de soluções viáveis para o problema.

Sabendo que o SD pode ser desagregado para cada um dos clientes $i \in I$, podemos portanto montar um novo subproblema dual que utilize a informação das variáveis de *core point*, conforme mostrado a seguir:

$$
\begin{aligned}
&\max\ v_i - \sum_{j\in J} y_j^0 w_{ij} - \sum_{k\in K} z_k^0 u_{ik} \\
&\text{s.a.: } v_i - \sum_{j\in J} \bar{y}_j w_{ij} - \sum_{k\in K} \bar{z}_k u_{ik} = \bar{v}_i \\
&\quad v_i - w_{ij} - u_{ik} \leq c_{ijk} && \forall j \in J, k \in K \\
&\quad w_{ij} \geq 0 && \forall j \in J \\
&\quad u_{ik} \geq 0 && \forall k \in K \\
&\quad v_i \in \mathbb{R}
\end{aligned} \tag{3.17}
$$

A partir do novo subproblema dual podemos associar o valor da função objetivo a uma função linear por partes da variável v_i, no qual a equação seguinte é respeitada.

$$
v_i - \sum_{j\in O_y} w_{ij} - \sum_{k\in O_z} u_{ik} = \bar{v}_i = c_{(i)}
$$

A equação anterior somente é valida pelo fato de que em qualquer solução ótima do subproblema, $w_{ij} = 0$ e $u_{ik} = 0$ para todas as facilidades abertas de ambos os níveis. Assim, é possível que seja feita uma substituição na função objetivo do subproblema dual da seguinte forma:

$$
\max\ \bar{v}_i - \sum_{j\in J} (\bar{y}_j - y_j^0) w_{ij} - \sum_{k\in K} (\bar{z}_k - z_k^0) u_{ik}
$$

Neste sentido, para montarmos uma função da variável v_i, $f(v_i)$ ((3.18)), é necessário que exista um coeficiente $\varepsilon_j^1 \equiv (\bar{y}_j - y_j^0)$ de w_{ij} e outro $\varepsilon_k^2 \equiv (\bar{z}_k - z_k^0)$ de u_{ik}, ambos menores ou iguais a zero. Além disso, precisaremos também dos coeficientes $\varepsilon_{j(i)}^1$ e $\varepsilon_{k(i)}^2$ associados aos clientes $i \in I$ atendidos pelas facilidades $j \in O_y$ e $k \in O_z$, respectivamente.

$$
f(v_i) = \bar{v}_i + \varepsilon_{j(i)}^1 (v_i - \bar{v}_i) + \sum_{j\in C_y} \varepsilon_j^1 w_{ij} + \varepsilon_{k(i)}^2 (v_i - \bar{v}_i) + \sum_{k\in C_z} \varepsilon_j^2 u_{ik} \tag{3.18}
$$

Utilizando os mesmos conceitos que Magnanti e Wong (1990), dois limites (L_i^1 e L_i^2) de expansão para a variável v_i foram estipulados, um para cada nível de facilidades, onde $\bar{v}_i \leq v_i \leq \min(L_i^1, L_i^2)$. O Algoritmo (4) mostra como são calculadas as variáveis

do SD para montagem do corte POt do TLUFLP.

Algoritmo 4: Algoritmo para construção do corte Pareto-Ótimo

início
  para todo $(j \in J)$ **faça**
    $\varepsilon_j^1 = (\bar{y}_j - y_j^0)$
  fim
  para todo $(k \in K)$ **faça**
    $\varepsilon_k^2 = (\bar{z}_k - z_k^0)$
  fim
  para todo $(i \in I)$ **faça**
    $v_i^1 = \min\{c_{ijk} : \ j \in O_y \wedge k \in K\}$
    $L_i^1 = \min\{c_{ijk} : \ j \in O_y \wedge j \neq j(i)\}$
    enquanto $(v_i^1 \neq v_i^{1*})$ **faça**
      $B \leftarrow \{j \in C_y : c_{ijk} \leq v_i^1\}$
      $s \leftarrow \varepsilon_{j(i)}^1 + \sum_{j \in B} \varepsilon_j^1$
      se $(s \leq 0)$ **então** $v_i^{1*} \leftarrow v_i^1$
      se $(s > 0 \wedge B = C_y)$ **então** $v_i^{1*} \leftarrow v_i^1 = L_i^1$
      $\theta_i \leftarrow \min\{c_{ijk} : j \in C_y \wedge j \notin B\}$
      se $(\theta_i > L_i^1)$ **então** $(v_i^1 \leftarrow L_i^1) \wedge (v_i^{1*} \leftarrow L_i^1)$ **senão** $v_i^1 \leftarrow \theta_i$
    fim
    $v_i^2 = \min\{c_{ijk} : \ k \in O_z \wedge j \in J\}$
    $L_i^2 = \min\{c_{ijk} : \ k \in O_z \wedge k \neq k(i)\}$
    enquanto $(v_i^2 \neq v_i^{2*})$ **faça**
      $B \leftarrow \{k \in C_z : c_{ijk} \leq v_i^2\}$
      $s \leftarrow \varepsilon_{k(i)}^2 + \sum_{k \in B} \varepsilon_k^2$
      se $(s \leq 0)$ **então** $v_i^{2*} \leftarrow v_i^2$
      se $(s > 0 \wedge B = C_z)$ **então** $v_i^{2*} \leftarrow v_i^2 = L_i^2$
      $\theta_i \leftarrow \min\{c_{ijk} : k \in C_z \wedge k \notin B\}$
      se $(\theta_i > L_i^2)$ **então** $(v_i^2 \leftarrow L_i^2) \wedge (v_i^{2*} \leftarrow L_i^2)$ **senão** $v_i^2 \leftarrow \theta_i$
    fim
    $v_i \leftarrow \min\{v_i^1, v_i^2\}$
    Resolver o SD com v_i *fixado para encontrar* w_{ij} *e* u_{ik}
  fim
fim

No Algoritmo (4), primeiramente deve-se calcular os coeficientes ε_j^1 e ε_k^2, por meio dos parâmetros $\bar{y}_j$ e $\bar{z}_k$ do SD e suas respectivas variáveis de *core point*, y_j^0 e z_k^0. Em seguida, para cada cliente i, deve-se estabelecer um valor inicial para a variável v_i^1 e o parâmetro L_i^1, ambos referentes ao segundo nível de facilidades. Feito isso, a variável v_i^1 será expandida de forma progressiva (controlada pelo parâmetro s), até que de v_i^1 assuma o melhor valor possível, representado por v_i^{1*}. Esse mesmo procedimento é feito para o primeiro nível de facilidades, por meio da variável v_i^2 e o parâmetro L_i^2. Ao final do algoritmo, a variável v_i receberá o $\min\{v_i^1, v_i^2\}$, o que permitirá calcular o valor das

variáveis w_{ij} e u_{ik}, por meio do SD.

A montagem do corte POt tem a mesma estrutura do corte de Benders Tradicional (3.7), porém o valor das variáveis do SD podem assumir valores diferentes, de forma a criar um corte superior.

3.2.4 Pareto-Ótimo/Papadakos

O Subproblema Dual de Papadakos (SDPk) pode ser montado segundo a mesma estrutura do SD, com a diferença da substituição das variáveis $z_k(k \in K)$ e $y_j(j \in J)$ do PM pelas variáveis $z_k^0(k \in K)$ e $y_j^0(j \in J)$ respectivamente, calculadas por meio do uso dos *core points*. Assim é possível escrever o SDPk sob a seguinte estrutura:

$$\begin{aligned}
\max \ & \sum_{i \in I} v_i - \sum_{i \in I} \sum_{j \in J} \bar{y}_j^0 w_{ij} - \sum_{i \in I} \sum_{k \in K} \bar{z}_k^0 u_{ik} & \\
\text{s.a.: } & v_i - w_{ij} - u_{ik} \leq c_{ijk} & \forall\ i \in I, j \in J, k \in K \\
& w_{ij} \geq 0 & \forall\ i \in I, j \in J \\
& u_{ik} \geq 0 & \forall\ i \in I, k \in K \\
& v_i \in \mathbb{R} & \forall\ i \in I
\end{aligned}$$

Para atualização dos *core points* é necessária a criação de um parâmetro λ tal que $0 \leq \lambda \leq 1$. Como sugerido em Papadakos (2008), também utilizamos $\lambda = \frac{1}{2}$. A atualização de $z_k^0(k \in K)$ e $y_j^0(j \in J)$ em cada iteração t é feita da seguinte forma:

$$\begin{aligned}
z_k^{0(t)} &= \lambda z_k^{0(t-1)} + (1-\lambda) z_k^{(t)} & \forall k \in K \\
y_j^{0(t)} &= \lambda y_j^{0(t-1)} + (1-\lambda) y_j^{(t)} & \forall j \in J
\end{aligned}$$

Essas equações tratam-se de combinações convexas entre o valor das variáveis $y_j^{(t)}$ e $z_k^{(t)}$ do PM da iteração atual (t) de Benders e os parâmetros $\bar{y}_j^{0(t-1)}$ e $\bar{z}_k^{0(t-1)}$ do SDPk da iteração anterior $(t-1)$, ponderadas pelo valor de λ adotado. Neste contexto, a cada iteração t de Benders o valor das variáveis *core points* z_k^0 e y_j^0 assumem uma nova configuração, o que nos permite resolver o SDPk e montar o corte que segue.

$$\eta \geq \sum_{i \in I} \bar{v}_i - \sum_{i \in I} \sum_{j \in J} y_j^0 \bar{w}_{ij} - \sum_{i \in I} \sum_{k \in K} z_k^0 \bar{u}_{ik}$$

É possível também que o SDPk seja resolvido de forma desagregada para cada

$i \in I$ utilizando as mesmas variáveis de *core points*, conforme mostrado no seguinte subproblema:

$$\begin{aligned}
&\max\ v_i - \sum_{j \in J} \bar{y}_j^0 w_{ij} - \sum_{k \in K} \bar{z}_k^0 u_{ik} \\
&\text{s.a.: } v_i - w_{ij} - u_{ik} \leq c_{ijk} && \forall j \in J, k \in K \\
&\quad w_{ij} \geq 0 && \forall j \in J \\
&\quad u_{ik} \geq 0 && \forall k \in K \\
&\quad v_i \in \mathbb{R}
\end{aligned}$$

Assim, é possível montar i cortes para cada um dos clientes i, como mostrado nas equações (3.19):

$$\eta_i \geq \bar{v}_i - \sum_{j \in J} y_j^0 \bar{w}_{ij} - \sum_{k \in K} z_k^0 \bar{u}_{ik} \qquad \forall i \in I \tag{3.19}$$

Vale ressaltar que para este trabalho o valor inicial adotado para as variáveis de *core point*, y_j^0 e z_k^0, tal como em Papadakos (2008), fixado em $\frac{1}{2}$ para todas facilidades $j \in J$ e $k \in K$, respectivamente.

3.3 Algoritmos Especializados para resolução do SD - AESD

Em toda iteração t do método de Decomposição de Benders padrão obtém-se dois vetores (y^t, z^t) referentes à solução do PM, que são parâmetros a serem usados no SD e guardam a informação sobre quais facilidades estão ativas e quais não.

A resolução do subproblema do TLUFLP desta forma pode ser vista como um problema de atribuição, no qual a solução ótima consiste em associar o par de facilidades ($j \in J$, $k \in K$) que atendem a cada cliente $i \in I$ com o menor custo. Assim, resolver esse problema por inspeção pode resultar em economia de tempo computacional. Para apresentar o método de resolução do AESD, algumas notações foram definidas:

$$O_y = \{j \in J : y_j^t = 1\} \text{ e } O_z = \{k \in K : z_k^t = 1\}$$

$$C_y = \{j \in J : y_j^t = 0\} \text{ e } C_z = \{k \in K : z_k^t = 0\}$$

Ao analisarmos a função objetivo do SD (3.4), nota-se que a função atingirá seu valor ótimo quando o somatório de todos os v_i for máximo e o as parcelas restantes forem zero. Assim, utilizando os conceitos de complementaridade de folga que são:

$$u_{ik}^t \left(z_k^t - \sum_{j \in J} x_{ijk}^t \right) = 0 \qquad \forall\ i \in I, k \in K$$

$$w_{ij}^t \left(y_j^t - \sum_{k \in K} x_{ijk}^t \right) = 0 \qquad \forall\ i \in I, j \in J$$

$$x_{ijk}^t (c_{ijk} - v_i^t + u_{ik}^t + w_{ij}^t) = 0 \qquad \forall\ i \in I, j \in J, k \in K$$

e explorando de maneira mais ampla as restrições (3.5), concluiremos que:

$$v_i^t = \min \{c_{ijk} : j \in O_y, k \in O_k\} \qquad \forall\ i \in I \tag{3.20}$$

$$u_{ik}^t = 0 \qquad \forall\ k \in O_z \tag{3.21}$$

$$w_{ij}^t = 0 \qquad \forall\ j \in O_y \tag{3.22}$$

$$u_{ik}^t \geq \max_{j \in O_y} \{\max \{0, v_i^t - c_{ijk}\}\} \qquad \forall\ k \in C_z \tag{3.23}$$

$$w_{ij}^t \geq \max_{k \in O_z} \{\max \{0, v_i^t - c_{ijk}\}\} \qquad \forall\ j \in C_y \tag{3.24}$$

Através das equações (3.20) à (3.24) é possível encontrar uma solução ótima para o SD de Benders sem a utilização do método SIMPLEX. As equações (3.23) e (3.24) não são calculadas de forma automática, o que demandou o desenvolvimento dos Algoritmos (5) e (6) para resolução do SD, que estão disponíveis no fim desta seção.

A diferença principal entre os dois algoritmos se dá única e exclusivamente na forma de calcular o valor das variáveis w_{ij}^t e u_{ik}^t no caso em que as facilidades y_j e z_k são do tipo C_y e C_z, respectivamente, para um mesmo $i \in I$.

Em relação ao Algoritmo (5), uma possível forma de se otimizar o valor das variáveis w_{ij}^t para todo $j \in C_y$ e u_{ik}^t para todo $k \in C_z$ é a minimização da soma dessas variáveis, por meio da resolução das seguintes equações:

$$u_{ik}^t = \max_{j \in O_y} \{\max \{0, v_i^t - c_{ijk}\}\} + \bar{u}_{ik} \qquad \forall\ k \in C_z$$

$$w_{ij}^t = \max_{k \in O_z} \{\max \{0, v_i^t - c_{ijk}\}\} + \bar{w}_{ij} \qquad \forall\ j \in C_y$$

onde $\bar{w}_{ij}$ e $\bar{u}_{ik}$ podem ser obtidos através do seguinte problema auxiliar:

$$
\begin{aligned}
\min \quad & \sum_{j \in C_y} \bar{w}_{ij} + \sum_{k \in C_y} \bar{u}_{ik} & \\
\text{s.a.:} \quad & \bar{w}_{ij} + \bar{u}_{ik} \geq \bar{c}_{ijk} & \forall j \in C_y, k \in C_z \\
& \bar{w}_{ij} \geq 0 & \forall j \in C_y \\
& \bar{u}_{ik} \geq 0 & \forall k \in C_z
\end{aligned}
$$

e $\bar{c}_{ijk}$ são calculados como segue:

$$
\begin{aligned}
\bar{c}_{ijk} &= v_i^t - c_{ijk} && \forall j \in O_y, k \in O_z \\
\bar{c}_{ijk} &= v_i^t - c_{ijk} - \max_{j \in O_y}\{\max\{0, v_i^t - c_{ijk}\}\} && \forall j \in O_y, k \in C_z \\
\bar{c}_{ijk} &= v_i^t - c_{ijk} - \max_{k \in O_z}\{\max\{0, v_i^t - c_{ijk}\}\} && \forall j \in C_y, k \in O_z \\
\bar{c}_{ijk} &= v_i^t - c_{ijk} - \max_{j \in O_y}\{\max\{0, v_i^t - c_{ijk}\}\} - \max_{k \in O_z}\{\max\{0, v_i^t - c_{ijk}\}\} && \forall j \in C_y, k \in C_z
\end{aligned}
$$

Ao montarmos o dual do problema auxiliar é possível notar que obtemos um conhecido problema da literatura citado por Wolsey (1998), o *Bipartite Maximum-Weight Matching Problem*. Segundo Wolsey (1998) esse problema pode ser resolvido eficientemente por meio da aplicação do método Húngaro (*Hungarian method*).

Uma alternativa diferente do método Húngaro, em que foram obtidos bons resultados para resolução de problemas de atribuição lineares foi apresentada em um trabalho de Jonker e Volgenant (1987), no qual eles desenvolveram um algoritmo chamado de LAPJV (*Jonker-Volgenant for Linear Assignment Problem*), atualmente mais conhecido como JVC (*Jonker-Volgenant-Castanon*). O JVC resolve problemas de atribuição lineares através do dual dos problemas originais.

Neste trabalho utilizou-se um código em C++ do JVC, disponível na página `http://www.assignmentproblems.com/lapjv.htm`, para a resolução do problema auxiliar, por se tratar de um algoritmo rápido e que resolve o problema de atribuição pelo dual.

