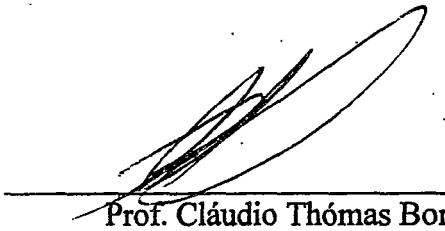
MÉTODO ÓTIMO E ÉPSILON-ÓTIMO PARA RESOLUÇÃO DO PROBLEMA DE LOCALIZAÇÃO CAPACITADO

Douglas Valiati

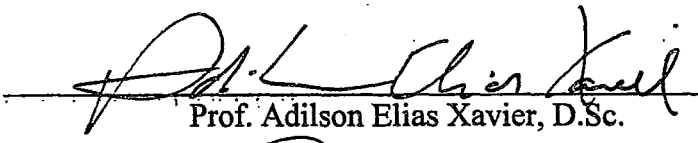
TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIËNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

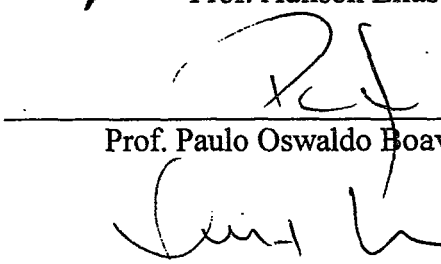


Prof. Cláudio Thómas Bornstein, Dr.Rer.Nat.

Prof. Nelson Maculan Filho, D.Habil.



Prof. Adilson Elias Xavier, D.Sc.



Prof. Paulo Oswaldo Boaventura Netto, D.Ing.

Prof. Luiz Satoru Ochi, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2006

VALIATI, DOUGLAS

Método Ótimo e Épsilon-Ótimo para Resolução do Problema de Localização Capacitado [Rio de Janeiro] 2006

IX, 281 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e computação, 2006)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Problema de Localização Capacitado
2. Problema de Transporte
3. Otimização Combinatória
4. Relaxação Lagrangeana
5. Testes de Redução
6. Branch and Bound

I. COPPE/UFRJ II. Título (série)

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

MÉTODO ÓTIMO E ÉPSILON-ÓTIMO PARA RESOLUÇÃO DO PROBLEMA DE LOCALIZAÇÃO CAPACITADO

Douglas Valiati

Março/2006

Orientador: Cláudio Thómas Bornstein

Programa: Engenharia de Sistemas e Computação

Será apresentado um método para solucionar o Problema de Localização Capacitado (*PLC*). O método pode ser utilizado como um método heurístico para problemas operacionais; como um método ε-ótimo para problemas de difícil resolução ou com alguma margem de erros no levantamento dos dados; ou ainda como método ótimo. O objetivo é resolver problemas grandes e obter uma solução com qualidade, ou seja, uma solução ótima ou muito próxima dela, em um tempo viável para o problema. Serão examinados testes de redução baseados em técnicas *ADD/DROP*. Essas técnicas permitem a redução das dimensões do problema. Dois limites inferiores serão utilizados, um baseado em relaxação Lagrangeana e outro na relaxação da restrição de integridade da solução. Serão desenvolvidas heurísticas, como estratégias de ramificação, que utilizam informações do testes de redução e dos limites inferiores. Essas heurísticas permitem que o algoritmo encontre uma solução com qualidade nas primeiras explorações da árvore de busca. A estratégia de busca na árvore é realizada através de uma combinação entre busca em profundidade e busca em largura.

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

METHOD OPTIMAL AND ÉPSILON-OPTIMAL TO SOLVE THE CAPACITATED LOCATION PROBLEM

Douglas Valiati

March/2006

Advisor: Cláudio Thómas Bornstein

Department: Computing and Systems Engineering

This work presents a method to solve the Capacitated Location Problem. The method can be utilized as a heuristic algorithm for operational problems, as an ε-optimal algorithm for difficult problems or as an optimal algorithm. The main objective of the work is to solve big problems and get good solutions, that is, to get optimal or very close solutions of them. Firstly a Transportation method will be development, although it is not the main objective of the work, it is essential for the development of *PLC*. Transportation problems are utilized in ADD/DROP techniques and to get inferior limits of *PLC*. Reduction tests will be examined. Those techniques allow the reduction of the problem dimensions. Two lower bounds will be used; one based on Lagrangian relaxation and another in the relaxation of the integer restriction. Heuristics will be developed as ramification strategies. It gets information of reductions tests and limits inferior. Those heuristics allows the algorithm to find a near-optimal solution in the first explorations of the search tree. The exploration strategy, in the search tree, is performed make a merge between depth first search and breath first search.

Índice

Capítulo 1

1 Introdução

O problema de localização tem sido estudado desde o início da civilização. No começo, os problemas que os seres humanos tinham eram de complexidade pequena, como, por exemplo, encontrar um local adequado para fixar sua residência ou colocar sua armadilha de caça. À medida que o tempo foi passando e a civilização evoluindo, o problema de localização passou a ter complexidade muito grande, ou seja, suas dimensões passaram a ser cada vez maiores.

Embora seja perfeitamente possível aplicar um modelo matemático a um problema de localização de um indivíduo comum, é nas instituições públicas e privadas que a aplicação de modelos matemáticos e técnicas de resolução de problemas de localização é realmente essencial. Normalmente, os problemas encontrados em organizações têm alta complexidade. A resolução desses problemas de forma inteligente pode levar a organização ao sucesso; caso contrário, as conseqüências podem ser desastrosas.

Um exemplo de alta complexidade seria quando uma determinada indústria decide se instalar em um país. Nesse caso, será necessário decidir onde instalar suas fábricas, seus depósitos ou armazéns, sua sede, suas filiais, agências de vendas, etc. Um exemplo no setor público seria a instalação de escolas em um município, estado ou país. O problema consiste em decidir em que locais serão instaladas as escolas para que toda a demanda seja atendida. Outro exemplo, no setor público, é o da localização do corpo de bombeiro. Deve-se decidir em que locais será instalado, a fim de possibilitar o atendimento à população em um tempo mínimo estipulado. As unidades que fornecem produtos ou serviços, ou recebem produtos, como os depósitos, são denominadas de facilidades.

A instalação de facilidades, como depósitos ou fábricas, afeta o custo de distribuição de mercadorias ou serviços aos clientes e pode levar a diferentes custos de transporte e instalação dessas. Esses custos variam conforme a localização e o tamanho das facilidades. Várias questões associadas com o problema de localização de

facilidades têm surgido na literatura. O leitor pode obter mais conhecimento sobre o assunto através dos artigos: Francis, McGinnis e White [25], Aikens [1], e Sridharan [37].

Existem diversos outros exemplos de problemas de localização, tais como, onde instalar antenas de celulares para atender uma determinada demanda, quais servidores de uma rede de computador podem fornecer um determinado serviço, etc. Em conseqüência dessa diversidade de problemas, existem diversos modelos matemáticos de localização, um para cada subconjunto de problemas. Na maioria dos casos, são modelos que levam a problemas não polinomiais (NP), ou seja, são problemas muito difíceis de serem resolvidos computacionalmente. O leitor pode obter conhecimento sobre problemas NP em Hopcrof, Ullman e Motwani [31].

Quando se fala em problema de localização, uma da primeiras questões que vêm à tona é a de custos, os quais são na maioria das vezes muito grandes. O que se deseja em um problema de localização é atender à demanda com qualidade e com o menor custo possível. Um erro de 1% a 5% na escolha das facilidades, considerado pequeno em relação à complexidade de resolução do problema, pode levar a instituição a ter prejuízos de milhões de reais. Esse fato é a principal justificativa para se continuar a pesquisa de soluções para problemas de localização.

A segunda consideração sobre o problema de localização refere-se à sua característica. Quando alguém se depara com um problema deste tipo, normalmente, é porque está fazendo um planejamento estratégico ou tático, isto é, trata-se de um problema de planejamento, e não operacional. Isso significa que, dado um problema de localização, sua resolução, para a instituição, não precisa ser imediata, em segundos, minutos ou horas. Assim sendo, é muito mais valioso obter uma solução com boa qualidade, em 24 horas, do que uma solução com qualidade discutível, em alguns segundos. Esse assunto será abordado novamente na seção 1.3, quando os objetivos serão discutidos.

Neste trabalho será abordado o Problema de Localização Capacitado (*PLC*). O *PLC* consiste em determinar o número e os locais de instalação das facilidades, respeitando-se a capacidade de cada facilidade e a demanda de cada cliente, de modo a

minimizar o custo total. O custo total é composto pelo custo fixo de instalação das facilidades abertas e do custo de transporte para distribuição dos produtos das facilidades abertas aos clientes. Respeitar a capacidade de cada facilidade significa que a distribuição de produtos de cada facilidade para os clientes não pode ultrapassar a quantidade de produtos disponíveis. Respeitar a demanda de cada cliente significa que o cliente deve ter toda sua demanda atendida.

De uma maneira geral, considerando-se economias de escala, o custo fixo é minimizado quando é considerado uma única facilidade. Neste caso, no entanto, os custos de transporte são elevados. Por outro lado, quando são instaladas todas as facilidades o custo de transporte decresce, contudo, o custo fixo aumenta. Assim sendo, é necessário encontrar um número de facilidades, para serem abertas, que atinja um equilíbrio entre o custo fixo e o custo de transporte, chegando, assim, à solução ótima.

1.1 Modelagem matemática do Problema de Localização Capacitado

O Problema de Localização Capacitado (PLC) consiste em, dado um conjunto de facilidades I em que cada facilidade i tem uma capacidade a_i e um conjunto de clientes J, o cliente j tendo demanda b_j, encontrar um conjunto $P \subseteq I$ que atenda todas as necessidades dos clientes, de forma a minimizar o custo total. O custo é composto pelo custo fixo f_i, que é o custo de instalação da facilidade i, e pelo custo variável c_{ij}, que é o custo de transporte da facilidade i ao cliente j. O problema de localização capacitado pode ser formulado como:

(1-1) $$(PLC) \min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i$$

S.a.

(1-2) $$\sum_{j \in J} x_{ij} \leq a_i y_i, \quad \forall i \in I$$

(1-3) $$\sum_{i \in I} x_{ij} = b_j, \quad \forall j \in J$$

(1-4) $$x_{ij} \geq 0, \quad \forall i \in I, \forall j \in J$$

(1-5) $$y_i \in \{0,1\}, \quad \forall i \in I$$

onde x_{ij} é a quantidade enviada da facilidade *i* para cliente *j*, e y_i representa a instalação ou não da facilidade *i*, ou seja, se $y_i = 1$, então a facilidade *i* será instalada; caso contrário, $y_i = 0$. As inequações (1-2) asseguram que nenhum cliente será atendido por uma facilidade fechada e que o total da demanda atendida pela facilidade não ultrapassa a sua capacidade. As expressões (1-3) asseguram que a demanda de cada cliente será satisfeita. As variáveis (1-4) asseguram que as quantidades transportadas não serão negativas, e (1-5) representam as varáveis inteiras. O PLC é um problema combinatório que pertence à classe de problemas NP-Difícel [36].

1.2 Métodos utilizados na resolução do PLC

O PLC tem sido amplamente considerado na literatura. Existem vários algoritmos exatos e heurísticos que podem ser encontrados.

1.2.1 Algoritmos exatos

Bartezzaghi e Colorni e Palermo [10] apresentaram um algoritmo de busca, sobre um árvore, baseado em um limite inferior derivado do Problema de Transporte, e forneceram resultados computacionais de problemas com dimensões de até 12 facilidades e 40 clientes.

Bitran, Chandru, Sempolinski e Shapiro [14] aplicaram uma técnica chamada de otimização inversa para o problema e apresentaram resultados computacionais para problemas de até 10 facilidades e 20 clientes.

Baker mostrou em [7] um algoritmo baseado em uma parte dual da formulação linear do problema e forneceu resultados de problemas de até 40 facilidades e 80 clientes.

Van Roy [45] apresentou um algoritmo baseado na técnica *Cross Decomposition*. Ele apresentou resultados computacionais para problemas de até 100 facilidades e 200 clientes.

Beasley [11] incorporou testes de reduções e um limite inferior baseado em Relaxação Lagrangeana em um procedimento numa árvore de busca.

Baker e Baía [8] apresentam um método *Branch and Bound* para resolver o problema. O método utiliza Relaxação Lagrangeana nas restrições de demanda para fornecer o limite inferior. Também apresenta um maneira para reduzir a capacidade das facilidades, apertando com isso o *gap* entre o limite superior e o limite inferior. Apresentam resultados computacionais para problemas com dimensão de até 50 facilidades por 50 clientes.

Ardal [3] propôs diversos cortes e os utilizou na resolução do primeiro nó da ferramenta MINTO sobre o CPLEX. Ela resolveu problemas com dimensão de até 75 facilidades por 100 clientes.

1.2.2 Algoritmos heurísticos

Jacobsen [30] apresentou as heurísticas ADD, DROP, SHIFT, ALA, and VSM para o problema, baseado em generalizações de heurísticas para o Problema de Localização não Capacitado. Domschke e Drexl [23] estenderam essas heurísticas, considerando situações em que as capacidades das facilidades são diferentes.

Barcelo e Casanovas [4] apresentaram algoritmos heurísticos, baseados em Relaxação Lagrangeana, para problemas em que cada cliente somente pode ser atendido por uma facilidade.

Barcelo [5] apresentou um algoritmo baseado em gerações de restrições, para tentar mover a solução da relaxação linear perto da ótima.

Beasley [12] desenvolveu uma heurística genérica, baseada em relaxação Lagrangeana, para problemas de Localização. Essa heurística foi especializada para vários tipos de problemas de localização, inclusive o de localização capacitado.

Campêlo Neto [17] apresentou um método heurístico onde utiliza testes de Redução e *interchange heuristics*.

Bornstein e Azlán [15] utilizaram *simulated annealing* e testes de redução para encontrar uma boa solução para o problema.

1.2.3 Outros trabalhos

Cornuejols, Sridharan e Thizy [20] compararam várias formas de relaxação Lagrangeana do *PLC*, e relaxações obtidas pela remoção de diferentes conjuntos de restrições. Também fizeram um abrangente estudo computacional usando essas relaxações e algumas heurísticas propostas na literatura.

Guignard-Spielberg e Kim [28] apresentaram um limite inferior para o problema baseado em Relaxação Lagrangeana. Resultados computacionais foram apresentados para problemas com até 20 facilidades por 35 clientes.

Baker [6] apresentou uma análise teórica, relativa ao fortalecimento da relaxação linear da formulação mista do problema.

1.3 Objetivos

Como dito anteriormente o *PLC* é um problema tipicamente de planejamento. Essa informação é essencial para a decisão de como atacar sua solução. Quando uma instituição se depara com um *PLC*, seu objetivo é encontrar uma solução para o problema, de preferência, com o menor custo possível, em um tempo viável. O tempo viável, nesse caso, vai desde a tomada de decisão de se instalar facilidades até o início de sua instalação. Com isso, o tempo viável dependerá de cada problema e instituição, podendo ter uma variação de dias, semanas, meses ou até anos. O *PLC* pode surgir, ainda, no estudo de viabilidade de um investimento, em que o tempo viável também pode levar de dias a anos.

De posse dessas informações, é possível concluir que o objetivo da resolução de um PLC é encontrar uma solução para o problema, com o menor custo possível, e dentro de um tempo viável para a instituição.

Este objetivo choca-se, de certa maneira, com grande parte dos métodos que estão sendo desenvolvidos para se encontrar a solução do problema, métodos puramente heurísticos. Esses métodos buscam a qualidade da solução e tempos computacionais rapidíssimos. Um dos problemas desses métodos é que esses objetivos são conflitantes, ou seja, quanto menor o tempo computacional mais difícil fica assegurar a qualidade da

solução. Outros fatos questionáveis são a diversidade de problemas em que os métodos são testados e as dimensões desses problemas.

Normalmente, os métodos puramente heurísticos são testados em um conjunto de problemas definidos na literatura, os quais têm características específicas. Por exemplo, somente problemas Euclidianos, isto é, problemas para os quais vale $c_{ij} \leq c_{ik} + c_{kj}$. Essas diferenças podem fazer com que um método tenha erros abaixo de 1%, em relação à solução ótima, para alguns conjuntos de problemas, mas tenha erros enormes para outros tipos de problemas.

Sabe-se que o *PLC* é um problema *NP-hard* [36], ou seja, um problema muito difícil de ser resolvido computacionalmente. Assim sendo, não é aconselhável um método que vise somente à solução ótima.

Com base nessas informações, foi traçado o objetivo deste trabalho: desenvolver um método que procure uma solução ótima ou ε-ótima para resolução do *PLC* mas que permita também encontrar uma boa solução, de forma heurística. Diz-se que uma solução é ε-ótima quando a solução fornecida tem um valor da função objetivo que é no máximo ε% do valor ótimo. Resumindo, a proposta do presente trabalho é:

1. Ser capaz de resolver, de forma exata e eficiente (rapidamente), os problemas encontrados na literatura: os problemas de Kühn & Hamburger [32], Beasley [11], Cornuejols, Sridharan e Thizy [20] e Ardal [3].
2. Ser capaz de resolver, em tempo viável, problemas teste complexos, não contemplados pela literatura. As dificuldades desses problemas são derivadas pela combinação das seguintes características:
    - ¬ Problemas com dimensões maiores do que as encontradas na literatura, principalmente no que se refere ao número de facilidades, isto é, ao número de variáveis binárias (0 ou 1). O objetivo, neste caso, é aumentar a tratabilidade dos Problemas de Localização Capacitado.
    - ¬ Problemas não Euclidianos.
    - ¬ Problemas com muitas variações em suas formulações.

3. Ser, em geral, melhor que o *software* comercial XPRESS (www.dashoptimization.com) na resolução do *PLC*, definido na seção 1.1, considerando-se principalmente problemas de grande porte.
4. Fornecer uma solução com boa qualidade no início da busca.
5. Ser capaz de encontrar soluções ótimas e/ou ε-ótimas.

Os resultados dos objetivos traçados para a solução do problema serão:

1. obter uma solução imediata com boa qualidade, possibilitando que o método seja aplicado a algum problema operacional;
2. parar somente quando se tiver encontrado uma solução ótima ou ε-ótima, ou ainda, quando o tempo viável para o problema for atingido. Caso o problema encontre a solução ótima ou ε-ótima, o objetivo da resolução foi atingido por completo. Quando a resolução for interrompida porque o tempo limite do problema foi atingindo, não é possível saber se a solução ótima foi encontrada. Contudo, sabe-se a máxima distância da solução encontrada, para a solução ótima. Esse cálculo é realizado utilizando o limite inferior do problema.

Alguns artigos, que servem de base para o desenvolvimento do método, podem ser encontrados em Valiati e Bornstein [41], [42], [43] e [44]. Outras informações importantes podem ser obtidas em Valiati [39] e Valiati [40].

1.4 Seqüência do trabalho

No capítulo 2 serão examinados testes de redução baseados em técnicas ADD/DROP. Testes de redução procuram reduzir o número de soluções a serem examinadas. Uma função $\Delta(.)$ que permite o balanço do aumento/decréscimo dos custos fixos e variáveis, devido à abertura/fechamento de uma facilidade é definida. Existem intervalos para $\Delta(.)$, onde uma decisão ótima de abertura ou fechamento de facilidades pode ser tomada. Na maioria dos casos, existe um intervalo em $\Delta(.)$ onde não existe decisão ótima. A dinâmica de abertura/fechamento de facilidades realimenta a redução do intervalo, fortalecendo o potencial de novas decisões ótimas. Ocasionalmente, o *intervalo* fecha, levando assim a uma solução completa. De qualquer modo, esta situação é muito especial para problemas grandes. Um grande número de facilidades indefinidas reduz o impacto da abertura/fechamento de uma determinada facilidade,

enfraquecendo o potencial dos procedimentos de ADD/DROP.

A tentativa de resolução de um problema NP-Difícil, de forma exata, implica em uma drástica redução no número de soluções a serem examinadas. Para isso é preciso que haja cortes inteligentes. Uma das formas de se realizar cortes é a utilização de limites inferiores. No Capítulo 3, serão apresentados dois métodos para se encontrar o limite inferior de um dado *PLC*. Os dois algoritmos serão utilizados no método desenvolvido para resolver o *PLC*, não necessariamente ao mesmo tempo. Além da utilização dos limites inferiores para realização de cortes, são utilizados testes de redução. Dois deles são apresentados no Capítulo 2, e os outros quatro são apresentados no Capítulo 3, estes últimos associados aos limites inferiores. Outra técnica para realização de cortes é a utilização do número mínimo e máximo de facilidades abertas na solução ótima. As formas de obtenção desses números são desenvolvidas no Capítulo 3 e no Capítulo 4.

A principal ênfase deste trabalho é à qualidade da solução. No Capítulo 4 será desenvolvido um método para se obter uma solução ótima. A principal motivação para o desenvolvimento do método exato é que geralmente os *PLC* não requerem uma solução imediata (em segundos ou minutos). Como os Problemas de Localização Capacitado freqüentemente envolvem custos altíssimos, um a aproximação de 1% a 5% do ótimo gerado por um método heurístico pode representar um custo relativamente grande.

Os testes de redução abrem e fecham facilidades *a priori*. Contudo, como será visto no Capítulo 2, para o seu bom desempenho, necessitam da abertura e ou do fechamento de facilidades. Este possível ciclo vicioso será quebrado utilizando um algoritmo *Branch and Bound* (algoritmo exato). No capítulo 4, será apresentado o método *Branch and Bound* desenvolvido, que utiliza, além de testes de redução, a relaxação Lagrangeana e relaxação linear das variáveis inteiras para o cálculo do limite inferior. Procura-se realizar um grande número de podas na árvore de busca. A busca é realizada, fazendo-se uma combinação de busca em profundidade e busca em largura. O *backtracking* busca um nó onde a probabilidade de erro na escolha do lado da ramificação seja grande. A condição de parada do método é identificada quando a lista de nós a serem percorridos é vazia. A maneira como é realizada a busca na árvore será vista na seção 4.6, o *backtracking* e a condição de parada do método serão vistos na

seção 4.5. As regras de ramificação serão apresentadas na seção 4.3. A ramificação ocorre, se todos os testes de redução falharem. Nesse caso, escolhe-se uma facilidade a ter seu valor fixado (0 ou 1). Esta escolha é realizada através de heurísticas, que são facilmente calculadas, pois usam os resultados dos testes que falharam e dos limites inferiores, isto é, resultados já calculados. A utilização de heurísticas fornece uma solução próxima da ótima já nas primeiras buscas em profundidade, isto é, fornece um bom limite superior para o problema no início da exploração da árvore. O método, além de garantir a otimalidade da solução, fornece, geralmente, devido as heurísticas de ramificação, uma solução próxima da ótima logo no início. O método também proporciona uma medida de qualidade da solução, já que em um determinado ponto do algoritmo pode informar qual a distância máxima que a solução encontrada está da solução ótima. Este cálculo é realizado com base nos limites inferiores calculados. Isso quer dizer que o usuário pode parar o algoritmo em um determinado momento, mesmo sem saber se a solução obtida é realmente a ótima, mas tendo um parâmetro de sua qualidade. Na seção 4.9 serão apresentados os resultados computacionais obtidos, comparando-os com os obtidos por outros métodos e com o *software* comercial XPRESS.

Os anexos mostram outras pesquisas que foram desenvolvidas. No Anexo 1 será desenvolvido um algoritmo para resolução do Problema de Transporte Capacitado. Embora o Problema de Transporte não seja o objetivo desse trabalho, é uma das principais ferramentas para se resolver o *PLC*. O problema de transporte é utilizado na realização de quatro testes de redução e no cálculo de um dos limites inferiores utilizados pelo *PLC*. Assim sendo, o método para resolver o *PLC*, que será apresentado no capítulo 4, é altamente dependente do cálculo do Problema de Transporte Capacitado. Um método ruim para o cálculo do Problema de Transporte pode levar o método para resolução do *PLC* ao insucesso.

No Anexo 2 são mostrados os métodos de análise de agrupamentos. Embora esses métodos não garantam a otimalidade da solução, têm como objetivo oferecer uma alternativa para problemas grandes e de difícil resolução. Os métodos que serão apresentados procuram explorar as características do problema. Procura-se fazer diversas combinações, produzindo, com isso, várias formas de se conduzir o processo de agrupamento. Os métodos passam basicamente por três fases: a primeira seria a de

agrupamento, na qual são formados grupos contendo facilidades e clientes; a segunda seria a da resolução dos problemas gerados em cada grupo; e a terceira seria um refinamento da solução gerada. É importante observar que, mesmo não garantindo a otimalidade, o método procura dar mais ênfase a ela do que ao tempo de processamento. A preocupação com relação ao tempo está no interessse de se tentar obter uma solução em um tempo viável para as características a que o problema se propõe. Serão apresentados os resultados computacionais dos métodos propostos, os quais obtiveram resultados muito bons para problemas que abrem, na solução ótima, um número de facilidades maior ou igual a 20% do total de facilidades existentes.

Com o intuito de resolver o *PLC* de forma ótima, no Anexo 3, é realizado um estudo sobre o método de Benders, já que ele propõe uma decomposição para problemas de programação linear mista. Nesse capítulo, será mostrado como se pode aplicar o método de Benders para resolver o *PLC*. Também será apresentado um método que foi desenvolvido, baseado no método de Benders. O método utiliza três tipos de decomposições do problema original, duas primais e uma dual. O método se baseia na decomposição de Benders e resolve subproblemas do problema original. Procura-se fazer cortes mais precisos no problema mestre de Benders. Com isso, a tendência é se obter a solução ótima do problema com o acréscimo de poucas restrições ao problema mestre. Assim sendo, além de ser necessário resolver um número menor de problemas mestres, esses ficam mais fáceis. Serão mostrados alguns resultados computacionais e algumas propostas de melhoria.

Capítulo 2

2 Testes de Redução

Nesse capítulo, serão apresentados dois testes de redução, cujo objetivo principal é diminuir a dimensão do *PLC*. Os testes de redução aplicados ao problema determinam a abertura ou o fechamento de uma facilidade *a priori*. Com isso, é possível reduzir a dimensão do problema original. Estes testes de abertura de uma determinada facilidade, denominados *O-Teste*, e de fechamento, denominados *C-Teste*, são baseados em critérios que levam em conta tanto o custo fixo como o custo variável do *PLC*. Além dos testes será apresentado um exemplo onde somente com a aplicação dos testes consegue-se definir o *status* de todas as facilidades.

Seja $w(K) = min\{\Sigma_{k\in K}\Sigma_{j\in J}\, c_{kj}x_{kj} \mid x \in X(K)\}$ o valor do Problema de Transporte associado ao *PLC*, onde $K \subseteq I$ é o conjunto de facilidades abertas e $X(K) = \{x \geq 0 | \Sigma_{j\in J}\, x_{kj} \leq a_k, \forall\, k\in K; \Sigma_{k\in K}x_{kj} = b_j, \forall j \in J\}$, o conjunto de soluções viáveis. Considera-se que, se $X(K) = \varnothing$, então $w(K) = \infty$. A função $w(K)$ tem uma importante propriedade, a supermodularidade. Uma função $w(K)$ é supermodular se $w(K) - w(K\cup i) \leq w(K') - w(K' \cup i)\ \forall\ K' \subseteq K$ e $\forall\ i\notin K$. Mais detalhes sobre o assunto podem ser encontrados em Wolsey [46].

