UM ALGORITMO PARA O PROBLEMA DE LOCALIZAÇÃO NÃO CAPACITADO BASEADO EM TESTES DE REDUÇÃO E HEURÍSTICAS *ADD/DROP*

André Barros Pereira

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Cláudio Thomas Bornstein, Dr.Rer.Nat.

Prof. Roberto Diéguez Galvão, Ph.D.

Prof. Manoel Bezerra Campêlo Neto, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
DEZEMBRO DE 2002

PEREIRA, ANDRÉ BARROS

Um algoritmo para o Problema de Localização Não Capacitado baseado em testes de redução e heurísticas *ADD/DROP* [Rio de Janeiro] 2002

X, 108 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2002)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 - Problema de Localização Não Capacitado

2 - Testes de redução

3 - Heurísticas ADD/DROP

I. COPPE/UFRJ II. Título (série)

Agradecimentos

A Deus.

Aos meus pais Amarílio e Mirtes e aos meus irmãos Adriana e Júnior, pelo amor, carinho, incentivo e compreensão sobretudo nesta época em que estive ausente.

À minha namorada Daniele Coelho que, apesar da distância, sempre se fez muito presente.

Ao professor Cláudio Bornstein, pela atenção, paciência e dedicação com as quais conduziu a orientação deste trabalho. Ao professor Manoel Campêlo pelas sugestões e revisões no texto da dissertação e ao professor Roberto Galvão pelas sugestões e por todo o material cedido que foi de grande valia para a realização deste trabalho.

A todos os meus professores de graduação, especialmente aos professores Marcos Negreiros e Wamberto Vasconcelos e aos professores do mestrado, em especial aos professores Nelson Maculan e Adilson Xavier.

Aos amigos Cláudio Prata, Tiberius Bonates e Douglas Valiati, pelos valiosos auxílios na realização deste trabalho.

Aos amigos que tive o prazer de conviver no Rio: Jorge Bergson, Sérgio Assunção, Mara Franklin, Talita Oliveira, Bruno Muniz, Marcelo Quinderé, Henrique Limaverde, Elder Macambira, Fábio Rabelo, Leonardo Holanda, Oscarina Viana e Rômulo Martins.

Aos amigos que fiz no Rio: Alexandre Henrique Porto, Herman Walenkamp, Joana Loureiro, Ana Lúcia Pimentel, Rosa Figueiredo, Ana Flávia Macambira, Rafael e Letícia Lopes, Moisés Junior, Luidi Simonetti, Carlos Henrique Sabóia, Michele Silva entre outros.

Aos meus parentes residentes no Rio: Dalva, Juarez e Juliana Sales e Margarida Costa.

A todos os parentes e amigos de Fortaleza, de uma forma muito especial. Citar os nomes significaria estender-me demais e ainda correr o risco de deixar alguém de fora por uma falha de minha memória.

A todos os funcionários do Programa. Em especial a Lourdes e Gercina.

Ao CNPq pela bolsa de estudos concedida.

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

UM ALGORITMO PARA O PROBLEMA DE LOCALIZAÇÃO NÃO CAPACITADO BASEADO EM TESTES DE REDUÇÃO E HEURÍSTICAS *ADD/DROP*

André Barros Pereira

Dezembro/2002

Orientador: Cláudio Thomas Bornstein

Programa: Engenharia de Sistemas e Computação

Utilizamos testes de redução na solução do Problema de Localização Não Capacitado. Infelizmente, nem sempre é possível solucionar instâncias deste problema utilizando somente esses testes. Então, desenvolvemos um conjunto de heurísticas *ADD/DROP* e as combinamos aos testes, dando origem a um algoritmo heurístico. Fizemos uma implementação desse algoritmo e a submetemos a uma série de testes computacionais. Nossos resultados foram comparados aos de uma implementação do algoritmo de Galvão e Raggi [21].

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

AN ALGORITHM FOR THE SIMPLE PLANT LOCATION PROBLEM BASED ON REDUCTION TESTS AND *ADD/DROP* HEURISTICS

André Barros Pereira

December/2002

Advisor: Cláudio Thomas Bornstein

Department: Computing and Systems Engineering

We begin using reduction tests to solve the Simple Plant Location Problem. Unfortunately, these tests are generally unable to fix the status of all facilities. Therefore, ADD/DROP heuristics are additionally used. The resulting algorithm was implemented and computational tests were made. The results were compared to the Galvão and Raggi's [21] algorithm.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Quando se pretende construir um *shopping center* em uma cidade, uma das principais preocupações que se tem é a escolha de sua localização. Isso acontece porque a região em que ele será instalado tem influência direta no desempenho dos negócios. Eis alguns fatores que comprovam esse fato:

1. *Facilidade de acesso de pedestres e veículos ao shopping.* Contribui para que mais clientes dêem preferência a ele para a aquisição de bens e serviços. Dessa forma, é importante que seus arredores sejam servidos de uma boa malha viária.

2. *Proximidade em relação a clientes em potencial.* Populações de baixa renda normalmente não são freqüentadoras assíduas de *shopping centers* para suprir suas necessidades de consumo. Em bairros onde predominam essas populações normalmente não se tem expectativas de se fazer bons negócios com a instalação de um *shopping* e, portanto, essa instalação passa a ser indesejada. De um modo geral, comércios de menor porte tais como mercearias, feiras-livres e pequenos merca-

dos predominam nesses locais. Por outro lado, populações de faixas salariais mais elevadas costumam adquirir não só bens em *shoppings* mas usufruem de diversos serviços. Assim, os hábitos de consumo das pessoas que residem nas proximidades de onde se pretende instalar o *shopping*, e que estão diretamente relacionados aos seus níveis salariais, também são importantes fatores.

3. *Preferencialmente ele deve ser instalado em locais em que haja pouca ou nenhuma concorrência.* Isso diz respeito à segurança de se manter uma clientela fiel. Quanto mais empreendimentos equivalentes existirem no mesmo local, mais provável será que a população local seja dividida em termos de preferência em relação a estes.

4. *Acessibilidade do preço de compra ou aluguel do terreno em que se pretende construir o shopping.* Esses preços variam bastante conforme a valorização imobiliária do local.

5. *Boas condições ambientais.* Um lugar de intenso barulho, poluição, etc, certamente prejudicaria a permanência de clientes. Esse é um importante fator a considerar pois diz respeito ao bem-estar dos clientes enquanto estão no *shopping*. Se suas permanências são comprometidas, certamente também estarão os negócios.

Esse exemplo é uma típica aplicação da localização no setor privado, onde a minimização de custos e a maximização de lucros é desejada. Nesse caso, uma má localização poderia ocasionar grandes perdas financeiras. Em aplicações de localização características do setor público, o que geralmente se almeja é distribuir serviços de forma a atender mais eficientemente a população. As conseqüências de uma má localização nessas aplicações podem ser

bem piores, como no caso de localizar ambulâncias em uma cidade de forma a atender com eficiência as chamadas de emergência da população: o tempo de atendimento de uma chamada é um fator crucial e pode ser bastante comprometido por uma localização precária, o que pode levar a perda de vidas humanas. Uma discussão sobre problemas de localização nos setores público e privado pode ser vista em Revelle e outros [52].

De forma similar, inúmeros problemas práticos envolvem tomar decisões em relação a onde localizar entidades fornecedoras de bens e/ou serviços (usualmente denominadas *facilidades*), dentro de um conjunto de possíveis locais, visando atender de forma mais vantajosa a demanda por esses bens e/ou serviços nas regiões onde se pretende instalá-las. Podemos encontrar um grande número de situações reais desse tipo em Daskin [17]. Esses problemas são chamados genericamente de *problemas de localização de facilidades.*

Mais formalmente, o problema de localização de facilidades pode ser definido, baseado em Mateus e Carvalho [45], como aquele no qual facilidades devem ser alocadas entre n possíveis locais com o objetivo de minimizar o custo total em satisfazer a demanda distribuída em m locais. O custo total consiste no custo fixo de instalar as facilidades e os custos variáveis para atender a demanda (também conhecidos por custos de distribuição, custos de transporte ou custos de alocação).

Nesse trabalho, desenvolvemos um algoritmo heurístico para a solução de um modelo específico de localização (*Simple Plant Location Problem*) e fizemos uma implementação computacional desse algoritmo. Dividiremos a apresentação do trabalho em 5 capítulos: no primeiro capítulo, situaremos o problema que trabalharemos dentro do contexto geral dos modelos

de localização. Os fundamentos teóricos do nosso algoritmo serão apresentados no Capítulo 2 e no terceiro capítulo exibiremos o algoritmo. No Capítulo 4 serão exibidos os testes computacionais que fizemos com nossa implementação e faremos comparações com o desempenho computacional de uma implementação do algoritmo de Galvão e Raggi [21]. Finalmente, no último capítulo, faremos algumas conclusões e apontaremos alguns tópicos do trabalho que poderão ser futuramente trabalhados.

A seguir, situaremos nosso modelo dentro de uma classificação taxionômica dos modelos de localização (seção 1.1), especificamente dentro dos modelos discretos de localização (seção 1.2). Veremos algumas técnicas de se transformar problemas reais de localização em modelos de localização (seção 1.3), bem como as principais técnicas de soluções desses modelos (seção 1.4). Apresentaremos ainda algumas características específicas do nosso problema (seção 1.5) e exibiremos seu estado-da-arte (seção 1.6).

1.1 Breve taxionomia dos problemas de localização

Os problemas de localização de facilidades podem ser classificados de diversas formas. Apresentaremos algumas classificações e situaremos os problemas que utilizaremos no nosso trabalho de acordo com elas. Taxionomias completas sobre problemas de localização de facilidades podem ser obtidas em Daskin [17] e Brandeau e Chiu [10].

1.1.1 Problemas capacitados e não-capacitados

Se para cada local candidato a localização de facilidades for estabelecida uma capacidade máxima de oferta, o problema é dito *capacitado*. Se essa capacidade for ilimitada, o problema será denominado *não-capacitado*.

A ênfase do nosso trabalho será no tratamento de problemas de localização de facilidades não-capacitados.

1.1.2 Problemas planares, em rede e discretos

Uma das principais diferenças entre os problemas de localização é a forma com que a demanda e as facilidades são representadas e como são estabelecidos os custos de transporte entre eles. Dividiremos em três tipos, segundo Chhajed e outros[11]: planares, em rede e discretos.

Nos problemas *planares*, a demanda ocorre em qualquer lugar em um plano. Normalmente ela é representada através de uma função que estabelece a demanda em cada coordenada (X, Y) do plano. Em tais problemas, facilidades podem ser localizadas em quaisquer pontos do plano. Encontramos discussões sobre esses modelos em Hurter e Martinich [28] e Love e outros [41].

Já nos modelos de localização *em rede*, a localização de facilidades se dá em um número finito de locais, objetivando atender a demanda associada a

um número também finito de locais. Estes últimos são usualmente chamados de *centros de demanda* ou *clientes*. Nesses modelos, as facilidades e os centros de demanda se localizam nos nós ou arcos de uma rede específica. Os custos de transporte entre as facilidades e centros de demanda são obtidos a partir dessa rede. Modelos em rede são discutidos em Handler e Mirchandani [27] e Mirchandani e Francis [47].

Os modelos *discretos* de localização também se caracterizam por possuirem um número finito de facilidades e centros de demanda mas diferem dos modelos em rede porque permite-se atribuir custos arbitrários de transporte entre as facilidades e centros de demanda, independentes de uma rede fixa.

A ênfase do nosso trabalho será na solução de problemas discretos de localização. Entretanto, como veremos adiante, resolveremos também problemas em rede, transformando-os em modelos discretos.

1.1.3 Problemas euclideanos e não-euclideanos

Os modelos de localização também se caracterizam pela métrica utilizada. Em alguns problemas de localização, os custos de transporte entre facilidades e demanda podem ser representados simplesmente pela distância entre eles. De acordo com a métrica obedecida no cálculo dessas distâncias, esses problemas podem ser classificados como *euclideanos* ou *não-euclideanos*. Nos primeiros, para quaisquer três pontos distintos i, j e k, valem as relações:

1. $d(i,i) = 0$;

2. $d(i,j) > 0$;

3. $d(i,j) = d(j,i)$;

4. $d(i,j) < d(i,k) + d(k,j)$;

onde $d(x,y)$ representa a distância entre os pontos x e y. Se pelo menos um desses quatro itens for violado, o problema será dito não-euclidiano.

No nosso trabalho, contemplaremos a solução de ambos os tipos de problemas.

1.2 Modelos discretos de localização

Definiremos nesta seção a formulação matemática para a qual desenvolvemos nosso algoritmo. Mas, antes disso, introduziremos os principais modelos não-capacitados, a começar pelo mais abrangente deles, o *Problema Geral de Localização Não Capacitado (PGLNC)*. Baseado em Mateus e Carvalho [45] e Galvão e Raggi [21], este problema pode ser assim definido:

$(PGLNC)$:

$$min \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \tag{1.1}$$

$$sujeito\ a: \quad \sum_{i \in I} x_{ij} = 1, \forall j \in J \tag{1.2}$$

$$\sum_{i \in I} y_i \leq p \tag{1.3}$$

$$x_{ij} \leq y_i, \forall i \in I, \forall j \in J \tag{1.4}$$

$$x_{ij} \geq 0, \forall i \in I, \forall j \in J \tag{1.5}$$

$$y_i \in \{0,1\}, \forall i \in I \tag{1.6}$$

onde $I = \{1, ..., n\}$ é o conjunto de locais candidatos à localização de facilidades e $J = \{1, ..., m\}$ é o conjunto de centros de demanda. f_i representa o custo fixo de instalar uma facilidade em um local $i \in I$ e c_{ij} o custo de transporte em atender a demanda em $j \in J$ pela facilidade $i \in I$. A variável x_{ij} corresponde à parcela da demanda de $j \in J$ atendida por $i \in I$ e $y_i, i \in I$ é uma variável binária igual a 1 se a facilidade é localizada em i ou 0, caso contrário. É comum usar o adjetivo *aberta* para designar uma facilidade que será localizada na solução e *fechada* para uma facilidade não localizada.

A função objetivo (1.1) minimiza a soma dos custos variáveis e fixos. Em (1.2) garante-se que a demanda é atendida em cada centro $j \in J$. (1.3) limita o número de facilidades abertas a p. A restrição (1.4) estabelece uma relação entre as variáveis x_{ij} e y_i e assegura que se o centro de demanda $j \in J$ é servido pela facilidade $i \in I$, então necessariamente há uma facilidade aberta em i. Em (1.5) determina-se que as parcelas x_{ij} são não-negativas e em (1.6) estabelece-se que a variável y_i é binária.

Percebemos que o PGLNC possui variáveis reais e inteiras. Entretanto, criaremos um modelo de programação linear inteira binária, acrescentando a seguinte restrição ao PGLNC: um centro de demanda $j \in J$ somente pode ser atendido por uma única facilidade $i \in I$. Ou seja, basta substituir no PGLNC a restrição (1.5) por:

$$x_{ij} \in \{0, 1\}, \forall i \in I, \forall j \in J \tag{1.7}$$

Denominaremos o modelo resultante, assim como é feito em Mateus e Carvalho [45], por *Problema de Localização Não Capacitado (PLNC)*. Ele será:

$(PLNC)$:

$$min \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \tag{1.8}$$

$$sujeito\ a: \quad \sum_{i \in I} x_{ij} = 1, \forall j \in J, \tag{1.9}$$

$$\sum_{i \in I} y_i \leq p \tag{1.10}$$

$$x_{ij} \leq y_i, \forall i \in I, \forall j \in J \tag{1.11}$$

$$x_{ij} \in \{0,1\}, \forall i \in I, \forall j \in J \tag{1.12}$$

$$y_i \in \{0,1\}, \forall i \in I \tag{1.13}$$

Observamos que o PGLNC e o PLNC são equivalentes. Como as facilidades do modelo não possuem restrições de capacidade, é fácil observar que sempre haverá uma solução ótima do PGLNC em que as variáveis x_{ij} são binárias. Basta, em uma dada solução ótima do PGLNC, para cada centro de demanda atendido por mais de uma facilidade na solução, associá-lo a uma única facilidade dentre essas, o que certamente não incorrerá em acréscimo no valor da função objetivo na solução encontrada.

A partir do PLNC, outros conhecidos modelos não-capacitados podem ser obtidos. Por exemplo, seja N o conjunto de vértices de uma rede. Se fizermos, no PLNC, $N \equiv I \equiv J$, $f_i = 0, \forall i \in N$ e substituirmos a restrição (1.10) por

$$\sum_{i \in N} y_i = p \tag{1.14}$$

teremos o clássico *Problema de P-Medianas (P-MED)*:

$(P-MED)$:

$$
\begin{aligned}
min \quad & \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \\
sujeito\ a: \quad & \sum_{i \in N} x_{ij} = 1, \forall j \in N \\
& \sum_{i \in N} y_i = p \\
& x_{ij} \leq y_i, \forall i \in N, \forall j \in N \\
& x_{ij}, y_i \in \{0,1\}, \forall i \in N, \forall j \in N
\end{aligned}
$$

onde os custos c_{ij} correspondem às distâncias entre os nós i e j obtidas através dos arcos da rede. Se, no P-MED, os custos fixos forem considerados, denominamos o modelo resultante por *Problema de P-medianas de Custo Fixo (P-MEDCF).*

Uma outra modificação que pode ser feita no PLNC é estabelecermos $p = \mid I \mid$ na restrição (1.10). Esta restrição se tornará supérflua no modelo, podendo portanto ser retirada. Teremos então, usando a mesma nomenclatura de Galvão e Raggi [21], Mateus e Carvalho [45] e Körkel [37], o *Simple Plant Location Problem (SPLP).* Nosso algoritmo foi desenvolvido para essa formulação. Eis o modelo:

$(SPLP)$:

$$
\begin{aligned}
min \quad & \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i && (1.15) \\
sujeito\ a: \quad & \sum_{i \in I} x_{ij} = 1, \forall j \in J && (1.16) \\
& x_{ij} \leq y_i, \forall i \in I, \forall j \in J && (1.17) \\
& x_{ij}, y_i \in \{0,1\}, \forall i \in I, \forall j \in J && (1.18)
\end{aligned}
$$

Diferentemente de outros problemas, na literatura há muita divergência a respeito da nomenclatura desse problema. Na língua inglesa, por exemplo, Krarup e Pruzan [35] apresentam até um arquétipo dos nomes usuais

que ele apresenta, como observamos na Figura 1.1. Segundo eles, incluindo omissões dos adjetivos, mais de 10 nomes já foram dados na literatura a partir desse arquétipo. Por exemplo, Erlenkotter [19] e Al-Sultan e Al-Fawzan [2] o denominam *Uncapacitated Facility Location Problem* enquanto Khumawala [32] usa o termo *Uncapacitated Warehouse Location Problem* para descrever esse problema.

Adjetivo		Substantivo		Expressão
Uncapacitated Simple Optimal	+	Plant Warehouse Facility Site	+	Location Problem

Figura 1.1: Arquétipo das usuais denominações do SPLP na literatura

Os modelos de localização que apresentamos são bem gerais e são considerados clássicos na literatura. Entretanto muitos outros modelos têm sido desenvolvidos incorporando diversas características intrínsecas a problemas do mundo real. Uma boa coletânea desses modelos pode ser vista em Revelle e Laporte [53].

Faremos na seção seguinte um breve comentário sobre a relação entre um problema real de localização de facilidades e sua modelagem matemática.

1.3 Modelagem matemática de problemas de localização

O processo de se abstrair de algumas particularidades de problemas práticos, representando-as por expressões matemáticas é conhecido por *modelagem matemática*. A quantidade de elementos do problema real a serem considerados, a forma de representação desses elementos e os objetivos da representação são fatores que distinguem uma modelagem matemática de outra para o mesmo problema.

Os *modelos matemáticos de localização* estabelecem essencialmente a representação das facilidades em potencial, dos centros de demanda e dos custos inerentes ao atendimento da demanda pelas facilidades. Tais modelos objetivam minimizar esses custos, respondendo a questões do tipo:

1. Quantas facilidades devem ser localizadas?
2. Onde cada facilidade será localizada?
3. De que forma as facilidades localizadas serão alocadas aos centros de demanda?

Na prática, o que se faz normalmente é modelar o problema de uma forma que se aproxime de um dos típicos modelos de localização, como os que vimos na seção anterior. Pois, para estes, diversas técnicas de solução já foram desenvolvidas e trabalhadas ao longo do tempo, como veremos na próxima seção.

Para exemplificar o processo de modelagem matemática de problemas de localização, consideremos novamente o exemplo do *shopping center*. Suponha, entretanto, que desejamos instalar não apenas um *shopping* mas diversas unidades de uma rede de *shopping centers*. Queremos que essa rede seja instalada com o menor custo possível. Consideremos uma modelagem simplificada desse problema como SPLP:

O conjunto de locais candidatos à instalação de unidades da rede de *shoppings* seria modelado como o conjunto de facilidades em potencial (I). Os custos monetários estimados de instalação de cada unidade da rede seriam modelados como os custos fixos de instalação das facilidades (f_i).

Fazendo levantamentos sobre os hábitos de consumo das populações em torno dos locais onde se pretende instalar unidades da rede de *shoppings*, pode-se delimitar dentro de uma mesma região, agrupamentos de clientes em potencial, cada um com sua respectiva demanda associada, que pode ser uma média dos gastos com produtos e serviços em *shoppings* em um determinado intervalo de tempo. A distância entre as unidades do *shopping* e esses agrupamentos representam dificuldades no atendimento dessa demanda. Cada grupo de clientes pode ser modelado como um centro de demanda. O conjunto de agrupamentos seria, no modelo, o conjunto de centros de demanda (J). De uma forma simplificada, os custos variáveis de se atender as demandas nesses centros podem ser dados pelo produto entre o valor da demanda média nesses locais e a distância entre eles e o local candidato à instalação do *shopping* ($c_{ij}, \forall i \in I, \forall j \in J$).

A decisão de instalar ou não uma unidade da rede de *shoppings* no modelo seria retratada pela variável y_i e o fato de uma unidade estar ou não aten-

dendo um aglomerado de clientes em potencial seria indicado pela variável x_{ij}.

A função objetivo seria a função de custos monetários estimada para que se concretizasse as instalações dos *shoppings* e fossem atendidas as demandas.

O modelo consistiria então em minimizar a função (1.15) sujeito às restrições (1.16), (1.17) e (1.18).

Uma modelagem um pouco mais sofisticada desse problema poderia, por exemplo, contemplar de alguma forma os aspectos citados no início deste capítulo, que tornam relevantes a escolha da localização do *shopping*:

1. *Facilidade de acesso de pedestres e veículos ao shopping.* A presença de uma malha viária deficiente em torno do *shopping* seria uma qualidade bastante indesejável e poderia ser modelada através de uma variável $p_1(i), i \in I$ que penalizaria os custos fixos f_i.

2. *Proximidade em relação a clientes em potencial.* Já contemplado através da definição de c_{ij}

3. *Preferencialmente ele deve ser instalado em locais em que haja pouca ou nenhuma concorrência.* A presença de um concorrente próximo a uma unidade do *shopping* poderia ser igualmente modelada como uma penalização de f_i. Utilizaremos uma variável $p_2(i), i \in I$ para esse fim. Além disso, a própria solução tenderia a evitar que uma unidade da rede de *shoppings* fosse colocada perto de outra da mesma rede.

4. *Acessibilidade do preço de compra ou aluguel do terreno em que se pretende construir o shopping.* Essa característica já está embutida na

definição dos custos fixos de instalação das facilidades.

5. *Boas condições ambientais.* Finalmente, uma terceira variável $(p_3(i), i \in I)$ indicaria a penalização que os custos fixos f_i sofreriam de acordo com os problemas ambientais da região onde se quer instalar o *shopping*.

