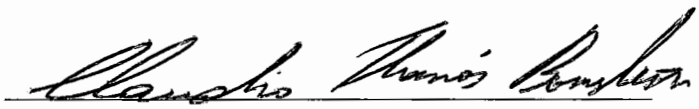
TESTES DE REDUÇÃO E HEURÍSTICAS ADD/DROP PARA O PROBLEMA DE LOCALIZAÇÃO CAPACITADO

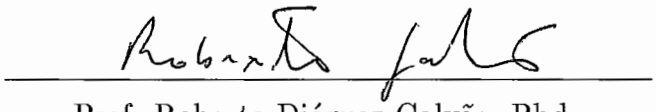
Manoel Bezerra Campêlo Neto

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 OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

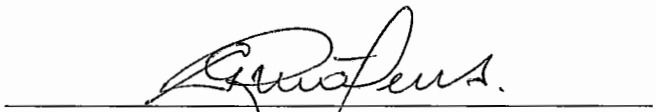
Aprovada por:



Prof. Cláudio Thomás Bornstein, D.Sc.
(Presidente)



Prof. Roberto Diéguez Galvão, Phd.



Prof. Geraldo Robson Mateus, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 1993

CAMPÊLO NETO, MANOEL BEZERRA

Testes de Redução e Heurísticas ADD/DROP para o Problema de Localização Capacitado [Rio de Janeiro] 1993

IX, 114 p., 29.7 cm, (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1993)

TESE – Universidade Federal do Rio de Janeiro, COPPE

1. Localização Capacitada.
2. Relaxação Lagrangeana.
3. Algoritmos Heurísticos.

I. COPPE/UFRJ II. Título (série).

A meus pais, Francisco e Iolanda.
A minha madrinha, Leonor.
A meus avós, Chagas e Hilda.

AGRADECIMENTOS:

- Ao Professor Cláudio Thomás Bornstein, pela sua orientação e confiança em meu trabalho.
- À Professora Susana Scheimberg de Makler, pela amizade e pelo exemplo.
- Ao Professor Antônio Clécio Fontenele Aragão, pelo incentivo à participação no curso de mestrado.
- A Ana Lúcia L. Vitoriano, Fernando Henrique M. de Carvalho e demais colegas de trabalho da SRH/UFC, pelo apoio e interesse em minha liberação para o curso.
- A Ana Paula e Cláudia, pela atitude solicita e pela eficiência.
- A Socorro Holanda, pela acolhida inicial.
- Aos companheiros de todos os dias, muito mais que amigos, Ademir, Arlene, Marcílio, Lucídio e Washington, pela família que construímos.
- A Ana Maria, Digna, Gláucia, Lícia e Sueli, Eveline, Jorge, Renato e Irene, por tornarem esses anos inesquecíveis.
- A Denise, Lilian, Odinaldo, Marcelo e Walmir, pela cumplicidade.
- A Jaudênia, pelos bons momentos que passamos juntos.
- A minha prima-irmã, "procuradora" e sempre amiga Hilda, por tudo.
- A meus pais, avó, tios e primos, a todos e a cada um individualmente, pelo amor, pelo cuidado, pela presença constante em minha vida.

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

TESTES DE REDUÇÃO E HEURÍSTICAS ADD/DROP PARA O PROBLEMA DE LOCALIZAÇÃO CAPACITADO

Manoel Bezerra Campêlo Neto

Agosto de 1993

Orientador: Cláudio Thomás Bornstein
Programa : Engenharia de Sistemas e Computação

Neste trabalho estudou-se o problema de localização capacitado (Capacitated Warehouse Location Problem – CWLP), onde custos fixos (inerentes à instalação das facilidades) e custos variáveis (principalmente associados ao transporte destas aos consumidores) devem ser minimizados.

Avaliando-se critérios de dominância entre cada uma destas parcelas de custo, foram apresentados testes de redução e heurísticas gulosas que permitem estabelecer condições sob as quais uma facilidade deve ser aberta ou fechada. Ainda mais, foram desenvolvidas diversas aproximações (limites inferiores ou superiores) para estes testes e heurísticas, a partir de diferentes formas de relaxação lagrangeana do problema, e então promovida uma análise comparativa de sua qualidade em termos numéricos e quanto à complexidade computacional.

Propôs-se também um algoritmo para a resolução do CWLP, construído com base nesses testes e heurísticas. Visando estimar sua eficiência, foram utilizados problemas-teste padrões, dos quais se conhecem as soluções ótimas, para a comparação dos resultados obtidos. Complementarmente, foi descrito o procedimento usado para resolver as diversas variações do problema de transporte subjacente ao CWLP.

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

REDUCTION TESTS AND ADD/DROP HEURISTICS FOR THE CAPACITATED WAREHOUSE LOCATION PROBLEM

Manoel Bezerra Campêlo Neto
August, 1993

Supervisor: Cláudio Thomás Bornstein
Department: Systems Engineering and Computer Science

We study the classical capacitated warehouse location problem (CWLP), where fixed costs (related to the instalation of warehouses) and variable costs (mainly associated to transportation costs from warehouse to customers) are minimized.

Dominance criteria between each of these kind of costs establish conditions under which a warehouse should be opened or closed. As a result, reduction tests and greedy heuristics are presented. Several aproximations (lower or upper bounds) to these tests and heuristics are developed based upon different lagrangean relaxations. A comparative analysis of their numerical and computacional complexity is made.

We also propose an algorithm for the CWLP and estimate its efficiency by applying it to a set of test-problems, comparing the results. The procedure used to solve the transportation problem underlying the CWLP are described.

Índice

Capítulo I

Apresentação

A intenção deste capítulo é definir a abrangência do trabalho que se pretende desenvolver. Sua primeira parte destina-se ao conhecimento e entendimento do problema a ser explorado. A seguinte procura situar, num contexto histórico, os diversos estudos já realizados sobre o tema, analisando os resultados então conseguidos. Por último, os objetivos do projeto são apresentados, ao mesmo tempo em que é discutida a organização de todo o texto.

I.1 Formulação do Problema

Considere a seguinte situação: uma empresa de produtos manufaturados planeja instalar, em pontos de sua área de atuação, armazéns (facilidades), cujas capacidades de estocagem permitam atender a seus clientes. E é claro: deseja fazê-lo minimizando os custos totais envolvidos, quais sejam, de instalação/manutenção dos armazéns e de transporte aos consumidores.

A descrição acima especifica uma circunstância que se enquadra exatamente no contexto do problema alvo deste trabalho: a localização de facilidades capacitadas (CWLP – Capacitated Warehouse Location Problem), mais genericamente apresentado a seguir.

Seja $I = \{1, \ldots, m\}$ o conjunto das possíveis localizações das facilidades e $J = \{1, \ldots, n\}$ o conjunto dos centros de consumo. Obviamente, cada consumidor $j \in J$ possui uma demanda b_j a atender. Por outro lado, cada facilidade $i \in I$ apresenta uma limitação a_i em sua capacidade máxima de fornecimento (armazenamento) e um custo fixo f_i, inerente a sua própria instalação. Considere ainda o custo unitário c_{ij} de atendimento (transporte) do consumidor j pelo armazém i.

A resolução do CWLP corresponde, então, à determinação de um subconjunto $S \subseteq I$ de facilidades, de forma que o custo total, incluindo custos fixos e variáveis, seja minimizado.

Seguindo a notação adotada, este problema pode ser formulado matematicamente através do seguinte modelo inteiro-misto:

minimizar

$$\sum_{i \in I} \sum_{j \in J} c_{ij}\, x_{ij} \;+\; \sum_{i \in I} f_i\, y_i \tag{I.1}$$

sujeito a

$$\sum_{j \in J} x_{ij} \;\leq\; a_i\, y_i \;\; \forall i \in I \tag{I.2}$$

$$\sum_{i \in I} x_{ij} \;=\; b_j \;\; \forall j \in J \tag{I.3}$$

$$x_{ij} \;\geq\; 0 \;\; \forall i \in I,\; \forall j \in J \tag{I.4}$$

$$y_i \;\in\; \{0,1\} \;\; \forall i \in I \tag{I.5}$$

onde

x_{ij} = demanda do consumidor j abastecida pelo armazém i

y_i = $\begin{cases} 1, \text{ se a facilidade } i \text{ for instalada} \\ 0, \text{ caso contrario} \end{cases}$

Na restrição (I.2) garantimos a observância aos limites de capacidade dos armazéns e, em (I.3), o atendimento total dos centros de demanda. Na função objetivo (I.1), as parcelas de contribuições dos custos fixos e variáveis estão caracterizadas.

Observe também como as consequências decorrentes das decisões de abrir ($y_i = 1$) ou fechar ($y_i = 0$) uma facilidade são refletidas pelo modelo.

Ativando-se um armazém , por exemplo, criam-se novas possibilidades de atendimento aos consumidores, ou seja, novos caminhos, o que pode contribuir para a redução dos custos de transporte. Por outro lado, estabelece-se um incremento nos custos fixos. Análise semelhante, porém de resultados opostos, pode ser realizada para o caso de se desativar uma facilidade.

Em ambos os casos, os efeitos sobre os custos fixos estão explicitamente caracterizados em (I.1). Já as variações nos custos de transporte, não diretamente visualizadas, são fruto da expansão ou redução da região viável do problema, provocada quando, em (I.2), ajustamos a disponibilidade de cada armazém.

Note que, em cada decisão, são esperadas contribuições contrárias para as parcelas de custos de transporte e custos fixos. Ou seja, a decisão de abrir (fechar) uma facilidade implica o crescimento (redução) dos últimos e pode resultar na redução (crescimento) dos primeiros. Ou ainda, a minimização dos custos de tranporte leva à ativação de muitos armazéns; a minimização dos custos fixos, à instalação de poucos. A busca de um equilíbrio nesta situação conflituosa sugere um caminho para a resolução do CLWP, que exploraremos posteriormente.

Voltando à formulação apresentada, embora seja esta a que mais se recorre para descrever o CWLP, outras frequentemente aparecem na literatura. As principais diferenças aparecem no conjunto de restrições e visam, geralmente, a abordar situações particulares ou permitir a aplicação de métodos específicos de resolução.

Para a primeira abordagem, podem-se citar restrições quanto ao número máximo e mínimo de facilidades abertas ($N_L \leq \sum_{i \in I} y_i \leq N_U$), quanto ao fluxo entre centros de oferta e demanda ($L_{ij} \leq x_{ij} \leq H_{ij}$) ou ainda quanto ao limite mínimo de fornecimento de um armazém ($\sum_{j \in J} x_{ij} \geq p_i y_i$). Para a segunda, por exemplo quando se usam relaxações para resolver o problema, restrições redundantes (tais como $x_{ij} \leq \min\{a_i, b_j\} \quad \forall i \in I, \forall j \in J$) são comumente acrescidas ao modelo.

No que diz respeito aos custos, muitas vezes a parcela variável pode incluir custos operacionais de cada facilidade, que são função $g_i(\sum_{j \in J} x_{ij})$ dos fluxos que nela se originam, isto é, da sua verdadeira dimensão (aquela capacidade efetivamente usada para atendimento às demandas).

Estas funções são, em geral, lineares, para problemas de pequeno porte, ou côncavas, quando se considera economia de escala.

No primeiro caso, $g_i(\sum_{j \in J} x_{ij}) = k_i \ \sum_{j \in J} x_{ij}$, os custos operacionais podem ser incorporados aos de transporte, adicionando-se a constante de proporcionalidade k_i aos custos unitários $c_{ij} \ \forall i \in I, \forall j \in J$.

No segundo, gera-se o modelo a seguir, semelhante ao anterior, e que pode ser considerado uma generalização daquele.

minimizar

$$\sum_{i \in I} \sum_{j \in J} c_{ij}\, x_{ij} \; + \; \sum_{i \in I} g_i(\sum_{j \in J} x_{ij} \;) \; + \; \sum_{i \in I} f_i\, y_i \tag{I.6}$$

sujeito a

$$\sum_{j \in J} x_{ij} \;\; \leq \;\; a_i\, y_i \;\; \forall i \in I \tag{I.7}$$

$$\sum_{i \in I} x_{ij} \;\; = \;\; b_j \;\; \forall j \in J \tag{I.8}$$

$$x_{ij} \;\; \geq \;\; 0 \;\; \forall i \in I, \; \forall j \in J \tag{I.9}$$

$$y_i \;\; \in \;\; \{0,1\} \;\; \forall i \in I \tag{I.10}$$

onde

$$x_{ij} \;\; = \;\; \text{demanda do consumidor } j \text{ abastecida pelo armazém } i$$

$$y_i \;\; = \;\; \begin{cases} 1, \text{ se a facilidade } i \text{ for instalada} \\ 0, \text{ caso contrario} \end{cases}$$

Neste trabalho só exploraremos, entretanto, o problema linear, e usaremos muitas vezes, num abuso de linguagem, a expressão custos de transporte como sinônimo de custos variáveis. Além disso, continuaremos referenciando, indistintamente, facilidades, centros de oferta, armazéns; bem como consumidores, centros de consumo, centros de demanda.

I.2 Histórico

O CWLP é um problema de otimização combinatória NP-completo, que corresponde a minimização de uma função côncava (linear) sobre um conjunto compacto e convexo (de restrições lineares). Logo é sabido que o mínimo sempre ocorrerá em um dos extremos (vértices) do conjunto de soluções viáveis.

Uma completa enumeração destes pontos extremos é, todavia, impraticável para a maioria dos problemas. Várias estratégias têm sido consideradas no sentido de selecionar apenas alguns vértices no caminho à procura da solução ótima.

Muitos pesquisadores têm trabalhado neste problema e diversos algoritmos, na sua maioria de natureza combinatória, vêm sendo apresentados.

De acordo com as técnicas empregadas, estes algorimos podem ser classificados em três grupos básicos: de decomposição, de enumeração e heurísticos.

Os algoritmos de decomposição, em geral, partem da análise da estrutura primal do problema e, frequentemente, aplicam o método de particionamento de Benders. É o que fez Geoffrion e Graves[27], para resolver ploblemas multi-produto. Já Van Roy[51], motivado pelos métodos duais de Galvão[24], Erlenkotter[20] e Guignard e Spielberg[31], explorou em seu algoritmo ("Cross Decomposition Algorithm"), ambas as estruturas primal e dual, usando simultaneamente, Benders e geração de colunas para resolver problemas de até 100 facilidades por 200 consumidores.

Os procedimentos de enumeração, na sua maioria do tipo "branch-and-bound", recorrem a testes de viabililidade/otimalidade para reduzir a árvore de busca. Este é o caso dos algoritmos propostos por Davis e Ray[17], Sá[48], Ellwein e Gray[19] e Akinc e Khumawala[3].

Na abordagem destes autores, os procedimentos de pesquisa de seus algoritmos usam limites fornecidos por relaxações lineares (para formulações linear-mistas dos problemas), conjugados a testes de redução que se baseiam em critérios para abrir ou fechar uma facilidade.

Akinc e Khumawala, por exemplo, apresentaram um eficiente conjunto de podas, que permite decidir, a priori, sobre o estado final de armazéns, e o empregaram em um algoritmo para resolver problemas-teste padrões, formalizados por Kuehn e Hamburger[38], com até 25 facilidades e 50 centros de demanda.

Nos últimos anos, vários artigos que aplicam relaxação lagrangeana ao problema, para a determinação de limites inferiores, têm sido divulgados. Um dos primeiros foi o de Geoffrion e McBride[28], cujo trabalho merece destaque pelo aspecto teórico. Nauss[45] e Christofides e Beasley[15] também apresentaram trabalhos nesta linha, obtendo resultados superiores a Akinc e Khumawala para os mesmos problemas-teste. Além disso, a aplicação de novos testes de redução possibilitaram a Christofides e Beasley testar problemas da ordem de 35 x 100.

Fundamentados também em relaxação lagrangeana, Guinard, Spielberg e Kim[32] desenvolveram um limite inferior para o problema e apresentaram resultados computacionais para casos que envolviam até 20 facilidades e 35 centros de consumo.

Mais recentemente Beasley[10] propôs um novo algoritmo, baseado em uma diferente formulação inteiro-mista. Este segundo trabalho mostrou-se teórica e praticamente superior ao primeiro. Além dos problemas de Kuhen e Hamburger, outros de maiores dimensões – 100x1000 e 500x1000 –, gerados randomicamente, foram resolvidos.

Quanto aos métodos heurísticos, sua preocupação principal é com a relação custo/benefício, ou mais especificamente, com a relação esforço computacional/qualidade da solução, que se torna importante à medida que a dimensão do problema cresce. Por isso as heurísticas devem prezar pela flexibilidade e simplicidade computacional.

Apesar de não garantirem a solução ótima, tais métodos têm demonstrado grande sucesso em conseguir boas soluções. Estes resultados são fruto da contribuição de diversos pesquisadores.

Jacobsen[33] generalizou para o CWLP as heurísticas ADD (indica a melhor facilidade a abrir), de Kuehn e Hamburger[38], e DROP (aponta a mais provável a fechar), de Feldman, Lehrer e Ray[21], ambas inicialmente empregadas em problemas não capacitados. Ele também considerou modificações nestas heurísticas, que calculam apenas aproximações (limite superior ou inferior) das reduções de custo, originando com isso as variações: ADD-LO, ADD-HI (equivalente a regra do maior ômega de Khumawala[37]), DROP-LO e DROP-HI (equivalente a regra do menor delta também de Khumawala[37]).

Outras heurísticas, inicialmente usadas nos problemas de p-medianas, também foram adaptadas por Jacobsen para o CWLP. São procedimentos que visam melhorar uma solução viável já encontrada e usam os testes ADD e DROP alternadamente. A heurística ALA (Alternate Location Alocation) de Rapp[47] e Cooper[16], ou SHIFT para Kuehn e Hamburger[38], intercala uma iteração DROP e uma iteração ADD, nesta ordem. Na heurística VSM (Vertex Substitution Method), proposta por Teitz e Bart[50], a ordem contrária é considerada: primeiro uma iteração ADD, em seguida uma iteração DROP.

Domschke e Drexl[18] estenderam as heurísticas ADD de Jacobsen para abranger situações onde as capacidades dos armazéns são diferentes.

Barcelo e Casanovas[7] também apresentaram diversas heurísticas, baseadas em relaxação lagrangeana, para problemas onde cada consumidor só pode ser atendido por um único armazém.

Mateus e Bornstein[40] desenvolveram um algoritmo de duas fases: a primeira, exata, e a segunda, baseada nas heurísticas ADD e DROP. E o aplicaram aos problemas de Kuehn e Hamburger, obtendo bons resultados.

Muitos outros trabalhos que estudam o CWLP, ou instâncias dele, têm sido apresentados e merecem menção. Diferentes referências ou comentários mais extensos podem ser encontrados em Aikens[2], Mateus[39] e Beasley[10].

I.3 Proposta de Trabalho

Nos últimos anos, algoritmos exatos e eficientes para resolver o problema de localização capacitada têm sido divulgados. Apesar disso, em alguns contextos, os métodos exatos ainda são computacionalmente inviáveis, especialmente em CWLP de grande porte ou em aplicações onde CWLPs são resolvidos repetidas vezes para obter a solução de um procedimento mais geral.

A existência destes casos, onde o uso de heurísticas pode tornar-se interessante, é o que motiva a realização deste trabalho. Nosso objetivo é justamente o desenvolvimento de um algoritmo heurístico que garanta uma boa relação esforço computacional/qualidade da solução.

Como em qualquer projeto de pesquisa, toda uma etapa de preparação, para a aquisição de conhecimentos relativos ao tema em foco, é indispensável antes de se partir definitivamente para a realização da meta estabelecida. A estrutura deste texto segue de perto o processo de estudo desenvolvido até a elaboração conclusiva do algoritmo.

A partir do Capítulo 2 estudaremos várias heurísticas já apresentadas por alguns pesquisadores. Primeiramente, porém, analisaremos a aplicação de testes exatos de redução, visando a minimizar o grau de incerteza das decisões ao longo do processo de solução. Este primeiro tópico reflete uma preocupação com a qualidade da solução a ser gerada.

Já no Capítulo 3, o enfoque será o esforço computacional. Nele serão sugeridas aproximações para os cálculos de grandezas que compõem os testes e heurísticas.

A base teórica dos dois primeiros capítulos já permite o desenvolvimento do algoritmo. Antes, contudo, vamos considerar o problema de transporte: um ponto fundamental, uma vez que diversos deles terão de ser resolvidos.

Eis aí um motivo suficiente para, no Capítulo 4, investirmos na confecção de um procedimento de transporte eficiente e flexível, integrado ao algoritmo principal, que permita simplificar a aplicação dos testes e heurísticas e facilite a incorporação das mudanças de contexto por eles determinadas.

Finalmente, no Capítulo 5, passamos a exposição do algoritmo, sua estrutura e funcionamento. A seguir, para fins de avaliação de desempenho, vamos aplicá-lo a problemas-teste padrões, dos quais conhecemos a solução ótima, e então analisar os resultados obtidos.

Capítulo II

Testes de Redução e Heurísticas

O confronto entre as variações nos custos fixos e de transporte, decorrentes da decisão de abrir ou fechar uma facilidade, possibilita determinar, ou pelo menos sugerir, seu estado definitivo na solução ótima.

Deste modo, estabelecendo-se critérios de dominância entre as contribuições a cada uma destas duas parcelas de custo, podem-se formular regras (exatas ou aproximadas) úteis à construção de algoritmos para o CWLP.

Se estas regras permitem precisar o "melhor" estado (aberto/fechado) para alguma(s) facilidade(s), então funcionam como um conjunto de cortes ao conjunto de soluções viáveis: a fixação, em 0 ou 1, de algumas das variáveis de estado reduz a dimensão do problema, eliminando uma série de combinações possíveis.

De outra forma, o resultado destes testes de redução, mesmo que não estabeleça exatamente a otimalidade de cada decisão sobre a implantação de um armazém, pode indicar seu estado mais provável na solução final. Com base nesta indicação, podem ser então desenvolvidas diversas heurísticas.

II.1 Introdução

Considere o CWLP

minimizar

$$\sum_{i \in I} \sum_{j \in J} c_{ij}\, x_{ij} \;+\; \sum_{i \in I} f_i\, y_i \tag{II.1}$$

sujeito a

$$\sum_{j \in J} x_{ij} \;\leq\; a_i\, y_i \;\; \forall i \in I \tag{II.2}$$

$$\sum_{i \in I} x_{ij} \;=\; b_j \;\; \forall j \in J \tag{II.3}$$

$$x_{ij} \;\geq\; 0 \;\; \forall i \in I,\, \forall j \in J \tag{II.4}$$

$$y_i \;\in\; \{0,1\} \;\; \forall i \in I \tag{II.5}$$

onde $I = \{1, \ldots, m\}$ representa o conjunto das possíveis localizações de facilidades e $J = \{1, \ldots, n\}$ os centros de demanda.

A partir de $S \subseteq I$, $S \neq \emptyset$, defina o conjunto

$$X^S = \{x \geq 0 \mid \sum_{j \in J} x_{ij} \leq a_i \;\; \forall i \in S,\; \sum_{i \in S} x_{ij} = b_j \;\; \forall j \in J\} \tag{II.6}$$

e as funções

$$W(S) \;=\; \begin{cases} \min\limits_{x \in X^S} \sum\limits_{i \in S} \sum\limits_{j \in J} c_{ij}\, x_{ij} & \text{se } X^S \neq \emptyset \\ \infty & \text{se } X^S = \emptyset \end{cases} \tag{II.7}$$

$$F(S) \;=\; \sum_{i \in S} f_i \tag{II.8}$$

onde

X^S representa o conjunto de soluções viáveis para o CWLP, em termos de fluxos para os consumidores, onde se fez $y_i = 1 \;\forall i \in S$ e $y_i = 0 \;\forall i \in I - S$;

$F(S)$ avalia o custo de implantação das facilidades em S (ativadas);

$W(S)$ corresponde à solução de um problema de transporte, a ser indentificado por $T(S)$, que relaciona as facilidades em S aos centros consumidores.

Sendo assim, o problema de localização capacitada consiste em encontrar o conjunto $S \neq \emptyset$ das facilidades abertas, com $X^S \neq \emptyset$, tal que a função $Z(S) = W(S) + F(S)$ seja minimizada.

Ambas as funções $W(.)$ e $F(.)$ são supermodulares (Fisher, Nemhauser e Wolsey[23]), conceito que generaliza a idéia de convexidade. Da mesma forma, por ser combinação linear das anteriores, $Z(.)$ também o é. Logo as três satisfazem as seguintes propriedades, enunciadas apenas em termos de $Z(.)$.

$$Z(S) - Z(S \cup i)^1 \leq Z(R) - Z(R \cup i) \;\; \forall R \subseteq S, \forall i \notin S \tag{II.9}$$

ou equivalentemente, fazendo-se $S = S \cup i$ e $R = R \cup i$,

$$Z(S - i) - Z(S) \leq Z(R - i) - Z(R) \;\; \forall R \subseteq S, \forall i \in R \tag{II.10}$$

Através destas propriedades podemos concluir sobre a variação no valor da função Z(.), com respeito a alterações em seu conjunto argumento:

1. Será tanto menor (maior) quanto maior (menor) for o conjunto;
2. Será decrescente (crescente) com a inclusão (exclusão) de elementos no conjunto.

A partir destes resultados iremos definir os testes de redução que fixarão os estados de certas facilidades. Note, entretanto, que a decisão de abrir ou fechar um armazém depende daqueles já anteriormente fixados.

Por isso é importante manter sempre a configuração corrente, o que será feito particionando-se o conjunto das facilidades $I = \{1, \ldots, m\}$ em três subconjuntos, a saber:

$$\begin{aligned} K_0 &= \{i \in I \mid y_i = 0\} &&= \text{conjunto das facilidades fixadas como fechadas} \\ K_1 &= \{i \in I \mid y_i = 1\} &&= \text{conjunto das facilidades fixadas como abertas} \\ K_2 &= I - K_0 - K_1 &&= \text{conjunto das facilidades ainda não fixadas} \end{aligned} \tag{II.11}$$

onde inicialmente temos $K_0 = K_1 = \emptyset$ e $K_2 = I$.

II.2 Teste Exato Para Abrir Facilidades

A idéia do teste é supor desativada uma facilidade ainda não fixada, para a seguir avaliar que consequências traria tal decisão, confrontando as variações nas parcelas de custo que compõem a função objetivo (II.1).

[1]Para simplicar a notação, usaremos $S \cup k$ e $S - k$ em substituição a $S \cup \{k\}$ e $S - \{k\}$.

Para isto, defina como Δ_i o acréscimo nos custos de transporte ao fecharmos o armazém i. Assim

$$\Delta_i = W(K_1 \cup K_2 - i) - W(K_1 \cup K_2) \ \forall i \in K_2 \tag{II.12}$$

Sabemos que, ao longo do processo de solução, os conjuntos K_0 e K_1 ganham elementos, à medida que K_2 os perde, pois armazéns estarão sendo aberto ou fechado.

Como K_0 cresce, o conjunto $K_1 \cup K_2 (= I - K_0)$ irá decrescer. Usando este fato em conjunção à propriedade (II.10), concluímos que Δ_i é não decrescente.

Desta forma, confrontando-se o aumento Δ_i nos custos de transporte, ocasionado pela tentativa de desativação do armazém i, ao custo fixo f_i de instalação deste, e verificando que a respectiva diferença tende a crescer, podemos formular o seguinte teste para abrir facilidades.

Teste 1: Dada uma configuração, representada pelos conjuntos K_0, K_1 e K_2, como definidos em (II.11), onde se supõem corretos os estados já fixados,

$$\text{Se } \Delta_i \geq f_i \text{ então } y_i^\star = 1 \tag{II.13}$$

onde $y_i^\star$ representa o valor de y_i na solução ótima.

Dem.: Devemos provar que $y_i^\star = 1$ para alguma solução ótima. Para tal considere $K_1^\star = \{l \mid y_l^\star = 1\}$ e suponha, ao contrário que $i \notin K_1^\star$. Então $K_1^\star \subseteq K_1 \cup K_2 - i$. Assim, pela propriedade (II.9),

$$\begin{aligned}
& Z(K_1^\star) - Z(K_1^\star \cup i) \\
& \quad \geq \ Z(K_1 \cup K_2 - i) - Z(K_1 \cup K_2) \\
& \quad = \ W(K_1 \cup K_2 - i) + F(K_1 \cup K_2 - i) - W(K_1 \cup K_2) - F(K_1 \cup K_2) \\
& \quad = \ (W(K_1 \cup K_2 - i) - W(K_1 \cup K_2)) + (F(K_1 \cup K_2 - i) - F(K_1 \cup K_2)) \\
& \quad = \ \Delta_i - f_i \\
& \quad \geq \ 0 \\
\Rightarrow \quad & Z(K_1^\star) \geq Z(K_1^\star \cup i)
\end{aligned}$$

Logo, ou $K_1^\star$ não representa a solução ótima, quando chegamos a uma contradição, ou $K_1^\star \cup i$ determina outra solução igualmente ótima. Em ambos os casos, conclui-se que podemos ter $y_i^\star = 1$.

■

Neste momento, algumas observações devem ser feitas:

1. Considerando-se que o problema original tem solução, isto é, os armazéns podem satisfazer as demandas ($\sum_{i\in I} a_i \geq \sum_{j\in J} b_j$), e observando que inicialmente faz-se $K_2 = I$, o problema de transporte $T(K_1 \cup K_2)$ tem solução finita.

2. Por outro lado, é fácil verificar se um armazém pode ser desativado, ou seja, se sua capacidade de oferta não é indispensável ao atendimento da demanda. Caso esta possibilidade seja verificada, isto é, se ($\sum_{l\in K_1\cup K_2-i} a_i \geq \sum_{j\in J} b_j$), então $T(K_1 \cup K_2 - i)$ tem solução finita. Do contrário, $W(K_1 \cup K_2 - i) \rightarrow \infty$, e o armazém i deve ser aberto.

3. Dos dois itens anteriores, conclui-se que inicialmente se pode aplicar o **teste 1** (II.13) para abrir armazéns, já que o cálculo de Δ_i depende apenas da resolução de $T(K_1 \cup K_2)$ e $T(K_1 \cup K_2 - i)$.

4. Embora bastante eficaz, a aplicação deste teste a todas as facilidades ainda não fixadas requer a solução de $\mid K_2 \mid +1$ problemas de transporte, o que pode resultar computacionalmente dispendioso. Por isso, iremos desenvolver, adiante, aproximações para o cálculo de Δ_i.