Para $i \notin K$, $\delta_i(K) = w(K) - w(K\cup i) \leq 0$ é o cálculo do aumento/decréscimo do custo no Problema de Transporte, se for fechada/aberta a facilidade i. $\Delta_i(K) = f_i - \delta_i(K)$ é o balanço entre o custo fixo e o custo variável com relação à facilidade i.

Sejam K_0 e K_1, respectivamente, o conjunto de facilidades fechadas e abertas, isto é, $K_0 = \{i\in I \mid y_i=0\}$ e $K_1 = \{i \in I \mid y_i= 1\}$. Facilidades, cujos valores não estejam definidos, pertencem a $K_2 = I - K_0 - K_1$. Inicialmente, tem-se $K_0 = K_1 = \varnothing$ e $K_2 = I$. Os testes de redução são realizados da seguinte forma:

- O-Teste: Se $\Delta_i(K_1 \cup K_2 - i) \leq 0$ então $y_i = 1$.
- C-Teste: Se $\Delta_i(K_1) \geq 0$ então $y_i = 0$.

Maiores detalhes sobre os testes podem ser encontrados em Bornstein e Azlan [15].

Os testes fornecem critérios para alocar facilidades aos conjuntos K_0 e K_1. A supermodularidade garante que o aumento de K_0 e K_1, isto é, o decréscimo de $K_1 \cup K_2$ não aumentará $\Delta_i(K_1 \cup K_2 - i)$. Da mesma maneira a supermodularidade garante que, o aumento de K_1 não decrescerá o valor de $\Delta_i(K_1)$. Assim, o balanço entre o custo fixo e o variável, detectado pelos *O* e *C-Testes*, tende a ficar mais significativo a cada decisão referente à abertura ou fechamento de uma determinada facilidade, ou seja, mais negativo para o *O-Teste* e mais positivo para o *C-Teste*. Este fato garante que a decisão tomada pelos testes é ótima.

A Figura 1 ilustra como os testes trabalham. As setas no topo da figura mostram como $\Delta_i(K_1)$ e $\Delta_i(K_1 \cup K_2 - i)$ se movem à medida que aumenta o número de iterações de um algoritmo baseado nos *O* e *C-Testes*. Se na aplicação dos testes $K_2 = \{i\}$, então, $\Delta_i(K_1) = \Delta_i(K_1 \cup K_2 - i)$, significando que $y_i = 1$, se $\Delta_i(K_1) < 0$ ou $y_i = 0$, se $\Delta_i(K_1) > 0$. Se $\Delta_i(K_1) = 0$, a facilidade *i* pode ser aberta ou fechada. Neste caso, consegue-se gerar uma solução ótima, isto é, o *O-Teste* e o *C-Teste* conseguiram fixar o *status* (definir a situação) de todas as facilidades, fazendo $K_2 = \varnothing$.

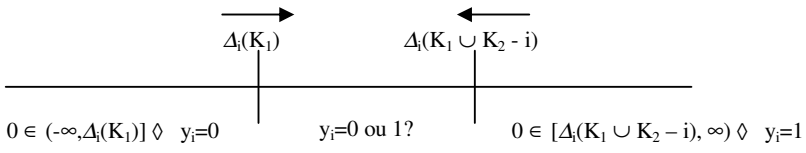


Figura 1 – Funcionamento dos testes de redução

Os testes alocam facilidades aos conjuntos K_0 e K_1, diminuindo o conjunto K_2. Se na aplicação dos *O* e *C-Testes*, todos os elementos de K_2 forem alocados aos conjuntos K_0 e K_1, a solução gerada é uma solução ótima. Contudo, na maioria dos problemas, estes testes não conseguem fazer $K_2 = \varnothing$. Conforme mostrado na Figura 1, quando o algoritmo não consegue fazer $K_2 = \varnothing$, fica um intervalo, chamado de *gap* de indefinição, que representa a região de indecisão, ou seja, representa as facilidades $i \in K_2$, para as quais $\Delta_i(K_1) < 0 < \Delta_i(K_1 \cup K_2 - i)$. É importante observar que, para a

realização do *C-Teste*, é necessário que K_1 seja viável, ou seja, $X(K_1) \neq \varnothing$. Quando $X(K_1) = \varnothing$, tem-se que $w(K_1) = \infty$, implicando que $\Delta_i(K_1)$ não pode ser calculado. Logo, só é permitido aplicar o *C-Teste* em casos onde existem elementos em K_1, tal que $X(K_1) \neq \varnothing$, ou seja, se existir um número de facilidades abertas tal que estas atendam às demandas dos clientes.

Em Valiati [39], podem ser encontradas aproximações para os testes. As aproximações são métodos para encontrar, de forma rápida, limites inferiores e superiores para os testes. Isso pode ser interessante, porque, para a realização dos testes, é necessária a resolução de Problemas de Transporte. Esta resolução, para problemas grandes, exige um tempo computacional considerável. As aproximações permitem maior rapidez do que os testes exatos, contudo, são menos poderosas, pois enquadram menos facilidades. Essas aproximações podem ser utilizadas no método que será apresentado no Capítulo 4.

Em Valiati [39], Também pode ser encontrado, um estudo, onde é realizada uma classificação de problemas e avaliado a forma como se comportam os testes de redução sobre eles. Através desse estudo, é possível verificar se os problemas são propensos ao sucesso do *O-Teste*, do *C-Teste*, de ambos ou de nenhum.

2.1 Aplicação do O-Teste e do C-Teste

Primeiramente, será mostrado, nas seções 2.1.1 e 2.1.2, por que os testes não são suficientemente fortes para garantir que o *gap* feche, ou seja, serão identificados quais os principais problemas que levam à falha de ambos os testes, na tentativa de definir a que conjunto pertence uma certa facilidade. Depois será mostrado, na seção 2.1.3, como o sucesso de um teste potencializa o outro, por exemplo, o sucesso do *O-Teste*, ou seja, a abertura de uma facilidade aumenta K_1, aumentando a chance de o *C-Teste* funcionar, fechando novas facilidades. Na seção 2.1.4 serão apresentados dois algoritmos utilizando dos testes de redução. Pode-se verificar que, em alguns casos, o uso destes algoritmos, ou seja, a aplicação sucessiva do *O-Teste* e do *C-Teste*, pode definir o *status* de todas as facilidades.

2.1.1 Dificuldades no fechamento do gap

Será examinado, a seguir, um caso completo de aplicação do *O-Teste* e *C-Teste,* verificando as razões de suas falhas.

Exemplo 1: Falhas no fechamento do *gap*.

Será apresentado, um exemplo, com quatro facilidades e seis clientes, sem restrições de capacidade para as facilidades. Com a eliminação das restrições de capacidade, tem-se que a demanda de um cliente *j* é atendida somente por uma facilidade *i* (veja Muller. et al [33]). Com isso, os dados relativos ao custo de transporte da facilidade *i* para o cliente *j* podem ser representados na forma $c_{ij}x_{ij}$, onde o valor de x_{ij} é toda a demanda do cliente *j*, logo $c_{ij}x_{ij} = c_{ij}b_j$. Os dados relativos a $c_{ij}b_j$ e f_i são apresentados na Tabela 1, onde F_i $i = 1,..,4$ representam as facilidades, C_j $j = 1,...,6$ os clientes e f_i $i = 1,...,4$ representam os custos fixos. O cruzamento da linha F_i com a coluna C_j representa o custo de a facilidade *i* atender toda a demando do cliente *j*. Por exemplo, o elemento na linha 2, coluna 2, cujo valor é 20, representa o custo de a facilidade F_1 atender toda a demanda do cliente C_1.

Centros Demanda/ Centros Oferta	C_1	C_2	C_3	C_4	C_5	C_6	f_i
F_1	20	20	20	10	10	20	*10*
F_2	10	10	10	20	20	10	*20*
F_3	30	30	15	30	20	5	*10*
F_4	*5*	*5*	*30*	*30*	*30*	*30*	*20*

Tabela 1 – Dados do PLC sem restrição de capacidade

Na resolução do problema, será primeiramente aplicado o *O-Teste* para todas as facilidades e, posteriormente, o *C-Teste*. Para a realização dos testes, é necessário calcular o Problema de Transporte associado a cada teste. Para que se possa ter uma visualização mais precisa do que está acontecendo, quando se aplicam os testes, apresentam-se na Figura 2 as resoluções dos Problemas de Transporte necessários aos testes. Os círculos representam as facilidades, os quadrados, os clientes, e o peso do arco que liga uma facilidade a um cliente representa o custo de atendimento do cliente pela facilidade.

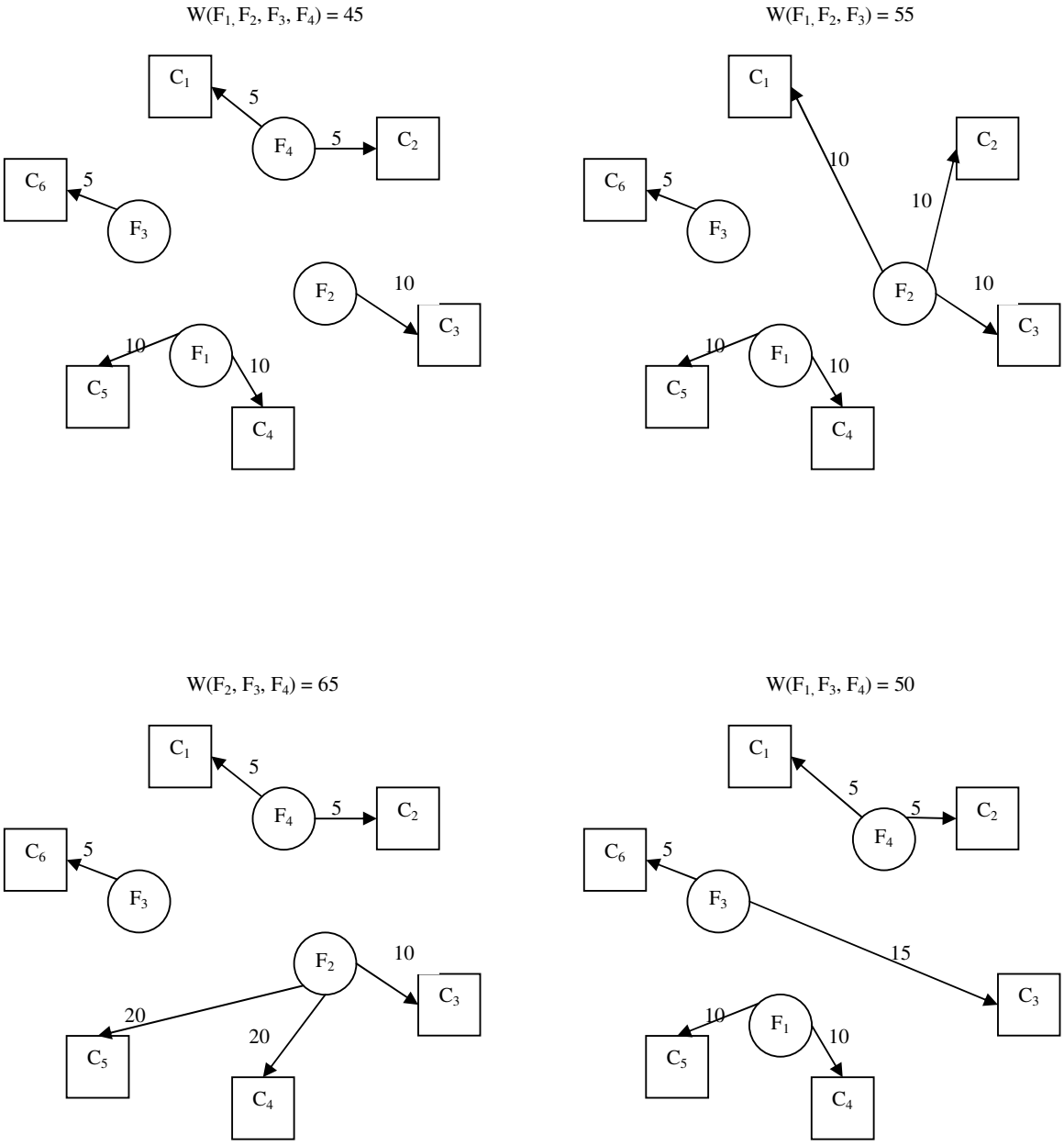


Figura 2 – Resolução de problemas de transporte

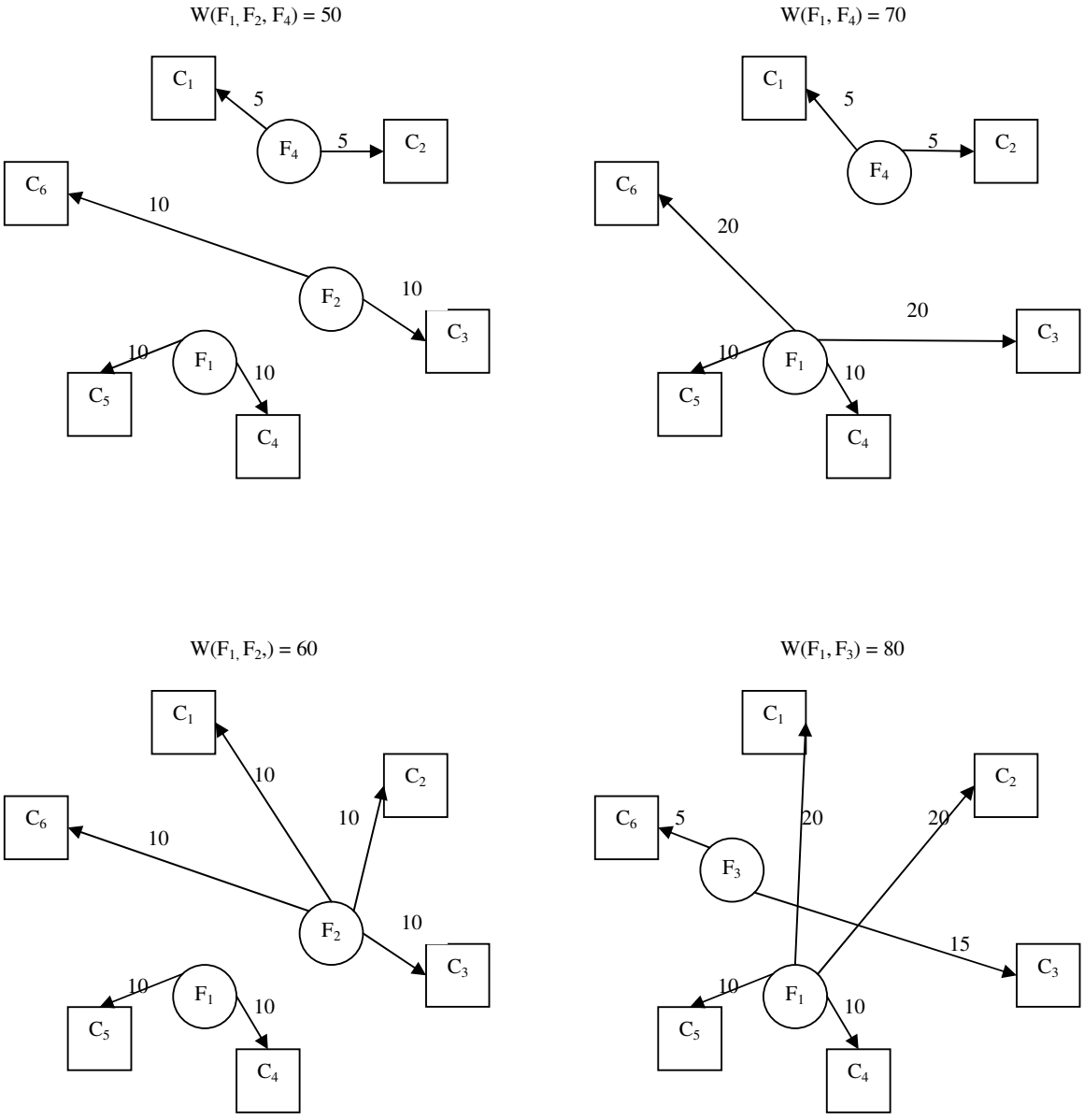


Figura 2 - Resolução de problemas de transporte - continuação

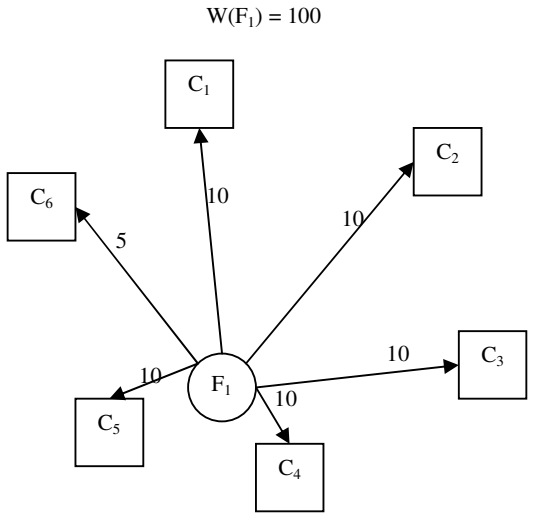


Figura 2 - Resolução de problemas de transporte - continuação

K_0 e K_1 são inicializados como vazio, enquanto K_2 é inicializado com todas as facilidades do problema. Primeiro, realiza-se os *O-Testes* $\forall i \in K_2$, mostrados na Tabela 2. Para facilidade de notação, denota-se no texto F_i por i.

O-Teste para i = 1	$K_0 = K_1 = \varnothing$ $K_2 = I = \{1,2,3,4\}$
$\delta_1(K_1 \cup K_2 - 1) = 65 - 45 = 20$	
$\Delta_1(K_1 \cup K_2 - 1) = 10 - 20 = -10$	
Logo, O-Teste de 1 teve sucesso	
O-Teste para i = 2	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_2(K_1 \cup K_2 - 2) = 50 - 45 = 5$	
$\Delta_2(K_1 \cup K_2 - 2) = 20 - 5 = 15$	
Logo, O-Teste de 2 falhou	
O-Teste para i = 3	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_3(K_1 \cup K_2 - 3) = 50 - 45 = 5$	
$\Delta_3(K_1 \cup K_2 - 3) = 10 - 5 = 5$	
Logo, O-Teste de 3 falhou	
O-Teste para i = 4	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_4(K_1 \cup K_2 - 4) = 55 - 45 = 10$	
$\Delta_4(K_1 \cup K_2 - 4) = 20 - 10 = 10$	
Logo, O-Teste de 4 falhou	

Tabela 2 – Aplicação do *O-Teste*

Como visto na Tabela 2, com a aplicação do *O-Teste*, obteve-se sucesso somente para a facilidade *1*, deixando $K_0 = \varnothing$, $K_1 = \{1\}$ e $K_2 = \{2,3,4\}$. Como o problema não tem restrição de capacidade, K_1 é viável, logo, pode-se aplicar os *C-Testes* $\forall\, i \in K_2$, mostrados na Tabela 3.

C-Teste para i = 2	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_2(K_1) = 100 - 60 = 40$	
$\Delta_2(K_1) = 20 - 40 = -20$	
Logo, C-Teste de 2 falhou	
C-Teste para i = 3	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_3(K_1) = 100 - 80 = 20$	
$\Delta_3(K_1) = 10 - 20 = -10$	
Logo, C-Teste de 3 falhou	
C-Teste para i = 4	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_4(K_1) = 100 - 70 = 30$	
$\Delta_4(K_1) = 20 - 30 = -10$	
Logo, C-Teste de 4 falhou	

Tabela 3 – Aplicação do *C-Teste*

A Tabela 3 mostra que nenhum *C-Teste* obteve sucesso, indicando a falha dos testes $\forall\, i \in K_2$.

O Exemplo 1 informa que mesmo para o problema sem restrições de capacidade, os testes de redução não conseguiram fechar o *gap*. Isso acontece por causa da distribuição de clientes a facilidades. Cada cliente procura ser atendido pela facilidade mais interessante, ou seja, a facilidade que forneça o custo variável menor. Com isso, a indefinição do *O-Teste* costuma ocorrer para problemas em que $K_1 \cup K_2$ é muito grande, isto é, quando K_0 é pequeno. Neste caso, clientes são atendidos por facilidades que na verdade deveriam estar fechadas na solução ótima. Assim, ao se retirar uma facilidade *i*, a falta que esta faz não é sentida em toda sua extensão, pois os clientes que deveriam ser atendidos por ela são atendidos por uma das muitas facilidades que possivelmente (na solução ótima) deveriam estar fechadas. Por exemplo, para $K_1 = \{1\}$ e $K_2 = \{2,3,4\}$, no *O-Teste* para $i = 2$, a qual é aberta na solução ótima atendendo os clientes C_1, C_2, C_3 e C_6, as facilidades *3* e $4 \in K_2$ que devem estar fechadas na solução

ótima "roubam" clientes de *2* e amenizam a falta que esta facilidade faz. A facilidade *4* atende C_1 e C_2 e a facilidade *3* atende C_6 e C_3. Com isto, o aumento nos custos de transporte ,ocasionado pela retirada de *2*, é pequeno e, conseqüentemente, o *O-Teste* deixa de ter sucesso para *2*. Caso qualquer uma das facilidades *3* ou *4* fossem fechadas, o *O-Teste* para *2* teria sucesso.

O problema inverso ocorre com o *C-Teste*. Neste caso, um K_1 muito pequeno provoca uma sensibilidade muito grande do custo de transporte, quando é inserida uma facilidade, dificultando o fechamento desta. Por exemplo, quando com $K_1 = \{1\}$ e $K_2 = \{2,3,4\}$, se aplica o C-Teste para $i = 4$, o ganho com a abertura de *4*, em termos de redução do custo de transporte, é de 30, superando o custo fixo 20 (indicando a falha do teste), enquanto que o ganho real é de somente 10, se a facilidade *2* estivesse aberta, conforme a solução ótima.

Resumindo, pode-se dizer que o *O-Teste* funciona bem, quando existe grande sensibilidade do custo de transporte à retirada/adição de uma facilidade $i \in K_2$. Por sensibilidade entende-se a variação dos custos de transporte. Ora, a sensibilidade é grande, quando o conjunto $(K_1 \cup K_2)$ de facilidades onde atua *i* é pequeno. Já o C-Teste funciona bem, quando existe pequena sensibilidade dos custos de transporte à retirada/adição de uma facilidade $i \in K_2$. A sensibilidade é pequena, quando o conjunto K_1 de facilidades onde atua *i* é grande.

O que enfraquece os testes e evita o fechamento do *gap* é que, na etapa inicial, o *O-Teste* e o *C-Teste* são aplicados, respectivamente, a conjuntos K_0 e K_1 muito pequenos.

2.1.2 Generalização das dificuldades do fechamento do *gap* para um *PLC* qualquer

Os problemas identificados na seção anterior não necessariamente acontecem para todas as facilidades. Podem existir facilidades e clientes agrupados de tal maneira que a decisão associada aos testes seja fácil. Por outro lado, podem existir facilidades e clientes que constituem uma *região de indecisão*, onde problemas como os vistos em 2.1.1 ocorrem.

A Figura 3 apresenta um problema onde as facilidades são representadas por círculos e os clientes por quadrados, consideram-se aplicados os testes de redução. Os círculos cinza representam as facilidades que os testes conseguiram abrir, e os círculos pretos representam as facilidades que os testes conseguiram fechar. As regiões R_1, R_2 e R_3 representam *regiões de indecisão*. Uma decisão tomada com o objetivo de solucionar a indecisão na região R_1 não influenciará em nada a resolução das regiões R_2 e R_3. Já uma decisão tomada com o objetivo de solucionar a indecisão de uma das regiões R_2 ou R_3 pode influenciar na resolução da outra. Isso porque ambas têm uma facilidade em comum, e a decisão relativa a esta pode influenciar positivamente o sucesso dos testes nas duas regiões, conforme será visto na seção 2.1.3. Isso mostra que existe um grau de dependência entre as *regiões de indecisão*, o qual depende do número de facilidades em comum que as regiões têm.

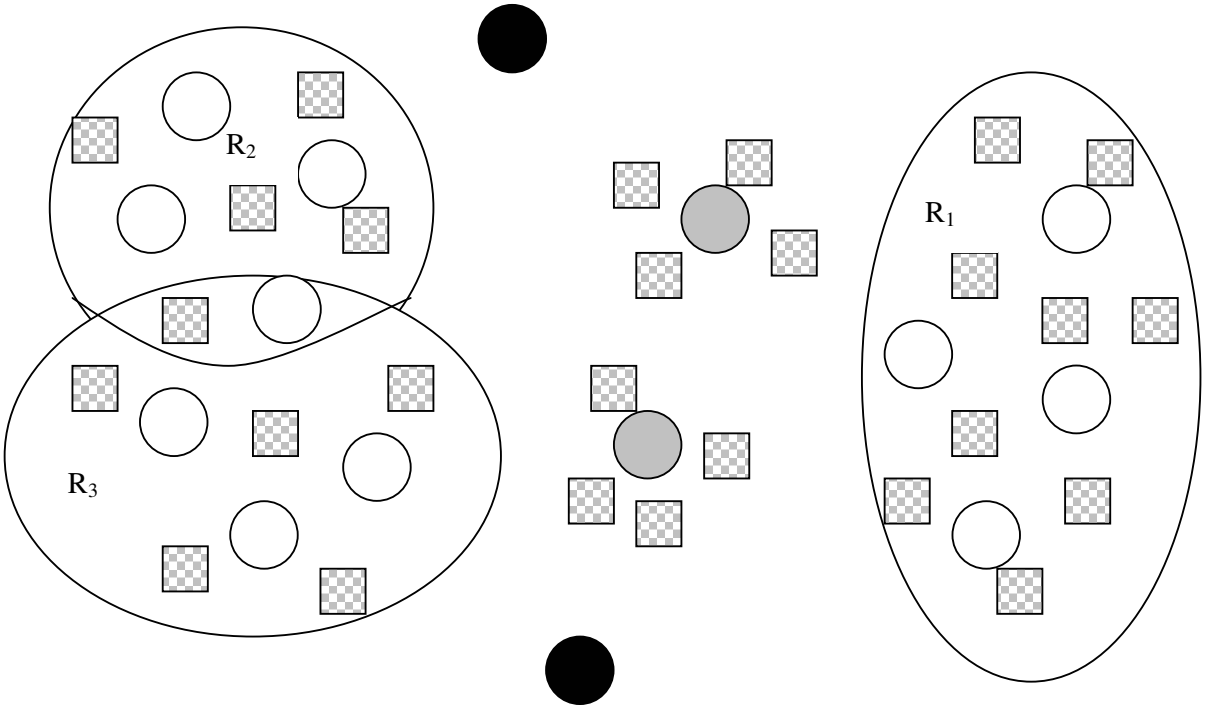


Figura 3 – Comportamento dos teste de redução

A decisão relativa à abertura/fechamento de uma facilidade dentro de uma *região de indecisão*, mesmo que temporária, isto é, a decisão pode ser revertida em etapas posteriores do algoritmo, aumenta o potencial dos testes com relação a outras facilidades. Isso porque, caso uma facilidade seja fechada, o conjunto $K_1 \cup K_2$ ficará

menor, aumentando assim as possibilidades de sucesso do *O-Teste*. Da mesma forma, caso uma facilidade seja aberta, aumentará o conjunto K_1, aumentando justamente as possibilidades de sucesso do *C-Teste*. Esta característica favorece o uso dos testes em conexão com um algoritmo *Branch and Bound*, que será visto no Capítulo 4.