Nesse novo modelo, o valor dos novos custos f_i' seriam dados por: $f_i' = f_i + p_1(i) + p_2(i) + p_3(i), \forall i \in I$. Os demais elementos permaneceriam inalterados.

1.4 Técnicas de soluções de problemas de localização

A principal dificuldade na solução de problemas de localização é a quantidade de mínimos locais que a função objetivo apresenta dentro do conjunto de soluções viáveis do problema e que nem sempre constituem mínimos globais ou mesmo boas soluções. Uma completa enumeração desses pontos é impraticável para a maioria dos problemas. Por isso, várias técnicas têm sido desenvolvidas visando determinar a melhor maneira de enumerar e selecionar esses pontos de forma que o mínimo global ou uma solução próxima dele possa ser obtida. Dividiremos a apresentação das técnicas de acordo com a classificação dos modelos de localização em planares, em rede ou discretos.

Os problemas planares se caracterizam por uma formulação contínua e para resolvê-los normalmente são empregadas técnicas de programação não-linear, geometria computacional e/ou otimização global. Um levantamento

da aplicação dessas técnicas a problemas planares pode ser encontrado em Plastria [50].

Existem duas abordagens principais na solução dos modelos em rede:

- Procedimentos que reduzem tais problemas a modelos discretos que garantidamente reproduzem a solução ótima dos modelos em rede. Em Daskin [17] podemos encontrar procedimentos como estes.
- Algoritmos que utilizam eficientemente a estrutura especial da rede para solucionar os modelos. Podemos obter mais detalhes sobre esses algoritmos em Labbe e outros [39].

Os modelos discretos de localização são normalmente constituídos por modelos de programação linear inteira ou modelos de programação linear inteira mista e geralmente técnicas de natureza combinatória são empregadas na solução desses problemas. Em Mateus e Carvalho [45] temos um excelente levantamento das principais técnicas empregadas. Eles dividem essas técnicas em 5 áreas:

- **Decomposição**: elas seguem, em geral, o método de decomposição de Benders. Como as formulações de alguns problemas envolvem variáveis inteiras e reais, este método separa a parte inteira da parte real, resolvendo problemas menores conectados a um problema mestre.
- **Enumeração**: procuram subdividir o problema original em subproblemas menores de fácil resolução, enumerando, implicitamente ou explicitamente, todos os pontos extremos. Os algoritmos de enumeração são basicamente do tipo de Separação e Avaliação (*branch-and-bound*),

que são técnicas de busca em árvore, onde cada nó da árvore representa um subconjunto de soluções viáveis. Um algoritmo de separação e avaliação fica caracterizado ao definirmos como separar o conjunto de soluções viáveis e como avaliar a função determinando limites superiores e inferiores.

- **Heurísticas**: objetivam determinar boas soluções de forma eficiente. Com essa prioridade, as heurísticas caracterizam-se pela flexibilidade e simplicidade computacional.

- **Métodos duais**: procuram resolver o dual do problema com o objetivo de obter limites inferiores justos.

- **Métodos baseados em Programação Linear**: determinam limites inferiores para os modelos, relaxando as restrições de integralidade e resolvendo o Problema de Programação Linear resultante.

Conforme mencionamos, nosso algoritmo foi desenvolvido direcionado à solução de modelos discretos de localização, mais especificamente, para o SPLP. Em nosso trabalho, fizemos uso sobretudo de heurísticas.

1.5 Simple Plant Location Problem

O SPLP é um problema que tem sido vastamente estudado na literatura. Muitos algoritmos têm aproveitado suas particularidades a fim de obter soluções de boa qualidade, em tempos razoáveis. Antes de listarmos uma coletânea de trabalhos sobre o SPLP, vamos conhecer algumas das propriedades desse problema:

1. *O SPLP é um problema NP-difícil*, como podemos observar em Garey e Johnson [23] e Lenstra e Kan [40]. Porém, em alguns casos especiais, ele é solucionável polinomialmente como vemos em Kolen [33] e Krarup e Bilde [34].

2. *O SPLP pode ser matematicamente decomposto em dois subproblemas interdependentes.*

   De acordo com Al-Sultan e Al-Fawzan [2], os subproblemas são:

   - **Localização**: determinar as facilidades a serem localizadas ($y_i's$).

   - **Alocação**: para aquelas facilidades localizadas, estabelecer como será a distribuição aos centros de demanda ($x'_{ij}s$)

   Para cada solução do subproblema de localização, uma solução ótima do subproblema de alocação é facilmente obtida. Mais especificamente, para qualquer dado vetor y, uma associação ótima aos valores de $x'_{ij}s$ pode ser obtida usando a seguinte fórmula:

$$k_j = \arg \min_{y_i=1, i \in I} c_{ij}$$

   Para todo $i \in I$ e $j \in J$:

$$x_{ij} = \begin{cases} 1, & \text{se } i = k_j \\ 0, & \text{caso contrário} \end{cases}$$

Utilizamos para simplificar a definição de k_j e utilizaremos outras vezes mais adiante a notação "arg". Para $K \subseteq I$, definimos

$$j = \arg \min / \max_{i \in K} f(i) \Leftrightarrow f(j) = \min / \max_{i \in K} f(i)$$

3. *O SPLP se relaciona aos problemas de empacotamento, cobertura e particionamento*, como se observa em Krarup e Pruzan [35].

1.6 Trabalhos relacionados

A primeira formulação explícita do SPLP data da década de 60 e é freqüentemente atribuída a Balinski [4]. Esse trabalho foi apresentado no simpósio *The IBM Scientific Symposium on Combinatorial Problems* em março de 1964 mas permaneceu sem ser publicado até o ano de 1966. Entretanto, alguns pesquisadores acreditam que o SPLP já teria sido abordado em trabalhos anteriores a este (ver Krarup e Pruzan [35]).

Vamos exibir o estado-da-arte do SPLP, de acordo com a classificação das técnicas de soluções vistas anteriormente, embora, em muitos trabalhos, mais de uma técnica seja empregada, como poderemos observar adiante. A literatura sobre esse problema é muito vasta e citaremos aqui alguns dos principais trabalhos.

Os algoritmos de decomposição, seguindo a linha Benders, conseguiram bons resultados através da geração de cortes no conjunto de soluções viáveis

como se observa em Magnanti [42] e Wolsey e Nemhauser [60].

Na linha de Separação e Avaliação, logo em seguida ao trabalho de Balinski, Efroymson e Ray [18] propuseram um algoritmo para uma formulação modificada do SPLP. Khumawala [32] utilizou a estrutura especial do SPLP para melhorar o algoritmo de separação e avaliação de Efroymson e Ray. Guignard [25] propôs um algoritmo que utiliza desigualdades de Benders geradas durante um procedimento dual ascendente lagrangeano. Galvão e Raggi [21] propuseram um algoritmo em três fases para o PLNC, que tem o SPLP como um caso especial.

Entre as heurísticas, destacamos as gulosas ADD propostas em Kuehn e Hamburguer [38], que procuram a cada iteração selecionar a facilidade mais econômica. Uma heurística que descarta a facilidade menos econômica a cada iteração foi desenvolvida em Feldman e outros [20] e foi denominada DROP. Manne [43] desenvolveu uma heurística denominada SAOPMA que combinava as duas anteriores. Rapp [51] e Cooper [13], independentemente, também combinaram ADD e DROP e desenvolveram uma heurística conhecida por ALA (Alternate Location Allocation). Uma variação de ALA chamada SHIFT foi desenvolvida por Kuehn e Hamburguer [38]. Conforme seja usada inicialmente a heurística ADD ou DROP teremos as heurísticas SHIFT ou ALA. Destaca-se também o método de substituição de vértices desenvolvido por Teitz e Bart [58], denominado VSM (Vertex Substitution Method) desenvolvido em princípio para o P-MED mas que foi adaptado para o SPLP por Cornuéjols e outros [14]. Mais adiante, Beasley [5] propôs heurísticas lagrangeanas para vários problemas de localização que incluem o SPLP. As metaheurísticas também têm sido amplamente aplicadas ao SPLP.

Nessa linha, destacamos os trabalhos de Al-Sultan e Al-Fawzan [2] e Michel e Hentenryck [46] que apresentaram algoritmos baseados na metaheurística busca tabu; Alves e Almeida [3] utilizaram *Simulated Annealing* na solução do problema e Kratica e outros [36] fizeram uso de um algoritmo genético.

Bilde e Krarup [7] e Erlenkotter [19] introduziram os métodos duais para a solução do SPLP. Mais tarde, Tcha e outros [57] desenvolveram uma heurística dual para esse problema bastante similar ao procedimento de Erlenkotter, afirmando que ela produz soluções que são, na maioria dos casos, superiores àquelas alcançadas pelo procedimento de Erlenkotter com um leve aumento no tempo computacional. Körkel [37] mostrou como modificar uma versão primal-dual do algoritmo exato de Erlenkotter para conseguir um procedimento melhorado. Conn e Cornuéjols [12] sugeriram um novo método baseado na solução exata do dual de uma relaxação linear do SPLP via projeções ortogonais. Gao e Robinson [22] apresentaram um modelo geral e um procedimento dual de separação e avaliação de solução para encontrar soluções ótimas para vários problemas de localização, incluindo o SPLP.

Nos métodos baseados em programação linear, destacam-se os trabalhos Spielberg [56], Schrage [54] e Garfinkel e outros [24], que exploram o fato de as variáveis serem limitadas superiormente. Simão e Thizy [55] desenvolveram um algoritmo dual-simplex direcionado ao problema. Guignard e Spielberg [26] e Cornuéjols e Thizy [16] aplicaram um algoritmo primal de subgradiente. Mais recentemente, Jain and Vazirani [30] formularam algoritmos primal-dual na solução de problemas não capacitados, dentre eles, o SPLP.

Capítulo 2

Fundamentação Teórica

Veremos, a seguir, as bases teóricas que fundamentam o nosso algoritmo para a solução do SPLP. Basicamente, utilizamos testes de redução e desenvolvemos um conjunto de heurísticas ADD/DROP. Vejamos em detalhes esses fundamentos:

2.1 Testes de redução

Os testes de redução determinam, *a priori*, se algumas facilidades devem ser abertas ou fechadas na solução ótima. O termo *redução* se deve ao fato de se conseguir, desta maneira, reduzir a dimensão do problema dado originalmente.

O uso das regras de redução é encontrado originalmente nos trabalhos de Efroymson e Ray [18] e Khumawala [32]. Posteriormente, diversos trabalhos foram desenvolvidos, estendendo alguns dos resultados dos primeiros

e aplicando-os ao problema de localização capacitado, como podemos observar em Akinc e Khumawala [1], que utiliza os testes dentro de um algoritmo de separação e avaliação. Jacobsen [29] e Mateus e Bornstein [44] utilizaram esses testes dentro de heurísticas ADD/DROP. Mais recentemente, temos em Bornstein e Azlan [8] a combinação desses testes à metaheurística *simulated annealing*. Em Campêlo e Bornstein [49], temos uma nova abordagem que combina os testes de redução a heurísticas ADD/DROP, que são implementadas com a ajuda de limites superiores e inferiores, usando relaxação lagrangeana.

Utilizaremos, em nosso trabalho, a abordagem unificada encontrada em Bornstein e Azlan [8] e Campêlo e Bornstein [49], que simplifica a apresentação dos testes de redução através da utilização da função $\Delta_i(.)$. Essa função nos dará os critérios para abertura/fechamento de facilidades. Mais adiante, vamos defini-la formalmente mas antes definiremos os elementos constituitivos dessa função:

Considere um subconjunto $K \subseteq I$ de facilidades. Seja $w(K)$ a função que nos dá o mínimo valor dos custos variáveis em se atender todos os centros de demanda de J pelas facilidades de K. Ou seja,

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

Se $K = \emptyset$, definimos $w(K) = +\infty$.

Para $i \in I - K$, seja $\delta_i(K) = w(K) - w(K \cup i)$. Essa função avalia o acréscimo/decréscimo nos custos variáveis se fechamos/abrimos a facilidade i. Se $K = \emptyset$ então $\delta_i(K) = \delta_i(\emptyset) = +\infty - w(i) = +\infty - \sum_{j \in J} c_{ij} = +\infty$. Se $K \neq \emptyset$, podemos alternativamente escrever a função $\delta_i()$ em função dos

custos variáveis c_{ij}, da seguinte forma:

$$\begin{aligned}
\delta_i(K) &= w(K) - w(K \cup i) \\
&= \sum_{j \in J} \min_{k \in K} c_{kj} - \sum_{j \in J} \min_{k \in K \cup i} c_{kj} \\
&= \sum_{j \in J} \min_{k \in K} c_{kj} - \sum_{j \in J} \min\{\min_{k \in K} c_{kj}, c_{ij}\} \\
&= \sum_{j \in J} \max\{0, \min_{k \in K} c_{kj} - c_{ij}\}
\end{aligned}$$

Definimos então a função $\Delta_i(.)$ da seguinte forma $\Delta_i(K) = f_i - \delta_i(K), \forall i \in I - K$. Essa função avalia o balanço entre os custos fixos e variáveis em relação a facilidade i. Se $K = \emptyset$ então $\Delta_i(\emptyset) = f_i - \delta_i(\emptyset) = f_i - \infty = -\infty$. Para $K \neq \emptyset$, podemos escrever:

$$\Delta_i(K) = f_i - \sum_{j \in J} \max\{0, \min_{k \in K} c_{kj} - c_{ij}\}$$

Podemos destacar duas importantes propriedades da função $w(K)$, de acordo com Wolsey [59] e Nemhauser e outros [48]: ela é não-crescente e supermodular. Uma função $w(K)$ é denominada não-crescente se $w(K) \leq w(K'), \forall K' \subseteq K$. E ela é dita supermodular (ou equivalentemente $-w(K)$ submodular) se $w(K) - w(K \cup i) \leq w(K') - w(K' \cup i), \forall K' \subseteq K, \forall i \notin K$. Observa-se que supermodularidade é uma espécie de concavidade. Mostremos que essas duas propriedades se verificam.

Propriedade 1: w() é não-crescente.

Prova:

Sejam $K \subseteq I$ e $K' \subseteq K$. Devemos mostrar que $w(K) \leq w(K')$. Vamos dividir em dois casos:

- Se $K' = \emptyset$ então temos imediatamente que $w(K') = w(\emptyset) = +\infty \geq$

$w(K)$.

- Se $K' \neq \emptyset$ então, por definição

$$w(K) = \sum_{j \in J} \min_{k \in K} c_{kj} \quad \text{e} \quad w(K') = \sum_{j \in J} \min_{k \in K'} c_{kj}$$

Temos, uma vez que $K' \subseteq K$, o seguinte fato

$$\min_{k \in K} c_{kj} \leq \min_{k \in K'} c_{kj}, \forall j \in J$$

Então,

$$\underbrace{\sum_{j \in J} \min_{k \in K} c_{kj}}_{w(K)} \leq \underbrace{\sum_{j \in J} \min_{k \in K'} c_{kj}}_{w(K')}$$

Logo, $w(K) \leq w(K'), \forall K' \subseteq K$. $\square$

Como conseqüência da propriedade 1, temos que:

$$\delta_i(K) \geq 0, \forall K \subseteq I, \forall i \notin K \tag{2.1}$$

De fato, $K \subseteq K \cup i \rightarrow w(K) \geq w(K \cup i) \rightarrow w(K) - w(K \cup i) \geq 0 \rightarrow \delta_i(K) \geq 0$.

Propriedade 2: w() é supermodular.

Prova:

Sejam $K \subseteq I$, $K' \subseteq K$ e $i \notin K$. Devemos mostrar que $w(K) - w(K \cup i) \leq w(K') - w(K' \cup i)$. Mas, temos que $\delta_i(K) = w(K) - w(K \cup i)$, logo, o que se quer provar é que $\delta_i(K) \leq \delta_i(K')$. Vamos dividir em dois casos:

- Se $K' = \emptyset$ então, de forma imediata, temos que $\delta_i(K') = \delta_i(\emptyset) = \infty \geq \delta_i(K)$.

- Se $K' \neq \emptyset$ então, por definição

$$\delta_i(K) = \sum_{j \in J} \max\{0, \min_{k \in K} c_{kj} - c_{ij}\} \quad \text{e} \quad \delta_i(K') = \sum_{j \in J} \max\{0, \min_{k \in K'} c_{kj} - c_{ij}\}$$

Temos, uma vez que $K' \subseteq K$, o seguinte fato

$$\min_{k \in K} c_{kj} \leq \min_{k \in K'} c_{kj}, \forall j \in J$$

Então,

$$\min_{k \in K} c_{kj} - c_{ij} \leq \min_{k \in K'} c_{kj} - c_{ij}, \forall j \in J$$

O que implica

$$\max\{0, \min_{k \in K} c_{kj} - c_{ij}\} \leq \max\{0, \min_{k \in K'} c_{kj} - c_{ij}\}, \forall j \in J$$

Assim,

$$\underbrace{\sum_{j \in J} \max\{0, \min_{k \in K} c_{kj} - c_{ij}\}}_{\delta_i(K)} \leq \underbrace{\sum_{j \in J} \max\{0, \min_{k \in K'} c_{kj} - c_{ij}\}}_{\delta_i(K')}$$

Logo, $\delta_i(K) \leq \delta_i(K'), \forall K' \subseteq K, \forall i \notin K$. □

Como conseqüência da propriedade 2, temos que:

$$\Delta_i(K) \geq \Delta_i(K'), \forall K' \subseteq K, \forall i \notin K \tag{2.2}$$

De fato, $\delta_i(K) \leq \delta_i(K') \rightarrow -\delta_i(K) \geq -\delta_i(K') \rightarrow f_i - \delta_i(K) \geq f_i - \delta_i(K') \rightarrow \Delta_i(K) \geq \Delta_i(K')$.

Antes de apresentarmos a definição dos testes de redução, faz-se necessário definir os conjuntos em que as facilidades serão inseridas de acordo com o seu status (fechadas, abertas ou indefinidas): seja K_0 o conjunto de facilidades fechadas ($K_0 = \{i \in I \mid y_i = 0\}$), K_1 o conjunto de facilidades abertas

($K_1 = \{i \in I \mid y_i = 1\}$) e K_2 o conjunto de facilidades cujos status estão indefinidos.

Estabelecemos então os seguintes testes de acordo com Akinc e Khumawala [1], Mateus e Bornstein [44] e Campêlo e Bornstein [49]:

O-test: Se $\Delta_i(K_1 \cup K_2 - i) \leq 0$ então $y_i = 1$

C-test: Se $\Delta_i(K_1) \geq 0$ então $y_i = 0$

Normalmente se inicia um processo iterativo formado por esses testes, considerando todos os status das facilidades indefinidos, ou seja, $K_0 = K_1 = \emptyset$ e $K_2 = I$. À medida em que os testes são aplicados, facilidades podem ter seus status definidos. Estas serão colocadas em K_0 ou K_1 e automaticamente retiradas de K_2.

Observa-se que a aplicação do *C-test* no início do processo não faz sentido, pois $\Delta_i(K_1) = -\infty$. Então, normalmente usa-se o *O-test* visando associar facilidades a K_1. Quando K_1 deixar de ser vazio, o *C-test* poderá atribuir facilidades a K_0.

A aplicação sucessiva dos testes de redução pode levar à determinação dos status de algumas facilidades na solução ótima. A otimalidade das decisões é assegurada pela supermodularidade de $w()$, que tem como conseqüência direta a desigualdade (2.2). O fechamento de facilidades diminui $K_1 \cup K_2$, pois $K_1 \cup K_2 = I - K_0$. Devido a (2.2) são gerados valores não-crescentes de $\Delta_i(K_1 \cup K_2 - i), \forall i \in K_2$. Por outro lado, a abertura de facilidades faz com que K_1 aumente e conseqüentemente sejam gerados valores não-decrescentes de $\Delta_i(K_1), \forall i \in K_2$.

Observamos na Figura 2.1 uma ilustração de como trabalham o *O-* e *C-*

testes. As setas indicam os sentidos em que $\Delta_i(K_1)$ e $\Delta_i(K_1 \cup K_2 - i)$ se movem à medida em que os testes são aplicados.

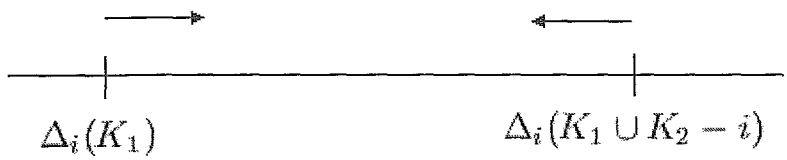

Figura 2.1: Deslocamento dos deltas de acordo com a aplicação dos testes

Se durante o processo iterativo de aplicação dos testes nós tivermos $K_2 = i$ então $\Delta_i(K_1) = \Delta_i(K_1 \cup K_2 - i)$. Neste caso, deve-se fazer ou $y_i = 1$ se $\Delta_i(K_1) < 0$ ou $y_i = 0$ se $\Delta_i(K_1) > 0$. Se $\Delta_i(K_1) = 0$, não faz diferença se fechamos ou abrimos a facilidade i. Assim, obteríamos $K_2 = \emptyset$ e os status de todas as facilidades seriam determinados de forma ótima. Porém, na prática, essa situação raramente acontece. Geralmente os testes conduzem a uma situação em que $\Delta_i(K_1) < 0 < \Delta_i(K_1 \cup K_2 - i), \forall i \in K_2$.

No sentido de amenizar as limitações dos testes de redução e evitar grandes esforços computacionais, desenvolvemos algumas heurísticas. É o que veremos a seguir.

2.2 Heurísticas ADD/DROP

Como vimos na seção 1.6, muitas heurísticas têm sido desenvolvidas para a solução de problemas de localização. De um modo geral, as heurísticas têm a limitação de não garantir a otimalidade das decisões, porém, são muito utilizadas na prática devido a simplicidade de implementação, flexibilidade e redução do esforço computacional, comparadas a procedimentos exatos com o mesmo propósito, principalmente quando se trata de solucionar problemas NP-difíceis, como é o caso do SPLP (propriedade 1 da seção 1.5).

Vimos também na seção 1.6 que heurísticas ADD/DROP são procedimentos que em uma dada iteração tentam selecionar/excluir as facilidades mais econômicas. São usualmente também chamados de procedimentos "gulosos". O emprego destes procedimentos tem sido bem sucedido na prática. Em nosso trabalho, desenvolvemos 6 heurísticas desse tipo.

Um problema comumente encontrado na utilização de procedimentos gulosos consiste na tendência que em geral eles apresentam de encontrar ótimos locais em detrimento dos globais. Visando tentar fugir dessa tendência, incorporamos às nossas heurísticas alguns parâmetros de forma que, conhecendo determinadas características dos problemas trabalhados, valores apropriados para esses parâmetros possam ser fixados e conseqüentemente melhores resultados possam ser obtidos. Na Tabela 2.1, vemos como nossas heurísticas serão denominadas bem como os parâmetros que lhes serão associados.

As duas primeiras heurísticas possuem forte conexão com o embasamento

Heurística		Parâmetro
Heurística de Soma de Deltas (HSD)		β
Heurística Para o Caso Especial $K_1 = \emptyset$ (HCE)		γ
Heurísticas de Reavaliação de Facilidades Abertas/Fechadas (HRF)	Heurística de Reavaliação de Facilidades Abertas (HRFA)	$\alpha_{fechar}^{inicial}$, ϵ_{fechar}
	Heurística de Reavaliação de Facilidades Fechadas (HRFF)	$\alpha_{abrir}^{inicial}$, ϵ_{abrir}
Heurísticas de Troca (HT)	Heurística T1	—
	Heurística T2	—

Tabela 2.1: Heurísticas desenvolvidas e parâmetros associados

teórico dos testes de redução pois foram desenvolvidas a partir das limitações destes. As 4 últimas heurísticas são procedimentos essencialmente práticos, cuja lógica e justificativa advém sobretudo da experiência prática e foram criadas visando melhorar os resultados obtidos pelas duas primeiras. Estes fatos fizeram com que nossas heurísticas tivessem sido concebidas muito fortemente vinculadas ao algoritmo como um todo e por isso serão apresentadas em detalhes no próximo capítulo.