II.3 Teste Exato Para Fechar Facilidades

Analogamente ao anterior, neste teste vai-se supor ativada uma facilidade, para, em seguida, avaliar as vantagens/desvantagens originadas, considerando as alterações ocorridas nos custos fixos e variáveis.

Seja então Ω_i a redução nos custos de transporte ao abrirmos o armazém i ainda não ativado, isto é,

$$\Omega_i = W(K_1) - W(K_1 \cup i) \;\; \forall i \in K_2 \tag{II.14}$$

Como K_1 é crescente, aplicando a propriedade (II.9), concluímos que Ω_i é não crescente.

Na verdade, Ω_i representa a economia obtida, considerando os custos de transporte, quando optamos por ativar o armazém i. Sabendo que esta economia é decrescente (ou pelo menos não crescente) ao longo do processo de solução, então, comparando-a com o custo de fixo implantação f_i, podemos estabelecer o seguinte teste para fechar facilidades.

Teste 2: Dada uma configuração, representada pelos conjuntos K_0, K_1 e K_2, como definidos em (II.11), onde se supõem corretos os estados já fixados,

$$\text{Se } \Omega_i \leq f_i \text{ então } y_i^\star = 0 \tag{II.15}$$

onde $y_i^\star$ representa o valor de y_i na solução ótima.

Dem.: Devemos provar que $y_i^\star = 0$ para alguma solução ótima. Para tal considere $K_1^\star = \{l \mid y_l^\star = 1\}$ e suponha, ao contrário, que $i \in K_1^\star$. Então $K_1 \cup i \subseteq K_1^\star$. Assim, usando a propriedade (II.10),

$$\begin{aligned}
& Z(K_1^\star - i) - Z(K_1^\star) \\
\leq\ & Z(K_1) - Z(K_1 \cup i) \\
=\ & W(K_1) + F(K_1) - W(K_1 \cup i) - F(K_1 \cup i) \\
=\ & (W(K_1) - W(K_1 \cup i)) + (F(K_1) - F(K_1 \cup i)) \\
=\ & \Omega_i - f_i \\
\leq\ & 0 \\
\Rightarrow\ & Z(K_1^\star - i) \leq Z(K_1^\star)
\end{aligned}$$

Logo, ou $K_1^\star$ não representa a solução ótima, quando chegamos a uma contradição, ou $K_1^\star - i$ determina outra solução igualmente ótima. Em ambos os casos, conclui-se que podemos ter $y_i^\star = 0$.

■

Novamente, neste ponto, cabem algumas observações:

1. O cálculo de Ω_i deriva da solução dos problemas $T(K_1)$ e $T(K_1 \cup i)$. Logo o **teste 2** (II.15) só pode ser aplicado quando o conjunto dos armazéns já fixados como abertos atender a demanda total, ou seja, $(\sum_{l \in K_1} a_l \geq \sum_{j \in J} b_j)$. Do contrário, teríamos $X^{K_1} = \emptyset$ e pelo menos um dos problemas apresentaria solução tendendo ao infinito, o que faria o teste perder o poder de comparação das grandezas envolvidas.

2. Similarmente ao caso anterior, a resolução de $|K_2|+1$ problemas de transporte, quando da aplicação deste teste a todas as facilidades, pode demandar esforço computacional bastante considerável. Isto motiva o desenvolvimento de cálculos aproximados para Ω_i, o que faremos adiante.

II.4 Heurísticas

A esta altura pode-se pensar no seguinte procedimento para resolver o CWLP:

1. Aplica-se, primeiramente, o teste 1 (II.13); cada armazém aberto é excluído de K_2 e passa a compor K_1;
2. Executa-se, em seguida, o teste 2 (II.15); as facilidades desativadas são transferidas de K_2 para K_0.

Se, ao final, os estados de todos os armazéns estiverem decididos, muito bem: encontramos a solução ótima, já que os testes aplicados são exatos. E se isto não acontecer, ou seja, se ainda permanecerem localizações indefinidas ($K_2 \neq \emptyset$) ? Pode-se pensar, então, em voltar ao primeiro passo ...

Observe, entretanto, que a reaplicação do teste 1 só tem sentido se alguma modificação no conjunto $(K_1 \cup K_2)$ ocorrer, ou melhor, se K_2 perder elementos para K_0. Logo esta alternativa só é viável caso o passo 2 tenha, anteriormente, proporcionado a desativação de alguma facilidade.

Da mesma forma, uma nova execução do teste 2 só deve ser considerada caso alguma alteração se verifique na última aplicação do passo 1, isto é, caso alguma facilidade tenha sido aberta. Mas se nem mesmo conseguirmos aplicar o passo 2 a primeira vez, por não serem os armazéns instalados pelo teste 1 suficientes para atender a demanda ($\sum_{l \in K_1} a_l < \sum_{j \in J} b_j$) ?

Por certo estas situações ocorrerão. Bem se vê que este procedimento na forma simples em que foi apresentado não funciona! Então como resolver o problema, se os testes exatos não são suficientes? É neste ponto que podem contribuir as heurísticas.

Sua aplicação incidirá sobre os centros de oferta que não satisfaçam ao teste 1, nem ao teste 2, isto é, qualquer armazém $i \in K_2$ onde $\Delta_i < f_i$ e $\Omega_i > f_i$. Verifiquemos que existe esta possibilidade.

$$\begin{aligned} \Omega_i &= W(K_1) - W(K_1 \cup i) \\ &= W(K_1 \cup i - i) - W(K_1 \cup i) \\ &\geq^{2} W(K_1 \cup K_2 - i) - W(K_1 \cup K_2) \\ &= \Delta_i \end{aligned}$$

[2] Usando a prop. (II.10) e $(K_1 \cup i) \subseteq (K_1 \cup K_2)$

Logo podem acontecer

- $\Omega_i \geq \Delta_i \geq f_i \Rightarrow$ teste 1
- $\Delta_i \leq \Omega_i \leq f_i \Rightarrow$ teste 2
- $\Delta_i < f_i < \Omega_i \Rightarrow$ heurísticas

As heurísticas que aqui serão apresentadas correspondem a procedimentos gulosos ADD e DROP (Jacobsen[33]), baseados igualmente nos critérios de dominância entre os custos fixos e os custos de transporte.

II.4.1 Heurística para abrir

Lembrando a interpretação do teste 2, onde fechávamos o armazém cuja diferença entre a economia nos custos variáveis ao tentar abri-lo e seu custo fixo era negativa ($\Omega_i - f_i \leq 0$), parece lógico que a facilidade mais provável de ser aberta é aquela onde esta diferença seja a maior possível. Então definimos

Heurística 1:

$$\Omega_s - f_s = \max_{i \in K_2}(\Omega_i - f_i)$$
$$\text{Se } \Omega_s - f_s > 0 \text{ então } y_s = 1 \tag{II.16}$$

Como Ω_s é não crescente, não podemos garantir $\Omega_s > f_s$ até o final do processo, por isso não se trata de um teste exato.

II.4.2 Heurística para fechar

Similarmente, o significado do teste 1, no qual instalávamos o armazém cuja diferença entre o acréscimo nos custos de transporte ao tentar fechá-lo e seu custo fixo era positiva ($\Delta_i - f_i \geq 0$), nos indica a facilidade onde esta diferença seja mínima como a mais provável de ser desativada. Então estabelecemos

Heurística 2:

$$\Delta_s - f_s = \min_{i \in K_2}(\Delta_i - f_i)$$
$$\text{Se } \Delta_s - f_s < 0 \text{ então } y_s = 0 \tag{II.17}$$

Devido Δ_s ser não decrescente, a desigualdade $\Delta_s < f_s$ pode não se conservar no decorrer do processo. Desta forma não se garante a exatidão do teste.

II.5 Aproximações

A determinação dos valores exatos de Δ_i ou de Ω_i para toda facilidade ainda não fixada, requer, como já comentado, $|K_2|+1$ resoluções de problemas de transporte: um básico, $T(K_1 \cup K_2)$ ou $T(K_1)$, e mais um para cada $i \in K_2$, $T(K_1 \cup K_2 - i)$ ou $T(K_1 \cup i)$. Assim, a aplicação inicial do teste 1, por exemplo, dependeria da resolução de $(m+1)$ destes problemas.

Além disso, se um algoritmo usa repetidamente os testes, a cada mudança na configuração dos estados dos armazéns, fica fácil perceber que sua eficiência computacional ficaria seriamente comprometida.

Por isso é interessante estimar aproximações (limites inferiores e/ou superiores) destes parâmetros, que, apesar de reduzirem o poder de poda dos testes, simplificam em muito a complexidade dos cálculos.

Diferentes caminhos podem ser seguidos visando a avaliação destas aproximações. Note, entretanto, que os objetivos são antagônicos: parece esperado que a eficiência dos testes não conduza à simplicidade dos cálculos e vice-versa.

Bons resultados têm sido conseguidos adotando-se a seguinte estratégia: resolve-se exatamente o problema básico, $T(K_1 \cup K_2)$ ou $T(K_1)$, e calcula-se uma aproximação do problema modificado, $T(K_1 \cup K_2 - i)$ ou $T(K_1 \cup i)$.

II.5.1 Aproximações para Δ_i

Sabemos que $\Delta_i = W(K_1 \cup K_2 - i) - W(K_1 \cup K_2)$. Então, seguindo o método proposto, a determinação de um limite inferior Δ_i^L ou superior Δ_i^U para Δ_i é feita, respectivamente, com base no cálculo, também, de um limite inferior $W^L(K_1 \cup K_2 - i)$ ou superior $W^U(K_1 \cup K_2 - i)$ de $W(K_1 \cup K_2 - i)$. Assim,

$$\Delta_i^L = W^L(K_1 \cup K_2 - i) - W(K_1 \cup K_2) \;\; \forall i \in K_2 \qquad \text{(II.18)}$$

$$\Delta_i^U = W^U(K_1 \cup K_2 - i) - W(K_1 \cup K_2) \;\; \forall i \in K_2 \qquad \text{(II.19)}$$

Usando-se $\Delta_i^L \leq \Delta_i$, temos os seguintes casos:

- $\Delta_i^L \geq f_i \;\Rightarrow\; \Delta_i \geq \Delta_i^L \geq f_i \;\Rightarrow\; \Delta_i \geq f_i$
- $\Delta_i^L < f_i \;\Rightarrow\; \begin{cases} f_i > \Delta_i > \Delta_i^L \;\Rightarrow\; \Delta_i < f_i \\ \Delta_i > f_i > \Delta_i^L \;\Rightarrow\; \Delta_i > f_i \end{cases}$

que nos permitem estabelecer

Teste 1 modificado:

$$\text{Se } \Delta_i^L \geq f_i \text{ então } y_i^\star = 1 \tag{II.20}$$

Heurística 2 modificada:

$$\Delta_s^L - f_s = \min_{i \in K_2}(\Delta_i^L - f_i)$$

$$\text{Se } \Delta_s - f_s < 0 \text{ então } y_s = 0 \tag{II.21}$$

Observe que:

1. O **teste 1 modificado** (II.20) continua exato, entretanto perde parte do poder de poda em relação ao anterior (II.13), pela ocorrência do caso $\Delta_i > f_i > \Delta_i^L$.
2. Para aumentar a eficácia da **heurística 2 modificada** (II.21), apenas a escolha do armazém a fechar é feita em termos de Δ_i^L; na decisão definitiva, considera-se Δ_i, como na heurística original (II.17).

Já com $\Delta_i^U \geq \Delta_i$, acontecem as seguintes situações:

- $\Delta_i^U \leq f_i \Rightarrow \quad \Delta_i \leq \Delta_i^U \leq f_i \quad \Rightarrow \quad \Delta_i \leq f_i$
- $\Delta_i^U > f_i \Rightarrow \begin{cases} f_i < \Delta_i < \Delta_i^U & \Rightarrow \quad \Delta_i > f_i \\ \Delta_i < f_i < \Delta_i^U & \Rightarrow \quad \Delta_i < f_i \end{cases}$

que não favorecem a aplicação dos testes.

II.5.2 Aproximações para Ω_i

Considerando $\Omega_i = W(K_1) - W(K_1 \cup i)$, a estratégia para a avaliação de um limite superior Ω_i^U ou inferior Ω_i^L para Ω_i toma por base, respectivamente, aproximações inferiores $W^L(K_1 \cup i)$ ou superiores $W^U(K_1 \cup i)$ de $W(K_1 \cup i)$. Então,

$$\Omega_i^U = W(K_1) - W^L(K_1 \cup i) \ \forall i \in K_2 \tag{II.22}$$

$$\Omega_i^L = W(K_1) - W^U(K_1 \cup i) \ \forall i \in K_2 \tag{II.23}$$

Tomando-se $\Omega_i^U \geq \Omega_i$, ocorrem as seguintes possibilidades:

- $\Omega_i^U \leq f_i \Rightarrow \quad \Omega_i \leq \Omega_i^U \leq f_i \quad \Rightarrow \quad \Omega_i \leq f_i$
- $\Omega_i^U > f_i \Rightarrow \begin{cases} f_i < \Omega_i < \Omega_i^U & \Rightarrow \quad \Omega_i > f_i \\ \Omega_i < f_i < \Omega_i^U & \Rightarrow \quad \Omega_i < f_i \end{cases}$

a partir dos quais formulamos

Teste 2 modificado:

$$\text{Se } \Omega_i^U \leq f_i \text{ então } y_i^\star = 0 \tag{II.24}$$

Heurística 1 modificada:

$$\Omega_s^U - f_s = \max_{i \in K_2}(\Omega_i^U - f_i)$$

$$\text{Se } \Omega_s - f_s > 0 \text{ então } y_s = 1 \tag{II.25}$$

Temos, analogamente, que:

1. Por não abordar o caso $\Omega_i < f_i < \Omega_i^U$, verifica-se uma redução no poder de poda do **teste 2 modificado** (II.24) em relação ao anterior (II.15). Note, porém, que esta modificação não compromete a exatidão do teste.

2. Embora na **heurística 1 modificada** (II.25) a escolha da facilidade considere Ω_i^U, a decisão de abri-la é beseada em Ω_i, da mesma forma que na heurística original (II.16).

Quanto a $\Omega_i^L \leq \Omega_i$ as situações que possibilita seu uso, quais sejam,

- $\Omega_i^L \geq f_i \;\Rightarrow\; \Omega_i \geq \Omega_i^L \geq f_i \;\Rightarrow\; \Omega_i \geq f_i$
- $\Omega_i^L < f_i \;\Rightarrow\; \begin{cases} f_i > \Omega_i > \Omega_i^L \;\Rightarrow\; \Omega_i < f_i \\ \Omega_i > f_i > \Omega_i^L \;\Rightarrow\; \Omega_i > f_i \end{cases}$

não estimulam a aplicação dos testes.

II.5.3 Complementos

Vale lembrar que os limites para Δ_i e Ω_i podem ser desenvolvidos adotando-se outras estratégias, além das aqui expostas.

Para Δ_i, por exemplo, Akinc e Khumawala[3] apresentaram um limite inferior (que particularizaremos com a notação $\overline{\Delta}_i$), baseado na relaxação da capacidade das facilidades em $(K_1 \cup K_2 - i)$, obtendo

$$\overline{\Delta}_i = \max_{0 \leq x_{ij} \leq b_j} \{\sum_{j \in J} \delta_{ij}\, x_{ij} \mid \sum_{j \in J} x_{ij} \leq a_i\}$$

onde

$$\delta_{ij} = \min_{l \in K_1 \cup K_2 - i} \{\max(c_{lj} - c_{ij}, 0)\}$$

Observe que, para o cálculo desta aproximação, apenas um problema da mochila necessita ser resolvido, o que, neste caso, pode ser feito por inspeção. Todavia, como veremos, existem outros limites mais eficazes.

Quanto às estimativas sugeridas nas subseções anteriores, as mais utéis parecem ser Δ_i^L e Ω_i^U, pois favorecem à aplicação dos testes e heurísticas estudados. Tais estimativas são calculadas através dos limites inferiores $W^L(K_1 \cup K_2 - i)$ e $W^L(K_1 \cup i)$ respectivamente, que podem ser obtidos aplicando relaxação lagrangeana aos problemas $T(K_1 \cup K_2 - i)$ e $T(K_1 \cup i)$. Este é o assunto que abordaremos no próximo capítulo.

Capítulo III

Relaxação Lagrangeana

Existem muitos problemas (este é o caso do CWLP) cuja estrutura favorece o particionamento das restrições em grupos praticamente independentes, no sentido de que encerram em si uma limitação ou requisito a satisfazer.

A retirada de um destes grupos do conjunto de restrições gera normalmente um problema relaxado bem mais simples de ser resolvido que o problema original. Tal idéia da retirada de restrições pode adicionalmente ser embutida em uma técnica mais elaborada, a relaxação lagrangeana.

O que se faz em relaxação lagrangeana é introduzir na função objetivo, através de multiplicadores, as restrições excluídas. Desta forma, vem a se constituir num método para decompor problemas complexos em subproblemas de fácil solução, que usualmente gera bons limites para o problema original.

Sobretudo em otimização combinatória, a relaxação lagrangeana vem se tornando uma das técnicas mais extensamente utilizadas. Desde a proposição do termo, como hoje é conhecido, por Geoffrion[26] e mais recentemente a partir do trabalho de Fisher[22], onde é analisada a aplicação de relaxação lagrangeana para diversas classes de problemas linear-inteiros, o número de aplicações desta técnica vem crescendo largamente, inclusive para a solução de problemas não lineares (Michelon e Maculan[44]).

Mais especificamente em problemas de localização, o uso de relaxação lagrangeana tem-se demonstrado adequado. Tal afirmativa pode ser comprovada em recentes trabalhos como os de Galvão[25] (problemas de localização não capacitados), Mateus[42,43] (problemas capacitados de localização em redes) e Beasley[11] (heurísticas lagrangeanas para diferentes tipos de problemas de localização).

Para o CWLP, em particular, muitos trabalhos também têm sido divulgados ultimamente (Diversos deles encomtram-se destacados na seção I.2). Os resultados de tais experiências nos motivaram neste capítulo ao emprego de relaxação lagrangeana na geração de limites e heurísticas para o CWLP.

III.1 O Problema Dual

Ao final do capítulo passado concluímos que os limites inferior de Δ_i e superior de Ω_i podem ser gerados aplicando-se relaxação lagrangeana aos problemas $T(K_1 \cup K_2 - i)$ e $T(K_1 \cup i)$. Por outro lado, lembremo-nos de que os multiplicadores de Lagrange estão associados, em um problema linear, às variáveis do dual deste problema, correspondente às restrições que se pretende introduzir na função objetivo.

Desta forma, faz-se necessário estudar o dual do problema que queremos relaxar.

Considere, então, o problema $T(S)$

minimizar

$$\sum_{l \in S \subseteq I} \sum_{j \in J} c_{lj}\, x_{lj} \tag{III.1}$$

sujeito a

$$\sum_{j \in J} x_{lj} \leq a_l \quad \forall l \in S \tag{III.2}$$

$$\sum_{l \in S} x_{lj} = b_j \quad \forall j \in J \tag{III.3}$$

$$x_{lj} \geq 0 \quad \forall l \in S,\ \forall j \in J \tag{III.4}$$

Seu dual $T^D(S)$ será

maximizar

$$\sum_{l \in S} a_l\, u_l + \sum_{j \in J} b_j\, v_j \tag{III.5}$$

sujeito a

$$u_l + v_j \leq c_{lj} \quad \forall l \in S,\ \forall j \in J \tag{III.6}$$

$$u_l \leq 0 \quad \forall l \in S \tag{III.7}$$

Observe que poderíamos acrescentar ao conjunto de restrições duais uma condição de não negatividade para as variáveis v_j: reescrevendo (III.6) como $v_j \leq c_{lj} - u_l$ verifica-se facilmente que $v_j \geq 0$, pois $c_{lj} - u_l \geq 0$ e a função objetivo dual (III.5) é de maximização. De outra forma, basta observar que a restrição (III.3) pode ser expressa por uma desigualdade do tipo maior ou igual ($\geq$), o que geraria na formulação dual $v_j \geq 0$.

Pode-se demonstrar também (Erlenkotter[20]) que v_j é não crescente com o acréscimo de elementos ao conjunto S e não decrescente com a retirada. Já u_l, ao contrário, é não decrescente com a inclusão e não crescente com a exclusão de elementos de S.

Ainda para melhor compreender o significado das variáveis duais, considere a seguinte interpretação econômica para $T^D(S)$, fazendo-se

$$\begin{aligned} (-u_l) &= \text{preço unitário do produto na facilidade } l \\ v_j &= \text{preço unitário do produto após a entrega em } j \end{aligned}$$

O objetivo é dimensionar os preços na origem e após a entrega, de forma a maximizar o lucro com o transporte do produto, obedecendo à restrição ($v_j \leq c_{lj} - u_l$) de que o preço cobrado em j será inferior ao custo do próprio cliente em comprar e transportar o produto.

III.2 Propriedades

Sejam $x^\star_{lj}$ e $(u^\star_l, v^\star_j)$ as soluções ótimas, respectivamente, de $T(S)$ e $T^D(S)$. Logo pelo teorema de folgas complementares temos:

$$(a_l - \sum_{j \in J} x^\star_{lj})\, u^\star_l = 0 \quad \forall l \in S \tag{III.8}$$

$$(\sum_{l \in S} x^\star_{lj} - b_j)\, v^\star_j = 0 \quad \forall j \in J \tag{III.9}$$

$$(c_{lj} - u^\star_l - v^\star_j)\, x^\star_{ij} = 0 \quad \forall l \in S,\ \forall j \in J \tag{III.10}$$

Agora, para $K \subseteq S$, defina

$$v_j(K) = \min_{l \in K}\{c_{lj} - u^\star_l\} \quad \forall j \in J \tag{III.11}$$

$$u_l(K) = \min(0, \min_{j \in J}\{c_{lj} - v_j(K)\}) \quad \forall l \in K \tag{III.12}$$

Então podemos estabelecer as seguintes propriedades:

Propriedade 1 $v_j^\star \leq v_j(L) \leq v_j(K) \ \ \forall j \in J \ \ \forall K \subseteq L \subseteq S$

Dem.:

$$\begin{aligned} v_j^\star &\leq c_{lj} - u_l^\star \ \ \forall l \in S, \ \forall j \in J \text{ (por III.6)} \\ v_j^\star &\leq \min_{l \in L \subseteq S} \{c_{lj} - u_l^\star\} \\ &= v_j(L) \\ &\leq \min_{l \in K \subseteq L \subseteq S} \{c_{lj} - u_l^\star\} \\ &= v_j(K) \end{aligned}$$

■

Propriedade 2 *Seja* $K(j) = \{l \in K \subseteq S \mid x_{lj}^\star \neq 0\}$, *onde* $j \in J$. *Se* $K(j) \neq \emptyset$ *então* $v_j^\star = v_j(L) = v_j(K) \ \ \forall K \subseteq L \subseteq S$.

Dem.:

$$\begin{aligned} v_j^\star &= c_{lj} - u_l^\star \ \ \forall l \in K(j) \text{ (por III.10)} \\ &\geq \min_{l \in K} \{c_{lj} - u_l^\star\} \\ &= v_j(K) \\ &\geq v_j(L) \\ &\geq v_j^\star \end{aligned}$$

■

Propriedade 3 $v_j^\star = v_j(S) \ \ \forall j \in J$

Dem.: Por (III.3) tem-se que $\forall j \ \exists l$ tal que $x_{lj}^\star \neq 0$. Então fazendo $K = S$ e usando este fato na prop. (2) temos $K(j) \neq \emptyset \ \ \forall j \in J$. Logo $v_j^\star = v_j(S) \ \ \forall j \in J$.

■

Propriedade 4 $u_l^\star = u_l(K) \;\; \forall l \in K \;\; \forall K \subseteq S.$

Dem.: Primeiro provemos que $u_l^\star = u_l(S) \;\; \forall l \in S$. Seja $J(l) = \{j \in J \mid x_{lj}^\star \neq 0\}$. Duas possibilidades exclusivas podem ocorrer:

- $J(l) = \emptyset$: Por (III.8) temos $u_l^\star = 0$. Então, usando (III.6)

$$0 = u_l^\star \leq \min_{j \in J}\{c_{lj} - v_j^\star\}$$

- $J(l) \neq \emptyset$: Então

$$\begin{aligned} u_l^\star &= c_{lj} - v_j^\star \;\; \forall j \in J(l) \text{ (por III.10)} \\ &\geq \min_{j \in J}\{c_{lj} - v_j^\star\} \\ &\geq u_l^\star \text{ (por III.6)} \\ \Rightarrow u_l^\star &= \min_{j \in J}\{c_{lj} - v_j^\star\} \leq 0 \end{aligned}$$

De ambos os casos conclui-se que $u_l^\star = \min(0, \min_{j \in J}\{c_{lj} - v_j^\star\}) = u_l(S)$. Mais genericamente,

$$\begin{aligned} u_l^\star &= \min(0, \min_{j \in J}\{c_{lj} - v_j^\star\}) \;\; \forall l \in S, \; \forall j \in J \\ &\geq \min(0, \min_{j \in J}\{c_{lj} - v_j(K)\}) \;\; \forall K \subseteq S \text{ (prop. 1)} \\ &= u_l(K) \\ &= \min(0, \min_{j \in J}\{c_{lj} - \min_{k \in K}\{c_{kj} - u_k^\star\}\}) \\ &\geq \min(0, \min_{j \in J}\{c_{lj} - c_{lj} + u_l^\star\}) \\ &= \min(0, u_l^\star) \\ &= u_l^\star \end{aligned}$$

■

Propriedade 5 *Seja* $K'(j) = \{l \in K \subseteq S \mid \sum_{r \in S-K} x_{rj}^\star + x_{lj}^\star < b_j\}$, *onde* $j \in J$. *Então* $v_j(K) = v_j(K - K') \;\; \forall j \in J \;\; \forall K' \subseteq K'(j)$. *Mais particularmente, se* $x_{ij}^\star \neq b_j$ *então* $v_j(S) = v_j(S - i) \;\; \forall j \in J \;\; \forall i \in S$.

Dem.: Se $K'(j) = \emptyset$ ou $K' = \emptyset$ a demonstração é trivial. Do contrário, temos que $\exists k \in (K - K'(j)) \subseteq (K - K')$ com $x_{kj}^\star \neq 0$. Usando-se este fato na prop. (2) e a seguir a prop. (1), temos

$$v_j(K - K') = v_j^\star \leq v_j(K) \leq v_j(K - K')$$

A segunda parte é uma particularização da primeira, tomando-se $K = S$ e $K' = \{i\}$.

■

Propriedade 6 *Seja* $\overline{J}(l) = \{j \in J \mid c_{lj} - v_j^\star = u_l^\star\}$, *onde* $l \in S$. *Se* $u_l \neq 0$ *então* $\sum_{j \in \overline{J}(l)} b_j \geq a_l$

Dem.: Considere primeiramente $J(l) = \{j \in J \mid x_{lj}^\star \neq 0\}$. Por (III.10) é trivial que $J(l) \subseteq \overline{J}(l)$. Sendo $u_l \neq 0$, então

$$\begin{aligned} a_l &= \sum_{j \in J} x_{lj}^\star \text{ (por III.8)} \\ &= \sum_{j \in J(l)} x_{lj}^\star \\ &\leq \sum_{j \in J(l)} b_j \\ &\leq \sum_{j \in \overline{J}(l)} b_j \end{aligned}$$

■

Propriedade 7 *Seja* $\overline{K}(j) = \{l \in S \mid c_{lj} - u_l^\star = v_j^\star\}$, *onde* $j \in J$. *Então verifica-se* $\sum_{l \in \overline{K}(j)} a_l \geq b_j$

Dem.: Considere primeiramente $K(j) = \{l \in S \mid x_{lj}^\star \neq 0\}$. Por (III.10) é trivial que $K(j) \subseteq \overline{K}(j)$. Então

$$\begin{aligned} b_j &= \sum_{l \in S} x_{lj}^\star \text{ (por III.3)} \\ &= \sum_{l \in K(j)} x_{lj}^\star \\ &\leq \sum_{l \in K(j)} a_l \\ &\leq \sum_{l \in \overline{K}(j)} a_l \end{aligned}$$

■

Todas estas propriedades foram apresentadas com o objetivo de permitir a escolha dos multiplicadores para as relaxações e, mais ainda, para facilitar o desenvolvimento dos limites inferiores que se fará nas próximas seções.

Verifique-se antes que os parâmetros $v_j(.)$ e $u_l(.)$ podem funcionar como variáveis duais para as variações do problema $T(S)$, em combinações com $u_l^\star$ e $v_j^\star$. Na verdade, apenas $v_j(.)$ é útil, já que $u_l(.) = u_l^\star \ \forall l \in S$ (prop. 4). Este é o resultado que apresentamos a seguir.

Propriedade 8 *O par* $(u_l^\star, v_j(L))$, $l \in K \subseteq L \subseteq S$, $j \in J$ *é solução viável para* $T^D(K)$. *Em particular,* $(u_l^\star, v_j^\star)$ *e* $(u_l^\star, v_j(S-i))$, $l \in S-i$, $j \in J$ *são soluções viáveis de* $T^D(S-i)$.

Dem.: Devemos mostrar que obedecem às restrições duais, mais especificamente, (III.6), já que a restrição de não positividade (III.7) é trivialmente satisfeita. Então,

$$\begin{aligned} u_l^\star + v_j(L) &= u_l^\star + \min_{r \in L}\{c_{rj} - u_r^\star\} \\ &\leq u_l^\star + c_{lj} - u_l^\star, \quad l \in K \\ &= c_{lj} \end{aligned}$$

A segunda parte é facilmente demonstrada, fazendo-se $K = S - i$ e, $L = S$ no primeiro caso, ou $L = K$ no segundo.

■

III.3 Limites Para Δ_i

Pelo visto até aqui, já se pode perceber que a qualidade da relaxação, isto é, do limite que ela gerará, vai depender da escolha das restrições a relaxar e dos multiplicadores que as introduzirão na função objetivo.

Limites para Δ_i, derivados de relaxação lagrangeana aplicada ao problema $T(K_1 \cup K_2 - i)$, podem ser obtidos relaxando-se ambas as restrições de oferta e demanda, ou cada uma delas individualmente.