2.1.3 Influência de um teste no outro

Nesta seção, será apresentado o relacionamento entre os testes, ou seja, como o sucesso de um influencia a resolução do outro.

Conforme será visto, o sucesso de um teste, que é a tomada de decisão de abrir ou fechar uma facilidade, influencia positivamente o sucesso do outro, ou seja, quando um teste obtém sucesso, ele aumenta as chances de sucesso do outro também. O sucesso do *C-Teste* diminui $K_1 \cup K_2$, fazendo com que no *O-Teste* a retirada de uma facilidade seja melhor sentida, ou seja, a falta da facilidade é mais representativa. Por outro lado, o sucesso do *O-Teste* aumenta K_1, implicando que no *C-Teste* a inserção de uma facilidade seja avaliada de forma mais realista.

Exemplificando, suponha que, ao se aplicar o *O-Teste* $\forall i \in K_2$, nenhum teste tenha tido sucesso, e aplicando-se o *C-Teste* este tenha sucesso para pelo menos um $i \in K_2$. Isto pode fazer com que uma nova aplicação do *O-Teste*, obtenha sucesso. O exemplo numérico, a seguir, reflete bem a influência de um teste no outro.

Exemplo 2: Influência de um teste no outro.

Os dados do exemplo são fornecidos na Tabela 4. Como no Exemplo 1 F_i $i = 1,...,4$ representam as facilidades, Cj $j = 1,...,6$ os clientes, e f_i $i = 1,...,4$ os custos fixos de instalação das facilidades. Como agora o problema é capacitado, o custo de transporte de uma facilidade para um cliente é unitário, ou seja, representa o custo de transporte da facilidade i para o cliente j de uma unidade. A demanda de cada cliente é representada por b_j $j = 1,...,6$, e a capacidade de cada facilidade por a_i $i = 1,...,4$.

Centros Demanda/ Centros Oferta	$\mathbf{C_1}$	$\mathbf{C_2}$	$\mathbf{C_3}$	$\mathbf{C_4}$	$\mathbf{C_5}$	$\mathbf{C_6}$	$\mathbf{f_i}$	a_i
$\mathbf{F_1}$	4	4	2.5	2.5	2	5	10	*31*
$\mathbf{F_2}$	2	2	1.25	5	4	2.5	20	*22*
$\mathbf{F_3}$	4.2	4.2	2.375	5.25	4	2.75	10	*20*
$\mathbf{F_4}$	1	1	3.75	7.5	6	7.5	20	*20*
b_j	*5*	*5*	*8*	*4*	*5*	*4*		

Tabela 4 – Dados do *PLC*

A Tabela 5 apresenta somente as soluções dos Problemas de Transporte necessários para a realização dos testes. Os conjuntos são inicializados da seguinte forma: $K_0 = K_1 = \varnothing$ e $K_2 = I = \{1, 2, 3, 4\}$. Primeiro realiza-se o *O-Teste*, até que não se tenha mais sucessos. A Tabela 6 mostra a realização do *O-Teste* $\forall i \in K_2$.

Problema de Transporte	**Solução**
$w(F_1, F_2, F_3, F_4)$	50
$w(F_1, F_2, F_3)$	60
$w(F_2, F_3, F_4)$	70
$w(F_1, F_3, F_4)$	60
$w(F_1, F_2, F_4)$	50
$w(F_1, F_4)$	70
$w(F_1, F_2)$	60
$w(F_1, F_3)$	90
$w(F_1)$	100

Tabela 5 - Resolução de alguns *w(K)*

O-Teste para i = 1	$K_0 = K_1 = \varnothing$ $K_2 = I = \{1,2,3,4\}$
$\delta_1(K_1 \cup K_2 - 1) = 70 - 50 = 20$	
$\Delta_1(K_1 \cup K_2 - 1) = 10 - 20 = -10$	
Logo, O-Teste de 1 teve sucesso	
O-Teste para i = 2	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_2(K_1 \cup K_2 - 2) = 60 - 50 = 10$	
$\Delta_2(K_1 \cup K_2 - 2) = 20 - 10 = 10$	
Logo, O-Teste de 2 falhou	
O-Teste para i = 3	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_3(K_1 \cup K_2 - 3) = 50 - 50 = 0$	
$\Delta_3(K_1 \cup K_2 - 3) = 10 - 0 = 10$	
Logo, O-Teste de 3 falhou	
O-Teste para i = 4	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_4(K_1 \cup K_2 - 4) = 60 - 50 = 10$	
$\Delta_4(K_1 \cup K_2 - 4) = 20 - 10 = 10$	
Logo, O-Teste de 4 falhou	

Tabela 6 - Aplicação do *O-Teste*

Com a realização dos *O-Testes*, obteve-se $K_0 = \varnothing$, $K_1 = \{1\}$ e $K_2 = \{2,3,4\}$. Agora se realiza o *C-Teste*, até que esse não tenha mais sucesso. A Tabela 7 mostra os resultados da aplicação dos *C-Testes*.

C-Teste para i = 2	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_2(K_1) = 100 - 60 = 40$	
$\Delta_2(K_1) = 20 - 40 = -20$	
Logo, C-Teste de 2 falhou	
C-Teste para i = 3	$K_0 = \varnothing$ $K1 = \{1\}$ $K_2 = I = \{2,3,4\}$
$\delta_3(K_1) = 100 - 90 = 10$	
$\Delta_3(K_1) = 10 - 10 = 0$	
Logo, C-Teste de 3 teve sucesso	
C-Teste para i = 4	$K_0 = \{3\}$ $K1 = \{1\}$ $K_2 = I = \{2,4\}$
$\delta_4(K_1) = 100 - 70 = 30$	
$\Delta_4(K_1) = 20 - 30 = -10$	
Logo, C-Teste de 4 falhou	

Tabela 7 - Aplicação do *C-Teste*

Com a realização dos *C-Testes*, fechou-se a facilidade *3*, ficando com $K_0 = \{3\}$, $K_1 = \{1\}$ e $K_2 = \{2,4\}$. Como, com a aplicação do *C-Teste*, obteve-se sucesso para $i = 3$, então, passa-se a realizar o *O-Teste* novamente $\forall\ i \in K_2$. Os resultados da nova aplicação dos testes estão na Tabela 8.

O-Teste para i = 2	$K_0 = \{3\}\ K1 = \{1\}\ K_2 = I = \{2,4\}$
$\delta_2(K_1 \cup K_2 - 2) = 70 - 50 = 20$	
$\Delta_2(K_1 \cup K_2 - 2) = 20 - 20 = 0$	
Logo, O-Teste de 2 teve sucesso	
O-Teste para i = 4	$K_0 = \{3\}\ K1 = \{1,2\}\ K_2 = I = \{4\}$
$\delta_4(K_1 \cup K_2 - 4) = 60 - 50 = 10$	
$\Delta_4(K_1 \cup K_2 - 4) = 20 - 10 = 10$	
Logo, O-Teste de 4 falhou	

Tabela 8 - Aplicação do *O-Teste*

Com a realização dos *O-Testes*, abriu-se a facilidade *2*, ficando com $K_0 = \{3\}$, $K_1 = \{1,2\}$ e $K_2 = \{4\}$. Com isso, pode-se aplicar novamente o *C-Teste* $\forall\ i \in K_2$. Os resultados da nova aplicação dos testes são mostrados na Tabela 9. Após a realização do *C-Teste* $\forall\ i \in K_2$, tem-se $K_0 = \{3,4\}$, $K_1 = \{1,2\}$ e $K_2 = \varnothing$, indicando o fechamento do *gap*, atingindo assim a solução ótima do problema.

C-Teste para i = 4	$K_0 = \{3\}\ K1 = \{1,2\}\ K_2 = I = \{4\}$
$\delta_4(K_1) = 60 - 50 = 10$	
$\Delta_4(K_1) = 20 - 10 = 10$	
Logo C-Teste de 4 teve sucesso	

Tabela 9 - Aplicação do *C-Teste*

2.1.4 Algoritmos que verificam o fechamento do *gap*

Os algoritmos que serão apresentados procuram através dos testes de redução encontrar a solução ótima de um *PLC*. Como dito anteriormente, utilizando-se o *O-Teste* e o *C-Teste*, somente se pode encontrar a solução ótima para o problema se, dado o impasse na aplicação de um tipo de teste, o outro teste obtiver sucesso, isto é, conseguir definir o *status* de pelo menos uma facilidade adicional, prosseguindo-se

assim até a definição do status de todas as facilidades Caso, desta forma, se consiga fechar o *gap*, tem-se uma solução ótima.

Os algoritmos que serão apresentados muitas vezes não conseguem encontrar a solução ótima, pois, como visto na seção 2.1.1, em muitos problemas os testes de redução não conseguem fechar o *gap*. Contudo, podem-se aplicar os algoritmos com o objetivo de reduzir a dimensão do problema.

2.1.4.1 Algoritmo de aplicação dos testes de redução

O Algoritmo 1 é um algoritmo geral para resolver o problema através do O-Teste e do C-Teste. O algoritmo usa, além dos dados do problema e dos conjuntos definidos, uma variável lógica, a qual é chamada de *Parada*. Esta variável representará a possibilidade ou não de sucesso dos testes. Ela assumirá o valor verdadeiro quando os testes não puderem ser realizados com sucesso para nenhuma facilidade pertencente a K_2. Parte-se do princípio que o problema é viável, ou seja, $\sum_{i \in I} a_i y_i \geq \sum_{j \in J} b_j$. O algoritmo trabalha da seguinte maneira: inicializa os conjuntos $K_0 = K_1 = \varnothing$ e $K_2 = I$ e começa a realização dos testes. Primeiro, é realizado o *O-Teste* $\forall\, i \in K_2$, passando, depois, para a realização dos *C-Testes*. Para que se possa realizar os *C-Testes*, é necessário que K_1 seja viável. Sempre que um teste obtiver sucesso para algum $i \in K_2$ o outro deve ser testado, conforme foi mostrado na seção 2.1.3. Este controle é realizado no algoritmo através da variável *Parada*, que neste caso recebe o valor Falso. Caso o *O-Teste* ou o *C-Teste* não tenham sucesso para nenhum $i \in K_2$, o algoritmo termina, pois a variável *Parada* assume o valor verdadeiro. Este processo de realização dos *O-Testes*, e depois *C-Testes*, repete-se até que eles consigam fechar o *gap*, ou até quando $\forall\, i \in K_2$ nenhum teste obtiver sucesso.

Algoritmo 1: aplicação dos testes de redução

$K_2 \Downarrow I$
$K_0 \Downarrow K_1 \Downarrow \varnothing$
Parada $\Downarrow$ Falso
Enquanto $K_2 \neq \varnothing$ e Parada = Falso ***fazer*** *{ laço 1 }*
Inicio
    Parada $\Downarrow$ Verdadeiro
    calcular $w(K_1 \cup K_2)$
    para cada $i \in K_2$ ***fazer*** *{laço 2 }*
    Inicio
        calcular $w(K_2 \cup K_1 - \{i\})$
        $\delta_i \Downarrow w(K_2 \cup K_1 - \{i\}) - w(K_2 \cup K_1)$
        $\Delta_i \Downarrow f_i - \delta_i$
        Se $\Delta_i \leq 0$ ***então***
        Inicio
            $K_1 \Downarrow K_1 \cup \{i\}$
            $K_2 \Downarrow K_2 - \{i\}$
            Parada $\Downarrow$ falso
        Fim
    Fim
    Se K_1 viável e Parada = falso ***então***
    Inicio
        Parada $\Downarrow$ verdadeiro
        calcular $w(K_1)$
        para cada $i \in K_2$ ***fazer*** *{laço 3 }*
        Inicio
            calcular $w(K_1 \cup \{i\})$
            $\delta_i \Downarrow w(K_1) - w(K_2 \cup \{i\})$
            $\Delta_i \Downarrow f_i - \delta_i$
            Se $\Delta_i \geq 0$ ***então***
            Inicio
                $K_0 \Downarrow K_0 \cup \{i\}$
                $K_2 \Downarrow K_2 - \{i\}$
                Parada $\Downarrow$ falso
            Fim
        Fim
    Fim
Fim
----- Fim Algoritmo 1 -----

O laço 3 representa a realização do *C-Teste* $\forall i \in K_2$, o laço 2 representa a realização do *O-Teste* $\forall i \in K_2$ e o laço 1 representa a execução do algoritmo como um todo, que terminará quando $K_2 = \varnothing$ ou quando for verificado que os testes não conseguem fechar o *gap*. Seja n o número de elementos em I e $O(w)$ a complexidade do algoritmo para resolver o Problema de Transporte. Como a cada iteração do laço 1 pelo menos uma facilidade é retirada de K_2, teremos, no máximo, n iterações do laço 1. Por

outro lado, dentro do laço 1, os laços 2 e 3 implicam, no pior caso, em resolver $2n$ problemas de transporte. Logo, a complexidade do Algoritmo 1 é $O(n^2O(w))$.

2.2 Conclusões

As aplicações dos testes de redução, *O-Teste* e *C-Teste* não garantem que se obtenha a solução ótima do *PLC*. Freqüentemente não é possível fechar o *gap* de indefinição. Isso acontece por causa da distribuição de clientes a facilidades. Cada cliente procura ser atendido pela facilidade mais interessante, ou seja, a facilidade que forneça o custo variável menor. Com isso, a indefinição do *O-Teste* costuma ocorrer para problemas em que $K_1 \cup K_2$ é muito grande, isto é, quando K_0 é pequeno. O problema inverso ocorre com o *C-Teste*. Neste caso, um K_1 muito pequeno provoca uma sensibilidade muito grande do custo de transporte, quando é inserida uma facilidade, dificultando o fechamento desta. Embora os testes não garantam a otimalidade, reduzem o tamanho do problema.

Foi observado que a decisão relativa à abertura/fechamento de uma facilidade, mesmo que temporária, aumenta o potencial dos testes com relação a outras facilidades, pois, caso uma facilidade seja fechada, o conjunto $K_1 \cup K_2$ ficará menor, aumentando assim as possibilidades de sucesso do *O-Teste*, pois quando o *O-Teste* for realizado, a retirada de uma facilidade será melhor sentida, ou seja, a falta da facilidade será mais representativa. Da mesma forma, caso uma facilidade seja aberta, aumentará o conjunto K_1, aumentando juntamente as possibilidades de sucesso do *C-Teste*. Esta característica favorece o uso dos testes em conexão com um algoritmo *Branch and Bound*, que será visto no Capítulo 4. Esta mesma observação também revela que o sucesso de um teste propicia o sucesso do outro, formando assim um ciclo virtuoso.

Pode-se dizer que o Algoritmo 1, apesar de verificar o caso $K_2 = \varnothing$, não o permite fazer *a priori*, ou seja, é preciso aplicar o algoritmo para saber se $K_2 = \varnothing$, pois ele não leva em conta os dados do problema diretamente.

No Capítulo 3 serão apresentados, em conjunto com outros testes de redução, alguns resultados computacionais do *O-Teste* e do *C-Teste*, na aplicação do Algoritmo 1.

Capítulo 3

3 Limites Inferiores e Testes de Redução

Quando se opta por encontrar a solução ótima de um problema combinatório, pertencente à classe de problemas NP (não polinomial), uma das primeiras coisas que vem em mente é que deve ser desenvolvido um algoritmo que funcione na prática. Isso porque, na teoria, sua complexidade é exponencial. Para que o algoritmo funcione na prática é preciso que ele não explore todas as soluções do espaço de busca, ou seja, todas as soluções das combinações existentes. Nesse capítulo serão abordadas duas formas para se tentar eliminar partes do espaço de busca. Uma será através de limites inferiores, a outra, através de testes de redução, como os testes apresentados no Capítulo 2, só que esses baseados nos resultados dos limites inferiores obtidos.

Tanto os limites inferiores quanto os testes de redução serão aplicados em um algoritmo *branch and bound*, que será apresentado no Capítulo 4. Infelizmente, não será possível apresentar todas as idéias, deste capítulo, sem as vincular com o método *branch and bound*. Assim sendo, será apresentado idéias gerais do método desenvolvido no Capítulo 4, para melhor compreensão do desenvolvimento que será apresentado na seqüência.

O algoritmo, apresentado Capítulo 4, utiliza um processo enumerativo. Esse processo pode ser representado através de uma árvore, denominada de árvore de busca, onde cada nó representa um item da enumeração. A enumeração será realizada através dos conjuntos K_0, K_1 e K_2, que representam respectivamente o conjunto de facilidades fechadas, abertas e indefinidas. Dessa forma, o processo de enumeração é realizado retirando uma facilidade $i \in K_2$ e a inserindo hora em K_0 e hora K_1. A árvore terá seu nó raiz formada pelos conjuntos $K_0 = K_1 = \varnothing$ e $K_2 = I$. A partir de então se inicia o processo de enumeração. A Figura 4 mostra os três primeiros nós da árvore.

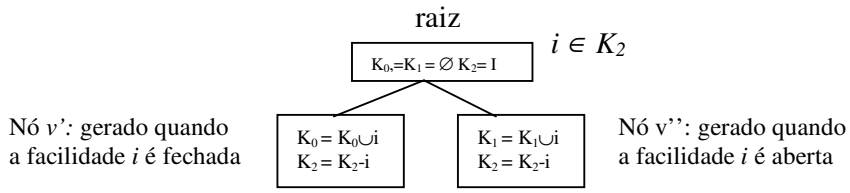


Figura 4 – Formato da Enumeração

Por abuso de linguagem um nó será denominado um problema de localização capacitado, denominado de $PLC_{k0,k1,k2}$, onde, com exceção do nó raiz, algumas facilidades são fixadas como abertas e ou outras como fechadas. Uma solução para o *PLC* somente poderá ser alcançada em um nó folha, isto é, quando todas as facilidades tiverem seus valores definidos. Além deste critério é necessário que a soma das capacidades das facilidades pertencentes a K_1 seja maior do que a soma da demanda, ou seja, que o problema associado ao nó folha seja viável. Caso sejam geradas todas as folhas ter-se-ão todas a soluções possíveis para o *PLC* e, naturalmente, a menor é a solução ótima. É justamente a exploração de todas as folhas que se deseja evitar, pois o número de folhas é exponencial em função do número de facilidades. Nesse capítulo, serão desenvolvidas algumas técnicas para que seja possível a eliminação de partes da árvore de busca, não explorando, com isso, todas as soluções possíveis.

Os limites inferiores serão aplicados sob os nós da árvore de busca, ou seja, em um $PLC_{K0,K1,K2}$. Quando for utilizado limite inferior para a redução do espaço de busca, e o mesmo tiver sucesso, é dito que foi realizado um corte (ou poda). O limite inferior é aplicado da seguinte forma: seja *LS* o valor de uma solução viável do *PLC* e $LI_{K0,K1,k2}$ o valor do limite inferior do $PLC_{K0,K1,K2}$. Se $LI_{K0,K1,k2} \geq$ LS então qualquer definição dada às facilidades de K_2 fornece soluções maiores ou iguais à solução já encontrada. Isso quer dizer que não é necessário explorar as possíveis combinações que podem ser obtidas através da abertura ou fechamento das facilidades pertencentes a K_2, ou seja, é realizado um corte.

Quando um teste de redução tiver sucesso é porque a situação de uma facilidade foi definida, conforme visto no Capítulo 2, em zero ou em um. Isto é, a facilidade foi

colocada no conjunto K_0 ou K_1. Nesse caso, também foi reduzido o espaço de busca. Exemplificando, suponha que a facilidade $i \in K_2$ tenha sido aberta pelo teste de redução, isso significa que $y_i=1$ e não é mais necessário explorar as combinações formadas com $y_i=0$. Assim, somente é necessário descer pelo lado direito da árvore de busca, eliminando todo o lado esquerdo.

3.1 Limite Inferior Utilizando Relaxação da Restrição Inteira

O *PLC* pode ser formulado como:

(3-1) $$\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i$$

S.a.

(3-2) $$\sum_{j \in J} x_{ij} \leq a_i y_i, \quad \forall i \in I$$

(3-3) $$\sum_{i \in I} x_{ij} = b_j, \quad \forall j \in J$$

(3-4) $$x_{ij} \geq 0, \quad \forall i \in I, \forall j \in J$$

(3-5) $$y_i \geq 0, \quad \forall i \in I$$

(3-6) $$y_i \leq 1, \quad \forall i \in I$$

(3-7) $$y_i \text{ int}, \quad \forall i \in I$$

Se as restrições (3-7) forem relaxadas completamente, ou seja, eliminada, tem-se um limite inferior para o problema. O problema formado com a eliminação das restrições (3-7) será denominado de *PLCRI* (Problema de **L**ocalização **C**apacitado com a **R**elaxação da restrição **I**nteira) e é representado por (3-1) a (3-6).

A resolução do *PLCRI* pode fornecer dois resultados. O primeiro é um limite inferior para o *PLC*. O segundo, que é obtido somente em alguns casos, é a própria solução do *PLC*. Esse caso acontece quando $\forall i \in I \; y_i = 0$ ou $y_i = 1$. Isto é, embora as restrições (3-7) tenham sido relaxadas elas acabaram sendo atendidas. Assim sendo, com a resolução do *PLCRI* tem-se a solução do *PLC*. O *PLC* associado aos K_0, K_1 e K_2, denominado de $PLC_{K0,K1,K2}$, é definido como:

(3-8) $$\min \sum_{i \in K1 \cup K2} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in K2} f_i y_i + \sum_{i \in K1} f_i$$

S.a.

(3-9) $$\sum_{j \in J} x_{ij} \leq a_i y_i, \quad \forall i \in K_2$$

(3-10) $$\sum_{j \in J} x_{ij} \leq a_i, \quad \forall i \in K_1$$

(3-11) $$\sum_{i \in K1 \cup K2} x_{ij} = b_j, \quad \forall j \in J$$

(3-12) $$x_{ij} \geq 0, \quad \forall i \in K_1 \cup K_2, \forall j \in J$$

(3-13) $$y_i \geq 0, \quad \forall i \in K_2$$

(3-14) $$y_i \leq 1, \quad \forall i \in K_2$$

(3-15) $$y_i \text{ int}, \quad \forall i \in K_2$$

Se as restrições (3-15) forem relaxadas completamente, ou seja, eliminadas, tem-se um limite inferior para o $PLC_{K0,K1,K2}$. O problema formado com a eliminação da restrição (3-15) será denominado de $PLCRI_{K0,K1,K2}$ (Problema de **L**ocalização **C**apacitado de um nó *v*, da árvore de busca, com a **R**elaxação da restrição **I**nteira) .

Teorema 1: Seja x_{ij}^* e y_i^* $\forall i \in I$ e $\forall j \in J$ uma solução ótima de *PLCRI* então

$$\sum_{j \in J} x_{ij}^* = a_i y_i^*, \text{para } a_i, f_i > 0, \quad \forall i \in I$$

Antes da prova ser iniciada, vale lembrar que o *PLCRI* é representado pelas restrições de (3-1) a (3-6).

Prova: Pelas restrições (3-2) sabe-se que $\sum_{j \in J} x_{ij}^*$ não pode ser maior do que $a_i y_i^*$ $\forall i \in I$. Logo é necessário provar que não temos:

(3-16) $$\sum_{j \in J} x_{ij}^* < a_i y_i^*, \quad \forall i \in I$$

Isso será provado por contradição. Suponha que exista *k* tal que a condição (3-16) seja satisfeita, então $\sum_{j \in J} x_{kj}^* < a_k y_k^*$. Com isso, pode-se fazer $y_k' = y_k^* - \Delta$, onde $\Delta > 0$, tal que:

(3-17) $$\sum_{j \in J} x_{kj}^{*} = a_k \, y_k'$$

Vale observar que $a_k > 0$ ($a_k = 0$ é um caso absurdo que pode ser descartado). A condição (3-17) continua satisfazendo a restrição (3-2). Como $\sum_{j \in J} x_{kj}^{*} \geq 0$ e $a_k \geq 0$, tem-se que $y_k' \geq 0$, satisfazendo também a restrição (3-5). Como $y_k' < y_k^{*}$ e y_k^{*} por hipótese é menor ou igual a *1*, isto é, $y_k^{*} \leq 1$, a condição satisfaz a restrição (3-6). Então se substituirmos y_k^{*} por y_k' a solução continua viável. Olhando para função objetivo e sabendo-se que $y_k' < y_k^{*}$, conclui-se que a solução dada por y_k' é menor do que a fornecida por y_k^{*} para $f_k > 0$. Logo se chega a uma contradição pois y_k^{*} era a solução ótima. •

Pelo Teorema 1 pode-se fazer:

(3-18) $$y_i^{*} = \frac{\sum_{j \in J} x_{ij}^{*}}{a_i} \quad \forall i \in I$$

Substituindo a equação (3-18) na equação (3-1) tem-se

(3-19) $$(PLCRI) \min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}^{*} + \sum_{i \in I} f_i \frac{\sum_{j \in J} x_{ij}^{*}}{a_i}$$

Simplificando:

(3-20) $$(PLCRI) \min \sum_{i \in I} \sum_{j \in J} \frac{f_i}{a_i} c_{ij} x_{ij}^{*}$$

Fazendo $C_{ij} = \frac{f_i}{a_i} c_{ij}$ tem-se:

(3-21) $$\min \sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij}^{*}$$

Como $y_i^{*} \leq 1$ a restrição (3-2) pode ser representa como:

(3-22) $$\sum_{j \in J} x_{ij}^{*} \leq a_i, \quad \forall i \in I$$

Considerando a função (3-21) e as equações (3-22) pode-se concluir que foi possível eliminar as variáveis $y_i \ \forall \ i \in \ I$ do *PLCRI*. Com isso o *PLCRI* pode ser formulado da seguinte maneira

(3-23) $$\min \sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij}$$

S.a.

(3-24) $$\sum_{j \in J} x_{ij} \leq a_i, \quad \forall i \in I$$

(3-25) $$\sum_{i \in I} x_{ij} = b_j, \quad \forall j \in J$$

(3-26) $$x_{ij} \geq 0, \quad \forall i \in I, \forall j \in J$$

Deste modo o *PLCRI* foi transformado em um Problema de Transporte Capacitado (*PTC*), ver seção 6.1. Para resolver o problema basta aplicar o método desenvolvido no Anexo 1. Após o problema ser resolvido, para se obter o valor das variáveis $y_i \ \forall \ i \in \ I$ basta aplicar as equações (3-18).