Já em relação ao Algoritmo (6), cinco maneiras simples de integralização do restante das variáveis w_{ij}^t e u_{ik}^t, quando necessário, foram criadas de forma a manter a complementaridade de folga. Todas elas utilizam a ideia de que se $(v_i^t - c_{ijk}) - (u_{ik}^t + w_{ij}^t) = df > 0$, devemos fazer com que essa diferença df seja atribuída a uma das duas variáveis, ou distribuída entre elas.

As cinco formas testadas podem ser resumidas em: Distribuir $\frac{df}{2}$ para w_{ij}^t e u_{ik}^t; Somar df em w_{ij}^t; Somar df em u_{ik}^t; Somar df em w_{ij}^t caso $(w_{ij}^t > u_{ik}^t)$, caso contrário somar em u_{ik}^t; Somar df em u_{ik}^t caso $(w_{ij}^t > u_{ik}^t)$, caso contrário somar em w_{ij}^t.

Algoritmo 5: Resolução do AESD-1

início
  para todo $(i \in I)$ **faça**
    $v_i^t \leftarrow \min\{c_{ijk}\} \quad \forall\ j \in O_y,\ k \in O_k$
    $u_{ik}^t \leftarrow 0 \quad \forall\ k \in O_z$
    $w_{ij}^t \leftarrow 0 \quad \forall\ j \in O_y$
    $u_{ik}^t \leftarrow \max\{0, v_i^t - c_{ijk}\} \quad \forall\ j \in O_y,\ k \in C_z$
    $w_{ij}^t \leftarrow \max\{0, v_i^t - c_{ijk}\} \quad \forall\ k \in O_z,\ j \in C_y$
    $JVC(\bar{u}_{ik}^t,\ \bar{w}_{ij}^t) \quad \forall\ k \in C_z,\ j \in C_y$
    $w_{ij}^t \leftarrow w_{ij}^t + \bar{w}_{ij}^t \quad \forall\ k \in C_z,\ j \in C_y$
    $u_{ik}^t \leftarrow u_{ik}^t + \bar{u}_{ik}^t \quad \forall\ k \in C_z,\ j \in C_y$
  fim
fim

Algoritmo 6: Resolução do AESD-2

início
  para todo $(i \in I)$ **faça**
    $v_i^t \leftarrow \min\{c_{ijk}\} \quad \forall\ j \in O_y,\ k \in O_k$
    $u_{ik}^t \leftarrow 0 \quad \forall\ k \in O_z$
    $w_{ij}^t \leftarrow 0 \quad \forall\ j \in O_y$
    $u_{ik}^t \leftarrow \max\{0, v_i^t - c_{ijk}\} \quad \forall\ j \in O_y,\ k \in C_z$
    $w_{ij}^t \leftarrow \max\{0, v_i^t - c_{ijk}\} \quad \forall\ k \in O_z,\ j \in C_y$
    para todo $(k \in C_z \wedge j \in C_y)$ **faça**
      se $((u_{ik}^t + w_{ij}^t) < (v_i^t - c_{ijk}))$ **então**
        $df \leftarrow (v_i^t - c_{ijk}) - (u_{ik}^t + w_{ij}^t)$
        selecione $\{1 \vee 2 \vee 3 \vee 4 \vee 5\}$(// O mesmo $\forall$ i) **faça**
          1. $u_{ik}^t \leftarrow u_{ik}^t + df/2$
             $w_{ij}^t \leftarrow w_{ij}^t + df/2$
          2. $u_{ik}^t \leftarrow u_{ik}^t + df$
          3. $w_{ij}^t \leftarrow w_{ij}^t + df$
          4. **se** $(w_{ij}^t > u_{ik}^t)$ **então** $u_{ik}^t \leftarrow u_{ik}^t + df$
             senão $w_{ij}^t \leftarrow w_{ij}^t + df$
          5. **se** $(w_{ij}^t < u_{ik}^t)$ **então** $u_{ik}^t \leftarrow u_{ik}^t + df$
             senão $w_{ij}^t \leftarrow w_{ij}^t + df$
        fim
      fim
    fim
  fim
fim

3.4 Resultados computacionais das técnicas exatas

Nesta seção serão apresentados dois conjuntos de instâncias do TLUFLP, bem como as análises e os testes computacionais realizados por meio das técnicas exatas, baseadas no método de decomposição de Benders, que foram desenvolvidas ao longo deste trabalho.

3.4.1 Instâncias do TLUFLP

O primeiro conjunto de instâncias foram gerados em formato semelhante ao de Ro e Tcha (1984). A montagem de cada instância considerou o custo de transporte entre facilidades de primeiro e segundo nível igual a 0.0125 por unidade de distância entre os níveis, ao passo que o custo de transporte entre as facilidades de segundo nível e os clientes de 0.0250 por unidade de distância entre eles.

As distâncias entre todos os locais (facilidades e clientes) foram geradas de maneira aleatória, com um intervalo de variação de 100 à 5000 unidades. Os custos fixos referentes às facilidades de segundo nível foram gerados aleatoriamente em um intervalo de 15000 à 20000 unidades, enquanto que os custos fixos das facilidades de primeiro nível variavam de 50000 à 60000 unidades.

A demanda de cada um dos clientes também foi gerada de forma randômica, com variação de 50 a 2000 unidades. No quesito dimensão, as dez primeiras instâncias foram geradas iguais às adotadas no trabalho de Ro e Tcha (1984), e para cada uma destas dez, outras quatro foram criadas com a mesma relação entre os níveis. Além disso, outras nove dimensões de instâncias maiores foram criadas, sendo estas também expandidas para outras quatro de mesma magnitude.

O segundo conjunto de instâncias foi construído em um padrão semelhante ao utilizado no trabalho de Gendron et al. (2013), que por sua vez baseou-se no trabalho de Landete e Marín (2009), no qual um conjunto de instâncias do UFLP eram transformadas em instâncias do TLUFLP. No total existem noventa instâncias de dimensão 100x100 do UFLP, distribuídas em três grupos denominados GapA, GapB e GapC, e que estão disponíveis na seguinte página da internet: http://www.math.nsc.ru/AP/benchmarks/UFLP/Engl/uflp_dg_eng.html.

Landete e Marín (2009) propuseram a utilização das instâncias de 100x100 para a criação de noventa instâncias de dimensões 50x50x50. Os primeiros cinquenta dados são iguais aos custos de transporte (pertencente ao conjunto {1,...,5} unidades ou 10000 unidades representando ∞) entre as facilidades de segundo nível e os clientes. Já os cinquenta últimos são os custos de transporte entre as facilidades de primeiro e segundo nível. Além disso, os custos de instalação de todas as facilidades nos dois níveis são fixados em 3000 unidades, e a demanda dos clientes fixada em 1 unidade.

Ainda em relação ao segundo conjunto de instâncias, quando analisadas as suas respectivas soluções ótimas utilizando o CPLEX foram indentificados doze instâncias com soluções inviáveis, nos quais seis eram referentes ao grupo GapA e outras seis do grupo GapC, sendo todas elas descartadas para uso neste trabalho.

3.4.2 Resultados dos testes exatos

Os testes computacionais exatos foram divididos em três diferentes etapas, que são: escolha de uma boa alternativa de integralização das variáveis do SD pelo Algoritmo (6), e análise preliminar de alguns métodos exatos; avaliação de desempenho dos diferentes tipos de cortes de Benders implementados; e a realização dos testes finais baseados em combinações de diferentes tipos de cortes de Benders. Ao longo das etapas foram utilizados apenas os códigos completamente finalizados, bem como as instâncias até então criadas. A ordem de implementação dos códigos segue enumerada a seguir:

1. Problema Original (OP) de Balinski (1964) e Manne (1964);
2. Método de Benders Tradicional (BT), referentes às equações de (3.6) à (3.9) do PM e de (3.4) à (3.5) do SD;
3. Técnicas de aceleração do método de Benders (Pré-aquecimento, Múltiplos cortes e funções *Callback*);
4. Algoritmos especializados (5) e (6) para a resolução do Subproblema Dual de Benders, I-jvc e I, respectivamente;
5. Cortes Pareto-Ótimo/Papadakos (PK) pela equação (3.19);
6. Resolução do Subproblema Dual de forma desagregada;
7. Cortes Fechamento de Facilidades (CF) pela equação (3.13);
8. Cortes Pareto-Ótimo - MW.

Para a primeira etapa de testes utilizou-se um notebook Asus, com um processador Intel Core I7-3610QM 2.3GHz, e 8GB de RAM, em um sistema operacional Linux de 64 bits. Os métodos OP, BT e o chamado Benders especializado (BE) foram implementados em Concert[6], utilizando o resolvedor CPLEX 12.4. Adotou-se como critérios de parada um tempo limite de 10800 segundos associado ao valor do *gap* de dualidade calculado por meio dos limites inferior e superior, ou o tempo de convergência para a solução ótima (*gap* igual a zero) dos métodos.

[6]Linguagem de programação C++ utilizando a biblioteca do resolvedor CPLEX

O BE consiste em um método que reúne as técnicas descritas pelos itens 3, 4 e 5 (da sequência de implementações), sob a seguinte estrutura: quatro iterações de pré-aquecimento com inserção de múltiplos cortes PK e múltiplos cortes tradicionais de Benders (equação (3.7)); uso de uma única árvore de *Branch-and-Bound* por meio da função *LazyConstraintCallback*, para exploração das soluções inteiras encontradas pelo CPLEX; e a inserção de múltiplos cortes PK e múltiplos cortes provenientes do Algoritmo (6), a cada iteração de Benders após a fase de pré-aquecimento.

Ainda em relação à primeira etapa de testes, apenas dezoito instâncias no formato de Ro e Tcha (1984) haviam sido criadas, o que possibilitou avaliar o comportamento do BE frente às mudanças na forma de realizar o ajuste das variáveis w_{ij}^t e u_{ik}^t do Algoritmo (6), conforme mostrado na Tabela (3.1).

Tabela 3.1: Tempos de resolução do TLUFLP para as variações do cálculo de w_{ij}^t e u_{ik}^t pelo BE.

Instâncias	**Insp. 1**	**Insp. 2**	**Insp. 3**	**Insp. 4**	**Insp. 5**
	t(s)	**t(s)**	**t(s)**	**t(s)**	**t(s)**
5-5-50	0,01	0,01	0,01	0,01	0,00
5-15-50	0,02	0,02	0,02	0,02	0,02
5-25-50	0,04	0,03	0,04	0,03	0,03
7-10-50	0,02	0,02	0,02	0,02	0,02
7-15-50	0,05	0,05	0,05	0,05	0,05
7-20-50	0,04	0,04	0,03	0,03	0,04
7-25-50	0,16	0,16	0,16	0,16	0,15
10-10-50	0,02	0,03	0,02	0,02	0,03
10-15-50	0,04	0,04	0,04	0,04	0,04
10-20-50	0,11	0,11	0,11	0,11	0,11
10-50-100	1,64	1,60	1,64	1,55	1,68
15-25-100	0,65	0,66	0,58	0,67	0,59
15-30-150	1,11	1,06	1,01	1,11	1,01
30-50-200	15,71	10,09	12,39	13,93	12,17
30-100-200	1190,66	547,91	664,2	797,42	779,37
30-100-300	936,46	446	347,57	224,73	622,22
40-100-300	8061,81	7347,27	*	4242,35	*
50-100-500	8772,49	*	3268,36	7062,12	5479,82

* Não resolvidos em 10800 segundos de processamento.

Pela Tabela (3.1) é possível notar que houve pouca variação entre métodos de integralização das variáveis w_{ij}^t e u_{ik}^t do SD para as treze primeiras instâncias testadas. Além disso, apenas os métodos de inspeção 1 e 4 resolveram todas as instâncias dentro do tempo limite, o que confere certa vantagem para os dois quando comparados com os demais. Quando analisados os tempos computacionais dos métodos 1 e 4, percebe-se que o método 4 é melhor do que o método 1. Neste sentido, optou-se pela utilização

do método 4 para a sequência do trabalho.

Uma vez escolhido o método de inspeção 4 para integralização das variáveis w_{ij}^t e u_{ik}^t do Algoritmo (6), decidiu-se comparar os resultados provenientes dos algoritmos OP, BT e BE por meio da Tabela (3.2).

Tabela 3.2: Tempos de resolução das instâncias para os três algoritmos.

Instâncias	**OP**		**BT**		**BE**	
	t(s)	**gap(%)**	**t(s)**	**gap(%)**	**t(s)**	**gap(%)**
5-5-50	0,00	-	0,01	-	0,01	-
5-15-50	0,01	-	0,07	-	0,02	-
5-25-50	0,03	-	0,37	-	0,03	-
7-10-50	0,01	-	0,05	-	0,02	-
7-15-50	0,06	-	0,11	-	0,05	-
7-20-50	0,02	-	0,31	-	0,03	-
7-25-50	0,12	-	7,74	-	0,16	-
10-10-50	0,02	-	0,08	-	0,02	-
10-15-50	0,02	-	0,48	-	0,04	-
10-20-50	0,02	-	2,01	-	0,11	-
10-50-100	1,07	-	10800,00	2,35	1,55	-
15-25-100	0,43	-	44,75	-	0,67	-
15-30-150	0,77	-	486,02	-	1,11	-
30-50-200	28,19	-	10800,00	10,14	13,93	-
30-100-200	600,46	-	10800,00	86,53	797,42	-
30-100-300	953,10	-	10800,00	92,26	224,73	-
40-100-300	5667,11	-	10800,00	95,59	4242,35	-
50-100-500	10800,00	0,94	10800,00	97,30	7062,12	-

- Gap % igual a zero.

Na Tabela (3.2) o *gap* de dualidade de cada um dos testes realizados foi obtido pela expressão:

$$gap(\%) = 100(\textit{limite superior - limite inferior})/\textit{limite superior}$$

Os resultados apresentados na Tabela (3.2) indicam que o método OP é a melhor alternativa de resolução do TLUFLP para os problemas de pequeno porte. Esses resultados evidenciam a qualidade dos resolvedores disponíveis no mercado, que apesar de não serem códigos específicos para um determinado problema, conseguem resolver muitos problemas com certa facilidade.

No que diz respeito ao BT percebe-se a fragilidade do método quando a dimensão dos problemas aumenta. Ao observar o comportamento do BT frente às oito últimas instâncias testadas nota-se que o mesmo não apresenta um desempenho satifatório, com tempos computacionais muito altos ou até mesmo não resolvíveis em 10800 segundos.

Já o BE mostrou-se competitivo quando comparado com o OP, e mais eficiente à medida que as dimensões do problema aumentam. O BE foi o único método que conseguiu obter *gap* zero em todos os testes, considerando o tempo limite.

Em uma segunda etapa de testes, foram utilizados todos os códigos enumerados anteriormente, e além disso, os experimentos exatos passaram a ser realizados em uma CPU com um processador XEON de 2.5 GHz, e 24GB de RAM, em um sistema operacional Linux de 64 bits. Nesta máquina a versão do resolvedor CPLEX era a 12.6.0.1, e o tempo limite dos experimentos foi aumentado para vinte e quatro horas de processamento.

A ideia principal da segunda etapa de testes era identificar quais cortes, entre todos os cortes apresentados neste trabalho, que mais potencializavam a convergência dos métodos. Assim, seria possível montar novos códigos a partir da combinação de um ou mais tipos de cortes, visando a alcançar melhores resultados.

Em primeiro lugar, foram selecionadas de forma aleatória seis instâncias no formato de Gendron et al. (2013) entre as setenta e oito existentes, e outras sete, de oitenta e oito no total, de Ro e Tcha (1984). Entre as seis primeiras foram selecionadas duas para cada um dos tipos de GapA, GapB e GapC. Em relação às outras sete, apenas as três menores puderam ser utilizadas nesta etapa, uma vez que o tempo computacional demandado para cada um dos testes era muito elevado, devido às características dos códigos em questão.

Basicamente, todos os códigos analisados nesta etapa, com exceção do método OP, foram montados em uma estrutura parecida com o método de decomposição de Benders tradicional (BT), no qual se resolvia o PM, seguido pelo SD, que por sua vez, gerava novos cortes para cada um dos clientes do TLUFLP, até que o algoritmo convergisse para a solução ótima. Porém, o mecanismo de resolução recursiva do PM e SD era feito por meio da função *LazyConstraintCallback* do CPLEX, que permitia a realização de todas as iterações de Benders dentro de uma única árvore de *Branch-and-Bound* a partir da primeira chamada de resolução do PM.

No total foram construídos quatorze códigos, com diferentes tipos de cortes associados à resolução do SD. Os três primeiros cortes foram o I (corte de Benders padrão), o CF e o MW, montados a partir da variáveis do SD obtidas através do Algoritmo (6). O mesmo vale para os três cortes seguintes, com a diferença do uso do Algoritmo (5) para o cálculo das variáveis do SD (I-jvc). Os oito últimos também correspondiam aos cortes I, CF e MW, acrescido do corte PK, montados a partir da resolução do SD de forma agregada (T) ou desagregada (D), utilizando o resolvedor CPLEX. Assim, considerando as instâncias teste, foram montadas as Tabelas (3.3) e (3.4) referentes aos tempos de execução de cada um dos métodos.

Tabela 3.3: Tempos em segundos das instâncias teste no estilo de Ro e Tcha (1984).