Quanto aos multiplicadores, temos como opções a solução ótima de $T^D(K_1 \cup K_2)$, de que dispomos, uma vez que $T(K_1 \cup K_2)$ tenha sido resolvido, ou algum outro conjunto a partir dela, igualmente viável para $T^D(K_1 \cup K_2 - i)$.

III.3.1 Relaxação da oferta e demanda

Considere o problema $T(K_1 \cup K_2 - i)$

minimizar

$$\sum_{l \in K_1 \cup K_2 - i} \sum_{j \in J} c_{lj}\, x_{lj} \tag{III.13}$$

sujeito a

$$\sum_{j \in J} x_{lj} \leq a_l \quad \forall l \in K_1 \cup K_2 - i \tag{III.14}$$

$$\sum_{l \in K_1 \cup K_2 - i} x_{lj} = b_j \quad \forall j \in J \tag{III.15}$$

$$x_{lj} \geq 0 \quad \forall l \in K_1 \cup K_2 - i,\ \forall j \in J \tag{III.16}$$

Relaxando os conjuntos de restrições (III.14) e (III.15), respectivamente, com α_l, $l \in K_1 \cup K_2 - i$ e β_j, $j \in J$, soluções viáveis de $T^D(K_1 \cup K_2 - i)$, obtemos o problema P^{α}_{β}

minimizar

$$\sum_{l \in K_1 \cup K_2 - i} \sum_{j \in J} (c_{lj} - \alpha_l - \beta_j)\, x_{lj} \;+\; \sum_{l \in K_1 \cup K_2 - i} \alpha_l\, a_l \;+\; \sum_{j \in J} \beta_j\, b_j$$

sujeito a

$$x_{lj} \geq 0 \quad \forall l \in K_1 \cup K_2 - i,\ \forall j \in J$$

Como (α_l, β_j) é solução viável de $T^D(K_1 \cup K_2 - i)$, temos

$$\alpha_l + \beta_j \leq c_{lj} \Longleftrightarrow c_{lj} - \alpha_l - \beta_j \geq 0$$

Logo, uma solução de P^{α}_{β} é obtida fazendo-se $x_{lj} = 0 \quad \forall l \in K_1 \cup K_2 - i,\ \forall j \in J$, com valor ótimo

$$VP^{\alpha}_{\beta} = \sum_{l \in K_1 \cup K_2 - i} a_l\, \alpha_l + \sum_{j \in J} b_j\, \beta_j$$

Desta forma, considerando

$$W^L(K_1 \cup K_2 - i) = VP^{\alpha}_{\beta} = \sum_{l \in K_1 \cup K_2 - i} a_l\, \alpha_l + \sum_{j \in J} b_j\, \beta_j$$

$$W(K_1 \cup K_2) = \sum_{l \in K_1 \cup K_2} a_l\, u^{\star}_l + \sum_{j \in J} b_j\, v^{\star}_j$$

obtemos em (II.18)

$$\Delta^L_i = \sum_{l \in K_1 \cup K_2 - i} (\alpha_l - u^{\star}_l)\, a_l + \sum_{j \in J} (\beta_j - v^{\star}_j)\, b_j - a_i\, u^{\star}_i \tag{III.17}$$

Para usar como multiplicadores (α_l, β_j), temos disponível a solução ótima $(u^{\star}_l, v^{\star}_j)$ de $T^D(K_1 \cup K_2)$ ou ainda $(u^{\star}_l, v_j(K_1 \cup K_2 - i))$, que de acordo com a prop. (8) são soluções viáveis de $T^D(K_1 \cup K_2 - i)$.

Por outro lado, conforme já comentado na seção III.1, em qualquer solução $(\hat{u}_l, \hat{v}_j)$ $l \in K_1 \cup K_2 - i$, $j \in J$ ótima para $T^D(K_1 \cup K_2 - i)$ ocorre $\hat{v}_j \geq v_j^\star \ \ \forall j \in J$. Este fato sugere que $v'_j = v_j(K_1 \cup K_2 - i)$ constituam melhores multiplicadores em comparação a $v_j^\star$, pois temos pela prop. (1) que $v'_j \geq v_j^\star$.

Consideremos, então, as duas opções, substituindo-as em (III.17)

- **Opção 1:** $(\alpha_l, \beta_j) = (u_l^\star, v_j^\star)$

$$\Delta_i^L = -a_i\, u_i^\star \tag{III.18}$$

- **Opção 2:** $(\alpha_l, \beta_j) = (u_l^\star, v'_j)$

$$\Delta_i^L = \sum_{j \in J} (v'_j - v_j^\star)\, b_j - a_i\, u_i^\star \tag{III.19}$$

Este último limite é o mesmo proposto por Mateus e Bornstein[41], que o denotaram por Δ_i^D. Confirmando a espectativa, constitui um limite visivelmente melhor que o primeiro, em termos de aproximação, já que $(v'_j - v_j^\star) \geq 0$.

Observe aqui que a solução do problema relaxado não mudaria se incluíssemos uma restrição quanto ao máximo fluxo entre armazéns e consumidores, por exemplo, do tipo $x_{lj} \leq \min(a_l, b_j) \quad \forall l \in K_1 \cup K_2 - i, \ \forall j \in J$ e, portanto, não se conseguiriam com isso melhores limites.

III.3.2 Relaxação da demanda

Considere o problema $T(K_1 \cup K_2 - i)$

minimizar

$$\sum_{l \in K_1 \cup K_2 - i} \ \sum_{j \in J} c_{lj}\, x_{lj} \tag{III.20}$$

sujeito a

$$\sum_{j \in J} x_{lj} \ \leq \ a_l \ \ \forall l \in K_1 \cup K_2 - i \tag{III.21}$$

$$\sum_{l \in K_1 \cup K_2 - i} x_{lj} \ = \ b_j \ \ \forall j \in J \tag{III.22}$$

$$x_{lj} \ \geq \ 0 \quad \forall l \in K_1 \cup K_2 - i, \ \forall j \in J \tag{III.23}$$

Relaxando (III.22) com β_j, $j \in J$, obtemos

minimizar

$$\sum_{l \in K_1 \cup K_2 - i} \sum_{j \in J} (c_{lj} - \beta_j)\, x_{lj} \quad + \quad \sum_{j \in J} \beta_j\, b_j$$

sujeito a

$$\begin{aligned} \sum_{j \in J} x_{lj} &\leq a_l \quad \forall l \in K_1 \cup K_2 - i \\ x_{lj} &\geq 0 \quad \forall l \in K_1 \cup K_2 - i,\ \forall j \in J \end{aligned}$$

que pode ser decomposto em $\mid K_1 \cup K_2 \mid -1$ subproblemas da mochila $P_\beta(l)$, um para cada $l \in K_1 \cup K_2 - i$, do tipo

minimizar

$$\sum_{j \in J} (c_{lj} - \beta_j)\, x_{lj}$$

sujeito a

$$\begin{aligned} \sum_{j \in J} x_{lj} &\leq a_l \\ x_{lj} &\geq 0 \quad \forall j \in J \end{aligned}$$

cuja solução $VP_\beta(l)$ é dada por

$$VP_\beta(l) = \min(0, \min_{j \in J}\{c_{lj} - \beta_j\})\, a_l \tag{III.24}$$

Então como

$$\begin{aligned} W^L(K_1 \cup K_2 - i) &= \sum_{l \in K_1 \cup K_2 - i} VP_\beta(l) + \sum_{j \in J} b_j\, \beta_j \\ W(K_1 \cup K_2) &= \sum_{l \in K_1 \cup K_2} a_l\, u_l^\star + \sum_{j \in J} b_j\, v_j^\star \end{aligned}$$

obtemos em (II.18)

$$\Delta_i^L = \sum_{l \in K_1 \cup K_2 - i} (VP_\beta(l) - u_l^\star\, a_l) + \sum_{j \in J} (\beta_j - v_j^\star)\, b_j - a_i\, u_i^\star \tag{III.25}$$

Quer usemos em (III.24) $\beta_j = v_j^\star$ ou $\beta_j = v_j' = v_j(K_1 \cup K_2 - i)$, obteremos, pela prop. (4),

$$VP_\beta(l) = u_l^\star\, a_l \tag{III.26}$$

Assim, respectivamente para cada opção, teríamos em (III.25)

$$\Delta_i^L = -a_i\, u_i^\star \tag{III.27}$$

$$\Delta_i^L = \sum_{j \in J} (v_j' - v_j^\star)\, b_j - a_i\, u_i^\star \tag{III.28}$$

ou seja, os mesmos limites conseguidos relaxando-se oferta e demanda.

Com um pouco de atenção, pode-se também verificar, neste caso, que a solução do problema relaxado não seria alterada se acrescentássemos a cada subproblema $P_\beta(l)$ a restrição $x_{lj} \leq b_j \ \ \forall j \in J$. Tal conclusão pode ser obtida considerando-se as seguintes possibilidades.

1. Se $u_l^\star = 0$ então $\min_{j \in J}\{c_{lj} - v_j^\star\} \geq 0$. Logo teríamos a mesma solução anterior $VP_\beta(l) = 0 = u_l^\star \, a_l$.

2. Do contrário, temos pela prop. (6) que a soma das demandas dos consumidores j que fornecem os menores valores para $(c_{lj} - v_j^\star)$ (todos iguais a $u_l^\star$) já preenchem a capacidade de oferta do armazém l. Logo temos, da mesma forma, $VP_\beta(l) = u_l^\star \, a_l$.

Sendo assim, as aproximações geradas seriam idênticas.

III.3.3 Relaxação da oferta

Considere o problema $T(K_1 \cup K_2 - i)$

minimizar

$$\sum_{l \in K_1 \cup K_2 - i} \ \sum_{j \in J} c_{lj} \, x_{lj} \tag{III.29}$$

sujeito a

$$\sum_{j \in J} x_{lj} \ \leq \ a_l \ \ \forall l \in K_1 \cup K_2 - i \tag{III.30}$$

$$\sum_{l \in K_1 \cup K_2 - i} x_{lj} \ = \ b_j \ \ \forall j \in J \tag{III.31}$$

$$x_{lj} \ \geq \ 0 \ \ \forall l \in K_1 \cup K_2 - i, \ \forall j \in J \tag{III.32}$$

Relaxando o conjunto de restrições (III.30) com $\alpha_l, \ l \in K_1 \cup K_2 - i$, obtemos

minimizar

$$\sum_{j \in J} \ \sum_{l \in K_1 \cup K_2 - i} (c_{lj} - \alpha_l) \, x_{lj} \ + \ \sum_{l \in K_1 \cup K_2 - i} \alpha_l \, a_l$$

sujeito a

$$\sum_{l \in K_1 \cup K_2 - i} x_{lj} \ = \ b_j \ \ \forall j \in J$$

$$x_{lj} \ \geq \ 0 \ \ \forall l \in K_1 \cup K_2 - i, \ \forall j \in J$$

equivalente à resolução de $\mid J \mid$ subproblemas da mochila $P_\alpha(j), \ j \in J$, do tipo

minimizar

$$\sum_{l \in K_1 \cup K_2 - i} (c_{lj} - \alpha_l)\, x_{lj}$$

sujeito a

$$\begin{aligned} \sum_{l \in K_1 \cup K_2 - i} x_{lj} &= b_j \\ x_{lj} &\geq 0 \quad \forall l \in K_1 \cup K_2 - i \end{aligned}$$

cuja valor ótimo $VP_\alpha(j)$ é dado por

$$VP_\alpha(j) = \min_{l \in K_1 \cup K_2 - i} \{c_{lj} - \alpha_l\}\, b_j \tag{III.33}$$

Logo, considerando que

$$\begin{aligned} W^L(K_1 \cup K_2 - i) &= \sum_{j \in J} VP_\alpha(j) + \sum_{l \in K_1 \cup K_2 - i} a_l\, \alpha_l \\ W(K_1 \cup K_2) &= \sum_{j \in J} b_j\, v_j^\star + \sum_{l \in K_1 \cup K_2} a_l\, u_l^\star \end{aligned}$$

gera-se, substituindo em (II.18),

$$\Delta_i^L = \sum_{j \in J} (VP_\alpha(j) - v_j^\star\, b_j) + \sum_{l \in K_1 \cup K_2 - i} (\alpha_l - u_l^\star)\, a_l - a_i\, u_i^\star \tag{III.34}$$

Neste caso temos uma única opção, isto é, considerarmos $\alpha_l = u_l^\star$, tornando, em (III.33),

$$VP_\alpha(j) = v_j(K_1 \cup K_2 - i)\, b_j = v_j'\, b_j \tag{III.35}$$

e, consequentemente, em (III.34),

$$\Delta_i^L = \sum_{j \in J} (v_j' - v_j^\star)\, b_j - a_i\, u_i^\star \tag{III.36}$$

Este resultado para Δ_i^L pode se justificar, se observarmos que, fixando $u_l = u_l^\star$, $T^D(K_1 \cup K_2 - i)$ ficaria reduzido ao problema

maximizar

$$\sum_{j \in J} b_j\, v_j$$

sujeito a

$$v_j \leq c_{lj} - u_l^\star \quad \forall l \in K_1 \cup K_2 - i,\ \forall j \in J$$

cuja solução é $v_j = \min_{l \in K_1 \cup K_2 - i}\{c_{lj} - u_l^\star\} = v_j' \ \ \forall j \in J$.

Novamente aqui não se obteria melhor limite, mesmo se incluíssemos em cada subproblema $P_\alpha(j)$ a restrição $x_{lj} \leq a_l \ \ \forall l \in K_1 \cup K_2 - i$, pois a solução do problema relaxado não se modificaria. Este é o resultado que se extrai da prop. (7), ao garantir que a soma de ofertas das facilidades l, correspondentes aos menores valores de $(c_{lj} - u_l^\star)$ (todos iguais a $v_j^\star$), são suficientes para abastecer cada consumidor j. Assim, teríamos igualmente $VP_\alpha(j) = v_j^\star\, b_j$.

III.3.4 Comentários

As diversas relaxações apresentadas conduziram, na verdade, aos mesmos limites, quais sejam

$$\Delta_i^L = -a_i\, u_i^\star \tag{III.37}$$

$$\Delta_i^D = \sum_{j \in J} (v'_j - v_j^\star)\, b_j - a_i\, u_i^\star \tag{III.38}$$

Enquanto Δ_i^L considera apenas o potencial $(-u_i)$ do próprio armazém i, o cálculo de Δ_i^D também leva em conta a variação (ou uma aproximação dela) do potencial de cada centro consumidor j, com a possível desativação da facilidade i.

Na verdade, variações de potencial só são avaliadas para os consumidores j totalmente atendidos pela facilidade i em questão, pois, pela prop. (2), $v'_j = v_j^\star$ se $x_{ij}^\star \neq b_j$, o que faz

$$\Delta_i^D = \sum_{\substack{j \in J \\ x_{ij} = b_j}} (v'_j - v_j^\star)\, b_j - a_i\, u_i^\star \tag{III.39}$$

Melhor seria se pudéssemos estimar, também sem dificuldade, algum efeito sobre os consumidores parcialmente por ela abastecidos.

Por outro lado, à medida que facilidades vão sendo desativadas, possivelmente novos clientes serão atendidos, parcial ou exclusivamente, por outros armazéns. (Esta é mais uma comprovação de que Δ_i tende a crescer.)

Numericamente é fácil verificar a superioridade do segundo limite sobre o primeiro, pois o termo $(v'_j - v_j^\star)$ é não negativo pela prop (1). Pode-se demonstrar também (Mateus e Bornstein[41]) que $\Delta_i^D \geq \overline{\Delta}_i$, ou seja, Δ_i^D é igualmente superior ao limite proposto por Akinc e Khumawala[3].

Além disso, o cálculo de Δ_i^D apresenta uma baixa complexidade de computação. Verifique por (III.39) que, considerando apenas $\mid \overline{J} \mid$ elementos de J, onde $\overline{J} = \{j \in J \mid x_{ij} = b_j \text{ e } i \in K_2\}$, podemos determinar todos os parâmetros $\Delta_i^D \ \forall i \in K_2$, tendo em vista que cada termo $(v'_j - v_j^\star)$, $j \in \overline{J}$, só poderá ser positivo para um único deles.

Pelo exposto, Δ_i^D parece ser teórica e computacionalmente o melhor limite a ser aplicado entre estes três.

III.4 Limites Para Ω_i

Esta seção será muito semelhante à anterior, sendo que aqui desenvolveremos limites superiores para Ω_i e, portanto, consideraremos relaxações do problema $T(K_1 \cup i)$. Novamente a idéia é relaxar as restrições de oferta e/ou demanda e tomar como multiplicadores a solução ótima de $T^D(K_1)$ ou alguma variação dela.

Entretanto, neste caso, diferentemente do anterior, a solução dual do problema básico $T(K_1)$ não fornece diretamente uma solução dual viável para o problema modificado $T(K_1 \cup i)$, uma vez que este apresenta uma restrição a mais (exatamente a restrição de capacidade do armazém i) e não se dispõe da variável dual a ela correspondente. Por isso, se se pretende relaxar também esta restrição, um outro multiplicador deve ser desenvolvido.

III.4.1 Relaxação da oferta e demanda

Considere o problema $T(K_1 \cup i)$

minimizar

$$\sum_{l \in K_1 \cup i} \sum_{j \in J} c_{lj}\, x_{lj} \tag{III.40}$$

sujeito a

$$\sum_{j \in J} x_{lj} \leq a_l \quad \forall l \in K_1 \cup i \tag{III.41}$$

$$\sum_{l \in K_1 \cup i} x_{lj} = b_j \quad \forall j \in J \tag{III.42}$$

$$x_{lj} \geq 0 \quad \forall l \in K_1 \cup i,\ \forall j \in J \tag{III.43}$$

Relaxando as restrições de oferta (III.41) e demanda (III.42), respectivamente, com α_l, $l \in K_1 \cup i$ e β_j, $j \in J$, soluções viáveis de $T^D(K_1 \cup i)$, obtemos o problema P^{α}_{β}

minimizar

$$\sum_{l \in K_1 \cup i} \sum_{j \in J} (c_{lj} - \alpha_l - \beta_j)\, x_{lj} \;+\; \sum_{l \in K_1 \cup i} \alpha_l\, a_l \;+\; \sum_{j \in J} \beta_j\, b_j$$

sujeito a

$$x_{lj} \geq 0 \quad \forall l \in K_1 \cup i,\ \forall j \in J$$

Como (α_l, β_j) é solução viável de $T^D(K_1 \cup i)$, temos

$$\alpha_l + \beta_j \leq c_{lj} \Longleftrightarrow c_{lj} - \alpha_l - \beta_j \geq 0$$

Logo, uma solução de P_β^α encontra-se em $x_{lj} = 0 \quad \forall l \in K_1 \cup i, \ \forall j \in J$, gerando o valor ótimo

$$VP_\beta^\alpha = \sum_{l \in K_1 \cup i} a_l \, \alpha_l + \sum_{j \in J} b_j \, \beta_j \tag{III.44}$$

Assim, aplicando

$$\begin{aligned} W^L(K_1 \cup i) \ &= VP_\beta^\alpha = \ \sum_{l \in K_1 \cup i} a_l \, \alpha_l + \sum_{j \in J} b_j \, \beta_j \\ W(K_1) \qquad &= \qquad \sum_{l \in K_1} a_l \, u_l^\star + \sum_{j \in J} b_j \, v_j^\star \end{aligned}$$

em (II.22) temos

$$\Omega_i^U = \sum_{l \in K_1} (u_l^\star - \alpha_l) \, a_l + \sum_{j \in J} (v_j^\star - \beta_j) \, b_j - a_i \, \alpha_i \tag{III.45}$$

Podemos então tomar $\alpha_l = u_l^\star, \ l \in K_1$ e $\beta_j = v_j^\star, \ j \in J$, onde $(u_l^\star, v_j^\star)$ é a solução ótima de $T^D(K_1)$. Mas que valor atribuir a α_i ? Se não limitarmos a capacidade do armazém i, então devemos fazer $\alpha_i = 0$, obtendo, $\Omega_i^U = 0$.

Uma opção mais adequada, uma vez fixados $\alpha_l = u_l^\star$ e $\beta_j = v_j^\star$, parece considerar a simplificação de $T^D(K_1 \cup i)$ então gerada,

$$\begin{array}{lll} \text{maximizar} & & a_i \, u_i \\ \text{sujeito a} & & \\ & u_i \ \leq \ c_{ij} - v_j^\star & \forall j \in J \\ & u_i \ \leq \ 0 & \end{array}$$

e tomar α_i como sua solução, isto é, fazer $\alpha_i = \min(0, \min_{j \in J}\{c_{lj} - v_j^\star\})$, o que, substituindo em (III.45), resultaria

$$\Omega_i^U = \max(0, \max_{j \in J}\{v_j^\star - c_{ij}\}) \, a_i \tag{III.46}$$

Observe que a solução do problema relaxado não seria modificada se adicionalmente incluíssemos uma restrição quanto ao máximo fluxo entre armazéns e consumidores, por exemplo, do tipo $x_{lj} \leq \min(a_l, b_j) \quad \forall l \in K_1 \cup i, \ \forall j \in J$. Em consequência, a aproximação para Ω_i permaneceria a mesma.

III.4.2 Relaxação da demanda

Considere o problema $T(K_1 \cup i)$

minimizar

$$\sum_{l \in K_1 \cup i} \sum_{j \in J} c_{lj}\, x_{lj} \tag{III.47}$$

sujeito a

$$\sum_{j \in J} x_{lj} \leq a_l \quad \forall l \in K_1 \cup i \tag{III.48}$$

$$\sum_{l \in K_1 \cup i} x_{lj} = b_j \quad \forall j \in J \tag{III.49}$$

$$x_{lj} \geq 0 \quad \forall l \in K_1 \cup i,\ \forall j \in J \tag{III.50}$$

Aplicando multiplicadores β_j, $j \in J$, às restrições (III.49) obtemos a relaxação lagrangeana

minimizar

$$\sum_{l \in K_1 \cup i} \sum_{j \in J} (c_{lj} - \beta_j)\, x_{lj} + \sum_{j \in J} \beta_j\, b_j$$

sujeito a

$$\sum_{j \in J} x_{lj} \leq a_l \quad \forall l \in K_1 \cup i$$

$$x_{lj} \geq 0 \quad \forall l \in K_1 \cup i,\ \forall j \in J$$

que equivale à solução de $\mid K_1 \mid +1$ subproblemas da mochila $P_\beta(l)$, $l \in K_1 \cup i$, apresentado abaixo

minimizar

$$\sum_{j \in J} (c_{lj} - \beta_j)\, x_{lj}$$

sujeito a

$$\sum_{j \in J} x_{lj} \leq a_l$$

$$x_{lj} \geq 0 \quad \forall j \in J$$

Sua solução $VP_\beta(l)$ satisfaz

$$VP_\beta(l) = \min(0, \min_{j \in J}\{c_{lj} - \beta_j\})\, a_l \qquad \text{(III.51)}$$

Então, aplicando em (II.22)

$$\begin{aligned} W^L(K_1 \cup i) &= \sum_{l \in K_1 \cup i} VP_\beta(l) + \sum_{j \in J} b_j\, \beta_j \\ W(K_1) &= \sum_{l \in K_1} a_l\, u_l^\star + \sum_{j \in J} b_j\, v_j^\star \end{aligned}$$

temos genericamente

$$\Omega_i^U = \sum_{l \in K_1} (u_l^\star\, a_l - VP_\beta(l)) \;+\; \sum_{j \in J} (v_j^\star - \beta_j)\, b_j - VP_\beta(i) \qquad \text{(III.52)}$$

Mais especificamente fazendo $\beta_j = v_j^\star$ em (III.51), pela prop. (4) temos

$$VP_\beta(l) = u_l^\star\, a_l \;\; \forall l \in K_1 \qquad \text{(III.53)}$$

e, em consequência, substituindo em (III.52),

$$\Omega_i^U = -VP_\beta(i) \qquad \text{(III.54)}$$

Mesmo acrescentando a restrição $x_{ij} \leq b_j \;\; \forall j \in J$ a cada subproblema $P_\beta(l)$, $l \in K_1$, sua solução permaneceria inalterada: é o que se extrai a partir da prop. (6), usando desenvolvimento análogo ao apresentado no final da subseção III.3.2, mas agora considerando que o problema base é $T(K_1)$. Com este resultado, Ω_i^U continuaria como em (III.54).

Quanto a $P_\beta(i)$, adotando $\beta_j = v_j^\star$ e incluindo a referida restrição, temos

minimizar

$$\sum_{j \in J} (c_{ij} - v_j^\star)\, x_{ij} \qquad \text{(III.55)}$$

sujeito a

$$\sum_{j \in J} x_{ij} \;\leq\; a_i \qquad \text{(III.56)}$$

$$x_{ij} \;\geq\; 0 \;\; \forall j \in J \qquad \text{(III.57)}$$

$$x_{ij} \;\leq\; b_j \;\; \forall j \in J \qquad \text{(III.58)}$$

onde podem-se analisar as seguintes variações:

1. A restrição (III.58) é desconsiderada

$$VP_\beta(i) = \min(0, \min_{j \in J}\{c_{ij} - v_j^\star\})\, a_l \tag{III.59}$$

2. A restrição (III.56) é desconsiderada

$$VP_\beta(i) = \sum_{j \in J} \min(0, c_{ij} - v_j^\star)\, b_j \tag{III.60}$$

3. Todas as restrições são consideradas

$$VP_\beta(i) = \min_{0 \le x_{ij} \le b_j} \{\sum_{j \in J} (c_{ij} - v_j^\star) x_{ij} \mid \sum_{j \in J} x_{ij} \ \le a_i\} \tag{III.61}$$

Desta forma, teríamos alternativamente em (III.54), para cada uma das opções (III.59),(III.60) e (III.61),

$$\Omega_i^U = \max(0, \max_{j \in J}\{v_j^\star - c_{ij}\})\, a_i \tag{III.62}$$

$$\Omega_i^U = \sum_{j \in J} \max(0, v_j^\star - c_{ij})\, b_j \tag{III.63}$$

$$\Omega_i^U = \max_{0 \le x_{ij} \le b_j} \{\sum_{j \in J} (v_j^\star - c_{ij}) x_{ij} \mid \sum_{j \in J} x_{ij} \ \le a_i\} \tag{III.64}$$

O segundo limite (III.63), que particularizaremos por Ω_i^D, é o mesmo proposto por Mateus e Bornstein[41]. Quanto ao último (III.64), corresponde ao limite de Akinc e Khumawala[3], a ser referenciado pela notação $\overline{\Omega}_i$.

III.4.3 Relaxação da oferta

Considere o problema $T(K_1 \cup i)$

minimizar

$$\sum_{l \in K_1 \cup i} \sum_{j \in J} c_{lj}\, x_{lj} \tag{III.65}$$

sujeito a

$$\sum_{j \in J} x_{lj} \ \le \ a_l \ \ \forall l \in K_1 \cup i \tag{III.66}$$

$$\sum_{l \in K_1 \cup i} x_{lj} \ = \ b_j \ \ \forall j \in J \tag{III.67}$$

$$x_{lj} \ \ge \ 0 \ \ \forall l \in K_1 \cup i, \ \forall j \in J \tag{III.68}$$

Relaxando o conjunto de restrições (III.66) com α_l, $l \in K_1 \cup i$, obtemos

minimizar

$$\sum_{j \in J} \sum_{l \in K_1 \cup i} (c_{lj} - \alpha_l)\, x_{lj} \;+\; \sum_{l \in K_1 \cup i} \alpha_l\, a_l$$

sujeito a

$$\begin{aligned} \sum_{l \in K_1 \cup i} x_{lj} &= b_j \;\; \forall j \in J \\ x_{lj} &\geq 0 \;\; \forall l \in K_1 \cup i,\; \forall j \in J \end{aligned}$$

que pode ser decomposto em $\mid J \mid$ subproblemas da mochila $P_\alpha(j)$, $j \in J$, do tipo

minimizar

$$\sum_{l \in K_1 \cup i} (c_{lj} - \alpha_l)\, x_{lj}$$

sujeito a

$$\begin{aligned} \sum_{l \in K_1 \cup i} x_{lj} &= b_j \\ x_{lj} &\geq 0 \;\; \forall l \in K_1 \cup i \end{aligned}$$

cuja solução $VP_\alpha(j)$ seria

$$VP_\alpha(j) = \min_{l \in K_1 \cup i} \{c_{lj} - \alpha_l\}\, b_j \tag{III.69}$$

Logo, como

$$\begin{aligned} W^L(K_1 \cup i) &= \sum_{j \in J} VP_\alpha(j) + \sum_{l \in K_1 \cup i} a_l\, \alpha_l \\ W(K_1) &= \sum_{j \in J} b_j\, v_j^\star + \sum_{l \in K_1} a_l\, u_l^\star \end{aligned}$$

obtém-se em (II.22)

$$\Omega_i^U = \sum_{l \in K_1} (u_l^\star - \alpha_l)\, a_l + \sum_{j \in J} (v_j^\star\, b_j - VP_\alpha(j)) - a_i\, \alpha_i \tag{III.70}$$

Se tomarmos $\alpha_l = u_l^\star \;\; \forall l \in K_1$, teremos em (III.69), pela prop. (3),

$$\begin{aligned} VP_\alpha(j) &= \min(v_j^\star, c_{ij} - \alpha_i)\, b_j & \text{(III.71)} \\ &= (v_j^\star - \max(0, v_j^\star - c_{ij} + \alpha_i))\, b_j & \text{(III.72)} \end{aligned}$$

tornando, em (III.70),

$$\Omega_i^U = \sum_{j \in J} \max(0, v_j^\star - c_{ij} + \alpha_i)\, b_j - a_i\, \alpha_i \tag{III.73}$$

Então, considerando as duas opções já propostas para o multiplicador α_i, temos em (III.73)

- **Opção 1:** $\alpha_i = 0$

$$\Omega_i^U = \sum_{j \in J} \max(0, v_j^\star - c_{ij})\, b_j \tag{III.74}$$

- **Opção 2:** $\alpha_i = \min(0, \min_{r \in J}\{c_{ir} - v_r^\star\})$

$$\begin{aligned} c_{ij} - \alpha_i &= c_{ij} + \max(0, \max_{r \in J}\{v_r^\star - c_{ir}\}) \\ &\geq c_{ij} + \max_{r \in J}\{v_r^\star - c_{ir}\} \\ &\geq c_{ij} + v_j^\star - c_{ij} \\ &= v_j^\star \end{aligned}$$

$$\Rightarrow \max(0, v_j^\star - c_{ij} + \alpha_i) = 0 \;\; \forall j \in J$$

Então

$$\Omega_i^U = \max(0, \max_{j \in J}\{v_j^\star - c_{ij}\})\, a_i \tag{III.75}$$

Vejamos agora que limites obteríamos se incluíssemos em cada $P_\alpha(j)$ a restrição $x_{lj} \leq a_l \;\; \forall l \in K_1 \cup i$.