Capítulo 3

Algoritmo

No capítulo anterior, vimos uma explanação sobre as bases teóricas do nosso algoritmo: testes de redução e heurísticas ADD/DROP. Neste capítulo, exibiremos em detalhes como essas heurísticas foram constituídas, combinadas entre si e combinadas aos testes de redução, dando origem ao nosso algoritmo.

Teremos, como entrada do nosso algoritmo, os elementos básicos para caracterizar um SPLP: um conjunto de facilidades (I), um conjunto de centros de demanda(J), o vetor de custos fixos de instalação das facilidades de I e a matriz de custos de transporte entre os elementos de I e J. Além disso, serão atribuídos valores aos parâmetros β, γ, $\alpha_{abrir}^{inicial}$, $\alpha_{fechar}^{inicial}$, ϵ_{abrir} e ϵ_{fechar}, que serão posteriormente apresentados na definição das heurísticas.

Fundamentalmente, no nosso algoritmo associamos a cada facilidade de I um status dentre 5 possíveis opções: indefinido, definitivamente aberta, definitivamente fechada, temporariamente aberta e temporariamente fechada. No início, consideramos todas as facilidades com status indefinido.

Então, a aplicação sistemática dos testes de redução e heurísticas ocasionará mudanças desses status. Todas as possíveis mudanças de status de facilidades de acordo com a aplicação de cada procedimento podem ser vistas na Figura 3.1. O algoritmo termina quando cada facilidade tiver como status: definitivamente aberta ou definitivamente fechada. Essa situação representa que, para a entrada considerada, essa é a melhor solução que nosso algoritmo pode obter.

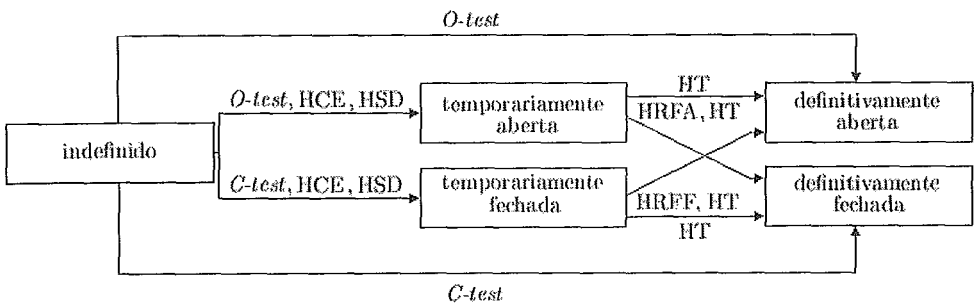

Figura 3.1: Possíveis trocas de status das facilidades

Em seguida, definiremos alguns conjuntos onde serão inseridas facilidades de I, de acordo com os status que elas apresentarem ao longo da aplicação do algoritmo. Mantendo a notação da seção 2.1, temos que: K_0 será o conjunto de facilidades fechadas (definitiva ou temporariamente), K_1 o conjunto de facilidades abertas (definitiva ou temporariamente) e K_2 o conjunto de facilidades cujos status estão indefinidos.

Além disso, definiremos os conjuntos: $\overline{K_0}$ como o conjunto de facilidades fechadas temporariamente e $\overline{K_1}$ como o conjunto de facilidades abertas temporariamente. Não definimos conjuntos específicos para facilidades definitivamente abertas e fechadas porque elas podem ser facilmente identificadas através de operações sobre os conjuntos que acabamos de definir. O conjunto de facilidades definitivamente abertas pode ser dado por $(K_1 - \overline{K_1})$ e

o conjunto de facilidades definitivamente fechadas por $(K_0 - \overline{K}_0)$.

O algoritmo inclui 3 fases: a fase 1 é a responsável pela determinação dos status de todas as facilidades e as fases 2 e 3 irão atuar iterativamente, questionando esse status, podendo modificá-lo. Essas fases são constituídas pela aplicação dos testes de redução e das heurísticas que desenvolvemos, da seguinte forma:

- *Fase 1:* O *O-test* e o *C-test* são aqui aplicados visando determinar de forma ótima os status de um conjunto de facilidades. Para as facilidades restantes, se houverem, há 3 procedimentos distintos que podem ser usados nessa fase: uma heurística denominada Heurística de Soma de Deltas (HSD), uma heurística desenvolvida para o caso especial em que nenhuma facilidade foi definitivamente aberta (HCE) e a reaplicação do *O-test* e *C-test*, após a execução de alguma dessas heurísticas, o que não garante mais a otimalidade das decisões.

- *Fase 2:* formada basicamente pelas Heurísticas de Reavaliação de Facilidades Abertas/Fechadas (HRF). Esta fase tem como objetivo refinar a solução obtida na fase 1, avaliando para facilidades temporariamente abertas ou fechadas, o impacto que a solução teria ao se efetuar uma mudança de status.

- *Fase 3:* constitui-se de um refinamento da solução obtida ao final da fase 2. Uma heurística de troca de status de pares de facilidades (HT) é utilizada para isso.

Ilustrativamente, podemos observar na Figura 3.2 como as fases se relacionam no nosso algoritmo e como os procedimentos se relacionam dentro de

cada fase.

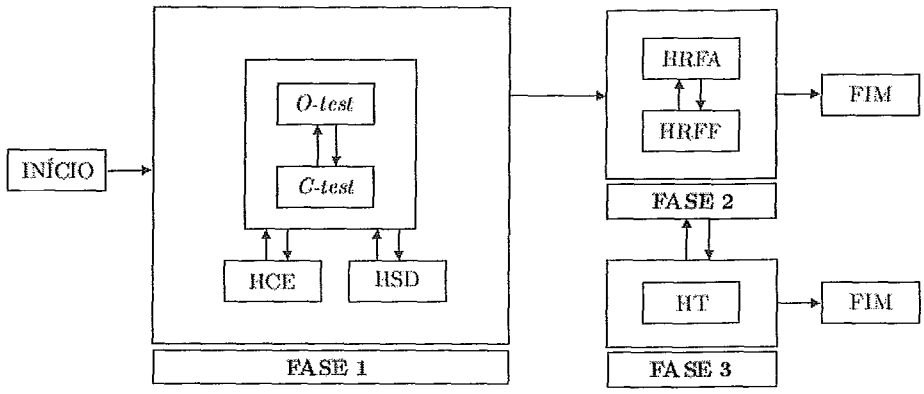

Figura 3.2: Comunicação entre as fases do algoritmo e entre os procedimentos em cada fase

A saída do algoritmo é fornecida pelos conjuntos K_0 e K_1 que, ao final da execução, conterão as facilidades que serão definitivamente fechadas e abertas (pois $\overline{K}_0 = \overline{K}_1 = \emptyset$).

Observa-se que, dados os conjuntos K_0 e K_1, temos de forma imediata a solução do SPLP. Basta lembrar que, de acordo com a propriedade 2 da seção 1.5, o SPLP pode ser decomposto em dois subproblemas: localização (cuja solução é dada pelos valores de $y_i, i \in I$) e alocação (cuja solução é dada pelos elementos $x_{ij}, i \in I, j \in J$). A solução do problema de localização está na constituição dos conjuntos K_0 e K_1 ($K_0 = \{i \in I \mid y_i = 0\}$ e $K_1 = \{i \in I \mid y_i = 1\}$). Na apresentação da propriedade 2 do SPLP mostra-se ainda como os elementos x_{ij} podem ser obtidos imediatamente a partir dos valores de y_i, daí teremos também a solução do problema de alocação.

Veremos a seguir detalhadamente cada fase do nosso algoritmo.

3.1 Fase 1

Na primeira fase serão determinados os status, definitivos ou temporários, de todas as facilidades. Isso será feito sobretudo através da aplicação do *O-test* e *C-test* e através da Heurística da Soma de Deltas. Há ainda nessa fase a aplicação de uma heurística projetada para o tratamento do caso em que nenhuma facilidade é definitivamente aberta através do *O-test*.

Como veremos adiante, todas as decisões heurísticas tomadas nessa fase serão questionadas na fase seguinte. Por isso, todas as facilidades abertas e fechadas através de heurísticas na fase 1, além de serem colocadas, respectivamente, em K_1 e K_0 também serão colocadas, respectivamente, em $\overline{K}_1$ e $\overline{K}_0$.

Iniciamos essa fase com $K_2 = I$ e todos os demais conjuntos vazios. Vimos no capítulo anterior que a aplicação sucessiva do *O-test* e *C-test* pode determinar de forma ótima os status de facilidades. Então aplicaremos de forma iterativa os testes de redução da seguinte forma: aplicamos o *O-test* tentando associar facilidades a K_1 e, caso alguma facilidade tenha sido aberta, pode-se ter potencializado o fechamento de facilidades. Aplica-se então o *C-test*, que poderá associar facilidades a K_0. Uma vez fechada uma facilidade, é possível que a abertura de facilidades tenha sido potencializada e então o *O-test* é novamente aplicado e assim sucessivamente.

É possível que o conjunto K_2 tenha se esvaziado e então podemos finalizar o algoritmo, garantindo a otimalidade da solução. No entanto, como vimos anteriormente, o que geralmente ocorre é que restam facilidades i em K_2 tais

que $\Delta_i(K_1) < 0 < \Delta_i(K_1 \cup K_2 - i)$. Uma vez que esses valores de $\Delta_i()$ foram calculados e não foram suficientes para a determinação dos status ótimos das facilidades i através dos testes de redução, pensamos em utilizá-los de forma aproximada, através de uma heurística. Fizemos isso através da Heurística de Soma de Deltas, que será descrita na próxima seção.

3.1.1 Heurística de Soma de Deltas (HSD)

A idéia principal da Heurística de Soma de Deltas é que o módulo da soma dos valores de $\Delta_i(K_1)$ e $\Delta_i(K_1 \cup K_2 - i)$ nos mostra o quão díspares são os valores de $\mid \Delta_i(K_1) \mid$ e $\mid \Delta_i(K_1 \cup K_2 - i) \mid$, para cada facilidade i. A facilidade que tem a maior disparidade entre esses valores é evidentemente aquela que tem maior $\mid \Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i) \mid$, dentre as que estão em K_2. Supondo que os conjuntos K_0 e K_1 crescem de forma balanceada, isto é, que as decisões com respeito às facilidades de K_2 implicam em dispô-las equilibradamente em K_0 e K_1, essa facilidade será a que terá seu status mais facilmente determinado dentre as que estão em K_2, bastando verificar o sinal de $\Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i)$. Se essa soma for negativa, significa que brevemente $\Delta_i(K_1 \cup K_2 - i)$ passaria a ser não-positivo e portanto i deveria ser aberta segundo o *O-test*. Por outro lado, se a soma for positiva, significa que $\Delta_i(K_1)$ está próximo de assumir um valor não-negativo, devendo então i ser fechada, de acordo com o *C-test*. Finalmente, se a soma for nula, então a heurística não tem, em princípio, informação relevante que a leve a optar por tomar a decisão de abrir ou fechar essa facilidade. Decidimos arbitrariamente abri-la nesse caso.

Ilustrativamente, temos nas figuras 3.3 e 3.4 os casos em que a Heurística de Soma de Deltas abre ou fecha uma facilidade $i \in K_2$.

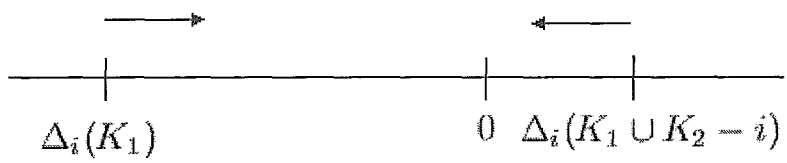

Figura 3.3: Heurística de Soma de Deltas - Caso em que i será aberta

Figura 3.4: Heurística de Soma de Deltas - Caso em que i será fechada

A Heurística de Soma de Deltas se baseia no cálculo de $\Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i)$, supondo um crescimento balanceado de K_0 e K_1, o que nem sempre ocorre. Na prática, na solução ótima de diversos problemas o número de facilidades abertas é bem inferior ao de facilidades fechadas ou vice-versa. Nesses casos, a Heurística de Soma de Deltas, da forma como foi definida, pode ser levada a não produzir bons resultados. Pensamos nestes casos em adicionarmos um parâmetro $\beta \geq 0$ para ponderar a soma dos deltas visando tornar a heurística um pouco mais "esperta".

A idéia desse refinamento é que, quando a heurística original enfrenta uma situação em que os valores de $\mid \Delta_i(K_1) \mid$ e $\mid \Delta_i(K_1 \cup K_2 - i) \mid$ são muito próximos, ela toma a decisão de abrir ou fechar a facilidade i baseada exclusivamente em uma pequena superioridade de um em relação a outro, o que não reflete tão claramente a tendência de abrir ou fechar i. Entretanto, sabendo *a priori* que o problema é, por exemplo, do tipo em que o número

de facilidades abertas é bem superior ao de facilidades fechadas, poderíamos privilegiar a decisão de abri-la. De forma análoga, se o problema é do tipo que o número de facilidades fechadas é bem superior ao de facilidades abertas, então a decisão de fechar essa facilidade é provavelmente mais acertada.

Para realizarmos essa modificação na Heurística de Soma de Deltas, basta multiplicar uma das parcelas, $\Delta_i(K_1)$ ou $\Delta_i(K_1 \cup K_2 - i)$ por β, atribuindo a β um valor conveniente de acordo com o tipo de problema. Escolhemos arbitrariamente a parcela $\Delta_i(K_1)$ para ser multiplicada por β. Consideraremos então a soma $\beta \cdot \Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i)$ para identificarmos na heurística a maior tendência de abrir ou fechar a facilidade i.

Formalmente, podemos escrever essa heurística assim:

Heurística de Soma de Deltas

Início

$q = \arg\max_{i \in K_2} \mid \beta \cdot \Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i) \mid$

Se $\beta \cdot \Delta_q(K_1) + \Delta_q(K_1 \cup K_2 - q) > 0$

Então $K_0 = K_0 \cup q$, $\overline{K}_0 = \overline{K}_0 \cup q$ e $K_2 = K_2 - q$

Senão $K_1 = K_1 \cup q$, $\overline{K}_1 = \overline{K}_1 \cup q$ e $K_2 = K_2 - q$

Fim

Em problemas onde o número de facilidades abertas é bem superior ao de facilidades fechadas, à medida em que as facilidades vão tendo seus status determinados, o valor de $\Delta_i(K_1 \cup K_2 - i)$ será menos modificado do que $\Delta_i(K_1)$, $\forall i \in K_2$. Portanto, o valor de $\Delta_i(K_1)$ será decisivo na determinação dos status das facilidades, cabendo ponderar o seu valor na soma,

de forma a favorecer a operação de abertura de facilidades. Basta escolher um $\beta \geq 1$. Quanto maior o valor de β, maiores serão as chances de que a soma $\beta \cdot \Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i)$ seja negativa e a facilidade i seja aberta.

Por outro lado, em problemas onde o número de facilidades fechadas é bem superior ao de facilidades abertas, é o valor de $\Delta_i(K_1)$ que permanece fixo a maior parte do tempo enquanto que $\Delta_i(K_1 \cup K_2 - i)$ varia bastante, à medida em que as facilidades vão sendo colocadas em K_0. Cabe portanto atribuir um peso pequeno a $\Delta_i(K_1)$ de modo a favorecer a operação de fechamento de facilidades. Para isso, escolhemos um $0 \leq \beta < 1$. Quanto menor o valor de β, mais provável será que a soma $\beta \cdot \Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i)$ seja positiva e a facilidade i seja fechada.

É interessante observar que para valores muito grandes de β, $max_{i \in K_2} \mid \beta \cdot \Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i) \mid$ será o equivalente a $max_{i \in K_2} \mid \beta \cdot \Delta_i(K_1) \mid$, que equivale a $max_{i \in K_2} - \beta \cdot \Delta_i(K_1)$, pois $\Delta_i(K_1) < 0$. Este resultado é equivalente a tomar $min_{i \in K_2} \Delta_i(K_1)$. Como $min_{i \in K_2} \Delta_i(K_1) < 0$, a facilidade i será aberta.

Entretanto, quando β assumir valores muito pequenos, $max_{i \in K_2} \mid \beta \cdot \Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i) \mid$ será correspondente a $max_{i \in K_2} \mid \Delta_i(K_1 \cup K_2 - i) \mid$, que será igual a $max_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$, pois $\Delta_i(K_1 \cup K_2 - i) > 0$. Como $max_{i \in K_2} \Delta_i(K_1 \cup K_2 - i) > 0$, então a facilidade i será fechada.

Esses critérios de abertura/fechamento de facilidades com valores extremos de β coincidem exatamente com os critérios de abertura e fechamento de facilidades de heurísticas desenvolvidas em Jacobsen [29] e Campêlo e Bornstein [49].

3.1.2 Heurística Para o Caso Especial $K_1 = \emptyset$ (HCE)

Há um caso em que nem os testes de redução, nem a Heurística de Soma de Deltas serão úteis na atribuição de status às facilidades. É o caso em que nenhuma facilidade satisfaz ao *O-test*. Neste caso, como vimos no capítulo anterior, o *C-test* também não seria útil pois não seria suficiente para fechar qualquer facilidade, uma vez que o valor de $\Delta_i(\emptyset) = -\infty, \forall i \in K_2$. A Heurística de Soma de Deltas também não produziria um resultado desejável nesse caso, pois: $\Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i) = \Delta_i(\emptyset) + \Delta_i(I - i)$. Como $\Delta_i(\emptyset)$ é $-\infty$ e $\Delta_i(I-i)$ possui um valor finito, o valor da soma $\Delta_i(K_1)+\Delta_i(K_1 \cup K_2 - i)$ será $-\infty$. Ora, como o valor dessa soma é negativo e o mesmo para todas as facilidades $i \in K_2$, teríamos, pela Heurística de Soma de Deltas, que seria indiferente a escolha da facilidade de K_2 a ser aberta. Essa decisão no entanto não faz sentido uma vez que essa heurística foi projetada tendo em vista a utilização de valores finitos para $\Delta_i()$. Então, desenvolvemos uma heurística bastante simples especificamente para esse caso e a denominamos Heurística Para o Caso Especial $K_1 = \emptyset$.

Essa heurística consiste em escolher para ser aberta a facilidade p tal que $\Delta_p(K_1 \cup K_2 - p) = min_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$. Ela se baseia na suposição de que a facilidade com menor $\Delta_i(K_1 \cup K_2 - i)$ deverá ser aberta, pois à medida em que facilidades são retiradas de $K_1 \cup K_2$, o valor de $\Delta_p(K_1 \cup K_2 - p)$ diminui, ficando mais próximo de atingir um valor menor ou igual a zero, satisfazendo então ao *O-test*.

Com o objetivo de tornar a heurística um pouco mais "esperta",

podemos fazer o seguinte refinamento: estabelecer um valor máximo de $\Delta_i(K_1 \cup K_2 - i)$ como satisfatório para que uma facilidade i seja aberta. Esse limite será definido por um parâmetro $\gamma \geq 0$. Enquanto esse limite não for atingido, fechamos facilidades que são pouco prováveis de serem abertas.

Mais precisamente, o refinamento acontece da seguinte forma: no início, escolhemos a facilidade p mais provável de ser aberta (aquela facilidade que produz $\min_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$). Se $\Delta_p(K_1 \cup K_2 - p) \leq \gamma$, então p é aberta e finalizamos a heurística. Se $\Delta_p(K_1 \cup K_2 - p) > \gamma$, determinamos a facilidade t mais improvável de ser aberta (considerando o raciocínio feito na concepção inicial desta heurística, será aquela facilidade que produz $\max_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$) e a fechamos. O conjunto $K_1 \cup K_2$ diminui, o que pode acarretar a diminuição dos valores de $\Delta_i(K_1 \cup K_2 - i), i \in K_2$. Então recalcula-se a facilidade p mais provável de ser aberta. E testa-se novamente se $\Delta_p(K_1 \cup K_2 - p) \leq \gamma$. O procedimento se repete até que tenhamos uma facilidade p tal que $\Delta_p(K_1 \cup K_2 - p) \leq \gamma$. Quando isso acontecer, então p será aberta e o procedimento é finalizado. Em forma de pseudo-código, o refinamento pode ser assim apresentado:

Heurística Para o Caso Especial $K_1 = \emptyset$

Início

$p = \arg\min_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$

Enquanto $\Delta_p(K_1 \cup K_2 - p) > \gamma$ faça

Início

$t = \arg\max_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$

$K_0 = K_0 \cup t$, $\overline{K}_0 = \overline{K}_0 \cup t$, $K_2 = K_2 - t$

$p = \arg\min_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$

Fim

$K_1 = K_1 \cup p$, $\overline{K}_1 = \overline{K}_1 \cup p$, $K_2 = K_2 - p$

Fim

É fácil observar que esse procedimento converge. À medida que facilidades vão sendo fechadas, $\Delta_i(K_1 \cup K_2 - i)$ tende a ser inferior ao valor de γ. Pois, independente do valor de γ, se chegarmos à situação em que $K_2 = p$, então $\Delta_p(K_1 \cup K_2 - p) = f_p - w(K_1 \cup K_2 - p) + w(K_1 \cup K_2) = f_p - w(\emptyset) + w(p) = f_p - \infty + \sum_{j \in J} c_{pj} = -\infty$. Portanto, a condição $\Delta_p(K_1 \cup K_2 - p) \leq \gamma$ seria satisfeita, uma facilidade seria aberta e o procedimento seria então finalizado.

Observamos que esse critério de identificação e fechamento de facilidades pouco prováveis de serem abertas é o mesmo utilizado no fechamento de facilidades por nossa Heurística de Soma de Deltas para valores muito pequenos de β e de heurísticas de Jacobsen [29] e Campêlo e Bornstein [49].

Constatou-se experimentalmente que os testes de redução funcionaram bem, mesmo após alguma heurística ter sido executada, ou seja, quando não é mais garantida a otimalidade das decisões desses testes. Então, tomamos proveito também desse fato na composição da fase 1 do algoritmo. Apresentaremos esta fase a seguir em forma de pseudo-código e em seguida comentaremos passo a passo.

3.1.3 Pseudo-código da Fase 1

Eis o pseudo-código da fase 1 do algoritmo:

FASE 1

Passo 0: (* Inicialização do algoritmo *)

$K_2 = I$

$K_0 = K_1 = \overline{K}_0 = \overline{K}_1 = \emptyset$

Passo 1: (* Abrir facilidades - Aplicação do *O-test* *)

$A = \{i \in K_2 \mid \Delta_i(K_1 \cup K_2 - i) \leq 0\}$

$K_1 = K_1 \cup A$, $K_2 = K_2 - A$

Se $\overline{K}_0 \cup \overline{K}_1 \neq \emptyset$ então $\overline{K}_1 = \overline{K}_1 \cup A$

Passo 2: (* Condição de término da fase 1 - $K_2 = \emptyset$ *)

Se $K_2 = \emptyset$

Então vá para a **fase 2**

Passo 3: (* Fechar facilidades - Aplicação do *C-test* *)

$F = \{i \in K_2 \mid \Delta_i(K_1) \geq 0\}$

$K_0 = K_0 \cup F$, $K_2 = K_2 - F$

Se $\overline{K}_0 \cup \overline{K}_1 \neq \emptyset$ então $\overline{K}_0 = \overline{K}_0 \cup F$

Se $F \neq \emptyset$ então volte para o passo 1

Passo 4: (* Heurísticas *)

Se $K_1 = \emptyset$

Então (* Heurística Para o Caso Especial $K_1 = \emptyset$ *)

$p = \arg\min_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$

Enquanto $\Delta_p(K_1 \cup K_2 - p) > \gamma$ faça

Início

$t = \arg\max_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$

$K_0 = K_0 \cup t$, $\overline{K}_0 = \overline{K}_0 \cup t$, $K_2 = K_2 - t$

$p = \arg\min_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$

Fim

$K_1 = K_1 \cup p$, $\overline{K}_1 = \overline{K}_1 \cup p$, $K_2 = K_2 - p$

Volte para o passo 1

Senão (* Heurística de Soma de Deltas *)

$q = \arg\max_{i \in K_2} \mid \beta \cdot \Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i) \mid$

Se $\beta \cdot \Delta_q(K_1) + \Delta_q(K_1 \cup K_2 - q) > 0$

Então $K_0 = K_0 \cup q$, $\overline{K}_0 = \overline{K}_0 \cup q$, $K_2 = K_2 - q$

Volte para o passo 1

Senão $K_1 = K_1 \cup q$, $\overline{K}_1 = \overline{K}_1 \cup q$, $K_2 = K_2 - q$

Volte para o passo 2

O passo 0 diz respeito a inicialização dos conjuntos K_0, K_1, K_2, $\overline{K}_0$ e $\overline{K}_1$. Inicialmente, todas as facilidades têm seus status indefinidos, por isso todas são armazenadas em K_2 e os demais conjuntos são vazios.