O mesmo raciocino de transformação do *PLCRI* em um problema de transporte pode ser aplicado ao $PLCRI_{K0,K1,K2}$. O Teorema 1 é aplicado na função objetivo (3-8). Assim o $PLCRI_{K0,K1,K2}$ pode ser formulado como:

(3-27) $$\min \sum_{i \in K2} \sum_{j \in J} C_{ij} x_{ij} \sum_{i \in K1} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in K1} f_i$$

S.a.

(3-28) $$\sum_{j \in J} x_{ij} \leq a_i, \quad \forall i \in K_1 \cup K_2$$

(3-29) $$\sum_{i \in K_1 \cup K_2} x_{ij} = b_j, \quad \forall j \in J$$

(3-30) $$x_{ij} \geq 0, \quad \forall i \in K_1 \cup K_2, \forall j \in J$$

Seja *LS* um limite superior para o *PLC*. Então se $PLCRI_{K0,K1,K2} \geq LS$ ocorre um corte. Isto é, não é necessário explorar as soluções formadas com as possíveis fixações de valores as facilidades pertencentes a K_2, pois todas serão maiores ou iguais a uma solução já existente, isso porque $PLCRI_{K0,K1,K2}$ é um limite inferior do $PLC_{K0,K1,K2}$.

Por outro lado, se os valores de $y_i \ \forall \ i \in K_2$ forem *0* ou *1* após a resolução do $PLCRI_{K0,K1,K2}$, não é necessário explorar as soluções formadas com as possíveis fixações de valores as facilidades pertencentes a K_2, pois os valores fixados a elas, quando da resolução do $PLCRI_{K0,K1,K2}$, são ótimos, isso porque a restrição (3-15) foi atendida, fazendo $PLCRI_{K0,K1,K2} = PLC_{K0,K1,K2}$.

O cálculo do $PLCRI_{K0,K1,K2}$ também ajudará na orientação do processo heurístico de exploração da árvore para resolução do *PLC*. No Capítulo 4 serão mostradas as heurística utilizadas para a exploração da árvore e onde se encaixam os resultados obtidos pelo $PLCRI_{K0,K1,K2}$.

3.2 Testes de redução utilizando $PLCRI_{K0,K1,K2}$

Da mesma forma que no Capítulo 2, será utilizado testes de redução para tentar diminuir a dimensão do problema e em alguns casos até obter a solução ótima. Os testes são baseados no $PLCRI_{K0,K1,K2}$.

Suponha calculado o $PLCRI_{K0,K1,K2}$ em um nó *v* da árvore de busca, tendo sido obtida uma solução (x^*,y^*), e suponha que já se tenha um limite superior *LS* para o *PLC*. Caso $PLCRI_{K0,K1,K2} \geq LS$ ocorre um corte, pois todos os nós descendentes de *v* terão como solução valores maiores ou iguais ao *LS*, já que o $PLCRI_{K0,K1,K2}$ é um limite inferior. Caso no entanto, $PLCRI_{K0,K1,K2} < LS$, será feito um estudo para as facilidades em K_2, estudando as alterações no valor do limite inferior caso estas variáveis sejam abertas ou fechadas. Para a realização desse estudo será definido o seguinte problema:

- $PLCRI_{K0,K1,K2}(y_k=1 \text{ ou } 0)$: relaxação inteira do *PLC* para um dado K_0, K_1 e K_2 abrindo/fechando adicionalmente uma facilidade $k \in K_2$.

Suponha uma facilidade *k*, tal que, na solução do $PLCRI_{K0,K1,K2}$ tenha sido obtido o valor $y_k = 0,3$, ou seja, existe uma "tendência" a fechar esta facilidade. Caso seja fixado $y_k = 1$, ou seja, forçado a abertura desta facilidade e recalculado novamente o problema relaxado, seja $PLCRI_{K0,K1,K2}(y_k=1)$ o novo valor obtido. Se $PLCRI_{K0,K1,K2}(y_k=1)$ agora for maior ou igual à *LS* isto é uma comprovação que para os valores de K_0, K_1 e K_2, dados no início, a facilidade *k* não pode ser aberta. Portanto, é possível fazer $K_0 = K_0 \cup \{k\}$ e $K_2 = K_2 - \{k\}$. Para evitar que este procedimento não seja

efetivo ele só será aplicado para $LS - VA \leq PLCRI_{K0,K1,K2} < LS$, onde *VA* é um valor médio de acréscimo, ao abrir facilidades, obtido em um dos nós anteriores da árvore.

Seja y^* a solução ótima dada pelo cálculo do $PLCRI_{K0,K1,K2}$, então se $y_k^* = 1$ o $PLCRI_{K0,K1,K2} = PLCRI_{K0,K1,K2}(y_k=1)$. Nesse caso, o $PLCRI_{K0,K1,K2}$ indica que a facilidade *k* tem "tendência" de ser aberta, de nada valerá o cálculo de $PLCRI_{K0,K1,K2}-(y_k=1)$. Para que o cálculo do $PLCRI_{K0,K1,K2}(y_k=1)$ seja mais efetivo, será considerado a tendência de uma facilidade ser aberta os valores $y_k^* \geq 0,5$. Assim sendo, o $PLCRI_{K0,K1,K2}(y_k=1)$ só poderá ser calculado quando $y_k^* < 0,5$ e $LS - VA \leq PLCRI_{K0,K1,K2} < LS$. A partir do cálculo do $PLCRI_{K0,K1,K2}(y_k=1)$ o valor de *VA* pode ser atualizado. Seja um nó *v* onde se tem K_0, K_1 e K_2, a atualização do valor médio de acréscimo (*VA*), ao abrir facilidades, é realizado como:

(3-31) $$VA = \frac{\sum_{k \in K_2 e y^*_k < 0,5} PLCRI_{K_0,K_1.K_2}(y_k = 1) - PLCRI_{K_0,K_1.K_2}}{|\{k \in K_2 \mid y^*_k < 0,5\}|}$$

O mesmo tipo de procedimento aplica-se para uma facilidade *k* onde $y_k^* \geq 0,5$, ou seja, trata-se de uma facilidade com uma tendência a ser aberta. Se for forçado o fechamento desta facilidade e recalculado novamente o problema relaxado, seja $PLCRI_{K0,K1,K2}(y_k=0)$ o novo valor obtido. Se $PLCRI_{K0,K1,K2}(y_k=0)$ agora for maior ou igual à *LS* isto é uma comprovação que para os valores de K_0, K_1 e K_2, dados no início, a facilidade *k* não pode ser fechada. Portanto, é possível fazer $K_1 = K_1 \cup \{k\}$ e $K_2 = K_2 - \{k\}$. Para evitar que este procedimento não seja efetivo ele só será aplicado para $LS - VF \leq PLCRI_{K0,K1,K2} < LS$, onde *VF* é um valor médio de acréscimo, ao fechar facilidades, obtido em um dos nós anteriores da árvore. Com isso, o $PLCRI_{K0,K1,K2}(y_k=0)$ só poderá ser calculado quando $y_k^* \geq 0,5$ e $LS - VF \leq PLCRI_{K0,K1,K2} < LS$. A partir do cálculo do $PLCRI_{K0,K1,K2}(y_k=0)$ o valor de *VF* pode ser atualizado. Seja um nó *v* onde se tem K_0, K_1 e K_2, a atualização do valor médio de acréscimo (*VF*), ao fechar facilidades, é realizado como:

(3-32) $$VF = \frac{\sum_{k \in K_2 e y^*_k \geq 0,5} PLCRI_{K_0,K_1.K_2}(y_k = 0) - PLCRI_{K_0,K_1.K_2}}{|\{k \in K_2 \mid y^*_k \geq 0,5\}|}$$

Seja *LS* um limite superior para o *PLC*, então o Algoritmo 2 sintetiza a realização dos testes de redução que utilizam como base o $PLCRI_{K0,K1,K2}$. O passo 1 é referente ao nó raiz da árvore. Nele os valores de *VA* e *VF* são inicializados com o valor infinito. Esta inicialização possibilita que os passos 5 e 7 (realização dos testes de redução) sejam executados no nó raiz, fazendo com que valores de *VA* e *VF* sejam inicializados com dados do problema. O passo 3 verifica a realização do corte através do limite inferior $PLCRI_{K0,K1,K2}$ e os passos 4 e 6 verificam a possibilidade de aplicação dos testes, os quais são realizados nos passos 5 e 7.

Algoritmo 2: aplicação dos testes de redução baseados em $PLCRI_{K0,K1,K2}$
Passo 1: *{inicialização}*

$K_0 = K_1 = \varnothing;$ $K_2 = I;$ $VA = VF = \infty;$

Passo 2:

Calcule $PLCRI_{K0,K1,K2}$. Seja (x*,y*) a solução correspondente.

Passo 3:

Se $PLCRI_{K0,K1,K2} \geq LS$ ***então*** *backtracking*

Senão ir para Passo 4

Passo 4:

Se $PLCRI_{K0,K1,K2} < LS - VA$ ***então*** ir para Passo 6

Senão ir para Passo 5

Passo 5: *{ teste de fechar }*

Para todo $k \in K_2 \mid y_k < 0,5$ ***fazer***

Calcule $PLCRI_{K0,K1,K2}(y_k=1)$

Se $PLCRI_{K0,K1,K2}(y_k=1) \geq LS$ ***então***

Fazer $K_0 = K_0 \cup \{k\}$ e $K_2 = K_2 - \{k\}$

Atualizar VA conforme (3-31)

Passo 6:

Se $PLCRI_{K0,K1,K2} < LF - VA$ ***então*** parar algoritmo

Senão ir para Passo 7

Passo 7: *{ teste de abrir }*

Para todo $k \in K_2 \mid y_k \geq 0,5$ ***fazer***

Calcule $PLCRI_{K0,K1,K2}(y_k=0)$

Se $PLCRI_{K0,K1,K2}(y_k=0) \geq LS$ ***então***

Fazer $K_1 = K_1 \cup \{k\}$ e $K_2 = K_2 - \{k\}$

Atualizar VF conforme (3-32)

----- Fim Algoritmo 2 -----

Para facilitar a referência no restante do texto, o procedimento que faz $K_1 = K_1 \cup \{k\}$ e $K_2 = K_2 - \{k\}$ será denominado de *O-RITeste* e o procedimento que faz $K_0 = K_0 \cup \{k\}$ e $K_2 = K_2 - \{k\}$ de *C-RITeste*.

3.3 Limite Inferior Utilizando Relaxação Lagrangeana

No método desenvolvido para resolução do *PLC*, conforme será visto no Capítulo 4, utiliza-se, além do limite inferior apresentado na seção 3.1, um limite inferior baseado em Relaxação Lagrangeana. A base desse método é exposta em Beasley [12]. Uma das diferenças é que não será considerada a restrição que fixa um número de facilidades a serem abertas, mas sim um intervalo, onde é especificado o número mínimo e máximo de facilidades a serem abertas, representadas respectivamente por *f_min* e *f_max*. Outra diferença é que o método foi adaptado para ser aplicado em um *Branch and Bound*. A formulação do *PLC* é um pouco diferente da trabalhada até agora. Considera-se c_{ij}, $\forall i \in I$ e $\forall j \in J$, como sendo o custo da facilidade i atender toda a demanda do cliente j. Com isso x_{ij} passa ser a fração de demanda do cliente j que será atendida pela facilidade i. Assim sendo, $0 \leq x_{ij} \leq 1$. As outras variáveis tem o mesmo significado da formulação dada no Capítulo 1. O modelo matemático é o seguinte:

(3-33) $\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i$

S.a.

(3-34) $\sum_{j \in J} b_j x_{ij} \leq a_i y_i, \quad \forall i \in I$

(3-35) $\sum_{i \in I} x_{ij} = 1, \quad \forall j \in J$

(3-36) $x_{ij} \leq 1, \quad \forall i \in I, \forall j \in J$

(3-37) $x_{ij} \geq 0, \quad \forall i \in I, \forall j \in J$

(3-38) $x_{ij} \leq y_i, \quad \forall i \in I, \forall j \in J$

(3-39) $\sum_{i \in I} y_i \geq f\_\min$

(3-40) $\sum_{i \in I} y_i \leq f\_\max$

(3-41) $y_i \in \{0,1\}, \quad \forall i \in I$

As restrições (3-34) asseguram que nenhum cliente seja atendido por uma facilidade fechada e que o total de demanda atendida pela facilidade não ultrapasse a sua capacidade. A restrições (3-35) asseguram que a demanda de cada cliente seja satisfeita. As restrições (3-36) e (3-37) proporcionam os limites de alocação para as variáveis x_{ij}, enquanto que as (3-41) representam as variáveis inteiras, onde $y_i = 1$ se a facilidade i for aberta e $y_i = 0$ caso contrário. As restrições (3-39) e (3-40) especificam a quantidade máxima e mínima de facilidades abertas.

Para obtenção do limite inferior realiza-se a relaxação Lagrangeana nas equações (3-34) e (3-35) com os multiplicadores t_i, $i \in I$ para (3-34) e os multiplicadores s_j, $j \in J$ para (3-35). O *PLC* com relaxação Lagrangeana (*PLCRL*) fica:

(3-42) $\min \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i + \sum_{j\in J} s_j(1-\sum_{i\in I} x_{ij}) + \sum_{i\in I} t_i(-y_i + \sum_{j\in J}\frac{b_j}{a_i}x_{ij})$

S.a.

(3-43) $x_{ij} \le 1, \quad \forall i \in I, \forall j \in J$

(3-44) $x_{ij} \ge 0, \quad \forall i \in I, \forall j \in J$

(3-45) $x_{ij} \le y_i, \quad \forall i \in I, \forall j \in J$

(3-46) $\sum_{i\in I} y_i \ge f\_\min$

(3-47) $\sum_{i\in I} y_i \le f\_\max$

(3-48) $t_i \ge 0, \quad \forall i \in I$

(3-49) $y_i \in \{0,1\}, \quad \forall i \in I$

Definindo

(3-50) $C_{ij} = c_{ij} - s_j + t_i(b_j/a_i)$

o problema torna-se:

(3-51) $(PLCRL) \min \sum_{i\in I}\sum_{j\in J} C_{ij}x_{ij} + \sum_{i\in I}(f_i - t_i)y_i + \sum_{j\in J} s_j$

S.a.

(3-43), **(3-44)**, **(3-45)**, **(3-46)**, **(3-47)**, **(3-48)** e **(3-49)**

O problema é facilmente resolvido porque se $y_i = 1$ (isto é a facilidade *i* é aberta) a contribuição para a função objetiva Lagrangeana é dada por:

(3-52) $p_i = f_i - t_i + \sum_{j\in J} \min(0, C_{ij})$

O *PLC* com relaxação Lagrangeana reduz-se para:

(3-53) $(PLCRL) \min \sum_{i\in I} p_i y_i + \sum_{j\in J} s_j$

S.a

(3-54) $\sum_{i\in I} y_i \ge f\_\min$

(3-55) $\sum_{i\in I} y_i \le f\_\max$

(3-56) $y_i \in \{0,1\}, \quad \forall i \in I$

Para resolver o problema *PLCRL* são necessários os passos do Algoritmo 3 .

Algoritmo 3: resolver *PLCRL*

1. Ordenar os valores de p_i em ordem crescente.
2. Tomar os *f_min* menores valores de p_i e fazer $y_i = 1$.
3. Para os p_i que estão ordenados entre $f\_min + 1$ e *f_max* fazer $y_i = 1$ se $p_i < 0$, e $y_i = 0$ caso contrário.
4. Para os p_i não considerados, ou seja, os p_i que estão em posições, na ordenação, maiores do que *f_max*, fazer $y_i = 0$.
5. Denota-se a solução por $\hat{y}$. Os valores correspondentes para as variáveis x_{ij} são dados por $\hat{x}_{ij} = 1$ se $\hat{y}_i = 1$ e $C_{ij} \leq 0$ e $\hat{x}_{ij} = 0$ caso contrário.

----- Fim Algoritmo 3-----

Para exemplificar os passos de 1 a 4 será suposto que o $f\_min = 2, f\_max = 5$ e que os valores de p_i sejam os representados na Tabela 10.

p_1=10	p_2=-3	p_3=-9	p_4=8	p_5=-2	p_6=-7	p_7=15

Tabela 10 – Valores de p_i

Passo 1: Após o a aplicação do passo 1 a Tabela 11 mostra como ficou a ordenação.

p_3=-9	p_6=-7	p_2=-3	p_5=-2	P_4=8	p_1=10	p_7=15

Tabela 11 – ordenação de p_i

Passo 2: Após a aplicação do passo 2 as facilidades 3 e 6 são abertas.

Passo 3: Após a aplicação do passo 3 as facilidades 2 e 5 são abertas e a facilidade 4 é fechada.

Passo 4: Após a aplicação do passo 4 as facilidades 1 e 7 são fechadas.

Como o limite inferior será utilizado na árvore de busca, juntamente com os conjuntos K_o, K_1 e K_2, o cálculo do limite inferior de um determinado nó *v* é denominado de $PLCRL_{K0,K1,K2}$. Seja $p_i = f_i - t_i + \sum_{j \in J} \min(0, C_{ij})$, onde $C_{ij} = c_{ij} - s_j + t_i(b_j/a_i)$. Seja, também, $f\_min_{K0,K1,K2}$ o número mínimo de facilidades abertas para o nó *v* e $f\_max_{K0,K1,K2}$ o número máximo de facilidades abertas para o nó *v*, então o $PLCRL_{K0,K1,K2}$ é obtido da seguinte forma:

(3-57) $$\min \sum_{i \in K_2} p_i y_i + \sum_{j \in J} s_j + \sum_{i \in K_1} p_i$$

S.a

(3-58) $$\sum_{i \in K1 \cup K2} y_i \geq f\_min_{K_0, K_1, K_2}$$

(3-59) $$\sum_{i \in K1 \cup K2} y_i \leq f\_\max_{K_0, K_1, K_2}$$

(3-60) $$y_i \in \{0,1\}, \quad \forall i \in K_2$$

(3-61) $$y_i = 1, \quad \forall i \in K_1$$

O valor do limite inferior é dado por:

(3-62) $$LI = \sum_{i \in K_2} p_i \hat{y}_i + \sum_{j \in J} s_j + \sum_{i \in K_1} p_i$$

O cálculo do $PLCRL_{K0,K1,K2}$ fornece $\hat{x}_{ij}$, $\forall i \in I$ e $\forall j \in J$ e $\hat{y}_i$, $\forall i \in I$. Sinteticamente serão denominados ($\hat{x}$, $\hat{y}$) a solução do $PLCRL_{K0,K1,K2}$ onde $\hat{x}$ é a matriz $\hat{x}_{ij}$ e $\hat{y}$ é o vetor $\hat{y}_i$. O Algoritmo 4 resolve o $PLCRL_{K0,K1,K2}$.

Algoritmo 4: resolver $PLCRL_{K0,K1,K2}$

1. $\forall i \in K_o$ fazer $y_i = 0$.
2. $\forall i \in K_1$ fazer $y_i = 1$.
3. Ordenar os valores de p_i, $\forall i \in K_2$, em ordem crescente.
4. Tomar os $f\_min_{K0,K1,K2} - |K_1|$ menores valores de p_i , $i \in K_2$, e fazer $y_i = 1$.
5. Para os p_i , $i \in K_2$, que estão ordenados entre $f\_min_{K0,K1,K2} - |K_1| + 1$ e $f\_max_{K0,K1,K2}$ fazer $y_i = 1$ se $p_i < 0$, e $y_i = 0$ caso contrário.
6. Para os p_i , $i \in K_2$, não considerados, ou seja, os p_i que estão em posições, na ordenação, maiores do que $f\_max_{K0,K1,K2}$, fazer $y_i = 0$.
7. Denota-se a solução por $\hat{y}$. Os valores correspondentes para as variáveis x_{ij} são dados por $\hat{x}_{ij} = 1$ se $\hat{y}_i = 1$ e $C_{ij} \leq 0$ e $\hat{x}_{ij} = 0$ caso contrário.

----- Fim Algoritmo 4-----

3.3.1 Número mínimo e máximo de facilidades utilizando *PLCRL*

Não são fornecidos, nos dados do problema, o número mínimo (*f_min*) e máximo (*f_max*) de facilidades abertas para o cálculo do *PLC*. O método do Capítulo 4 e o $PLCRL_{K0,K1,K2}$ utilizam esses parâmetros. Assim sendo, é necessário detectá-los.

Uma das formas é através do próprio $PLCRL_{K0,K1,K2}$. No Capítulo 4 serão fornecidas outras opções.

A expressão $\sum_{i \in K_2} p_i y_i + \sum_{j \in J} s_j + \sum_{i \in K_1} p_i$ representa o valor da função objetivo do $PLCRL_{K0,K1,K2}$. Sendo v o nó da árvore representado pelos conjuntos K_0, K_1 e K_2, $\hat{y}$ a solução ótima do $PLCRL_{K0,K1,K2}$ e LS um limite superior do PLC, então, se $\sum_{i \in K_2} p_i \hat{y}_i + \sum_{j \in J} s_j + \sum_{i \in K_1} p_i \geq LS$, ocorre um corte; caso contrário, poder-se-á tentar encontrar valores mais justos para f_min e f_max do nó v e seus descendentes. Esses valores serão denominados de $f\_min_{K0,K1,K2}$ e $f\_max_{K0,K1,K2}$ A idéia é abrir o menor número de facilidades que faz a função: $\sum_{i \in K_2} p_i y_i + \sum_{j \in J} s_j + \sum_{i \in K_1} p_i$, menor do que o LS. O modelo, a seguir, representa $f\_min_{K0,K1,K2.}$

(3-63) $\min \sum_{i \in K_1 \cup K_2} y_i$

$S.a$

(3-64) $\sum_{i \in K_2} p_i y_i + \sum_{j \in J} s_j + \sum_{i \in K_1} p_i < LS$

(3-65) $y_i = 1, \quad \forall i \in K_1$

(3-66) $y_i \in \{0,1\}, \quad \forall i \in K_2$

O modelo representa o menor número de facilidades, na resolução do $PLCRL_{K0,K1,K2}$, que faz com que o valor do $PLCRL_{K0,K1,K2}$ seja menor do que um limite superior encontrado. Isto é, o menor número de facilidades para que seja possível ter uma solução melhor do que a encontrada até então, no caso a representada por LS. O Algoritmo 5 resolve o modelo.

Algoritmo 5: encontrar $f\_min_{K0,K1,K2}$

1. $\forall i \in K_o$ fazer $y_i = 0$.
2. $\forall i \in K_1$ fazer $y_i = 1$.
3. Ordenar os valores de p_i, $\forall i \in K_2$, em ordem crescente.
4. Tomar os menores valores de p_i , $i \in K_2$, e fazer $y_i = 1$, até que a equação (3-64) seja satisfeita.
5. Para os p_i , $i \in K_2$, não considerados, fazer $y_i = 0$.
6. Seja $\hat{y}$ a solução gerada, então atualizar $f\_min_{K0,K1,K2}$ da seguinte forma:

$$\textbf{\textit{Se}} \sum_{i \in K_1 \cup K_2} \hat{y}_i > f\_\min_{K_0,K_1,K_2} \textbf{\textit{ então}} \; f\_\min_{K_0,K_1,K_2} \leftarrow \sum_{i \in K_1 \cup K_2} \hat{y}_i$$

----- Fim Algoritmo 5-----

Da mesma forma que foi possível tentar encontrar um número mais justo para $f\_min_{K0,K1,K2}$ (número mínimo de facilidades abertas em um nó *v*), será tentado determinar um número mais justo $f\_max_{K0,K1,K2}$ (número máximo de facilidades abertas em um nó *v*). A idéia é, após a abertura do número mínimo de facilidades, abrir o maior número de facilidades onde a função $\sum_{i \in K_2} p_i y_i + \sum_{j \in J} s_j + \sum_{i \in K_1} p_i$ continue menor do que o *LS*. O modelo, a seguir, representa $f\_max_{K0,K1,K2}$.

$$\textbf{(3-67)} \qquad \max \sum_{i \in K_1 \cup K_2} y_i$$

$$S.a$$

$$\textbf{(3-68)} \qquad \sum_{i \in K_2} p_i y_i + \sum_{j \in J} s_j + \sum_{i \in K_1} p_i < LS$$

$$\textbf{(3-69)} \qquad \sum_{i \in K_1 \cup K_2} y_i \geq f\_\min_{K_0,K_1,K_2}$$

$$\textbf{(3-70)} \qquad y_i = 1, \quad \forall i \in K_1$$

$$\textbf{(3-71)} \qquad y_i \in \{0,1\}, \quad \forall i \in K_2$$

O modelo representa o maior número de facilidades, na resolução do $PLCRL_{K0,K1,K2}$, que faz com que o valor do $PLCRL_{K0,K1,K2}$ seja menor do que o limite superior encontrado. Isto é, o maior número de facilidades que podem ser abertas para que seja possível ter uma solução melhor do que a encontrada até então, no caso a representada por *LS*. O Algoritmo 6 resolve o modelo.

Algoritmo 6: encontrar $f\_max_{K0,K1,K2}$

1. $\forall i \in K_o$ fazer $y_i = 0$.
2. $\forall i \in K_1$ fazer $y_i = 1$.
3. Ordenar os valores de p_i, $\forall i \in K_2$, em ordem crescente.
4. Tomar os menores valores de p_i , $i \in K_2$, e fazer $y_i = 1$, até que a equação (3-69) seja satisfeita.
5. Continuar tomando os menores valores de p_i , $i \in K_2$, ainda não abertos, e fazer $y_i = 1$, enquanto a restrição (3-68) estiver sendo satisfeita.
6. Para os p_i , $i \in K_2$, não considerados, fazer $y_i = 0$.
7. Seja $\hat{y}$ a solução gerada, então atualizar f_max do nó v da seguinte forma:

$$\textbf{\textit{Se}} \sum_{i \in K_1 \cup K_2} \hat{y}_i < f\_\max_{K_0,K_1,K_2} \textbf{\textit{então}} \; f\_\max_{K_0,K_1,K_2} \leftarrow \sum_{i \in K_1 \cup K_2} \hat{y}_i$$

----- Fim Algoritmo 6 -----

Exemplo 3: aplicação do Algoritmo 5 e Algoritmo 6

Para facilitar a compreensão será mostrado um exemplo de aplicação de cada algoritmo, onde é considerado $K_0 = K_1 = \varnothing$ e $K_2 = I$, ou seja, o nó raiz. Nesse caso, $f\_min_{K0,K1,K2} = 1$ e $f\_max_{K0,K1,K2} = |I|$. Será suposto que $\sum_{j \in J} s_j + \sum_{i \in K_1} p_i = 20$, $LS=9$ e os valores de p_i, $i \in K_2$, conforme Tabela 15.

p_1=10	P_2=-3	P_3=-7	p_4=8	p_5=2	p_6=-5	p_7=15

Tabela 12 – valores de p_i

Aplicação do Algoritmo 10

Inicialmente será mostrado a execução do Algoritmo 10. Como $K_0 = K_1 = \varnothing$ os passos 1 e 2 não realizam nada.

Passo 3: Após a aplicação do passo 3 a Tabela 13 mostra como ficou a ordenação.

p_3=-7	p_6=-5	p_2=-3	p_5=2	P_4=8	P_1=10	p_7=15

Tabela 13 – ordenação de p_i

Passo 4: Após a aplicação do passo 4 as facilidades 3 e 6 são abertas.