Considere novamente os dois casos (agora em ordem inversa para facilitar o desenvolvimento)

- **Opção 2:** $\alpha_i = \min(0, \min_{r \in J}\{c_{ir} - v_r^\star\})$.

  Vimos que $c_{ij} - \alpha_i \geq v_j^\star = \min_{l \in K_1}\{c_{lj} - u_l^\star\}$. Então, empregando a prop. (7) e um raciocínio similar ao exposto no final da subseção III.3.3, temos que a solução de cada subproblema $P_\alpha(j) \;\; \forall j \in J$ permaneceria inalterada, isto é, $VP_\alpha(j) = v_j^\star\, b_j$, gerando o mesmo limite anterior.

- **Opção 1:** $\alpha_i = 0$

  Usando novamente a prop. (7), concluímos que só haveria possibilidade de alteração em $VP_\alpha(j)$ para algum $j \in J$ caso o menor $(c_{lj} - \alpha_l)$, $l \in K_1 \cup i$, fosse único e ocorresse exatamente em i, isto é, se tivéssemos

$$\min(v_j^\star, c_{ij} - \alpha_i) = \min(v_j^\star, c_{ij}) = c_{ij} < v_j^\star$$

  Além disso deveria também acontecer $a_i < b_j$.

Assim, considerando todas as possibilidades para esta opção, teríamos:

1. $c_{ij} < v_j^\star$
   - $a_i < b_j \Rightarrow VP_\alpha(j) = c_{ij}a_i + v_j^\star(b_j - a_i) = (c_{ij} - v_j^\star)a_i + v_j^\star b_j$
   - $a_i \geq b_j \Rightarrow VP_\alpha(j) = c_{ij}b_j = (c_{ij} - v_j^\star)b_j + v_j^\star b_j$
2. $c_{ij} \geq v_j^\star \Rightarrow VP_\alpha(j) = v_j^\star b_j$

ou genericamente

$$VP_\alpha(j) = \min(0, c_{ij} - v_j^\star)\min(a_i, b_j) + v_j^\star b_j$$

o que geraria em (III.70)

$$\Omega_i^U = \sum_{j \in J} \max(0, v_j^\star - c_{ij})\min(a_i, b_j) \tag{III.76}$$

uma simples modificação de Ω_i^D

III.4.4 Comentários

Os três limites apresentados, quais sejam,

$$\Omega_i^U = \max(0, \max_{j \in J}\{v_j^\star - c_{ij}\})\, a_l \tag{III.77}$$

$$\Omega_i^D = \sum_{j \in J} \max(0, v_j^\star - c_{ij})\, b_j \tag{III.78}$$

$$\overline{\Omega}_i = \max_{0 \leq x_{ij} \leq b_j} \{\sum_{j \in J}(v_j^\star - c_{ij})x_{ij} \mid \sum_{j \in J} x_{ij} \leq a_i\} \tag{III.79}$$

estimam a economia ao se ativar uma facilidade, contudo a partir de diferentes considerações.

O primeiro, Ω_i^U, leva em conta o possível potencial do armazém i acrescentado, considerando que a distribuição de fluxo seria pouco alterada, isto é, que toda a capacidade de i seria destinada a demandas parciais anteriormente supridas por outros armazéns.

Já Ω_i^D confronta a forma atual de fornecimento a cada consumidor j com a nova possibilidade criada pela ativação do armazém i, sem contudo, limitar sua capacidade de oferta.

Quanto a $\overline{\Omega}_i$, considera tanto a capacidade de oferta do armazém i, quanto avalia a variação criada com a nova possibilidade de atendimento proporcionada por sua ativação. Espera-se, portanto, que seja um limite mais "estreito".

Comprovemos esta espectativa em termos numéricos. Para isso considere $\overline{J}(i) = \{j \in J \mid v_j^\star - c_{ij} > 0\}$. Se $\overline{J}(i) = \emptyset$ então $\Omega_i^U = \Omega_i^D = \overline{\Omega}_i = 0$. Do contrário considere as duas situações abaixo.

- **Caso 1:** $\sum_{j\in\overline{J}(i)} b_j \leq a_i$

$$\begin{aligned}
\overline{\Omega}_i &= \max_{0\leq x_{ij}\leq b_j} \{ \sum_{j\in\overline{J}(i)} (v_j^\star - c_{ij}) x_{ij} \mid \sum_{j\in\overline{J}(i)} x_{ij} \leq a_i \} \\
&= \sum_{j\in\overline{J}(i)} (v_j^\star - c_{ij}) b_j \\
&= \Omega_i^D \\
&\leq \max_{j\in\overline{J}(i)} \{v_j^\star - c_{ij}\} \sum_{\overline{J}(i)} b_j \\
&\leq \Omega_i^U
\end{aligned}$$

- **Caso 2:** $\sum_{j\in\overline{J}(i)} b_j > a_i$

Considere $(j_1, j_2, \ldots, j_{\overline{n}}), \overline{n} = \mid \overline{J}(i) \mid$ a disposição dos consumidores $j \in \overline{J}(i)$ em ordem crescente de valores de $(v_j^\star - c_{ij})$. Então seja

$$a_i = b_{j_1} + b_{j_2} +, \ldots, b_{j_{r-1}} + \lambda\, b_{j_r}$$

onde $r \leq \overline{n}$, $0 \leq \lambda < 1$. Temos que

$$\begin{aligned}
\overline{\Omega}_i &= \sum_{k=1}^{r-1} b_{j_k} (v_{j_k}^\star - c_{ij_k}) + \lambda\, b_{j_r} (v_{j_r}^\star - c_{ij_r}) \\
&< \sum_{k=1}^{\overline{n}} b_{j_k} (v_{j_k}^\star - c_{ij_k}) \\
&= \Omega_i^D
\end{aligned}$$

e que

$$\begin{aligned}
\overline{\Omega}_i &= \sum_{k=1}^{r-1} b_{j_k} (v_{j_k}^\star - c_{ij_k}) + \lambda\, b_{j_r} (v_{j_r}^\star - c_{ij_r}) \\
&\leq \max_{1\leq k\leq r} \{v_{j_k}^\star - c_{ij_k}\} (\sum_{k=1}^{r-1} b_{j_k} + \lambda\, b_{j_r}) \\
&\leq \Omega_i^U
\end{aligned}$$

Resumindo temos $\overline{\Omega}_i \leq \Omega_i^D$ e $\overline{\Omega}_i \leq \Omega_i^U$ ou, particularmente, caso ocorra $\sum_{j\in\overline{J}(i)} b_j \leq a_i$ então $\overline{\Omega}_i = \Omega_i^D \leq \Omega_i^U$.

O limite $\overline{\Omega}_i$ constitui-se, desta forma, na melhor aproximação. Todavia, seu cálculo necessita da resolução de um problema da mochila, o que não acontece com Ω_i^D. Verifique-se ainda que no caso 1 temos $\overline{\Omega}_i = \Omega_i^D$. Cria-se então uma dúvida: qual o melhor limite a usar, considerando tanto o esforço computacional quanto a qualidade da aproximação?

Se se decide por $\overline{\Omega}_i$, o ideal seria aplicá-lo apenas no caso 2.

A determinação de qual caso irá ocorrer, entretanto, impõe a determinação de $\overline{J}(i)$, o que pode ser feito paralelamente ao cálculo de Ω_i^D.

Se a decisão é mesmo pela aplicação de $\overline{\Omega}_i$, temos então três opções para o seu cálculo:

1. Ordenação dos valores $(v_j^\star - c_{ij})\ \forall j \in \overline{J}(i)$
2. Determinação dos r maiores valores de $(v_j^\star - c_{ij})$, $r \leq \overline{n}$
3. Determinação dos $(\overline{n} - r)$ menores valores de $(v_j^\star - c_{ij})$, $r \leq \overline{n}$, e aplicação em

$$\overline{\Omega_i} = \Omega_i^D - (1-\lambda) b_{j_r}(v_{j_r}^\star - c_{ij_r}) - \sum_{k=r+1}^{\overline{n}} b_{j_k}\,(v_{j_k}^\star - c_{ij_k})$$

e, é claro, escolher a mais vantajosa em termos de complexidade computacional.

Note ainda que, com a ativação de facilidades e a consequente criação de novas possibilidades de atendimento, o conjunto $\overline{J}(i)$ tende a ser cada vez menor, ou seja, o caso 1 deve passar a ocorrer mais frequentemente.

Parece então ser boa política, sempre se determinar Ω_i^D primeiro. Além disso, pode-se considerar a alternativa de cálculo

$$\Omega_i^D = \sum_{j \in J} \max(0, v_j^\star - c_{ij})\ \min(a_i, b_j)$$

desenvolvida na seção anterior.

Capítulo IV

O Problema de Transporte

Definida uma configuração, isto é, fixadas as variáveis y_i em 0 ou 1, vimos que o CWLP se resume a um problema de transporte.

E como a idéia do algoritmo que iremos propor é exatamente fixar, ao longo do processo de solução, facilidades como abertas ou fechadas, espera-se que diversos problemas de transporte necessitem ser resolvidos.

O método simplex padrão poderia ser então utilizado, pois, como veremos a seguir, o problema de transporte se constitui num modelo linear. Entretanto, desta forma, estaríamos ignorando sua estrutura especial, além da esparsidade da matriz de restrições.

Muitos métodos específicos de solução têm sido propostos. Aqui apresentaremos um algoritmo primal, baseado no desenvolvido por Ahrens e Finke[1], para a resolução de problemas simples-produto de fluxos em redes.

IV.1 Definição do Problema

Mantendo a correspondência com a notação usada para o CWLP, o problema de transporte pode ser definido como a determinação da melhor forma de se enviar produtos de alguns m pontos a n outros, considerando o custo c_{ij} para transportar uma unidade entre um ponto origem i e um ponto destino j e sabendo que cada ponto origem i tem uma capacidade de fornecimento a_i e cada ponto destino j, uma necessidade de atendimento b_j.

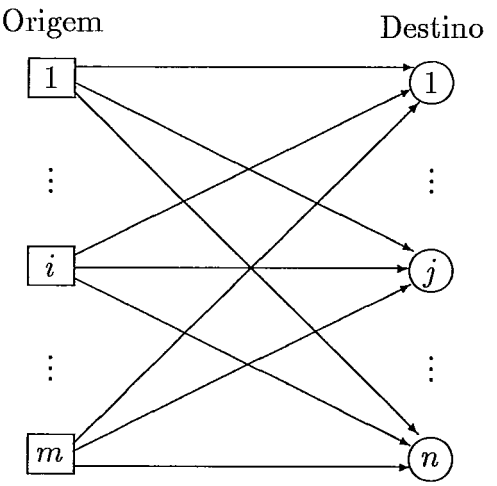


Figura IV.1: Grafo genérico representativo do problema de transporte

Se x_{ij} é o número de unidades transportadas da origem i, $i = 1, \ldots, m$ ao destino j, $j = 1, \ldots, n$, então o modelo matemático representativo do problema seria

minimizar

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij} \tag{IV.1}$$

sujeito a

$$\sum_{j=1}^{n} x_{ij} \leq a_i \quad i = 1, \ldots, m \tag{IV.2}$$

$$\sum_{i=1}^{m} x_{ij} \geq b_j \quad j = 1, \ldots, n \tag{IV.3}$$

$$x_{lj} \geq 0 \quad i = 1, \ldots, m \; j = 1, \ldots, n \tag{IV.4}$$

O grafo subjacente a esta descrição está ilustrado na figura IV.1, onde os nós representam os pontos origem e destino, e os arcos, as ligações entre eles.

Considera-se que seja um grafo completo, no sentido de que todas as possibilidades de arcos estão presentes, isto é, cada nó origem possui n arcos de saída, uma para cada nó destino. Assume-se que, se não é possível qualquer ligação entre um par (i, j), então o correspondente custo c_{ij} é tomado suficientemente grande para tornar este caminho proibitivo.

Assumimos também que $\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j$, pois do contrário o problema não teria solução viável.

Sem perda de generalidade, é possível transformar em igualdades o conjunto de restrições (IV.2), se introduzirmos um nó destino fictício (a ser identificado por nó 0) com demanda

$$b_0 = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$$

e custos

$$c_{i0} = 0 \;\; i = 1, \ldots, n$$

de forma a absorver todo o excesso de oferta, sem exercer qualquer influência sobre a solução do problema original.

Além disso, considerando que as restrições (IV.3), na solução ótima, são sempre satisfeitas na igualdade, podemos gerar o seguinte modelo equivalente

minimizar

$$\sum_{i=1}^{m} \sum_{j=0}^{n} c_{ij}\, x_{ij} \tag{IV.5}$$

sujeito a

$$\sum_{j=0}^{n} x_{ij} = a_i \;\; i = 1, \ldots, m \tag{IV.6}$$

$$\sum_{i=1}^{m} x_{ij} = b_j \;\; j = 0, \ldots, n \tag{IV.7}$$

$$x_{lj} \geq 0 \;\; i = 1, \ldots, m \; j = 0, \ldots, n \tag{IV.8}$$

cujo dual é

maximizar

$$\sum_{i=1}^{m} a_i\, u_i + \sum_{j=0}^{n} b_j\, v_j \tag{IV.9}$$

sujeito a

$$u_i + v_j \leq c_{ij} \;\; i = 1, \ldots, m \; j = 0, \ldots, n \tag{IV.10}$$

obtido associando-se as variáveis duais $u_i \;\; i = 1, \ldots, m$ e $v_j \;\; j = 1, \ldots, n$, respectivamente, às restrições (IV.6) e (IV.7).

Este problema dual possui infinitas soluções, pois se $(u_i^\star, v_j^\star)$ é solução ótima, então $(\hat{u}_i, \hat{v}_j) = (u_i^\star + k, v_j^\star - k)$ o será igualmente. A veracidade desta afirmação pode ser comprovada se verificarmos que:

- $(\hat{u}_i, \hat{v}_j)$ é viável

$$u'_i + v'_j = u^\star_i + k + v^\star_j - k = u^\star_i + v^\star_j \leq c_{ij}$$

- O valor da função objetivo se conserva

$$\begin{aligned}\sum_{i=1}^{m} a_i\, u'_i + \sum_{j=0}^{n} b_j\, v'_j &= \sum_{i=1}^{m} a_i\, u^\star_i + \sum_{j=0}^{n} b_j\, v^\star_j + k\underbrace{(\sum_{i=1}^{m} a_i - \sum_{j=0}^{n} b_j)}_{0} \\ &= \sum_{i=1}^{m} a_i\, u^\star_i + \sum_{j=0}^{n} b_j\, v^\star_j\end{aligned}$$

Entretanto, somente uma delas será de fato solução ótima se considerarmos a formulação dual para o problema sem o acréscimo do nó demanda fictício: exatamente aquela onde $v_0 = 0$. Assim, se $(u^\star_i, v^\star_j)$ é solução ótima para o modelo acima, então $(u^\star_i + v_0, v^\star_j - v_0)$ será ótima dual com relação ao problema original.

Considere, agora, o modelo modificado na forma reduzida

$$\begin{array}{lrcl} \text{minimizar} & cx & & \\ \text{sujeito a} & & & \\ & Ax & = & b \\ & x & \geq & 0 \end{array}$$

fazendo-se

$$\begin{aligned} x &= (x_{10}, \ldots, x_{1n}, x_{20}, \ldots, x_{2n}, \ldots, x_{m0}, \ldots, x_{mn})^t \\ c &= (c_{10}, \ldots, c_{1n}, c_{20}, \ldots, c_{2n}, \ldots, c_{m0}, \ldots, c_{mn}) \\ b &= (a_1, a_2, \ldots, a_m, -b_0, -b_1, \ldots, -b_n)^t \\ A &= (a_{10}, \ldots, a_{1n}, a_{20}, \ldots, a_{2n}, \ldots, a_{m0}, \ldots, a_{mn}) \end{aligned}$$

onde $a_{ij} = e_i - e_j$, e_i e e_j vetores unitários em R^{m+n+1}, com 1 na i-ésima e $(m+j+1)$-ésima posições respectivamente. Note-se que multiplicamos as equações (IV.7) por -1, apenas para facilitar a exposição a seguir.

Pode-se perceber, sem dificuldades, que A corresponde à matriz de incidência do grafo G associado ao problema de transporte, isto é, A tem uma linha para cada nó e uma coluna para cada arco de G, onde a coluna correspondente ao arco (i, j) tem $+1$ na linha i, -1 na linha $(m + j + 1)$ e 0 nas demais.

Devido à correspondência arco × coluna, podemos nos referir a arco básico sempre que a coluna de A for básica. Assim, a cada base B de A está associado um subgrafo de G composto por todos os arcos básicos e os nós a eles adjacentes. Este subgrafo é na verdade uma árvore geradora, conforme veremos no teorema abaixo.

Teorema: Uma submatriz B da matriz A é uma base se, e somente se, for matriz de incidência de uma árvore geradora do grafo G.

Dem.: A prova deste teorema está baseada nos seguintes fatos demonstrados em Ahrens e Finke[1] e Bazaraa[9].

1. As colunas de A correspondentes aos arcos de uma árvore geradora de G são linearmente independentes.
2. O posto da matriz A é igual a $(m+n)$, isto é, o total de nós menos 1.
3. O grafo correspondente a uma base B não pode conter ciclos.

$\boxed{\leftarrow}$ **Árvore geradora $\Longrightarrow$ Base B:** Existem árvores geradoras uma vez que G é conexo. Logicamente elas possuem $(m+n)$ arcos. De (1) e (2) temos então que as colunas a eles correspondentes formam uma base.

$\boxed{\rightarrow}$ **Base $B \Longrightarrow$ Árvore geradora:** Por (2) cada base possui $(m+n)$ colunas, cujos arcos correspondentes não podem formar ciclo, segundo (3). Logo constituem uma árvore geradora.

■

Desta forma, a cada solução básica do problema de transporte está associada uma árvore geradora do grafo a ele subjacente, que então chamaremos de árvore básica. Além disso, a determinação de sua solução ótima pode ser entendida como a construção de uma árvore ótima. A elaboração do nosso algoritmo será justamente fundamentada nesta estrutura particular do problema.

IV.2 Estrutura de Dados

A eficiência de um algoritmo matemático depende não só do método que implementa, como também da forma como as informações que manipula são armazenadas.

O procedimento principal de um método primal é a geração de uma solução básica após a outra ou, no algoritmo que apresentaremos, a construção de sucessivas árvores geradoras. Por isso é importante escolher uma estrutura adequada para o armazenamento da árvore básica, de modo a facilitar sua atualização e evitar buscas desnecessárias. Além disso, seria interessante que todas as demais informações relevantes fossem mantidas numa estrutura compatível com a da árvore.

A árvore básica em si estará adequadamente definida se selecionarmos um nó raiz (no nosso caso escolheremos o nó demanda artificial) e a partir dele indicarmos, para cada nó l, seu pai (ou predecessor) $P(l)$, que é simplesmente o próximo nó depois de l na única cadeia que o conecta à raiz.

Associada a esta conexão, pode também ser definida a especificação dos fluxos $X(l)$, que estará alternativamente no sentido $l \rightarrow P(l)$, caso l seja nó origem, ou no sentido contário, $l \leftarrow P(l)$, se l for nó destino.

Observe que não é necessário manter toda uma matriz x_{ij}, pois apenas as variáveis básicas podem ser diferentes de zero, e elas correspondem exatamente aos arcos da árvore geradora.

É possível passar de uma árvore básica a outra apenas usando a estrutura que define o predecessor de cada nó (Srinivasan e Thompson[49]). Entretanto, a atualização dos custos reduzidos, ou melhor, dos potenciais dos nós (variáveis duais), implica percorrer seus descendentes (sucessores).

Por isso, uma outra estrutura que facilite a identificação dos sucessores de um nó se faz necessária. Duas alternativas têm sido propostas na literatura.

A primeira, descrita por Johnson[35] e implementada por Glover, Karney e Klingman[29], constitui-se, na verdade, em dois índices: um que identifica o sucessor imediato (primeiro filho), se houver; e outro que aponta o irmão. A segunda, comprovadamente mais eficiente, sugerida por Glover, Klingman e Stutz[30], especifica a sequência em pré-ordem ao longo da árvore, que percorre os nós do topo (raiz) à base (folhas) e da esquerda para a direita, até retornar finalmente ao nó raiz, após passar por todos os outros. (Para maiores referências sobre estas estruturas ver Jacobsen[34] ou Kennington e Helgason[36]).

Esta segunda alternativa é, portanto, a mais comumente utilizada, e no nosso caso a implementaremos através da estrutura W, onde $W(l)$ representará o sucessor em pré-ordem do nó l.

Como veremos na seção seguinte, o trabalho de identificação do arco (coluna) a sair da base é grandemente reduzido se definirmos um outro apontador para indicar a profundidade (ou altura) dos nós. A profundidade $D(l)$ do nó l corresponde ao número de arcos entre l e a raiz, na cadeia que os interliga.

Em vez da profundidade, alguns algoritmos consideram outras alternativas, como o "número de sucessores" ou a "distância em pré-ordem" de cada nó. Kennington e Helgason[36] e Bradley, Brown e Graves[14], comparando as três opções,

constataram a superioridade, em termos de tempo de computação, da primeira sobre as seguintes, que serão, portanto, preteridas.

Note, agora, que cada árvore básica apresenta uma estrutura especial, qual seja: alternam-se níveis compostos ora apenas por nós oferta, ora apenas por nós demanda, a partir do primeiro, formado unicamente pelo nó demanda artificial.

Desta forma, se j é nó destino, então $P(j) = i$ é nó origem. Esta disposição particular dos nós na árvore básica sugere que é dispensável manter a profundidade para ambos os grupos de pontos origem e destino, pois se conservarmos no vetor D entradas apenas para, por exemplo, os nós oferta i, podemos simplesmente calcular para cada nó demanda j, diferente da raiz,

$$D(j) = D(P(j)) + 1 = D(i) + 1$$

Ainda mais, se observarmos que, pelas condições de folgas complementares, para todo arco básico (i, j) devemos ter $v_j = c_{ij} - u_i$, então podemos armazenar somente o potencial dos nós origem e calcularmos para os nós destino j,

$$v_j = c_{P(j),j} - u_{P(j)}$$

Consequentemente, tendo a estrutura W sido criada para possibilitar a determinação das variáveis duais, ela também só precisa ser mantida para os nós origem.

Evidentemente, o mesmo raciocínio pode ser desenvolvido em termos da manutenção dos valores de D,W e U apenas para os nós destino. Entretanto, a opção contrária será preferida, uma vez que normalmente temos muito menos nós origem que nós destino.

Resumindo, implementaremos no algoritmo, através de vetores, as seguintes estruturas:

IDENTIFICAÇÃO	DIMENSÃO	DESCRIÇÃO
X	$m+n$	fluxo nos arcos básicos
P	$m+n$	predecessor de cada nó
U	m	potencial dos nós origem
D	m	profundidade dos nós origem
W	m	sucessor em pré-ordem dos nós origem

Na verdade, apenas para generalizar as operações no algoritmo, cada um destes vetores possuirá uma posição a mais, com valor constante, exceto para W, correspondente ao nó raiz.

Nas próximas seções usaremos indistintamente a notação vetorial ($U(i)$) ou de subíndice (u_i), conforme seja mais conveniente. Quanto à indexação dos vetores X e P, pode surgir a seguinte dúvida: como não confundir o nó origem 1 com o nó destino 1, se ambos possuem entrada nestas estruturas ?

No Apêndice B apresentaremos uma forma diferente de numeração dos nós que resolve este problema. Contudo, para não nos retringirmos a uma implementação em particular, neste capítulo notaremos indiferentemente, sendo l origem ou destino, $P(l)$ e $X(l)$, considerando que a referência será bem entendida.

IV.3 O Algoritmo

O algoritmo a seguir proposto baseia-se no método simplex: segue, portanto, enumerando árvores básicas (soluções básicas) até atingir uma solução ótima.

Todos os passos serão aqui construídos explorando a estrutura da árvore básica, por isso o raciocínio frequentemente se desenvolverá dentro deste contexto. Adicionalmente, sempre que possível, faremos associações ao método simplex padrão.

IV.3.1 Critério de otimalidade

A partir dos problemas primal e dual podemos estabelecer a seguinte condição de folgas complementares

$$(c_{ij} - u_i - v_j)x_{ij} = 0$$

que nos permite verificar condições de otimalidade para um par de soluções x_{ij} e (u_i, v_j) primal e dual viáveis:

$$x_{ij} > 0 \quad \Rightarrow \quad c_{ij} - u_i - v_j = 0 \tag{IV.11}$$

$$x_{ij} = 0 \quad \Rightarrow \quad c_{ij} - u_i - v_j \geq 0 \tag{IV.12}$$

Suponha então dada uma solução básica primal viável não degenerada (posteriormente trataremos o caso de degeneração). Logo, temos $(m + n)$ variáveis $x_{ij} > 0$ (Bazaraa[9]), que usadas no conjunto de equações (IV.11) geram um sistema de dimensão $(m + n) \times (m + n + 1)$, indeterminado, com um grau de liberdade.

Então atribuindo um valor a qualquer das variáveis u_i ou v_j, podemos obter uma solução dual correspondente, que será ótima se for viável, ou seja, se também verificar o conjunto de equações (IV.12).

A variável a ser inicializada é justamente v_0, com valor zero, pois desta forma estaremos construindo, como vimos, uma solução do dual do problema original: a que de fato nos interessa. Na verdade, esta inicialização é executada implicitamente no algoritmo quando consideramos o nó demanda artificial como nó raiz, com potencial constantemente zero, e a partir dele é que determinamos o potencial dos demais nós da árvore básica.

IV.3.2 Entrada na Base

Uma vez que a otimalidade não seja atingida, temos por (IV.12) que existe $x_{ij} = 0$ tal que $c_{ij} - u_i - v_j < 0$. Então os arcos (i, j) não básicos que satisfizerem esta desigualdade são os candidatos a entrar na base.

A melhor escolha, isto é, aquela que reduz o número de pivoteamentos, seria o arco que minimizasse a expressão $(c_{ij} - u_i - v_j)$ ou, de forma equivalente no simplex usual, a variável de menor custo reduzido. Todavia, esta opção implicaria a avaliação de todos os arcos não básicos antes de cada mudança de base.

A experiência tem demonstrado que a aplicação desta regra ("Most Negative Rule") não é aconselhável, especialmente para problemas grandes. Da mesma forma, a regra oposta ("First Negative Rule"), que escolhe o primeiro candidato encontrado, também não deve ser utilizada.

Por outro lado tem-se verificado na prática que uma opção intermediária pode oferecer melhores resultados. Uma delas ("Inward Modified Negative Rule") seria fixar um nó destino j e considerar o mínimo de $(c_{ij} - u_i)$ entre os nós origem i. Outra ("Outward Modified Most Negative Rule") fixa, ao contrário, um nó origem i e escolhe como arco a sair aquele que minimize $(c_{ij} - v_j)$.

Tendo em vista sua eficiência e considerando que nos problemas práticos a quantidade de consumidores é normalmente bem maior que a de centros de oferta, usaremos a "Inward Modified Most Negative Rule". Adicionalmente acoplaremos uma pequena modificação: serão fixados NEXAM consumidores, em vez de apenas um.

Os nós destino serão considerados em forma cíclica, isto é, sendo j o último nó demanda fixado no pivoteamento anterior, $(j + 1 \pmod{n + 1})$ será o primeiro nó a ser tomado no pivoteamento atual. Ou ainda, o próximo a se considerar após o nó n é novamente o nó 0.

Sendo assim, o arco (INORIG,INDEST) a entrar na base pode ser determinado pela execução do seguinte bloco

$cont = 0$
repita
$cont = cont + 1$
$j = (j + 1) \bmod (n + 1)$
$c_{lj} - u_l = \min_{i \in I}\{c_{ij} - u_i\}$
UINCR $= c_{lj} - u_l - v_j$
até ($cont \geq$ NEXAM e UINCR < 0) ou ($cont = n + 1$)
se UINCR < 0
então (INORIG, INDEST) $= (l, j)$
senão *sol. ótima encontrada*

Como se vê, examinaremos, a cada iteração, um mínimo de NEXAM nós destino, ou até encontrar um custo reduzido negativo. Se todos são não negativos, a solução corrente é ótima (obedece a IV.12).

IV.3.3 Saída da Base

A entrada de um novo arco na árvore básica irá certamente determinar a geração de um único ciclo. Percorrendo-se este ciclo no sentido do novo arco, vão-se encontrar, alternadamente, arcos direcionados neste sentido e em sentido contrário.

Então, de modo a não enviar quantidades negativas, isto é, para preservar a viabilidade primal, o máximo fluxo que pode ser gerado no arco que entra e, consequentemente, no ciclo, é limitado pelos fluxos atuais dos arcos em sentido contrário.

O arco a sair da base será exatamente o que mais limitar o novo fluxo, isto é, aquele de menor fluxo atual, que então passará a ser zero.

Observe a semelhança deste procedimento com o clássico teste do ratio do método simplex: os arcos no sentido contrário correspondem aos elementos positivos da nova coluna básica, todos iguais a 1, já que a matriz associada ao problema é totalmente unimodular. Então a mínima razão é dada simplesmente pelo menor fluxo atual nestes arcos.