Códigos	Instâncias		
	10-15-50-A	**15-25-100**	**30-50-200-A**
OP	0,01	0,78	15,20
I	0,13	1,44	106,69
I-CF	0,15	1,84	149,28
I-MW	0,29	3,53	205,82
I-jvc	0,11	2,33	263,92
I-CF-jvc	0,11	2,28	359,45
I-MW-jvc	0,21	2,59	284,85
T	0,13	1,83	150,92
T-CF	0,24	3,91	119,13
T-MW	0,41	3,78	199,64
T-PK	0,09	1,55	55,11
D	0,41	3,51	147,31
D-CF	0,20	3,78	727,70
D-MW	0,48	4,99	631,27
D-PK	0,31	1,87	60,17

Tabela 3.4: Tempos em segundos das instâncias teste no estilo de Gendron et al. (2013).

Códigos	Instâncias					
	532-A	**2932-A**	**1831-B**	**2031-B**	**1833-C**	**2733-C**
OP	22,92	50,14	26,05	82,92	35,89	29,82
I	1,95	4,62	12,50	13,61	3,33	3,25
I-CF	1,69	2,92	15,23	12,28	7,83	3,02
I-MW	4,78	12,15	17,82	28,58	10,04	7,38
I-jvc	1,07	1,61	2,28	16,89	7,40	4,38
I-CF-jvc	1,61	1,84	1,55	17,11	11,00	4,18
I-MW-jvc	2,67	4,29	5,02	22,60	8,66	11,43
T	5,45	10,72	7,51	19,05	8,27	7,28
T-CF	6,81	8,84	9,79	18,31	15,82	9,58
T-MW	12,19	22,51	11,56	43,23	13,13	23,53
T-PK	5,79	7,84	6,57	7,42	8,81	6,72
D	11,90	26,09	13,70	21,77	13,08	13,76
D-CF	13,81	20,00	9,32	19,10	13,50	20,65
D-MW	20,98	11,31	10,92	23,89	14,26	22,45
D-PK	11,79	15,94	11,27	20,30	16,42	16,71

Os resultados expostos nas Tabelas (3.3) e (3.4) mostram que o método OP é muito eficiente na resolução de instâncias pequenas no formato de Ro e Tcha (1984). Por outro lado, o uso de OP não se mostrou uma boa alternativa quando aplicado às instâncias que seguem a estrutura proposta por Gendron et al. (2013), uma vez que, para todas as instâncias da Tabela (3.4) o método OP teve o pior desempenho em relação ao tempo

computacional. A queda de rendimento do método OP para o segundo conjunto de instâncias torna-se ainda mais clara quando comparamos o tempo computacional da instância 30-50-200-A, com todas as instâncias da Tabela (3.4) que possuem menos da metade do número de variáveis x_{ijk} do TLUFLP.

Em se tratando do comportamento dos métodos baseados na técnica de decomposição de Benders, não foi possível identificar diferenças significativas inerentes à resolução dos dois conjuntos de instâncias, o que pode ser vista como uma vantagem em relação ao método OP. Nota-se também que é difícil indicar quais, entre as quatorze técnicas de Benders, são as melhores alternativas de resolução do TLUFLP, mesmo que os métodos T-PK, D-PK e I aparentem ser os mais competitivos quando comparados com os demais, em relação ao tempo computacional.

Em um trabalho desenvolvido por Dolan e Moré (2002), eles propuseram o desenvolvimento de uma ferramenta gráfica que permite analisar e comparar a performance de diferentes técnicas exatas de otimização. Assim, o uso desta ferramenta auxiliaria na seleção das melhores técnicas de Benders, que por sua vez seriam utilizados na última etapa de testes.

A métrica proposta Dolan e Moré (2002) é constituída de alguns conceitos e definições que serão apresentados em sequência. Consideremos S ser o conjunto de s técnicas de Benders para resolução do TLUFLP, assim como Q é o conjunto de q instâncias teste, e t_{qs} é o tempo gasto para o método s resolver a instância s. Chamaremos de r_{qs} a razão de desempenho para todos os métodos s e instâncias q, calculados através da seguinte equação:

$$r_{qs} = \frac{t_{qs}}{\min\{t_{qh} : h \in S\}}$$

Além disso, o desempenho geral de cada um dos métodos $s \in S$ pode ser obtido através de uma função de distribuição acumulativa $\psi_s(\tau)$ para a razão de desempenho r_{qs}, a partir da seguinte definição:

$$\psi_s(\tau) = \frac{1}{n_q}|\{q \in Q : r_{qs} \leq \tau\}| \tag{3.25}$$

Na equação (3.25), n_q representa a quantidade de instâncias utilizadas na análise, que no nosso caso seria igual a nove, enquanto τ é um fator numérico (dentro do espaço dos números reais) que define um intervalo de desempenho a partir dos valores de r_{qs}. As figuras (3.2), (3.3), (3.4) e (3.5) ilustram os gráficos de desempenho dos quatorze métodos de Benders, gerados a partir da transformação dos dados de tempo

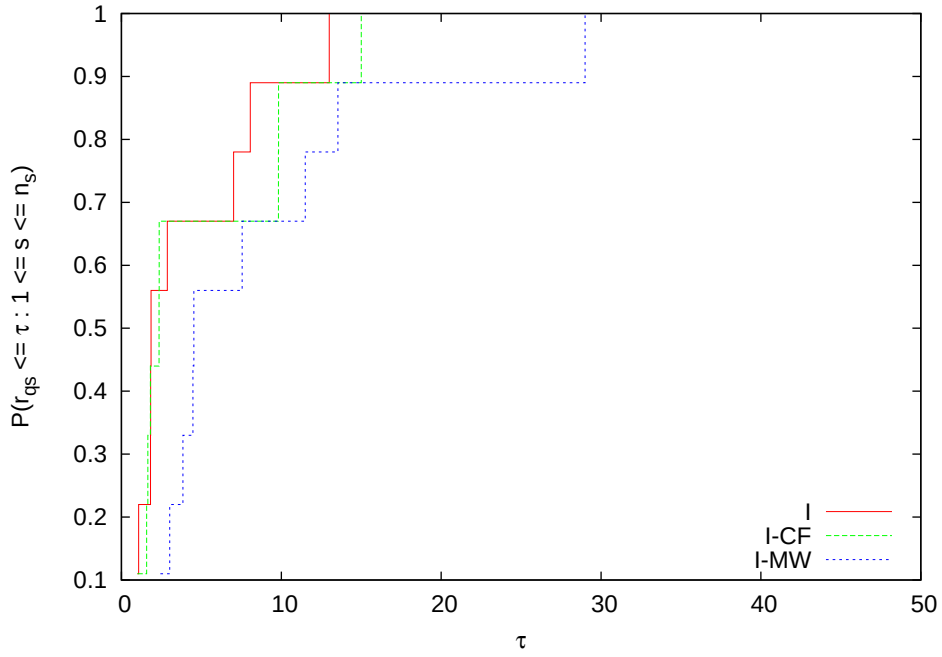

Figura 3.2: Funções $\psi_s(\tau) : s \in$ *métodos com I*

das Tabelas (3.3) e (3.4) em razões de desempenho r_{qs}. Em todos os gráficos o eixo das abscissas representa a evolução do valor de τ, já o eixo das ordenadas representa a $\mathrm{P}(r_{qs} \leq \tau : 1 \leq s \leq n_s)$, onde n_s representa a quantidade de métodos de resolução.

Primeiramente, é valido destacar os gráficos ilustrados nas quatro figuras anteriores poderiam ser condensados em apenas um gráfico, porém a identificação de cada uma das quatorze curvas seria muito difícil. Desta forma, todos os quatro gráficos foram gerados usando a mesma escala para facilitar as análises.

Em linhas gerais, a análise de performance proposta por Dolan e Moré (2002) se baseia na ideia de que uma vez estabelecido um valor limite de τ, o método de resolução mais eficiente é aquele que detêm o maior percentual de problemas com razão de desempenho r_{qs} menor do que τ. Na figura (3.2), os métodos I e I-CF mostram-se mais eficientes que o método I-MW, pois se limitarmos o valor de τ em 10, por exemplo, os dois primeiros teriam cerca de 20% de $p \in P$ com r_{qs} menores do que 10 a mais do que o I-MW. Neste sentido, mantendo o exemplo em que τ vale 10 e estendendo a análise para todos os gráficos (figuras (3.2) à (3.5)) notaremos que apenas o T-PK atingiria a totalidade de problemas com r_{qs} inferior a 10.

Portanto, se levarmos em consideração apenas a análise de desempenho gráfico, os cinco melhores métodos seriam respectivamente, T-PK, I, T, I-CF e I-jvc. No

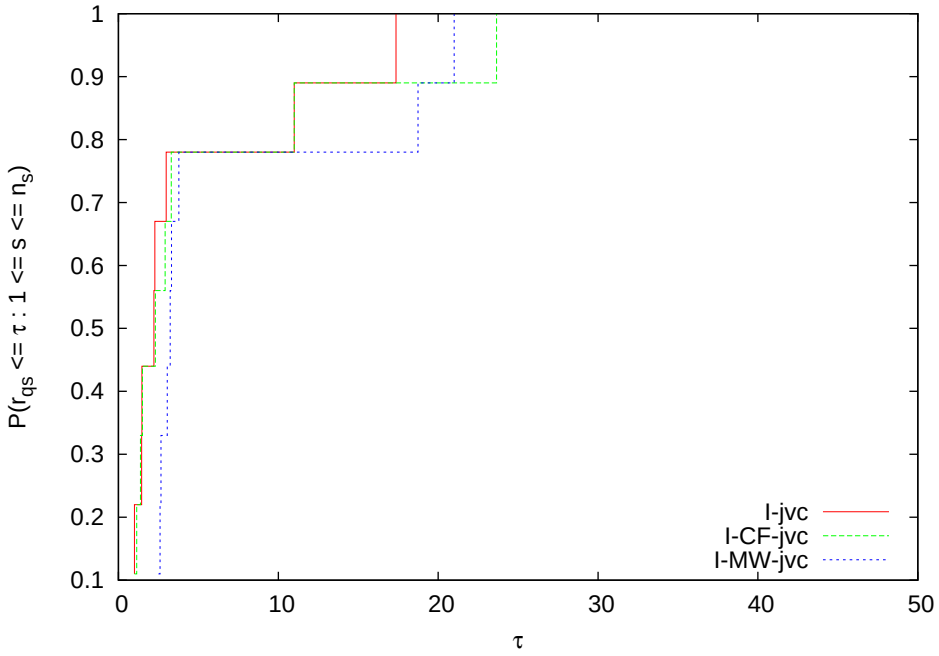

Figura 3.3: Funções $\psi_s(\tau) : s \in$ *métodos com I-jvc*

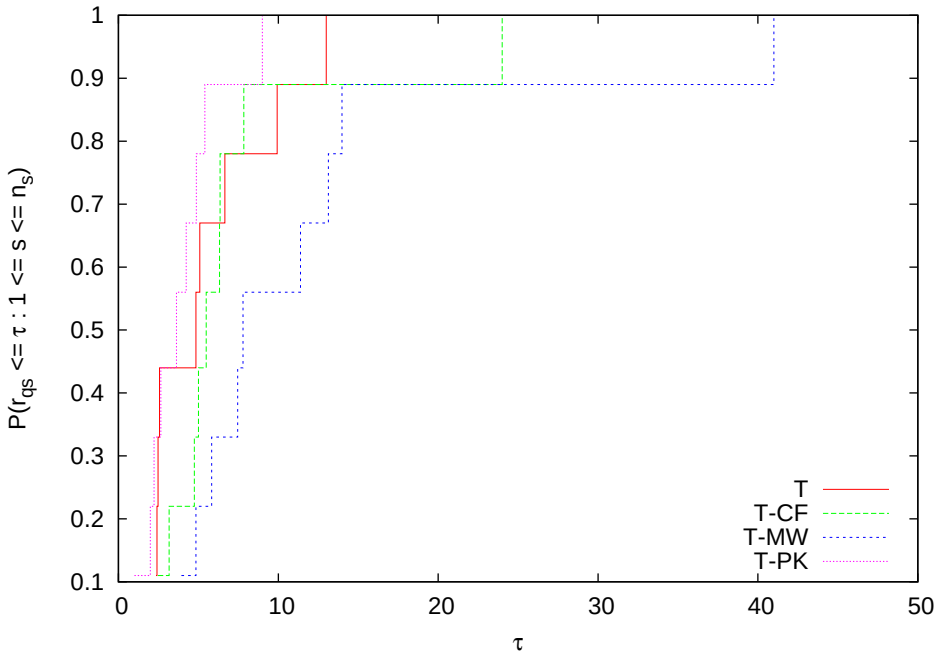

Figura 3.4: Funções $\psi_s(\tau) : s \in$ *métodos com T*

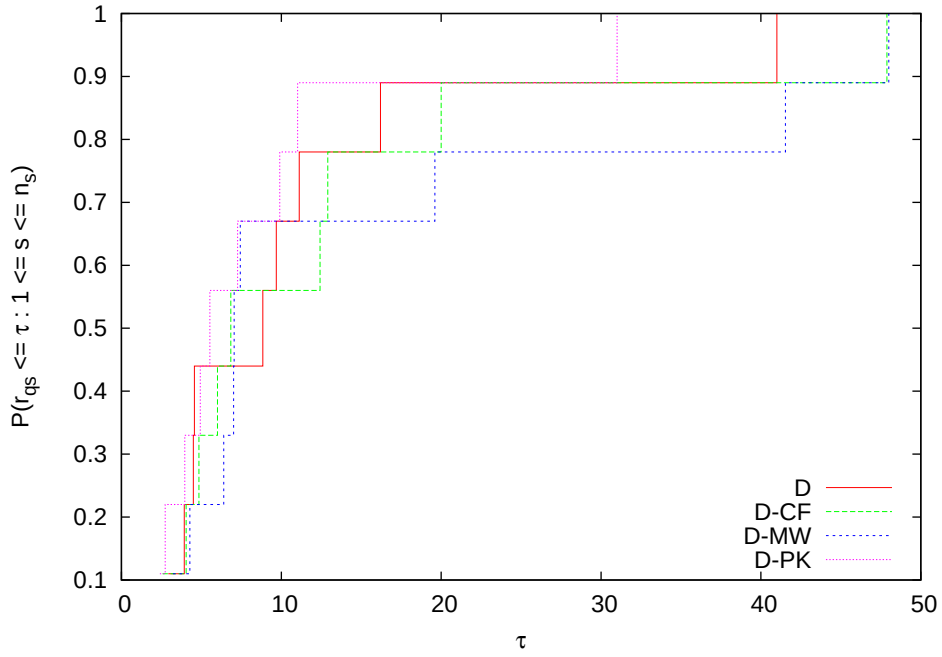

Figura 3.5: Funções $\psi_s(\tau) : s \in$ *métodos com D*

entanto, como a ideia dessa segunda etapa de testes consistia em estabelecer os métodos que possuem cortes de Benders mais promissores, considerando tempo computacional e razão de desempenho a medida que se aumentava a complexidade das instâncias, optou-se pela seleção daqueles que se destacavam em ambos os quesitos, o qual foram escolhidos os cortes T-PK, I e I-CF.

Por outro lado, como nesta segunda etapa não foi possível o uso de instâncias teste maiores, devido a tempo hábil de máquina, resolvemos incluir outros três tipos de corte, I-MW, I-jvc e I-CF-jvc, pois gostaríamos de avaliar também seus desempenhos em relação a todas as instâncias utilizadas na etapa final.

Na terceira e última etapa de testes, cinco diferentes códigos foram implementados sob os mesmos princípios, mas com diferentes tipos de inserção de cortes após a finalização das quatro iterações de pré-aquecimento. Além disso, para todos os códigos a mesma estrutura de função *CutCallback* foi adotada, no qual múltiplos cortes T-PK e T eram adicionados ao PM sempre que uma nova solução fracionária do TLUFLP fosse encontrada (limitado em cinco novas soluções fracionárias). As diferenças entre os cinco códigos estão explicitadas nos itens a seguir:

- B-P-I: Inserção de múltiplos cortes do tipo I e I-PK;

- B-P-CF: Inserção de múltiplos cortes do tipo I-CF e I-PK;
- B-MW: Inserção de múltiplos cortes do tipo I-MW;
- B-P-I-jvc: Inserção de múltiplos cortes do tipo I-jvc e I-PK-jvc;
- B-P-CF-jvc: Inserção de múltiplos cortes do tipo I-CF-jvc e I-PK-jvc.

Todas as instâncias, com exceção das usadas na etapa anterior foram resolvidas por meio dos cinco códigos citados anteriormente, acrescido do método OP. Em todos os testes foram coletados a quantidade de nós inteiros da árvore de *Branch-and-Bound* que foram explorados, os valores do limite superior e inferior, bem como o tempo computacional de resolução (limitado a vinte e quatro horas de processamento). Já o valor do *gap* de dualidade foi obtido a partir dos limites superior e inferior.

As Tabelas de (3.6) à (3.8), no fim deste capítulo, apresentam os tempos computacionais referentes às instâncias propostas por Gendron et al. (2013), enquanto a Tabela (3.9) mostra os resultados das intâncias de Ro e Tcha (1984). As soluções para as instâncias no formato de Ro e Tcha (1984), com ordem de complexidade relativamente baixa não serão apresentadas neste trabalho, uma vez que todas elas foram resolvidas com tempos inferiores a cinco segundos, e com ampla vantagem do método OP (padrão que pode ser observado através da Tabela (3.2), como exemplo).