Passo 5: Após a aplicação do passo 5 as facilidades 2, 5, 4,1 e 7 são fechadas.

Passo 6: Após a aplicação do passo 6 $f\_min_{K0,K1,K2} = 2$.

Aplicação do Algoritmo 6

Agora será mostrado a execução do Algoritmo 6. Como $K_0 = K_1 = \varnothing$ os passos 1 e 2 não realizam nada.

Passo 3: Após a aplicação do passo 3 a Tabela 13 mostra como ficou a ordenação.

P_3=-7	p_6=-5	p_2=-3	p_5=2	P_4=8	P_1=10	p_7=15

Tabela 14 – ordenação de p_i

Passo 4: Após a aplicação do passo 4 as facilidades 3 e 6 são abertas.

Passo 5: Após a aplicação do passo 5 as facilidades 2 e 5 são abertas.

Passo 6: Após a aplicação do passo 6 as facilidades 4, 1 e 7 são fechadas.

Passo 7: Após a aplicação do passo 7 $f\_max_{K0,K1,K2} = 4$.

Com isso, após a aplicação do Algoritmo 10 e do Algoritmo 6, tem-se $f\_min_{K0,K1,K2} = 2$ e $f\_max_{K0,K1,K2} = 4$.

Supondo p_i ordenado crescentemente e $r_1 \leq r_2$, a Figura 5 mostra de maneira geral como é o comportamento da expressão $\sum_{i \in K_2} p_i y_i + \sum_{j \in J} s_j + \sum_{i \in K_1} p_i$ em relação ao *LS*, quando se vai abrindo facilidades pertencentes a K_2, na seqüência da ordenação de p_i. No início, da abertura das facilidades, o valor da expressão é maior ou igual ao *LS*, após um determinado número de facilidades abertas o valor da expressão passa a ser menor do que *LS*. Nesse ponto é identificado o número mínimo de facilidades abertas, representado na figura por **min**. Após a abertura de mais facilidades a expressão volta a ter um valor maior ou igual ao *LS*, nesse ponto é identificado o número máximo de facilidades abertas, representado na figura por **max**.

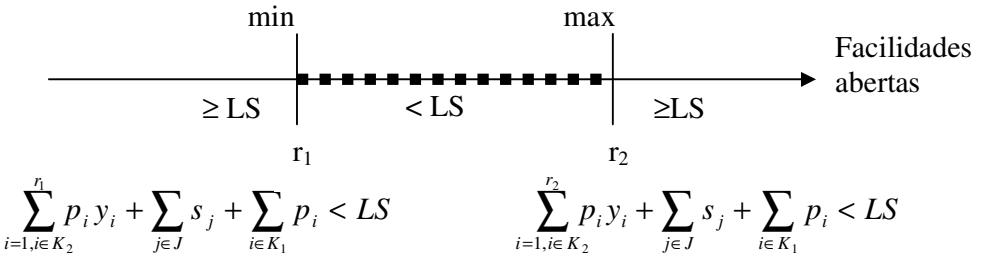


Figura 5 – Comportamento de $\sum_{i\in K_2} p_i y_i + \sum_{j\in J} s_j + \sum_{i\in K_1} p_i$

A Figura 6 mostra os valores da função f = $\sum_{i\in K_2} p_i y_i + \sum_{j\in J} s_j + \sum_{i\in K_1} p_i$, na medida em que se realiza a abertura das facilidades pertencentes a K_2, do Exemplo 3.

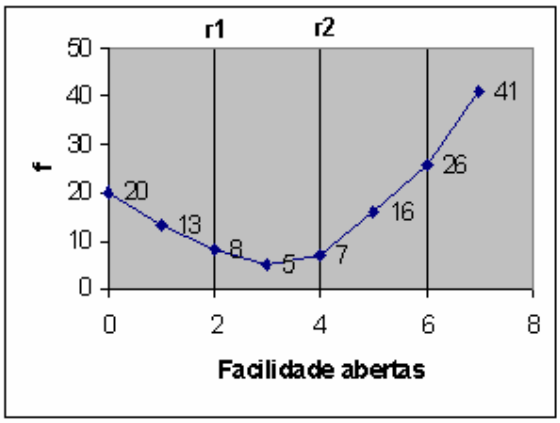


Figura 6 - valores de $\sum_{i\in K_2} p_i y_i + \sum_{j\in J} s_j + \sum_{i\in K_1} p_i$

Vale observar que $f\_min_{K0,K1,K2}$ e $f\_max_{K0,K1,K2}$ também ajudam na realização de cortes. Se $|K_1| = f\_max_{K0,K1,K2}$ então toda as facilidades pertencentes a K_2 serão fechadas, não necessitando, com isso, explorar as combinações que poderiam ser obtidas através da fixação (em zero ou um) das facilidades de K_2. Se $|K_1 \cup K_2| = f\_min_{K0,K1,K2}$ então toda as facilidades pertencentes a K_2 serão abertas, não necessitando, com isso, explorar as combinações que poderiam ser obtidas através da fixação (em zero ou um) das facilidades de K_2.

3.3.2 Cálculo do Subgradiente

A otimização com subgradientes é usada para tentar maximizar o limite inferior obtido pelo *PLCRL*. O subgradiente é calculado conforme Beasley [12]. Seja a solução do *PLCRL*, calculada anterior, dada por $\hat{x} \in \hat{y}$, para os quais temos os valores $\hat{t}$, $\hat{s}$, e seu valor ótimo *LI*, e seja ainda, *LS* um limite superior do *PLC* e $f \leq 2$ um escalar para ajuste do passo (veja Beasley [12]), então o procedimento para o cálculo do subgradiente adotado é o seguinte:

Algoritmo 7: otimização com subgradientes

1. **Calcular os subgradientes G_j e H_i usando:**

   $$G_j \leftarrow 1 - \sum_{i \in I} \hat{x}_{ij}, \forall j \in J$$

   $$H_i \leftarrow -\hat{y}_i + \sum_{j \in J} \frac{b_j}{a_i} \hat{x}_{ij}, \forall i \in I$$

2. **Ajustar H_i usando:**

   $H_i \Downarrow 0$ se $\hat{t}_i = 0$ e $H_i < 0, \forall i \in I$

3. **Definir o tamanho do passo T usando:**

   $$T \leftarrow \frac{f(LS - LI)}{\sum_{j \in J} (G_j)^2 + \sum_{i \in I} (H_i)^2}$$

4. **Atualizar os multiplicadores de Lagrange usando:**

   $s_j \Downarrow \hat{s}_j + TG_j, \forall j \in J$

   $t_i \Downarrow max(0, \hat{t}_i + TH_i), \forall i \in I$

----- Fim Algoritmo 7 -----

3.3.3 Algoritmo Geral para o Cálculo do Limite Inferior utilizando o $PLCRL_{K0,K1,K2}$ e Subgradientes

Será apresentado um algoritmo para o cálculo do $PLCRL_{K0,K1,K2}$, ou seja, um limite inferior para um nó v utilizando relaxação lagrangeana, isto é, resolver o problema representado por (3-57) a (3-60) e determinar valores para os multiplicadores s_j e t_i. Este procedimento utiliza o parâmetro f para a determinação do tamanho do passo (veja 3.3.2) bem como os parâmetros de parada u e N_{max}.

O algoritmo tem uma inicialização para o caso do nó raiz, onde não se tem valores para s_j e t_i. Para os demais nós estes valores já existem, podendo-se, portanto, calcular os subgradientes segundo o Algoritmo 7. Determinados os subgradientes o

algoritmo resolve o $PLCRL_{K0,K1,K2}$, conforme Algoritmo 4, atualizando se for o caso o valor do limite inferior $LI\_max_{K0,K1,K2}$ para o problema dado. Este procedimento é repetido de forma iterativa até que o tamanho do passo seja atendido ou o limite inferior gerado seja maior ou igual do que *LS* (limite superior para o *PLC*). No máximo a cada N_{max} iterações, sem o aumento de $LI\_max_{K0,K1,K2}$, é diminuído o tamanho do passo. Para a resolução do $PLCRL_{K0,K1,K2}$, é preciso de valores de $f\_min_{K0,K1,K2}$ e $f\_max_{K0,K1,K2}$. A partir de valores iniciais $f\_min_{K0,K1,K2} = 1$ e $f\_max_{K0,K1,K2} = |I|$ o Algoritmo 10 e o Algoritmo 6 (veja seção 3.3.1) permitem a determinação de valores mais justos (maiores para $f\_min_{K0,K1,K2}$ e menores para $f\_max_{K0,K1,K2}$). No Capítulo 4 será visto outras formas de determinação de $f\_min_{K0,K1,K2}$ e $f\_max_{K0,K1,K2}$.

O Algoritmo 8 calcula o limite inferior utilizando o $PLCRL_{K0,K1,K2}$ e otimizações com Subgradientes O melhor limite inferior encontrado é representado por $LI\_max_{K0,K1,K2}$, *N* representa o número de iterações de subgradiente que não aumentaram $LI\_max_{K0,K1,K2}$, e *LS* é um limite superior do *PLC*. Vale observar que a parada do algoritmo é dada quando $f > u$ ou $LI \geq LS$. O primeiro critério é fornecido para dar um limite de iterações ao algoritmo, já o segundo significa que o objetivo do limite inferior foi alcançado, ou seja, ocorrerá um corte na árvore de busca.

Algoritmo 8: Calcular LI por Otimização de Subgradientes e Relaxação Lagrangeana

Se *v* nó raiz ***então***
inicio

$t_i \Downarrow 0, \forall i \in I$ e $s_j \Downarrow min(c_{ij} | c_{ij} > 0, \forall i \in I$ e $\forall j \in J)$

Resolver $PLCRL_{K0,K1,K2}$ segundo Algoritmo 4
$LI\_max_{K0,K1,K2} \mid PLCRL_{K0,K1,K2}$

fim
$N \Downarrow 0$
Enquanto $f \leq u$ e $LI\_max_{K0,K1,K2} < LS$ ***fazer***
Inicio
Calcular o Subgradiente segundo Algoritmo 7
Resolver $PLCRL_{K0,K1,K2}$ segundo Algoritmo 4
Se $LI\_max_{K0,K1,K2} < PLCRL_{K0,K1,K2}$ ***então***
Inicio
$LI\_max_{K0,K1,K2} \Downarrow PLCRL_{K0,K1,K2}$
$N \Downarrow 0$
Atualizar $f\_min_{K0,K1,K2}$ e $f\_max_{K0,K1,K2}$ segundo o Algoritmo 5 e o Algoritmo 6
Fim
Senão
$N \Downarrow N + 1$
Se $N = N_{max}$ ***então***
Inicio
$f \Downarrow f/2$
$N \Downarrow 0$
Fim
$PLCRL_{K0,K1,K2} \Downarrow LI\_max_{K0,K1,K2}$
Fim
----- Fim Algoritmo 8 -----

Para o nó *v* definido por K_0, K_1 e K_2, após a aplicação da parte iterativa do Algoritmo 8 é feito $PLCRL_{K0,K1,K2} \Downarrow LI\_max_{K0,K1,K2}$.

Os três parâmetros utilizados são definidos da seguinte maneira:

- $f = 2$ conforme Beasley [12].
- $u = 0,0005$. Beasley [12] usa 0,005. Verificamos que com 0,0005 obtivemos melhores resultados.
- N_{max} depende do problema e do local aplicação. Para o nó raiz é utilizado $N_{max} = 60$. Para os demais nós é utilizado $N_{max} = 2$ para problemas com até 150 facilidades e com custos variáveis muito próximos um dos outros, caso contrário, é utilizado $N_{max} = 20$. Estes valores foram obtidos através de experiências práticas realizadas.

3.3.4 Atualização do Limite Inferior

Toda vez que uma facilidade for aberta/fechada o $PLCRL_{K0,K1,K2}$ pode ser recalculado, conforme Algoritmo 8. Contudo, este algoritmo é caro, computacionalmente falando, não sendo viável a sua aplicação a cada abertura/fechamento de uma facilidade. Com isso, será apresentado, nesta seção, uma forma alternativa ao Algoritmo 8, a simples atualização do $PLCRL_{K0,K1,K2}$ quando da abertura/fechamento de uma facilidade.

a) Abertura de uma nova facilidade

Dada uma solução (x^*,y^*) do $PLCRL_{K0,K1,K2}$, se for aberta uma nova facilidade $i \in K_2$ podem ocorrer duas situações:

1. $y_i^* = 1$, ou seja, a facilidade i já estava aberta na solução (x^*,y^*). Nesse caso, p_i já foi considerado e nada será mudado na solução do $PLCRL_{K0,K1,K2}$.
2. $y_i^* = 0$, ou seja, a facilidade i estava fechada na solução ótima (x^*,y^*). Nesse caso, é preciso considerar a facilidade i como aberta na solução do $PLCRL_{K0,K1,K2}$. Para isso, o valor de p_i deve ser acrescentado ao valor do $PLCRL_{K0,K1,K2}$. Neste acréscimo alguns cuidados devem ser tomados. O primeiro é a verificação se $\sum_{k \in K1 \cup K2} y_k^* = f\_max_{K_0,K_1,K_2}$. Essa condição indica que o número de facilidade abertas, na solução do $PLCRL_{K0,K1,K2}$, é igual ao número máximo permitido. Com isso, como será aberta a facilidade i, é necessário fechar uma facilidade aberta na solução y^* (a facilidade aberta com o maior p_k, $k \in K_2$). O segundo cuidado a ser tomado é quando, na solução do $PLCRL_{K0,K1,K2}$, foi aberta alguma facilidade com $p_k>0$, $k \in K_2$. Esse caso acontece quando o número mínimo de facilidades não foi atingido com as facilidades com $p_k \leq 0$, $k \in K_2$. Com isso, como será aberta a facilidade i, é necessário fechar a facilidade com maior $p_k>0$, $k \in K_2$.

Seja l a última facilidade aberta na solução (x^*,y^*), ou seja, a facilidade aberta com o maior p_k, $k \in K_2$, então quando a facilidade i é aberta, o Algoritmo 9 mostra a atualização do $PLCRL_{K0,K1,K2}$.

Algoritmo 9: Abrir facilidade no $PLCRL_{K0,K1,K2}$

Se $y_i^* \neq 1$ ***então*** *{ condição 1 }*

Inicio

Se $\sum_{k \in K1 \cup K2} y_k^* = f\_max_{K_0,K_1,K_2}$ ***então*** *{ condição 2 }*

Inicio

$PLCRL_{K0,K1,K2} \Downarrow PLCRL_{K0,K1,K2} + p_i - p_l$

$y_l^* \Downarrow 0$

$(\forall j \in J)\ x_{lj} \Downarrow 0$

Fim

Senão

Inicio

Se $p_l > 0$ ***então*** *{ condição 3 }*

Inicio

$PLCRL_{K0,K1,K2} \Downarrow PLCRL_{K0,K1,K2} + p_i - p_l$

$y_l^* \Downarrow 0$

$(\forall j \in J)\ x_{lj} \Downarrow 0$

Fim

Senão

$PLCRL_{K0,K1,K2} \Downarrow PLCRL_{K0,K1,K2} + p_i$

Fim

$y_i^* \Downarrow 1$

$(\forall j \in J)$

Inicio

Se $C_{ij} < 0$ ***então***

$x_{ij}^* \Downarrow 1$

Senão

$x_{ij}^* \Downarrow 0$

Fim

Fim

Senão

Permanecem inalterados os valores de y^* e x^*

----- Fim Algoritmo 9 -----

A condição 1, do Algoritmo 9, verifica se a facilidade já tinha seu valor em 1 na solução do $PLCRL_{K0,K1,K2}$. A condição 2 verifica se o número máximo de facilidades abertas tinha sido atingido na solução do $PLCRL_{K0,K1,K2}$. A condição 3 acontece caso a facilidade l tenha sido aberta para atender o número mínimo de facilidades, contudo $p_l > 0$.

b) Fechamento de uma nova facilidade

Dada uma solução (x^*,y^*) do $PLCRL_{K0,K1,K2}$, se for fechada uma nova facilidade $i \in K_2$ podem ocorrer duas situações:

3. $y_i^* = 0$, ou seja, a facilidade i já estava fechada na solução (x^*,y^*). Nesse caso, nada será mudado na solução do $PLCRL_{K0,K1,K2}$.
4. $y_i^* = 1$, ou seja, a facilidade i estava aberta na solução ótima (x^*,y^*). Nesse caso, é preciso considerar a facilidade i como fechada na solução do $PLCRL_{K0,K1,K2}$. Para isso, o valor de p_i deve ser desconsiderado, ou seja, retirado do valor do $PLCRL_{K0,K1,K2}$. Para desconsiderar p_i, alguns cuidados devem ser tomados. O primeiro é a verificação se $\sum_{k \in K1 \cup K2} y_k^* = f\_min_{K_0,K_1,K_2}$. Essa condição indica que o número de facilidade abertas, na solução do $PLCRL_{K0,K1,K2}$, é igual ao número mínimo permitido. Com isso, como será fechada a facilidade i, é necessário abrir uma facilidade que estava fechada na solução y* (a facilidade fechada com o menor p_k, $k \in K_2$). O segundo cuidado é quando $\sum_{k \in K1 \cup K2} y_k^* = f\_max_{K_0,K_1,K_2}$, ou seja, quando o número de facilidades abertas é igual ao máximo permitido. Nesse caso, como a facilidade i será fechada, caso exista $y_k^*=0$, $k \in K_2$ com $p_k<0$, é necessário fechar a facilidade k como o menor p_k.

Seja l a próxima facilidade que seria aberta na solução (x^*,y^*), ou seja, a facilidade fechada com o menor p_k, $k \in K_2$, então quando a facilidade i é fechada, o Algoritmo 10 mostra a atualização do $PLCRL_{K0,K1,K2}$.

Algoritmo 10: Fechar facilidade no $PLCRL_{K0,K1,K2}$

Se $y_i^* \neq 0$ ***então*** *{ condição 1 }*

Inicio

Se $\sum_{k \in K1 \cup K2} y_k^* = f\_min_{K_0,K_1,K_2}$ ***então*** *{ condição 2 }*

Inicio

$PLCRL_{K0,K1,K2} \Downarrow PLCRL_{K0,K1,K2} - p_i + p_l$

$y_l \Downarrow 1$

$(\forall j \in J)$

Inicio

Se $C_{kj} < 0$ ***então***

$x_{lj} \Downarrow 1$

Senão

$x_{lj} \Downarrow 0$

Fim

Fim

Senão

Inicio

Se $p_l < 0$ ***então*** *{ condição 3 }*

Inicio

$PLCRL_{K0,K1,K2} \Downarrow PLCRL_{K0,K1,K2} - p_i + p_l$

$y_l \Downarrow 1$

$(\forall j \in J)$

Inicio

Se $C_{kj} < 0$ ***então***

$x_{lj} \Downarrow 1$

Senão

$x_{lj} \Downarrow 0$

Fim

Fim

Senão

$PLCRL_{K0,K1,K2} \Downarrow PLCRL_{K0,K1,K2} - p_i$

Fim

$y_i^* \Downarrow 0$

$(\forall j \in J)\ x_{ij}^* \Downarrow 0$

Fim

Senão

Permanecem inalterados os valores de y* e x^*

----- Fim Algoritmo 10 -----

A condição 1, do Algoritmo 10, verifica se a facilidade já tinha seu valor em *0* na solução do $PLCRL_{K0,K1,K2}$. A condição 2 verifica se o número de facilidades abertas é igual ao mínimo permitido. A condição 3 acontece caso a facilidade *l* tenha $p_l < 0$, podendo com isso diminuir o valor de *PLCRL*. Esse caso acontece quando o número de facilidades abertas na solução (y^*,x^*) é igual ao máximo permitido, e ainda existe $p_k < 0$, $k \in K_2$, com $y_k^* = 0$.

3.4 Testes de redução utilizando $PLCRL_{K0,K1,K2}$

Da mesma forma que no Capítulo 2, será utilizado testes de redução para tentar diminuir a dimensão do problema e em alguns casos até obter a solução ótima. Os testes são baseados no $PLCRL_{K0,K1,K2}$.

Suponha calculado o $PLCRL_{K0,K1,K2}$ em um nó v da árvore de busca, tendo sido obtida uma solução (x^*,y^*), e suponha que já se tenha um limite superior LS para o PLC. Caso $PLCRL_{K0,K1,K2} \geq LS$ ocorre um corte, pois todos os nós descendentes de v terão como solução valores maiores ou iguais ao LS, já que o $PLCRL_{K0,K1,K2}$ é um limite inferior. Caso no entanto, $PLCRL_{K0,K1,K2} < LS$, será feito um estudo para as facilidades em K_2, estudando as alterações no valor do limite inferior caso estas variáveis sejam abertas ou fechadas. Para a realização desse estudo será definido o seguinte problema:

- $PLCRL_{K0,K1,K2}(y_k=1 \text{ ou } 0)$: relaxação Lagrangeana do PLC para um dado K_0, K_1 e K_2 abrindo/fechando adicionalmente uma facilidade $k \in K_2$.

Suponha uma facilidade k, tal que, na solução do $PLCRL_{K0,K1,K2}$ tenha sido obtido o valor $y_k = 0$, ou seja, existe uma "tendência" a fechar esta facilidade. Caso seja fixado $y_k = 1$, ou seja, forçado a abertura desta facilidade e recalculado novamente o problema relaxado, seja $PLCRL_{K0,K1,K2}(y_k=1)$ o novo valor obtido. Se $PLCRL_{K0,K1,K2}(y_k=1)$ agora for maior ou igual à LS isto é uma comprovação que para os valores de K_0, K_1 e K_2, dados no início, a facilidade k não pode ser aberta. Portanto, é possível fazer $K_0 = K_0 \cup \{k\}$ e $K_2 = K_2 - \{k\}$. O $PLCRL_{K0,K1,K2}(y_k=1)$ será calculado segundo o Algoritmo 9. Por este algoritmo ser bastante eficiente, diferentemente do $PLCRI_{K0,K1,K2}(y_k=1)$, o cálculo do $PLCRL_{K0,K1,K2}(y_k=1)$ não sofrerá nenhum controle mais rígido.

Seja y^* a solução ótima dada pelo cálculo do $PLCRL_{K0,K1,K2}$, então se $y_k^* = 1$ o $PLCRL_{K0,K1,K2} = PLCRL_{K0,K1,K2}(y_k=1)$. Nesse caso, o $PLCRL_{K0,K1,K2}$ indica que a facilidade k tem "tendência" de ser aberta, de nada valerá o cálculo de $PLCRL_{K0,K1,K2}(y_k=1)$. Para que o cálculo do $PLCRI_{K0,K1,K2}(y_k=1)$ seja efetivo, serão somente consideradas as facilidades com "tendência" de serem fechadas, ou seja, as facilidades com $y_k^* = 0, \forall k \in K_2$.

O mesmo tipo de procedimento aplica-se para uma facilidade k onde $y_k^* = 1$, ou seja, trata-se de uma facilidade com uma tendência a ser aberta. Se for forçado o fechamento desta facilidade e recalculado novamente o problema relaxado, seja $PLCRL_{K0,K1,K2}(y_k=0)$ o novo valor obtido. Se $PLCRL_{K0,K1,K2}(y_k=0)$ agora for maior ou igual à LS isto é uma comprovação que para os valores de K_0, K_1 e K_2, dados no início, a facilidade k não pode ser fechada. Portanto, é possível fazer $K_1 = K_1 \cup \{k\}$ e $K_2 = K_2 - \{k\}$. O $PLCRL_{K0,K1,K2}(y_k=1)$ será calculado segundo o Algoritmo 10. Por este algoritmo ser bastante eficiente, diferentemente do $PLCRI_{K0,K1,K2}(y_k=0)$, o cálculo do $PLCRL_{K0,K1,K2}(y_k=0)$ não sofrerá nenhum controle mais rígido.

Seja LS um limite superior para o PLC, então o Algoritmo 11 sintetiza a realização dos testes de redução que utilizam como base o $PLCRL_{K0,K1,K2}$. O passo 1 é referente ao nó raiz da árvore. O passo 3 verifica a realização do corte através do limite inferior $PLCRL_{K0,K1,K2}$ e os passos 4 e 5 aplicam os testes de redução.

Algoritmo 11: aplicação dos testes de redução baseados em $PLCRL_{K0,K1,K2}$

Passo 1: *{inicialização}*

$K_0 = K_1 = \varnothing$; $K_2 = I$;

Passo 2:

Calcule $PLCRL_{K0,K1,K2}$. Seja (x*,y*) a solução correspondente.

Passo 3:

Se $PLCRL_{K0,K1,K2} \geq LS$ ***então*** *backtracking*

Senão ir para Passo 4

Passo 4: *{ teste de fechar }*

Para todo $k \in K_2 \mid y_k = 0$ ***fazer***

Calcule $PLCRL_{K0,K1,K2}(y_k=1)$

Se $PLCRL_{K0,K1,K2}(y_k=1) \geq LS$ ***então***

Fazer $K_0 = K_0 \cup \{k\}$ e $K_2 = K_2 - \{k\}$

Passo 5: *{ teste de abrir }*

Para todo $k \in K_2 \mid y_k = 1$ ***fazer***

Calcule $PLCRL_{K0,K1,K2}(y_k=0)$

Se $PLCRL_{K0,K1,K2}(y_k=0) \geq LS$ ***então***

Fazer $K_1 = K_1 \cup \{k\}$ e $K_2 = K_2 - \{k\}$

----- Fim Algoritmo 11 -----

Para facilitar a referência no restante do texto, o procedimento que faz $K_1 = K_1 \cup \{k\}$ e $K_2 = K_2 - \{k\}$ será denominado de *O-RLTeste* e o procedimento que faz $K_0 = K_0 \cup \{k\}$ e $K_2 = K_2 - \{k\}$ de *C-RLTeste*.

3.5 Resultados Computacionais

Os resultados que serão apresentados aqui não tem a intenção de identificar qual a melhor técnica a ser utilizada, mas sim dar noções da potencialidade dos testes de redução e do comportamento dos limites inferiores. Serão apresentados resultados dos testes de redução: *O-RLTeste*, *C-RLTeste*, *O-RITeste*, *C-RITeste*, *O-Teste* e *C-Teste*; e dos limites inferiores: *PLCRL* e *PLCRI*. Os testes serão rodados sob problemas da literatura com $K_0 = K_1 = \varnothing$ e $K_2 = I$, isto é, serão aplicados testes sob o primeiro nó da árvore de busca, a qual será apresentada no Capítulo 4. Por esta razão os resultados não serão, de todo, representativos, pois à medida que os conjuntos K_0, K_1 e K_2 vão se alterando, na árvore de busca, os resultados dos testes e dos limites inferiores também se alteram. O computador utilizado para obtenção dos resultados foi um Pentium III, 700 mhz com 256 Mega bytes de memória principal. Vale observar que é uma máquina defasada para os dias de hoje.