Na árvore básica este ciclo pode ser percorrido movendo-se, paralelamente, em ambas as cadeias que interligam os nós INORIG e INDEST à raiz, até que o primeiro ponto de interseção seja encontrado. Mais precisamente, a partir de INORIG e INDEST, a cada passo avança-se um arco na direção da raiz, em uma ou outra cadeia,

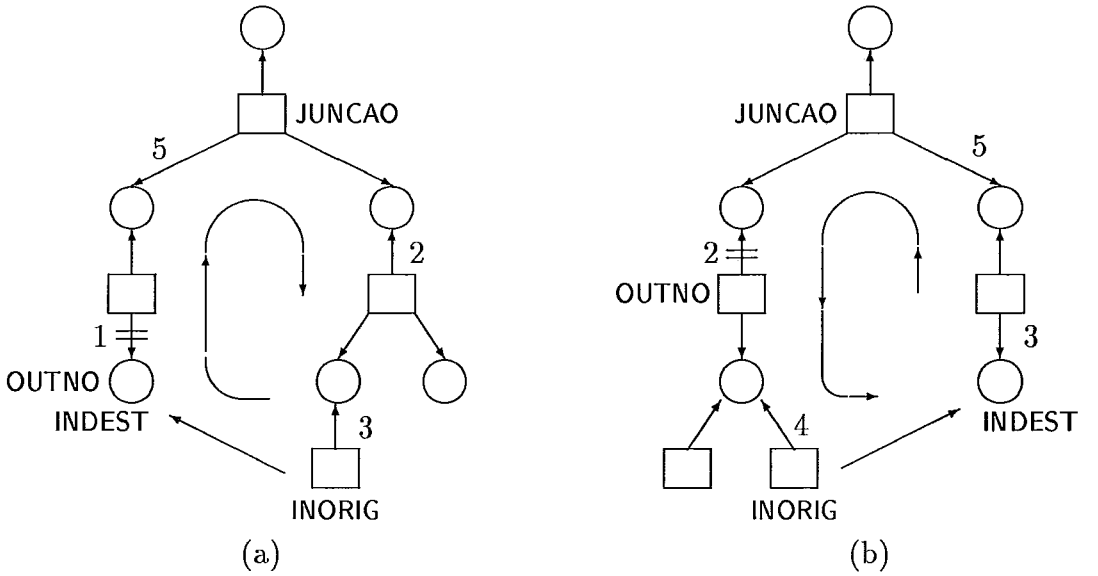


Figura IV.2: Escolha do arco a sair – duas situações

procurando sempre manter a mínima diferença positiva entre a maior das profundidades de seus nós[1], até que finalmente o nó JUNCAO seja determinado.

Assim, dividindo o ciclo nos ramos COJ e CDJ, que conectam respectivamente INORIG e INDEST a JUNCAO, a escolha do arco a sair da base será efetuada entre os arcos $(i,j) \in$ COJ onde $P(i) = j$ ou arcos $(i,j) \in$ CDJ com $P(j) = i$.

Se a aresta a sair é então tomada entre OUTNO e P(OUTNO), os candidatos a OUTNO são, como mostra a Figura IV.2, os nós origem em COJ e nós destino em CDJ. Assim, definindo

$$\begin{aligned} I(R) &= \{i \in \{1,\ldots,m\} \mid (i,j) \in R,\ R \text{ uma cadeia}\} \\ J(R) &= \{j \in \{0,\ldots,n\} \mid (i,j) \in R,\ R \text{ uma cadeia}\} \end{aligned}$$

podemos determinar

$$X(\text{OUTNO}) = \min_{l \in I(\text{COJ}) \cup J(\text{CDJ}) - \{\text{JUNCAO}\}} \{X(l)\}$$

Além disso X(OUTNO) será a variação de fluxo no ciclo.

A avaliação deste mínimo pode ser realizada paralelamente à determinação do nó JUNCAO observando que, em qualquer ramo da árvore, nós origem e destino se alternam, sendo que COJ começa por um nó origem e CDJ, por um nó destino.

[1]Observe que neste processo usamos o indíce P de predecessor, quando nos movemos nas cadeias, e a estrutura D, para comparação das profundidades de seus nós.

IV.3.4 Atualização da Árvore

A mudança de árvore básica implica não só atualizações nos vetores que mantêm sua própria estrutura (P,W e D), como também naqueles que armazenam os valores das variáveis primal e dual (X e U).

Como os novos valores são normalmente calculados com base nos antigos, de forma a não confundir uns com os outros, iremos diferenciar os vetores atualizados com uma linha (primo). Assim, por exemplo, $D(l)$ representa a profundidade corrente do nó l e $D'(l)$, sua nova profundidade.

Ainda para facilitar a exposição da maneira pela qual estas atualizações serão processadas, vamos primeiramente nomear alguns nós de forma particular.

Considere novamente o ciclo formado na árvore básica, agora sem os arcos de entrada e saída, constituído então por duas cadeias R e R'. Seja R' aquela que não contém o nó JUNCAO. Então defina:

ENTRADA nó em R adjacente ao arco (INORIG,INDEST)

PRIMEIRO a outra extremidade de (INORIG,INDEST)

ULTIMO o nó de saída (OUTNO)

SAIDA a outra extremidade do arco de saída (P(OUTNO))

INTIE[2] nó origem, predecessor em pré-ordem de ULTIMO

OUTTIE sucessor em pré-ordem de PRIMVISIT, ou seja, W(PRIMVISIT)

PRIMVISIT futuro predecessor em pré-ordem de PRIMEIRO: ENTRADA, predecessor em pré-ordem de ENTRADA ou INTIE

Observe, pela Figura IV.3, que a cadeia R compreende os ramos CEJ (de ENTRADA a JUNCAO) e CSJ (de SAIDA a JUNCAO), enquanto que R' constitui-se de um único ramo de extremidades em PRIMEIRO e ULTIMO.

Na cadeia R e nas subárvores a seus nós conectadas, todos os vetores, exceto X, permanecem inalterados. Isto é fácil perceber para os vetores P e D e igualmente para U, se considerarmos que o potencial do nó raiz v_0 é sempre mantido em zero. Quanto a W, só as entradas referentes a INTIE e PRIMVISIT deverão ser atualizadas, como veremos adiante.

[2]Preferiu-se manter a mesma denominação usada por Ahrens e Finke, que melhor representa o significado do nó. O mesmo comente-se com relação a OUTTIE.

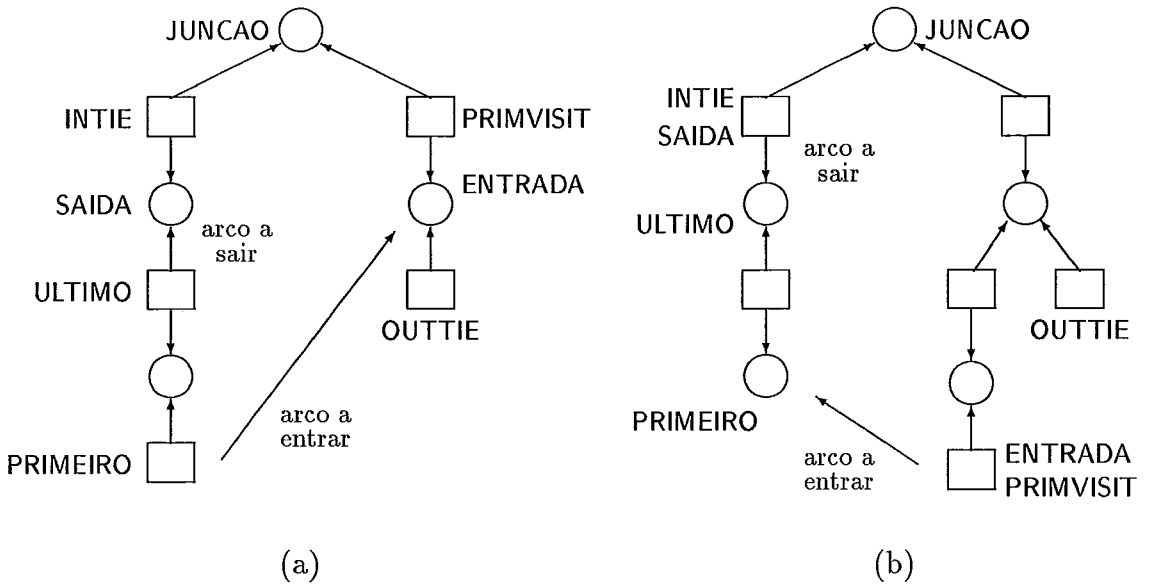


Figura IV.3: Nós particulares na árvore básica – duas situações

Nos fluxos X, as alterações ocorrem em todos os arcos do ciclo, isto é, tanto em R, quanto em R'. A variação XINCR, para mais ou para menos, depende do sentido destes arcos.

Para os nós em R, que não sofrem alteração de predecessor (em P), pode-se fazer

$$X'(l) = X(l) - \mathsf{XINCR} \quad \forall l \in J(\mathsf{CEJ}) \cup I(\mathsf{CSJ}) - \{\mathsf{JUNCAO}\}$$
$$X'(l) = X(l) + \mathsf{XINCR} \quad \forall l \in I(\mathsf{CEJ}) \cup J(\mathsf{CSJ}) - \{\mathsf{JUNCAO}\}$$

onde

$$\mathsf{XINCR} = \begin{cases} X(\mathsf{OUTNO}), & \text{se ENTRADA=INDEST} \\ -X(\mathsf{OUTNO}), & \text{se ENTRADA=INORIG} \end{cases}$$

De outra forma, observando que o sinal de XINCR muda alternadamente ao longo dos ramos CEJ e CSJ, devido a intercalação de nós origem e destino, poderíamos iterativamente fazer

XINCR $= -X(\mathsf{OUTNO})$
para l de ENTRADA a JUNCAO(exclusive) faça
  $X'(l) = X(l) + \mathsf{XINCR}$
  XINCR $= -$XINCR
XINCR $= X(\mathsf{OUTNO})$
para l de SAIDA a JUNCAO(exclusive) faça
  $X'(l) = X(l) + \mathsf{XINCR}$
  XINCR $= -$XINCR

Concentremos agora nossa atenção na área R', isto é, nos nós que formam o ramo R' e nas subárvores a eles conectadas. Observe que o ramo R' divide a área R' em duas subáreas: uma à esquerda e outra à direita. Teremos, assim, subárvores da esquerda ou subárvores da direita.

As informações relativas a esta área R' serão atualizadas considerando os nós do ramo R', na sequência de PRIMEIRO a ULTIMO e, a cada um deles, pesquisando também, se necessário, suas subárvores da direita e da esquerda. A visita aos nós do ramo é efetuada seguindo-se a estrutura de predecessor P, enquanto a pesquisa nas subárvores usa o apontador de pré-ordem W. Vejamos agora, separadamente, como se processariam as atualizações em cada uma das estruturas.

As mudanças em P só ocorrem para os nós $l \in N(R') = I(R') \cup J(R')$, nós do ramo R', e são facilmente verificadas: a relação pai–filho é invertida, ou seja, o pai de um nó passa então a ser seu filho e vice-versa. Então devemos fazer

$$\begin{aligned} P'(\text{PRIMEIRO}) &= \text{ENTRADA} \\ P'(l) &= p \quad \forall l \in N(R') - \{\text{PRIMEIRO}\} \end{aligned}$$

onde p é tal que $P(p) = l$, ou mais explicitamente executar o trecho

prev = ENTRADA
para todo $l \in N(R')$ (de PRIMEIRO a ULTIMO) faça
    $P'(l) = prev$
    $prev = l$

Verificando também a inversão na relação pai–filho, podemos agora estabelecer a atualização do vetor X para os nós em R', fazendo

$$\begin{aligned} X'(\text{PRIMEIRO}) &= X(\text{OUTNO}) \\ X'(l) &= X(p) - \text{XINCR} \quad \forall l \in J(R') - \{\text{PRIMEIRO}\} \\ X'(l) &= X(p) + \text{XINCR} \quad \forall l \in I(R') - \{\text{PRIMEIRO}\} \end{aligned}$$

onde p é tal que $P(p) = l$ e XINCR como já anteriormente definido, ou executar iterativamente

xprev = 0
XINCR = X(OUTNO)
para todo $l \in N(R')$ (de PRIMEIRO a ULTIMO) faça
    $X'(l) = xprev +$ XINCR
    $xprev = X(l)$
    XINCR = $-$XINCR

Quanto à variação no potencial dos nós, retomemos o valor de UINCR, determinado na escolha do arco a entrar na base,

$$\begin{aligned} \text{UINCR} &= C(\text{INORIG}, \text{INDEST}) - U(\text{INORIG}) - \overline{U}(\text{INDEST}) \\ &= C(\text{INORIG}, \text{INDEST}) - \overline{U}(\text{ENTRADA}) - \overline{U}(\text{PRIMEIRO}) \end{aligned}$$

onde, por simplificação, fizemos

$$\overline{U}(l) = \begin{cases} U(l), & \text{se } l \in I \\ C(P(l), l) - U(P(l)), & \text{se } l \in J \end{cases}$$

Ou rearranjando os termos, obtém-se

$$C(\text{INORIG}, \text{INDEST}) = (\overline{U}(\text{PRIMEIRO}) + \text{UINCR}) + \overline{U}(\text{ENTRADA})$$

Como para o novo arco básico devemos ter

$$C(\text{INORIG}, \text{INDEST}) = \overline{U}'(\text{PRIMEIRO}) + \overline{U}'(\text{ENTRADA})$$

podemos então tornar

$$\overline{U}'(\text{PRIMEIRO}) = \overline{U}(\text{PRIMEIRO}) + \text{UINCR}$$

e conservar

$$\overline{U}'(\text{ENTRADA}) = \overline{U}(\text{ENTRADA})$$

pois, desta forma, mantendo o potencial do nó ENTRADA, as alterações de potenciais ficariam restritas aos nós pertencentes à area R', o que efetivamente se deseja.

Note, adicionalmente, que não só PRIMEIRO, como todos os demais nós da área R' do seu mesmo tipo (origem/destino) devem receber o mesmo incremento de potencial. Já os nós de tipo contrário recebem o incremento inverso $-$UINCR.

Uma vez que somente os potenciais dos nós origem são mantidos, deveríamos assim fazer, para todo nó origem i da área R'

$$U'(i) = \begin{cases} U(i) + \text{UINCR}, & \text{se PRIMEIRO} \in I(R') \\ U(i) - \text{UINCR}, & \text{se PRIMEIRO} \in J(R') \end{cases}$$

Mais detalhadamente, mantendo a uniformidade na sequência de consideração dos nós que vem sendo empregada, a execução do bloco a seguir processaria a atualização do vetor U, a partir do valor UINCR já avaliado na subseção IV.3.2

se PRIMEIRO $\in J(R')$ então UINCR $= -$UINCR
para todo $l \in N(R')$ (de PRIMEIRO a ULTIMO) faça
  se $l \in I(R')$ então $U'(l) = U(l) +$ UINCR
  para todo nó origem i das subárvores da esquerda de l faça
    $U'(i) = U(i) +$ UINCR
  para todo nó origem i das subárvores da direita de l faça
    $U'(i) = U(i) +$ UINCR

Para a atualização da estrutura D, considere DINCR a variação na profundidade do nó PRIMEIRO, ou seja,

$$\begin{aligned} \text{DINCR} &= \overline{D}'(\text{PRIMEIRO}) - \overline{D}(\text{PRIMEIRO}) \\ &= \overline{D}(\text{ENTRADA}) + 1 - \overline{D}(\text{PRIMEIRO}) \end{aligned}$$

onde, apenas por simplificação, fizemos

$$\overline{D}(l) = \begin{cases} D(l), & \text{se } l \in I \\ D(P(l)), & \text{se } l \in J \end{cases}$$

Seja, então, l o k-ésimo nó entre PRIMEIRO e ULTIMO. Sua profundidade era

$$\overline{D}(l) = \overline{D}(\text{PRIMEIRO}) - k$$

devendo tornar-se

$$\begin{aligned} \overline{D}'(l) &= \overline{D}'(\text{PRIMEIRO}) + k \\ &= \text{DINCR} + \overline{D}(\text{PRIMEIRO}) + k \\ &= \overline{D}(l) + \text{DINCR} + 2k \end{aligned}$$

Logo DINCR deve ser incrementada de 2 a cada novo nó $l \in N(R')$ considerado. Observe, ainda, que as subárvores de cada um destes nós herdam também a mesma variação na profundidade.

Desta forma, a atualização em D poderia ser efetuada como a seguir

para todo $l \in N(R')$ (de PRIMEIRO a ULTIMO) faça
  se $l \in I(R')$ então $D'(l) = D(l) +$ DINCR
  para todo nó origem i das subárvores da esquerda de l faça
    $D'(i) = D(i) +$ DINCR
  para todo nó origem i das subárvores da direita de l faça
    $D'(i) = D(i) +$ DINCR
  DINCR $=$ DINCR $+ 2$

A necessidade de atualização em W deve-se também ao fato da inversão na relação pai–filho dos nós do ramo R'.

A ocorrência de um novo pai para cada nó l de R' determina, em consequência, a mudança do seu predecessor em pré-ordem. Além disso, se seu novo filho (seu antigo pai) é posicionado à direita de todos os demais, então suas subárvores esquerda e direita passarão a ficar lado-a-lado, mudando também o sequenciamento em pré-ordem.

Assim as mudanças em W podem ser efetuadas pela execução do seguinte trecho

VISIT = PRIMVISIT
para todo $l \in N(R')$ (de PRIMEIRO a ULTIMO) faça
  se $l \in I(R')$ então
    W'(VISIT) = l
    VISIT = l
  se l possui algum nó origem à esquerda então
    LGUIA = *primeiro deles*
    W'(VISIT) = LGUIA
    VISIT = *último deles*
  se l possui algum nó origem à direita então
    RGUIA = *primeiro deles*
    W'(VISIT) = RGUIA
    VISIT = *último deles*

que aborda ambos os casos acima mencionados, ou seja,

- Interliga o último nó origem visitado (VISIT) na sequência, antes de um nó do ramo, ao próximo nó origem
- Une a última subárvore à esquerda de um nó do ramo a sua primeira à direita

Na verdade, não é preciso determinar a cada $l \in N(R')$ o seu primeiro nó à direita e à esquerda.

O primeiro nó à direita de um nó l do ramo, caso exista, estará sempre disponível se conservarmos o referencial RGUIA = W(VISIT), enquanto VISIT visita os nós à direita do nó anterior $P'(l)$.

O primeiro nó à esquerda de um nó $l \in I(R')$, por sua vez, obtém-se trivialmente com LGUIA $= W(l)$. Para um nó $l \in J(R')$, consegue-se mantendo, de forma similar, o referencial LGUIA $= W$(VISIT), enquanto VISIT percorre os nós à direita do próximo nó do ramo $P(l)$.

Pelo que vimos, para um nó origem em R', temos disponível LGUIA e RGUIA, entretanto, se o nó do ramo é destino, embora RGUIA esteja sempre à mão, LGUIA só estará determinado quando o próximo nó do ramo for visitado.

Assim, visando incrementar a eficiência do bloco anterior, poderíamos modificá-lo da seguinte forma

VISIT = PRIMVISIT
RGUIA =*nó origem sucessor de* PRIMEIRO *em pré-ordem*
para todo $l \in N(R')$ (de PRIMEIRO a ULTIMO) faça
  se $l \in I(R')$ então
    visitant = VISIT (* marca onde começariam os nós à esquerda *)
    se RGUIA é no direito de l
      W'(VISIT) = RGUIA
      VISIT =*último deles*
      RGUIA = W(VISIT)
  senão
    W'(VISIT) = l
    VISIT = l
    LGUIA = W(VISIT)
    se LGUIA é nó esquerdo de l
      VISIT =*último deles*
      LGUIA = W(VISIT)
    se RGUIA é nó direito de l
      W'(VISIT) = RGUIA
      VISIT =*último deles*
      RGUIA = W(VISIT)
    se LGUIA é nó esquerdo de $P(l)$
      troca = W(*visitant*)
      W'(*visitant*) = LGUIA
      visitant =*último deles*
      W'(*visitant*) = *troca*

Note que o bloco acima falha quando ULTIMO é nó demanda, pois não encadearia seus possíveis descendentes da esquerda, já que não haverá próximo nó do ramo a considerar.

Observando que o primeiro nó a esquerda de ULTIMO, caso exista, é o sucessor de INTIE na árvore não atualizada, o pequeno trecho a seguir, acoplado ao anterior, resolveria o problema.

Se ULTIMO $\in J(R')$
  LGUIA $= W$(INTIE)
  se LGUIA é nó esquerdo de ULTIMO
    $troca = W(visitant)$
    $W'(visitant) =$ LGUIA
    $visitant =$*último deles*
    $W'(visitant) = troca$

A identificação de um nó origem como esquerdo (ou direito) a um nó do ramo, isto é, como pertencente a suas subárvores esquerda (ou direita) é facilmente constatada analisando-se a profundidade e filiação deste nó. O último deles é determinado a partir do primeiro, seguindo a estrutura W já disponível.

Analisando os diversos blocos construídos para efetuar as alterações ocorridas na área R' com a mudança de árvore básica, verifica-se que todas estas atualizações podem ser realizadas em um único laço, da seguinte forma: para cada nó do ramo, de PRIMEIRO a ULTIMO, caso este seja nó destino, serão considerados seus nós à direita; caso seja nó origem, serão percorridos primeiramente seus nós à esquerda, em seguida, seus nós à direita e, finalmente, os nós à esquerda do nó anterior do ramo. Em cada nó do ramo, P e X e, se origem, D,W e U são atualizados; em cada nó esquerdo ou direito, D,W e U recebem novos valores.

Só nos resta, agora, analisar a ligação final entre as áreas R e R', que pode ser efetuada fazendo-se

- como sucessor de PRIMVISIT(nó em R), o primeiro nó origem da área R' visitado. (Observe que este passo já é realizado no próprio laço).
- como sucessor do último nó visitado (VISIT), o antigo sucessor de PRIMVISIT, ou seja, OUTTIE.
- como sucessor de INTIE(antigo predecessor de ULTIMO), o antigo sucessor do último nó visitado, disponível em RGUIA. (Este passo só é necessário quando PRIMVISIT for diferente de INTIE).

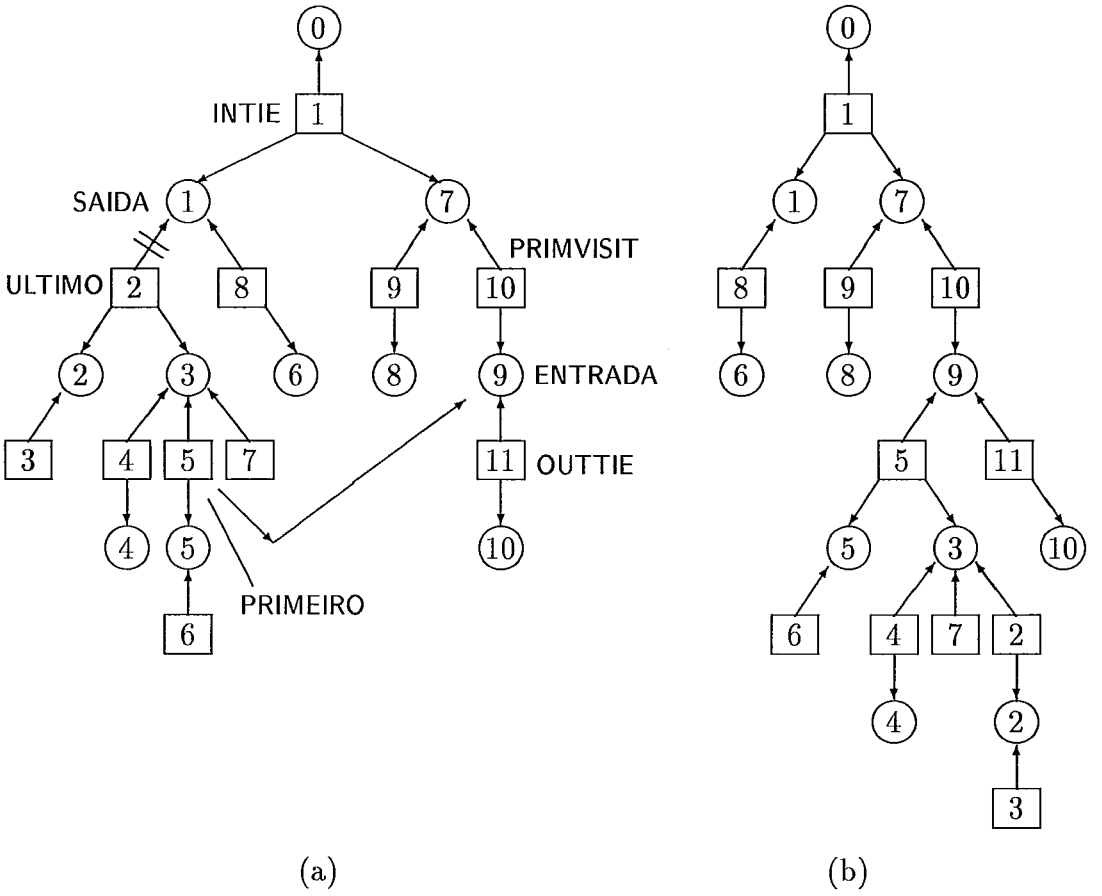


Figura IV.4: Atualização de uma árvore básica

A figura IV.4 esquematiza uma situação particular de atualização da árvore básica (a), a partir da introdução do arco $(5, 9)$ e retirada do arco $(2, 1)$, gerando a nova árvore básica (b). O objetivo principal desta figura é demonstrar as modificações na estrutura W, certamente a mais difícil de ser explicada textualmente.

Para uma melhor compreensão do completo funcionamento do algoritmo, a sugestão é, sem dúvida, uma análise de sua implementação no Apêndice B, onde se pode perceber o fluxo do programa como um todo e acompanhar mais de perto a função de cada variável de controle.

IV.4 Degeneração

Na ausência de degeneração, nosso algoritmo converge, como o simplex padrão, em um número finito de iterações. Além disso, observando-se que existe solução viável (p.e. $x_{ij} = a_i b_j / d$, $d = \sum_{i=1}^{m} a_i = \sum_{j=0}^{n} b_j$) e o problema é limitado ($0 \leq x_{ij} \leq \min(a_i, b_j)$), conclui-se que existe solução ótima, e a convergência será para ela.

Entretanto, exemplos podem ser construídos para mostrar que o algoritmo pode entar em *loop* indefinido na presença de degeneração. (Em nosso contexto entendida como a existência de um arco básico com fluxo igual a zero).

Portanto, regras para prevenir ciclagem devem ser acrescentadas. Examinemos, antes, em que condições pode ocorrer degeneração.

Suponha que numa dada iteração gerou-se uma solução degenerada. Então, retirando o arco degenerado, a árvore básica fica dividida em duas partes P e P'. Somando-se as restrições de oferta e as de demanda sobre uma destas partes, por exemplo P, temos

$$\begin{aligned} \sum_{(i,j)\in P} x_{ij} &= \sum_{i\in I(P)} a_i \\ \sum_{(i,j)\in P} x_{ij} &= \sum_{j\in J(P)} b_j \end{aligned}$$

ou seja,

$$\sum_{i\in I(P)} a_i = \sum_{j\in J(P)} b_j$$

Então uma condição necesária para a existência de degeneração é que um subconjunto próprio de pontos origem e outro de pontos destino tenham total de oferta igual ao total de demanda, isto é, que existam $I_1 \subset \{1,\ldots,m\}$ e $J_1 \subset \{0,\ldots,n\}$ tal que

$$\sum_{i\in I_1} a_i = \sum_{j\in J_1} b_j$$

Por outro lado, fazendo-se

$$\begin{aligned} a'_i &= a_i - \epsilon \; i = 1,\ldots,m \\ b'_j &= b_j \quad j = 1,\ldots,n \\ b'_0 &= b_0 - m\epsilon \end{aligned}$$

com $0 \leq \epsilon \leq \min\{a_i\}$ adequado, gera-se um problema de transporte equivalente, onde a possibilidade de degeneração é totalmente eliminada.

De qualquer forma, existe um meio mais eficiente para evitar ciclagem: um tipo especial de método lexicográfico bastante simples, aplicável ao problema, baseado na manutenção do que se convencionou chamar árvores básicas viáveis fortes (SFT - Strong Feasable Tree), que são árvores cujos arcos degenerados estão todos direcionados para a raiz.

Pode-se identificar (ver Bazaraa[9]) uma correspondência direta entre arvores básicas fortes e bases lexicograficamente positivas.

Dada uma árvore básica inicial forte, uma simples consideração na regra para a seleção do arco a sair permite mantê-la com SFT. Na verdade, só ocorrerão problemas no caso de empate no teste do ratio, pois iremos consequentemente gerar arcos degenerados.

Neste caso, a idéia é começar a percorrer o ciclo (na direção do arco que entra), a partir no nó JUNCAO e escolher como arco a sair o último dos que determinarem o empate.

Observe na figura IV.5 que esta regra funciona: os arcos candidatos a sair estão em sentido contrário ao que se está percorrendo o ciclo; então, se o escolhido é o último dos que determinaram o empate, todos os demais, que passarão a ser degenerados, ficam direcionados para a raiz.

Quanto aos arcos já degenerados, se estão fora do ciclo, não sofrerão alteração nem de fluxo, nem de sentido. Caso algum deles pertença ao ciclo, temos duas situações:

- Se constitue a cadeia INDEST–JUNCAO, deixará de ser degenerado, pois passará a ter fluxo positivo, a menos que o incremento nos fluxos seja zero (pivoteamento degenerado).
- Se, ao contrário, compõe a cadeia INORIG–JUNCAO, então proporcionará a única possibilidade de pivoteamento degenerado (o arco pivot tem fluxo zero). Mas como o escolhido é o último, todos os anteriores (também pertencentes a mesma cadeia) ficam direcionados à raiz.

Desta forma constatamos que esta regra simples mantém a árvore básica forte.