Pelos resultados expostos na Tabelas de (3.6) à (3.9) é possível notar que o método OP é aquele que, de maneira geral, demanda o maior tempo para convergência das instâncias. É possível perceber também que nenhum dos métodos foi capaz de resolver as últimas quatro instâncias da Tabela (3.8) dentro do tempo limite.

Visando identificar o método mais eficiente para resolução do TLUFLP, foram contruídos a Tabela (3.5) e o gráfico ilustrado pela Figura (3.6). A Tabela (3.5) traz as informações de quantas vezes cada um dos métodos obtiveram o menor e o segundo menor tempo computacional de resolução dos problemas, enquanto o gráfico apresenta as curvas de desempenho de todos os métodos, obtidas por meio dos valores de razão de desempenho r_{qs}. Vale ressaltar que a Tabela (3.5) e a Figura (3.6) não contabilizaram os casos em que a solução ótima não foi alcançada, o que resultou em noventa e três instâncias avaliadas.

Através do gráfico mostrado na Figura (3.6) e da Tabela (3.5) é possível notar que o B-P-CF é o melhor código para resolução do TLUFLP, entre todos apresentados. O B-P-CF se destaca tanto em relação à evolução da razão de desempenho à medida que a complexidade das instâncias aumenta, quanto na quantidade de vezes em que o método obtém os melhores tempos de resolução das instâncias do TLUFLP. Em cerca de 88% das instâncias testadas o método B-P-CF apresentou o melhor ou segundo melhor

Tabela 3.5: Quantidade de melhores resultados para cada um dos métodos exatos

Métodos	Melhor	Seg. Melhor
OP	1	1
B-P-I	3	5
B-P-CF	60	22
B-MW	9	30
B-P-I-jvc	19	13
B-P-FC-jvc	1	22

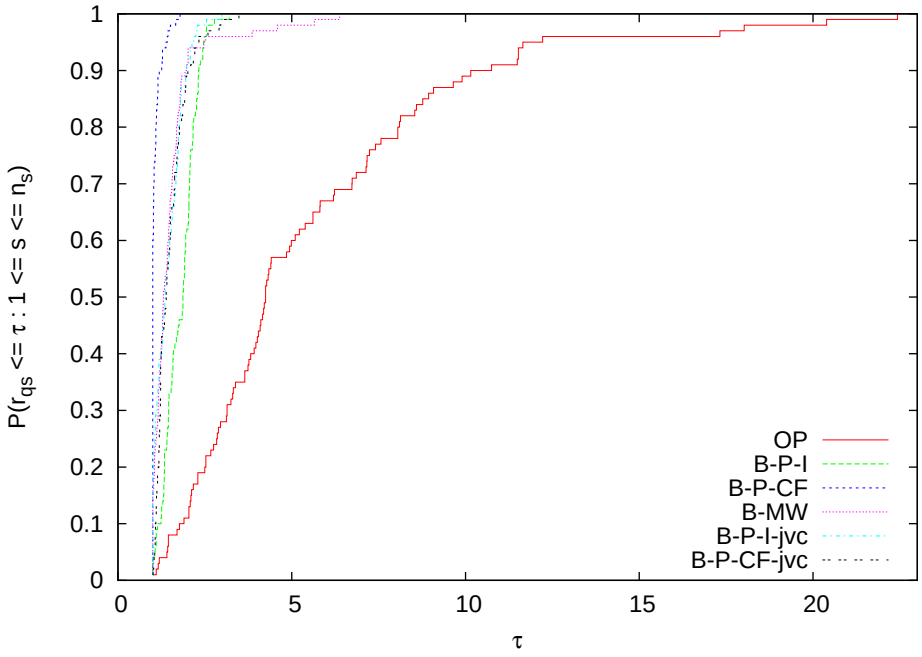

Figura 3.6: Curvas de desempenho para os métodos exatos, utilizando os dados das Tabelas de (3.6) à (3.9)

tempo de resolução, o que justifica a sua performance superior no gráfico mostrado na Figura (3.6).

É válido destacar novamente a fragilidade do método OP em relação à performance dos demais, uma vez que este consegue resolver eficientemente apenas problemas com baixo grau de complexidade.

Tabela 3.6: Resultados das instâncias tipo GapA propostas por Gendron et al. (2013)

Instâncias	**OP**	**B-P-I**	**B-P-CF**	**B-MW**	**B-P-I-jvc**	**B-P-CF-jvc**
	t(s)	**t(s)**	**t(s)**	**t(s)**	**t(s)**	**t(s)**
332-A	12,13	6,46	4,54	5,72	4,64	4,97
432-A	7,41	5,15	3,61	5,30	3,53	3,87
632-A	10,02	7,64	4,98	7,07	4,81	5,08
832-A	12,90	7,51	5,89	6,00	4,69	5,02
1032-A	22,62	10,81	4,99	5,67	3,89	4,67
1132-A	36,38	6,55	4,79	5,87	4,15	5,20
1332-A	4,93	10,11	5,13	7,78	4,48	6,20
1432-A	34,40	8,68	4,65	6,33	4,03	5,50
1532-A	63,86	12,92	5,80	6,42	5,56	8,60
1632-A	6,89	11,20	5,26	5,65	4,89	10,6
1732-A	39,38	13,24	5,74	11,36	5,88	9,69
1832-A	19,78	10,22	5,27	7,65	4,90	8,98
1932-A	15,52	11,40	5,26	9,61	5,60	7,59
2032-A	32,48	12,08	5,24	9,44	5,69	6,62
2232-A	16,47	9,34	4,37	7,30	4,73	4,98
2432-A	22,29	11,58	6,59	9,47	8,66	6,77
2532-A	15,99	10,58	5,09	7,91	10,61	6,84
2632-A	46,28	13,08	6,39	10,01	9,14	8,44
2732-A	69,56	15,79	9,49	16,62	9,39	11,57
3032-A	10,92	6,79	4,74	7,02	5,58	5,33
3132-A	9,39	9,12	4,60	7,92	7,98	5,74
3232-A	7,58	9,67	5,24	8,36	9,53	6,28
Média	23,51	10,00	5,35	7,93	6,02	6,75

Tabela 3.7: Resultados das instâncias tipo GapB propostas por Gendron et al. (2013)

Instâncias	OP	B-P-I	B-P-CF	B-MW	B-P-I-jvc	B-P-CF-jvc
	t(s)	t(s)	t(s)	t(s)	t(s)	t(s)
331-B	11,93	10,70	5,49	6,71	10,70	6,03
431-B	54,65	14,52	7,62	15,39	15,01	8,12
531-B	74,63	15,76	7,73	12,86	13,79	8,82
631-B	128,97	13,41	5,75	7,38	11,75	7,06
731-B	156,01	15,53	7,81	7,65	13,27	9,35
831-B	22,77	12,72	6,23	9,33	9,55	7,58
931-B	45,22	13,68	5,56	10,09	10,01	6,66
1031-B	99,02	24,88	11,62	24,06	9,76	22,72
1131-B	19,92	9,68	5,36	5,52	5,22	8,45
1231-B	61,02	20,01	9,80	12,76	16,35	11,56
1331-B	33,33	14,85	8,00	7,84	10,88	8,17
1431-B	26,35	12,37	5,99	6,62	6,37	7,36
1531-B	28,55	12,71	6,81	6,89	7,34	10,12
1631-B	5,89	11,75	5,65	4,96	6,32	9,63
1731-B	21,77	13,53	6,56	9,31	7,11	7,26
1931-B	28,81	11,95	5,86	5,77	6,03	9,84
2131-B	100,56	21,16	10,16	10,67	14,71	12,57
2231-B	46,48	16,81	8,63	8,93	14,24	10,29
2331-B	104,13	18,88	14,14	12,93	22,47	19,33
2431-B	11,88	5,31	3,78	4,54	6,80	7,39
2531-B	38,25	9,67	6,57	9,21	10,81	9,83
2631-B	10,39	7,07	4,89	5,68	11,19	6,78
2731-B	23,56	8,34	5,73	6,27	9,94	9,31
2831-B	17,21	7,55	6,21	6,01	13,08	8,73
2931-B	101,55	15,14	9,71	8,72	11,84	13,14
3031-B	65,43	11,47	8,05	5,68	9,24	14,70
3131-B	67,39	8,43	6,72	6,27	7,08	18,25
3231-B	63,20	9,34	6,96	11,23	8,22	12,99
Média	52,46	13,12	7,26	8,90	10,68	10,43

Tabela 3.8: Resultados das instâncias tipo GapC propostas por Gendron et al. (2013)

Instâncias	**OP**	**B-P-I**	**B-P-CF**	**B-MW**	**B-P-I-jvc**	**B-P-CF-jvc**
	t(s)	**t(s)**	**t(s)**	**t(s)**	**t(s)**	**t(s)**
333-C	24,04	10,68	5,91	5,56	7,67	5,99
433-C	48,41	7,42	5,98	6,67	10,07	7,03
533-C	11,39	5,89	4,56	6,68	7,92	5,02
633-C	23,76	4,83	3,32	3,90	4,06	4,06
733-C	31,31	12,46	5,58	7,91	11,02	9,08
833-C	282,03	20,37	16,28	20,24	24,13	28,38
933-C	16,58	8,15	5,31	6,23	9,68	10,21
1033-C	14,79	7,07	4,55	6,91	6,12	6,85
1133-C	67,10	19,14	9,97	14,27	10,12	15,07
1233-C	22,16	10,77	5,65	7,69	6,48	7,84
1333-C	36,33	13,76	7,36	7,70	8,22	10,57
1433-C	50,25	11,07	5,85	7,48	7,50	10,91
1533-C	15,03	10,53	4,55	5,40	5,92	8,87
1633-C	5,89	10,07	4,30	4,30	4,12	8,00
2133-C	18,79	10,86	5,02	6,48	5,93	6,20
2233-C	36,17	11,06	5,37	5,76	9,53	6,60
2333-C	69,9	20,82	9,23	12,35	19,63	18,54
2533-C	38,14	12,92	7,86	8,99	12,30	11,12
2833-C	19,47	5,82	3,73	4,90	6,06	4,98
2933-C	17,84	6,13	4,36	5,83	11,13	7,73
3033-C	29,53	8,77	7,30	6,69	9,86	15,12
3233-C	19,83	10,75	4,94	5,78	7,35	8,03
Média	40,85	10,88	6,23	7,62	9,31	9,83

Tabela 3.9: Resultados das instâncias no formato proposto por Ro e Tcha (1984)

Instâncias	OP		B-P-I		B-P-CF		B-MW		B-P-I-jvc		B-P-CF-jvc	
	t(s)	gap %	t(s)	gap %	t(s)	gap %	t(s)	gap %	t(s)	gap %	t(s)	gap %
30-50-200	44,94	-	23,60	-	22,08	-	22,67	-	33,87	-	32,44	-
30-50-200-B	9,5800	-	12,74	-	13,84	-	12,09	-	18,45	-	21,19	-
30-50-200-C	31,37	-	23,82	-	18,30	-	17,75	-	23,35	-	20,51	-
30-50-200-D	15,10	-	13,48	-	14,28	-	13,01	-	18,03	-	14,58	-
30-100-200	926,51	-	770,30	-	617,08	-	1225,94	-	486,96	-	1439,26	-
30-100-200-B	1291,50	-	1039,00	-	360,99	-	2073,57	-	325,02	-	586,78	-
30-100-200-C	231,12	-	72,38	-	82,95	-	54,90	-	72,41	-	65,46	-
30-100-200-D	328,23	-	189,85	-	142,70	-	175,73	-	165,46	-	151,1	-
30-100-300	1432,45	-	521,78	-	840,83	-	901,85	-	505,04	-	806,17	-
30-100-300-A	599,85	-	274,83	-	141,23	-	238,52	-	180,72	-	241,25	-
30-100-300-C	3330,49	-	4967,95	-	3501,08	-	3767,77	-	1956,63	-	3318,7	-
30-100-300-D	356,00	-	108,09	-	91,63	-	140,15	-	111,62	-	81,94	-
40-100-300	8548,30	-	3844,24	-	3375,82	-	5194,88	-	5671,31	-	11733,86	-
40-100-300-A	894,06	-	236,68	-	214,76	-	274,99	-	276,78	-	263,75	-
40-100-300-B	1394,91	-	481,95	-	550,60	-	599,88	-	1442,21	-	486,78	-
40-100-300-C	1929,08	-	215,77	-	223,00	-	361,51	-	483,69	-	450,54	-
40-100-300-D	2725,89	-	1033,52	-	534,51	-	1077,46	-	797,03	-	869,72	-
50-100-500	31027,95	-	7981,75	-	7249,73	-	33307,86	-	11457,63	-	10192,39	-
50-100-500-A	86206,63	-	7054,65	-	9742,71	-	39878,12	-	12040,57	-	8987,67	-
50-100-500-B	73023,65	-	10040,07	-	4052,58	-	15654,42	-	7280,04	-	4818,43	-
50-100-500-C	2750,53	-	663,52	-	385,02	-	688,10	-	432,36	-	651,45	-
100-200-1000-A	86400,00	8,799	86400,00	4,323	86400,00	3,797	86400,00	3,952	86400,00	3,549	86400,00	3,800
100-200-1000-B	86400,00	36,241	86400,00	3,672	86400,00	3,418	86400,00	3,604	86400,00	3,490	86400,00	3,252
100-200-1000-C	86400,00	6,953	86400,00	3,602	86400,00	3,299	86400,00	3,368	86400,00	3,420	86400,00	3,645
100-200-1000-D	86400,00	35,240	86400,00	5,005	86400,00	4,962	86400,00	4,817	86400,00	4,280	86400,00	4,465
Média	23445,36	21,808*	15406,80	4,150*	15111,03	3,869*	18051,25	3,935*	15575,17	3,685*	15633,36	3,791*

- Gap % igual a zero.

* Valores médios considerando apenas as últimas quatro instâncias.

Capítulo 4

Metaheurísticas aplicadas ao TLUFLP

Para este trabalho foram desenvolvidos dois diferentes algoritmos para resolução do TLUFLP. No primeiro deles combinou-se a metaheurística GRASP com uma técnica de Perturbações Orientadas (POr)[7] das soluções. No segundo, combinou-se a metaheurística GRASP com Path Relinking (PR). A ideia de ambos é utilizar a aleatoriedade do método GRASP combinado com VND (buscas locais) e a diversificação de soluções resultantes do POr e o PR.

Antes de descrever as técnicas, vamos apresentar algumas definições e considerações utilizadas ao longo da construção dos pseudocódigos dos algoritmos deste capítulo. Adotou-se uma nomenclatura bem parecida com a do capítulo anterior acrescida da informação da solução s envolvida, no qual facilidades de primeiro e segundo nível fechadas serão representadas por $C_{z(s)}$ e $C_{y(s)}$, respectivamente, assim como $O_{z(s)}$ e $O_{y(s)}$ representarão as facilidades abertas. A diferenciação entre os níveis hierárquicos continuam as mesmos, onde $k \in K$ são facilidades de primeiro nível, $j \in J$ são as de segundo nível e $i \in I$ são os clientes. Utilizou-se também $fo(s)$ como sendo o valor da função objetivo do TLUFLP para uma determinada solução s, assim como $fo(s^*)$ é o valor da função objetivo da melhor solução s (representada por s^*), durante a execução das metaheurísticas.

As formas de armazenamento e representação reais dos dados e soluções para o TLUFLP nas metaheurísticas desenvolvidas para este trabalho tiveram as seguintes características: os custos entre as facilidades $k \in K$ e $j \in J$, e entre as facilidades $j \in J$ e os clientes $i \in I$ foram representadas por matrizes de números reais alocadas dinamicamente; os parâmetros de custos fixos de instalação de $k \in K$ e de $j \in J$, bem como as demandas de cada um dos clientes $i \in I$ foram representados por vetores de números reais alocados dinamicamente; as informações sobre quais facilidades estavam abertas ou fechadas em cada um dos níveis foram representadas por vetores booleanos

[7] Nome dado à técnica de perturbação das soluções que foi desenvolvida para o TLUFLP

(*true* igual a facilidade aberta e *false* igual a facilidade fechada) alocados dinamicamente; e as atribuições referentes a qual conjunto de facilidades abertas (primeiro e segundo nível) atendem cada um dos clientes $i \in I$ foi feita por meio de uma matriz de inteiros alocada dinamicamente, com dimensão 2 por $|I|$ (quantidade de clientes). Cada coluna da matriz representa um cliente $i \in I$. Na primeira linha indica-se o número da facilidade $j \in J$ que atende cada cliente $i \in I$ e a segunda linha o número da facilidade $k \in K$ que atende ao mesmo cliente $i \in I$.

A Figura (4.1) ilustra a representação real (à esquerda) e esquemática (à direita) de uma solução do TLUFLP, onde v_j e v_k são vetores binários que indicam quais facilidades estão instaladas, e s é a matriz solução que mostra a atribuição de cada um dos clientes $i \in I$, conforme indicado na representação esquemática.