A Tabela 15 mostra os resultados obtidos, para os problemas de Kühn & Hamburger [32] e Beasley [11], para os limites inferiores considerando $K_0 = K_1 = \varnothing$ e $K_2 = I$. A primeira coluna representa o problema considerado, a segunda coluna representa as dimensões do problema $|I|x|J|$, a terceira coluna representa o valor de $PLCRI_{K0,K1,K2}$, a quarta coluna representa o tempo em segundo para se resolver o $PLCRI_{K0,K1,K2}$, a quinta coluna representa *gap* entre o valor de $PLCRI_{K0,K1,K2}$ e o valor ótimo, a sexta coluna o valor do $PLCRL_{K0,K1,K2}$, a setíma coluna representa o tempo em segundo para se resolver o $PLCRL_{K0,K1,K2}$, a oitava coluna representa *gap* entre o valor de $PLCRL_{K0,K1,K2}$ e o valor da solução ótima e a nona coluna representa o *gap* entre $PLCRI_{K0,K1,K2}$ e $PLCRL_{K0,K1,K2}$ dado pela expressão $(100 - (PLCRI_{K0,K1,K2}*100)/PLCRL_{K0,K1,K2})$.

		Relaxação linear			Relaxação Lagrangeana			
Problema	**|I| x |J|**	**RI**	**RI seg.**	**gap RI**	**RL**	**RLseg.**	**gap RL**	**gap**
cap41.txt	16x50	1018151,62	0,00	2,14	1040182,81	0,00	0,03	2,12
cap42.txt		1071419,62	0,00	2,42	1096548,63	0,00	0,13	2,29
cap43.txt		1124687,62	0,00	2,46	1150724,75	0,00	0,20	2,26
cap44.txt		1204589,62	0,00	2,50	1231111,13	0,00	0,36	2,15
cap51.txt		941395,13	0,00	8,18	1024631,19	0,00	0,06	8,12
cap61.txt		865538,69	0,00	7,19	932615,75	0,00	0,00	7,19
cap62.txt		883917,69	0,00	9,60	977799,38	0,00	0,00	9,60
cap63.txt		902175,20	0,00	11,04	1012710,25	0,00	0,14	10,91
cap64.txt		928920,97	0,00	11,16	1045632,94	0,00	0,00	11,16
cap71.txt		845067,17	0,00	9,39	932615,75	0,00	0,00	9,39
cap72.txt		849798,51	0,00	13,09	977799,38	0,00	0,00	13,09
cap73.txt		854529,84	0,00	15,45	1010641,44	0,00	0,00	15,45
cap74.txt		861626,83	0,00	16,75	1034977,00	0,00	0,00	16,75
cap81.txt	25x50	771766,59	0,00	7,96	837010,00	0,00	0,18	7,79
cap82.txt		825982,09	0,00	9,32	910388,13	0,00	0,06	9,27
cap83.txt		879250,09	0,00	9,90	975288,13	0,00	0,06	9,85
cap84.txt		959152,09	0,00	10,31	1067357,13	0,00	0,19	10,14
cap91.txt		680700,15	0,00	14,65	796648,44	0,00	0,11	14,55
cap92.txt		699639,48	0,00	18,24	854972,31	0,00	0,09	18,17
cap93.txt		718457,34	0,00	19,87	894851,13	0,00	0,20	19,71
cap94.txt		746143,37	0,00	21,13	942201,56	0,00	0,41	20,81
cap101.txt		659604,51	0,00	17,29	796648,44	0,00	0,11	17,20
cap102.txt		664480,09	0,00	22,33	854704,19	0,00	0,09	22,26
cap103.txt		669355,68	0,00	25,13	893782,13	0,00	0,03	25,11
cap104.txt		676669,04	0,00	27,16	928941,75	0,00	0,00	27,16
cap111.txt	50x50	734545,12	0,00	11,09	825823,00	0,00	0,04	11,05
cap112.txt		789791,52	0,00	12,38	900144,19	0,00	0,14	12,26
cap113.txt		843059,52	0,00	13,14	968348,63	0,00	0,23	12,94
cap114.txt		922961,52	0,00	13,20	1060584,00	0,00	0,26	12,98
cap121.txt		652929,95	0,00	17,78	793439,56	0,00	0,09	17,71
cap122.txt		672168,95	0,00	21,16	851801,94	0,00	0,08	21,09
cap123.txt		691407,95	0,00	22,77	894329,25	0,00	0,11	22,69
cap124.txt		719830,41	0,00	23,91	942083,94	0,00	0,42	23,59
cap131.txt		631500,53	0,00	20,48	793439,56	0,00	0,09	20,41
cap132.txt		636453,25	0,00	25,25	851448,00	0,00	0,01	25,25
cap133.txt		641405,98	0,00	28,19	893076,69	0,00	0,02	28,18
cap134.txt		648835,06	0,00	30,15	928941,75	0,00	0,00	30,15
capa1	100x1000	13820444,53	0,00	28,17	18364890,00	60,00	4,55	24,75
capa2		11748732,72	0,00	36,28	17814624,00	96,00	3,38	34,05
capa3		10347597,20	0,00	41,75	17365986,00	159,00	2,25	40,41
capa4		9337014,25	0,00	45,59	17158252,00	32,00	0,01	45,58
capb1		10053757,27	0,00	26,38	13557287,00	34,00	0,73	25,84
capb2		8908494,38	0,00	33,33	13260002,00	33,00	0,76	32,82
capb3		8081721,80	0,00	38,77	13133381,00	92,00	0,49	38,46
capb4		7457909,28	0,00	42,99	13054975,00	61,00	0,21	42,87
capc1		7991461,82	0,00	31,38	11567044,00	28,00	0,68	30,91
capc2		7329249,20	0,00	36,66	11496743,00	24,00	0,64	36,25
capc3		6816861,51	0,00	40,82	11491557,00	28,00	0,24	40,68
capc4		6409201,06	0,00	44,30	11491915,00	26,00	0,12	44,23

Média			***0,00***	***20,42***		***13,73***	***0,37***	***20,16***

Tabela 15 – resultados dos limites inferiores

A Tabela 16 mostra os resultados obtidos, sobre os problemas de Cornuejols, Sridharan e Thizy [20], para os limites inferiores considerando $K_0 = K_1 = \varnothing$ e $K_2 = I$. A primeira coluna representa o problema considerado, a segunda coluna representa as dimensões do problema $|I|x|J|$, a terceira coluna representa o valor de $PLCRI_{K0,K1,K2}$, a quarta coluna representa o tempo em segundo para se resolver o $PLCRI_{K0,K1,K2}$, a quinta coluna representa *gap* entre o $PLCRI_{K0,K1,K2}$ e a solução ótima, a sexta coluna o valor do $PLCRL_{K0,K1,K2}$, a setíma coluna representa o tempo em segundo para se resolver o $PLCRL_{K0,K1,K2}$, a oitava coluna representa *gap* entre o $PLCRL_{K0,K1,K2}$ e a solução ótima e a nona coluna representa o *gap* entre $PLCRI_{K0,K1,K2}$ e $PLCRL_{K0,K1,K2}$ dado pela expressão $(100 - (PLCRI_{K0,K1,K2} * 100)/PLCRL_{K0,K1,K2})$.

			Relaxação linear			Relaxação Lagrangeana			
Problema	**|I| x |J|**	**Z**	**RI**	**RI seg.**	**gap RI**	**RL**	**RL seg.**	**gap RL**	**gap**
thiz1-a1.dad	8x25	9397,00	8847,86	0,00	5,84	8858,31	0,00	5,73	0,12
thiz1-a2.dad		11399,00	10581,90	0,00	7,17	10670,76	0,00	6,39	0,83
thiz1-a3.dad		10426,00	9801,28	0,00	5,99	9816,22	0,00	5,85	0,15
thiz1-a4.dad		10206,00	10066,33	0,00	1,37	10073,28	0,00	1,30	0,07
thiz1-a5.dad		12237,00	11514,68	0,00	5,90	11565,49	0,00	5,49	0,44
thiz1-b1.dad	16x25	14844,00	14376,43	0,00	3,15	14389,75	0,00	3,06	0,09
thiz1-b2.dad		12355,00	11964,64	0,00	3,16	11968,48	0,00	3,13	0,03
thiz1-b3.dad		15534,00	15191,05	0,00	2,21	15202,27	0,00	2,14	0,07
thiz1-b4.dad		14827,00	14580,83	0,00	1,66	14584,75	0,00	1,63	0,03
thiz1-b5.dad		14226,00	13760,49	0,00	3,27	13785,81	0,00	3,09	0,18
thiz1-c1.dad	25x25	16197,00	15969,24	0,00	1,41	15969,15	0,00	1,41	0,00
thiz1-c2.dad		18593,00	18492,53	0,00	0,54	18492,43	0,00	0,54	0,00
thiz1-c3.dad		16803,00	16524,47	0,00	1,66	16524,39	0,00	1,66	0,00
thiz1-c4.dad		15833,00	15521,70	0,00	1,97	15526,11	0,00	1,94	0,03
thiz1-c5.dad		16904,00	16642,25	0,00	1,55	16641,14	0,00	1,56	-0,01
thiz1-d1.dad	16x50	19377,00	18928,00	0,00	2,32	18951,68	0,00	2,19	0,12
thiz1-d2.dad		18749,00	18650,81	0,00	0,52	18670,44	0,00	0,42	0,11
thiz1-d3.dad		21115,00	20627,36	0,00	2,31	20632,93	0,00	2,28	0,03
thiz1-d4.dad		20227,00	19764,60	0,00	2,29	19807,21	0,00	2,08	0,22
thiz1-d5.dad		19416,00	19188,51	0,00	1,17	19236,95	0,00	0,92	0,25
thiz1-e1.dad	33x50	27189,00	26758,37	0,00	1,58	26785,43	0,00	1,48	0,10
thiz1-e2.dad		27891,00	27564,55	0,00	1,17	27564,38	0,00	1,17	0,00
thiz1-e3.dad		28194,00	28141,70	0,00	0,19	28141,09	0,00	0,19	0,00
thiz1-e4.dad		29080,00	28774,95	0,00	1,05	28777,40	0,00	1,04	0,01
thiz1-e5.dad		28963,00	28494,22	0,00	1,62	28490,56	0,00	1,63	-0,01
thiz1-f1.dad	50x50	33647,00	33561,36	0,00	0,25	33557,84	0,00	0,26	-0,01
thiz1-f2.dad		36237,00	36200,89	0,00	0,10	36199,83	0,00	0,10	0,00
thiz1-f3.dad		36195,00	36057,07	0,00	0,38	36053,69	0,00	0,39	-0,01

thiz1-f4.dad		34073,00	34005,48	0,00	0,20	34000,80	0,00	0,21	-0,01
thiz1-f5.dad		34609,00	34609,00	0,00	0,00	34608,81	0,00	0,00	0,00
thiz2-a1.dad	8x25	9052,00	8376,57	0,00	7,46	8513,94	0,00	5,94	1,61
thiz2-a2.dad		10298,00	9057,19	0,00	12,05	9234,28	0,00	10,33	1,92
thiz2-a3.dad		11179,00	10187,94	0,00	8,87	10340,16	0,00	7,50	1,47
thiz2-a4.dad		8898,00	8565,06	0,00	3,74	8699,10	0,00	2,24	1,54
thiz2-a5.dad		9033,00	8374,41	0,00	7,29	8559,02	0,00	5,25	2,16
thiz2-b1.dad	16x25	10415,00	9891,47	0,00	5,03	9900,19	0,00	4,94	0,09
thiz2-b2.dad		12820,00	12500,25	0,00	2,49	12635,74	0,00	1,44	1,07
thiz2-b3.dad		10838,00	10577,58	0,00	2,40	10611,10	0,00	2,09	0,32
thiz2-b4.dad		12751,00	12166,62	0,00	4,58	12194,92	0,00	4,36	0,23
thiz2-b5.dad		11045,00	10736,34	0,00	2,79	10741,89	0,00	2,74	0,05
thiz2-c1.dad	25x25	14660,00	14280,95	0,00	2,59	14325,75	0,00	2,28	0,31
thiz2-c2.dad		15623,00	15377,50	0,00	1,57	15400,48	0,00	1,42	0,15
thiz2-c3.dad		15354,00	15213,89	0,00	0,91	15232,79	0,00	0,79	0,12
thiz2-c4.dad		14466,00	14194,93	0,00	1,87	14210,30	0,00	1,77	0,11
thiz2-c5.dad		14000,00	13547,28	0,00	3,23	13573,75	0,00	3,04	0,19
thiz2-d1.dad	16x50	17830,00	17147,55	0,00	3,83	17175,45	0,00	3,67	0,16
thiz2-d2.dad		16756,00	16198,76	0,00	3,33	16327,08	0,00	2,56	0,79
thiz2-d3.dad		17181,00	16979,57	0,00	1,17	17101,12	0,00	0,46	0,71
thiz2-d4.dad		18444,00	17864,29	0,00	3,14	17990,00	0,00	2,46	0,70
thiz2-d5.dad		16910,00	16910,00	0,00	0,00	16909,95	0,00	0,00	0,00
thiz2-e1.dad	33x50	22570,00	22182,40	0,00	1,72	22230,36	0,00	1,50	0,22
thiz2-e2.dad		24362,00	24036,98	0,00	1,33	24078,08	0,00	1,17	0,17
thiz2-e3.dad		23591,00	23161,08	0,00	1,82	23230,38	0,00	1,53	0,30
thiz2-e4.dad		24718,00	24530,90	0,00	0,76	24551,18	0,00	0,67	0,08
thiz2-e5.dad		23962,00	23569,99	0,00	1,64	23640,09	0,00	1,34	0,30
thiz2-f1.dad	50x50	27905,00	27832,67	0,00	0,26	27836,92	0,00	0,24	0,02
thiz2-f2.dad		28352,00	28038,52	0,00	1,11	28064,88	0,00	1,01	0,09
thiz2-f3.dad		27624,00	27566,83	0,00	0,21	27584,17	0,00	0,14	0,06
thiz2-f4.dad		27117,00	27073,37	0,00	0,16	27081,42	0,00	0,13	0,03
thiz2-f5.dad		29136,00	29116,04	0,00	0,07	29130,91	0,00	0,02	0,05
thiz3-a1.dad	8x25	5058,00	3998,10	0,00	20,95	4698,63	0,00	7,11	14,91
thiz3-a2.dad		4489,00	3923,91	0,00	12,59	4422,04	0,00	1,49	11,26
thiz3-a3.dad		4365,00	3654,28	0,00	16,28	4110,13	0,00	5,84	11,09
thiz3-a4.dad		5326,00	4549,70	0,00	14,58	5057,35	0,00	5,04	10,04
thiz3-a5.dad		4449,00	3932,63	0,00	11,61	4264,88	0,00	4,14	7,79
thiz3-b1.dad	16x25	4818,00	4387,57	0,00	8,93	4528,45	0,00	6,01	3,11
thiz3-b2.dad		5234,00	4917,49	0,00	6,05	5157,71	0,00	1,46	4,66
thiz3-b3.dad		5461,00	4893,84	0,00	10,39	5221,39	0,00	4,39	6,27
thiz3-b4.dad		5267,00	4510,63	0,00	14,36	4979,64	0,00	5,46	9,42
thiz3-b5.dad		5270,00	4801,53	0,00	8,89	5077,34	0,00	3,66	5,43
thiz3-c1.dad	25x25	5327,00	5201,96	0,00	2,35	5305,62	0,00	0,40	1,95
thiz3-c2.dad		5783,00	5403,50	0,00	6,56	5514,99	0,00	4,63	2,02
thiz3-c3.dad		6605,00	6207,26	0,00	6,02	6574,58	0,00	0,46	5,59
thiz3-c4.dad		6033,00	5705,35	0,00	5,43	5908,69	0,00	2,06	3,44
thiz3-c5.dad		6149,00	5858,34	0,00	4,73	6013,50	0,00	2,20	2,58
thiz3-d1.dad	16x50	8135,00	7630,53	0,00	6,20	7797,69	0,00	4,15	2,14
thiz3-d2.dad		8251,00	7596,69	0,00	7,93	7801,74	0,00	5,44	2,63
thiz3-d3.dad		8244,00	7679,10	0,00	6,85	8133,07	0,00	1,35	5,58

thiz3-d4.dad		8438,00	7528,80	0,00	10,78	8267,15	0,00	2,02	8,93
thiz3-d5.dad		7680,00	7097,26	0,00	7,59	7519,49	0,00	2,09	5,62
thiz3-e1.dad	33x50	9826,00	9265,96	0,00	5,70	9578,40	0,00	2,52	3,26
thiz3-e2.dad		10291,00	9733,79	0,00	5,41	9991,57	0,00	2,91	2,58
thiz3-e3.dad		10204,00	9742,30	0,00	4,52	10043,68	0,00	1,57	3,00
thiz3-e4.dad		9526,00	9221,07	0,00	3,20	9452,63	0,00	0,77	2,45
thiz3-e5.dad		10350,00	9844,10	0,00	4,89	10135,55	0,00	2,07	2,88
thiz3-f1.dad	50x50	11214,00	10947,99	0,00	2,37	11053,41	0,00	1,43	0,95
thiz3-f2.dad		11937,00	11626,11	0,00	2,60	11800,42	0,00	1,14	1,48
thiz3-f3.dad		12235,00	11975,75	0,00	2,12	12124,87	0,00	0,90	1,23
thiz3-f4.dad		12242,00	11971,63	0,00	2,21	12177,37	0,00	0,53	1,69
thiz3-f5.dad		11374,00	11228,37	0,00	1,28	11294,91	0,00	0,70	0,59
thiz5-a1.dad	8x25	4894,00	3519,24	0,00	28,09	4325,27	0,00	11,62	18,64
thiz5-a2.dad		5582,00	3592,34	0,00	35,64	5217,73	0,00	6,53	31,15
thiz5-a3.dad		4444,00	2965,19	0,00	33,28	4204,45	0,00	5,39	29,48
thiz5-a4.dad		4596,00	3132,66	0,00	31,84	4071,41	0,00	11,41	23,06
thiz5-a5.dad		4606,00	3633,36	0,00	21,12	4606,00	0,00	0,00	21,12
thiz5-b1.dad	16x25	4449,00	3955,34	0,00	11,10	4379,05	0,00	1,57	9,68
thiz5-b2.dad		4854,00	3925,21	0,00	19,13	4833,68	0,00	0,42	18,79
thiz5-b3.dad		4681,00	3888,24	0,00	16,94	4500,70	0,00	3,85	13,61
thiz5-b4.dad		4880,00	4054,08	0,00	16,92	4792,97	0,00	1,78	15,42
thiz5-b5.dad		4907,00	3893,39	0,00	20,66	4827,71	0,00	1,62	19,35
thiz5-c1.dad	25x25	5040,00	4561,49	0,00	9,49	4847,50	0,00	3,82	5,90
thiz5-c2.dad		5267,00	4722,72	0,00	10,33	5126,44	0,00	2,67	7,88
thiz5-c3.dad		4120,00	3624,38	0,00	12,03	3876,78	0,00	5,90	6,51
thiz5-c4.dad		4862,00	4483,10	0,00	7,79	4742,72	0,00	2,45	5,47
thiz5-c5.dad		5086,00	4663,15	0,00	8,31	4975,07	0,00	2,18	6,27
thiz5-d1.dad	16x50	6697,00	5880,54	0,00	12,19	6661,42	0,00	0,53	11,72
thiz5-d2.dad		7581,00	5971,50	0,00	21,23	7308,54	0,00	3,59	18,29
thiz5-d3.dad		7834,00	6313,63	0,00	19,41	7679,99	0,00	1,97	17,79
thiz5-d4.dad		8115,00	5889,06	0,00	27,43	7734,18	0,00	4,69	23,86
thiz5-d5.dad		7847,00	6689,81	0,00	14,75	7522,53	0,00	4,13	11,07
thiz5-e1.dad	33x50	8133,00	7219,01	0,00	11,24	7987,01	0,00	1,79	9,62
thiz5-e2.dad		7575,00	6906,73	0,00	8,82	7504,26	0,00	0,93	7,96
thiz5-e3.dad		8354,00	7688,98	0,00	7,96	8248,37	0,00	1,26	6,78
thiz5-e4.dad		7929,00	7271,09	0,00	8,30	7854,07	0,00	0,94	7,42
thiz5-e5.dad		8364,00	7540,79	0,00	9,84	8183,16	0,00	2,16	7,85
thiz5-f1.dad	50x50	9675,00	9020,80	0,00	6,76	9550,51	0,00	1,29	5,55
thiz5-f2.dad		9110,00	8339,18	0,00	8,46	8853,21	0,00	2,82	5,81
thiz5-f3.dad		9098,00	8653,00	0,00	4,89	8875,12	0,00	2,45	2,50
thiz5-f4.dad		9194,00	8715,90	0,00	5,20	9110,87	0,00	0,90	4,34
thiz5-f5.dad		10111,00	9398,78	0,00	7,04	9859,98	0,00	2,48	4,68
thizx-a1.dad	8x25	4100,00	2707,04	0,00	33,97	3851,23	0,00	6,07	29,71
thizx-a2.dad		4494,00	2861,87	0,00	36,32	4494,00	0,00	0,00	36,32
thizx-a3.dad		4459,00	2680,16	0,00	39,89	4459,00	0,00	0,00	39,89
thizx-a4.dad		5387,00	3249,85	0,00	39,67	5060,55	0,00	6,06	35,78
thizx-a5.dad		5287,00	2561,74	0,00	51,55	5100,88	0,00	3,52	49,78
thizx-b1.dad	16x25	4140,00	2797,28	0,00	32,43	3809,51	0,00	7,98	26,57
thizx-b2.dad		4633,00	2904,64	0,00	37,31	4238,96	0,00	8,51	31,48
thizx-b3.dad		4545,00	2732,38	0,00	39,88	4093,08	0,00	9,94	33,24

thizx-b4.dad		3899,00	2710,56	0,00	30,48	3899,00	0,00	0,00	30,48
thizx-b5.dad		4570,00	2898,54	0,00	36,57	4570,00	0,00	0,00	36,57
thizx-c1.dad	25x25	4948,00	3654,24	0,00	26,15	4892,58	0,00	1,12	25,31
thizx-c2.dad		3930,00	2748,94	0,00	30,05	3373,24	0,00	14,17	18,51
thizx-c3.dad		4653,00	3447,84	0,00	25,90	4478,28	0,00	3,76	23,01
thizx-c4.dad		4442,00	3230,21	0,00	27,28	4173,72	0,00	6,04	22,61
thizx-c5.dad		4112,00	3152,67	0,00	23,33	3867,44	0,00	5,95	18,48
thizx-d1.dad	16x50	7818,00	5228,54	0,00	33,12	7487,71	0,00	4,22	30,17
thizx-d2.dad		6923,00	4308,35	0,00	37,77	6433,61	0,00	7,07	33,03
thizx-d3.dad		7366,00	4452,77	0,00	39,55	7036,63	0,00	4,47	36,72
thizx-d4.dad		6295,00	4273,56	0,00	32,11	6295,00	0,00	0,00	32,11
thizx-d5.dad		7591,00	4457,97	0,00	41,27	7413,91	0,00	2,33	39,87
thizx-e1.dad	33x50	7168,00	5307,26	0,00	25,96	6956,90	0,00	2,95	23,71
thizx-e2.dad		7106,00	5495,42	0,00	22,67	7086,89	0,00	0,27	22,46
thizx-e3.dad		7837,00	5902,35	0,00	24,69	7630,34	0,00	2,64	22,65
thizx-e4.dad		6925,00	5348,02	0,00	22,77	6787,26	0,00	1,99	21,21
thizx-e5.dad		7262,00	5352,55	0,00	26,29	7210,05	0,00	0,72	25,76
thizx-f1.dad	50x50	7392,00	6096,56	0,00	17,52	7221,23	0,00	2,31	15,57
thizx-f2.dad		7994,00	6536,72	0,00	18,23	7927,22	0,00	0,84	17,54
thizx-f3.dad		7814,00	6197,28	0,00	20,69	7533,90	0,00	3,58	17,74
thizx-f4.dad		7659,00	6490,50	0,00	15,26	7539,44	0,00	1,56	13,91
thizx-f5.dad		7759,00	6372,97	0,00	17,86	7450,79	0,00	3,97	14,47
Média				***0,00***	***11,57***		***0,00***	***2,81***	***9,10***

Tabela 16 - resultados dos limites inferiores

A Tabela 17 mostra os resultados obtidos, para os problemas de Ardal [3], para os limites inferiores considerando $K_0 = K_1 = \varnothing$ e $K_2 = I$. A primeira coluna representa o problema considerado, a segunda coluna representa as dimensões do problema $|I|x|J|$, a terceira coluna representa o valor de $PLCRI_{K0,K1,K2}$, a quarta coluna representa o tempo em segundo para se resolver o $PLCRI_{K0,K1,K2}$, a quinta coluna representa *gap* entre o $PLCRI_{K0,K1,K2}$ e a solução ótima, a sexta coluna o valor do $PLCRL_{K0,K1,K2}$, a setíma coluna representa o tempo em segundo para se resolver o $PLCRL_{K0,K1,K2}$, a oitava coluna representa *gap* entre o $PLCRL_{K0,K1,K2}$ e a solução ótima e a nona coluna representa o *gap* entre $PLCRI_{K0,K1,K2}$ e $PLCRL_{K0,K1,K2}$ dado pela expressão $(100 - (PLCRI_{K0,K1,K2}*100)/PLCRL_{K0,K1,K2})$.

			Relaxação linear			Relaxação Lagrangeana			
Problema	**|I|x|J|**	**Z**	**RI**	**RI seg.**	**gap RI**	**RL**	**RL seg.**	**gap RL**	**gap**
karen1-1.dad	75x100	79779,00	79528,71	0,00	0,31	79535,93	1,00	0,30	0,01
karen1-2.dad	75x100	74900,00	74461,91	0,00	0,58	74461,02	2,00	0,59	0,00
karen1-3.dad	75x100	74667,00	74548,67	0,00	0,16	74566,56	1,00	0,13	0,02
karen1-4.dad	75x100	75403,00	75123,77	0,00	0,37	75138,29	2,00	0,35	0,02
karen1-5.dad	75x100	71599,00	71529,00	0,00	0,10	71506,45	2,00	0,13	-0,03
karen2-1.dad	75x100	54177,00	54116,60	0,00	0,11	54116,47	1,00	0,11	0,00
karen2-2.dad	75x100	59705,00	59513,36	0,00	0,32	59541,23	1,00	0,27	0,05
karen2-3.dad	75x100	55546,00	55338,22	0,00	0,37	55355,64	1,00	0,34	0,03
karen2-4.dad	75x100	61624,00	61386,29	0,00	0,39	61440,91	1,00	0,30	0,09
karen2-5.dad	75x100	51829,00	51473,70	0,00	0,69	51493,14	1,00	0,65	0,04
karen3-1.dad	75x100	18469,00	18172,71	0,00	1,60	18365,71	1,00	0,56	1,05
karen3-2.dad	75x100	19023,00	18613,94	0,00	2,15	18852,04	1,00	0,90	1,26
karen3-3.dad	75x100	17535,00	17211,47	0,00	1,85	17360,86	1,00	0,99	0,86
karen3-4.dad	75x100	18324,00	17954,38	0,00	2,02	18210,51	1,00	0,62	1,41
karen3-5.dad	75x100	19999,00	19612,90	0,00	1,93	19785,60	1,00	1,07	0,87
Média				***0,00***	***0,86***		***1,20***	***0,49***	***0,38***

Tabela 17 - resultados dos limites inferiores

A Tabela 18 mostra os resultados obtidos, para os problemas de Kühn & Hamburger [32] e Beasley [11], para os testes de redução *O-RLTeste*, *C-RLTeste*, *O-RITeste*, *C-RITeste, C-Teste* e *O-Teste* considerando $K_0 = K_1 = \varnothing$ e $K_2 = I$, isto é, para o nó raiz. Vale observar que o *C-Teste* só é aplicado caso se obtenha K_1 viável. A primeira coluna representa o problema considerado, a segunda coluna representa as dimensões do problema |I|x|J|, a terceira coluna representa a soma dos sucessos do *O-RLTeste* e do *C-RLTeste*, a quarta coluna representa a soma dos sucessos do *O-RITeste* e de *C-RITeste* e a quinta coluna representa a soma dos sucessos do *O-Teste* e do *C-Teste*. O número de sucessos indica que os testes conseguiram abrir ou fechar facilidades. Não foram considerados os tempos para o cálculo dos testes, pois todos foram desprezíveis.