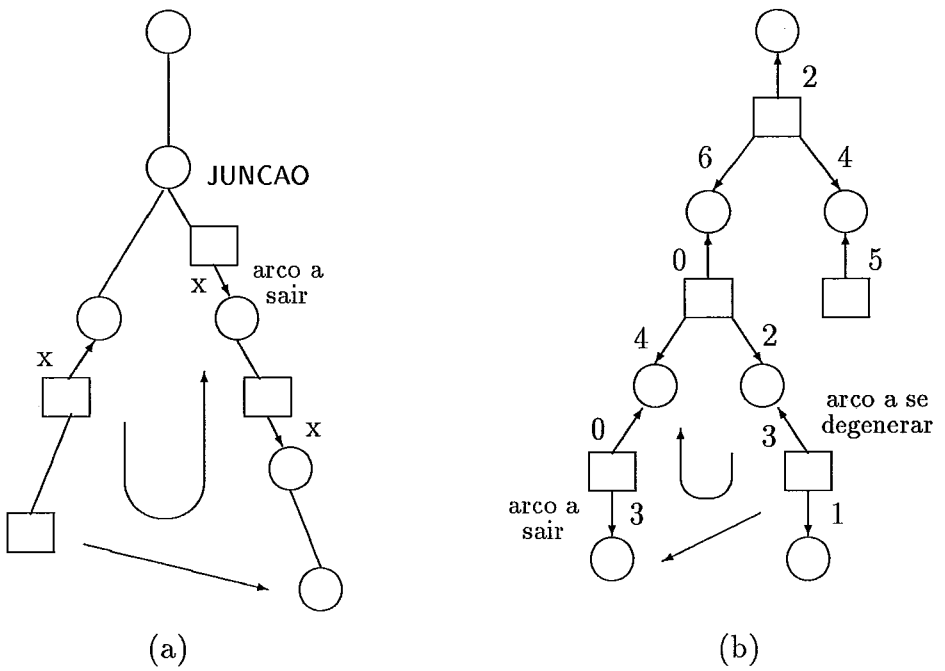


Figura IV.5: Atualização da árvore básica na presença de degeneração: (a) situação genérica, (b) um exemplo

IV.5 Solução Inicial

Todo método primal inicia com uma solução viável básica. Então precisamos gerar uma solução inicial do problema de transporte para dar partida ao nosso algoritmo.

Na seção anterior apresentamos uma solução viável, contudo não básica. Poderíamos, então, desenvolver um procedimento para convertê-la numa solução básica, o que seria perfeitamente possível.

No entanto, considerando-se também a necessidade de construção da própria árvore básica, ou melhor, das estruturas que a mantém, optamos pela utilização de outro método, a conhecida *regra do canto noroeste* (Northwest Corner Rule), que favorece tal procedimento.

IV.5.1 Primeiro Problema

Dos diversos problemas de transporte que serão resolvidos, ao longo do processo de solução do CWLP, apenas o primeiro, muito provavelmente $T(K_1 \cup K_2)$, é completamente novo, no sentido de que pouco se conhece sobre sua solução.

Uma solução viável para ele poderia ser construída da seguinte forma: toma-se um arco (i, j) qualquer e envia-se o máximo fluxo possível, isto é, o mínimo entre a capacidade de oferta a_i e a de demanda b_j. Se a oferta é menor que a demanda, o ponto origem i e todos os arcos a ele adjacentes são retirados do grafo, e a demanda b_j é reduzida de a_i unidades. Caso contrário, o ponto demanda j e todos os arcos a ele conectados é que são excluídos, bem como a oferta a_i é decrementada de b_j. O problema de transporte resultante é tratado de igual forma, até que, finalmente, $(n + m)$ arcos são preenchidos.

A regra do canto noroeste segue este mesmo raciocínio e adicionalmente impõe uma ordem para a consideração dos arcos, ou melhor, dos nós que os compõem: a ordem ascendente de numeração dos nós.

Assim, considerando-se como a'_i e b'_j os valores reduzidos de a_i e b_j (inicialmente toma-se $a'_i = a_i$ e $b'_j = b_j$), se o último arco preenchido foi (i, j), com $x_{ij} = \min(a'_i, b'_j)$, então faremos

$$\begin{aligned} a'_i &= a'_i - x_{ij} \\ b'_j &= b'_j - x_{ij} \end{aligned}$$

Se $(i, j) = (m, n)$, então $(m + n)$ arcos foram construídos. Senão, um dos dois casos ocorrem:

- Caso 1: $a'_i = 0$ $(b'_j \geq 0)$

  Então o próximo arco será $(i + 1, j)$ (que com certeza existe, pois ainda tem demanda a atender), para o qual faremos

  $$x_{i+1,j} = \min(a'_{i+1}, b'_j)$$

- Caso 2: $a'_i > 0$ $(b'_j = 0)$

  Neste caso o arco $(i, j + 1)$ existe, pois ainda temos capacidade de atendimento, e será o próximo a ser preenchido com

  $$x_{i,j+1} = \min(a'_i, b'_{j+1})$$

Note que, quando no cálculo de x_{ij} acontecer $a'_i = b'_j$, na próxima iteração teremos um arco degenerado. Este caso deve ser considerado, se se pretende construir uma árvore inicial forte.

Na verdade, não apenas os fluxos básicos precisam ser definidos, como também toda a estrutura árvore básica em si.

O bloco abaixo adapta a regra do canto noroeste ao nosso contexto, visando à implementação das estruturas que armazenam as informações da árvore.

Passo 0:

$i = 0 \quad j = 0$

$v_j = 0 \quad d_j = 0$

$x_{ij} = 0 \quad a'_0 = 0$

Passo 1:

$a'_i = a'_i - x_{ij}$

$b'_j = b'_j - x_{ij}$

Passo 2:

se $(a'_i = 0)$ e $(i \neq m)$ então

$i = i + 1$

$x_{ij} = \min(a'_i, b'_j)$

$P(i) = j$

$X(i) = x_{ij}$

$U(i) = c_{ij} - v_j$

$D(i) = d_j + 1$

$W(i) = i + 1$

volte a 1

Passo 3:

se $(j \neq n)$ então

$j = j + 1$

$x_{ij} = \min(a'_i, b'_j)$

$P(j) = i$

$X(j) = x_{ij}$

$v_j = c_{ij} - U(i)$

$d_j = D(i) + 1$

Volte a 1

Sobre este procedimento cabem algumas observações:

1. A inicialização $a'_0 = 0$ força que o nó raiz seja $j = 0$.

2. Considere que numa iteração tenhamos $a'_i = b'_j$. Quer estejamos no passo 2 ou passo 3, teremos $x_{ij} = a'_i = b'_j$. Na iteração seguinte faremos $a'_i = b'_j = 0$ e, forçosamente, será executado o passo 2. Então o novo arco degenerado estará apontando para a raiz. Logo, constrói-se desta forma uma árvore básica inicial forte.

3. As variáveis indexadas v_j e d_j foram criadas porque os vetores U e D não possuem entradas para os nós destino. Note, entretanto, que um único par de variáveis v e d, sem indexação, resolveria o problema. Da mesma forma, os conjuntos x_{ij}, a'_i e b'_j podem ser substituídos por uma única variável global ao procedimento, como é feito no Apêndice B.

Vale ressaltar também que outros métodos podem ser empregados na geração de uma solução básica viável inicial.

Na regra do canto noroeste, a escolha dos arcos, sem nenhuma consideração quanto aos custos a eles associados, conduz rapidamente a uma solução inicial. Todavia, nada se pode garantir sobre a qualidade desta, que possivelmente estará muito distante da solução ótima.

Resultados contrários se esperam no método da matriz mínima ("Matrix Minimum Method"), pois, muito embora conduza a uma boa solução inicial, dado que em cada iteração o arco remanescente de menor custo é escolhido, certamente irá consumir tanto tempo quanto o próprio algoritmo em si.

Um método intermediário ("Modified Column(or Row) Minimum Method") que tende a produzir uma solução tão boa quanto a da matriz mínima e não tão computacionalmente dispendiosa, seria percorrer os nós destino (ou origem) em forma cíclica e selecionar o arco de custo mínimo entre os adjacentes ao nó em consideração.

Há ainda o método de aproximação de Vogel, que procede como a seguir: a cada nó origem são associados valores correspondentes ao primeiro e segundo menor custo dos arcos a ele adjacentes; o mesmo é feito com relação aos nós destino; em seguida, é determinado o nó que apresente a máxima diferença positiva entre os valores a ele associados; finalmente, o arco escolhido será aquele que gerou o menor dos valores deste nó.

Estes dois últimos métodos poderiam possivelmente conduzir a soluções iniciais melhores que aquela gerada pela regra do canto noroeste. Contudo, a construção da estrutura da árvore básica seria dificultada. Além do mais, considerando que só para o primeiro problema de transporte do CWLP geraremos uma solução inicial artificial, decidimos por não implementar estes outros métodos.

IV.5.2 Demais Problemas

No processo de solução do CWLP todos os problemas de transporte resolvidos são variações do primeiro, ou melhor, do mais recentemente solucionado. Assim, a própria solução ótima do último problema resolvido pode ser usada como base para a construção de uma solução viável do problema seguinte.

Tendo em vista que o processo de solução consiste em, a cada iteração, abrir ou fechar facilidades, então devemos considerar, consequentemente, as variações no problema de transporte quando um ponto origem é acrescentado ou excluído.

Um ponto origem pode ser considerado desativado, se toda sua capacidade de oferta é destinada ao nó demanda arificial. Isto pode ser conseguido associando-se aos caminhos que partem desta facilidade um custo suficientemente elevado, que torne todas estas ligações proibitivas.

Desta forma, a partir da solução ótima do problema de transporte $T(S)$, que associa os pontos origem em S aos pontos destino, pode-se construir uma solução básica viável para o problema modificado $T(S-i)$, no qual a facilidade i será desativada, fazendo $c_{ij} = \mathsf{INF}$ para cada arco (i,j) $j = 1,\ldots,n$, onde INF representa um valor suficientemente grande.

Por outro lado, a mudança nos custos de arcos (i,j) básicos implica, consequentemente, alterações nos custos reduzidos, ou melhor, nos potenciais dos nós i e/ou j.

Considere que a configuração atual da árvore básica seja a da figura IV.6 abaixo, onde i é o nó a ser desativado e j e l representam, genericamente, possíveis nós

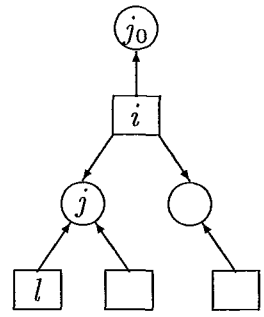


Figura IV.6: Desativação de um nó origem

sucessores de i, pontos destino e origem respectivamente.

A idéia é manter o potencial do nó j_0 e alterar o potencial de i e seus descendentes. Na verdade, apenas o potencial dos nós origem descendentes necessitam ser atualizados. Então se os valores modificados dos potenciais de nós e dos custos de arcos são diferenciados por uma linha (primo), temos

$$\begin{aligned} u'_i &= c'_{ij_0} - v_{j_0} \\ &= c'_{ij_0} - (c_{ij_0} - u_i) \\ u'_i &= u_i + (c'_{ij_0} - c_{ij_0}) \end{aligned}$$

e

$$\begin{aligned} u'_l &= c_{lj} - v'_j \\ &= (u_l + v_j) - (c'_{ij} - u'_i) \\ &= u_l + (u'_i - u_i) + (c_{ij} - c'_{ij}) \\ u'_l &= u_l + (c'_{ij_0} - c_{ij_0}) + (c_{ij} - c'_{ij}) \\ u'_l &= u_l + (c'_{ij_0} - c'_{ij}) + (c_{ij} - c_{ij_0}) \end{aligned}$$

Duas situações podem ocorrer

- Caso 1: *j_0 é o nó raiz*

  Neste caso temos $c'_{ij_0} = c_{ij_0} = 0$ e $c'_{ij} = \mathsf{INF}$ e portanto

$$\begin{aligned} u'_i &= u_i \\ u'_l &= u_l + (c_{ij} - \mathsf{INF}) \end{aligned}$$

- Caso 2: *j_0 não é o nó raiz*

  Como $c'_{ij_0} = c_{ij} = \mathsf{INF}$ então

$$\begin{aligned} u'_i &= u_i + (\mathsf{INF} - c_{ij_0}) \\ u'_l &= u_l + (c_{ij} - c_{ij_0}) \end{aligned}$$

Genericamente teremos para os dois caos

$$\begin{aligned} u'_i &= u_i + (\mathsf{INF} - c') \\ u'_l &= u_l + (c_{ij} - c') \end{aligned}$$

onde

$$c' = \begin{cases} \mathsf{INF}, & \text{se } j_0 = 0 \\ c_{ij_0}, & \text{se } j_0 \neq 0 \end{cases}$$

Observe que qualquer outro nó origem descendente de j terá o mesmo incremento $(c_{ij} - c')$ no seu potencial. Note ainda que o desenvolvimento foi realizado para um nó j qualquer, filho de i. Assim, de forma a processar todas as mudanças de potencial dos nós origem da árvore, com a desativção do ponto origem i, devemos fazer

se $P(i) = 0$
  então $c' = \mathsf{INF}$
  senão $c' = c_{iP(i)}$
$u_i = u_i + \mathsf{INF} - c'$
para todo nó j filho de i $(P(i) = j)$ faça
  $\mathsf{UINCR} = c_{ij} - c'$
  para todo nó l origem descendente de j faça
    $u'_l = u_l + \mathsf{UINCR}$

Quanto à ativação de uma facilidade, a inclusão de um ponto origem na árvore básica é simplesmente efetuada subordinando-o ao nó raiz e a este enviando toda sua capacidade de oferta. Consequentemente, obedecendo à condição de folgas complementares (III.10), ao potencial deste novo nó deve-se atribuir zero.

Na verdade, como todos os nós origem estão presentes no primeiro problema de transporte resolvido, $T(K_1 \cup K_2)$, eles poderão permanecer sempre compondo a árvore básica, mesmo desativados; desde que, neste caso, estejam sempre abastecendo apenas o nó demanda artificial, isto é, que o único arco básico incidente em cada um destes nós desativados seja aquele que os interliga à raiz.

Esta condição é efetivamente cumprida, se no procedimento "Inward Modified Most Negative Rule", para a escolha do arco a entrar na base, determine-se o mínimo de $(c_{ij} - u_i)$ somente entre os nós origem i não desativados.

Preferindo-se esta alternativa, os custos artificiais INF só necessitam ser atribuídos aos arcos (i, j) $j = 1, \ldots, n$ no momento em que o nó i estiver sendo desativado; depois, pode-se retornar aos verdadeiros valores c_{ij}. Desta forma, para tornar uma facilidade ativa, basta também considerá-la na avaliação do mínimo que determina cada novo arco a compor a árvore básica

Capítulo V

Um Algoritmo para o CWLP

Finalmente neste capítulo vamos apresentar o fruto concreto do nosso trabalho de pesquisa: um algoritmo heurístico para a resolução do problema de localização de facilidades capacitadas (CWLP - Capacitated Warehouse Location Problem).

O instrumental básico a ser empregado no desenvolvimento deste algoritmo compreenderá os testes e heurísticas discutidos no Capítulo 2 e as aproximações apresentadas no Capítulo 3.

A avaliação de sua eficiência será fundamentada nos resultados computacionais obtidos mediante aplicação a vários problemas-teste padrões, publicados na literatura internacional.

V.1 Apresentação do Algoritmo

Relembremos as definições das partições, propostas no Capítulo 2, para o conjunto I das possíveis localizações das facilidades. Quais sejam:

$$K_0 = \{i \in I \mid y_i = 0\} = \text{conjunto das facilidades fixadas como fechadas}$$
$$K_1 = \{i \in I \mid y_i = 1\} = \text{conjunto das facilidades fixadas como abertas}$$
$$K_2 = I - K_0 - K_1 = \text{conjunto das facilidades ainda não fixadas}$$

Como vimos, em um processo de solução onde se consideram a princípio indefinidos os estados de todos os armazéns e segue-se fixando alguns deles como abertos ou fechados, pode-se empregar, desde o início, o teste 1 e a heurística 2. Isto porque a inicialização ($K_2 = I$) garante a viabilidade do problema de transporte $T(K_1 \cup K_2)$, cuja solução é a base comparativa que compõe o teste 1 e a heurística 2, seja em sua versão original ou modificada.

Por outro lado, a aplicação do teste 2 e da heurística 1, em qualquer das formulações original ou modificada, somente é possível quando facilidades em número suficiente para atender a demanda tiverem sido ativadas, uma vez que, em ambos, o parâmetro de comparação é estabelecido a partir da solução do problema de transporte $T(K_1)$.

À primeira vista, parece uma idéia interessante gerar uma configuração viável inicial, fixando os estados de algumas facilidades, possivelmente já através da aplicação do teste 1 e heurística 2, para então definir os estados das demais, utilizando todos os testes e heurísticas disponíveis. Todavia, deve-se considerar que esta configuração inicial não pode ser qualquer: desde cedo precisa haver uma preocupação com a "otimalidade" do processo.

O algoritmo que vamos propor segue, aproximadamente, este raciocínio. Na verdade constitui-se de duas fases: a primeira, que procura determinar uma "boa" solução viável, e a segunda, que funciona como uma espécie de ajuste da anterior.

V.1.1 Primeira Fase

A primeira fase do algoritmo pode ser vista, resumidamente, como a execução cíclica dos dois passos a seguir:

1. Abrir o máximo possível de facilidades indicadas pelo teste 1 modificado
2. Fechar uma facilidade: aquela que a heurística 2 modificada apontar.

Como se vê, a preocupação desta primeira fase é ativar facilidades; e fazê-lo de modo coerente, pois a decisão de quais armazéns abrir é tomada considerando aqueles já ativados. Observe ainda a lógica desta sequência de passos: se o teste 1 não consegue indicar exatamente facilidades a abrir, a desativação de uma delas em (2) faz crescer seu poder de decisão. Logo, em algum momento, novos armazéns a instalar deverão ser escolhidos por (1).

Detalhando este procedimento, chegamos ao trecho abaixo, onde se fazem necessárias as definições dos seguintes conjuntos auxiliares

$\overline{K}_0$ = subconjunto das facilidades fechadas na primeira fase que serão reconsideradas na segunda.

$\overline{K}_2$ = subconjunto das facilidades indeterminadas para as quais a economia exata Δ_i foi calculada.

Algoritmo Fase 1

Passo 0: (* Inicialização *)

$K_0 = K_1 = \emptyset$

$\overline{K}_0 = \overline{K}_2 = \emptyset$

$K_2 = \{1, 2, \ldots, m\}$

$\Delta_i^L = 0 \;\; i \in K_2$

Calcular $W(K_1 \cup K_2)$

Passo 1: (* Aproximações *)

Calcular $\Delta_i^D = W^D(K_1 \cup K_2 - i) - W(K_1 \cup K_2) \;\; i \in K_2$

$\Delta_i^L = \max(\Delta_i^D, \Delta_i^L) \;\; i \in K_2$

Passo 2: (* Armazéns a abrir *)

$A = \{i \in K_2 \mid \Delta_i^L \geq f_i\}$

$K_1 = K_1 + A$

$K_2 = K_2 - A$

Passo 3: (* Teste de parada *)

Se $K_2 = \emptyset$, FASE 2

Passo 4: (* Possível arm. a fechar *)

$f_s - \Delta_s^L = \max_{i \in K_2}\{f_i - \Delta_i^L\}$

Passo 5: (* Economia Exata *)

Calcular $W(K_1 \cup K_2 - s)$

$\Delta_s = W(K_1 \cup K_2 - s) - W(K_1 \cup K_2)$

Passo 6: (* Abrir armazém ? *)

Se $\Delta_s \geq f_s$

$K_1 = K_1 \cup \{s\}$

$K_2 = K_2 - \{s\}$

$\overline{K}_2 = \overline{K}_2 - \{s\}$

voltar a 3

Passo 7: (* Fechar armazém ?*)

Se $(\Delta_s^L / \Delta_s \geq \epsilon_1)$ e $(f_s - \Delta_s \geq \max_{i \in \overline{K}_2 \cup \{s\}}\{f_i - \Delta_i\})$

$K_0 = K_0 + \{s\}$

$K_2 = K_2 - \{s\}$

$\overline{K}_2 = \overline{K}_2 - \{s\}$

Se $\Delta_s / f_s > \epsilon_2$, $\overline{K}_0 = \overline{K}_0 \cup \{s\}$

voltar a 1

Passo 8: (* Continua indeterminado *)

$\overline{K}_2 = \overline{K}_2 \cup \{s\}$

$\Delta_s^L = \Delta_s$

voltar a 4

Analisando a estruturação[1] apresentada, constata-se que os armazéns são instalados no passo 2, a partir do teste 1 modificado, enquanto que a facilidade s a fechar é escolhida, no passo 4, através da heurística 2 modificada.

Ambos, teste e heurística, são baseados em Δ_i^L, um limite inferior para Δ_i, determinado no passo 1 como a melhor aproximação até então obtida: a aproximação Δ_i^D (Mateus e Bornstein[41]) calculada na iteração corrente ou, possivelmente, a própria economia exata anteriormente avaliada para algumas facilidades $i \in \overline{K}_2$. (O conjunto $\overline{K}_2$ é construído no passo 8 como veremos).

Para a determinação inicial das aproximações Δ_i^L, torna-se indispensável a resolução do problema $T(K_1 \cup K_2)$ no passo 0. Este é o primeiro problema de transporte, para o qual, portanto, devemos construir uma solução inicial.

Quanto aos problemas seguintes $T(K_1 \cup K_2 - s)$, sua resolução, no passo 5, fornece o valor $W(K_1 \cup K_2 - s)$, que é parte integrante do cálculo para as aproximações Δ_i^L na próxima execução do passo 1. Ou mesmo que não ocorra uma próxima iteração, este pode constituir o valor final da fase 1.

Cada novo problema $T(K_1 \cup K_2 - s)$ é na verdade uma redução do anterior $T(K_1 \cup K_2)$ e pode ser resolvido, como vimos na subseção IV.5.2, a partir da solução ótima deste. Tal inicialização contribui para que a nova solução ótima seja encontrada em poucos pivoteamentos, principalmente no começo da primeira fase, quando a desativação de um armazém não implica tantas mudanças na estrutura da árvore básica.

No passo 6, pode parecer estranho que tentemos abrir a facilidade escolhida para ser desativada[2]. A utilidade maior deste passo, que não incorre em nenhum esforço computacional extra, uma vez que Δ_s pode ser calculado diretamente no passo 5, está na parte final da fase 1, quando possivelmente a heurística 2 já não funcione tão bem. Isto porque a escolha entre uma ou outra facilidade deve estar difícil, especialmente se se baseia em critérios aproximados.

Se a facilidade escolhida para fechar deveria, comprovadamente, ser aberta, então a decisão contrária é adotada, isto é, este armazém é de fato instalado. Por conseguinte, um outro a fechar deve ser determinado.

Do contrário, vamos ainda considerar a qualidade da aproximação que baseou a escolha desta facilidade. Este é o significado do primeiro teste do passo 7:

[1]Quando uma estrutura condicional Se abranger diversos comandos, estes aparecerão a seguir indentados; no caso de compreender apenas um, tal estrutura constituirá um comando único disposto na mesma linha.

[2]Observe que este teste ainda faz parte da heurística 2 modificada

se o parâmetro ϵ_1 representa o quão próxima se espera a aproximação de seu verdadeiro valor, então deve-se exigir a ocorrência da condição $\Delta_s^L/\Delta_s < \epsilon_1$. Além disso, estabelecendo-se o teste $f_s - \Delta_s \geq \max_{i \in \overline{K}_2 \cup \{s\}}\{f_i - \Delta_i\}$, a decisão de se desativar a facilidade s, escolhida no passo 4 com base na heurística 2 modificada, estará também de acordo com a heurística 2 original, avaliada entre os armazéns i cujas variações Δ_i exatas estão disponíveis.

Não sendo verificados estes critérios adicionais de seleção, o estado da facilidade s é ainda mantido como indeterminado, e uma outra facilidade a fechar deverá ser escolhida. Antes, porém, no passo 8, Δ_s^L é atualizado ($\Delta_s^L = \Delta_s$) e o conjunto $\overline{K}_2$ ganha um novo elemento ($\overline{K}_2 = \overline{K}_2 \cup \{s\}$). Constate assim que os elementos de $\overline{K}_2$ são armazéns indicados para fechar, contudo através de uma aproximação considerada não satisfatória e que, devido a isso, decidiu-se manter seu estado indeterminado.

Esta volta ao passo 4 para a escolha de uma outra facilidade a fechar, sem qualquer decisão sobre o estado daquela anteriormente tomada, trará custo adicional de computação para o algoritmo, especialmente se considerarmos que um novo problema de transporte irá ser resolvido. Por este motivo, não se deve exigir muito da qualidade da aproximação, ou seja, tomar o valor de ϵ_1 muito próximo a unidade.

Por outro lado, mesmo se decidindo no passo 7 pela desativação da facilidade s, ela poderá ainda ser reservada para posterior consideração na segunda fase, caso satisfaça o teste $\Delta_s/f_s > \epsilon_2$.

O parâmetro ϵ_2 representa uma cota inferior do percentual que se estima ser a economia atual Δ_s da economia final $\Delta_s^\star = W(K_1^\star - i) - W(K_1^\star)$, onde $K_1^\star$ representa a configuração final de K_1.

Então considerando que $\Delta_s > \epsilon_2 \Delta_s^\star$, caso ocorra $\Delta_s/f_s > \epsilon_2$, existe a possibilidade de termos

$$\frac{\Delta_s}{\epsilon_2} > \Delta_s^\star > f_s$$

e, por conseguinte, o armazém s deveria estar aberto, isto é, a decisão da fase 1 necessitaria ser reconsiderada.

Ao contrário, se $\Delta_s/f_s \leq \epsilon_2$, então certamente

$$\Delta_s^\star < \frac{\Delta_s}{\epsilon_2} \leq f_s$$

de modo que o armazém s deve permanecer desativado, o que indica uma decisão já corretamente tomada na primeira fase.

Observe, sob outro aspecto, que ϵ_2 estima o quanto Δ_s ainda poderia crescer até o final do processo de solução. Tomando-se $\epsilon_2 = 0$, isto é, considerando-se

que Δ_s pode crescer indefinidamente, teremos $\overline{K}_0 = K_0$: assim toda decisão de fechar um armazém na primeira fase seria reavaliada na segunda.

O melhor seria que ϵ_2 fosse não decrescente ao longo do processo de solução e individualizado para cada facilidade. Entretanto, a avaliação de um valor adequado é extremamente difícil, senão imposível. Uma opção seria usar um valor constante, estimado inferiormente, para garantir a abrangência das possíveis situações de melhora na solução da fase 1. Note que, tomado como constante, ϵ_2 estabelece um ponto a partir do qual as decisões de desativar facilidades serão reavaliadas, pois a partir de então estas facilidades estarão compondo $\overline{K}_0$.

Na verdade, o objetivo maior deste teste é reduzir o número de armazéns em $\overline{K}_0$, que serão reconsiderados na segunda fase. Desta forma, um outro teste, por exemplo que determinasse o número máximo destes armazéns, poderia ser opcionalmente empregado.

A primeira fase termina, no passo 3, quando os estados de todos os armazéns estão decidos, pelo menos provisoriamente. A convergência desta fase pode ser facilmente demonstrada se constatarmos que a cada iteração o estado de no mínimo uma facilidade é decidido, ou seja, K_2 perde pelo menos um elemento, e portanto o teste de parada ($K_2 = \emptyset$) será fatalmente satisfeito.

De outra forma, poder-se-ia ter antecipado este final para o momento em que a viabilidade fosse alcançada ($\sum_{l \in K_1} a_l \geq \sum_{j \in J} b_j$) e então desenvolvido uma segunda fase semelhante à primeira, agora baseada no teste 2 e heurística 1, intercalando a desativação de diversas facilidades e a ativação de uma.

Esta opção foi considerada, e inclusive implementada, para resolver problemas de pequeno porte. Entretanto não apresentou resultados tão bons, nem do ponto de vista numérico, nem de esforço computacional, quanto os obtidos mantendo-se a primeira fase desta forma e construindo-se uma segunda fase como apresentada na seção seguinte.

Algumas explicações para estes resultados seriam:

1. A avaliação de $\overline{\Omega}_i$ (Akinc e Khumawala[3]), aproximação "mais estreita" que Ω_i^D (Mateus e Bornstein[41]), impõe a resolução de um problema da mochila para cada $i \in K_2$, o que pode demandar tempo de computação considerável.

2. Além disso, mesmo o limite $\overline{\Omega}_i$ não constitui uma boa aproximação para Ω_i logo após atingida a viabilidade. Desta forma vai reduzir em muito o poder de decisão da heurística 1 e do teste 2.

V.1.2 Segunda Fase

A segunda fase objetiva melhorar a solução obtida na primeira, através da reavaliação de decisões anteriormente tomadas quanto aos estados de algumas facilidades. Constitui, mais especificamente, um procedimento heurístico similar ao VSM (Vertex Method Substitution) de Jacobsen[33], intercalando iterações ADD e DROP, da seguinte forma:

1. Acrescenta à solução corrente uma facilidade r indicada pela heurística 1 modificada, dentre os elementos de $\overline{K}_0$
2. Avaliando a nova configuração, decide por retirar a própria facilidade r ou alguma das anteriormente abertas

Como descrito, este procedimento permite exclusivamente trocar uma facilidade por outra. Diferentes estratégias que possibilitem a troca de uma por várias, várias por uma ou, mais genericamente, várias por várias, também podem ser desenvolvidas. A opção de trocar uma por várias será também abordada no final desta seção. Quanto às demais, observa-se que o acréscimo de diversas facilidades à solução gera, normalmente, um contexto semelhante ao da primeira fase, de modo que as decisões seriam novamente as mesmas. Por esse motivo não as aprofundaremos.

Na verdade o procedimento que vamos apresentar até permite que diversas facilidades sejam acrescidas à solução, desde que certas condições aconteçam, como veremos a seguir.

Algoritmo Fase 2

Passo 0: (* Teste de parada *)
Se $\overline{K}_0 = \emptyset$, PARE

Passo 1: (* Aproximações *)
Calcular $\Omega_i^D = W(K_1) - W^D(K_1 \cup i) \;\; i \in \overline{K}_0$

Passo 2: (* Armazém a acrescentar *)
$\Omega_r^D - f_r = \max_{i \in \overline{K}_0} \{\Omega_i^D - f_i\}$
$\overline{K}_0 = \overline{K}_0 - \{r\}$

Passo 3: (* Economia Exata *)
Calcule $W(K_1 \cup r)$
$\Delta_r = W(K_1) - W(K_1 \cup r)$

Passo 4: (* Abrir armazém ? *)
Se $\Delta_r \geq f_r$
$K_1 = K_1 \cup \{r\}$
$K_0 = K_0 - \{r\}$
voltar a 0

Passo 5: (* Aproximações *)
Calcular $\Delta_i^D = W^D(K_1 \cup r - i) - W(K_1 \cup r)$ $i \in K_1$
$\Delta_i^L = \Delta_i^D$ $i \in K_1$

Passo 6: (* Possível arm. a fechar *)
$f_s - \Delta_s^L = \max_{i \in K_1}\{f_i - \Delta_i^L\}$

Passo 7: (* Voltar a acrescentar *)
Se $(f_s - \Delta_s^L) \leq (f_r - \Delta_r)$, voltar a 0

Passo 8: (* Economia Exata *)
Calcule $W(K_1 \cup r - s)$
$\delta_s = W(K_1 \cup r - s) - W(K_1)$

Passo 9: (* Efetuar a troca *)
Se $\delta_s < f_s - f_r$
$K_1 = K_1 - \{s\} \cup \{r\}$
$K_0 = K_0 \cup \{s\} - \{r\}$
voltar a 0

Passo 10: (* Tentar nova troca *)
$\Delta_s^L = \delta_s + \Delta_r$
voltar a 6

A iteração ADD compreende os passos de 1 a 4. A facilidade r a ser acrescentada é escolhida no passo 2, pela aplicação da heurística 1 modificada, com base no cálculo de Ω_i^D.