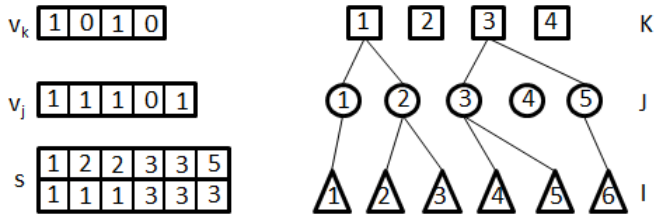

Figura 4.1: Representação da solução do TLUFLP

4.1 GRASP aplicado ao TLUFLP

A estrutura de um algoritmo baseado na técnica GRASP possui dois parâmetros α e *iter*, que precisam ser calibrados para cada problema em específico. O parâmetro α determina o percentual de variação que as soluções construídas podem se distanciar de uma solução puramente gulosa[8]. Já o parâmetro *iter* indica o número de iterações do algoritmo GRASP. Ambos os parâmetros precisam ser calibrados em função da qualidade da solução do TLUFLP que se deseja alcançar. O Algoritmo (7) ilustra a ideia básica do GRASP aplicado ao TLUFLP.

Em relação ao Algoritmo (7) vamos primeiramente falar da composição da solução inicial para o TLUFLP. Entre todas as combinações de pares de facilidades $j \in J$ e $k \in K$ deve-se selecionar uma única que irá atender a todos os clientes $i \in I$, inicialmente. O par deve ser selecionado de forma aleatória e pertencer ao conjunto P, que corresponde aos $(\alpha * 100)\%$ melhores pares. Dessa forma, a cada iteração do algoritmo $GRASP$ espera-se que haja certa variabilidade em relação à solução inicial s.

[8]Para este trabalho entende-se como uma solução gulosa a atribuição de menor custo proveniente de um único par de facilidades de primeiro e segundo nível ativas a todos os clientes

Algoritmo 7: GRASP aplicado no TLUFLP

Procedimento $GRASP(\alpha, iter)$
início
  para todo $(j \in J \wedge k \in K)$ **faça**
    $fo^* \leftarrow +\infty$
    $max \leftarrow$ *Máxima fo(s) em que um par jk atende todos os clientes* $i \in I$
    $min \leftarrow$ *Mínima fo(s) em que um par jk atende todos os clientes* $i \in I$
  fim
  $fat \leftarrow min + \alpha(max - min)$
  $P \leftarrow$ *todos os pares* $jk \mid fo(s) \leq fat$
  repita
    $s \leftarrow$ *aleatório* $jk \in P$
    $O_{z(s)} \leftarrow O_{z(s)} \cup \{k\}$
    $O_{y(s)} \leftarrow O_{y(s)} \cup \{j\}$
    $INSERE(\alpha, s)$
    se $fo(s) < fo(s^*)$ **então** $(s^* \leftarrow s) \wedge (fo^* \leftarrow fo(s))$
    $VND(s, s^*)$
    se $fo(s) < fo(s^*)$ **então** $(s^* \leftarrow s) \wedge (fo^* \leftarrow fo(s))$
    $iter \leftarrow iter - 1$
  até $(iter = 0)$;
  Retorne s^*
fim

A segunda etapa para montagem da solução inicial consiste no Algoritmo (8) de inserção de facilidades, para diminuição do valor da função objetivo do TLUFLP.

A inserção das facilidades foi feita seguindo uma ordem específica mas que também com certo grau de aleatoriedade controlado pelo parâmetro α. A ordem de inserção começa pelas facilidades $j \in C_{y(s)}$, seguido das facilidades $k \in C_{k(s)}$ e por pares de facilidades $j \in C_{y(s)}$ e $k \in C_{k(s)}$.

A lógica do Algoritmo (8) inicia-se pela inserção de facilidades $j \in C_{y(s)}$ em $O_{y(s)}$ e verificação do novo valor de $fo(s)$. Caso haja redução no valor de $fo(s)$, a facilidade j é armazenada em F^y. Em seguida, mantém-se em F^y apenas as facilidades j que pertençam ao grupo das $(\alpha 100)\%$ melhores reduções de custo no valor de $fo(s)$. Quando a inserção das facilidades $j \in C_{y(s)}$ em $O_{y(s)}$ deixa de ser vantajosa em relação à $fo(s)$, deve-se testar a abertura de facilidades $k \in C_{z(s)}$ sob os mesmos princípios de aleatoriedade e redução de custos. Uma vez que houve melhoria ao acrescentar uma facilidade $k \in C_{z(s)}$ em $O_{z(s)}$, testa-se novamente a viabilidade de inserção de novas facilidades $j \in C_{y(s)}$ em $O_{y(s)}$. Por fim, quando a inserção de uma única facilidade de primeiro e segundo nível não mais trazer benefícios, analisa-se a inserção de cada par de facilidades $j \in C_{z(s)}$ e $k \in C_{y(s)}$ em $O_{y(s)}$ e $O_{z(s)}$, respectivamente, de s. Todo o processo descrito deve ser repetido até que nenhuma facilidades fechada (ou par de facilidades) reduza o valor de $fo(s)$.

Algoritmo 8: Inserção de facilidades na solução inicial do GRASP

Procedimento $INSERE(\alpha, s)$
início
  $(aux \leftarrow 0) \wedge (fo \leftarrow fo(s))$
  enquanto $(aux \neq 2)$ **faça**
    para todo $(j \in C_{y(s)})$ **faça**
      $O_{y(s)} \leftarrow O_{y(s)} \cup \{j\}$
      se $(fo > fo(s))$ **então** $(F^y \leftarrow F^y \cup \{j\})$
      $O_{y(s)} \leftarrow O_{y(s)} \setminus \{j\}$
    fim
    se $(|F^y| > 0)$ **então**
      $F^y \leftarrow (\alpha * 100)\%\ melhores\ fo(s)\ usando\ j \in F^y$
      $O_{y(s)} \leftarrow O_{y(s)} \cup \{j\} \mid j \in F^y$
      $(fo \leftarrow fo(s)) \wedge (F^y \leftarrow \emptyset)$
    fim
    senão
      para todo $(k \in C_{z(s)})$ **faça**
        $O_{z(s)} \leftarrow O_{z(s)} \cup \{k\}$
        se $(fo > fo(s))$ **então** $(F^z \leftarrow F^z \cup \{k\})$
        $O_{z(s)} \leftarrow O_{z(s)} \setminus \{k\}$
      fim
      se $(|F^z| > 0)$ **então**
        $F^z \leftarrow (\alpha * 100)\%\ melhores\ fo(s)\ usando\ k \in F^z$
        $O_{z(s)} \leftarrow O_{z(s)} \cup \{k\} \mid k \in F^z$
        $(fo \leftarrow fo(s)) \wedge (F^z \leftarrow \emptyset)$
      fim
      senão $(aux \leftarrow 1)$
    fim
    se $(aux = 1)$ **então**
      para todo $(j \in C_{y(s)} \wedge k \in C_{z(s)})$ **faça**
        $(O_{y(s)} \leftarrow O_{y(s)} \cup \{j\}) \wedge (O_{z(s)} \leftarrow O_{z(s)} \cup \{k\})$
        se $(fo > fo(s))$ **então** $(F^{yz} \leftarrow F^{yz} \cup \{jk\})$
        $(O_{y(s)} \leftarrow O_{y(s)} \setminus \{j\}) \wedge (O_{z(s)} \leftarrow O_{z(s)} \setminus \{k\})$
      fim
      se $(|F^{yz}| > 0)$ **então**
        $F^{yz} \leftarrow (\alpha * 100)\%\ melhores\ fo(s)\ usando\ jk \in F^{yz}$
        $(O_{y(s)} \leftarrow O_{y(s)} \cup \{j\}) \wedge (O_{z(s)} \leftarrow O_{z(s)} \cup \{k\}) \mid jk \in F^{yz}$
        $(fo \leftarrow fo(s)) \wedge (F^z \leftarrow \emptyset) \wedge (aux \leftarrow 0)$
      fim
      senão $(aux \leftarrow 2)$
    fim
  fim
fim

Tendo construído uma solução inicial s, o próximo passo do Algoritmo (7) é a realização de buscas locais em diferentes espaços de vizinhanças de s por meio de uma

função VND. Por fim, após o término de todas as iterações do algoritmo, retorna-se a melhor solução do TLUFLP para o método GRASP.

4.1.1 VND e suas vizinhanças para o TLUFLP

A função VND realiza buscas locais exaustivas por uma sequência específica de vizinhanças, que a cada situação de melhora ($fo(s) < fo(s^*)$) retorna à primeira vizinhança de busca, até que não seja mais possível reduzir o valor da função objetivo. Neste sentido, percebe-se que a escolha de quais vizinhanças realizar as buscas e a ordem de exploração das mesmas tem interferência direta na construção de uma solução s.

Nove tipos de movimentos foram criados visando a exploração de diferentes espaços de vizinhança. Para apresentá-los, consideremos uma solução inicial hipotética do TLUFLP ilustrada pela Figura (4.2), no qual as facilidades de primeiro ($k \in K$) e segundo ($j \in J$) nível são representadas pelos retângulos e círculos, respectivamente, e os clientes pelos losângulos. Assim, baseado na estrutura da solução inicial é possível representar todos os nove movimentos implementados (Figura (4.3)).

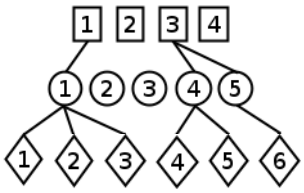

Figura 4.2: Solução Inicial

Em todos os movimentos mostrados na Figura (4.3) as facilidades $k \in K$ e $j \in J$ que sofreram alterações em relação a solução incial estão destacadas. Além disso, após a realização dos movimentos, considera-se todos os clientes i sejam atendidos pelo par de facilidades $k \in O_{z(s)}$ e $j \in O_{y(s)}$, de primeiro e segundo níveis, respectivamente, de menor custo. A descrição resumida de cada um dos movimentos é dada pelos ítens abaixo, numerados conforme indicado na Figura (4.3).

1. Fechar uma facilidade $j \in O_{y(s)}$: A facilidade $j = 5$ deixa de ser do tipo $O_{y(s)}$ e passa a ser do tipo $C_{y(s)}$;

2. Fechar uma facilidade $k \in O_{z(s)}$: A facilidade $k = 1$ deixa de ser do tipo $O_{z(s)}$ e passa a ser do tipo $C_{z(s)}$;

3. Trocar uma facilidade $j \in O_{y(s)}$ por uma $j \in C_{y(s)}$: A facilidade $j = 3$ torna-se $O_{y(s)}$, ao mesmo tempo que a facilidade $j = 5$ torna-se $C_{y(s)}$;

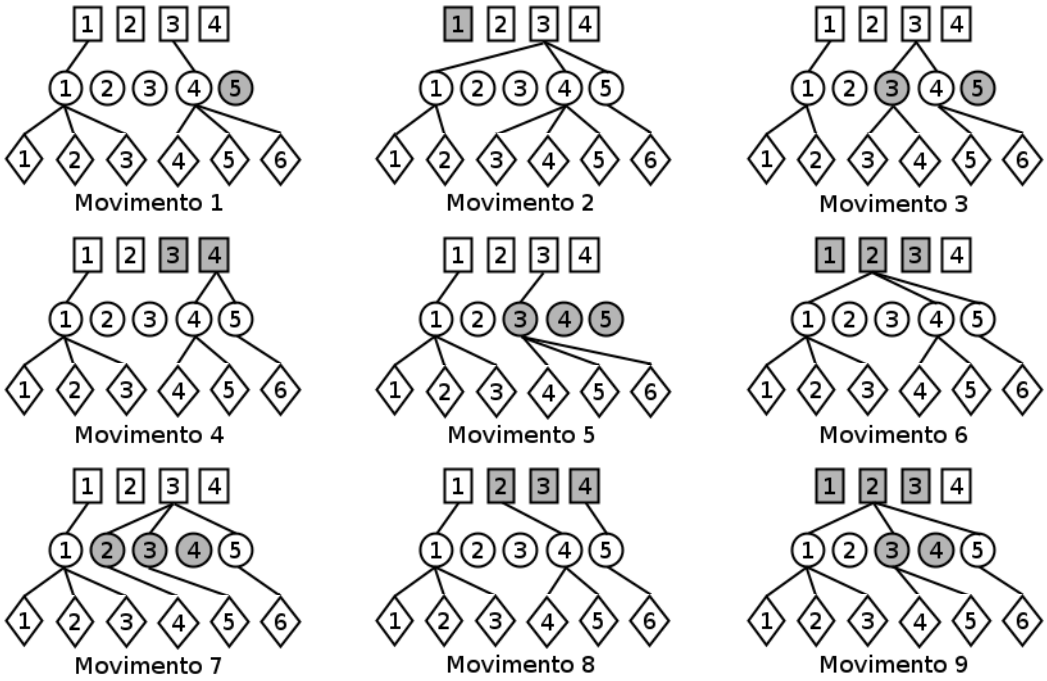

Figura 4.3: Movimentos

4. Trocar uma facilidade $k \in O_{z(s)}$ por uma $k \in C_{z(s)}$: A facilidade $k = 4$ torna-se $O_{z(s)}$, ao mesmo tempo que a facilidade $k = 3$ torna-se $C_{z(s)}$;

5. Trocar duas facilidades $j \in O_{y(s)}$ por uma $j \in C_{y(s)}$: As facilidades $j = 4$ e $j = 5$ passam a ser do tipo $C_{y(s)}$, enquanto a facilidade $j = 3$ torna-se $O_{y(s)}$;

6. Trocar duas facilidades $k \in O_{z(s)}$ por uma $k \in C_{z(s)}$: As facilidades $k = 1$ e $k = 3$ passam a ser do tipo $C_{z(s)}$, enquanto a facilidade $k = 2$ torna-se $O_{z(s)}$;

7. Trocar uma facilidade $j \in O_{y(s)}$ por duas $j \in C_{y(s)}$: A facilidade $j = 4$ torna-se $C_{y(s)}$, ao mesmo tempo que as facilidades $j = 2$ e $j = 3$ tornam-se $O_{y(s)}$;

8. Trocar uma facilidade $k \in O_{z(s)}$ por duas $k \in C_{z(s)}$: A facilidade $k = 3$ torna-se $C_{z(s)}$, ao mesmo tempo que as facilidades $k = 2$ e $k = 4$ tornam-se $O_{z(s)}$;

9. Trocar duas facilidades $k \in O_{z(s)}$ por uma $k \in C_{z(s)}$ e trocar uma facilidade $j \in O_{y(s)}$ por uma $j \in C_{y(s)}$: No primeiro nível, as facilidades $k = 1$ e $k = 3$ tornam-se $C_{z(s)}$, enquanto a facilidade $k = 2$ torna-se $O_{z(s)}$, já no segundo nível, a facilidade $j = 3$ passa a ser do tipo $O_{y(s)}$, enquanto a facilidade $j = 4$ torna-se $C_{y(s)}$.

A utilização de qualquer um dos movimentos objetiva explorar uma nova vizinhança de forma a encontrar diferentes soluções que por ventura reduzam o valor da $fo(s^*)$ atual.

Todos os nove movimentos foram testados e validados em relação às melhorias no valor da função objetivo de soluções s do TLUFLP. Por outro lado, a combinação destes movimentos para uso na função VND (seja em relação a quais, quantos e em que ordem) não foi testada segundo um padrão de metodologia científica (baseado em conceitos estatísticos) disponível na literatura.

Os critérios empíricos para a seleção de quais movimentos adotar para este trabalho foram: primeiramente limitar em um máximo de três movimentos para uso no VND e escolher um conjunto de instâncias aleatórias para realização de testes; eliminar os movimentos que demandavam maior tempo computacional (no caso, os movimentos 7, 8 e 9 foram descartados); identificar os três movimentos que apresentavam comportamentos superiores em relação aos aspectos: qualidade da solução e quantidade de iterações de melhora (se destacaram os movimentos 3, 4 e 5); definir a ordem em que os três movimentos selecionados seriam colocados na função VND, de forma a alcançar melhores soluções, com maior grau de diversificação e menor tempo computacional. Pelos testes realizados, foi escolhida a sequência 4, 3 e 5.

Cabe acrescentar que, como os algoritmos foram sendo contruídos e testados ao longo de suas implementações, os critérios citados anteriormente se mostraram mais apropriados ao utilizar somente o método GRASP. Neste contexto, quando os métodos de POr e PR foram combinados com o GRASP verificou-se (por meio de observações) que a utilização do movimento 5 influênciava pouco na qualidade das soluções, ao mesmo tempo que influenciava significativamente no aumento do tempo computacional de resolução dos TLUFLP. Assim, optou-se pela retirada do movimento 5 da função VND durante os testes computacionais apresentados na última seção deste capítulo.

4.2 Perturbações Orientadas e Path Relinking combinadas com GRASP para o TLUFLP

A ideia de se associar o GRASP com outros dois métodos (POr e PR) de diversificação de soluções surgiu a partir de análises preliminares das soluções que o GRASP estava entregando. Como as soluções do GRASP tinham certo grau de variabilidade (devido ao parâmetro α) a cada iteração do algoritmo, e que de maneira geral eram soluções relativamente boas em se tratando de tempo computacional, poderia ser interessante a exploração dessas soluções para obtenção de novas soluções para o TLUFLP.