Problemas	**|I| x |J|**	**RL**	**RI**	**OC**
cap41.txt	16x50	13	11	0
cap42.txt		14	10	0
cap43.txt		13	10	0
cap44.txt		12	9	0
cap51.txt		9	1	0
cap61.txt		0	0	2
cap62.txt		0	0	14
cap63.txt		6	0	8
cap64.txt		11	0	0

cap71.txt		0	0	2
cap72.txt		0	0	14
cap73.txt		0	0	6
cap74.txt		0	0	6
cap81.txt	25x50	10	0	0
cap82.txt		12	0	0
cap83.txt		15	0	0
cap84.txt		6	0	0
cap91.txt		0	0	22
cap92.txt		15	0	12
cap93.txt		14	0	10
cap94.txt		0	0	0
cap101.txt		0	0	22
cap102.txt		0	0	12
cap103.txt		0	0	8
cap104.txt		0	0	4
cap111.txt	50x50	36	0	0
cap112.txt		1	0	0
cap113.txt		25	0	0
cap114.txt		0	0	0
cap121.txt		0	0	14
cap122.txt		38	0	10
cap123.txt		32	0	8
cap124.txt		0	0	0
cap131.txt		0	0	14
cap132.txt		0	0	10
cap133.txt		0	0	6
cap134.txt		0	0	4
capa1	100x1000	58	0	0
capa2		30	0	0
capa3		68	0	0
capa4		0	0	0
capb1		52	0	0
capb2		0	0	0
capb3		0	0	0
capb4		63	0	0
capc1		45	0	0
capc2		3	0	0
capc3		5	0	0
capc4		51	0	0
Total		***657***	***41***	***208***

Tabela 18 – resultados dos testes de redução

A Tabela 19 mostra os resultados obtidos, sobre os problemas de Ardal [3], para os testes de redução *O-RLTeste*, *C-RLTeste*, *O-RITeste*, *C-RITeste* e *O-Teste* considerando $K_0 = K_1 = \varnothing$ e $K_2 = I$, isto é, para o nó raiz. Vale observar que o *C-Teste* só é aplicado caso se obtenha K_1 viável. A primeira coluna representa o problema considerado, a segunda coluna representa as dimensões do problema |I|x|J|, a terceira

coluna representa o número de sucessos do *O-RLTeste* e do *C-RLTeste*, a quarta coluna representa o número de sucessos do *O-RITeste* e de *C-RITeste* e a quinta coluna representa o número de sucessos do *O-Teste* e do *C-Teste*. O número de sucessos indica que os testes conseguiram abrir ou fechar facilidades. Não foram considerados os tempos para o cálculo dos testes, pois todos foram desprezíveis.

Problemas	|I| x |J|	RL	RI	OC
karen1-1.dad	75 x 100	7	13	0
karen1-2.dad	75 x 100	3	7	0
karen1-3.dad	75 x 100	32	28	0
karen1-4.dad	75 x 100	13	15	0
karen1-5.dad	75 x 100	22	25	0
karen2-1.dad	75 x 100	26	19	0
karen2-2.dad	75 x 100	1	1	0
karen2-3.dad	75 x 100	10	13	0
karen2-4.dad	75 x 100	0	0	0
karen2-5.dad	75 x 100	0	0	0
karen3-1.dad	75 x 100	0	0	0
karen3-2.dad	75 x 100	0	0	0
karen3-3.dad	75 x 100	0	0	0
karen3-4.dad	75 x 100	0	0	0
karen3-5.dad	75 x 100	0	0	0
Total		***114***	***121***	***0***

Tabela 19 – resultados dos testes de redução

3.6 Conclusões

O limite inferior *PLCRL* normalmente é melhor do que o *PLCRI*, contudo é mais caro computacionalmente. Para os problemas de Kühn & Hamburger [32] e Beasley [11] a diferença média foi 20% a favor do *PLCRL*, já para os problemas de Ardal [3] a diferença foi muito pequena, em média de 0,38% a favor do *PLCRL*. Provavelmente a diferença entre o *PLCRL* e *PLCRI* é pequena quando a diferença entre os custos de transporte forem pequenas, como é o caso dos problemas de Ardal [3] e Cornuejols, Sridharan e Thizy [20]. A tendência é que à medida em que forem sendo fixados valores para a K_0 e K_1 a diferença entre os dois limites vá caindo. Uma vantagem do *PLCRI* em relação ao *PLCRL* é que o mesmo pode encontrar a solução do *PLC* em certas situações.

Os resultados computacionais mostram a potencialidade dos testes de redução, que mesmo com todas as facilidades indefinidas conseguem determinar o *status* de diversas facilidades. A tendência é que os testes de redução se fortaleçam à medida que o *status* das facilidades forem sendo fixado, principalmente o *O-Teste* e o *C-Teste*, visto

que o sucesso de um influencia o sucesso do outro. O *O-RITeste* e *C-RITeste* são testes que tem um alto desempenho se o *PLCRI* fornecer um bom limite inferior, que é o caso dos problemas de Ardal [3]. Já para os problemas em que o *PLCRI* não fornece um bom limite inferior os testes têm um número baixo de decisões de fechar e ou abrir facilidades. O *O-RLTeste*, e o *C-RLTeste* tem um custo baixo, exceto pela resolução do *PLCRL* que é alto, e fornecem um bom desempenho geral.

É natural que o *O-RLTeste*, *C-RLTeste*, *O-RITeste* e *C-RITeste* tenham um desempenho similar se o gap entre o *PLCRI* e o *PLCRL* for pequeno, que é o caso dos problemas de Ardal [3]. O interessante é que o *O-Teste* e o *C-Teste* de certa forma são complementares aos testes: *O-RLTeste*, *C-RLTeste*, *O-RITeste* e *C-RITeste*. Ou seja, os testes atacam diferentes características dos problemas. Nos testes realizados para os problemas de Kühn & Hamburger [32] e Beasley [11] é fácil encontrar problemas onde só se obteve sucesso com a aplicação do *O-RLTeste* e *C-RLTeste* e problemas onde só se obteve sucesso com a aplicação do *O-Teste* e *C-Teste*.

Capítulo 4

4 Algoritmo Branch and Bound

O objetivo deste capítulo é o desenvolvimento de um algoritmo exato para a resolução do *PLC*. A idéia é fazer um *Branch and Bound* especializado para o problema, que forneça uma solução cujo valor seja próximo do valor ótimo nas primeiras buscas em profundidade e que consiga medir a qualidade da melhor solução encontrada conforme o método for realizando a exploração na árvore de busca. Assim, o método pode ser utilizado como uma heurística em casos onde o tempo requerido para encontrar a solução ótima seja maior que o tempo disponível para a tomada de decisão. Como o problema é altamente combinatório nunca se sabe qual o tempo requerido pelo algoritmo para encontrar a solução ótima. Assim sendo, o usuário pode utilizar o produto sabendo que pelo menos obterá uma solução próxima da ótima, e saberá o erro máximo gerado por esta. Procura-se fazer podas de forma eficiente. Tenta-se, além dessa, uma estratégia de busca na árvore, a qual faz uma combinação de busca em profundidade e em largura. O método pode ser utilizado como um algoritmo ε-ótimo, onde é possível informar qual a distância máxima que a solução fornecida pelo método pode ter em relação à solução ótima. Ou seja, um algoritmo onde é garantido que o erro máximo gerado seja conhecido.

Os testes de redução, *O-Teste* e *C-Teste*, abrem/fecham facilidades *a priori*, contudo para o bom desempenho necessitam da abertura/fechamento das facilidades *a priori*, pois para K_0 pequeno tem-se baixa sensibilidade no custo de transporte de $K_1 \cup K_2$ para i, ocasionado a falha do *O-Teste*, e para K_1 pequeno tem-se grande sensibilidade no custo de transporte de K_1 para i, ocasionando a falha do *C-Teste*. Isso pode ocasionar um ciclo vicioso, o qual pode ser quebrado utilizando um método *Branch and Bound*, conforme ilustra a Figura 7.

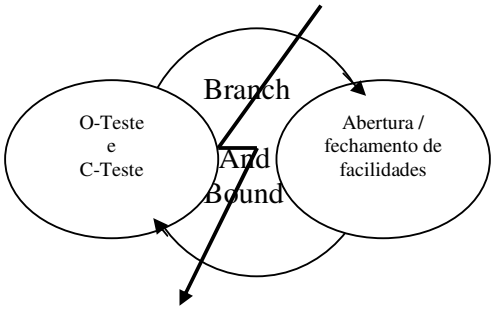


Figura 7 – Testes de redução de *branch and bound*

Levando em conta o fato do *PLC* ser altamente combinatório, é necessário que exista um processo de cortes e redução muito eficiente. Para atingir esse objetivo serão utilizados os testes de redução, vistos no Capítulo 2 e no Capítulo 3, limites inferiores, vistos no Capítulo 3, e número máximo e mínimo de facilidades a serem abertas. O método despenderá muito esforço no cálculo de limites inferiores, como visto no Capítulo 3, com objetivo de se encontrar bons resultados. Contudo, não basta a obtenção de bons limites inferiores para que esses levem à geração de cortes. É necessário também que se obtenha rapidamente um bom limite superior, o qual servirá de parâmetro para a realização de cortes através de limites inferiores. Caso o limite superior seja ruim os tempos gastos para se obter bons limites inferiores não servirão de nada. De qualquer forma, a obtenção de uma boa solução para o *PLC* é um dos requisitos do método que será desenvolvido, fazendo com que o mesmo também possa ser utilizado com um método heurístico. Para que esse objetivo seja alcançado serão utilizadas heurísticas como regras de ramificação e uma estratégia de busca na árvore que combina busca em profundidade com busca em largura.

4.1 Definições

Denominaremos de *PLC* o problema de localização capacitado originalmente dado. Ele é representado pela raiz da árvore de busca como será visto a seguir.

- **Nó:** representa uma alocação de elementos aos conjuntos K_0, K_1 e K_2. Por abuso de linguagem um nó será referenciado como um problema ou subproblema do problema original, visto que, na tentativa de fixar valores para os elementos do conjunto K_2 freqüentemente será resolvido um *PLC* associado a estas facilidades. Este subproblema é denominado de $PLC_{K0,K1,K2}$.

- **Árvore de Busca:** é um conjunto de nós que representa todas as combinações possíveis de elementos nos conjuntos K_0, K_1 e K_2. Então, seja n o tamanho de I, a árvore de busca terá um máximo de $2^{n+1}-1$ nós.

- **Folha:** uma folha é um nó cujo conjunto $K_2 = \varnothing$. Vale observar que se o $PLCRI_{K0,K1,K2}$ encontrar uma solução ótima do *PLC* é como se ele tivesse distribuído as facilidades de K_2 em K_0 e K_1.

- **Solução Viável:** Uma solução viável para o *PLC* é quando w(K_1) for viável. Uma solução viável fornece $K_0=I-K_1$, e o valor da solução é dada por $w(K_1) + \Sigma_{i \in K1} f_i$. No método uma solução viável é calculada em uma folha e quando o $PLCRI_{K0,K1,K2}$ fornecer uma solução viável para o $PLC_{K0,K1,K2}$.

- **Ramificação (*branching*):** ramificação na árvore é dada da seguinte maneira: Seja v um nó não folha da árvore de busca, K_0, K_1 e K_2 os conjuntos de facilidades associados a v, e i uma facilidade pertencente a K_2, então se podem ter duas ramificações conforme ilustrado na Figura 8: uma à esquerda que faz $K_0 = K_0 \cup \{i\}$ e $K_2 = K_2 - \{i\}$ e outra à direita que faz $K_1 = K_1 \cup \{i\}$ e $K_2 = K_2 - \{i\}$, gerando respectivamente os nós v' e v''. O que ocorre na ramificação é a divisão do problema em dois, onde em um deles considera-se uma determinada facilidade como sendo fechada e no outro como sendo aberta.

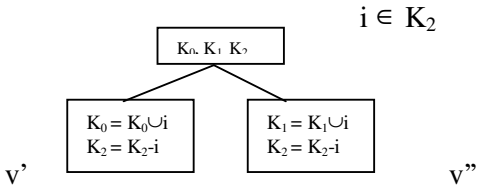


Figura 8 - ramificação

- **Pai e Filho:** Um nó v é pai de um nó u se u é uma ramificação a esquerda ou direita de v. Se v é pai de u então u é filho de v.

- **Raiz:** Um nó é raiz quando ele não possui pai.

- **Subir na árvore:** Subir na árvore é o ato de estar em um determinado nó *v* e ir para o pai de *v*. Não é possível subir se o nó for raiz.

- **Descer na árvore:** Descer na árvore é o ato de estar em um determinado nó *v* e ir para um de seus filhos. Não é possível descer caso *v* seja folha.

- **Nó Descendente:** Um nó *u* é descendente do nó *v* se subindo na árvore a partir de *u*, encontra-se *v*.

- **Soluções Descendentes:** Seja *v* um nó da árvore de busca, uma solução descendente de *v* é uma solução viável associada a um nó descendente de *v*.

- **Nó explorado:** um nó é considerado explorado quando todas suas soluções descendentes já foram encontradas de forma explícita ou implícita.

- **Limite Superior:** O limite superior, valor da função objetiva, do problema originalmente dado, será fornecido por qualquer solução viável deste. O melhor limite superior encontrado durante a exploração da árvore de busca é, dentre os valores viáveis encontrados, aquele de menor valor da função objetiva. Este será denotado por LS_{min}.

- **Limite Inferior:** O cálculo do limite inferior do *PLC*, denominado de *LI*, é um limite inferior do valor dda solução ótima do problema originalmente dado, ou seja, corresponde ao limite inferior associado à raiz da árvore. Seja *v* o nó raiz ao qual estão associados os conjuntos $K_0 = K_1 = \varnothing$ e $K_2 = I$. Uma das maneiras de calcular *LI* é fazer:

  $LI = max(PLCRI_{K0,K1,K2}, PLCRL_{K0,K1,K2})$

Sejam *v'* e *v"* os filhos de *v*, obtidos respectivamente, fazendo $K_0 = K_0 \cup \{i\}$, $K_2 = K_2 - \{i\}$ e $K_1 = K_1 \cup \{i\}$, $K_2 = K_2 - \{i\}$, e sejam K_0', K_1', K_2' e K_0'', K_1'', K_2'' os conjuntos associados aos nós. Suponha calculado $LI_{K0',K1',K2'} = max(PLCRI_{K0',K1',K2'}, PLCRL_{K0',K1',K2'})$ e $LI_{K0'',K1'',K2''} = max(PLCRI_{K0'',K1'',K2''}, PLCRL_{K0'',K1'',K2''})$. Então se $min(LI_{K0',K1',K2'}, LI_{K0'',K1'',K2''}) > LI$ é possível atualizar o limite inferior do PLC, fazendo:

$LI = min(LI_{K0';K1';K2'}, LI_{K0'';K1'';K2''})$

$LS_{min} - LI$ fornece o intervalo (*gap)* da solução ótima e será calculado para cada problema gerado (ver tabelas da seção resultados computacionais). Enquanto que o $LI_{K0,K1,K2}$ (limite de inferior para um nó *v* associado aos conjuntos K_0, K_1, e K_2) fornece um parâmetro para realização de um corte, isto é, se $LI_{K0,K1,K2} \geq LS_{min}$ então ocorre um corte.

4.2 Podas ou Cortes

A exploração completa da árvore de busca é inviável do ponto de vista computacional para problemas de grande porte. Supondo que o conjunto *I* tenha tamanho *50*, isso fornece $2^{51}-1$ nós na árvore de busca. Caso o tempo computacional para exploração de um nó fosse de *0.00001* segundos (um valor muito abaixo do que os computadores de mercado podem atingir), demorar-se-ia aproximadamente 7,2 séculos para explorar toda a árvore. Com isso, devem-se encontrar maneiras que explorem um número pequeno de nós, através podas ou cortes. O método desenvolvido fornece cinco maneiras de se realizar podas.

1. **Através de Limites Inferiores:** Seja *v* o nó a ser explorado, ao qual estão associados os conjuntos K_0, K_1, e K_2, e $LI_{K0,K1,K2}$ o valor do seu limite inferior, então se $LI_{K0,K1,K2} \geq LS_{max}$ se pode efetuar uma poda na árvore, pois não existem soluções descendentes de *v* melhores que a melhor solução encontrada até então. Isto indica que não é necessário explorar as soluções descendentes de *v*.
2. **Através de Testes de Redução:** é a principal contribuição dos testes de redução ao *Branch and Bound* na tentativa de explorar um número pequeno de nós. Testes de redução podem ser aplicados a qualquer nó não folha da árvore. As podas consistem na aplicação do *O-Teste, C-Teste, O-RITeste, C-RITeste, O-RLTeste* e *C-RLTeste*. Os testes alocam valores em K_0 e K_1, indicando que não há necessidade de considerar o valor alternativo. Com isto poupa-se uma ramificação. Isto é, se o Teste decide que $i \in K_1$ então evidentemente não é preciso considerar o ramo correspondente a $i \in K_0$.
3. **Através da solução do $PLCRI_{K0,K1,K2}$:** quando na solução do limite inferior $PLCRI_{K0,K1,K2}$ os valores de $y_i \ \forall i \in K_2$ forem *0* ou *1* não é necessário explorar as soluções formadas com as possíveis fixações de valores as facilidades pertencentes a K_2, pois os valores fixados a elas, são ótimos, isso porque as

restrições **(3-7)** foram atendidas, as quais foram relaxadas na obtenção do problema *PLCRI'*. Nesse caso, a solução do *PLCRI'* distribuí as facilidades de K_2 em K_0 e K_1.

4. **Através do número mínimo de facilidades abertas:** Seja $f\_min_{K0,K1,K2}$ o número mínimo de facilidades abertas na solução ótima de um nó v, então, se $|K_1 \cup K_2| = f\_min_{K0,K1,K2}$ todas as facilidades pertencentes a K_2 serão abertas, não necessitando, com isso, explorar as combinações que poderiam ser obtidas através da fixação (em zero ou um) das facilidades de K_2.
5. **Através do número máximo de facilidades abertas:** Seja $f\_max_{K0,K1,K2}$ o número máximo de facilidades abertas na solução ótima de um nó v, então se $|K_1| = f\_max_{K0,K1,K2}$ todas as facilidades pertencentes a K_2 serão fechadas, não necessitando, com isso, explorar as combinações que poderiam ser obtidas através da fixação (em zero ou um) das facilidades de K_2.

A Figura 9 mostra como funcionam as podas. A tesoura 1 representa uma poda gerada pelo sucesso de um teste de fechar, a tesoura 2 representa uma poda gerada pelo sucesso de um teste de abrir e a tesoura 3 representa uma poda gerado pelo uso de um dos limites inferiores ou pelo uso da quantidade mínima ou máxima de facilidades abertas na solução ótima. As linhas pontilhadas representam sub-árvores que devem ser exploradas. Note que a poda 1 e 2 são feitas somente em uma das sub-árvore, enquanto que as outras podas são realizadas nas duas sub-árvores semelhantemente.

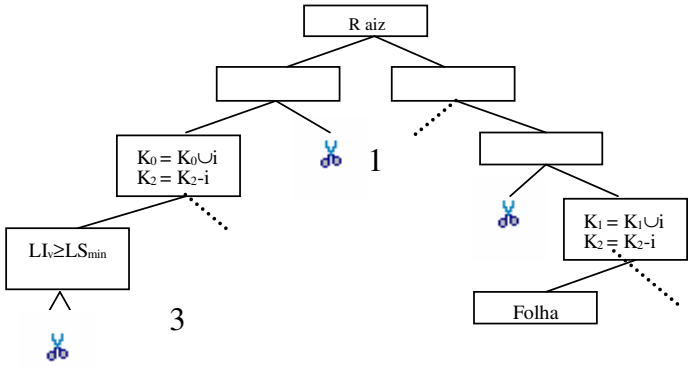


Figura 9 – Cortes na árvore de busca

4.3 Regras de Ramificação (Branching)

Ocorre uma ramificação em um nó v quando $K_2 \neq \varnothing$ e nenhuma poda puder ser realizada. Isso quer dizer que foi detectada pelo menos uma "*região de indecisão*" e é necessário decidir temporariamente o valor de alguma facilidade para. A questão aqui está em qual facilidade $i \in K_2$ escolher para alocar nos conjuntos K_0 e K_1 e qual ramificação deve ser explorada primeiro, ou seja, qual nó, v' ou v', $'$deve ser explorado primeiro. Uma boa escolha da facilidade a ser fixada e do lado da ramificação que será explorado primeiro pode aumentar as chances de sucesso dos testes além de propiciar a obtenção de novos/melhores limites superiores. Ambos os casos implicam em uma melhora de eficiência das podas, pois novos limites superiores aumentam as chances de podas através dos limites inferiores.

Antes de apresentar as regras de *branching* do método, será mostrado a *O-Heurística* e a *C-Heurística* apresentadas em Jacobsen [30].

- *O-Heurística*: tomar $i \mid min\ \Delta_i(K_1)$ fazendo $y_i = 1$.
- *C-Heurística*: tomar $i \mid max\ \Delta_i(K_1 \cup K_2 - i)$ fazendo $y_i = 0$.

As heurísticas são utilizadas para as facilidades $i \in K_2$ que não mais satisfazem as condições do *O-* e *C-Teste*. Neste caso tem-se $\Delta_i(K_1) < 0 < \Delta_i(K_1 \cup K_2 - i)$. A função $\Delta_i(K_1)$ representa o menor aumento de custo ao abrir i. Com isso, é natural que se tome i com o menor valor de $\Delta_i(K_1)$, $i \in K_2$, e se faça $y_i = 1$ (abrir a facilidade i). De outro modo, $\Delta_i(K_1 \cup K_2 - i)$ representa o maior aumento de custo ao abrir i. Com isso, é natural que se tome i com o maior valor de $\Delta_i(K_1)$, $i \in K_2$, e se faça $y_i = 0$ (fechar a facilidade i).

No método desenvolvido quando K_1 é viável criou-se uma heurística, para obtenção da facilidade a ser fixada (veja também Valiati [39]), que leva em conta tanto a *O-Heurística* como a *C-Heurística*, ou seja, procura-se fazer um balanço entre as duas heurísticas. A facilidade a ser fixada é a facilidade cujo índice fornece o valor da expressão $max_{i \in K2} \mid (\Delta_i(K_1 \cup K_2 - i) + \Delta_i(K_1)) \mid$ .Se $\Delta_i(K_1 \cup K_2 - i) + \Delta_i(K_1) > 0$ então se explora primeiro v', fazendo $K_0 = K_0 \cup \{i\}$ e $K_2 = K_2 - \{i\}$. Caso contrário $\Delta_i(K_1 \cup K_2 - i) + \Delta_i(K_1) \leq 0$ explora-se primeiro v'', fazendo $K_1 = K_1 \cup \{i\}$ e $K_2 = K_2 - \{i\}$. Esta heurística será referenciada por ***O/C-Heurística***.

A ***O/C-Heurística*** é um aprimoramento da *O-* e *C-Heurística* e foi desenvolvida para suprir a falha dessas. Na *O-Heurística* tomando-se o *min* $\Delta_i(K_1)$ não se pode garantir que $|\Delta_i(K_1)| < |\Delta_i(K_1 \cup K_2 - i)|$. Neste caso parece razoável que a facilidade *i* seja fechada em vez de ser aberta como a *O-Heurística* propõe. Seja $\Delta_p(K_1) = min_{i \in I} \Delta_i(K_1) = -5$ e $\Delta_p(K_1 \cup K_2 - p) = 25$ então, conforme mostra a Figura 10, a facilidade *p* está muito mais próxima de ser fechada do que ser aberta. Se, no entanto, fosse aplicada a *O-Heurística* a facilidade *p* seria aberta. Na *C-Heurística* pode ocorre uma falha parecida. Quando se toma o *max* $\Delta_i(K_1 \cup K_2 - i)$ pode ser que $|\Delta_i(K_1 \cup K_2 - i)| < |\Delta_i(K_1)|$, indicando que é melhor a facilidade *i* ser aberta do que fechada conforme estabelece a *C-Heurística*. Seja $\Delta_p(K_1 \cup K_2 - p) = max_{i \in I} \Delta_i(K_1 \cup K_2 - i) = 5$ e $\Delta_p(K_1) = -25$ então, conforme mostra a Figura 11, a facilidade *p* está muito mais próxima de ser aberta do que fechada. Se fosse aplicada a *C-Heurística* a facilidade *p*, no entanto, seria fechada. Usando a ***O/C-Heurística*** procura-se a facilidade *i* com maior diferença em valor absoluto entre $|\Delta_i(K_1 \cup K_2 - i)|$ e $|\Delta_i(K_1)|$, pois é esta a facilidade que tem mais claramente definida a sua tendência a ser aberta/fechada. Caso para esta facilidade $|\Delta_i(K_1 \cup K_2 - i)| > |\Delta_i(K_1)|$ então a facilidade *i* está mais próxima da região em que as facilidades são fechadas, logo a prioridade é considerar o fechamento de *i*. Caso contrário, $|\Delta_i(K_1 \cup K_2 - i)| < |\Delta_i(K_1)|$ então a prioridade é para abertura de *i*. Cabe lembrar que se está em uma situação em que os *O-* e *C-Testes* não se aplicam mais, isto é, $|\Delta_i(K_1)| < 0 < |\Delta_i(K_1 \cup K_2 - i)|$, ou seja, não mais se está na região de abertura/fechamento, e ,portanto, o que se faz é testar a maior proximidade a uma destas regiões. Por exemplo, na Figura 12 a facilidade *1* fornece o valor de *max* $\Delta_i(K_1 \cup K_2 - i)$ e a facilidade *2* o *min* $\Delta_i(K_1)$. Contudo é a facilidade *3* a mais propícia a ser fechada, pois é a facilidade *3* que tem mais claramente definido o balanço entre as tendências a ser fechada e aberta. No caso, o balanço tende para o fechamento da facilidade *3*.