No passo 4, verifica-se se esta facilidade pode ser incluída à solução, melhorando-a, sem que nenhuma precise ser retirada. Neste caso, ela certamente será aberta e outra facilidade em $\overline{K}_0$ é considerada, criando a possibilidade de trocas do tipo várias por uma ou várias por várias.

O teste que permite esta verificação não incorre em qualquer esforço adicional de computação, já que o valor $W(K_1 \cup r)$ determinado é necessário à iteração DROP seguinte como base de cálculo das aproximações Δ_i^L.

Quanto às aproximações para Ω_i, podemos utilizar tanto Ω_i^D (Mateus e Bornstein[41]), quanto $\overline{\Omega}_i$ (Akinc e Khumawala[3]). Verifica-se, no entanto, que ambas proporcionam praticamente os mesmos resultados: isto porque, como K_1 já representa uma boa configuração, teremos Ω_i^D muito próximo a $\overline{\Omega}_i$, senão igual (verifique seção III.2). De qualquer forma, como a heurística 1 modificada, neste caso, indica apenas a ordem em que as facilidades serão consideradas, podemos decidir pela aplicação de Ω_i^D, tendo em vista que se trata de uma aproximação bem mais simples de ser calculada.

A iteração DROP abrange os passos de 5 a 10. A escolha do possível armazém s a fechar, entre os atualmente abertos, ocorre no passo 6, numa indicação da heurística 2 modificada, que portanto deve ser aplicada aos elementos de K_1.

Entretanto, se tal facilidade s, mais provável de ser trocada com r, possui menor potencial para ser fechada que esta, ou seja, satisfaz no passo 7

$$f_s - \Delta_s \leq f_s - \Delta_s^L \leq f_r - \Delta_r$$

então nenhuma troca com r trará economia. Assim a facilidade r deve permanecer desativada (continuar em K_0) e uma outra em $\overline{K}_0$ deve ser considerada.

Mesmo que a desigualdade acima não se verifique, a troca só será realmente efetivada caso esta decisão, de fato, venha a melhorar a solucão corrente, isto é, se $Z(K_1 \cup r - s) < Z(K_1)$, ou ainda se

$$\begin{aligned} W(K_1 \cup r - s) + F(K_1 \cup r - s) &< W(K_1) + F(K_1) \\ W(K_1 \cup r - s) - W(K_1) &< F(K_1) - F(K_1 \cup r - s) \\ \delta_s &< f_s - f_r \end{aligned}$$

É o que faz o passo 9 quando efetua a atualização $K_1 = K_1 \cup \{r\} - \{s\}$.

Do contrário, uma nova tentativa de troca com r será considerada. Por isso o passo 10 estabelece a volta para a escolha de uma nova facilidade a fechar.

A atualização $\Delta_s^L = \delta_s + \Delta_r$ é apropriada, uma vez que

$$\begin{aligned} \delta_s + \Delta_r &= W(K_1 \cup r - s) - W(K_1) + W(K_1) - W(K_1 \cup r) \\ &= W(K_1 \cup r - s) - W(K_1 \cup r) \\ &= \Delta_s \end{aligned}$$

E, além disso, tal atribuição faz com que a facilidade s não seja novamente considerada, pois, como $f_s - \delta_s \leq f_r$ (negação do passo 9), temos

$$f_s - \Delta_s^L = f_s - \delta_s - \Delta_r \leq f_r - \Delta_r$$

de modo que uma nova escolha de s no passo 6 seria descartada no passo 7.

De outra forma, poderia ser criado um conjunto para manter as facilidades que não devem ser retiradas.

A segunda fase termina finalmente no passo 0, quando todas as facilidades em $\overline{K}_0$ tiverem sido consideradas. É fácil verificar que o teste de parada ($\overline{K}_0 = \emptyset$) será eventualmente satisfeito, uma vez que a cada iteração ADD a facilidade escolhida é retirada de $\overline{K}_0$.

Adicionalmente, para permitir, na fase 2, a troca de uma facilidade por várias, poder-se-ia, ao sair no passo 9, de posse de uma nova solução (melhorada), em vez de retornar logo ao passo 0, aplicar um procedimento DROP muito semelhante a este primeiro.

Passo 11: (* Aproximações *)
Calcular $\Delta_i^D = W^D(K_1 - i) - W(K_1) \;\; i \in K_1$
$\Delta_i^L = \Delta_i^D \;\; i \in K_1, \; i \neq r$
$\Delta_r^L = \max(\Delta_r^D, \Delta_r)$

Passo 12: (* Possível arm. a fechar *)
$f_s - \Delta_s^L = \max_{i \in K_1 - r}\{f_i - \Delta_i^L\}$

Passo 13: (* Voltar a acrescentar *)
Se $(f_s - \Delta_s^L) \leq (f_r - \Delta_r^L)$, voltar a 0

Passo 14: (* Economia Exata *)
Calcule $W(K_1 - s)$
$\Delta_s = W(K_1 - s) - W(K_1)$

Passo 15: (* Efetuar retirada *)
Se $\Delta_s < f_s$
$K_1 = K_1 - \{s\}$
$K_0 = K_0 \cup \{s\}$
voltar a 11

Passo 16: (* Tentar outra retirada *)
$\Delta_s^L = \Delta_s$
voltar a 12

Na verdade, esta segunda parte da iteração DROP pode ser incorporada à primeira, a partir de algumas adaptações, de modo a constituir um único laço. Optou-se, todavia, por apresentá-las separadamente não só para facilitar a exposição, como também para sugerir que esta segunda parte não necessariamente precisa ser empregada. É claro que, se se decide implementar ambas, um único laço deve ser construído.

As duas opções, isto é, só a primeira parte da iteração DROP ou ambas, foram consideradas no nosso estudo. Verificou-se que, para apenas dois dos problemas-teste (descritos na seção a seguir) a segunda opção proporcionou melhor resultado numérico. Por outro lado, como era de se esperar, a primeira opção requereu menor esforço computacional em todos os testes. Desta forma, parece melhor considerarmos apenas a troca simples entre uma facilidade e outra. Por isso, no Apêndice A, a versão apresentada implementa somente esta opção.

Vale ressaltar aqui que todos os procedimentos apresentados só efetuam uma troca se ela realmente proporciona alguma economia. Outras estratégias que, mesmo sem vislumbrar uma possibilidade imediata de melhora, permitissem o acréscimo de uma facilidade à solução corrente para futura avaliação, poderiam também ser empregadas. Contudo, estas opções conduzem, em geral, a métodos enumerativos, que não constituem a proposta do nosso trabalho.

V.2 Problemas Teste

Os problemas aos quais aplicaremos o algoritmo compreendem exemplos de CWLP, disponíveis na literatura internacional, de uso frequente entre pesquisadores, para comparação, mais realística, da eficiência de seus algoritmos.

Na verdade, são diferentes grupos de dados, mais especificamente custos de transporte e demandas de consumidores, que geram uma vasta gama de problemas, a partir da variação de alguma das características das facilidades (custo fixo e/ou capacidades), conforme tabela abaixo:

GRUPO PROB.	No. POSSÍVEIS LOCALIZAÇÕES (m)	No. CENTROS DE DEMANDA (n)	CAPACIDADE DOS ARMAZÉNS (unid 1000)	CUSTO FIXO DOS ARMAZÉNS ($ 1000)
IV	16	50	5	7.5 / 12.5 / 17.5 / 25.0
V	16	50	10	17.5
VI	16	50	15	7.5 / 12.5 / 17.5 / 25.0
VIII	25	50	5	7.5 / 12.5 / 17.5 / 25.0
IX	25	50	15	7.5 / 12.5 / 17.5 / 25.0
XI	50	50	5	7.5 / 12.5 / 17.5 / 25.0
XII	50	50	15	7.5 / 12.5 / 17.5 / 25.0
A	100	1000	8/10/12/14	Randômico
B	100	1000	5/ 6/ 7/ 8	Randômico
C	100	1000	5/5.75/6.5/7.25	Randômico

Tabela V.1: Descrição dos Poblemas

Os grupos de problemas IV a XII são originários dos dados formalizados por Kuehn e Hamburger[38], sendo que IV a IX seguem a mesma numeração empregada por Akinc e Khumawala[3]. Os problemas que faltam na sequência são de dimensão bastante reduzida (I,II e III), ou não capacitados (III, VII, X e XIII), que, por isso, não os consideraremos.

Já os grupos A, B e C compreendem problemas gerados randomicamente por Beasley[10]. Nestes casos, também os custos fixos são aleatórios.

Como é de costume usaremos a notação (P-x) para indicar a variação x do grupo de problemas P. Assim, por exemplo, IV-1 representa o primeiro problema do grupo IV, ou seja, aquele onde o custo fixo é \$7500. Similarmente A-3 especifica o problema do grupo A onde os armazéns têm capacidade 12000.

Os dados que compõem todos estes problemas foram adquiridos junto a OR-Library (Beasley[12]), uma biblioteca eletrônica organizada no Imperial College, que contém vários problemas-teste relativos a diferentes áreas da Pesquisa Operacional.

Apenas a título de curiosidade, ou ainda, para melhor compreender a estrutura destes problemas, vamos explicar o modo como foram gerados.

Nos problemas originários de Khuen e Hamburger, os 50 centros de consumo representam cidades dos Estados Unidos, onde a cada 1000 habitantes[3] fez-se corresponder uma unidade de demanda. Algumas dessas cidades (16, 25 ou 50 delas) poderão ser tomadas também como possíveis localizações de armazéns, e uma delas foi fixada para sediar a fábrica (centro produtor).

Os custos de transporte aos centros consumidores foram admitidos como proporcionais às distâncias[4] que as mercadorias deveriam percorrer. Assim, o custo unitário de transporte c_{ij} entre a facilidade i e o centro de demanda j foi calculado como

$$c_{ij} = 0.0125 d_{0i} + 0.0250 d_{ij}$$

onde

$$\begin{aligned} d_{0i} &= \text{distância entre a fábrica e o armazém } i \\ d_{ij} &= \text{distância entre o armazém } i \text{ e o consumidor } j \end{aligned}$$

Quanto aos problemas gerados por Beasley, os centros de oferta e demanda foram alocados em um quadrado de lado 1000 e o custo unitário de transporte entre um cliente e um consumidor foi então tomado como propocional à distância euclideana entre eles, por um fator multiplicativo aleatório no intervalo [1.00,1.25]. A demanda de cada consumidor foi também determinada randomicamente, como um inteiro entre 1 e 100.

Para gerar os custos fixos, Beasley procedeu da seguinte forma: primeiro, especificou um número desejado P_0 de facilidades abertas na solução ótima (5,

[3]Fonte: *The World Almanac*, 1960; New York World-Telegram and the Sun, Nova York, 1960
[4]Fonte: *Rand McNally-Cosmopolitan World Atlas*, Rand McNally & Co., Chicago, 1951

10 e 15 respectivamente para A, B e C); a seguir, formou randomicamente um conjunto de (P_0+1) facilidades e avaliou o custo $C1$ de atender a demanda, alocando cada consumidor ao seu fornecedor "mais barato"; então, retirando (aleatoriamente) duas facilidades deste conjunto, determinou o custo semelhante $C2$, agora considerando apenas as (P_0-1) facilidades restantes; o custo fixo de cada facilidade foi então tomado como $(C2-C1)/2$ multiplicado por um número aleatório do intervalo [0.75,1.25]. A idéia subjacente a este procedimento é assegurar que na solução ótima existam mais ou menos P_0 facilidades abertas.

V.3 Resultados Computacionais

Os resultados que apresentaremos nesta seção foram obtidos em um microcomputador PC-AT 486, pela implementação em FORTRAN-Lahey do algoritmo apresentado, onde na segunda fase apenas a primeira parte da iteração DROP foi considerada.

Para a resolução do problema de transporte foi programado o procedimento descrito no Capítulo 4. Ambas as implementações, do algoritmo em si e do módulo de transporte, encontram-se nos Apêndices A e B, respectivamente.

As tabelas a seguir apresentam as soluções obtidas para os diversos problemas descritos na tabela V.1, organizados de acordo com suas dimensões.

Cada uma dessas tabelas (V.2 e V.3) apresenta a solução ótima do problema, em termos de número do facilidades abertas e custo total de instalação e transporte, as soluções determinadas pela primeira e segunda fases do algoritmo e as diferenças percentuais entre a solução de cada fase e a solução ótima.

As diferenças percentuais são calculadas pelas fórmulas

$$\mathsf{DIF}^1 = \frac{Z^1 - Z^*}{Z^*} \times 100$$

$$\mathsf{DIF}^2 = \frac{Z^2 - Z^*}{Z^*} \times 100$$

onde Z^1 e Z^2 representam, respectivamente, as soluções da primeira e segunda fases, obtidas tomando $\epsilon_1 = 0.70$ e $\epsilon_2 = 0.05$ na fase 1.

Analisando-se a coluna DIF^2, constata-se que nosso algoritmo obteve resultados consideráveis, tendo em vista que se trata de um procedimento heurístico.

Para os problemas pequenos (IV a XII) em apenas três as soluções obtidas não são as ótimas, mesmo assim muito próximas (diferenças percentuais de

PROB.	SOLUÇÃO ÓTIMA		SOLUÇÃO 1ª FASE			SOLUÇÃO 2ª FASE		
	No. ARM	CUSTO (Z^*)	No. ARM	CUSTO (Z^1)	DIF[1] (%)	No. ARM	CUSTO (Z^2)	DIF[2] (%)
IV–1	13	1040444.375	13	1040444.375	0.00	13	1040444.375	0.00
IV–2	12	1098000.450	12	1098000.450	0.00	12	1098000.450	0.00
IV–3	12	1153000.450	12	1153000.450	0.00	12	1153000.450	0.00
IV–4	12	1235500.450	12	1235500.450	0.00	12	1235500.450	0.00
V–3	8	1025208.225	8	1025208.225	0.00	8	1025208.225	0.00
VI–1	11	932615.750	11	932615.750	0.00	11	932615.750	0.00
VI–2	9	977799.400	9	977799.400	0.00	9	977799.400	0.00
VI–3	7	1014062.050	7	1014062.050	0.00	7	1014062.050	0.00
VI–4	5	1045650.250	5	1053207.550	0.72	5	1045650.250	0.00
VIII–1	17	838499.288	17	838499.288	0.00	17	838499.288	0.00
VIII–2	14	910889.563	14	910889.563	0.00	14	910889.563	0.00
VIII–3	14	975889.563	14	975889.563	0.00	14	975889.563	0.00
VIII–4	13	1069369.525	14	1073389.563	0.37	14	1073389.563	0.37
IX–1	15	796648.438	15	796648.438	0.00	15	796648.438	0.00
IX–2	11	855733.500	10	856496.150	0.09	11	855733.500	0.00
IX–3	8	896617.538	8	896617.538	0.00	8	896617.538	0.00
IX–4	7	946051.325	7	946051.325	0.00	7	946051.325	0.00
XI–1	17	826124.713	17	826124.713	0.00	17	826124.713	0.00
XI–2	15	901377.213	15	901377.213	0.00	15	901377.213	0.00
XI–3	14	970567.750	15	971377.213	0.08	15	971377.213	0.08
XI–4	13	1063356.488	13	1069199.475	0.55	13	1066450.575	0.29
XII–1	15	793439.563	15	793439.563	0.00	15	793439.563	0.00
XII–2	11	852524.625	10	853287.275	0.09	11	852524.625	0.00
XII–3	9	895302.325	9	895302.325	0.00	9	895302.325	0.00
XII–4	7	946051.325	7	946051.325	0.00	7	946051.325	0.00

Tabela V.2: Resultados para os problemas de pequeno porte

PROB.	SOLUÇÃO ÓTIMA		SOLUÇÃO 1ª FASE			SOLUÇÃO 2ª FASE		
	No. ARM	CUSTO (Z^*)	No. ARM	CUSTO (Z^1)	DIF[1] (%)	No. ARM	CUSTO (Z^2)	DIF[2] (%)
A–1	7	19240822.449	7	19403570.393	0.85	7	19261175.421	0.11
A–2	6	18438046.543	6	18955703.325	2.81	6	18455175.288	0.09
A–3	5	17765201.949	5	18521532.393	4.26	5	17825226.586	0.33
A–4	4	17160439.012	4	18452408.399	7.53	4	17160439.013	0.00
B–1	11	13656379.578	10	13869130.115	1.56	11	13656379.578	0.00
B–2	9	13361927.449	9	13600525.405	1.79	9	13413189.773	0.38
B–3	8	13198556.434	9	13389064.462	1.44	9	13293291.442	0.72
B–4	7	13082516.496	7	13185657.689	0.79	7	13097158.330	0.11
C–1	11	11646596.974	11	11754305.070	0.92	11	11646596.974	0.00
C–2	10	11570340.289	10	11701610.459	1.13	10	11570340.289	0.00
C–3	9	11518743.744	9	11655369.686	1.19	9	11539339.572	0.18
C–4	9	11505767.394	9	11647844.397	1.23	9	11535255.510	0.26

Tabela V.3: Resultados para os problemas de grande porte

0.08%, 0.29% e 0.37%). Observe ainda que já na primeira fase a maioria dos problemas (76% deles) estava definitivamente resolvida. Isto demonstra que já esta primeira fase gera boas soluções.

Para os problemas grandes (grupos A, B e C), as diferenças percentuais permaneceram reduzidas (média de 0.18%). Algumas das soluções encontradas são ótimas e mesmo a única que gerou uma diferença superior a 0.4% (problema B-3 com 0.72%) seria melhorada caso aplicássemos também a segunda parte da iteração DROP, gerando uma diferença de apenas 0.33% (Com isso a média cairia para menos de 0.15%).

Também para estes problemas as soluções da primeira fase já podem ser consideradas satisfatórias, especialmente para os grupos B e C, onde as diferenças ficaram na faixa de 1%. Mesmo assim, a aplicação da segunda fase conseguiu reduzi-las significativamente.

No geral, considerando todos os 37 problemas resolvidos, a diferença percentual média ficaria inferior a 0.08%, para a versão do algoritmo constituída apenas pela primeira parte da iteração DROP, e inferior a 0.07%, tomando-se os resultados obtidos com a outra versão.

Quanto aos tempos de computação, a eficiência do algoritmo no que se refere a este aspecto pode ser estimada analisando-se as tabelas V.4 e V.5, nas quais os problemas foram organizados segundo grupo e variação. Nessas tabelas está

TEMPOS DE COMPUTAÇÃO (s)					
GRUPO PROB.	FASE	VARIAÇÕES PROBLEMAS 1	2	3	4
IV	1ª	0.04	0.06	0.05	0.06
	2ª	0.01	0.00	0.00	0.00
	TOT.	0.05	0.06	0.05	0.06
V	1ª			0.05	
	2ª			0.01	
	TOT.			0.06	
VI	1ª	0.04	0.03	0.06	0.09
	2ª	0.02	0.02	0.05	0.02
	TOT.	0.06	0.05	0.11	0.11
VIII	1ª	0.10	0.15	0.16	0.21
	2ª	0.07	0.01	0.06	0.01
	TOT.	0.17	0.16	0.22	0.22
IX	1ª	0.08	0.11	0.13	0.14
	2ª	0.08	0.05	0.09	0.02
	TOT.	0.16	0.16	0.22	0.16
XI	1ª	0.46	0.52	0.63	0.70
	2ª	0.36	0.36	0.25	0.40
	TOT.	0.82	0.88	0.88	1.10
XII	1ª	0.41	0.44	0.48	0.48
	2ª	0.14	0.11	0.07	0.12
	TOT.	0.55	0.55	0.55	0.60

Tabela V.4: Tempos de computação para os problemas de pequeno porte

totalizado, considerando-se ambas as fases do algoritmo, o tempo gasto (em segundos) para a resolução de cada problema, não incluído o tempo inicial de leitura dos arquivos de dados.

A divulgação destes tempos de processamento constitui o meio mais comumente utilizado para mensurar o esforço computacional dispendido. Verifique então nos dados fornecidos o bom comportamento do algoritmo: para os problemas de pequeno porte, os tempos de computação foram ínfimos (menos de $1\,s$), e mesmo para os problemas maiores, os resultados foram gerados em tempo bastante reduzido (média inferior a $1\,min\ 10\,s$).

Mesmo reconhecendo a debilidade de qualquer comparação entre algoritmos a partir de tempos de computação, tendo em vista as grandes diferenças de máquinas empregadas e até mesmo de formas de implementação, apresentamos a seguir

TEMPOS DE COMPUTAÇÃO (s)					
GRUPO PROB.	FASE	VARIAÇÕES PROBLEMAS 1	2	3	4
A	1ª	51.51	51.88	49.14	50.13
	2ª	13.19	10.90	11.16	7.43
	TOT.	64.70	62.78	60.30	57.56
B	1ª	50.76	52.08	54.73	52.55
	2ª	21.80	32.34	17.99	22.65
	TOT.	72.56	84.42	72.72	75.20
C	1ª	51.84	52.24	52.07	51.21
	2ª	19.07	16.14	18.73	13.82
	TOT.	70.91	68.38	70.80	65.03

Tabela V.5: Tempos de computação para os problemas de grande porte

as tabelas V.6 e V.7, que relacionam o algoritmo proposto aos algoritmos enumerativo de Beasley[10] e heurístico de Mateus e Bornstein[39], do qual é uma extensão.

Os tempos referentes ao algoritmo de Mateus e Bornstein foram obtidos também em um microcomputador AT-486, utilizando o código implementado por Rangel[46]. No que se refere ao de Beasley, ressalte-se que se trata de um algoritmo enumerativo, mas que por outro lado os tempos foram gerados em um 'supercomputador' CRAY-1S, que possui capacidade de processamento vetorial.

Analisando-se as diferenças percentuais calculadas

$$\frac{T^1 - T^0}{\overline{T}^1} * 100 \qquad \text{e} \qquad \frac{T^2 - T^0}{\overline{T}^2} * 100$$

onde

$$\begin{aligned} T^0 &= \text{tempo gasto pelo algoritmo proposto} \\ T^1 &= \text{tempo gasto pelo algoritmo de Mateus e Bornstein} \\ T^2 &= \text{tempo gasto pelo algoritmo de Beasley} \\ \overline{T}^1 &= \max(T^1, T^0) \\ \overline{T}^2 &= \max(T^2, T^0) \end{aligned}$$

verificamos que os tempos demandados pelo algoritmo proposto para resolver os problemas de pequeno porte foram todos menores que aqueles consumidos pelos dois outros em análise: 86.21% em média inferior a Beasley e 93.62% a Mateus e Bornstein. Convém registrar que, além disso, as soluções obtidas com este último algoritmo

TEMPOS DE COMPUTAÇÃO (s)					
ALGORITMOS	PROPOSTO	MATEUS e BORNSTEIN		BEASLEY	
PROB.	TEMPO (T^0)	TEMPO (T^1)	$\frac{T^1-T^0}{T^1} * 100$	TEMPO (T^2)	$\frac{T^2-T^0}{T^2} * 100$
IV–1	0.05	0.65	92.31	1.20	95.83
IV–2	0.06	0.54	88.89	1.00	94.00
IV–3	0.05	0.82	93.90	1.90	97.37
IV–4	0.06	1.15	94.78	0.90	93.33
V–3	0.06	1.15	94.78	1.20	95.00
VI–1	0.06	0.87	93.10	0.30	80.00
VI–2	0.05	0.87	94.25	0.50	90.00
VI–3	0.11	1.37	91.97	1.10	90.00
VI–4	0.11	1.15	90.43	0.30	63.33
VIII–1	0.17	3.24	94.75	3.80	95.53
VIII–2	0.16	3.89	95.89	1.80	91.11
VIII–3	0.22	3.40	93.53	1.80	87.78
VIII–4	0.22	4.17	94.72	4.00	94.50
IX–1	0.16	2.74	94.16	0.80	80.00
IX–2	0.16	3.68	95.65	0.90	82.22
IX–3	0.22	3.24	93.21	2.40	90.83
IX–4	0.16	3.40	95.29	3.10	94.84
XI–1	0.82			2.40	65.83
XI–2	0.88			4.30	79.53
XI–3	0.88			8.70	89.89
XI–4	1.10			8.80	87.50
XII–1	0.55			1.50	63.33
XII–2	0.55			2.30	76.04
XII–3	0.55			3.80	85.53
XII–4	0.60			7.50	92.00

Tabela V.6: Tempos de computação para os problemas de pequeno porte

TEMPOS DE COMPUTAÇÃO (s)			
ALGORITMOS	PROPOSTO	BEASLEY	
PROB.	TEMPO (T^0)	TEMPO (T^2)	$\frac{T^2-T^0}{T^2} * 100$
A–1	64.70	87.40	25.97
A–2	62.78	79.90	21.43
A–3	60.30	59.30	-1.66
A–4	57.56	28.10	-51.18
B–1	72.56	74.20	2.21
B–2	84.42	321.60	73.75
B–3	72.72	244.90	70.31
B–4	75.20	89.50	15.98
C–1	70.91	150.60	52.92
C–2	68.38	294.70	76.80
C–3	70.80	108.70	34.87
C–4	65.03	87.70	25.85

Tabela V.7: Tempos de computação para os problemas de grande porte

apresentaram diferença percentual média de 1.72% em relação à solução ótima dos problemas, taxa consideravelmente superior aos 0.03% atingidos por nosso algoritmo.

A princípio pode parecer estranho que os tempos obtidos por Rangel[46] sejam superiores, tempo em vista que o algoritmo de Mateus e Bornstein realize um número menor de operações que o algoritmo proposto. Entretanto, como já mencionado, estes resultados podem ser também decorrentes das diferenças de implementação, por exemplo, devido às estruturas de dados utilizadas.

Quanto aos problemas maiores, para os quais só dispomos dos tempos de Beasley, permaneceu, de um modo geral, a mesma tendência. Apenas dois deles consumiram maior tempo, sendo mínima a diferença para um (1.66%) e significativa para o outro (51.18%). Ressalte-se, todavia, que em Beasley o tempo dispendido por este segundo problema é bastante inferior aos demandados pelos demais de seu grupo. No total, tomando-se os 12 problemas de grande porte testados, constata-se uma diferença percentual média de 28.94%, favorável ao algoritmo proposto.

Capítulo VI

Conclusões

O objetivo do trabalho, como inicialmente definido, era o desenvolvimento de um algoritmo heurístico para o problema de localização capacitado. Para tal foram estudados diversos testes de redução e heurísticas gulosas já conhecidos. Neste ponto não houve contribuições adicionais aos resultados existentes. Ressalte-se, entretanto, a apresentação, no Capítulo 2, de uma demonstração mais concisa para a otimalidade destes testes.

No Capítulo 3, ainda com relação aos testes e heurísticas para abrir ou fechar uma facilidade, destaca-se a avaliação de aproximações que permitem incrementar a eficiência computacional, no aspecto tempo dispendido, dos algoritmos que os empreguem. Encontra-se aí uma efetiva contribuição do nosso trabalho, não no que diz respeito à obtenção de novos limites, mas a forma como foram gerados.

Para cada caso, ativação e desativação de facilidades, foram consideradas várias formas de relaxação lagrangeana do problema, relaxando-se, através de diferentes multiplicadores, ora ambas as restrições de oferta e demanda, ora cada uma delas individualmente, e ainda, por vezes, acrescentando-se restrições redundantes.

Constatou-se que todas as formas de relaxação aplicadas conduziram, em geral, a limites já conhecidos, à exceção de uma ou outra variações. Verificou-se adicionalmente a superioridade tanto numérica quanto computacional do limite Δ_i^D (Mateus e Bornstein[39]) sobre os demais limites gerados para Δ_i, inclusive $\overline{\Delta}_i$ (Akinc e Khumawala[3]), e a superioridade em termos de aproximação do limite $\overline{\Omega}_i$ (Akinc e Khumawala[3]) com relação aos outros limites encontrados para Ω_i. Analisou-se ainda condições em que ocorre a igualdade de $\overline{\Omega}_i$ a Ω_i^D (Mateus e Bornstein[39]) e situações de preferência do uso deste limite ao daquele, tendo em vista sua bastante menor complexidade computacional.

Estudados os testes e heurísticas, foi então proposto um algoritmo novo para a resolução do CWLP, composto por duas fases: a primeira, que procura determinar uma "boa" solução, intercalando a ativação e desativação de facilidades, e a segunda, que funciona como uma espécie de ajuste da anterior, reconsiderando decisões já tomadas, através da permuta de estados anteriormente atribuídos a certos armazéns.

Para avaliação de seu desempenho, o algoritmo foi aplicado a problemas-teste padrões, descritos na seção V.2, que podem ser divididos em dois grupos básicos: problemas pequenos (dimensão de até 50 facilidades $\times$ 50 consumidores) e problemas grandes (dimensão de 100 facilidades $\times$ 1000 consumidores).

Os resultados obtidos foram significativos. Boa parte das soluções encontradas são ótimas (88% dos problemas pequenos e 33.3% dos grandes). Mesmos para as soluções não ótimas, as diferenças percetuais permaneceram bastante reduzidas, inferiores a 0.4%, e podem ser consideradas satisfatórias. No geral, tomando-se todos os 37 problemas resolvidos, a diferança percentual média ficou inferior a 0.08%.