A estrutura básica da combinação dos métodos POr e PR com o GRASP é a mesma,

porém a forma de diversificação das soluções é diferente, apesar de ambas terem como característica principal a utilização de algumas informações provenientes de soluções já existentes. O Algoritmo (9) ilustra essa combinação dos métodos.

Algoritmo 9: GRASP combinado com POr ou PR

Procedimento $GPOrPR(\alpha, it, q, itTotal)$
início
  $cont \leftarrow 0$
  enquanto $(|S| \leq q \vee cont < it)$ **faça**
    $s \leftarrow GRASP(\alpha, 1)$
    se $(\text{fo}(s) > \text{fo}(s^*))$ **então** $(s^* \leftarrow s)$
    se $(s \notin S)$ **então** $(S \leftarrow S \cup \{s\})$
    $cont \leftarrow cont + 1$
  fim
  $max \leftarrow itTotal - cont$
  $cont \leftarrow 0$
  enquanto $(cont < max)$ **faça**
    $S^{aux} \leftarrow \emptyset$
    $s \leftarrow GRASP(\alpha, 1)$
    $S^{aux} \leftarrow S^{aux} \cup \{s\}$
    $s^{'} \leftarrow Aleatório\ s_1 \in S$
    $S^{aux} \leftarrow S^{aux} \cup \{s^{'}\}$
    selecione $\{1 \vee 2\}$ **faça**
      1. $PR(s, s^{'}, S^{aux}, S) \wedge PR(s^{'}, s, S^{aux}, S)$
      2. $POr(s, s^{'}, S^{aux})$
    fim
    $S \leftarrow S \cup S^{aux}$
    $S \leftarrow q\ melhores\ soluções \in S$
    $Atualiza\ s^*\ se\ necessário$
    $cont \leftarrow cont + 1$
  fim
  $Retorne\ s^*$
fim

No Algoritmo (9) o conjunto S é chamado de conjunto elite e é composto pelas q melhores soluções do TLUFLP. S^{aux} é um conjunto auxiliar que armazena todas as soluções obtidas por meio das técnicas de diversificação POr e PR. Os parâmetros q, it e max são respectivamente, a quantidade máxima de elementos que irão compor S, o número máximo de iterações para preenchimento de S inicial, e o número máximo de iterações de diversificação que o algoritmo irá realizar. Já o parâmetro $itTotal$ indica o quantidade máxima de vezes que o método GRASP será chamado ao longo da execução do Algoritmo (9).

A execução do Algoritmo (9) se inicia a partir de um *loop*, controlado pelos parâmetros q e it, que possibilita a construção do conjunto S inicial por meio das soluções

s do método GRASP, que por sua vez tem seu parâmetro *iter* fixado em apenas uma iteração.

Uma vez que o conjunto elite inicial está montado, serão realizadas max iterações de diversificação por um dos dois métodos, POr ou PR. Vale destacar que apesar de ser possível a utilização do POr juntamente com o PR em um mesmo código, por meio da seleção aleatória ou sob algum critério específico, optou-se pela escolha de apenas uma das duas para realização de todas as max iterações, o que permitiu a comparação das técnicas na seção de resultados.

As técnicas de diversificação de soluções começam pela determinação de duas soluções base: $s \in S$ escolhida de forma aleatória e $s^{'}$ proveniente de uma nova solução do método GRASP. Logo após é selecionado um dos métodos (POr ou PR) e neles é feita a atualização do S^{aux}. Por fim, atualiza-se S com as q melhores soluções existentes até então e outro passo de diversificação é realizado.

4.2.1 GRASP com POr

A técnica POr baseia-se na criação de uma nova solução por meio da fusão de duas outras soluções, como mostrado no Algoritmo (10).

Algoritmo 10: POr aplicado ao TLUFLP

Procedimento $POr(s, s^{'}, S^{aux})$
início
  para todo $(j \in J)$ **faça**
    se $(j \notin O_{y(s)} \ \wedge \ j \in O_{y(s^{'})})$ **então** $O_{y(s)} \leftarrow O_{y(s)} \cup \{j\}$
  fim
  para todo $(k \in K)$ **faça**
    se $(k \notin O_{z(s)} \ \wedge \ k \in O_{z(s^{'})})$ **então** $O_{z(s)} \leftarrow O_{z(s)} \cup \{k\}$
  fim
  Encontrar o novo s *e sua* $fo(s)$ *utilizando* $O_{y(s)}$ *e* $O_{z(s)}$
  para todo $(j \in O_{y(s)})$ **faça**
    se (j não está alocado a algum $i \in I$) **então** $O_{y(s)} \leftarrow O_{y(s)} \setminus \{j\}$
  fim
  para todo $(k \in O_{z(s)})$ **faça**
    se (k não está alocado a algum $i \in I$) **então** $O_{z(s)} \leftarrow O_{z(s)} \setminus \{k\}$
  fim
  $VND(s, s^{''})$
  se $(fo(s^{''}) > fo(s))$ **então** $S^{aux} \leftarrow S^{aux} \cup \{s^{''}\}$
  else $S^{aux} \leftarrow S^{aux} \cup \{s\}$
fim

No Algoritmo (10) é possível perceber que uma vez selecionadas as soluções s e $s^{'}$, deve-se acrescentar todas as facilidades $j \in O_{y(s^{'})}$ e $k \in O_{z(s^{'})}$ referentes a $s^{'}$, e

acrecentá-las em $j \in O_{y(s)}$ e $k \in O_{z(s)}$ de s, respectivamente. Desta forma a nova solução s passa a conter todas as facilidades abertas de ambas as soluções iniciais. A segunda etapa do algoritmo consiste na realocação de todos os clientes $i \in I$ para o par de facilidades $j \in O_{y(s)}$ e $k \in O_{z(s)}$ de menor custo.

No processo de realocação existe a possibilidade que alguma(s) das facilidades $j \in O_{y(s)}$ e $k \in O_{z(s)}$ não esteja(m) alocada(s) a nenhum dos clientes $i \in I$, o que torna necessário a retirada da(s) mesma(s) para o cálculo da nova $fo(s)$. Por fim, na última etapa do Algoritmo (10) são realizadas de buscas locais utilizando a mesma configuração da função VND escolhida para o GRASP, e a atualização de S^{aux}.

4.2.2 GRASP com PR

A técnica de reconexão por caminhos implementada para o TLUFLP tem como característica principal a mudança gradativa das facilidades $O_{y(s)}$ e $O_{z(s)}$ de uma solução s, visando obter as mesmas facilidades $O_{y(s')}$ e $O_{z(s')}$ de outra solução s'. As mudanças realizadas no PR vão formando um caminho constituído por novas soluções, que em alguns casos, podem vir a ser melhores do que as duas iniciais. O Algoritmo (11) combinado com os Algoritmos (12) e (13), formam o método PR para o TLUFLP.

Na primeira parte do Algoritmo (11) as funções denominadas FL (*First Level*) e SL (*Second Level*) são responsáveis pelas etapas de transformação da solução s na solução s', bem como a inserção das novas soluções em S^{aux}. Em seguida, todas as soluções $s'' \in S^{aux}$ com valor de $fo(s'')$ superior ao maior valor de função objetivo das soluções do conjunto elite S são removidos de S^{aux}. Ao final, são realizadas buscas locais sobre todas as soluções $s'' \in S^{aux}$, utilizando a função VND.

As funções FL e SL são funções idênticas no quesito estrutura, porém a FL realiza mudanças apenas no primeiro nível de facilidades, enquanto a SL trata as facilidades de segundo nível. Os Algoritmos (12) e (13) ilustram as funções de diversificação do PR.

Tomando a função FL como exemplo (que se extende a SL), inicialmente é verificado se existe diferença entre a quantidade de facilidades abertas no primeiro nível entre as soluções s e s'. Caso exista diferença e esta seja positiva ($|O_{z(s)}| > |O_{z(s')}|$) então será preciso remover tantas facilidades quanto forem necessárias para que essa quantidade seja igualada. Vale ressaltar que somente facilidades $k \in O_{z(s)}$ que não existissem em $O_{z(s')}$ poderiam ser removidas. No que tange os casos em que a diferença fosse negativa ($|O_{z(s)}| < |O_{z(s')}|$) seriam incluídas em $O_{z(s)}$, facilidades $k \in O_{z(s')}$.

Terminada a fase de equalização da quantidade de facilidades no primeiro nível, serão feitas trocas sistemáticas entre facilidades $k \in O_{z(s')}$ de s' e $k \notin O_{z(s)}$ de s, até que $O_{z(s)}$ seja igual a $O_{z(s')}$. Um detalhe importante, diz respeito à escolha de

Algoritmo 11: PR aplicado no TLUFLP

Procedimento $PR(s, s^{'}, S^{aux}, S)$
início
  $FL(s, s^{'}, S^{aux}, S)$
  $SL(s, s^{'}, S^{aux}, S)$
  $maiorS \leftarrow \max\{fo(s^{''}) \mid s^{''} \in S\}$
  para todo $(s^{''} \in S^{aux})$ **faça**
    se $(fo(s^{''}) > maiorS)$ **então** $S^{aux} \leftarrow S^{aux} \setminus \{s^{''}\}$
  fim
  para todo $(s^{''} \in S^{aux})$ **faça**
    $VND(s^{'''}, s^{''})$
    se $(fo(s^{''}) > fo(s^{'''}))$ **então**
      $S^{aux} \leftarrow S^{aux} \setminus \{s^{''}\})$
      $S^{aux} \leftarrow S^{aux} \cup \{s^{'''}\})$
    fim
  fim
fim

quais facilidades seriam trocadas em cada passo, o qual seriam selecionadas aquelas que trouxessem maior benefício para $fo(s)$.

4.3 Resultados computacionais das metaheurísticas

Nesta seção serão apresentados todos os testes computacionais provenientes das metaheurísticas POr e PR combinadas com o método GRASP, ao qual chamaremos de GPOr e GPR, respectivamente, bem como as alternativas propostas para a calibração dos parâmetros dos algoritmos.

4.3.1 Calibração dos parâmetros das metaheurísticas

A primeira etapa de testes para as metaheurísticas se deu durante a calibração dos parâmetros α referente ao percentual de variação das soluções ao longo da execução do Algoritmo (9) combinado com o Algoritmo (10), e o parâmetro $itTotal$ que estabelece a quantidade de iterações do código em questão. A calibração dos parâmetros somente foi aplicada na metaheurística GRASP combinada com o POr, sendo os resultados obtidos nesta etapa, extendidos à metaheurística que contemplava o PR.

Todos os testes referentes às metaheurísticas foram realizados em uma CPU com um processador XEON de 2.4GHz, e 96GB de RAM, em um sistema operacional Linux de 64 bits, utilizando o compilador G++.

Algoritmo 12: Modificações do PR no primeiro nível de facilidades

Procedimento $FL(s, s^{'}, S^{aux}, S)$
início
  $K_s \leftarrow |O_{z(s)}|$
  $K_{s'} \leftarrow |O_{z(s')}|$
  se $(K_s > K_{s'})$ **então**
    enquanto $(K_s > K_{s'})$ **faça**
      $O_{z(s)} \leftarrow O_{z(s)} \setminus k \mid (k \in O_{z(s)} \wedge k \notin O_{z(s')})$
      se $s \notin S$ **então** $S^{aux} \leftarrow S^{aux} \cup \{s\}$
      $K_s \leftarrow K_s - 1$
    fim
  fim
  se $(K_s < K_{s'})$ **então**
    enquanto $(K_s < K_{s'})$ **faça**
      $O_{z(s)} \leftarrow O_{z(s)} \cup k \mid (k \in O_{z(s')} \wedge k \notin O_{z(s)})$
      se $s \notin S$ **então** $S^{aux} \leftarrow S^{aux} \cup \{s\}$
      $K_s \leftarrow K_s + 1$
    fim
  fim
  repita
    para todo $(k_1 \in O_{z(s')}) \wedge (k_1 \notin O_{z(s)})$ **faça**
      para todo $(k_2 \in O_{z(s)}) \wedge (k_2 \notin O_{z(s')})$ **faça**
        $G \leftarrow k_1 k_2$
      fim
    fim
    Escolher $k_1 k_2 \in G$ *de maior benefício em relação a* s
    $O_{z(s)} \leftarrow O_{z(s)} \cup \{k_1\}$
    $O_{z(s)} \leftarrow O_{z(s)} \setminus \{k_2\}$
    se $s \notin S$ **então** $S^{aux} \leftarrow S^{aux} \cup \{s\}$
  até $(O_{z(s)} = O_{z(s')})$;
fim

As instâncias utilizadas durante o processo de calibração faziam parte de um conjunto de instâncias teste selecionadas aleatoriamente durante as etapas de definição sobre quais cortes de Benders seriam utilizados na última etapa de testes dos métodos exatos, apresentada no capítulo anterior. Dentre as treze instâncias teste, todas as seis no formato de Gendron et al. (2013) e outras duas (30-100-200 e 30-100-300-B) no formato de Ro e Tcha (1984) foram utilizadas.

Definiu-se de forma empirica que o parâmetro q (quantidade máxima de soluções dentro do conjunto elite) seria igual a três, e que o parâmetro it (quantidade máxima de iterações para preenchimento do conjunto elite inicial dos algoritmos) seria fixado em quatro. Estabeleceu-se também uma faixa de variação do parâmetro α dentro do conjunto A = {0.05, 0.1, 0.15,..., 0.9}, assim como o valor de $itTotal$ seria um valor

Algoritmo 13: Modificações do PR no segundo nível de facilidades

Procedimento $SL(s, s^{'}, S^{aux}, S)$
início
  $J_s \leftarrow |O_y(s)|$
  $J_{s'} \leftarrow |O_{y(s')}|$
  se $(J_s > J_{s'})$ **então**
    enquanto $(J_s > J_{s'})$ **faça**
      $O_{y(s)} \leftarrow O_{y(s)} \setminus j \mid (j \in O_{y(s)} \wedge j \notin O_{y(s')})$
      se $s \notin S$ **então** $S^{aux} \leftarrow S^{aux} \cup \{s\}$
      $J_s \leftarrow J_s - 1$
    fim
  fim
  se $(J_s < J_{s'})$ **então**
    enquanto $(J_s < J_{s'})$ **faça**
      $O_{y(s)} \leftarrow O_{y(s)} \cup j \mid (j \in O_{y(s')} \wedge j \notin O_{y(s)})$
      se $s \notin S$ **então** $S^{aux} \leftarrow S^{aux} \cup \{s\}$
      $J_s \leftarrow J_s + 1$
    fim
  fim
  repita
    para todo $(j_1 \in O_{y(s')}) \wedge (j_1 \notin O_{y(s)})$ **faça**
      para todo $(j_2 \in O_{y(s)}) \wedge (j_2 \notin O_{y(s')})$ **faça**
        $G \leftarrow j_1 j_2$
      fim
    fim
    Escolher $j_1 j_2 \in G$ *de maior benefício em relação a* s
    $O_{y(s)} \leftarrow O_{y(s)} \cup \{j_1\}$
    $O_{y(s)} \leftarrow O_{y(s)} \setminus \{j_2\}$
    se $s \notin S$ **então** $S^{aux} \leftarrow S^{aux} \cup \{s\}$
  até $(O_{y(s)} = O_{y(s')})$;
fim

dentro do conjunto B = {8, 10, 15, 20, 25, 30, 35, 40}.

Para a realização dos testes de calibração foi gerada uma sequência aleatória de mil cento e cinquenta duas combinações, referentes à associação de uma instância teste a um par de parâmetros α e $itTotal$, de forma a evitar possíveis vícios de resultados dentro da estrutura randômica do GRASP. Cada combinação foi replicada trinta vezes, e em todas elas foram armazenadas as seguintes informações: valor da melhor solução; tempo e iteração em que a melhor solução foi encontrada; e o tempo total de execução do algoritmo.

Concluídos todos os testes, foram contabilizados ainda a quantidade de vezes em que solução ótima (obtida através da resolução do método OP apresentado no capítulo anterior) foi alcançada, e a diferença percentual (*gap*) média entre as soluções da

metaheurística e o valor da solução ótima.

Devido a quantidade elevada de informações obtidas durante a etapa de calibração, optou-se pela apresentação de apenas uma pequena amostra, como exemplo, dos resultados provenientes da instância 2733GapC nas Tabelas de (4.1) à (4.3), onde em pelo menos uma das trinta repetições a solução ótima foi encontrada.