No passo 1, o *O-test* é aplicado, podendo abrir facilidades. Se nenhuma heurística tiver sido executada até então, as facilidades que forem abertas aqui asseguradamente pertencem ao conjunto de facilidades abertas na solução ótima. Porém, se alguma heurística tiver sido executada antes desse passo, então o teste possibilitará apenas a abertura temporária de facilidades.

Temos, no passo 2, a condição de término da fase 1 do algoritmo. Ela

termina quando $K_2 = \emptyset$, ou seja, quando não existirem mais facilidades com status indefinido.

O *C-test* é aplicado no passo 3, possibilitando o fechamento de facilidades. Similarmente ao passo 1, as facilidades fechadas nesse passo, antes que se tenha executado qualquer heurística, também pertencem ao conjunto de facilidades fechadas na solução ótima. Se alguma heurística tiver sido executada, então o teste fechará facilidades temporariamente.

Caso alguma facilidade tenha sido fechada no passo 3, é possível que se tenha potencializado a abertura de novas facilidades pois $K_1 \cup K_2$ foi reduzido. Então, retorna-se para o passo 1.

Este é o primeiro *loop* do algoritmo: passos 1 ao 3. Ele é o responsável por aplicar iterativamente os testes de redução.

Depois da aplicação do *O-test* e *C-test*, muitas facilidades ou até mesmo todas as facilidades podem estar ainda com seus status indefinidos. No intuito de mudar esse contexto, é aplicada a Heurística de Soma de Deltas, no passo 4. Mas, como vimos anteriormente, ela não deve ser aplicada se K_1 for vazio. Então, somente nessa situação, utilizamos a Heurística Para o Caso Especial $K_1 = \emptyset$. Quando essa heurística é acionada, uma facilidade necessariamente é aberta. Dependendo do parâmetro γ, de 0 a $\mid K_2 \mid -1$ facilidades podem ser fechadas. Após a utilização dessa heurística voltamos aos passos 1 a 3, tentando abrir/fechar novas facilidades. Como essa heurística pode ter fechado alguma facilidade, o conjunto $K_1 \cup K_2$ pode ter sido alterado e portanto volta-se ao passo 1, tentando aplicar o *O-test*.

Constitui-se o segundo *loop* do algoritmo: passos 1 ao 4. Ele só será

executado no máximo uma vez e tem a função de, quando não for possível abrir facilidades na primeira execução do *O-test*, abrir inicialmente uma facilidade através de uma heurística e retornar ao primeiro *loop*.

Se, no passo 4, $K_1 \neq \emptyset$, a Heurística de Soma de Deltas será aplicada e necessariamente uma facilidade será aberta ou fechada, podendo ser influenciada pelo parâmetro β. Volta-se novamente para o *O-test* e *C-test*. Como a Heurística de Soma de Deltas pode ter fechado uma facilidade, pode ter alterado $K_1 \cup K_2$. Neste caso, faz sentido voltar a tentar aplicar o *O-test*. Voltamos portanto ao passo 1. No caso de uma facilidade ter sido aberta pela Heurística de Soma de Deltas, $K_1 \cup K_2$ não se altera e não faz sentido aplicar o *O-test*. O algoritmo vai então para o passo 2 pois a facilidade fechada pela heurística pode ter sido a última de K_2.

Temos o terceiro e o quarto *loops* do algoritmo: passos 1 ao 4 e passos 2 ao 4. Eles são responsáveis por, respectivamente, fechar e abrir uma facilidade através da Heurística de Soma de Deltas e, em seguida, retornar para o primeiro *loop*.

Observamos que a partir da primeira execução do passo 4 do algoritmo, independente da heurística utilizada neste passo, não se pode mais garantir a otimalidade das decisões de abrir/fechar facilidades.

Pode-se verificar facilmente a convergência da fase 1. A menos que a aplicação do primeiro *loop* faça $K_2 = \emptyset$, caso em que a convergência fica garantida, qualquer uma dos outros loops incluem o passo 4, onde certamente o status de ao menos uma facilidade será fixado, retirando-se ao menos uma facilidade de K_2. Assim, acaba-se por obter $K_2 = \emptyset$.

3.2 Fases 2 e 3

Ao final da fase 1 temos $K_2 = \emptyset$, o que significa que todos os status das facilidades já foram determinados ao menos temporariamente. Neste ponto, já temos uma solução para o problema. Mas as decisões provenientes da Heurística de Soma de Deltas, da Heurística Para o Caso Especial $K_1 = \emptyset$ e dos testes de redução após a aplicação de alguma heurística, não são assegu-radamente ótimas e pode ser possível melhorar a solução.

As fases 2 e 3 irão atuar de forma iterativa, visando melhorar a solução. Os conjuntos $\overline{K}_0$ e $\overline{K}_1$ serão essenciais nessas fases. A fase 2 será constituída basicamente por duas heurísticas que tentarão mudar os status das facilidades de $\overline{K}_0$ e $\overline{K}_1$ e a fase 3 tentará através de uma heurística realizar uma mudança simultânea dos status de uma facilidade de $\overline{K}_0$ e outra de $\overline{K}_1$. É característico das fases 2 e 3 a retirada de facilidades dos conjuntos $\overline{K}_0$ e $\overline{K}_1$, sem que novas facilidades sejam adicionadas a estes conjuntos. Assim, nessas fases, estamos mudando os status das facilidades de temporários para definitivos.

A seguir, apresentaremos as heurísticas que estão associadas às fases 2 e 3, bem como os pseudo-códigos referentes a estas fases.

3.2.1 Heurísticas de Reavaliação de Facilidades Abertas/Fechadas (HRF)

As Heurísticas de Reavaliação de Facilidades Abertas/Fechadas têm como objetivo questionar os status das facilidades temporariamente abertas e fechadas na fase 1, a começar pelas "mais prováveis" de serem abertas/fechadas. Basicamente, verificamos se a troca do status de uma determinada facilidade resulta em ganho para a função objetivo. No caso positivo, a troca é feita e uma melhora é obtida. A seguir, veremos em cada heurística como é feita a escolha das facilidades mais prováveis de serem abertas/fechadas e de que forma é avaliada a possibilidade de melhora na função objetivo.

Na Heurística de Reavaliação de Facilidades Fechadas (HRFF), classificamos como a facilidade mais provável de ser aberta, aquela que possui o menor valor de $\Delta_i(K_1), \forall i \in \overline{K}_0$. Seja r a facilidade mais provável de ser aberta, então o teste de avaliação de ganho no valor da função objetivo será o seguinte: Se $\Delta_r(K_1) < 0$ então fazemos $K_0 = K_0 - r$, $K_1 = K_1 \cup r$, $\overline{K}_0 = \overline{K}_0 - r$.

A idéia dessa heurística é a de que se a inserção de uma facilidade no conjunto K_1 resultar em uma redução no valor da função objetivo, então ela será efetuada.

Na Heurística de Reavaliação de Facilidades Abertas (HRFA) classificaremos como a facilidade mais provável de ser fechada aquela facilidade s tal que $\Delta_s(K_1 - s) = max_{i \in \overline{K}_1} \Delta_i(K_1 - i)$. O teste de viabilidade da troca do status

de s será definido por: Se $\Delta_s(K_1 - s) > 0$ então $K_1 = K_1 - s$, $K_0 = K_0 \cup s$, $\overline{K_1} = \overline{K_1} - s$.

Analogamente à heurística anterior, o que está por trás dessa é que se a retirada de uma facilidade do conjunto K_1 resultar em ganho na função objetivo, então ela será feita.

De forma semelhante ao que fizemos com os testes de redução, essas heurísticas também podem ser combinadas, levando em conta que a abertura de uma facilidade pode potencializar o fechamento de outra e vice-versa.

Ao invés de compararmos diretamente os valores de $\Delta_i(K_1 - i)$ e $\Delta_i(K_1)$ a zero e efetuarmos as modificações de status daí resultantes, podemos também fazer essas modificações gradativamente através da utilização de parâmetros.

Seja $\alpha_{fechar}, \alpha_{abrir} \geq 0$. Visando fechar uma facilidade $s \in \overline{K_1}$ verificaremos se $\Delta_s(K_1 - s) \geq \alpha_{fechar} \cdot f_s$. Isso significa dizer que estamos verificando se $f_s - \delta_s(K_1 - s) \geq \alpha_{fechar} \cdot f_s \rightarrow -\delta_s(K_1 - s) \geq \alpha_{fechar} \cdot f_s - f_s \rightarrow \delta_s(K_1 - s) \leq (1 - \alpha_{fechar}) \cdot f_s$. Quanto maior o valor de α_{fechar}, mais rigoroso será o critério de fechamento de facilidades. Como $\delta_s(K_1 - s) \geq 0$, devido a (2.1), evidentemente devemos fazer $\alpha_{fechar} \leq 1$.

Analogamente, objetivando abrir uma facilidade $r \in \overline{K_0}$, verificaremos se $\Delta_r(K_1) \leq -\alpha_{abrir} \cdot f_r$. Isto implica em verificarmos se $f_r - \delta_r(K_1) \leq -\alpha_{abrir} \cdot f_r \rightarrow -\delta_r(K_1) \leq -\alpha_{abrir} \cdot f_r - f_s \rightarrow \delta_r(K_1) \geq (1 + \alpha_{abrir}) \cdot f_r$. Novamente, quanto maior o valor de α_{abrir}, mais rigoroso será o critério de abertura de facilidades.

Começamos o procedimento atribuindo valores iniciais para α_{abrir} e

α_{fechar} através, respectivamente, dos parâmetros $\alpha_{abrir}^{inicial}$ e $\alpha_{fechar}^{inicial}$. À medida em que forem sendo abertas e/ou fechadas facilidades, decresceremos α_{abrir} e α_{fechar} de quantidades respectivamente iguais a ϵ_{abrir} e ϵ_{fechar}, com o intuito de abrir e/ou fechar as facilidades restantes. Procedemos assim sucessivamente até que esse valores sejam nulos. Atingida essa situação, estaremos aplicando o teste conforme as definições originais das heurísticas apresentadas no início desta seção.

Observamos que os parâmetros $\alpha_{abrir}^{inicial}$ e $\alpha_{fechar}^{inicial}$ são os responsáveis por estabelecer o rigor inicial com que serão avaliadas, respectivamente, as facilidades abertas e fechadas. Já os parâmetros ϵ_{abrir} e ϵ_{fechar} quantificam o decréscimo de rigor com que serão avaliadas, respectivamente, as facilidades abertas e fechadas a cada iteração da heurística.

Vejamos a seguir, como essas heurísticas foram combinadas para constituir a fase 2.

3.2.2 Pseudo-código da Fase 2

Antes de apresentarmos o pseudo-código da fase 2, temos que definir um conjunto que será utilizado como auxiliar nessa fase. Trata-se do conjunto K_0' que será definido como o conjunto de facilidades temporariamente fechadas no início da execução da fase 2. Exibiremos então a fase 2 do algoritmo:

FASE 2

Passo 5: (* Inicialização *)

$\alpha_{abrir} = \alpha_{abrir}^{inicial}$, $\alpha_{fechar} = \alpha_{fechar}^{inicial}$, $K_0' = \overline{K}_0$

Passo 6: (* Condição de término do algoritmo - $\overline{K}_0 \cup \overline{K}_1 = \emptyset$ *)

Se $\overline{K}_0 \cup \overline{K}_1 = \emptyset$ então **PARE**

Passo 7: (* Identificação da facilidade s mais provável de ser fechada *)

Se $\overline{K}_1 = \emptyset$ então vá para o passo 9

$s = \arg\max_{i \in \overline{K}_1} \Delta_i(K_1 - i)$

Passo 8: (* Fechar facilidades abertas *)

Se $\Delta_s(K_1 - s) > \alpha_{fechar} \cdot f_s$

Então $K_1 = K_1 - s$, $K_0 = K_0 \cup s$, $\overline{K}_1 = \overline{K}_1 - s$

Volte para o passo 6

Passo 9: (* Identificação da facilidade r mais provável de ser aberta *)

Se $\overline{K}_0 = \emptyset$ então vá para o passo 11

$r = \arg\min_{i \in \overline{K}_0} \Delta_i(K_1)$

Passo 10: (* Abrir facilidades fechadas *)

Se $\Delta_r(K_1) < -\alpha_{abrir} \cdot f_r$

Então $K_0 = K_0 - r$, $K_1 = K_1 \cup r$, $\overline{K}_0 = \overline{K}_0 - r$

Volte para o passo 9

Passo 11: (* Alguma facilidade foi aberta? *)

Se $\mid K_0' \mid > \mid \overline{K}_0 \mid$ então $K_0' = \overline{K}_0$ e volte para o passo 6

Passo 12: (* Diminuição dos valores de α_{abrir} e α_{fechar} *)

Se $\alpha_{abrir} = \alpha_{fechar} = 0$ então vá para a **fase 3**

$\alpha_{abrir} = \max(\alpha_{abrir} - \epsilon_{abrir}, 0)$

$\alpha_{fechar} = \max(\alpha_{fechar} - \epsilon_{fechar}, 0)$

Volte para o passo 7

No passo 5 serão inicializadas as variáveis α_{abrir}, α_{fechar}, respectivamente, com os valores dos parâmetros $\alpha_{abrir}^{inicial}$ e $\alpha_{fechar}^{inicial}$. Estas variáveis serão usadas mais adiante, respectivamente, na Heurística de Reavaliação de Facilidades Fechadas e na Heurística de Reavaliação de Facilidades Abertas. Também nesse passo será inicializado o conjunto K_0', que só será usado novamente no passo 11 pra verificar se alguma facilidade foi aberta na iteração corrente.

Temos, no passo 6, a condição de término do algoritmo na fase 2. Em duas situações o final do algoritmo poderá se dar nesse passo: a primeira é a situação em que ao final da fase 1, $\overline{K}_0 \cup \overline{K}_1 = \emptyset$, significando que todos os status das facilidades de I foram determinados de forma exata pelos testes de redução e os refinamentos das fases 2 e 3 serão desnecessários. A segunda é a situação em que todas as facilidades inicialmente colocadas em $\overline{K}_0$ e $\overline{K}_1$ já tiveram seus status modificados pelas heurísticas das fases 2 e 3; nesse ponto, mais nenhuma melhora na solução corrente poderá ser feita pelo nosso algoritmo e ele é então finalizado.

Determinamos, no passo 7, dentre as facilidades de $\overline{K}_1$, a facilidade s que uma vez fechada provoque a maior redução na função objetivo. Obviamente, $\overline{K}_1$ deverá ser diferente de vazio para que essa escolha faça sentido. Se não for, o algoritmo vai para a Heurística de Reavaliação de Facilidades Fechadas.

A Heurística de Reavaliação de Facilidades Abertas é então aplicada no passo 8. Quando ocorre a troca de status, o algoritmo irá permitir com que se tente novamente determinar dentre as facilidades restantes em $\overline{K}_1$, a próxima facilidade mais provável de ser fechada e o passo 8 será novamente executado. Mas antes, retorna-se para o passo 6 para verificar se a facilidade que acabou de ser aberta não foi a última de $\overline{K}_0 \cup \overline{K}_1$, o que resultaria na finalização do algoritmo.

Temos o primeiro *loop* da fase 2: passos 6 ao 8. Ele tenta fechar facilidades que foram abertas na fase 1, selecionando primeiramente as que causam maior ganho imediato na função objetivo.

O passo 9 é similar ao 7: determinaremos, dentre as facilidades de $\overline{K}_0$, a facilidade r que provoque a maior redução no valor da função objetivo, caso seja aberta. Enquanto $\overline{K}_0 \neq \emptyset$ selecionaremos uma facilidade nesse passo. Se $\overline{K}_0$ for vazio, então o algoritmo vai para o primeiro passo após a Heurística de Reavaliação de Facilidades Fechadas (passo 11).

Analogamente ao passo 8, a Heurística de Reavaliação de Facilidades Fechadas é aplicada no passo 10. Nesse passo, se houver a troca de status, volta-se para o passo 9, visando determinar dentre as facilidades restantes em $\overline{K}_0$, a mais provável de ser aberta.

Este é o segundo *loop* da fase 2: passos 9 ao 10. Nesse *loop* tenta-se abrir facilidades fechadas na fase 1, começando pelas que proverão as maiores reduções no valor da função objetivo.

O passo 11 é o responsável pela repetição dos passos 6 a 10. Se alguma facilidade tiver sido aberta no passo 10, é possível que se tenha potencializado

o fechamento de outras facilidades, uma vez que K_1 terá sido aumentado. O teste para verificar se alguma facilidade foi aberta é feito comparando o conjunto $\overline{K}_0$ no início (armazenado em K'_0) e no final da iteração. Se sua cardinalidade tiver diminuido significa que alguma facilidade foi aberta pois essa é a única situação na fase 2 em que se retira elementos de $\overline{K}_0$. Neste caso, volta-se para o passo 6 para verificar se ainda existem facilidades que possam ter seus status modificados.

Temos o terceiro *loop* dessa fase: passos 6 a 11. Ele é responsável por aplicar iterativamente as Heurísticas de Reavaliação de Facilidades Abertas/Fechadas, sem que sejam modificados os parâmetros α_{abrir} e α_{fechar}.

Finalmente, no passo 12, as variáveis α_{abrir} e α_{fechar} têm seus valores decrementados de, respectivamente, ϵ_{abrir} e ϵ_{fechar}. O algoritmo vai então para o passo 7, aplicando novamente as heurísticas agora potencializadas pela diminuição dos parâmetros α_{abrir} e α_{fechar}. Vale ressaltar que no passo 12, $\overline{K}_0 \cup \overline{K}_1 \neq \emptyset$, por isso não se faz necessário retornar ao passo 6.

O decréscimo dos valores de α_{abrir} e α_{fechar} será feito em cada execução do passo 12 até que eles atinjam o valor 0. Quando essas variáveis são iguais a zero, será feita a última tentativa da fase 2 de modificar status de facilidades. Se o algoritmo voltar para o passo 12 e as variáveis α_{abrir} e α_{fechar} forem iguais a 0, significa dizer que as heurísticas da fase 2 não mais poderão ser úteis, passando então para a fase 3.

O quarto e último *loop* da fase 2 é constituído pelos passos 7 a 12. Esse *loop* é o responsável pelo decréscimo das variáveis α_{abrir} e α_{fechar} e por retornar para as Heurísticas de Reavaliação de Facilidades Abertas/Fechadas.

Para mostrar a convergência da fase 2, mostremos que nenhum *loop* dessa fase é percorrido infinitamente.

- *1.º loop:* cada vez que é executado uma facilidade é descartada de $\overline{K}_1$. Pode ser percorrido no máximo $\mid \overline{K}_1 \mid$ vezes.
- *2.º loop:* cada vez que é executado uma facilidade é descartada de $\overline{K}_0$. Pode ser percorrido no máximo $\mid \overline{K}_0 \mid$ vezes.
- *3.º loop:* só é executado quando $\mid K'_0 \mid > \mid \overline{K}_0 \mid$, ou seja, quando pelo menos uma facilidade tiver sido retirada de $\overline{K}_0$. Pode ser percorrido no máximo $\mid \overline{K}_0 \mid$ vezes.
- *4.º loop:* cada vez que é executado implica no decréscimo de α_{abrir} e α_{fechar}. Pode ser percorrido até $\max(\alpha_{abrir}/\epsilon_{abrir}, \alpha_{fechar}/\epsilon_{fechar}) + 1$ vezes.

3.2.3 Heurísticas de Troca (HT)

Visando melhorar a solução obtida ao final da fase 2, decidimos incorporar uma Heurística de Troca ao algoritmo. Trata-se de um procedimento baseado em uma rotina de troca de status entre um par de facilidades, que segue o esquema:

Início

Seja $r \in \overline{K}_0$ e $s \in \overline{K}_1$

Se as mudanças simultâneas dos status das facilidades r e s

resultarem em um ganho na função objetivo

Então elas serão efetuadas.

Fim

Seja $E(K_1, i, j)$, $i \in \overline{K}_0$, $j \in \overline{K}_1$, a função que, para um dado K_1, fornece a economia nos custos fixos e variáveis que se teria ao abrir a facilidade i e fechar a facilidade j, simultaneamente. Formalmente, $E(K_1, i, j) = (f_j - f_i) + (w(K_1) - w(K_1 \cup i - j))$. Dessa forma, se $E(K_1, i, j)$ for positivo, haverá uma melhora no valor da função objetivo caso a troca de status entre i e j seja realizada e se for negativo, então haverá um acréscimo no valor da função objetivo uma vez realizada a troca. $E(K_1, i, j) = 0$ significa, evidentemente, que a troca entre de status entre i e j não altera o valor da função objetivo.

$E(K_1, i, j)$ pode ser escrita em função de $\Delta()$. Uma possível forma seria:

$$
\begin{aligned}
E(K_1, i, j) &= (f_j - f_i) + (w(K_1) - w(K_1 \cup i - j)) \\
&= f_j - f_i + w(K_1) - w(K_1 \cup i - j) + w(K_1 \cup i) - w(K_1 \cup i) \\
&= f_j - w(K_1 \cup i - j) + w(K_1 \cup i) - f_i + w(K_1) - w(K_1 \cup i) \\
&= \{f_j - [w(K_1 \cup i - j) - w(K_1 \cup i)]\} - \{f_i - [w(K_1) - w(K_1 \cup i)]\} \\
&= \{f_j - \delta_j(K_1 \cup i - j)\} - \{f_i - \delta_i(K_1)\} \\
&= \Delta_j(K_1 \cup i - j) - \Delta_i(K_1)
\end{aligned}
$$

Desenvolvemos 2 Heurísticas de Troca, as quais denominamos T1 e T2. Vejamos a descrição de cada uma delas.

Heurística de Troca T1

Início

Se $\overline{K}_0 \neq \emptyset$ e $\overline{K}_1 \neq \emptyset$

Então Encontre $E(K_1, r, s) = \max_{i \in \overline{K}_0, j \in \overline{K}_1} E(K_1, i, j)$.

Se $E(K_1, r, s) > 0$

Então $K_0 = K_0 - r$, $K_1 = K_1 \cup r$

$K_1 = K_1 - s$, $K_0 = K_0 \cup s$

$\overline{K}_1 = \overline{K}_1 - s$, $\overline{K}_0 = \overline{K}_0 - r$

Fim

Essa heurística tem uma idéia bem simples: ela avalia o impacto na função objetivo da troca de status de todos os pares de facilidades $(i, j), \forall i \in \overline{K}_0, \forall j \in \overline{K}_1$. Aquele par de facilidades que produzir a maior redução no valor da função objetivo terá seu status trocado, se a redução for positiva. Quando a maior redução for negativa, ela não será propriamente uma redução mas sim o menor acréscimo e então não será vantajoso fazer a troca e o procedimento é finalizado.