Com relação aos tempos de computação do algoritmo, obtidos em um microcomputador AT-486, são ínfimos para os problemas de pequeno porte (inferiores a $1s$) e bastante animadores para os problemas de maior porte (média inferior a $1min\ 10s$). Apenas para fornecer uma base comparativa, registre-se que estes tempos são inferiores aos de Beasley[10] em média 86% para os problemas pequenos e 29% para os problemas grandes. Ressalte-se entretanto que o algoritmo de Beasley é enumerativo, mas que por outro lado os tempos foram obtidos em um 'supercomputador' CRAY-1S, que efetua certas operações de cálculo de forma paralela.

Os resultados expostos acima, e mais detalhadamente do Capítulo 5, analisados tanto no aspecto qualidade da solução quanto no que se refere ao esforço computacional, indicam que o algoritmo proposto funciona igualmente bem em problemas de pequeno e grande porte e é capaz de resolvê-los sem maiores dispêndios computacionais.

Apêndice A

Implementação do Algoritmo para o CWLP

O algoritmo proposto no Capítulo 5 está implementado a seguir, em FORTRAN-Lahey, subdividido em vários módulos, que comporão as diversas subrotinas do programa. Por simplicidade, nem todos estes módulos serão explicitamente apresentados aqui, apenas os essenciais. Para os outros, especificaremos simplesmente suas funções.

Antes, porém, vamos identificar as estruturas que armazenam os dados do problema e entender o significado das variáveis de controle utilizadas.

- Os vetores `ctrans`, `cfixo`, `capofr` e `capdem` mantêm, respectivamente, os valores de c_{ij}, f_i, a_i e b_j; `nofr` e `ndem` correspondem ao número de possíveis localizações de facilidades e total de centros de demanda a atender.
- A estrutura `waprox` contém o melhor valor das aproximações Δ_i^L.
- Os conjuntos K_1, K_2 e $\overline{K}_0$, utilizados no algoritmo, como são disjuntos, serão representados num único vetor `K` onde `K(i)=1`, se $i \in K_1$, `K(i)=2`, se $i \in K_2$ e `K(i)=3`, se $i \in \overline{K}_0$. As variáveis `NK1`, `NK2` e `NK3` especificam o número de elementos de cada um destes conjuntos respectivamente.
- O referencial `capfecho` indica o quanto de capacidade ainda se pode fechar, considerando os armazéns já desativados, de forma a não tornar o problema inviável.

Isto bem entendido, podemos passar à exposição do programa, que então não conterá a área de definição de variáveis.

```
!!----------------- programa principal --------------!!

        call leitura
        do i=1,nofr
          waprox(i)=0.0
          K(i)=2
        end do
        NK2=nofr
        NK1=0
        NK3=0
        capfecho=capdem(0)

       !!*------ primeira fase --------*
        call inicializa_transporte
        call transporte(wki)
        call atualiza_solucao
        do while (NK2.ne.0)      !!* e' um repita *
          call escolhe_arm_fechar(s,incerto)
          if (s.ne.0) then
            call fechar_armazem(s,incerto)
            call atualiza_solucao
          endif
        enddo
        call solfase1

       !!*------ segunda fase --------*
        do while (NK3.ne.0)

         !!*----- iteracao ADD -----*
          r=0
          do while ((r.eq.0) .and. (NK3.ne.0))
            call escolhe_arm_acrescentar(r)
            call indeterminar_armazem(r)
            call transporte(wki)
            if (wk-wki >= cfixo(r)) then
              call abrir_armazem(r)
              call atualiza_solucao
              r=0
            end if
          end do
```

```
          !!*----- iteracao DROP ----*
           incerto = .false.
           call escolhe_arm_retirar(s)
           call fechar_armazem(s,incerto)
           if (s.ne.r) then
             call abrir_armazem(r)
             call atualiza_solucao
           else
             call retorna_solucao
           endif
         end do
         call solfase2
       stop
       end
!!-------------------------------------------------!!
```

Vejamos agora a função de cada sub-rotina:

`leitura` obtém os dados iniciais do problema (`nofr, ndem, ctrans, cfixo, capofr` e `capdem`) e cria o nó demanda artificial.

`inicializa_transporte` constrói a solução inicial (árvore básica) do primeiro problema de transporte. (Ver Apêndice B).

`transporte(cost)` implementa o algoritmo de transporte, retornando em `cost` o custo de atendimento às demandas (Ver Apêndice B).

`atualiza_solucao` preserva a melhor solução encontrada, ou seja, a mais recente. O referencial `wk` preserva o custo desta solução.

`fechar_armazem(arm)` transforma o estado da facilidade `arm` em 3, se `incerto`, ou em 0, se não `incerto`. (`incerto` é pois o resultado do teste $\Delta_s/f_s > \epsilon_2$). Além disso o valor de `capfecho` é atualizado.

`abrir_armazem(arm)` transforma o estado da facilidade `arm` em 1.

`indeterminar_armazem(arm)` converte o estado da facilidade `arm` de 3 para 2, provocando assim sua reconsideração na fase 2. Também atualiza `capfecho`.

`solfase1, solfase2` apresentam o resultado da fase 1 e fase 2 respectivamente.

As três subrotinas escolhe_arm_fechar, escolhe_arm_acrescentar e escolhe_arm_retirar constituem partes fundamentais do algoritmo e por isso serão apresentadas mais detalhadamente.

```
!!------ teste exato p/ abrir e heuristica p/ fechar ---------!!
      subroutine escolhe_arm_fechar(s,incerto)
      parameter(zero=1.0d-4, e1=0.70, e2=0.05, tipo=2)

        call calcula_deltaprox(daprox,tipo)

        difsant=-1  !!* um numero negativo *
        do i=1,nofr
        if (K(i).eq.tipo) then
           if (waprox(i)-daprox(i) > zero) then
             difs=cfixo(i) - waprox(i)
             if (difs>difsant) difsant=difs
           else
             waprox(i)=daprox(i)
           endif
        endif
        enddo


        s=0
        do while ((s.eq.0) .and. (NK2.ne.0))

          difs=0
          do i=1,nofr
          if (K(i).eq.tipo) then
            if ((capofr(i)>capfecho) .or. (waprox(i)>=cfixo(i))) then
              call abrir_armazem(i)
            else if (difs < cfixo(i)-waprox(i)) then
                   s=i
                   difs=cfixo(i)-waprox(i)
            endif
          endif
          enddo
```

```
        if (NK2.ne.0) then
          call altera_custo(s)
          call transporte(wki)
          call retorna_custo(s)
          dexato=wki - wk
          if (dexato .eq. 0.0) return
          difs=cfixo(s) - dexato

          if ((difs<=0) .or. (waprox(s)/dexato < e1) .or.
     &         (difsant>difs)) then
            if (difs<=0) call abrir_armazem(s)
            if ((abs(cfixo(s)-waprox(s)-difsant)<=zero) .or.
     &           (difs>difsant)) difsant=difs
            call retorna_solucao
            waprox(s)=dexato
            s=0

          else
            incerto = (dexato/cfixo(s) > e2)
          endif

        endif
      enddo
      return
    end
!!-------------------------------------------------------------!!
```

Este módulo realiza a escolha da facilidade a fechar, aplicando a heurística 2 modificada, ao mesmo tempo em que seleciona, com o teste 1 modificado, os armazéns a serem ativados.

As aproximações para Δ_i, especificamente no nosso caso Δ_i^D, empregadas em ambos, teste e heurística, são determinadas em `daprox` pela sub-rotina `calcula_deltaprox(daprox,tipo)`, para cada i, onde `K(i)=tipo`.

Ainda sobre o bloco acima cabem as seguintes considerações:

- O referencial `difsant` conserva $\sup_{i \in \overline{K}_2}\{f_i - \Delta_i\}$. Observe que é dispensável manter o conjunto $\overline{K}_2$.
- As sub-rotinas `altera_custo` e `retorna_custo` adaptam a estrutura da árvore básica para a resolução do próximo problema de transporte. (Ver Apêndice B).
- A sub-rotina `retorna_solucao` retoma a melhor solução encontrada, que constitui também uma melhor base para a solução inicial do próximo problema de transporte.

```
!!-------------- acrescentar armazem a solucao --------------!!
      subroutine escolhe_arm_acrescentar(s)
      parameter (inf=1.0d40, tipo=3)

       call calcula_omegaprox(oaprox,tipo)

       difs=-inf       !!*-- um numero muito pequeno --*
       do i=1,nofr
       if (K(i).eq.tipo) then
         if (difs < oaprox(i) - cfixo(i)) then
           s=i
           difs=oaprox(i) - cfixo(i)
         end if
       end if
       enddo
       return
      end
!!-------------------------------------------------------------!!
```

A função deste bloco é selecionar a facilidade a ser acrescentada à solução corrente, numa tentativa de gerar alguma possibilidade de melhora. Para tal, implementa a heurística 1 modificada, onde as aproximações para Ω_i, `K(i)=tipo`, são determinadas em `oaprox` pela sub-rotina `calcula_omegaprox(oaprox,tipo)`, que pode implementar opcionalmente Ω_i^D ou $\overline{\Omega}_i$.

```
!!--------------- retirar armazem da solucao ----------------!!
      subroutine escolhe_arm_retirar(s)
      parameter(tipo=1)

        call calcula_deltaprox(waprox,tipo)
        waprox(r)=wk-wki

        s=0
        do while (s.eq.0)
          s=r
          difs=cfixo(s)-waprox(s)
          do i=1,nofr
          if (K(i).eq.tipo) then
            if ((difs < cfixo(i)-waprox(i)) .and.
     &           (capofr(i)<=capfecho)) then
              s=i
              difs=cfixo(i)-waprox(i)
            endif
          endif
          enddo

          if (s.ne.r) then
            call altera_custo(s)
            call transporte(wki)
            call retorna_custo(s)
            if ((wki - wk) >= (cfixo(s)-cfixo(r)) then
              waprox(s)=wki-wk + waprox(r)
              s=0
            endif
          end if
        end do
        return
      end
!!-------------------------------------------------------------!!
```

Este módulo procura reotimizar a solução após o acréscimo da facilidade determinada no bloco anterior: mais especificamente, procura a facilidade a ser retirada. Em termos das subrotinas que utiliza, não traz nenhuma novidade: todas são idênticas àquelas de `escolhe_arm_fechar`.

Apêndice B

Implementação do Algoritmo de Transporte

Neste apêndice apresentaremos a implementação do algoritmo de transporte descrito no Capítulo 4, cujo funcionamento fundamenta-se na associação solução básica × árvore geradora.

A manutenção das árvores básicas, como vimos extensivamente na oportunidade, compreende a atualização das estruturas P (predecessor), X (fluxo), U (potencial do nó), D (profundidade) e W (sucessor em pré-ordem).

Tais estruturas serão implementadas em vetores, identificados aqui pela mesma notação já adotada. Entretanto, a forma de indexação para os vetores X e P, que possuem entradas para ambos pontos origem e destino, terá que ser reformulada, de modo a diferenciar referências a nós origem e destino de mesma numeração.

Para isto, será criado um referencial `raiz (=nofr+1)` e toda indexação a um nó demanda j será da forma `X(j+raiz)` ou `P(j+raiz)`. Deste modo, as primeiras `nofr` posições de `X` e `P` correspondem aos pontos origem, e as `(ndem+1)` seguintes referem-se aos nós destino.

Também os nós particularizados na exposição do Capítulo 4 serão aqui identificados de forma semelhante. De um modo geral, as variáveis de controle utilizadas têm significado bastante óbvio e, portanto, não serão definidas.

Passemos então ao programa, que está particionado conforme desenvolvimento do Capítulo 4. O módulo `transporte(cost)` devolve em `cost` o custo mínimo para atender as demandas a partir somente dos centros de oferta disponíveis (facilidades não desativadas).

```
!!---------- resolucao do problema de transporte -------------!!

    subroutine transporte(cost)
    parameter(zero=-1.0d-10, large=-1.0d10/2, max=10**9, nexam=1)


!!----------------- escolha do arco a entrar -----------------!!

    dem=0
    do while (.true.)    !!* sai quando nao tem mais arco a entrar *
      start=dem
      uincr=zero
      npraux=0
      fimrepita=.false.
      do while (.not. fimrepita)
        npraux=npraux+1
        dem=mod(dem+1,ndem+1)
        indem=dem + raiz
        if (indem.eq.raiz) then
           udem=0.0
        else
           udem=u(p(indem)) - ctrans(p(indem),dem)
        endif
        do ofr=1,nofr
          if ((K(ofr).eq.'1').or.(K(ofr).eq.'2')) then
            rarc=ctrans(ofr,dem) - u(ofr) + udem
            if (rarc < uincr) then
              uincr=rarc
              inofr=ofr
            endif
          endif
        enddo
        fimrepita = ((npraux>=nexam).and.(uincr.ne.zero))
   &                .or. (uincr<=large) .or. (dem.eq.start)
      enddo

!!------------------ testa fim do laco ------------------------!!

      if (uincr.eq.zero) go to 100   !!* solucao ja' e' otima *
```

```
!!----------------- escolha do arco a sair ---------------------!!

      out=0
      xincr=max
      join=inofr
      noaux=indem
      djoin=d(join)
      daux=d(p(noaux)) + 1
      do while (noaux.ne.join)       !!* e' um repita *
        if (djoin < daux) then
          if ((noaux > raiz) .and. (x(noaux) <= xincr)) then
            xincr=x(noaux)
            out=noaux
          endif
          noaux=p(noaux)
          daux=daux-1
        else
          if ((join < raiz) .and. (x(join) < xincr)) then
            xincr=x(join)
            out=join
          endif
          join=p(join)
          djoin=djoin - 1
        endif
      enddo


!!-------------- preparacao p/ atualizacao da arvore -------------!!

      last=out
      exit=p(out)
      if (last > raiz) then       !!* e' no'demanda ? *
        entry=inofr
        first=indem
       !!*--- intie = predecessor de last ---*
        intie=exit
        daux=d(exit)+1     !!*  d(last)  *
        noaux=w(intie)
       !!* enquanto succ(exit) .and. nao filho(last) *
```

```
      do while ((d(noaux)>daux) .and. (p(noaux).ne.last))
        intie=noaux
        noaux=w(intie)
      enddo
     !!*--- rscout = successor de first ---*
      noaux=p(first)
      daux=d(noaux)+1      !!*  d(first)  *
      rscout=w(noaux)
     !!* enquanto succ(p(first)) .and. nao filho(first) *
      do while ((d(rscout)>daux) .and. (p(rscout).ne.first))
        rscout=w(rscout)
      enddo
      primvisit=entry
      uincr=-uincr
      dincr=d(entry) - daux + 1      !!* dincr=d(entry)-d(first)+1 *
    else
      entry=indem
      first=inofr
      intie=p(exit)         !!* predecessor de last *
      do while (w(intie).ne.last)
        intie=w(intie)
      enddo
     !!*--- primvisit = predecessor de entry ---*
      primvisit=p(entry)
      daux=d(primvisit)+1      !!*  d(entry)  *
      noaux=w(primvisit)
     !!* enquanto succ(p(entry)) .and. nao filho(entry) *
      do while ((d(noaux)>daux) .and. (p(noaux).ne.entry)
&        .and. (primvisit.ne.intie))
        primvisit=noaux
        noaux=w(primvisit)
      enddo
      rscout=w(first)
      dincr=daux - d(first) + 1    !!* dincr=d(entry)-d(first)+1 *
    endif
    outtie=w(primvisit)
    visit=primvisit
    visitant=visit
    rear=rscout
    stem=first
```

```
      prev=entry

!!---------------- atualizacao da arvore -------------------------!!

      xant=0
      do while (prev .ne. last)     !!* e' um repita *
        xatu=xant + xincr
        xant=x(stem)
        x(stem)=xatu
        if (stem > raiz) then  !!* stem eh centro de demanda *
          daux=d(p(stem))+3

         !!*--- encadeia nos a direita de stem ---*
          if ((d(rscout)>daux).or.(p(rscout).eq.stem)) then
            w(visit)=rscout
            do while ((d(rscout)>daux).or.(p(rscout).eq.stem))
              visit=rscout
              d(visit)=d(visit) + dincr
              u(visit)=u(visit) + uincr
              rscout=w(rscout)
            enddo
          endif
         !!*-- marca onde comecarao os nos a esquerda de stem ---*
          visitant=visit

        else    !!* stem eh centro de oferta *
          daux = d(stem) + 1
          d(stem)=d(stem) + dincr
          u(stem)=u(stem) + uincr
          w(visit)=stem

         !!*--- encadeia nos a esquerda de stem ---*
          visit=stem
          lscout=w(visit)
          do while ((p(lscout).ne.prev) .and. (lscout.ne.rear))
            visit=lscout
            d(visit)=d(visit) + dincr
            u(visit)=u(visit) + uincr
            lscout=w(lscout)
          enddo
```

```
   !!*--- encadeia nos a direita de stem ---*
    if (d(rscout)>daux) then
      w(visit)=rscout
      do while (d(rscout)>daux)   !!* e' um repita *
        visit=rscout
        d(visit)=d(visit) + dincr
        u(visit)=u(visit) + uincr
        rscout=w(rscout)
      enddo
    endif


   !!*--- encadeia nos a esquerda de prev ---*
    if (lscout .ne. rear) then
      dincr=dincr - 2
      w(visitant)=lscout
      do while (lscout .ne. rear)       !!* e' um repita *
        visitant=lscout
        d(visitant)=d(visitant) + dincr
        u(visitant)=u(visitant) + uincr
        lscout=w(lscout)
      enddo
      w(visitant)=stem
      dincr=dincr + 2
    endif

    rear=stem          !!* ult no ofr visitado no caminho *
  endif

  noaux=p(stem)
  p(stem)=prev
  prev=stem
  stem=noaux
  xincr= -xincr
  dincr=dincr + 2
enddo
```

```
!!*--- encadeia nos a esquerda caso last eh dem ---*
 if (last > raiz) then
   if (intie.ne.primvisit) then
     lscout=w(intie)
   else
     lscout=outtie
   endif
   if (lscout .ne. rear) then
     dincr=dincr - 2
     w(visitant)=lscout
     do while (lscout.ne.rear)     !!* e' um repita *
       visitant=lscout
       d(visitant)=d(visitant) + dincr
       u(visitant)=u(visitant) + uincr
       lscout=w(lscout)
     enddo
     visit=visitant
     dincr= dincr + 2
   endif
 endif

!!*--- ultimos ajustes de atualizacao ---*
 if (primvisit .eq. intie) then
   outtie=rscout
   if ((last<raiz) .and. (p(entry).eq.join)) then
     daux=djoin + 1    !!* d(entry) *
   !!* enquanto succ(join) .and. nao filho(entry) *
     do while ((d(outtie)>daux) .and. (p(outtie).ne.entry))
       primvisit=outtie
       outtie=w(outtie)
     enddo
     if (primvisit .ne. intie) then
       w(primvisit)=first
       w(intie)=rscout
     endif
   endif
 else
   w(intie)=rscout
 endif
 w(visit)=outtie
```

```
    !!*--- atualizacao dos demais fluxos ----*
      if (xincr.ne.0) then
        xincr=-abs(xincr)
        visit=entry
        do while (visit.ne.join)
          x(visit)=x(visit) + xincr
          xincr=-xincr
          visit=p(visit)
        enddo
        xincr=abs(xincr)
        visit=exit
        do while (visit.ne.join)
          x(visit)=x(visit) + xincr
          xincr=-xincr
          visit=p(visit)
        enddo
      endif

    !!*--- valor da funcao no ponto otimo ---*
  100 cost=0.0
      do ofr=1,nofr
        cost=cost + ctrans(ofr,p(ofr)-raiz) * x(ofr)
      enddo
      do dem=raiz+1,raiz+ndem
        cost=cost + ctrans(p(dem),dem-raiz) * x(dem)
      enddo
      return
    end
!!-------------------------------------------------------------!!
```

Por se tratar de um algoritmo primal, este procedimento deve iniciar com uma solução básica viável. Para o primeiro problema de transporte a ser resolvido é possível construir uma solução inicial, como já comentado, através da regra do canto noroeste, a seguir implementada.

```
!!--------- primeira inicializacao do problema ----------!!
      subroutine inicializa_transporte
```

```
      !!*--- inicializacao do no raiz ----*
        p(raiz)=raiz     !!* esta convencao e' importante *
        d(raiz)=-1       !!* esta convencao e' importante *
        x(raiz)=0
        u(raiz)=0
        w(raiz)=1

        j=0
        i=0
        dincr=1
        uincr=0.0
        sobra=capdem(j)
        do while ((i.ne.nofr) .or. (j.ne.ndem))    !!* e' um repita *
          do while ((sobra>=0) .and. (i.ne.nofr))  !!* e' um repita *
            i=i+1
            sobra=sobra - capofr(i)
            p(i)=j+raiz
            x(i)=capofr(i)
            d(i)=dincr
            u(i)=ctrans(i,j) - uincr
            w(i)=i+1
          enddo
          x(i)= x(i) + sobra
          sobra=-sobra
          if (j.ne.ndem) then
            do while (sobra>0)         !!* e' um repita *
              j=j+1
              sobra=sobra - capdem(j)
              p(j+raiz)=i
              x(j+raiz)=capdem(j)
            enddo
            x(j+raiz)= x(j+raiz) + sobra
            dincr=dincr+2
            uincr=ctrans(i,j) - u(i)
            sobra=-sobra
          endif
        enddo
        return
      end
!!-------------------------------------------------------------!!
```

Resolvido o primeiro problema de transporte, as soluções iniciais para os próximos podem ser obtidas a partir de modificações em soluções ótimas anteriormente determinadas, conforme procedimento descrito na seção IV.5.2 e implementado na sub-rotina abaixo.

```
!!---------- reinicializacao do prob.transporte  ----------!!
      subroutine altera_custo(arm)
      parameter(inf=1.0d10)

        if (p(arm).eq.raiz) then
          cij=inf
        else
          cij=ctrans(arm,p(arm)-raiz)
        endif
        u(arm)=u(arm) + inf - cij
        l=w(arm)
        do while (d(l)>=d(arm)+2)
          if (d(l).eq.d(arm)+2) uincr=ctrans(arm,p(l)-raiz) - cij
          u(l)=u(l) + uincr
          l=w(l)
        enddo

        do j=1,ndem
          c(j)=ctrans(arm,j)
          ctrans(arm,j)=inf
        enddo
        return
      end


      subroutine retorna_custo(arm)

        do j=1,ndem
          ctrans(arm,j)=c(j)
        enddo
        return
      end
!!-------------------------------------------------------!!
```

Observe que só é necessário manter custos artificiais (*INF*) para os caminhos originados em uma facilidade, enquanto ela estiver sendo avaliada pela heurística 2 se deve ou não ser desativada. A seguir, mesmo sendo fechada tal facilidade, estes custos a ela associados podem retornar a seus verdadeiros valores, se na parte "arco a entrar na base" só considerarmos os armazéns disponíveis, como aqui é feito. (Este resultado já foi verificado ao final da seção IV.5.2.)

Desta forma, só um vetor (que denotamos por `c`) de dimensão `ndem` precisa ser criado para preservar os reais custos de transporte associados a uma facilidade, enquanto é avaliada a variação exata nos custos ocasionada pela desativação desta, para então serem retomados pela sub-rotina `retorna_custo`, logo em seguida à resolução do problema de transporte.

Referências Bibliográficas

[1] J. H. Ahrens and G. Finke. Primal transportation and transshipment algorithms. *Zeitschrift fur Operations Research*, 24:1–32, 1980.

[2] C. H. Aikens. Facility location models for distribuiton planning. *European Journal of Operational Research*, 22:263–279, 1985.

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

[4] B. M. Baker. Linear relaxations of capacitated warehouse location problem. *Journal of Operational Research Society*, 33(5):475–479, 1982.

[5] B. M. Baker. A partial-dual algorithm for the capacitated warehouse location problem. *European Journal of Operational Research*, 23:48–56, 1986.

[6] J. Barcelo. *Computacional Experiments with Location Problems Using Automatic Constraint Generation.* Technical Report RR85/06, Dept. d'Investigacion Operativa i Estadistica – Universitat Politecnica de Barcelona, 1985.

[7] J. Barcelo and J. Casanovas. A heuristic lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research*, 15:212–226, 1984.

[8] E. Bartezzaghi, A. Colorni, and P. C. Palermo. A search tree algorithm for plant location problems. *European Journal of Operational Research*, 7:371–379, 1981.

[9] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Lineac Programming and Network Flows.* John Wiley & Sons, Inc., second edition, 1990.

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

[11] J. E. Beasley. Lagrangean heuristics for location problems. JUL 1988. to appear (revised SET/90).

[12] J. E. Beasley. Or-library: distributing test problems by electronic mail. JUN 1990. working paper.

[13] G. R. Bitran, V. Chandru, D. E. Sempolinsk, and J. F. Shapiro. Inverse optimization: an application to the capacitated plant location problem. *Management Science*, 27(10):1120–1141, 1981.

[14] G. H. Bradley, G. G. Brown, and G. W. Graves. Design and implementation of large scale primal transshipment algorithms. *Management Science*, 24(1):1–34, 1977.

[15] N. Christofides and J. E. Beasley. Extensions to a lagrangean relaxation approach for the capacitated warehouse location problem. *European Journal of Operational Research*, 12(1):19–28, 1983.

[16] L. Cooper. Heuristic methods for location-allocation problems. *SIAM Review*, 6(1):37–53, 1964.

[17] P. S. Davis and T. L. Ray. A branch and bound algorithm for the capacitated facilities location problem. *Naval Research Logistics Quartely*, 16(3):331–334, 1969.

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

[19] L. B. Ellewein and P. Gray. Solving fixed charge location-allocation problems with capacity and configuration constraints. *AIIE Transactions*, 3(4):290–298, 1971.

[20] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6):992–1009, 1978.

[21] E. Feldman, F. A. Lehrer, and T. L. Ray. Warehouse location under continuous economies of scale. *Management Science*, 12(9):670–684, 1966.

[22] M. L. Fisher. The lagrangean relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[23] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodulao set functions - II. *Math. Programming Study*, 8:73–87, 1978.

[24] R. D. Galvão. *The Optimal Location of Facilities on a Network.* PhD thesis, Imperial College, Londres, 1977.

[25] R. D. Galvão. The use of lagrangean relaxation in the solution of uncapacitated facility location problems. *Location Science*, 1(1):57–79, 1993.

[26] A. M. Geoffrion. Lagrangean relaxation for integer programming. *Math. Programming Study*, 2:82–114, 1974.

[27] A. M. Geoffrion and G. W. Graves. Multicomodity distribution system design by benders decomposition. *Management Science*, 20:822–844, 1974.

[28] A. M. Geoffrion and R. McBride. Lagrangean relaxation applied to capacitated facility location problems. *AIIE Transactions*, 10(1):40–47, 1978.

[29] F. Glover, D. Karney, and D. Klingman. The augmented predecessor index method for locating stepping-stone paths and assigning dual prices in distribution problems. *Transportation Science*, 6:171–179, 1972.

[30] F. Glover, D. Klingman, and J. Stutz. Augmented threaded index method for network optimization. *INFOR, Canad. J. Operational Res. and Information Processing*, 12(3):293–298, 1974.

[31] M. Guignard and K. Spielberg. A direct dual method for the mixed plant location problem with some side constrains. *Math. Programming*, 17:198–228, 1979.

[32] M. Guignard, K. Spielberg, and S. Kim. *A Strong Lagrangean Relaxation for Capacitated Plant Location Problems.* Technical Report, Departament of Statistics – Wharton School, University of Pennsylvania, 1983.

[33] S. K. Jacobsen. Heuristics for the capacitated plant location model. *European Journal of Operational Research*, 12:253–261, 1983.

[34] S. K. Jacobsen. On the use of tree-indexing methods in transportation algorithms. *European Journal of Operational Research*, 2:54–65, 1978.

[35] E. Johnson. Networks and basic solutions. *Operations Research*, 14:619–623, 1966.

[36] J. L. Kennington and R. V. Helgason. *Algorithms for Network Programming.* John Wiley & Sons, Inc., 1980.

[37] B. M. Khumawala. An efficient heuristic procedure for the capacitated warehouse location problem. *Naval Research Logistics Quartely*, 20:609–623, 1974.

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

[39] G. R. Mateus. *Algoritmo Exato e Heurísticas para o Problema de Localização.* PhD thesis, Programa Engenharia de Sistemas e Computação – COPPE/UFRJ, Rio de Janeiro, 1986.

[40] G. R. Mateus and C. T. Bornstein. Dominance criteria for the capacitated warehouse location problems. *Journal of Operational Research Society*, 42(2):145–149, 1991.

[41] G. R. Mateus and C. T. Bornstein. *Dominance Criteria for the Capacitated Warehouse Location Problems.* Technical Report ES-182/89, Programa Engenharia de Sistemas e Computação – COPPE/UFRJ, Rio de Janeiro, 1986.

[42] G. R. Mateus and Z. K. G. P. Júnior. An algorithm for large scale capacitated location problem in networks. MAY 1992. to appear.

[43] G. R. Mateus, J. Thizy, and Z. K. G. P. Júnior. Approximate and exact solution methods for network location problem. JUN 1992. to appear.

[44] P. Michelon and N. Maculan. Lagrangean methods for 0–1 quadratic problems. *Discrete Applied Mathematics*, 42:257–269, 1993.

[45] R. M. Nauss. An improved algorithm for the capacitated facility location problems. *Journal of Operational Research Society*, 29(12):1195–1202, 1978.

[46] M. C. Rangel. *Critério de Dominância para Problemas de Localização de Armazéns Capacitados de Grande Porte.* Master's thesis, Programa Engenharia de Sistemas e Computação – COPPE/UFRJ, Rio de Janeiro, 1991.

[47] Y. Rapp. Planning of exchange locations and boundaries. *Ericsson Technics*, 2:1–22, 1962.

[48] G. Sá. Branch-and-bound and aproximate solutions to the capacitated plant location problem. *Operations Research*, 17:1005–1016, 1969.

[49] V. Srinivasan and G. L. Thompson. Benefit cost analisys of coding techniques for the primal transportation algorithm. *J. Assoc. Comput. Mach.*, 20(2):194–213, 1973.

[50] M. B. Teitz and P. Bart. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, 16(5):955–961, 1968.

[51] T. J. Van Roy. A cross decomposition algorithm for the capacitated facility location. *Operations Research*, 34(1):145–163, 1986.

[52] R. A. Whitaker. Some add-drop and drop-add interchange heuristics for non-linear warehouse location. *Journal of Operational Research Society*, 36(1):61–70, 1985.