Tabela 4.1: Tempos médios de resolução da instância 2733GapC pelo algoritmo GPOr

α	***itTotal***							
	8	**10**	**15**	**20**	**25**	**30**	**35**	**40**
0,05	9,66	12,99	23,41	58,28	50,42	56,12	60,12	69,26
0,1	13,50	14,14	23,39	33,23	43,34	61,52	81,70	102,91
0,15	10,09	23,22	73,51	72,50	49,38	64,85	69,63	85,89
0,2	10,72	43,27	26,28	57,89	64,85	57,31	69,52	82,87
0,25	19,87	32,08	31,16	59,81	47,35	74,47	74,38	85,55
0,3	16,43	16,06	29,19	44,67	52,98	58,05	68,60	136,50
0,35	18,91	18,59	40,84	101,33	52,23	61,31	102,52	87,97

Tabela 4.2: Número de vezes que o GPOr encontrou a solução ótima 2733GapC

α	***itTotal***							
	8	**10**	**15**	**20**	**25**	**30**	**35**	**40**
0,05	1	2	3	4	3	6	3	5
0,1	2	2	1	5	3	7	11	9
0,15	1	2	1	6	4	5	7	7
0,2	1	1	1	5	7	4	5	2
0,25	1	1	2	4	6	5	4	3
0,3	1	1	3	7	4	8	8	9
0,35	2	3	3	3	2	7	4	7

Tabela 4.3: Valor do *gap* médio entre as trinta soluções de GPOr para 2733GapC

α	***itTotal***							
	8	**10**	**15**	**20**	**25**	**30**	**35**	**40**
0,05	5,08	3,52	2,93	1,96	1,37	0,01	0,40	0,01
0,1	4,49	4,49	2,73	1,38	1,57	1,18	0,59	0,79
0,15	5,28	3,92	2,16	1,37	1,18	0,79	0,41	0,79
0,2	4,69	3,33	2,94	1,19	0,99	1,57	0,99	1,18
0,25	6,06	4,50	2,55	1,58	1,37	1,56	0,41	0,60
0,3	5,87	2,94	2,55	2,16	1,96	1,37	1,57	0,60
0,35	6,45	3,72	3,91	3,32	1,97	0,80	0,99	0,79

Observando as Tabelas de (4.1) à (4.3) é possível notar que a escolha da melhor configuração dos parâmetros α e $itTotal$ pode ser feita de várias formas, sendo que em cada um delas, um ou mais fatores podem ser levados em consideração. Se observarmos apenas a Tabelas (4.1), escolheríamos a combinação de α e $itTotal$ que resultasse nos menores tempos de processamento, ou seja, 0.05 e 8, respectivamente. Por outro lado, se observamos apenas os valores de *gap* médio na Tabela (4.3), escolheríamos valores mais baixos de α e mais altos de $itTotal$. Agora se analisarmos apenas a Tabela (4.2), perceberemos que o parâmetro $itTotal$ exerce maior influêcia do que o α no quesito quantidade de vezes em que a melhor solução é encontrada, e que fica difícil selecionar uma combinação de parâmetros que seja amplamente dominante em relação às outras.

Neste contexto, foram definidos dois passos estratégicos que permitissem transformar os dados das três tabelas em informações de desempenho para a metaheurística GPOr. O primeiro deles consiste em estabelecer um valor mínimo de *gap* médio para cada instância teste, de forma a reduzir os intervalos de análise dos dois parâmetros, e ao mesmo tempo satisfazer as metas mínimas de todas as instâncias, ou a maioria delas. Considerando a Tabela (4.3), se fixamos uma meta de 1% para *gap* médio, poderíamos excluir todos os valores de $itTotal$ menores ou iguais a 20, por exemplo.

O segundo se baseia principalmente na análise conjunta do tempo computacional de resolução com a quantidade de vezes em que se encontra a solução ótima (ou as melhores soluções do GPOr). A ideia básica é selecionar uma faixa equilibrada dos parâmetros α e $itTotal$, de forma a obter os menores tempos computacionais com a quantidade máxima de melhores soluções. Neste segundo passo é válido também observar se os valores médios de *gap* tendem a ser os menores dentro da faixa escolhida. Seguindo novamente o exemplo da instância 2733GapC, poderíamos estabelecer a faixa de 0.05 à 0.2 para o α, e de 25 à 35 para o $itTotal$, ou até mesmo definir valores fixos para α e $itTotal$ em 0.05 e 30, respectivamente.

Portanto, a metodologia de seleção dos parâmetros somente apresentará resultados satisfatórios se houver um comportamento semelhante (ou bem próximo) para todas as instâncias testadas, característica esta que pôde ser percebida nos testes com o algoritmo GPOr.

Por fim, foi possível perceber que os melhores resultados do GPOr foram obtidos quando α assumia os valores 0.1, 0.15 e 0.2, e $itTotal$ assumia valores próximos à quantidade de facilidades de primeiro nível dividido por dois. Desta forma, ficou estabelecido que em cada execução dos algoritmos GPOr e GPR seria feito um sorteio do valor α no intervalo [0.1 ; 0.2], bem como o valor de $itTotal$ seria igual à metade do número de facilidades de primeiro nível de cada instância.

Tabela 4.4: Resumos dos resultados das Tabelas de (4.5) à (4.11)

Índices	GPOr	GPR
t(s) total médio	210,58	178,07
gap % total médio	1,73	1,70
rep total médio	13,57	14,19
Qtd melhores t(s)	52	100
Qtd melhores gap %	64	60
Qtd melhores rep	49	46

4.3.2 Resultados computacionais das metaheurísticas

As Tabelas de (4.5) à (4.11) disponíveis no fim desta seção, apresentam as principais informações extraídas dos testes computacionais, provenientes de trinta execuções dos algoritmos GPOr e GPR, utilizando as instâncias do TLUFLP, que são:

- best gap%: Distância percentual para a solução ótima do TLUFLP;
- t(s): Tempo médio, em segundos, entre as trinta execuções das metaheurísticas;
- gap%: Distância percentual média das trinta soluções das metaheurísticas e o valor da solução ótima;
- rep: Número de vezes em que as metaheurísticas encontraram a solução ótima.

Nas Tabelas de (4.5) à (4.11) também são mostradas as informações de tempo computacional e valor da solução ótima, referentes à resolução do problema original do TLUFLP (OP) de forma exata pelo CPLEX (apresentado no capitulo anterior).

As informações referentes às quatro maiores instâncias no formato proposto por Ro e Tcha (1984) não foram coletadas, por demandarem um tempo de execução superior a vinte e quatro horas, o que inviabilizaria a realização de trinta execuções dos algoritmos.

A Tabela (4.4) sintetiza alguns índices de desempenho referentes às metaheurísticas GPOr e GPR. Nela são apresentados os valores médios dos itens t(s), gap% e rep, considerando todas as instâncias resolvidas. São mostrados também a quantidade de vezes em que cada metaheurística obteve resultados superiores para três itens citados.

Analisando a Tabela (4.4) é possível perceber que o algoritmo GPR foi um pouco melhor do que o GPOr. O ítem de maior destaque para o GPR foi a quantidade de vezes em que se obteve um tempo computacional inferior ao GPOr, onde GPR foi superior em mais de 64% da instâncias testadas. Por outro lado, essa diferença não foi tão expressiva se considerarmos o tempo total médio. Houve pouca diferença quanto aos itens gap % e rep, porém como o algoritmo GPR conseguiu obter valores médios

melhores mesmo perdendo em quantidade, pode-se considerar que o GPR foi mais eficiente.

Como todas as instâncias foram resolvidas em tempo hábil pelas técnicas exatas, e além disso o tempo computacional de resolução das metaheurísticas foi relativamente alto, os algoritmos GPOr e GPR não se mostraram boas alternativas para a resolução do TLUFLP.

Tabela 4.5: Resultados computacionais das metaheurísticas 1

Instância	OP		GPOr				GPR			
	sol ótima	t(s)	best gap%	gap%	t(s)	rep	best gap%	gap%	t(s)	rep
332GapA	42189	12,13	-	1,882	45,27	19	-	4,238	40,81	10
432GapA	54119	7,41	-	4,048	70,69	6	-	1,840	62,73	18
632GapA	54238	10,02	5,085	5,556	63,53	0	5,085	5,735	56,92	0
832GapA	51144	12,9	-	5,039	60,66	4	5,490	5,812	54,54	0
1032GapA	57120	22,62	-	0,002	64,40	20	-	0,001	59,94	23
1132GapA	57133	36,38	-	0,006	66,99	15	-	0,009	59,66	11
1332GapA	54176	4,93	0,007	4,553	65,76	0	0,007	5,277	60,26	0
1432GapA	54149	34,4	0,006	3,319	62,47	0	-	3,684	58,62	4
1532GapA	54142	63,86	-	0,574	61,47	8	-	1,307	56,84	6
1632GapA	45150	6,89	-	4,194	53,33	11	-	0,002	49,02	28
1732GapA	51191	39,38	5,428	7,509	59,99	0	-	8,874	54,83	1
1832GapA	57131	19,78	-	2,109	68,43	3	-	2,626	64,43	4
1932GapA	42121	15,52	-	0,251	43,01	23	-	0,020	43,62	22
2032GapA	48112	32,48	-	5,378	53,91	4	-	5,171	52,44	5
2232GapA	54133	16,47	-	2,952	61,97	3	0,009	3,134	59,80	0
2432GapA	51153	22,29	-	1,369	58,69	14	-	1,180	56,74	11
2532GapA	54182	15,99	5,168	5,842	65,38	0	0,004	5,846	64,29	0
2632GapA	51175	46,28	5,464	6,383	58,42	0	0,029	6,384	56,38	0
2732GapA	48136	69,56	-	1,049	49,84	4	-	1,263	49,02	4
3032GapA	48191	10,92	5,741	6,096	53,08	0	5,741	6,311	54,80	0
3132GapA	48148	9,39	-	4,735	56,00	8	-	4,532	58,15	8
3232GapA	48161	7,58	-	1,652	59,39	17	-	2,066	61,38	17
Média		23,51	1,223	3,386	59,21	7	0,744	3,423	56,15	8

- Gap % igual a zero.

Tabela 4.6: Resultados computacionais das metaheurísticas 2

Instância	OP		GPOr				GPR			
	sol ótima	t(s)	best gap%	gap%	t(s)	rep	best gap%	gap%	t(s)	rep
331GapB	45157	11,93	-	1,339	46,97	6	-	1,346	47,05	4
431GapB	45170	54,65	-	3,524	49,52	8	-	3,955	44,08	10
531GapB	45170	74,63	-	3,533	49,01	1	-	4,400	42,39	4
631GapB	45120	128,97	-	0,018	43,35	5	-	0,005	38,92	23
731GapB	48122	156,01	-	0,641	49,33	18	-	0,428	49,28	13
831GapB	42150	22,77	-	4,476	40,55	7	-	5,411	41,82	6
931GapB	48095	45,22	-	0,625	49,29	24	-	0,417	52,81	25
1031GapB	42177	99,02	6,539	7,259	43,57	0	6,539	7,020	46,73	0
1131GapB	42191	19,92	6,512	6,977	47,79	0	6,512	7,452	51,66	0
1231GapB	48123	61,02	-	2,711	50,33	5	-	3,123	51,52	4
1331GapB	42160	33,33	-	5,891	44,22	5	-	6,120	44,89	6
1431GapB	45110	26,35	-	0,450	47,41	15	-	0,451	50,41	14
1531GapB	48163	28,55	-	2,487	49,50	1	-	2,282	54,50	4
1631GapB	45198	5,89	-	7,659	52,08	3	-	7,658	57,18	2
1731GapB	45183	21,77	-	6,506	50,63	1	6,100	6,511	53,56	0
1931GapB	39170	28,81	-	5,585	39,51	3	-	3,808	42,33	11
2131GapB	48143	100,56	-	2,931	49,26	1	-	3,538	51,80	1
2231GapB	48163	46,48	-	6,351	55,26	1	5,773	6,772	58,95	0
2331GapB	48138	104,13	-	0,641	51,28	10	-	0,840	51,78	8
2431GapB	45147	11,88	-	5,522	51,75	13	-	6,406	54,65	12
2531GapB	45131	38,25	-	5,535	57,32	6	-	5,975	54,96	3
2631GapB	48154	10,39	-	3,100	71,48	5	-	2,474	56,30	11
2731GapB	42124	23,56	-	4,980	56,19	2	-	5,216	45,41	2
2831GapB	48139	17,21	-	0,018	69,03	6	-	0,221	55,05	7
2931GapB	48139	101,55	-	3,733	62,57	4	-	4,350	53,80	1
3031GapB	42123	65,43	-	1,432	45,19	23	-	0,250	47,13	25
3131GapB	42163	67,39	-	3,313	44,69	7	-	1,676	47,23	9
3231GapB	48143	63,2	-	3,526	52,07	1	-	3,322	57,26	1
Média		52,46	0,466	3,599	50,68	6	0,890	3,622	50,12	7

- Gap % igual a zero.

Tabela 4.7: Resultados computacionais das metaheurísticas 3

Instância	OP		GPOr				GPR			
	sol ótima	t(s)	best gap%	gap%	t(s)	rep	best gap%	gap%	t(s)	rep
333GapC	54163	24,04	-	2,034	65,68	7	-	2,405	71,43	8
433GapC	45140	48,41	-	6,176	55,66	3	-	6,847	54,49	2
533GapC	51138	11,39	-	0,202	57,32	5	-	0,007	63,92	3
633GapC	57134	23,76	-	0,007	70,98	24	-	0,528	89,39	25
733GapC	51157	31,31	-	5,426	80,34	1	-	5,038	72,18	3
833GapC	48127	282,03	-	0,664	50,29	2	-	0,458	57,30	1
933GapC	54118	16,58	-	1,486	63,47	2	-	1,857	84,14	2
1033GapC	54170	14,79	-	6,570	68,14	2	-	5,651	77,14	2
1133GapC	51204	67,1	5,397	5,719	60,14	0	5,397	5,724	65,21	0
1233GapC	54146	22,16	-	4,060	62,12	2	-	3,689	80,18	1
1333GapC	51138	36,33	-	3,322	55,89	1	0,059	3,517	70,83	0
1433GapC	48159	50,25	-	1,646	49,25	21	-	2,668	64,44	17
1533GapC	54161	15,03	-	4,045	64,41	4	-	3,856	79,70	3
1633GapC	57151	5,89	-	1,400	68,32	3	-	1,061	81,21	5
2133GapC	51166	18,79	-	1,794	59,10	3	-	1,600	62,78	1
2233GapC	51139	36,17	-	1,952	58,77	8	-	0,195	71,64	26
2333GapC	45125	69,9	-	2,419	46,34	17	-	1,539	56,76	23
2533GapC	51141	38,14	-	4,877	55,44	3	-	4,873	66,89	3
2833GapC	48159	19,47	-	4,346	50,50	9	-	3,725	54,45	12
2933GapC	48149	17,84	-	2,698	73,24	7	-	2,074	78,45	14
3033GapC	51164	29,53	5,471	5,994	83,79	0	-	5,415	74,36	1
3233GapC	45132	19,83	-	0,014	66,09	10	-	0,021	58,08	8
Média		40,85	0,494	3,039	62,06	6	0,248	2,852	69,77	7

- Gap % igual a zero.

Tabela 4.8: Resultados computacionais das metaheurísticas 4

Instância	OP		GPOr				GPR			
	sol ótima	t(s)	best gap%	gap%	t(s)	rep	best gap%	gap%	t(s)	rep
5-5-50	1553409,38	0,00	-	-	0,00	30	-	-	0,00	30
5-5-50-A	1862056,41	0,01	-	-	0,00	30	-	-	0,00	30
5-5-50-B	2238109,50	0,01	-	-	0,00	30	-	-	0,00	30
5-5-50-C	1531929,20	0,01	-	-	0,00	30	-	-	0,00	30
5-5-50-D	2176304,00	0,00	-	-	0,00	30	-	-	0,00	30
5-15-50	1136714,05	0,01	-	-	0,04	30	-	-	0,04	30
5-15-50-A	1399323,17	0,02	-	0,171	0,05	29	-	-	0,03	30
5-15-50-B	1141878,38	0,07	-	-	0,03	30	-	-	0,02	30
5-15-50-C	1373068,71	0,02	-	0,077	0,07	27	-	-	0,04	30
5-15-50-D	1421784,28	0,02	-	-	0,08	30	-	-	0,08	30
5-25-50	1092181,81	0,01	-	0,424	0,12	24	-	0,029	0,07	29
5-25-50-A	1164840,03	0,02	-	-	0,13	30	-	-	0,12	30
5-25-50-B	1203784,97	0,03	-	1,898	0,10	1	1,926	1,963	0,14	0
5-25-50-C	1336783,25	0,03	-	-	0,24	30	-	-	0,22	30
5-25-50-D	1179221,84	0,02	-	-	0,09	30	-	-	0,09	30
7-10-50	1139630,92	0,01	-	-	0,02	30	-	-	0,02	30
7-10-50-A	1367304,07	0,01	-	-	0,02	30	-	-	0,01	30
7-10-50-B	1548530,99	0,01	0,634	0,638	0,01	0	0,634	0,638	0,01	0
7-10-50-C	1593559,70	0,02	-	0,158	0,02	24	-	-	0,02	30
7-10-50-D	1547171,83	0,02	-	-	0,03	30	-	-	0,03	30
7-15-50	1351425,47	0,06	-	0,090	0,03	23	-	0,091	0,03	23
7-15-50-A	1221551,45	0,02	-	0,602	0,03	18	-	0,099	0,02	28
7-15-50-B	1127630,61	0,02	-	0,416	0,04	25	-	0,582	0,04	23
7-15-50-C	1404260,03	0,10	-	-	0,05	30	-	-	0,04	30
7-15-50-D	1506196,62	0,02	-	-	0,07	30	-	-	0,05	30
Média		0,02	0,025	0,179	0,05	26	0,102	0,136	0,05	27

- Gap % igual a zero.