Através de experiências computacionais, obtivemos bons resultados com a utilização da Heurística T1, sobretudo para problemas de pequeno porte. Porém, exige-se em cada iteração desse método $| \overline{K}_0 | \times | \overline{K}_1 |$ avaliações de possibilidade de troca de status de facilidades. Essas avaliações serão computacionalmente custosas se os valores de $| \overline{K}_0 |$ e $| \overline{K}_1 |$ forem elevados.

Com o intuito de desenvolver uma heurística igualmente simplificada e computacionalmente viável mesmo para problemas de grande porte, desenvolvemos T2:

Heurística de Troca T2

Início

Se $\overline{K}_0 \neq \emptyset$ e $\overline{K}_1 \neq \emptyset$

Então Encontre r tal que $\Delta_r(K_1) = \min_{i \in \overline{K}_0} \Delta_i(K_1)$,

Encontre s tal que $\Delta_s(K_1 - s) = \max_{i \in \overline{K}_1} \Delta_i(K_1 - i)$,

Se $E(K_1, r, s) > 0$

Então $K_0 = K_0 - r$, $K_1 = K_1 \cup r$

$K_1 = K_1 - s$, $K_0 = K_0 \cup s$

$\overline{K}_1 = \overline{K}_1 - s$, $\overline{K}_0 = \overline{K}_0 - r$

Fim

Nesse procedimento, os valores de r e s são encontrados como nas Heurísticas de Reavaliação de Facilidades Abertas/Fechadas, mostradas anteriormente. Então, a avaliação do impacto na função objetivo da realização da troca entre os status dessas facilidades é feita. Se essa avaliação verificar que há ganho na função objetivo, então a troca é efetuada.

Fizemos uma série de testes computacionais com implementações de T1 e T2. Constatamos que, de um modo geral, para um mesmo problema, a utilização da Heurística T1 no algoritmo consegue refinar mais a solução obtida em relação a utilização de T2, tornado-a mais próxima da solução ótima. Entretanto, a Heurística T1 leva mais tempo para ser executada em relação a T2, sobretudo nos problemas de dimensões elevadas.

A fase 3 do algoritmo será constituída basicamente pela aplicação de uma Heurística de Troca (T1 ou T2) e sua integração com a fase 2. Vejamos a seguir o pseudo-código da fase 3.

3.2.4 Pseudo-código da Fase 3

Para simplificar a exibição da fase 3 do algoritmo, escolheremos a Heurística T2 para compor essa fase. Eis o pseudo-código da fase 3:

FASE 3

Passo 13: (* Heurística de Troca *)

Se $\overline{K}_0 \neq \emptyset$ e $\overline{K}_1 \neq \emptyset$

Então Se $E(K_1, r, s) > 0$

Então $K_0 = K_0 - r$, $K_1 = K_1 \cup r$

$K_1 = K_1 - s$, $K_0 = K_0 \cup s$

$\overline{K}_1 = \overline{K}_1 - s$, $\overline{K}_0 = \overline{K}_0 - r$

Volte para a **fase 2.**

Senão $\overline{K}_0 = \overline{K}_1 = \emptyset$ e **PARE**

Senão $\overline{K}_0 = \overline{K}_1 = \emptyset$ e **PARE**

Essa fase é formada por um único passo onde se verifica se há ainda elementos em $\overline{K}_0$ e $\overline{K}_1$ para que se possa tentar realizar uma troca de status. No caso em que $\overline{K}_0 \neq \emptyset$ e $\overline{K}_1 \neq \emptyset$ e no caso de utilizarmos a Heurística T2, então já temos como resultado da fase 2, a facilidade mais provável de ser aberta (último valor atribuído a r) e a facilidade mais provável de ser fechada (última atribuição a s). Se selecionarmos a Heurística T1 então as facilidades r e s não são aproveitadas da fase 2 mas sim calculadas na própria fase 3, como aquelas que fornecerem o maior valor de $E(K_1, i, j), \forall i \in \overline{K}_0, \forall j \in \overline{K}_1$.

A função $E(K_1, r, s)$ será usada para verificar se há ganho na função

objetivo pela troca de status entre as facilidades r e s. Em caso afirmativo, a troca é realizada e então reexecuta-se a fase 2, pois ela pode ter potencializado a abertura e/ou o fechamento de facilidades.

O último *loop* do algoritmo é então constituído: passos 5 ao 13. Ele é o responsável por iterativamente aplicar as fases 2 e 3. Tenta-se trocar ora individualmente o status de facilidades (fase 2) ora simultaneamente os status de pares de facilidades (fase 3).

Há dois casos em que a troca não é realizada: no caso de $\overline{K}_0$ ou $\overline{K}_1$ ser vazio ou no caso de $E(K_1, r, s) \leq 0$. Em ambos os casos, o algoritmo conclui sua execução no passo 13. Isso significa que não podemos mais melhorar a solução corrente através do nosso algoritmo e ele é então terminado. Evidentemente, se ainda houver facilidades em $\overline{K}_0$ e $\overline{K}_1$, elas não mais serão avaliadas e automaticamente passarão a ser, respectivamente, definitivamente fechadas e definitivamente abertas.

A convergência da fase 3 é simples de ser observada. Se a troca não for realizada, o algoritmo é finalizado imediatamente. Mas, se a troca for realizada, o algoritmo passa para a fase 2.

Para provarmos a convergência do algoritmo, basta mostrarmos que o último loop não pode ser realizado um número infinito de vezes. E, para isso, basta verificar que cada vez que o passo 13 é executado implica em terminar o algoritmo ou em descartar uma facilidade de $\overline{K}_0$ e outra de $\overline{K}_1$. Portanto, após sucessivas iterações da fase 3, os conjuntos $\overline{K}_0$ e $\overline{K}_1$ acabam tornando-se vazios e a condição de término do algoritmo na fase 2 será satisfeita.

3.3 Exemplos de utilização do algoritmo

Veremos a seguir, dois exemplos bastante simples criados para mostrar o funcionamento do nosso algoritmo. São exemplos pequenos (3 facilidades e 5 centros de demanda), mas que ilustrarão significativamente a combinação que fizemos entre testes de redução e heurísticas ADD/DROP.

Os dois exemplos utilizarão a distribuição das facilidades e centros de demanda no plano de acordo com a Figura 3.5.

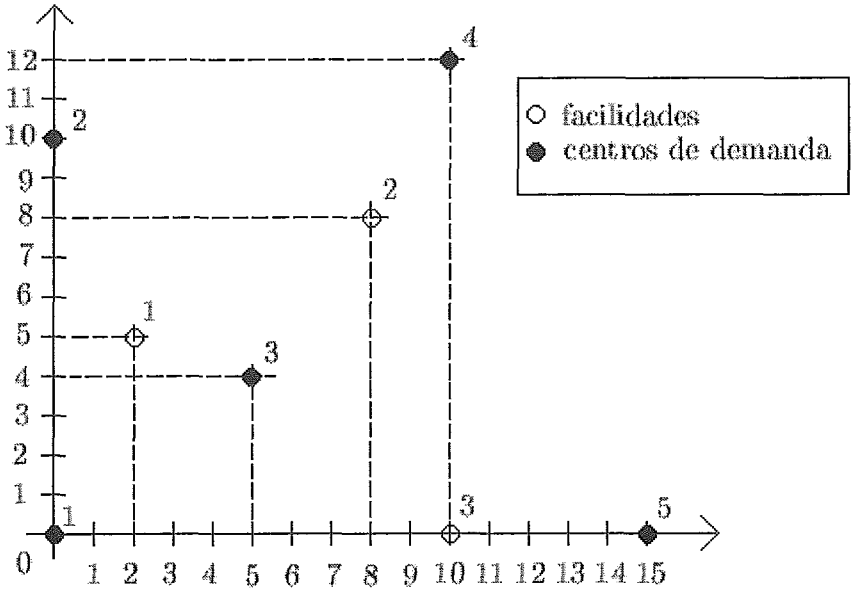

Figura 3.5: Exemplos de aplicação do algoritmo - Distribuição das facilidades e centros de demanda no plano

Consideremos os custos de transporte c_{ij} como sendo a distância euclideana entre a facilidade i e o centro de demanda j. Na Tabela 3.1 visualizamos a matriz de custos de transporte já com as distâncias calculadas.

C_{ij}	1	2	3	4	5
1	$\sqrt{29}$	$\sqrt{29}$	$\sqrt{10}$	$\sqrt{113}$	$\sqrt{194}$
2	$\sqrt{128}$	$\sqrt{68}$	5	$\sqrt{20}$	$\sqrt{113}$
3	10	$\sqrt{200}$	$\sqrt{41}$	12	5

Tabela 3.1: Exemplos de aplicação do algoritmo - Matriz de custos de transporte

Para simplificar, fixaremos todos os parâmetros associados às heurísticas de forma a impossibilitar qualquer refinamento às definições originais das heurísticas: $\beta = 1, \gamma = +\infty$, $\alpha_{abrir}^{inicial} = 0$, $\alpha_{fechar}^{inicial} = 0, \epsilon_{abrir} = 1$ e $\epsilon_{fechar} = 1$.

3.3.1 Exemplo 1

Consideremos o vetor de custos fixos das facilidades da Tabela 3.2.

Facilidades	1	2	3
Custos Fixos	5	8	10

Tabela 3.2: Exemplo 1 - Vetor de custos fixos

Vamos agora mostrar, passo a passo, como nosso algoritmo irá trabalhar sobre os dados desse problema:

FASE 1

Passo 0:

$K_0 = K_1 = \overline{K}_0 = \overline{K}_1 = \emptyset$, $K_2 = \{1, 2, 3\}$

Passo 1:

Calcular $\Delta_i(K_1 \cup K_2 - i), \forall i \in K_2$.

i=1: $5-(10+\sqrt{68}+5+\sqrt{20}+5)+(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{20}+5)=$

$-4,3136<0 \rightarrow$ **Abrir 1**

i=2: $8-(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{113}+5)+(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{20}+5)=$

$1,8419>0$

i=3: $10-(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{20}+\sqrt{113})+(\sqrt{29}+\sqrt{29}+\sqrt{20}+\sqrt{20}+5)=$

$4,3698>0$

$A=\{i \in K_2 \mid \Delta_i(K_1 \cup K_2 - i) \leq 0\}=\{1\}$

$K_1=K_1 \cup A=\emptyset \cup \{1\}=\{1\}$, $K_2=K_2-A=\{1,2,3\}-\{1\}=\{2,3\}$

Passo 2:

Nenhuma operação é realizada pois $K_2 \neq \emptyset$

Passo 3:

Calcular $\Delta_i(K_1), \forall i \in K_2$.

i=2: $8-(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{113}+\sqrt{194})+(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{20}+\sqrt{113})=$

$-1,4562<0$

i=3: $10-(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{113}+\sqrt{194})+(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{113}+5)=$

$1,0716>0 \rightarrow$ **Fechar 3**.

$F=\{i \in K_2 \mid \Delta_i(K_1) \geq 0\}=\{3\}$

$K_0=K_0 \cup F=\emptyset \cup \{1\}=\{3\}$, $K_2=K_2-F=\{2,3\}-\{3\}=\{2\}$

Volte ao passo 1

Passo 1:

Calcular $\Delta_i(K_1 \cup K_2 - i), \forall i \in K_2$.

i=2: $8-(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{113}+\sqrt{194})+(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{20}+\sqrt{113})=$

$-1,4562<0 \rightarrow$ **Abrir 2**.

$A=\{i \in K_2 \mid \Delta_i(K_1 \cup K_2 - i) \leq 0\}=\{2\}$

$K_1=K_1 \cup A=\{1\} \cup \{2\}=\{1,2\}$, $K_2=K_2-A=\{2\}-\{2\}=\emptyset$

Passo 2:

$K_2 = \emptyset \rightarrow$ Vá para a fase 2

FASE 2

Passo 5:

$\alpha_{abrir} = 0$, $\alpha_{fechar} = 0$, $K_0' = \emptyset$

Passo 6:

$\overline{K_0} \cup \overline{K_1} = \emptyset \rightarrow$ **Fim do algoritmo**

Solução

$K_0 = \{3\}, K_1 = \{1, 2\}$

Função objetivo: $(\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{20} + \sqrt{113}) + (5 + 8) = 42,0349$

Nesse exemplo, percebemos que os testes de redução foram suficientes para a determinação dos status de todas as facilidades. Portanto, a solução encontrada pelo nosso algoritmo coincide com a solução ótima do problema. Enumeramos todas as possíveis soluções (Tabela 3.3) e ratificamos este fato.

K_1	Função Objetivo		
$\{1\}$	$(\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{113} + \sqrt{194}) + 5$	=	43,4911
$\{2\}$	$(\sqrt{128} + \sqrt{68} + 5 + \sqrt{20} + \sqrt{113}) + 8$	=	47,6622
$\{3\}$	$(10 + \sqrt{200} + \sqrt{41} + 12 + 5) + 10$	=	57,5453
$\{$ **1,2** $\}$	$(\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{20} + \sqrt{113}) + (5 + 8)$	=	**42,0349**
$\{1, 3\}$	$(\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{113} + 5) + (5 + 10)$	=	44,5628
$\{2, 3\}$	$(10 + \sqrt{68} + 5 + \sqrt{20} + 5) + (8 + 10)$	=	50,7183
$\{1, 2, 3\}$	$(\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{20} + 5) + (5 + 8 + 10)$	=	46,4047

Tabela 3.3: Exemplo 1 - Solução exaustiva

3.3.2 Exemplo 2

Consideremos novamente a matriz de custos de transporte da Tabela 3.1 e o vetor de custos fixos da Tabela 3.4.

Facilidades	1	2	3
Custos Fixos	14	8	7

Tabela 3.4: Exemplo 2 - Vetor de custos fixos

Eis a solução do novo problema:

FASE 1

Passo 0:

$K_0 = K_1 = \overline{K}_0 = \overline{K}_1 = \emptyset$, $K_2 = \{1, 2, 3\}$

Passo 1:

Calcular $\Delta_i(K_1 \cup K_2 - i), \forall i \in K_2$.

i=1: $14-(10+\sqrt{68}+5+\sqrt{20}+5)+(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{20}+5) = 4,6864 > 0$

i=2: $8-(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{113}+5)+(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{20}+5) =$ $1,8419 > 0$

i=3: $7-(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{20}+\sqrt{113})+(\sqrt{29}+\sqrt{29}+\sqrt{20}+\sqrt{20}+5) =$ $1,3698 > 0$

$A = \{i \in K_2 \mid \Delta_i(K_1 \cup K_2 - i) \leq 0\} = \emptyset$

Passo 2:

Nenhuma operação é realizada pois $K_2 \neq \emptyset$

Passo 3:

Calcular $\Delta_i(K_1), \forall i \in K_2$.

i=1: $-\infty < 0$

i=2: $-\infty < 0$

i=3: $-\infty < 0$

$F = \{i \in K_2 \mid \Delta_i(K_1) \geq 0\} = \emptyset$

Passo 4:

Caso especial - $K_1 = \emptyset$

$p = \arg\min_{i \in K_2} \Delta_i(K_1 \cup K_2 - i)$

$min_{i \in K_2} \Delta_i(K_1 \cup K_2 - i) = min\{4,6864; 1,8419; 1,3698\} = 1,3698 \rightarrow p = 3$

$\Delta_p(K_1 \cup K_2 - p) < \gamma \rightarrow$**Abrir 3, temporariamente.**

$K_1 = K_1 \cup p = \emptyset \cup \{3\} = \{3\}$, $\overline{K}_1 = \overline{K}_1 \cup p = \emptyset \cup \{3\} = \{3\}$

$K_2 = K_2 - p = \{1,2,3\} - \{3\} = \{1,2\}$

Volte para o passo 1

Passo 1:

Os valores de $\Delta_i(K_1 \cup K_2 - i), \forall i \in K_2$ não se alteraram

i=1: $4,6864 > 0$

i=2: $1,8419 > 0$

$A = \{i \in K_2 \mid \Delta_i(K_1 \cup K_2 - i) \leq 0\} = \emptyset$

Passo 2:

Nenhuma operação é realizada pois $K_2 \neq \emptyset$

Passo 3:

Calcular $\Delta_i(K_1), \forall i \in K_2$.

i=1: $14 - (10 + \sqrt{200} + \sqrt{41} + 12 + 5) + (\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{113} + 5) =$

$-3,9825 < 0$

i=2: $8-(10+\sqrt{200}+\sqrt{41}+12+5)+(10+\sqrt{68}+5+\sqrt{20}+5) = -6,8269 < 0$

$F = \{i \in K_2 \mid \Delta_i(K_1) \geq 0\} = \emptyset$

Passo 4:

Calcular $\Delta_i(K_1 \cup K_2 - i) + \Delta_i(K_1), \forall i \in K_2$.

i=1: $14 - (10 + \sqrt{68} + 5 + \sqrt{20} + 5) + (\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{20} + 5) + 14 -$
$(10 + \sqrt{200} + \sqrt{41} + 12 + 5) + (\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{113} + 5) = 0,7039$

i=2: $8 - (\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{113} + 5) + (\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{20} + 5) +$
$8 - (10 + \sqrt{200} + \sqrt{41} + 12 + 5) + (10 + \sqrt{68} + 5 + \sqrt{20} + 5) = -4,9849$

$q = \arg\max_{i \in K_2} |\beta \cdot \Delta_i(K_1) + \Delta_i(K_1 \cup K_2 - i)|$

$max\{0,7039; \mid -4,9849 \mid\} = 4,9849 \rightarrow q = 2$

$\beta \cdot \Delta_q(K_1) + \Delta_q(K_1 \cup K_2 - q) = -4,9849 < 0 \rightarrow$ **Abrir 2, temporariamente**

$K_1 = K_1 \cup q = \{3\} \cup \{2\} = \{2,3\}$, $\overline{K}_1 = \overline{K}_1 \cup q = \{3\} \cup \{2\} = \{2,3\}$

$K_2 = K_2 - q = \{1,2\} - \{2\} = \{1\}$

Volte para o passo 2

Passo 2:

Nenhuma operação é realizada pois $K_2 \neq \emptyset$

Passo 3:

Calcular $\Delta_i(K_1), \forall i \in K_2$.

i=1: $14 - (10 + \sqrt{68} + 5 + \sqrt{20} + 5) + (\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{20} + 5) =$
$4,6864 > 0 \rightarrow$ **Fechar 1, temporariamente**

$F = \{i \in K_2 \mid \Delta_i(K_1) \geq 0\} = \{1\}$

$K_0 = K_0 \cup F = \emptyset \cup \{1\} = \{1\}$, $\overline{K}_0 = \overline{K}_0 \cup F = \emptyset \cup \{1\} = \{1\}$

$K_2 = K_2 - F = \{1\} - \{1\} = \emptyset$

Volte para o passo 1

Passo 1:

Calcular $\Delta_i(K_1 \cup K_2 - i), \forall i \in K_2$

$K_2 = \emptyset \rightarrow A = \emptyset$

Passo 2:

$K_2 = \emptyset \rightarrow$ Vá para a fase 2

FASE 2

Passo 5:

$\alpha_{abrir} = 0$, $\alpha_{fechar} = 0$, $K_0' = \overline{K}_0 = \{1\}$

Passo 6:

O algoritmo não pode ser finalizado pois $\overline{K}_0 \cup \overline{K}_1 \neq \emptyset$

Passo 7:

Calcular $\Delta_i(K_1 - i), \forall i \in \overline{K}_1$

i=2: $8 - (10 + \sqrt{200} + \sqrt{41} + 12 + 5) + (10 + \sqrt{68} + 5 + \sqrt{20} + 5) = -6,8269$

i=3: $7 - (\sqrt{128} + \sqrt{68} + 5 + \sqrt{20} + \sqrt{113}) + (10 + \sqrt{68} + 5 + \sqrt{20} + 5) = 0,0561$

$s = \arg\max_{i \in \overline{K}_1} \Delta_i(K_1 - i)$

$\max_{i \in \overline{K}_1} \Delta_i(K_1 - i) = max\{-6,8269; 0,0561\} = 0,0561 \rightarrow s = 3$

Passo 8:

$\Delta_s(K_1 - s) = 0,0561 > 0 \rightarrow$ **Fechar 3**

$K_1 = K_1 - s = \{2,3\} - \{3\} = \{2\}$, $\overline{K}_1 = \overline{K}_1 - s = \{2,3\} - \{3\} = \{2\}$

$K_0 = K_0 \cup s = \{1\} \cup \{3\} = \{1,3\}$

Volte para o passo 7

Passo 7:

Calcular $\Delta_i(K_1 - i), \forall i \in \overline{K}_1$

i=2: $-\infty$

$s = \arg\max_{i \in \overline{K}_1} \Delta_i(K_1 - i)$

$\overline{K}_1 = \{2\} \rightarrow s = 2$

Passo 8:

Nenhuma operação é realizada pois $\Delta_s(K_1 - s) = -\infty < 0$

Passo 9:

Calcular $\Delta_i(K_1), \forall i \in \overline{K}_0$

i=1: $14-(\sqrt{128}+\sqrt{68}+5+\sqrt{20}+\sqrt{113})+(\sqrt{29}+\sqrt{29}+\sqrt{10}+\sqrt{20}+\sqrt{113}) =$

$3,3727$

$r = \arg\min_{i \in \overline{K}_0} \Delta_i(K_1)$

$\overline{K}_0 = \{1\} \rightarrow r = 1$

Passo 10:

Nenhuma operação é realizada pois $\Delta_r(K_1) = 3,3727 > 0$

Passo 11:

Nenhuma operação é realizada pois $\mid \overline{K}_0 \mid = \mid K'_0 \mid$

Passo 12:

$\alpha_{abrir} = \alpha_{fechar} = 0 \rightarrow$ Vá para a fase 3

Passo 13:

$E(K_1, r, s) = (f_r - f_s) + (w(K_1) - w(K_1 \cup r - s)) = (14-8) + [(\sqrt{128} + \sqrt{68} + 5 + \sqrt{20} + \sqrt{113}) - (\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{113} + \sqrt{194})] = -9,1652 < 0 \rightarrow$ A troca não deve ser realizada $\rightarrow \overline{K}_0 = \overline{K}_1 = \emptyset$, **Fim do algoritmo**

Solução

$K_0 = \{1, 3\}, K_1 = \{2\}$

Função objetivo: $(\sqrt{128} + \sqrt{68} + 5 + \sqrt{20} + \sqrt{113}) + 8 = 47,6622$

Verificamos nesse exemplo que o *O-test* e *C-test* não foram suficientes para determinar com exatidão o status de nenhuma facilidade. A Heurística Para o Caso Especial $K_1 = \emptyset$ escolheu para ser aberta uma facilidade que, na solução ótima, deveria ser fechada. Porém, na fase 2, a Heurística de Reavaliação de Facilidades Abertas conseguiu consertar esse erro, melhorando a solução. Enumerando todas as soluções viáveis para o problema (Tabela 3.5), observamos que novamente o ótimo foi atingido pelo nosso algoritmo.

K_1	Função Objetivo		
$\{1\}$	$(\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{113} + \sqrt{194}) + 14$	=	52,4911
$\{\mathbf{2}\}$	$(\sqrt{128} + \sqrt{68} + 5 + \sqrt{20} + \sqrt{113}) + 8$	=	**47,6622**
$\{3\}$	$(10 + \sqrt{200} + \sqrt{41} + 12 + 5) + 7$	=	54,5453
$\{1, 2\}$	$(\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{20} + \sqrt{113}) + (14 + 8)$	=	51,0349
$\{1, 3\}$	$(\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{113} + 5) + (14 + 7)$	=	50,5628
$\{2, 3\}$	$(10 + \sqrt{68} + 5 + \sqrt{20} + 5) + (8 + 7)$	=	47,7183
$\{1, 2, 3\}$	$(\sqrt{29} + \sqrt{29} + \sqrt{10} + \sqrt{20} + 5) + (14 + 8 + 7)$	=	52,4047

Tabela 3.5: Exemplo 2 - Solução exaustiva

Capítulo 4

Resultados Computacionais

Para verificar o desempenho computacional do algoritmo que desenvolvemos, fizemos uma implementação e a submetemos a uma série de testes.

Adaptamos ao SPLP diversos problemas-teste comumente utilizados na literatura e geramos aleatoriamente mais uma série de problemas. Os testes que utilizamos foram os seguintes:

1. *Problemas não-capacitados do Beasley*: adaptação dos problemas de localização capacitados disponíveis na *OR-Library* [6] de J. E. Beasley.

2. *Problemas do Thizy*: adaptação de problemas de localização também capacitados, enfocados em Cornuèjols, Sridharan e Thizy [15].

3. *Problemas de Karg e Thompson*: adaptação de instâncias do caixeiro viajante presentes em Karg e Thompson [31].

4. *Problemas de p-medianas do Beasley*: adaptação dos problemas de p-medianas disponíveis na *OR-Library* [6] de J. E. Beasley.

5. *Problemas não-euclideanos gerados aleatoriamente*: problemas produzidos a partir de um gerador aleatório de SPLP não-euclideanos que desenvolvemos.

Todos estes grupos de problemas que testamos são modelos discretos de localização, exceto os Problemas de p-medianas do Beasley que são modelos em rede. Os quatro primeiros grupos de problemas são euclideanos e o último grupo é formado por problemas não-euclideanos.

Utilizamos os dois primeiros grupos de problemas para verificarmos a qualidade da solução obtida pelo nosso algoritmo bem como a contribuição dos testes de redução e das diversas heurísticas na determinação dessa solução. Através dos demais grupos de problemas objetivamos comparar o desempenho computacional do nosso algoritmo frente ao de Galvão e Raggi.

Resumidamente, o algoritmo de Galvão e Raggi é um procedimento que encontra a solução ótima do PLNC e se divide em três fases: um algoritmo primal-dual, um algoritmo de otimização subgradiente para resolver o dual lagrangeano e um algoritmo de separação e avaliação. Ele tem uma estrutura hierárquica onde um dado estágio só é ativado se o ótimo não for atingido no estágio precedente. Esse método foi bem-sucedido não só na solução do PLNC, mas em outros problemas não capacitados como o P-MED, o P-MEDCF e o SPLP.

A implementação do algoritmo apresentado nesse trabalho foi feita na linguagem C padrão. Utilizamos o ambiente de desenvolvimento Bloodshed Dev-C++ 4.0. Já o algoritmo de Galvão e Raggi foi implementado originalmente em Fortran mas, em 1999, uma nova implementação foi feita em

Borland Delphi pelos autores. Utilizamos essa implementação mais recente para compararmos os resultados computacionais obtidos. Executamos ambas as implementações em um PC com processador AMD-K6, 450Mhz, 128Mb de RAM.

Sobre a implementação do algoritmo de Galvão e Raggi, vale ressaltar que apesar da formulação do algoritmo deles ter sido feita abrangendo o PLNC de uma forma geral, as implementações feitas por eles são restritas a problemas que incorporam uma estrutura de rede tal que $I \equiv J$ e $[c_{ij}] \equiv [d_{ij}]$, a matriz de distâncias da rede, segundo se observa em Galvão e Raggi [21]. Ou seja, tomando N como o conjunto de nós da rede, em todos os problemas, $\mid I \mid = \mid J \mid = \mid N \mid$, $c_{ij} = c_{ji}, \forall i,j \in N$, $c_{ij} > 0$, se $i \neq j, \forall i,j \in N$ e $c_{ii} = 0, \forall i \in N$. Então, só utilizaremos problemas que possuam essas características para comparamos o desempenho da nossa implementação ao da implementação do algoritmo de Galvão e Raggi. Observamos que o que vai diferenciar os problemas euclideanos dos não-euclideanos que vamos trabalhar nos dois algoritmos será exatamente se, para cada 3 nós distintos da rede, a desigualdade triangular é ou não satisfeita (veja seção 1.1.3).

Não utilizamos o algoritmo de Galvão e Raggi como parâmetro de comparação nos dois primeiros grupos de problemas devido a estrutura destes problemas: na maioria deles, $\mid I \mid \neq \mid J \mid$, o que inviabilizaria de imediato o uso da implementação cedida pelos autores na solução desses problemas.

Em relação a nossa implementação, cabe mencionar que, em princípio, atribuiremos a todos os parâmetros associados às heurísticas valores que impossibilitem qualquer refinamento às definições originais das heurísticas: $\beta = 1, \gamma = +\infty$, $\alpha_{abrir}^{inicial} = 0$, $\alpha_{fechar}^{inicial} = 0, \epsilon_{abrir} = 1$ e $\epsilon_{fechar} = 1$. Além disso,

utilizaremos para compor a fase 3 do algoritmo a Heurística T1 em todos os grupos de problemas testados exceto nos problemas de p-medianas do Beasley, onde ela obteve um desempenho muito ruim em termos de tempos computacionais sobretudo nos problemas de dimensões elevadas. Para esse grupo de problemas utilizamos a Heurística T2.

Antes de apresentarmos os resultados dos testes que fizemos, faremos algumas convenções para simplificar essa apresentação. Convencionamos definir o tamanho de um problema como sendo $(|I| \times |J|)$. Nas tabelas de resultados computacionais que veremos a seguir, adotamos uma sigla para cada procedimento, conservando as siglas associadas a cada heurística nos capítulos 2 e 3 e diferenciando a aplicação do *O-test* e *C-test* antes ou após a execução de alguma heurística (no passo 4 do algoritmo). As siglas associadas a cada procedimento podem ser vistas na Tabela 4.1.

Sigla	Procedimento
TR	Testes de redução antes da primeira execução do passo 4
TR'	Testes de redução após a primeira execução do passo 4
HCE	Heurística Para o Caso Especial $K_1 = \emptyset$
HSD	Heurística de Soma de Deltas
HRF	Heurísticas de Reavaliação de Facilidades Abertas/Fechadas
HT	Heurística de Troca

Tabela 4.1: Siglas usadas para os testes e heurísticas do algoritmo

Para cada um desses procedimentos, exceto a Heurística de Troca, apresentaremos um par de números "a/b" , significando que a facilidades foram abertas e b fechadas. Mais especificamente, na Tabela 4.2, observamos o significado particular de a e b em cada procedimento.

O valor da solução heurística obtida pelo nosso algoritmo será representado pela sigla SH e a solução ótima por S*. Os tempos computacionais

Procedimento	a	b
TR	n.º de facilidades abertas pelo *O-test* antes da 1.ª execução do passo 4	n.º de facilidades fechadas pelo *C-test* antes da 1.ª execução do passo 4
TR'	n.º de facilidades abertas pelo *O-test* após a 1.ª execução do passo 4	n.º de facilidades fechadas pelo *C-test* após a 1.ª execução do passo 4
HCE	assume somente dois valores: 0 ou 1. "1", caso tenha sido aplicada e "0", caso contrário	n.º de facilidades fechadas pela heurística até que tenha sido escolhida uma facilidade a ser aberta
HSD	n.º de facilidades abertas pela heurística	n.º de facilidades fechadas pela heurística
HRF	n.º de facilidades de $\overline{K}_0$ que foram transferidas para K_1	n.º de facilidades de $\overline{K}_1$ que foram transferidas para K_0

Tabela 4.2: Significado de a e b nos procedimentos

obtidos pelo nosso algoritmo serão indicados pela sigla TH e os tempos obtidos pela execução do algoritmo de Galvão e Raggi por TG. Ambos serão dados em segundos. Teremos ainda T como sendo a quantidade de trocas de status de pares de facilidades efetuadas na fase 3 do algoritmo e "Total" que será representado também por um par de números "a/b", significando que ao todo foram abertas a facilidades e fechadas b facilidades.

Calcularemos os erros percentuais obtidos usando a seguinte fórmula: Erro = ((SH - S*)/S*)×100%. Exibiremos os erros percentuais com duas casas decimais de precisão bem como os tempos computacionais.

Vejamos a seguir detalhadamente os testes que fizemos e analisaremos os resultados obtidos.

4.1 Problemas não-capacitados do Beasley

Esses problemas foram desenvolvidos originalmente para o problema de localização capacitado mas também se encontra na *OR Library* [6] de J. E. Beasley a solução ótima desses problemas, desprezando-se as capacidades de cada facilidade , o que resulta em problemas não capacitados. Eles são ao todo 15 problemas, cujos nomes e tamanhos são os seguintes: cap71 - cap74 (16 × 50), capl01 - capl04 (25 × 50), capl31 - capl34 (50 × 50), cap a - cap c (100 × 1000). Os tempos computacionais e os erros obtidos pelo nosso algoritmo se encontram na Tabela 4.3 e a contribuição dos testes de redução e heurísticas na solução dos problemas podemos visualizar na Tabela 4.4.

Problema	S*	Fase 1		Fases 2 e 3		TH
		SH	Erro	SH	Erro	
Cap71	932615,75	932615,75	0,00	932615,75	0,00	0,00
Cap72	977799,40	977799,40	0,00	977799,40	0,00	0,00
Cap73	1010641,45	1010641,45	0,00	1010641,45	0,00	0,00
Cap74	1034976,98	1034976,98	0,00	1034976,98	0,00	0,00
Capl01	796648,44	796648,44	0,00	796648,44	0,00	0,00
Capl02	854704,20	854704,20	0,00	854704,20	0,00	0,00
Capl03	893782,11	894801,16	0,11	893782,11	0,00	0,00
Capl04	928941,75	928941,75	0,00	928941,75	0,00	0,00
Capl31	793439,56	793439,56	0,00	793439,56	0,00	0,00
Capl32	851495,33	852762,88	0,15	851495,33	0,00	0,00
Capl33	893076,71	893076,71	0,00	893076,71	0,00	0,00
Capl34	928941,75	934586,98	0,61	928941,75	0,00	0,00
Cap a	17156454,48	18374078,20	7,10	17156454,48	0,00	2,39
Cap b	12979071,58	13501789,70	4,03	12979071,58	0,00	3,52
Cap c	11505594,33	11825903,93	2,78	11509361,66	0,03	2,45
Média	—	—	**0,99**	—	**0,00**	**0,56**

Tabela 4.3: Problemas não-capacitados do Beasley - Erros obtidos e tempos computacionais

Problema	Fase 1				Fases 2 e 3		Total
	TR	HCE	HSD	TR'	HRF	T	
Cap71	10/02	00/00	01/00	00/03	00/00	0	11/05
Cap72	07/03	00/00	01/02	01/02	00/00	0	09/07
Cap73	03/07	00/00	01/02	01/02	00/00	0	05/11
Cap74	03/11	00/00	00/01	01/00	00/00	0	04/12
Cap101	11/05	00/00	03/01	01/04	00/00	0	15/10
Cap102	06/06	00/00	02/03	03/05	00/00	0	11/14
Cap103	04/07	00/00	02/03	01/08	00/00	1	07/18
Cap104	02/11	00/00	01/01	01/09	00/00	0	04/21
Cap131	07/09	00/00	06/08	02/18	00/00	0	15/35
Cap132	05/18	00/00	03/09	03/12	00/00	1	11/39
Cap133	03/22	00/00	02/13	03/07	00/00	0	08/42
Cap134	02/29	00/00	01/07	01/10	00/00	1	04/46
Cap a	00/00	01/00	02/28	02/67	00/01	2	04/96
Cap b	00/00	01/00	04/25	04/66	00/02	4	07/93
Cap c	00/00	01/00	07/12	01/79	02/03	1	08/92

Tabela 4.4: Problemas não-capacitados do Beasley - Contribuição dos testes de redução e heurísticas

Percebemos que para 9 dos 15 problemas, a solução ótima foi obtida logo na fase 1. Evidentemente, as fases 2 e 3 não conseguiram modificar os status de quaisquer facilidades na solução desses 9 problemas. Porém, nos 6 restantes, as fases 2 e 3 conseguiram significativamente melhorar a solução obtida na fase 1.

Nessa primeira bateria de testes, obtivemos soluções bem próximas aos valores ótimos e tempos computacionais bastante reduzidos. Observamos que o nosso algoritmo obteve um erro máximo de $0,03\%$ em relação à solução ótima e os tempos computacionais foram, em média, inferiores a 0,6s.

4.2 Problemas do Thizy

Em Cornuéjols, Sridharan e Thizy [15] encontramos resultados computacionais pertinentes a soluções de uma série de problemas de localização capacitados. Resolvemos adaptar esses problemas ao nosso algoritmo e, para isso, desprezamos as capacidades das facilidades.

E para verificarmos a qualidade da solução que obtivemos, utilizamos uma implementação do algoritmo de Bornstein e Valiati [9] para obtermos os valores ótimos desses problemas. Esse algoritmo é voltado para a solução de problemas de localização capacitados e, na solução de cada problema, atribuímos à capacidade de cada facilidade o valor da soma das demandas, tornando-o não-capacitado.

São ao todo 150 problemas, divididos em 6 subgrupos de 25 problemas, conforme seus tamanhos. O desempenho computacional do nosso algoritmo frente a esses problemas se encontra nas tabelas: 4.5 (8×25), 4.6 (16×25), 4.7 (25×25), 4.8 (16×50), 4.9 (33×50) e 4.10 (50×50).

Esses problemas possuem dimensões bastante inferiores aos do Beasley que exibimos anteriormente. Porém, uma dificuldade intrínseca a esses problemas pode ser percebida pela incapacidade dos testes de redução em conseguir determinar de forma ótima o status das facilidades. Ou seja, em todos os 150 problemas, a Heurística Para o Caso Especial $K_1 = \emptyset$ foi acionada. Por isso, omitimos nas tabelas a coluna TR (todas as entradas seriam "00/00") e a coluna HCE (todas as entradas seriam "01/00").

Problema	S*	SH	Erro	Fase 1		Fases 2 e 3		Total	TH
				HSD	TR'	HRF	T		
Thiz1-a1	2309	2309	0,00	00/00	00/07	00/00	0	01/07	0,00
Thiz1-a2	2908	2908	0,00	00/00	00/07	00/00	0	01/07	0,00
Thiz1-a3	2231	2231	0,00	00/00	00/07	00/00	0	01/07	0,00
Thiz1-a4	3230	3230	0,00	00/00	01/06	00/01	0	01/07	0,00
Thiz1-a5	3678	3678	0,00	00/00	00/07	00/00	1	01/07	0,00
Thiz2-a1	2852	2852	0,00	00/00	00/07	00/00	0	01/07	0,00
Thiz2-a2	3231	3231	0,00	00/00	00/07	00/00	0	01/07	0,00
Thiz2-a3	4804	4804	0,00	00/00	00/07	00/00	0	01/07	0,00
Thiz2-a4	4042	4042	0,00	00/00	00/07	00/00	1	01/07	0,00
Thiz2-a5	2763	2763	0,00	00/00	00/07	00/00	0	01/07	0,00
Thiz3-a1	2701	2701	0,00	00/00	00/07	00/00	0	01/07	0,00
Thiz3-a2	2699	2699	0,00	00/00	00/07	00/00	0	01/07	0,00
Thiz3-a3	2267	2267	0,00	00/00	00/07	00/00	1	01/07	0,00
Thiz3-a4	3022	3022	0,00	00/00	00/07	00/00	1	01/07	0,00
Thiz3-a5	3055	3055	0,00	00/00	00/07	00/00	0	01/07	0,00
Thiz5-a1	3393	3393	0,00	00/00	00/07	00/00	1	01/07	0,00
Thiz5-a2	3726	3726	0,00	00/00	00/07	00/00	1	01/07	0,00
Thiz5-a3	2522	2522	0,00	00/02	01/04	00/01	0	01/07	0,00
Thiz5-a4	3024	3024	0,00	00/00	00/07	00/00	1	01/07	0,00
Thiz5-a5	2834	2834	0,00	00/00	00/07	00/00	0	01/07	0,00
Thizx-a1	2708	2708	0,00	00/00	00/07	00/00	0	01/07	0,00
Thizx-a2	3203	3203	0,00	00/00	00/07	00/00	0	01/07	0,00
Thizx-a3	3204	3204	0,00	00/00	00/07	00/00	0	01/07	0,00
Thizx-a4	3852	3852	0,00	00/00	00/07	00/00	0	01/07	0,00
Thizx-a5	3534	3534	0,00	00/00	00/07	00/00	0	01/07	0,00
Média	—	—	**0,00**	—	—	—	—	—	**0,00**

Tabela 4.5: Problemas do Thizy (8×25)

Se os testes de redução foram os mais importantes na solução dos problemas não-capacitados do Beasley, nos problemas do Thizy percebemos que as heurísticas foram as peças fundamentais. Mesmo assim, conseguimos soluções de qualidade muito boa: alcançamos o ótimo em $93,33\%$ dos problemas (em 140 dos 150 problemas). O máximo erro médio por grupo de problemas foi de $0,45\%$ e a maior disparidade em relação ao valor ótimo encontrada na solução de um problema foi de $8,49\%$.

Problema	S*	SH	Erro	Fase 1		Fases 2 e 3		Total	TH
				HSD	TR'	HRF	T		
Thiz1-b1	2894	2894	0,00	00/00	00/15	00/00	1	01/15	0,00
Thiz1-b2	2051	2051	0,00	01/00	00/14	00/01	0	01/15	0,00
Thiz1-b3	2865	2865	0,00	00/00	01/14	00/00	0	02/14	0,00
Thiz1-b4	2491	2491	0,00	00/00	01/14	00/01	0	02/14	0,00
Thiz1-b5	2727	2727	0,00	00/00	00/15	00/00	1	01/15	0,00
Thiz2-b1	2500	2500	0,00	01/03	00/11	00/01	0	01/15	0,00
Thiz2-b2	2366	2366	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz2-b3	2041	2041	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz2-b4	2648	2648	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz2-b5	2736	2763	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz3-b1	2026	2138	5,53	01/00	00/14	00/00	1	02/14	0,00
Thiz3-b2	2264	2264	0,00	01/05	00/09	00/00	0	02/14	0,00
Thiz3-b3	2243	2243	0,00	00/01	01/13	00/00	0	02/14	0,00
Thiz3-b4	2176	2176	0,00	00/02	01/12	00/01	0	01/15	0,00
Thiz3-b5	2029	2029	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz5-b1	2133	2133	0,00	01/01	00/13	00/01	0	01/15	0,00
Thiz5-b2	2234	2234	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz5-b3	2420	2420	0,00	01/00	00/14	00/01	0	01/15	0,00
Thiz5-b4	2533	2533	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz5-b5	2767	2779	0,43	00/02	01/12	00/00	0	02/14	0,00
Thizx-b1	2970	2970	0,00	00/00	00/15	00/00	1	01/15	0,00
Thizx-b2	2751	2751	0,00	00/00	00/15	00/00	0	01/15	0,00
Thizx-b3	3200	3200	0,00	00/01	01/13	00/01	0	01/15	0,00
Thizx-b4	2501	2501	0,00	00/00	00/15	00/00	0	01/15	0,00
Thizx-b5	3039	3039	0,00	00/00	00/15	00/00	0	01/15	0,00
Média	—	—	**0,24**	—	—	—	—	—	**0,00**

Tabela 4.6: Problemas do Thizy (16 × 25)

Os tempos computacionais foram bastante reduzidos. Obtivemos tempos desprezíveis para 4 dos 6 grupos de problemas testados. E nos 2 grupos em que o tempo foi relevante, o máximo valor médio obtido foi de 0,02s.

Através da variação dos parâmetros que são associados às heurísticas é possível obter soluções ainda melhores. Em todos os problemas, observamos que a Heurística Para o Caso Especial $K_1 = \emptyset$ foi acionada, tomando $\gamma = +\infty$, o que não produz qualquer refinamento a essa heurística. Uti-

Problema	S*	SH	Erro	Fase 1		Fases 2 e 3		Total	TH
				HSD	TR'	HRF	T		
Thiz1-c1	2093	2101	0,38	00/05	01/18	00/00	0	02/23	0,00
Thiz1-c2	2547	2547	0,00	00/00	01/23	00/00	0	02/23	0,00
Thiz1-c3	2704	2764	2,22	00/02	01/21	00/00	0	02/23	0,00
Thiz1-c4	2297	2410	4,92	01/07	00/16	00/00	0	02/23	0,00
Thiz1-c5	2420	2420	0,00	00/00	00/24	00/00	0	01/24	0,00
Thiz2-c1	2861	2861	0,00	00/01	01/22	00/01	0	01/24	0,00
Thiz2-c2	2767	2767	0,00	00/00	00/24	00/00	0	01/24	0,00
Thiz2-c3	2723	2723	0,00	00/01	01/22	00/00	0	02/23	0,00
Thiz2-c4	2382	2382	0,00	00/01	01/22	00/01	0	01/24	0,00
Thiz2-c5	2117	2117	0,00	01/05	00/18	00/01	0	01/24	0,00
Thiz3-c1	1743	1743	0,00	01/03	00/20	00/00	0	02/23	0,00
Thiz3-c2	1671	1671	0,00	00/00	00/24	00/00	0	01/24	0,00
Thiz3-c3	2387	2387	0,00	00/06	01/17	00/00	1	02/23	0,00
Thiz3-c4	1707	1707	0,00	00/02	02/20	00/01	0	02/23	0,00
Thiz3-c5	1976	1976	0,00	00/00	01/23	00/00	0	02/23	0,00
Thiz5-c1	1956	1956	0,00	00/00	00/24	00/00	0	01/24	0,00
Thiz5-c2	2119	2119	0,00	00/01	01/22	00/01	0	01/24	0,00
Thiz5-c3	1604	1604	0,00	00/03	01/20	00/01	0	01/24	0,00
Thiz5-c4	1741	1741	0,00	00/02	01/21	00/01	0	01/24	0,00
Thiz5-c5	1785	1785	0,00	00/00	01/23	00/00	0	02/23	0,00
Thizx-c1	3036	3036	0,00	00/09	01/14	00/01	0	01/24	0,00
Thizx-c2	1836	1836	0,00	00/00	00/24	00/00	0	01/24	0,00
Thizx-c3	2362	2362	0,00	00/00	01/23	00/01	0	01/24	0,00
Thizx-c4	2400	2400	0,00	00/01	01/22	00/00	0	02/23	0,00
Thizx-c5	1788	1788	0,00	00/00	00/24	00/00	0	01/24	0,00
Média	—	—	**0,30**	—	—	—	—	—	**0,00**

Tabela 4.7: Problemas do Thizy (25×25)

lizando um valor mais conveniente para γ, foi possível obter alguma melhora. Executamos então novamente os 10 problemas em que o valor ótimo não foi atingido, utilizando $\gamma = 0$ e mantivemos os valores dos demais parâmetros. Os resultados podem ser vistos na Tabela 4.11, onde verificamos que em 5 dos 10 problemas o ótimo foi atingido. As soluções dos demais problemas não sofreram alterações.