Tabela 4.9: Resultados computacionais das metaheurísticas 5

Instância	OP		GPOr				GPR			
	sol ótima	t(s)	best gap%	gap%	t(s)	rep	best gap%	gap%	t(s)	rep
7-20-50	1159520,79	0,02	-	0,014	0,06	29	-	-	0,05	30
7-20-50-A	1139668,12	0,02	-	0,525	0,10	13	-	0,492	0,08	14
7-20-50-B	1133657,99	0,02	-	0,385	0,13	25	-	0,314	0,09	23
7-20-50-C	1282003,23	0,16	-	0,050	0,13	29	-	0,089	0,08	29
7-20-50-D	1268976,50	0,03	-	-	0,13	30	-	-	0,13	30
7-25-50	1132242,32	0,15	-	0,120	0,15	15	-	0,094	0,12	12
7-25-50-A	1427243,89	0,17	-	0,779	0,23	20	-	0,378	0,15	25
7-25-50-B	1137535,43	0,04	-	0,206	0,18	26	-	-	0,14	30
7-25-50-C	1391980,62	0,03	-	-	0,24	30	-	-	0,21	30
7-25-50-D	1165122,42	0,03	-	0,509	0,16	24	-	0,560	0,13	19
10-10-50	1503203,15	0,02	-	0,252	0,03	23	-	-	0,03	30
10-10-50-A	1261509,42	0,02	-	-	0,03	30	-	-	0,03	30
10-10-50-B	1147027,21	0,01	-	-	0,05	30	-	-	0,05	30
10-10-50-C	1442341,73	0,02	-	-	0,04	30	-	-	0,04	30
10-10-50-D	1523556,90	0,02	-	-	0,04	30	-	-	0,05	30
10-15-50	1098104,80	0,01	-	-	0,11	30	-	-	0,09	30
10-15-50-B	1287352,98	0,02	-	-	0,11	30	-	-	0,08	30
10-15-50-C	1193463,41	0,12	-	-	0,10	30	-	-	0,08	30
10-15-50-D	1312137,12	0,02	-	0,020	0,09	23	-	-	0,06	30
10-20-50	954389,62	0,03	-	1,408	0,14	17	-	1,188	0,11	19
10-20-50-A	1280957,13	0,01	-	-	0,20	30	-	-	0,17	30
10-20-50-B	1001655,48	0,02	-	-	0,16	30	-	-	0,18	30
10-20-50-C	1278004,79	0,03	-	-	0,24	30	-	-	0,19	30
10-20-50-D	1097358,36	0,08	-	0,195	0,18	16	-	0,226	0,16	15
Média		0,05	0,000	0,186	0,13	26	0,000	0,139	0,10	27

- Gap % igual a zero.

Tabela 4.10: Resultados computacionais das metaheurísticas 6

Instância	OP		GPOr				GPR			
	sol ótima	t(s)	best gap%	gap%	t(s)	rep	best gap%	gap%	t(s)	rep
10-50-100	1763531,48	2,24	-	0,173	4,59	14	-	0,059	3,68	21
10-50-100-A	1476262,25	1,35	-	0,228	3,56	25	-	0,473	2,60	19
10-50-100-B	1702023,64	1,61	0,136	0,676	4,19	0	0,136	0,797	2,79	0
10-50-100-C	2003906,61	0,98	-	0,153	4,38	18	-	0,133	3,43	18
10-50-100-D	1596335,92	0,16	-	0,088	2,99	25	-	-	2,64	30
15-25-100-A	1945842,05	0,76	-	0,196	1,30	1	0,089	0,238	1,06	0
15-25-100-B	2025666,16	0,61	-	0,373	1,35	11	-	0,224	0,93	17
15-25-100-C	1983877,72	0,69	-	0,335	1,39	13	-	0,254	1,09	16
15-25-100-D	1922833,39	0,08	-	0,054	1,32	29	-	-	1,05	30
15-30-150	2323108,14	1,02	-	0,005	3,47	29	-	-	2,68	30
15-30-150-A	2512880,18	1,11	-	-	6,01	30	-	-	4,14	30
15-30-150-B	2452632,55	0,96	-	-	4,80	30	-	-	3,65	30
15-30-150-C	2469652,28	0,16	-	0,007	6,06	29	-	0,007	4,68	29
15-30-150-D	2664514,51	0,15	-	-	4,22	30	-	-	3,34	30
30-50-200	2702434,06	44,94	-	0,144	69,90	16	-	0,191	47,67	9
30-50-200-B	2824350,58	9,58	-	0,035	77,28	29	-	0,211	52,53	24
30-50-200-C	2739287,77	31,37	0,297	0,788	80,12	0	0,396	0,758	57,19	0
30-50-200-D	2750655,84	15,10	-	0,673	106,23	3	-	0,691	68,56	2
30-100-200	2779218,00	926,51	-	0,552	427,78	1	0,315	0,727	278,67	0
30-100-200-B	2627461,14	1291,50	-	1,811	436,96	1	1,333	1,637	297,84	0
30-100-200-C	2610127,69	231,12	-	1,235	422,25	1	-	1,158	281,16	1
30-100-200-D	2579724,75	328,23	-	1,335	508,32	1	0,313	1,433	335,20	0
Média		131,37	0,020	0,403	99,02	15	0,117	0,409	66,21	15

- Gap % igual a zero.

Tabela 4.11: Resultados computacionais das metaheurísticas 7

Instância	OP		GPOr				GPR			
	sol ótima	t(s)	best gap%	gap%	t(s)	rep	best gap%	gap%	t(s)	rep
30-100-300	3420062,08	1432,45	0,060	0,217	1040,23	0	0,060	0,244	652,33	0
30-100-300-A	3456442,66	599,85	-	0,371	806,30	14	-	0,408	689,54	16
30-100-300-C	3647069,87	3330,49	0,478	1,482	906,74	0	0,478	1,398	647,37	0
30-100-300-D	3367121,02	356,00	-	0,263	670,50	15	-	0,278	501,70	14
40-100-300	3399890,22	8548,30	-	1,099	1017,64	2	0,658	1,443	738,70	0
40-100-300-A	3292450,69	894,06	-	1,222	1124,71	1	0,246	1,118	730,96	0
40-100-300-B	3381330,15	1394,91	0,085	1,084	1221,88	0	0,088	1,285	829,66	0
40-100-300-C	3427186,43	1929,08	-	0,821	1555,14	1	0,487	0,975	995,05	0
40-100-300-D	3466509,83	2725,89	0,036	0,178	1614,19	0	0,036	0,180	1088,75	0
50-100-500	4804176,09	31027,95	0,022	0,536	4707,30	0	0,154	0,630	3724,62	0
50-100-500-A	4925340,42	86206,63	-	0,755	4276,50	1	0,077	0,693	4815,50	0
50-100-500-B	4928096,69	73023,65	-	1,043	4076,08	1	0,695	1,095	4062,76	0
50-100-500-C	4861224,99	2750,53	-	0,784	3563,11	1	0,350	0,840	2668,89	0
Média		16478,45	0,052	0,758	2044,64	3	0,256	0,814	1703,53	2

- Gap % igual a zero.

Capítulo 5

Conclusão

O presente trabalho teve como objetivo o desenvolvimento de duas diferentes abordagens para a resolução do problema de localização em dois níveis de facilidades não capacitadas, ao qual denominamos TLUFLP. A primeira delas, uma técnica exata baseada no método de decomposição de Benders. A segunda, consistia em dois algoritmos heurísticos (GPOr e GPR), o qual o primeiro utilizava a metaheurística GRASP combinada com o método de perturbações orientadas (POr) e uma função VND para realização de buscas locais, enquanto o segundo, também utilizava o GRASP e o VND, com a diferença da utilização da metaheurística de reconexão por caminhos (PR), para diversificação de soluções.

Em relação ao método exato, foi verificado que o método de Benders quando aplicado na íntegra, não é uma boa alternativa para resolução do TLUFLP. Por outro lado, foi verificado também que o uso de técnicas de aceleração do método de decomposição de Benders (inserção de múltiplos cortes, uso de uma única árvore de *Branch-and-Bound*, utilização de uma fase de pré-aquecimento e a resolução do subproblema dual de Benders (SD) via algoritmos especializados), bem como a inserção de diferentes tipos de cortes de Benders, como os cortes Fechamento de Facilidades (CF), Pareto-Ótimo via inspeção e Pareto-Ótimo via subproblema de Papadakos (2008) (P), quando associados ao método tradicional, possibilita resolver o TLUFLP de forma eficiente.

O trabalho apresentou dois algoritmos especializados para o cálculo das variáveis do SD, o qual um deles utilizava um algoritmo JVC (de Jonker e Volgenant (1987)), e o outro propunha cinco diferentes alternativas para tal. Considerando os dois algoritmos, foi verificado que a melhor forma de resolução do SD via inspeção é o uso de uma das alternativas de segundo algoritmo, mais especificamente o método denominado "Insp. 4".

Considerando todas estruturas de algoritmos exatos testados, o algoritmo B-P-CF, que reunia todas as técnicas de aceleração e acrescentava, em cada iteração do método

de decomposição de Benders, cortes do tipo CF e P, foi aquele que apresentou melhor desempenho computacional, apesar de que nem mesmo ele foi capaz de resolver as quatro maiores instâncias dentro do tempo pré-estabelecido.

No que tange às metaheurísticas implementadas, pode-se dizer que ambas falharam quanto ao objetivo inicial proposto. Esperava-se construir alternativas de solução do TLUFLP que resultassem em tempos computacionais significativamente inferiores aos encontrados nos métodos exatos, o que não foi verificado. Por outro lado, acredita-se que o aprimoramento de ambas as metaheurísticas em relação à aspectos como estrutura de dados adotada, pré-processamento dos dados, e a utilização de algumas informações provenientes das técnicas exatas, permitam obter melhores resultados para o TLUFLP, uma vez que tanto o GPOr quanto o GPR, apesar de não muito precisos, apresentaram boa acurácia para a maioria dos testes.

Como sugestão para trabalhos futuros, fica a implementação e validação do corte δ-vizinhança do TLUFLP, o aprimoramento das técnicas heurísticas para o problema, bem como o desenvolvimento de uma técnica híbrida (combinação de métodos exatos e heurísticos) para resolução do TLUFLP.

Referências Bibliográficas

Aardal, K., Chudak, F. A., e Shmoys, D. B. (1999). A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Information Processing Letters*, 72:161–167.

Aardal, K., Labbé, M., Leung, J., e Queyranne, M. (1996). On the two-level uncapacitated facility location problem. *INFORMS J. COMPUT*, 8:289–301.

Aikens, C. (1985). Facility location models for distribution planning. *European Journal of Operational Research*, 22(3):263 – 279.

Akinc, U. e Khumawala, B. M. (1977). An efficient branch and bound algorithm for the capacitated warehouse location problem. *Management Science*, 23(6):585–594.

Baïou, M. e Barahona, F. (2014). A polyhedral study of a two level facility location model. *RAIRO - Operations Research*, 48(2):153–165.

Balinski, M. (1964). On finding integer solutions to linear programs. pages 225–248.

Barros, A. e Labbé, M. (1994). A general model for uncapacitated facility and depot location problem. *Location Science*, 2(3):173–191.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerisch Mathematik*, 4:238–252.

Binato, S., de Oliveira, G. C., e de Araujo, J. L. (2001). A greedy randomized adaptive search procedure for transmission expansion planning. *Power Systems, IEEE Transactions on*, 16(2):247–253.

Birge, J. R. e Louveaux, F. V. (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392.

Camargo, R. S., Miranda Jr, G., e Luna, H. P. (2008). Benders decomposition for the uncapacitated multiple allocation hub location problem. *Computers & Operations Research*, 35(4):1047–1064.

Camargo, R. S., Miranda Jr, G., e Luna, H. P. (2009). Benders decomposition for hub location problems with economies of scale. *Transportation Science*, 43:86–97.

Contreras, I., Cordeau, J.-F., e Laporte, G. (2011). Benders decomposition for large-scale uncapacitated hub location. *Operations Research*, 59(6):1477–1490.

Cordeau, J. F., Soumis, F., e Desrosiers, J. (2000). A Benders decomposition approach for the locomotive and car assignment problem. *Transportation Science*, 34:133–149.

Cordeau, J.-F., Stojkovic, G., Soumis, F., e Desrosiers, J. (2001). Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388.

Costa, A. M. (2005). A survey on benders decomposition applied to fixed-charge network design problems. *Computers & operations research*, 32(6):1429–1450.

Dolan, E. D. e Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

Efroymson, M. e Ray, T. (1966). A branch-and-bound algorithm for plant location. *Operational Research*, 14:361 – 368.

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operational Research*, 26:992–1009.

Farahani, R., Abedian, M., e Sharahi, S. (2009). Dynamic facility location problem. In Zanjirani Farahani, R. and Hekmatfar, M., editors, *Facility Location*, Contributions to Management Science, pages 347–372. Physica-Verlag HD.

Feo, T. A. e Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67 – 71.

Feo, T. A. e Resende, M. G. C. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6:109–133.

Fortz, B. e Poss, M. (2009). An improved benders decomposition applied to a multi-layer network design problem. *Oper. Res. Lett.*, 37(5):359–364.

Gabor, A. F. e van Ommeren, J.-K. C. (2010). A new approximation algorithm for the multilevel facility location problem. *Discrete Applied Mathematics*, 158(5):453 – 460.

Gendron, B., Khuong, P.-V., e Semet, F. (2013). A Lagrangian-based branch-and-bound algorithm for the two-level uncapacitated facility location problem with single-assignment constraints. Technical report, CIRRELT.

Glover, F. (1996). *Tabu search and adaptive memory programing – Advances, applications and challenges*. Kluwer.

Goldengorin, B., Ghosh, D., e Sierksma, G. (2003). Branch and peg algorithms for the simple plant location problem. *Computers and Operations Research*, 30:967–981.

Jain, K., Mahdian, M., Markakis, E., Saberi, A., e Vazirani, V. V. (2003). Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *J. ACM*, 50(6):795–824.

Jonker, R. e Volgenant, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

Kaufman, L., Eede, M. V., e Hansen, P. (1977). A plant and warehouse location problem. *Operational Research Quarterly*, 28(3):547 – 554.

Klincewicz, J. (1992). Avoiding local optima in thep-hub location problem using tabu search and grasp. *Annals of Operations Research*, 40(1):283–302.

Klose, A. e Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research*, 162(1):4 – 29.

Krarup, J. e Pruzan, P. M. (1983). The simple plant location problem: Survey and synthesis. *European Journal of Operational Research*, 12(1):36 – 81.

Kuehn, A. A. e Hamburger, M. J. (1963). A heuristic program for locating warehouses. *Management Science*, 9(4):643–666.

Laguna, M. e Martí, R. (1999). Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52.

Landete, M. e Marín, A. (2009). New facets for the two-stage uncapacitated facility location polytope. *Computational Optimization and Applications*, 44(3):487–519.

Magnanti, T. L. e Wong, R. T. (1981). Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464 – 483.

Magnanti, T. L. e Wong, R. T. (1990). Decomposition methods for facility location problems. Discrete location theory, 209-262 (1990).

Manne, A. (1964). Plant location under economies of scale. *Management Science*, 11(2):213–235.

Martins, A. X., Castro, R. R. M., e Souza, M. J. F. (2009). Algoritmos simulated annealing e grasp para o planejamento de aulas de um departamento. *Rev. Eletrônica Produção e Engenharia*, 2(1):24 – 33.

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

Mateus, G. R. e Thizy, J. M. (1999). Exact sequential choice of locations in a network. *Annals of Operations Research*, 86:199–219.

McDaniel, D. e Devine, M. (1977). A modified benders' partitioning algorithm for mixed integer programming. *Management Science*, 24(3):312–319.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Comput. Oper. Res.*, 24(11):1097–1100.

Narula, S. C. e Ogbu, U. (1979). An hierarchal location–allocation problem. *Omega*, 7(2):137–143.

Papadakos, N. (2008). Practical enhancements to the magnanti–wong method. *Operations Research Letters*, 36(4):444 – 449.

Reeves, C. R. e Yamada, T. (1998). Genetic algorithms, path relinking, and the flowshop sequencing problem. *Evol. Comput.*, 6(1):45–60.

Resende, M. e Ribeiro, C. (2003). Greedy randomized adaptive search procedures. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers.

Resende, M. G. e Werneck, R. F. (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174(1):54 – 68.

R.M., N. (1978). An improved algorithm for capacitated facility location problem. *Operations Research*, 29:1195–1201.

Ro, H. B. e Tcha, D. W. (1984). A branch and bound algorithm for the two-level uncapacitated facility location problem with some side constraints. *European Journal of Operational Research*, 18(3):349 – 358.

Verter, V. (2011). Uncapacitated and capacitated facility location problems. In Eiselt, H. A. and Marianov, V., editors, *Foundations of Location Analysis*, volume 155 of

International Series in Operations Research and Management Science, pages 25–37. Springer US.

Weber, A. (1909). Ueber den standort der industrien. *1. Teil: Reine Theorie des Standort. Verlag J.C.B. Mohr, Tubingen.*

Wolsey, L. A. (1998). *Integer programming.* Wiley-Interscience, New York, NY, USA.

You, F. e Grossmann, I. (2013). Multicut benders decomposition algorithm for process supply chain planning under uncertainty. *Annals of Operations Research*, 210(1):191–211.

Zhang, P. (2007). A new approximation algorithm for the k-facility location problem. *Theoretical Computer Science*, 384(1):126 – 135. Theory and Applications of Models of Computation.