Na verdade, testamos esses problemas utilizando diversos valores de γ e,

Problema	S*	SH	Erro	Fase 1		Fases 2 e 3		Total	TH
				HSD	TR'	HRF	T		
Thiz1-d1	3890	3890	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz1-d2	4302	4302	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz1-d3	4510	4510	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz1-d4	4885	5011	2,58	00/05	01/09	00/01	1	01/15	0,00
Thiz1-d5	4182	4182	0,00	01/02	00/12	00/00	1	02/14	0,00
Thiz2-d1	5348	5348	0,00	00/00	00/15	00/00	1	01/15	0,00
Thiz2-d2	5053	5053	0,00	00/00	00/15	00/00	1	01/15	0,00
Thiz2-d3	5006	5006	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz2-d4	6019	6530	8,49	00/00	01/14	00/00	0	02/14	0,00
Thiz2-d5	5925	5925	0,00	00/00	01/14	00/00	1	02/14	0,00
Thiz3-d1	3878	3878	0,00	01/00	00/14	00/01	0	01/15	0,00
Thiz3-d2	3676	3676	0,00	01/04	01/09	00/01	0	02/14	0,00
Thiz3-d3	4075	4075	0,00	00/06	02/07	00/01	0	02/14	0,00
Thiz3-d4	4233	4233	0,00	01/06	01/07	00/00	0	03/13	0,00
Thiz3-d5	4344	4344	0,00	01/02	00/12	00/00	0	02/14	0,00
Thiz5-d1	4629	4629	0,00	01/06	00/08	00/00	0	02/14	0,00
Thiz5-d2	4278	4278	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz5-d3	5094	5094	0,00	00/06	01/08	00/00	1	02/14	0,00
Thiz5-d4	4917	4917	0,00	00/00	00/15	00/00	0	01/15	0,00
Thiz5-d5	5026	5026	0,00	00/00	00/15	00/00	1	01/15	0,00
Thizx-d1	5337	5337	0,00	00/03	01/11	00/00	1	02/14	0,00
Thizx-d2	5781	5781	0,00	00/00	00/15	00/00	1	01/15	0,00
Thizx-d3	4915	4928	0,26	01/02	00/12	00/00	0	02/14	0,00
Thizx-d4	4575	4575	0,00	00/00	01/14	00/01	0	01/15	0,00
Thizx-d5	5444	5444	0,00	00/00	01/14	00/00	0	02/14	0,00
Média	—	—	**0,45**	—	—	—	—	—	**0,00**

Tabela 4.8: Problemas do Thizy (16×50)

para todos eles, obtivemos soluções iguais ou piores que as obtidas quando tomamos $\gamma = 0$. Por isso escolhemos esse valor. De forma similar, nos próximos grupos de problemas que apresentaremos, quando for estabelecido um valor para um parâmetro diferente dos iniciais, subentende-se que essa escolha foi feita após experimentos com vários parâmetros.

Foram também feitos experimentos variando o valor de β. Como os problemas do Thizy têm a tendência de fechar um grande número de facilidades,

Problema	S*	SH	Erro	Fase 1		Fases 2 e 3		Total	TH
				HSD	TR'	HRF	T		
Thiz1-e1	3891	3891	0,00	01/11	00/20	00/01	1	01/32	0,01
Thiz1-e2	4352	4352	0,00	00/07	01/24	00/00	0	02/31	0,01
Thiz1-e3	4143	4143	0,00	01/13	01/17	00/01	1	02/31	0,01
Thiz1-e4	4479	4479	0,00	01/10	00/21	00/00	1	02/31	0,01
Thiz1-e5	4551	4551	0,00	00/09	00/22	00/00	1	02/31	0,01
Thiz2-e1	3903	3903	0,00	01/05	00/26	00/01	0	01/32	0,00
Thiz2-e2	3653	3653	0,00	00/01	01/30	00/00	0	02/31	0,00
Thiz2-e3	4435	4435	0,00	01/02	00/29	00/00	1	02/31	0,01
Thiz2-e4	4164	4164	0,00	01/04	00/27	00/00	0	02/31	0,01
Thiz2-e5	4272	4272	0,00	00/01	02/29	00/00	0	03/30	0,01
Thiz3-e1	2989	2989	0,00	01/05	01/25	00/01	0	02/31	0,00
Thiz3-e2	3345	3345	0,00	01/01	01/29	00/01	0	02/31	0,00
Thiz3-e3	3399	3399	0,00	01/00	00/31	00/00	0	02/31	0,00
Thiz3-e4	2679	2679	0,00	01/02	00/29	00/00	0	02/31	0,00
Thiz3-e5	4005	4005	0,00	01/01	01/29	00/00	2	03/30	0,00
Thiz5-e1	3667	3667	0,00	00/01	01/30	00/00	0	02/31	0,00
Thiz5-e2	3850	3850	0,00	01/06	01/24	00/00	0	03/30	0,01
Thiz5-e3	3582	3582	0,00	01/07	00/24	00/00	1	02/31	0,01
Thiz5-e4	3986	3986	0,00	01/15	00/16	00/00	1	02/31	0,01
Thiz5-e5	4026	4026	0,00	01/00	00/31	00/00	0	02/31	0,01
Thizx-e1	4016	4046	0,75	00/02	01/29	00/00	0	02/31	0,01
Thizx-e2	4071	4071	0,00	00/02	01/29	00/00	0	02/31	0,00
Thizx-e3	4692	4692	0,00	01/09	00/22	00/00	0	02/31	0,01
Thizx-e4	3963	3963	0,00	01/07	00/24	00/01	1	01/32	0,00
Thizx-e5	4242	4242	0,00	01/10	00/21	00/00	1	02/31	0,01
Média	—	—	**0,03**	—	—	—	—	—	**0,01**

Tabela 4.9: Problemas do Thizy (33×50)

escolhemos valores de $\beta < 1$. No entanto, nenhuma melhora adicional foi obtida.

De uma forma geral, observamos que todos os procedimentos de determinação/modificação de status de facilidades que desenvolvemos no nosso algoritmo foram utilizados nesses conjuntos de problemas. As soluções obtidas foram bastante próximas aos valores ótimos e os tempos computacionais reduzidos que obtivemos são indícios razoáveis da viabilidade computacional

Problema	S*	SH	Erro	Fase 1		Fases 2 e 3		Total	TH
				HSD	TR'	HRF	T		
Thiz1-f1	3138	3138	0,00	01/00	01/47	00/00	0	03/47	0,01
Thiz1-f2	3861	3861	0,00	01/11	01/36	00/00	0	03/47	0,02
Thiz1-f3	3850	3850	0,00	01/17	01/30	00/00	0	03/47	0,02
Thiz1-f4	3745	3745	0,00	01/05	00/43	00/00	0	02/48	0,02
Thiz1-f5	3825	3825	0,00	01/17	01/30	00/00	1	03/47	0,02
Thiz2-f1	3704	3704	0,00	00/17	01/31	00/00	0	02/48	0,02
Thiz2-f2	3919	3919	0,00	00/17	02/30	00/00	1	03/47	0,02
Thiz2-f3	3472	3472	0,00	01/14	01/33	00/00	0	03/47	0,02
Thiz2-f4	3014	3014	0,00	01/13	02/33	00/01	0	03/47	0,02
Thiz2-f5	3460	3460	0,00	01/04	00/44	00/00	0	02/48	0,01
Thiz3-f1	2789	2789	0,00	01/03	01/44	00/00	0	03/47	0,01
Thiz3-f2	2989	2989	0,00	02/05	00/42	00/00	0	03/47	0,02
Thiz3-f3	3019	3019	0,00	02/20	00/27	00/00	0	03/47	0,02
Thiz3-f4	2832	2832	0,00	01/01	01/46	00/00	0	03/47	0,01
Thiz3-f5	2622	2622	0,00	00/00	03/46	00/01	0	03/47	0,01
Thiz5-f1	3475	3475	0,00	02/02	00/45	00/00	2	03/47	0,02
Thiz5-f2	3061	3061	0,00	01/06	00/42	00/00	0	02/48	0,01
Thiz5-f3	3019	3019	0,00	01/00	01/47	00/00	0	03/47	0,01
Thiz5-f4	3263	3263	0,00	01/10	01/37	00/00	0	03/47	0,02
Thiz5-f5	3386	3386	0,00	02/21	00/26	00/00	0	03/47	0,02
Thizx-f1	3604	3604	0,00	00/07	02/40	00/01	0	02/48	0,01
Thizx-f2	4138	4166	0,68	01/00	01/47	00/00	0	03/47	0,01
Thizx-f3	4073	4073	0,00	01/17	00/31	00/00	0	02/48	0,02
Thizx-f4	3442	3442	0,00	01/04	01/43	01/01	0	03/47	0,01
Thizx-f5	3654	3654	0,00	01/12	00/36	00/01	0	01/49	0,02
Média	—	—	**0,03**	—	—	—	—	—	**0,02**

Tabela 4.10: Problemas do Thizy (50 × 50)

da implementação do nosso algoritmo.

Porém, os problemas mostrados até aqui têm algumas semelhanças: em quase todos eles o número de facilidades abertas na solução é inferior ao número de facilidades fechadas, a cardinalidade do conjunto I não foi superior a 100 e todos eles são problemas euclideanos. Então, objetivando observar o comportamento do nosso algoritmo também diante de problemas com características diferentes dessas, fizemos mais uma série de testes.

Problema	SH	Erro	Fase 1			Fases 2 e 3		Total	TH
			HCE	HSD	TR'	HRF	T		
Thiz3-b1	2026	0,00	01/14	00/00	00/01	00/00	0	01/15	0,00
Thiz5-b5	2779	0,43	01/13	00/01	01/00	00/00	0	02/14	0,01
Thiz1-c1	2093	0,00	01/23	00/00	00/01	00/00	0	01/24	0,01
Thiz1-c3	2764	2,22	01/22	01/00	00/01	00/00	0	02/23	0,01
Thiz1-c4	2297	0,00	01/23	00/00	00/01	00/00	0	01/24	0,01
Thiz1-d4	4885	0,00	01/14	00/00	00/01	00/00	1	01/15	0,01
Thiz2-d4	6530	8,49	01/14	00/00	01/00	00/00	0	02/14	0,00
Thizx-d3	4928	0,26	01/14	00/00	01/00	00/00	0	02/14	0,01
Thizx-e1	4016	0,00	01/29	00/01	01/01	00/00	0	02/31	0,02
Thizx-f2	4166	0,68	01/45	01/00	01/02	00/00	0	03/47	0,03

Tabela 4.11: Problemas do Thizy ($\gamma = 0$)

Além disso, nossa implementação foi testada em uma máquina bastante moderna, com elevada capacidade de processamento e os tempos reduzidos não significam necessariamente que são bons. Por isso é imprescindível comparar a nossa implementação a alguma implementação reconhecidamente boa executada na mesma máquina. Para isso, utilizamos uma implementação do algoritmo de Galvão e Raggi [21] como parâmetro de comparação.

Nas próximas seções veremos em detalhes esses testes.

4.3 Problemas de Karg e Thompson

Utilizamos um conjunto de testes publicado em 1964 por Karg e Thompson [31]. A idéia desse trabalho era desenvolver heurísticas para o problema do caixeiro viajante e, para testar esses procedimentos, um conjunto de problemas foi desenvolvido baseado na distribuição geográfica de importantes cidades norte-americanas. Esses testes foram usados em diversos trabalhos posteriormente como em Erlenkotter [19] e, mais recentemente, em Al-Sultan e Al-Fawzan[2]. Eles utilizaram particularmente dois problemas: um de tamanho 33×33 e outro de 57×57. Entretanto, estes testes não fornecem os custos fixos de instalação das facilidades. Então foi atribuído um mesmo valor para o custo fixo de todas as facilidades em cada problema. A variação desse valor deu origem a diversos outros problemas.

Em nosso trabalho, utilizamos exatamente as mesmas variações que Erlenkotter [19] e Al-Sultan e Al-Fawzan[2] propuseram aos problemas de tamanho 33×33 e 57×57 de Karg e Thompson. Os resultados computacionais da aplicação do nosso algoritmo e do algoritmo de Galvão e Raggi a esses problemas se encontram nas tabelas 4.12 e 4.13.

Nesses 2 grupos de problemas, verifica-se que a atribuição de valores que foi feita aos custos fixos das facilidades conseguiu gerar problemas em que as facilidades abertas estão em maior número na solução e outros em que as facilidades fechadas são mais numerosas. Isso significativamente aumenta as chances de nossa Heurística de Soma de Deltas incorrer em erros, quando não devidamente parametrizada, o que poderia comprometer a qualidade da

Custo fixo	S*	SH	Erro	Total	TH	TG
184	6024	6024	0,00	31/02	0,00	0,01
295	8673	8673	0,00	19/14	0,00	0,01
500	11267	11287	0,18	11/22	0,02	0,02
1000	14832	14832	0,00	06/27	0,01	0,01
1500	17832	18227	2,22	05/28	0,01	0,01
2000	20363	20625	1,29	04/29	0,00	0,02
2500	22127	22127	0,00	03/30	0,00	0,01
3000	23474	23474	0,00	02/31	0,00	0,01
4000	25474	25474	0,00	02/31	0,00	0,00
5000	27474	27474	0,00	02/31	0,01	0,01
Média	—	—	**0,37**	—	**0,00**	**0,01**

Tabela 4.12: Problemas de Karg e Thompson (33×33)

Custo fixo	S*	SH	Erro	Total	TH	TG
50	2821	2821	0,00	55/02	0,00	0,00
200	9142	9142	0,00	30/27	0,05	0,08
500	15261	15324	0,41	12/45	0,03	0,05
1000	20307	20307	0,00	09/48	0,02	0,02
1500	23943	23943	0,00	07/50	0,05	0,03
2000	27222	27522	1,10	05/52	0,04	0,04
2500	30022	30022	0,00	05/52	0,04	0,03
3000	32136	32559	1,32	04/53	0,02	0,04
4000	35547	35617	0,20	03/54	0,02	0,05
5000	38547	38742	0,51	02/55	0,01	0,04
Média	—	—	**0,35**	—	**0,03**	**0,04**

Tabela 4.13: Problemas de Karg e Thompson (57×57).

solução. Mas, apesar disso, tivemos uma solução de boa qualidade: o máximo erro registrado foi de $2,2\%$ e, em cada grupo de problemas, a média desses erros não foi superior a $0,37\%$.

Observamos que os problemas em que não atingimos o ótimo são do tipo em que o número de facilidades fechadas supera o de facilidades abertas. Então, escolhemos um valor pequeno para β (0,01) e executamos novamente esses problemas. Vejamos os resultados nas tabelas 4.14 e 4.15. Percebemos

que dos 8 problemas reexecutados, somente 1 deles obteve uma solução pior, 2 deles mantiveram a mesma solução e os 5 restantes melhoraram a solução, sendo que 3 destes atingiram o ótimo.

Custo fixo	S*	SH	Erro	Total	TH
500	11267	11411	1,28	10/23	0,01
1500	17832	17832	0,00	06/27	0,02
2000	20363	20363	0,00	05/28	0,02

Tabela 4.14: Problemas de Karg e Thompson (33 × 33) - ($\beta = 0,01$)

Custo fixo	S*	SH	Erro	Total	TH
500	15261	15301	0,26	13/44	0,10
2000	27222	27522	1,10	05/52	0,08
3000	32136	32136	0,00	04/53	0,07
4000	35547	35617	0,20	03/54	0,06
5000	38547	38623	0,51	02/55	0,05

Tabela 4.15: Problemas de Karg e Thompson (57 × 57) - ($\beta = 0,01$).

Em termos de tempos computacionais, para $\beta = 1$, obtivemos uma discreta superioridade (0,01s em média) em ambos os grupos de problemas. Em apenas dois problemas tivemos tempos piores que os do algoritmo de Galvão e Raggi. Em todos os demais problemas, nossos tempos foram iguais ou melhores. Os problemas que reexecutamos utilizando $\beta = 0,01$ sofreram um aumento considerável em seus tempos computacionais. Isso se deve ao fato de ter havido nesses casos uma maior participação das heurísticas das fases 2 e 3 na obtenção da solução, o que tornou a execução do nosso algoritmo mais demorada.

4.4 Problemas de p-medianas do Beasley

Os problemas de Karg e Thompson são problemas de pequeno porte. Visando observar o comportamento do nosso algoritmo frente ao algoritmo de Galvão e Raggi diante de instâncias de grande porte, resolvemos adaptar ao nosso problema um conjunto de 40 problemas, disponíveis também na *OR-Library* [6] de J. E. Beasley. Seus tamanhos variam de 100×100 a 900×900.

São problemas formulados para o P-MED, originalmente. Para cada um desses problemas uma rede é dada. Então, para obtermos a matriz de custos de transporte a partir dessa rede, utilizamos o Algoritmo de Floyd, como foi feito em Galvão e Raggi [21] na solução de modelos em rede. Os custos fixos foram gerados aleatoriamente, dentro de 3 intervalos de valores: um intervalo de custos considerados reduzidos (em relação aos valores médios de c_{ij}) variando de 0 a 50, como veremos na Tabela 4.16, outro de custos intermediários (50 a 500), conforme observamos na Tabela 4.17 e outro de custos elevados (500 a 1000), como podemos observar na Tabela 4.18.

Problema	Tamanho	S*	SH	Erro	TH	TG
Pmed1	100 × 100	1564	1564	0,00	0,01	0,08
Pmed2	100 × 100	1463	1464	0,07	0,02	0,03
Pmed3	100 × 100	1677	1677	0,00	0,02	0,03
Pmed4	100 × 100	1701	1703	0,12	0,02	0,02
Pmed5	100 × 100	1467	1473	0,41	0,04	0,03
Pmed6	200 × 200	2300	2300	0,00	0,31	0,14
Pmed7	200 × 200	2468	2474	0,24	0,20	0,15
Pmed8	200 × 200	2542	2542	0,00	0,16	0,08
Pmed9	200 × 200	2379	2379	0,00	0,29	0,13
Pmed10	200 × 200	2034	2034	0,00	0,12	0,14
Pmed11	300 × 300	3060	3062	0,07	1,10	0,36
Pmed12	300 × 300	3058	3059	0,03	0,96	0,68
Pmed13	300 × 300	3040	3040	0,00	0,89	0,42
Pmed14	300 × 300	3147	3151	0,13	0,91	0,71
Pmed15	300 × 300	3039	3043	0,13	0,78	2,35
Pmed16	400 × 400	3366	3366	0,00	2,37	0,94
Pmed17	400 × 400	3435	3435	0,00	2,40	1,31
Pmed18	400 × 400	3693	3693	0,00	2,84	1,18
Pmed19	400 × 400	3441	3450	0,26	2,61	0,81
Pmed20	400 × 400	3625	3631	0,17	2,75	8,57
Pmed21	500 × 500	3855	3865	0,26	4,98	15,51
Pmed22	500 × 500	3954	3957	0,08	5,63	2,59
Pmed23	500 × 500	3871	3875	0,10	4,62	1,95
Pmed24	500 × 500	3762	3764	0,05	5,12	1,61
Pmed25	500 × 500	3947	3950	0,08	5,03	1,67
Pmed26	600 × 600	4239	4239	0,00	10,11	7,66
Pmed27	600 × 600	4145	4151	0,14	9,75	3,40
Pmed28	600 × 600	3964	3980	0,40	9,09	2,98
Pmed29	600 × 600	4070	4070	0,00	7,86	3,57
Pmed30	600 × 600	4354	4356	0,05	7,91	35,31
Pmed31	700 × 700	4094	4094	0,00	11,19	51,68
Pmed32	700 × 700	4306	4312	0,14	11,22	50,76
Pmed33	700 × 700	4343	4346	0,07	13,30	4,11
Pmed34	700 × 700	4343	4358	0,35	13,91	3,57
Pmed35	800 × 800	4428	4433	0,11	15,92	5,39
Pmed36	800 × 800	4799	4804	0,10	12,50	14,48
Pmed37	800 × 800	4859	4867	0,16	14,56	93,63
Pmed38	900 × 900	4791	4799	0,17	15,91	13,08
Pmed39	900 × 900	4789	4790	0,02	17,09	157,10
Pmed40	900 × 900	5182	5182	0,00	17,74	8,23
Média	—	—	—	**0,10**	**5,81**	**12,41**

Tabela 4.16: Problemas de p-medianas do Beasley - Custos fixos reduzidos

Problema	Tamanho	S*	SH	Erro	TH	TG
Pmed1	100 × 100	5517	5527	0,18	0,03	0,03
Pmed2	100 × 100	5378	5378	0,00	0,03	0,02
Pmed3	100 × 100	5689	5689	0,00	0,07	0,05
Pmed4	100 × 100	5770	5810	0,69	0,04	0,02
Pmed5	100 × 100	4680	4680	0,00	0,03	0,01
Pmed6	200 × 200	6638	6638	0,00	0,27	0,08
Pmed7	200 × 200	6801	6801	0,00	0,19	0,08
Pmed8	200 × 200	7078	7085	0,10	0,41	0,11
Pmed9	200 × 200	6240	6252	0,19	0,21	0,13
Pmed10	200 × 200	5668	5725	1,01	0,24	0,14
Pmed11	300 × 300	7650	7650	0,00	0,72	0,27
Pmed12	300 × 300	7732	7743	0,14	0,64	0,61
Pmed13	300 × 300	7690	7772	1,07	0,53	0,33
Pmed14	300 × 300	7983	8036	0,66	0,84	0,85
Pmed15	300 × 300	7405	7411	0,08	0,64	0,33
Pmed16	400 × 400	7750	7775	0,32	1,11	2,57
Pmed17	400 × 400	7674	7674	0,00	1,47	0,52
Pmed18	400 × 400	8571	8571	0,00	1,52	1,59
Pmed19	400 × 400	7958	7958	0,00	1,00	0,98
Pmed20	400 × 400	8310	8310	0,00	1,63	0,43
Pmed21	500 × 500	8774	8798	0,27	2,79	1,48
Pmed22	500 × 500	9141	9225	0,92	3,58	33,13
Pmed23	500 × 500	8683	8708	0,29	2,14	2,72
Pmed24	500 × 500	8661	8809	1,71	3,46	1,44
Pmed25	500 × 500	8666	8689	0,27	1,71	1,45
Pmed26	600 × 600	9092	9103	0,12	6,37	3,82
Pmed27	600 × 600	9007	9007	0,00	3,86	14,33
Pmed28	600 × 600	8767	8806	0,44	2,95	4,26
Pmed29	600 × 600	9020	9030	0,11	5,56	4,00
Pmed30	600 × 600	9449	9486	0,39	4,73	4,39
Pmed31	700 × 700	9268	9309	0,44	8,33	4,92
Pmed32	700 × 700	9876	9912	0,36	9,83	200,25
Pmed33	700 × 700	10081	10145	0,63	8,80	10,93
Pmed34	700 × 700	9316	9333	0,18	9,42	6,74
Pmed35	800 × 800	9447	9489	0,44	12,90	9,76
Pmed36	800 × 800	10631	10639	0,08	24,45	24,85
Pmed37	800 × 800	10607	10696	0,84	19,20	143,84
Pmed38	900 × 900	9935	9941	0,06	15,15	5,61
Pmed39	900 × 900	9902	9916	0,14	9,70	4,57
Pmed40	900 × 900	11024	11054	0,27	16,24	44,87
Média				**0,31**	**4,57**	**13,41**

Tabela 4.17: Problemas de p-medianas do Beasley - Custos fixos intermediários

Problema	Tamanho	S*	SH	Erro	TH	TG
Pmed1	100 × 100	8886	8886	0,00	0,04	0,33
Pmed2	100 × 100	8803	8852	0,56	0,06	0,25
Pmed3	100 × 100	9038	9054	0,18	0,03	0,64
Pmed4	100 × 100	9471	9471	0,00	0,04	0,06
Pmed5	100 × 100	7593	7593	0,00	0,02	0,05
Pmed6	200 × 200	10769	10769	0,00	0,33	3,81
Pmed7	200 × 200	9896	9896	0,00	0,16	1,65
Pmed8	200 × 200	10850	10850	0,00	0,35	2,07
Pmed9	200 × 200	10029	10084	0,55	0,18	0,21
Pmed10	200 × 200	8479	8479	0,00	0,27	1,17
Pmed11	300 × 300	10409	10484	0,72	0,29	8,11
Pmed12	300 × 300	11118	11178	0,54	0,41	13,36
Pmed13	300 × 300	10590	10590	0,00	0,35	4,31
Pmed14	300 × 300	12075	12109	0,28	1,26	116,34
Pmed15	300 × 300	10215	10215	0,00	0,36	0,37
Pmed16	400 × 400	11080	11126	0,42	1,45	64,88
Pmed17	400 × 400	11455	11455	0,00	1,48	13,62
Pmed18	400 × 400	12309	12325	0,13	0,59	1,43
Pmed19	400 × 400	11823	11870	0,40	2,31	7,69
Pmed20	400 × 400	12676	12678	0,02	3,40	46,71
Pmed21	500 × 500	12019	12019	0,00	2,86	6,53
Pmed22	500 × 500	13090	13191	0,77	1,45	66,32
Pmed23	500 × 500	12650	12650	0,00	5,19	4,40
Pmed24	500 × 500	12508	12565	0,46	4,17	50,01
Pmed25	500 × 500	11942	11942	0,00	1,29	16,00
Pmed26	600 × 600	12799	12799	0,00	5,06	44,58
Pmed27	600 × 600	12426	12426	0,00	2,93	2,31
Pmed28	600 × 600	12587	12602	0,12	4,28	57,74
Pmed29	600 × 600	12819	12819	0,00	2,94	2,95
Pmed30	600 × 600	13895	13895	0,00	1,64	51,61
Pmed31	700 × 700	13041	13041	0,00	2,29	143,45
Pmed32	700 × 700	13989	13989	0,00	15,55	245,61
Pmed33	700 × 700	13982	13982	0,00	12,05	277,99
Pmed34	700 × 700	13377	13377	0,00	10,65	60,18
Pmed35	800 × 800	13447	13570	0,91	15,62	365,02
Pmed36	800 × 800	14646	14646	0,00	8,89	403,90
Pmed37	800 × 800	15353	15361	0,05	12,85	126,29
Pmed38	900 × 900	13972	13972	0,00	24,45	541,56
Pmed39	900 × 900	13931	13931	0,00	22,56	310,20
Pmed40	900 × 900	15258	15258	0,00	16,52	100,06
Média	—	—	—	**0,15**	**4,67**	**79,09**

Tabela 4.18: Problemas de p-medianas do Beasley - Custos fixos elevados

Em termos de qualidade da solução, apesar de usarmos na fase 3 a Heurística T2, obtivemos soluções bem próximas do ótimo. Tivemos um erro máximo de $1,71\%$ e em média os erros não superaram $0,31\%$.

Em relação aos tempos computacionais, nos problemas de custos fixos reduzidos, obtivemos melhores tempos somente em 13 problemas. Em apenas 1 problema obtivemos tempos iguais e nos demais 26 problemas o algoritmo de Galvão e Raggi obteve a melhor performance.

Já nos problemas de custos fixos intermediários, a superioridade do algoritmo de Galvão e Raggi foi um pouco maior: em 27 problemas o algoritmo deles levou menos tempo que o nosso. Novamente, em um único problema obtivemos tempos iguais e nos restantes 12 problemas nosso algoritmo levou menos tempo.

Entretanto, nos problemas de custos fixos elevados, o algoritmo heurístico foi superado em apenas 2 problemas obtendo tempos melhores em todos os demais, com diferenças que chegaram a superar 8 minutos de execução, para um mesmo problema.

Observa-se que nos dois primeiros grupos de problemas, a média de tempos computacionais do nosso algoritmo foi melhor. O significado disso é que, para alguns problemas, especialmente os de maior porte, o algoritmo de Galvão e Raggi levou muito tempo, o que acarretou o aumento dessa média.

4.5 Problemas não-euclideanos gerados aleatoriamente

Os problemas não-euclideanos, de um modo geral, são problemas mais difíceis de serem resolvidos em relação aos euclideanos de mesmo porte.

Considerando os custos de transporte como distâncias não-euclideanas entre facilidades e demanda, criamos um gerador aleatório de SPLP não-euclideanos. Ele possui uma estrutura bastante simples: tem-se como entrada as dimensões do problema, os limites inferiores e superiores para geração dos custos fixos e os limites inferiores e superiores para geração dos custos variáveis. Então são gerados aleatoriamente o vetor de custos fixos e a matriz de custos variáveis, cujos elementos obedecem aos intervalos predefinidos.

Aplicamos então nosso algoritmo e o de Galvão e Raggi ao conjunto de problemas produzido pelo gerador. Para que o algoritmo de Galvão e Raggi pudesse ser utilizado, obrigamos o gerador a respeitar todas as características dos problemas euclideanos, exceto a desigualdade triangular.

São ao todo 25 problemas divididos em cinco grupos, de acordo com o intervalo de geração dos custos fixos e estabelecemos arbitrariamente o intervalo de geração dos custos variáveis de 10 a 1000. As características dos custos fixos de cada grupo estão listadas na Tabela 4.19.

Os resultados computacionais referentes a solução dos problemas dos grupos 1, 2, 3, 4 e 5 se encontram, respectivamente, nas tabelas 4.20, 4.21, 4.22, 4.23 e 4.24.

Grupo	Descrição dos custos fixos	Intervalo de geração dos custos fixos
1	muito reduzidos	1 a 20
2	reduzidos	1 a 500
3	intermediários	300 a 800
4	elevados	500 a 1000
5	muito elevados	1000 a 1020

Tabela 4.19: Características dos custos fixos dos grupos de problemas não-euclideanos gerados aleatoriamente

Problema	S*	SH	Erro	TH	TG
50 × 50	488	488	0,00	0,00	0,01
100 × 100	914	914	0,00	0,01	0,02
150 × 150	1438	1439	0,07	0,01	0,04
200 × 200	1860	1860	0,00	0,05	0,55
400 × 400	3504	3505	0,03	0,46	1,58
600 × 600	5054	5055	0,02	0,93	4,60
800 × 800	6621	6621	0,00	1,72	8,05
Média	—	—	**0,01**	**0,45**	**2,12**

Tabela 4.20: Problemas não-euclideanos - Grupo 1 - Custos fixos muito reduzidos

Problema	S*	SH	Erro	TH	TG
50 × 50	2864	2864	0,00	0,02	0,01
100 × 100	4618	4628	0,22	0,34	0,05
150 × 150	5536	5536	0,00	1,10	0,24
200 × 200	6495	6516	0,32	2,55	0,58
400 × 400	10025	10039	0,14	22,87	2,10
600 × 600	12418	12427	0,07	54,26	5,75
800 × 800	15627	15632	0,03	280,52	231,43
Média	—	—	**0,11**	**51,67**	**34,31**

Tabela 4.21: Problemas não-euclideanos - Grupo 2 - Custos fixos reduzidos

Problema	S*	SH	Erro	TH	TG
50 × 50	5701	5805	1,82	0,03	0,10
100 × 100	9243	9418	1,89	0,33	55,49
150 × 150	11314	11449	1,19	0,97	1033,13
200 × 200	13043	13073	0,23	2,09	2677,36
Média	—	—	**1,29**	**0,86**	**941,52**

Tabela 4.22: Problemas não-euclideanos - Grupo 3 - Custos fixos intermediários

Problema	S*	SH	Erro	TH	TG
50 × 50	7290	7319	0,40	0,02	0,45
100 × 100	11314	11549	2,08	0,34	72,21
150 × 150	13785	14020	1,70	1,04	627,78
200 × 200	16083	16111	0,17	2,06	7439,73
Média	—	—	**1,09**	**0,87**	**2035,04**

Tabela 4.23: Problemas não-euclideanos - Grupo 4 - Custos fixos elevados

Problema	S*	SH	Erro	TH	TG
50 × 50	9214	9214	0,00	0,04	1,06
100 × 100	13685	13788	0,75	0,36	38,26
150 × 150	17713	17713	0,00	0,94	5218,49
Média	—	—	**0,25**	**0,45**	**1752,60**

Tabela 4.24: Problemas não-euclideanos - Grupo 5 - Custos fixos muito elevados

Em todos os grupos obtivemos erros bastante reduzidos. O máximo erro que tivemos em um problema foi de $2,08\%$ e em média eles se mantiveram inferiores a $1,30\%$ para cada grupo de problemas.

Mais uma vez modificaremos os valores dos parâmetros associados às heurísticas visando aproveitar as características individuais dos problemas a fim de se obter melhores soluções. Nas experiências computacionais que realizamos com esses grupos de problemas não-euclideanos, observamos que muitas facilidades tiveram seus status modificados na fase 2. Então, disciplinar a abertura/fechamento de facilidades nessa fase pode melhorar a solução. Para isso, escolhemos $\alpha_{abrir}^{inicial} = 1, \alpha_{fechar}^{inicial} = 1, \epsilon_{abrir} = 0.1$ e $\epsilon_{fechar} = 0.1$. Aliado a isso, estabeleceremos um valor para β de acordo com a proporção entre as facilidades abertas e fechadas. Executamos novamente os 17 problemas em que não tínhamos atingido o ótimo. Os resultados se encontram na Tabela 4.25.

Observamos que desses 17 problemas, 13 deles melhoraram suas soluções sendo que 10 deles atingiram o valor ótimo. Apenas 2 problemas não alteraram suas soluções e 2 deles pioraram.

Com relação aos tempos computacionais, de um modo geral, conseguimos uma vantagem bastante significativa em relação ao algoritmo de Galvão e Raggi.

Obtivemos tempos menores na solução de todos os problemas do grupo 1. No maior dos problemas obtivemos uma diferença de pouco mais de $6s$.

O grupo 2 foi o único em que não tivemos o melhor desempenho em termos de tempos computacionais. Levamos mais tempo em todos os 7 proble-

Grupo	Problema	β	S*	SH	Erro	TH
1	150 × 150	1,50	1438	1439	0,07	0,02
1	400 × 400	1,50	3504	3504	0,00	0,51
1	600 × 600	1,50	5054	5054	0,00	1,10
2	100 × 100	0,10	4618	4618	0,00	0,23
2	200 × 200	0,10	6495	6495	0,00	4,98
2	400 × 400	0,10	10025	10025	0,00	39,15
2	600 × 600	0,10	12418	12426	0,06	184,12
2	800 × 800	0,10	15627	15636	0,06	482,21
3	50 × 50	0,10	5701	5805	1,82	0,05
3	100 × 100	0,10	9243	9352	1,18	0,54
3	150 × 150	0,10	11314	11404	0,80	1,01
3	200 × 200	0,10	13043	13414	2,84	5,00
4	50 × 50	0,10	7290	7290	0,00	0,05
4	100 × 100	0,10	11314	11314	0,00	0,32
4	150 × 150	0,10	13785	13785	0,00	1,43
4	200 × 200	0,10	16083	16083	0,00	4,54
5	100 × 100	0,05	13685	13685	0,00	0,42

Tabela 4.25: Problemas não-euclideanos - Utilização de parâmetros apropriados nas heurísticas

mas. No problema de tamanho 800 × 800 desse grupo, levamos quase 50s de execução a mais para resolvê-lo.

Porém, nos problemas do grupo 3, obtivemos as primeiras diferenças de tempos de execução bastante significativas. Destaque para os problemas de tamanho 150 × 150, o qual nosso algoritmo resolveu em $0,97s$ enquanto que o algoritmo de Galvão e Raggi levou pouco mais de 17 minutos e o problema de tamanho 200 × 200 que resolvemos em $2,09s$ enquanto o procedimento de Galvão e Raggi levou quase 45 minutos. Resolvemos ainda problemas de tamanho 400 × 400, 600 × 600 e 800 × 800 e obtivemos, respectivamente, tempos iguais a $32,98s$, $102,90s$ e $297,98s$. Testamos o algoritmo de Galvão e Raggi para o problema de tamanho 400 × 400 e após 3 horas de execução este algoritmo ainda não tinha concluído sua execução.

Obtivemos melhores tempos também em todas as instâncias do grupo 4. Vale ressaltar que no problema de tamanho 200×200, resolvemos em $2,06s$ enquanto que o algoritmo de Galvão e Raggi o resolveu em cerca de 2 horas de execução. Novamente, executamos instâncias de tamanhos 400×400, 600×600 e 800×800 e obtivemos, respectivamente, tempos iguais a $31,64s$, $165,12s$ e $408,28s$. No problema de tamanho 400×400 o algoritmo de Galvão e Raggi novamente ultrapassou a marca de 3 horas de execução sem que tivesse sido encontrada a solução para este problema.

Finalmente, no grupo 5, também resolvemos mais rapidamente todas as instâncias, com destaque para o problema de tamanho 150×150 que nosso algoritmo resolveu em quase uma hora e meia de execução a menos. Resolvemos problemas com as características desse grupo de tamanhos 200×200, 400×400, 600×600 e 800×800 e obtivemos, respectivamente, tempos iguais a $2,78s$, $34,68s$, $164,15s$ e $438,52s$. Já no problema de tamanho 200×200, o algoritmo de Galvão e Raggi não concluiu sua execução em menos de 3 horas.

Percebemos que, de uma forma geral, nosso algoritmo obteve ótimos resultados frente ao algoritmo de Galvão e Raggi em problemas de custos fixos elevados. Nos problemas não-euclideanos essa superioridade foi mais evidente. Nos problemas de custos fixos reduzidos e intermediários, nosso algoritmo foi um pouco melhor nos problemas de Karg e Thompson, pior na maior parte dos problemas de p-medianas do Beasley e melhor em dois dos três grupos destes tipos nos problemas não-euclideanos.

Capítulo 5

Conclusão e Trabalhos Futuros

A literatura a respeito dos problemas de localização é bastante vasta e diversificada. Procuramos, através desse trabalho, adicionar às contribuições na solução do SPLP, a combinação entre testes de redução e heurísticas ADD/DROP.

Fizemos uma implementação desses fundamentos e os submetemos a uma série de testes computacionais e comparamos os resultados obtidos aos da implementação do algoritmo de Galvão e Raggi.

Observamos que nas instâncias testadas tivemos soluções de boa qualidade. Atingimos o valor ótimo em 73,33% dos problemas testados (em 242 dos 330 problemas) e, tomando-se o erro médio de cada grupo de problemas testado, o máximo desses valores foi de $1,29\%$. Em cada problema individualmente testado, o máximo erro que obtivemos foi de $8,49\%$.

Em termos de tempos computacionais, obtivemos nas instâncias euclideanas tempos próximos aos do algoritmo de Galvão e Raggi, o que não

representa uma vantagem em relação ao algoritmo deles pois nosso procedimento é heurístico e o deles é exato. Entretanto, nas instâncias não-euclideanas testadas, conseguimos uma boa vantagem na maior parte dos problemas testados, com diferenças que chegaram a superar em 2 horas de execução, para um mesmo problema.

Como sugestões para trabalhos futuros, seguem alguns tópicos que podem ser mais explorados:

- Introduzir rotinas de perturbação da solução obtida, forçando a introdução e/ou a remoção de uma ou mais facilidades na solução obtida, reexecutando os testes de redução e/ou heurísticas e verificando se são obtidas melhoras.

- Nas fases 2 e 3 somente retiramos facilidades de $\overline{K}_0$ e $\overline{K}_1$, transformando seu status de temporário para definitivo. Estes fatos facilitam a convergência do método, o que diminui o espaço para possíveis melhorias. Talvez fosse interessante continuar considerando as facilidades que tiveram seus status trocados nessas fases ainda como temporários, para que continuasse a existir a possibilidade de trocas futuras. As regras de parada relativas a inexistência de elementos em $\overline{K}_0$ e $\overline{K}_1$ nessas fases seriam reformuladas, referindo-se agora à impossibilidade de ganho no valor da função objetivo, o que garantiria a convergência.

- Tentar estabelecer de forma automática ou semi-automática a atribuição dos parâmetros associados às heurísticas do algoritmo através de um pré-processamento dos problemas. Esse procedimento poderia classificar os problemas segundo um conjunto de critérios e de-

terminar de forma imediata parâmetros apropriados para usufruir dos aspectos identificados, melhorando a solução.

- Produzir uma heurística de geração de limites inferiores para o algoritmo. Assim, teríamos uma garantia da distância entre a nossa solução e a solução ótima. Ou ainda, desenvolver um algoritmo de separação e avaliação que utilize nosso algoritmo na geração dos limites superiores.

- Adaptar nossas heurísticas à solução de outros problemas não-capacitados como o P-MED e o PLNC, ou mesmo à solução do Problema de Localização Capacitado.

Bibliografia

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

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

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

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

[5] Beasley, J. E. "An algorithm for solving large capacitated warehouse location problems". *European Journal of Operational Research*, v. 33, pp. 314–325, 1988.

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

[7] Bilde, O., Krarup, J. "Sharp lower bounds and efficient algorithms for the simple plant location problem". *Annals of Discrete Mathematics*, v. 1, pp. 79–97, 1977.

[8] Bornstein, C. T., Azlan, H. B. "The use of reduction tests and simulated annealing for the capacited plant location problem". *Location Science*, v. 6, pp. 67–81, 1998.

[9] Bornstein, C. T., Valiati, D. "a branch and bound algorithm for the capacitated plant location problem based on add/drop procedures". In *ISOLDE VII - VIIIth International Symposium on Locational Decisions*, Coimbra & Estoril, Portugal, 1999.

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

[11] Chhajed, D., Francis, R. L., Lowe, T. J. "Contributions of Operations Research to Location Analysis". *Location Science*, n. 1, pp. 263–287, 1993.

[12] Conn, A. R., Cornuéjols, G. "A projection method for the uncapacitated facility location problem". *Mathematical Programming*, v. 46, pp. 273–298, 1990.

[13] Cooper, L. "Heuristic methods for location-allocation problems". *SIAM Rev.*, v. 6, n. 1, pp. 37–53, 1964.

[14] Cornuéjols, G., Fisher, M. L., Nemhauser, G. L. "Location of banks accounts to optimize float: an analytic study of exact and approximate algorithms". *Management Science*, n. 23, pp. 789–810, 1977.

[15] Cornuéjols, G., Sridharan, R., Thizy, J. M. "a comparison of heuristics and relaxations for the capacitated plant location problem". *European Journal of Operational Research*, v. 50, pp. 280–297, 1991.

[16] Cornuéjols, G., Thizy, J. M. "A primal approach to the simple plant location problem". *SIAM Journal of algebraic and discrete methods*, v. 3, n. 4, pp. 504–510, 1982.

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

[18] Efroymson, M. A., Ray, T. L. "A branch-and-bound algorithm for plant location". *Operations Research*, v. 14, pp. 361–368, 1966.

[19] Erlenkotter, D. "A dual-based procedure for uncapacitated location problems". *Operations Research*, v. 26, n. 6, pp. 992–1009, 1978.

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

[21] Galvão, R. D., Raggi, L. A. "A method for solving to optimality uncapacitated location problems". *Annals of Operations Research*, v. 18, pp. 225–244, 1989.

[22] Gao, L. L., Robinson, E. P. "Uncapacitated facility location: general solution procedure and computational experience". *European Journal of Operational Research*, v. 76, pp. 410–427, 1994.

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

[24] Garfinkel, R. S., Neebe, A. W., Rao, M. R. "An algorithm for the m-median plant location problem". *Transportation Science*, v. 8, pp. 217–236, 1974.

[25] Guignard, M. "A Lagrangian dual ascent algorithm for simple plant location problems". *European Journal of Operational Research*, v. 35, pp. 193–200, 1977.

[26] Guignard, M., Spielberg, K. "Algorithms for exploiting the structure of the simple plant location problem". *Annals of Discrete Mathematics*, v. 1, pp. 247–271, 1977.

[27] Handler, G. Y., Mirchandani, P. B. *Location on networks.* MIT Press, Cambridge, 1979.

[28] Hurter, A. P., Martinich, J. S. *Facility Location and the Theory of Production.* Kluwer, Boston, 1989.

[29] Jacobsen, S. K. "Heuristics for the capacitated plant location model". *Mathematical Programming*, v. 79, pp. 369–396, 1997.

[30] Jain, K., Vazirani, V. V. "Primal-Dual Approximation Algorithms for Metric Facility Location and k-Median Problems". In *IEEE Symposium on Foundations of Computer Science*, pp. 2–13, 1999.

[31] Karg, R. L., Thompson, G. L. "A heuristic approach to solving travelling salesman problem". *Management Science*, v. 10, n. 2, pp. 225–248, 1964.

[32] Khumawala, B. M. "An efficient branch and bound algorithm for the warehouse location problem". *Management Science*, v. 18, n. 6, pp. 718–731, 1972.

[33] Kolen, A. "Solving covering problems and the uncapacitated plant location". *European Journal of Operational Research*, v. 12, pp. 266–278, 1983.

[34] Krarup, J., Bilde, O. *Plant location, Set Covering and Economic Lot Size: An O(mn) Algorithm for Structural Problems.* Technical report, 1977.

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

[36] Kratica, J., Tošic, D., Filipović, V., Ljubić, I. "Solving the Simple Plant Location Problem by Genetic Algorithm". *RAIRO Operations Research*, v. 35, pp. 127–142, 2001.

[37] Körkel, M. "On exact solution of large-scale simple plant location problems". *European Journal of Operational Research*, v. 39, pp. 157–173, 1989.

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

[39] Labbe, M., Peeters, D., Thisse, J. F. "location on networks". In Ball, M. O. (editor), *Network routing.* North-Holland, Amsterdam, 1995.

[40] Lenstra, J. K., Kan, A. H. G. Rinnooy. "Complexity of packing, covering and partitioning problems". In Schrijver, A. (editor), *Packing and Covering Combinatorics*, pp. 257–291. Mathematical Centre Tracts 106, Mathematical Centre, Amsterdam, 1979.

[41] Love, R. F., Morris, J. G., Wesolowsky, G. O. *Facility Location: Models and Methods.* North-Holland, New York, 1988.

[42] Magnanti, T. L., Wong, R. T. *Accelerating Benders Decomposition for Network Design.* Technical report, Massachusetts Institute of Technology, 1977.

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

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

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

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

[47] Mirchandani, P. B., Francis, R. L. *Discrete Location Theory.* Wiley, New York, 1990.

[48] Nemhauser, G. L., Wolsey, L. A., Fisher, M. L. "An analysis of approximations for maximizing submodular set functions - I". *Mathematical Programming*, v. 14, pp. 265–294, 1978.

[49] Neto, M. B. Campêlo, Bornstein, C. T. "ADD/DROP procedures for the Capacitated Plant Location Problem". *Discrete Applied Mathematics.* artigo submetido.

[50] Plastria, F. "continuous location problems". In Drezner, Z. (editor), *Facility location.* Springer Verlag, Berlim, 1995.

[51] Rapp, Y. "Planning of exchange locations and boundaries". *Ericsson Technics*, v. 2, pp. 1–22, 1962.

[52] Revelle, C., Marks, D., Liebman, J. C. "An analysis of Private and Public Sector Location Models". *Management Science*, v. 16, pp. 692–707, 1970.

[53] Revelle, C. S., Laporte, G. "The plant location problem: new models and research prospects". *Operations Research*, v. 44, n. 6, pp. 864–874, 1996.

[54] Schrage, L. "Implicit representation of variable upper bounds in linear programming". *Mathematical Programming Study*, v. 4, pp. 118–132, 1975.

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

[56] Spielberg, K. "Algorithms for the simple plant location problem with side constraints". *Operations Research*, v. 17, pp. 85–111, 1969.

[57] Tcha, D. W., Ro, H. B., Yoo, C. B. "A dual-based add heuristic for uncapacitated facility location". *Journal of the Operational Research Society*, v. 39, pp. 873–878, 1988.

[58] Teitz, M. B., Bart, P. "Heuristic methods for estimating the generalizated vertex median of a weighted graph". *Operations Research*, v. 16, n. 5, pp. 955–961, 1968.

[59] Wolsey, L. A. "Fundamental poperties of certain discrete location problems". In Thisse, J. F., Zoller, H. G. (eds), *Locational Analysis of Public Facilities*, pp. 331–355. North-Holland, 1983.

[60] Wolsey, L. A., Nemhauser, G. L. "Maximizing submodular set functions: formulations and analysis of algorithms". *Annals of Discrete Mathematics*, v. 11, pp. 279–301, 1981.