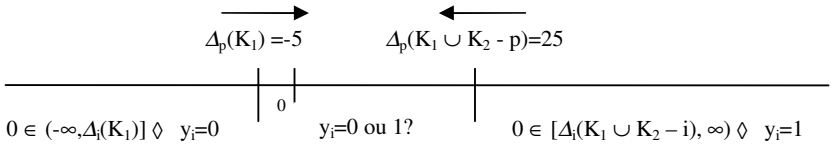


Figura 10 – falha na *O-Heurística*

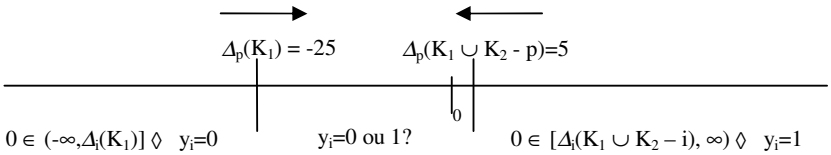


Figura 11 - *C-Heurística*

Pode ser fornecida uma interpretação para a ***O/C-Heurística*** que justifica a sua aplicação de forma mais precisa. $\delta_i(K_1 \cup K_2 - i)$ representa o mínimo ganho(sensibilidade) dos custos de transporte e $\delta_i(K_1)$ o máximo ganho(sensibilidade) se a facilidade *i* for aberta. Quando o custo fixo for muito pequeno, ou seja, menor que o mínimo ganho dos custos de transporte, a facilidade será aberta (*O-Teste*). Neste caso o custo fixo fica à esquerda de $\delta_i(K_1 \cup K_2 - i)$, conforme mostra a Figura 13. Quando o custo fixo for maior que o máximo ganho dos custos de transporte a facilidade será fechada (*C-Teste*). Neste caso o custo fixo fica à direita de $\delta_i(K_1)$, conforme mostra a Figura 13. Quando o custo fixo ficar entre $\delta_i(K_1 \cup K_2 - i)$ e $\delta_i(K_1)$ nada se pode garantir. É neste caso que se aplica a ***O/C-Heurística***. Caso f_i esteja mais próximo de $\delta_i(K_1 \cup K_2 - i)$ do que $\delta_i(K_1)$ abre-se a facilidade *i*. Isso porque o custo fixo se aproxima do mínimo ganho. Na solução ótima, provavelmente, o ganho real irá superar o custo fixo. Quando f_i está mais próximo de $\delta_i(K_1)$ do que $\delta_i(K_1 \cup K_2 - i)$ a facilidade *i* é fechada. Isso porque que o custo fixo se aproxima do máximo ganho dos custos de transporte, ou seja, existe pouca chance da facilidade *i* contribuir para a redução global dos custos. Na solução ótima, provavelmente, o ganho real nos custos de transporte irá ser superado pelo custo fixo.

Quando K_1 é inviável é impossível calcular $\Delta_i(K_1)$ e se tem somente os dados da *C-Heurística*. Neste caso, foram desenvolvidas duas heurísticas (veja também Valiati [39]). Já que somente $\Delta_i(K_1 \cup K_2 - i)$ é considerado, para relativar está medida, levando em conta diversas facilidades *i* com diversas capacidades a_i, as medidas são obtidas por unidade de capacidade. A primeira heurística consiste em fechar uma facilidade $i \in K_2$ cujo índice fornece o valor da expressão $max_{i \in K2}\ (\Delta_i(K_1 \cup K_2 - i)/a_i)$. A expressão fornece entre as facilidades indefinidas aquela que deve ser fechada, indicando que se deve explorar primeiro *v'*, fazendo $K_0 = K_0 \cup \{i\}$ e $K_2 = K_2 - \{i\}$. Esta heurística será

referenciada por ***CMax-Heurística***. A segunda heurística consiste em abrir uma facilidade $i \in K_2$ cujo índice fornece o valor da expressão $min_{i \in K2}\ (\Delta_i(K_1 \cup K_2 - i)/a_i)$. A expressão fornece entre as facilidades indefinidas aquela que deve ser aberta, indicando que se deve explorar primeiro v', fazendo $K_1 = K_1 \cup \{i\}$ e $K_2 = K_2 - \{i\}$. Esta heurística será referenciada por ***CMin-Heurística***.

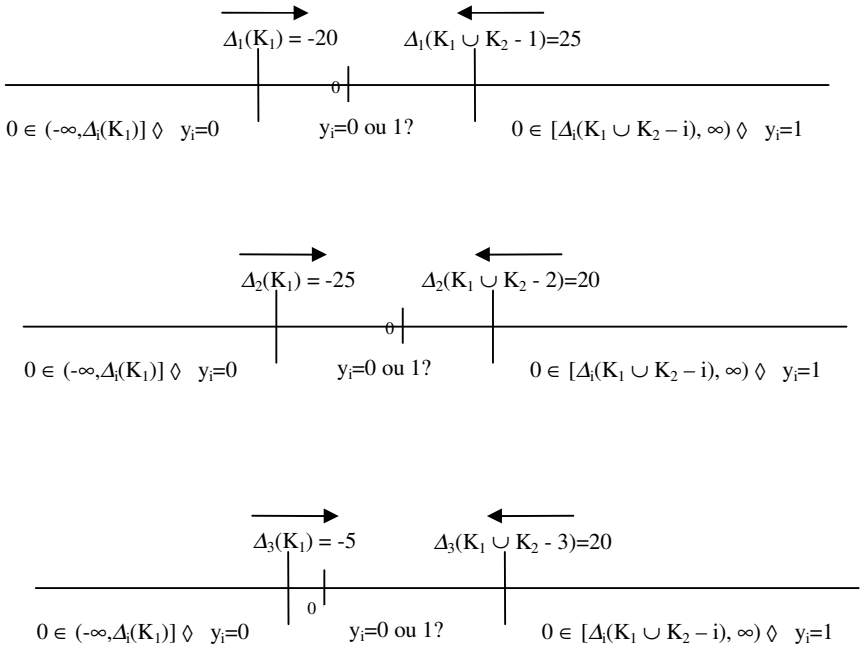


Figura 12 – Aplicação do O/*C-Heurística*

CMax-Heurística e ***CMin-Heurística*** fornecem respectivamente uma maneira de fechar e abrir uma determinada facilidade quando K_1 for inviável. ***CMin-Heurística*** procura a facilidade mais próxima da região de abertura, enquanto que ***CMax-Heurística*** a que está mais distante da região de abertura. Estas heurísticas podem ser muito falhas, isso porque como a região de fechamento não está definida a noção de perto ou longe perde um pouco o sentido. Por exemplo, uma facilidade que está longe da "região de abertura" poderia estar mais longe ainda da "região de fechamento" (quando esta vier a ser definida). Tal facilidade deveria, portanto ser aberta e não fechada. Por outro lado uma facilidade pode parecer estar perto da região de abertura e, no entanto, estar longe se tomarmos como referencial a sua proximidade da região de fechamento.

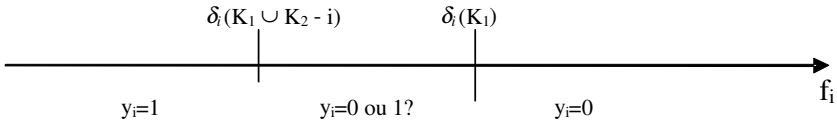


Figura 13 – outra interpretação para *O/C-Heurística*

Segue-se outro tipo de interpretação. Tem-se que $\delta_i(K_1 \cup K_2 - i)$ representa o mínimo ganho quando a facilidade *i* é aberta. Quando o custo fixo de *i* é menor que o mínimo ganho a facilidade *i* será aberta (*O-Teste*). Por outro lado, quando o custo fixo for maior que o máximo ganho $\delta_i(K_1)$ a facilidade será fechada (*C-Teste*). Quando, no entanto, o custo fixo for maior que o mínimo ganho, mas menor que o máximo ganho nada se pode garantir sobre abertura/fechamento de *i*. Neste caso aplicam-se as heurísticas. Quando o custo fixo de *i* está próximo do mínimo ganho abre-se *i* (*CMin-Heurística*). Nesta condição está implícita a suposição que o máximo ganho está distante do custo fixo. Com isso, provavelmente o ganho real abrindo-se *i* irá superar o custo fixo. Quando o custo fixo de *i* está distante do mínimo ganho fecha-se *i* (*CMax-Heurística*). Nesta condição está implícita a suposição que o máximo ganho está próximo do custo fixo. Assim abrindo-se *i* provavelmente o ganho real nos custos de transporte irá ser superado pelo custo fixo. A Figura 14 mostra através das linhas pontilhadas as chances da facilidade *i* ser aberta. Quanto mais próximo o custo fixo estiver do mínimo ganho maiores as chances de *i* ser aberta. Contudo, como já dito anteriormente, pode ser que alguma coisa que pareça perto na verdade esteja longe se for considerado o máximo ganho, que aqui apenas se está supondo.

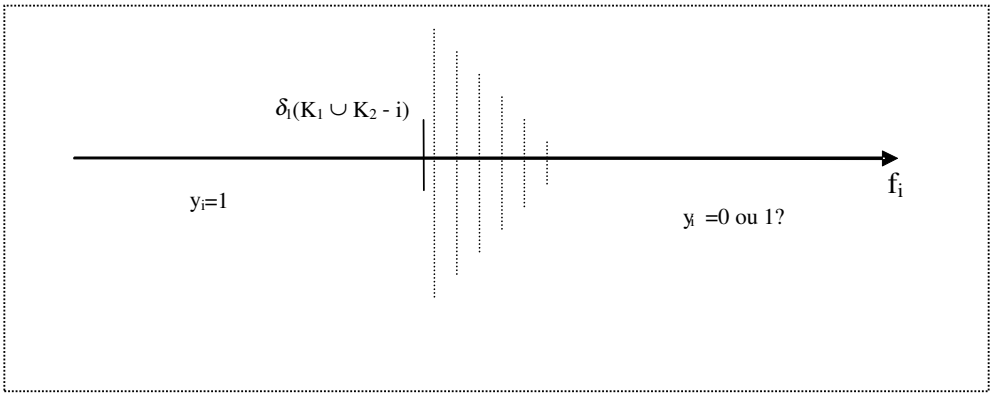


Figura 14 – Possibilidade da facilidade *i* ser aberta

Para um determinado nó *v* para o qual se tem K_0, K_1 e K_2, suponha calculado o $PLCRI_{K0,K1,K2}$ a partir da relaxação das condições de integralidade. Seja y* a solução obtida para o $PLCRI_{K0,K1,K2}$. A ramificação do nó *v* em *v'* e *v"* será obtida a partir das variáveis $0 < y_i^* < 1$, ou seja, tomando uma das variáveis deste tipo e fixando-a em zero para *v'* e em um para a geração de *v"*. As variáveis $y_i^* = 0$ ou $y_i^* = 1$ não são consideradas para efeito de ramificação. A questão consiste em determinar qual das variáveis $y_i \in (0,1)$ será escolhida. Para isto, nos valemos da *O/C-Heurística*, *CMin-Heurística* e da *CMax-Heurística*. Quando K_1 é viável utilizamos a *O/C-Heurística*. Em caso contrário, utilizamos a *CMin-Heurística* e a *CMax-Heurística*. Examinemos cada um dos três casos:

a) Utilização da *O/C-Heurística*

Seja $\Delta_i^{O/C}(v) = \left|\Delta_i(K_1 \cup K_2 - i) + \Delta_i(K_1)\right|$. Pela *O/C-Heurística*, deve-se escolher, para efeito de ramificação, a variável y_p, tal que, $\Delta_p^{O/C}(v) = \max(\Delta_i^{O/C}(v))$. Isto, no entanto, não leva em conta os resultados obtidos pela relaxação Lagrangeana ao determinarmos o outro limite inferior, o $PLCRL_{K0,K1,K2}$. Seja $\hat{y}$ a solução obtida para o $PLCRL_{K0,K1,K2}$. Como foi visto, $\hat{y}_i \in \{0,1\}$. Seja $\hat{y}_p = 1$, este fato indica que a solução obtida por relaxação Lagrangeana "recomenda" a abertura da facilidade p. O mesmo acontece para $\hat{y}_q = 0$, onde a relaxação Lagrangeana "recomenda" o fechamento da facilidade *q*. Ora, é justo que estas "recomendações" recebam um peso na decisão de qual facilidade deve ser ramificada. Com isso, a *O/C-Heurística* receberá um peso, conforme recomendação fornecida pelo resultado do $PLCRL_{K0,K1,K2}$. O peso, para uma facilidade *i*, foi definido, através de experimentos, da seguinte forma:

(4-1) $\quad peso_i = \Delta_i^{O/C}(v) * 0{,}2$

,ou seja, o peso, de uma facilidade *i*, representa 20% do valor $\Delta_i^{O/C}(v)$.

Definido o peso que as recomendações da relaxação Lagrangeana terão, é necessário aplicá-lo a *O/C-Heurística*. Supondo que a recomendação do $PLCRL_{K0,K1,K2}$, para a facilidade *i*, seja de abrir, isto é, $\hat{y}_i = 1$. Caso a $\Delta_i^{O/C}(v)$ indique, também, a abertura de *i*, ou seja, $|\Delta_i(K_1 \cup K_2 - i)| < |\Delta_i(K_1)|$, o peso será somado ao valor de $\Delta_i^{O/C}(v)$. Caso contrário, o peso será subtraído do valor de $\Delta_i^{O/C}(v)$. Da mesma forma, supondo que a recomendação do $PLCRL_{K0,K1,K2}$, para a facilidade *i*, seja de fechar, isto

é, $\hat{y}_i = 0$. Caso a $\Delta_i^{O/C}(v)$ indique, também, o fechamento de *i*, ou seja, $|\Delta_i(K_1 \cup K_2 - i)| > |\Delta_i(K_1)|$, o peso será somado ao valor de $\Delta_i^{O/C}(v)$. Caso contrário, o peso será subtraído do valor de $\Delta_i^{O/C}(v)$. Seja $K_2' = \{i \in K_2 \mid 0 < y_i^* < 1$, onde y* representa a solução do $PLCRI_{K0,K1,K2}\}$, então o Algoritmo 12, denominado *O/C-HeurísticaLI*, incluem os resultados dos limites inferiores, conforme descrito.

Algoritmo 12: O/C-HeurísticaLI:

$\forall$ i $\in K_2'$ ***Fazer***

Inicio

$\Delta_i^{O/C}(v) \leftarrow \left|\Delta_i(K_1 \cup K_2 - i) + \Delta_i(K_1)\right|$

Se $\hat{y}_i = 1$ ***então***

Se $|\Delta_i(K_1 \cup K_2 - i)| < |\Delta_i(K_1)|$ ***então***

$\Delta_i^{O/C}(v) = \Delta_i^{O/C}(v) + peso_i$

Senão

$\Delta_i^{O/C}(v) = \Delta_i^{O/C}(v) - peso_i$

Senão $\{\hat{y}_i = 0\}$

Se $|\Delta_i(K_1 \cup K_2 - i)| < |\Delta_i(K_1)|$ ***então***

$\Delta_i^{O/C}(v) = \Delta_i^{O/C}(v) - peso_i$

Senão

$\Delta_i^{O/C}(v) = \Delta_i^{O/C}(v) + peso_i$

Fim

Seja *p* a facilidade, tal que, $max_{i \in K2'} \Delta_i^{O/C}(v)$

Se $|\Delta_p(K_1 \cup K_2 - p)| > |\Delta_p(K_1)|$ ***então***

Fechar facilidade *p*

Senão

Abrir facilidade *p*

----- Fim Algoritmo 12 -----

Para efeito de simplificação, $max_{i \in K2'} \Delta_i^{O/C}(v)$ será denominado, simplesmente, por $\Delta^{O/C}(v)$.

b) Utilização da *CMin-Heurística*

Seja $\Delta_i^{CMin}(v) = \frac{\Delta_i(K_1 \cup K_2 - i)}{a_i}$. Pela *CMin-Heurística*, deve-se escolher, para efeito de ramificação, a variável y_p, tal que, $\Delta_p^{CMin}(v) = \min(\Delta_i^{CMin}(v))$. Isto, da mesma forma que na *O/C-Heurística*, não leva em conta os resultados obtidos pela relaxação Lagrangeana ao determinarmos o $PLCRL_{K0,K1,K2}$. Seja $\hat{y}$ a solução obtida para o $PLCRL_{K0,K1,K2}$. Como foi visto, $\hat{y}_i \in \{0,1\}$. Seja $\hat{y}_p = 1$, este fato indica que a solução obtida por relaxação Lagrangeana "recomenda" a abertura da facilidade p. O mesmo acontece para $\hat{y}_q = 0$, onde a relaxação Lagrangeana "recomenda" o fechamento da facilidade q. Ora, é justo que estas "recomendações" recebam um peso na decisão de qual facilidade deve ser ramificada. Com isso, a *CMin-Heurística* receberá um peso, conforme recomendação fornecida pelo resultado do $PLCRL_{K0,K1,K2}$. O peso, para uma facilidade i, foi definido, através de experimentos, da seguinte forma:

(4-2) $\quad peso_i = \Delta_i^{CMin}(v) * 0{,}2$

,ou seja, o peso, de uma facilidade i, representa 20% do valor $\Delta_i^{CMin}(v)$.

Definido o peso que as recomendações da relaxação Lagrangeana terão, é necessário aplicá-lo à *CMin-Heurística*. Supondo que a recomendação do $PLCRL_{K0,K1,K2}$, para a facilidade i, seja de abrir, isto é, $\hat{y}_i = 1$. Diferentemente da *O/C-Heurística*, a *CMin-Heurística* somente considera a abertura de uma facilidade, a qual é indicada pelo menor valor de $\Delta_i^{CMin}(v)$. Então, quando $\hat{y}_i = 1$, o peso será subtraído de $\Delta_i^{CMin}(v)$. Da mesma forma, supondo que a recomendação do $PLCRL_{K0,K1,K2}$, para a facilidade i, seja de fechar, isto é, $\hat{y}_i = 0$, o peso será somado a $\Delta_i^{CMin}(v)$. Seja $K_2' = \{i \in K_2 \mid 0 < y_i^* < 1$, onde y* representa a solução do $PLCRI_{K0,K1,K2}\}$, então o Algoritmo 13, denominado *CMin-HeurísticaLI*, incluem os resultados dos limites inferiores, conforme descrito.

Algoritmo 13: CMin-HeurísticaLI:

$\forall$ i $\in K_2^{'}$ ***Fazer***

Inicio

$$\Delta_i^{\text{CMin}}(v) \leftarrow \frac{\Delta_i(K_1 \cup K_2 - i)}{a_i}$$

Se $\hat{y}_i = 1$ ***então***

$$\Delta_i^{\text{CMin}}(v) = \Delta_i^{\text{CMin}}(v) - peso_i$$

Senão $\{\hat{y}_i = 0\}$

$$\Delta_i^{\text{CMin}}(v) = \Delta_i^{\text{CMin}}(v) + peso_i$$

Fim

Seja p a facilidade, tal que, $min_{i \in K2'}\ \Delta_i^{\text{CMin}}(v)$

Abrir facilidade p

----- Fim Algoritmo 13 -----

Para efeito de simplificação, $min_{i \in K2'}\ \Delta_i^{\text{CMin}}(v)$ será denominado, simplesmente, por $\Delta^{\text{CMin}}(v)$.

c) Utilização da *CMax-Heurística*

Seja $\Delta_i^{\text{CMax}}(v) = \frac{\Delta_i(K_1 \cup K_2 - i)}{a_i}$. Pela *CMax-Heurística*, deve-se escolher, para efeito de ramificação, a variável y_p, tal que, $\Delta_p^{\text{CMax}}(v) = \min(\Delta_i^{\text{CMax}}(v))$. Isto, da mesma forma que na *O/C-Heurística*, não leva em conta os resultados obtidos pela relaxação Lagrangeana ao determinarmos o $PLCRL_{K0,K1,K2}$. Seja $\hat{y}$ a solução obtida para o $PLCRL_{K0,K1,K2}$. Como foi visto, $\hat{y}_i \in \{0,1\}$. Seja $\hat{y}_p = 1$, este fato indica que a solução obtida por relaxação Lagrangeana "recomenda" a abertura da facilidade p. O mesmo acontece para $\hat{y}_q = 0$, onde a relaxação Lagrangeana "recomenda" o fechamento da facilidade q. Ora, é justo que estas "recomendações" recebam um peso na decisão de qual facilidade deve ser ramificada. Com isso, a *CMax-Heurística* receberá um peso, conforme recomendação fornecida pelo resultado do $PLCRL_{K0,K1,K2}$. O peso, para uma facilidade i, foi definido, através de experimentos, da seguinte forma:

(4-3) $$peso_i = \Delta_i^{\text{CMax}}(v) * 0{,}2$$

,ou seja, o peso, de uma facilidade i, representa 20% do valor $\Delta_i^{\text{CMax}}(v)$.

Definido o peso que as recomendações da relaxação Lagrangeana terão, é necessário aplicá-lo à *CMax-Heurística*. Supondo que a recomendação do $PLCRL_{K0,K1,K2}$, para a facilidade *i*, seja de abrir, isto é, $\hat{y}_i = 1$. Diferentemente da *O/C-Heurística*, a *CMax-Heurística* somente considera o fechamento de uma facilidade, o qual é indicado pelo maior valor de $\Delta_i^{CMax}(v)$. Então, quando $\hat{y}_i = 1$, o peso será subtraído de $\Delta_i^{CMax}(v)$. Da mesma forma, supondo que a recomendação do $PLCRL_{K0,K1,K2}$, para a facilidade *i*, seja de fechar, isto é, $\hat{y}_i = 0$, o peso será somado a $\Delta_i^{CMax}(v)$. Seja $K_2' = \{i \in K_2 \mid 0 < y_i^* < 1$, onde y* representa a solução do $PLCRI_{K0,K1,K2}\}$, então o Algoritmo 14, denominado *CMax-HeurísticaLI*, incluem os resultados dos limites inferiores, conforme descrito.

Algoritmo 14: CMax-HeurísticaLI:

$\forall$ i $\in K_2'$ ***Fazer***

Inicio

$$\Delta_i^{CMax}(v) \leftarrow \frac{\Delta_i(K_1 \cup K_2 - i)}{a_i}$$

Se $\hat{y}_i = 1$ ***então***

$$\Delta_i^{CMax}(v) = \Delta_i^{CMax}(v) \text{ - } peso_i$$

Senão $\{\hat{y}_i = 0\}$

$$\Delta_i^{CMax}(v) = \Delta_i^{CMax}(v) + peso_i$$

Fim

Seja *p* a facilidade, tal que, $max_{i \in K2'} \Delta_i^{CMax}(v)$

Fechar facilidade *p*

----- Fim Algoritmo 14 -----

Para efeito de simplificação, $max_{i \in K2'} \Delta_i^{CMax}(v)$ será denominado, simplesmente, por $\Delta^{CMax}(v)$.

Embora a *CMax-HeurísticaLI* tenha sido desenvolvida, não a utilizamos como regra de ramificação, no algoritmo. Quando K_1 for viável utilizamos a *O/C-Heurística*, caso contrário, a *CMin-HeurísticaLI* é aplicada. A escolha da *CMin-HeurísticaLI*, em

detrimento da *CMax-HeurísticaLI*, foi devido ao fato desta tornar K_1 viável mais rapidamente. Com K_1 viável o C-Teste e a *O/C-Heurística* podem ser aplicados.

4.4 Obtenção do número mínimo e máximo de facilidades

Não são fornecidos, nos dados do problema, o número mínimo e máximo de facilidades abertas para o cálculo do *PLC*. Esses números são utilizados como critério de podas. Assim sendo, é necessário detectá-los. Na seqüência, serão apresentadas, maneiras para se obter o número mínimo (*f_min*) e máximo (*f_max*) de facilidades abertas na solução ótima.

4.4.1 Determinação de um número máximo, *f_max,* de facilidades abertas na solução ótima do *PLC*:

Seja *LS* o valor de uma solução viável do *PLC*, ou seja, o limite superior do *PLC*. Sabendo que *w(I)* representa o menor custo variável que é possível se obter para o *PLC* dado, então se deve tomar o máximo de facilidades que somando seus custos fixos a *w(I)* forneça um valor menor do que *LS*. Então o número máximo de facilidades abertas, na solução ótima, é dado pelo seguinte modelo:

(4-4) $$\max \sum_{i \in I} y_i$$

S.a

(4-5) $$w(I) + \sum_{i \in I} f_i y_i < LS$$

(4-6) $$y_i \in \{0,1\}, \quad \forall i \in I$$

Se as facilidades não tiverem o mesmo custo fixo, para solucionar o modelo, há que ordená-las crescentemente. Seja uma ordenação das facilidades, tal que, $f_1 \bullet f_2 \bullet ... \bullet f_n$, então, *f_max* será o maior valor de *k*, tal que, $w(I) + \sum_{i=1}^{k} f_i < LS$.

4.4.2 Determinação do número máximo, $f\_max_{K0,K1,K2}$, de facilidades abertas em um nó *v*

A diferença com relação à determinação de *f_max* é que se devem considerar as facilidades em K_0, K_1 e K_2. Seja *LS* um limite superior do *PLC* então o número máximo de facilidades abertas para soluções descendentes de *v*, é dado pelo seguinte modelo:

(4-7) $\max \sum_{i \in K_1 \cup K_2} y_i$

S.a

(4-8) $w(K_1 \cup K_2) + \sum_{i \in K_1 \cup K_2} f_i y_i < LS$

(4-9) $y_i = 1, \quad \forall i \in K_1$

(4-10) $y_i \in \{0,1\}, \quad \forall i \in K_2$

Se as facilidades, pertencentes a K_2, não tiverem o mesmo custo fixo, para solucionar o modelo, há que ordená-las crescentemente. Seja uma ordenação das facilidades, tal que, $f_1 \bullet f_2 \bullet \ldots \bullet f_n$, então, $f\_max_{K0,K1,K2}$ será dado por $k+|K_1|$, onde k é o maior valor, tal que, $w(I) + \sum_{i \in K_1} f_i + \sum_{i=1}^{k} f_i < LS$.

4.4.3 Determinação do número mínimo, *f_min,* de facilidades que devem ser abertas na solução ótima do *PLC*

Seria o menor número de facilidades que atenderia toda a demanda. Para isso, basta tomar as facilidades com as maiores capacidades que atendam a demanda. Então o número mínimo de facilidades abertas é dado pelo seguinte modelo:

(4-11) $\min \sum_{i \in I} y_i$

S.a

(4-12) $\sum_{i \in I} a_i y_i \geq \sum_{j \in J} b_j$

(4-13) $y_i \in \{0,1\}, \quad \forall i \in I$

Se as facilidades não tiverem a mesma capacidade, para solucionar o modelo, há que ordená-las decrescentemente. Seja uma ordenação das facilidades, tal que, $a_1 \bullet a_2 \bullet \ldots \bullet a_n$, então, *f_min* será o menor valor de k, tal que, $\sum_{i=1}^{k} a_i \geq \sum_{j \in J} b_j$.

4.4.4 Determinação do número mínimo, $f\_min_{K0,K1,K2}$, de facilidades que devem ser abertas em um nó *v*

A diferença com relação à determinação do número mínimo de facilidades que devem ser abertas na solução ótima é que se devem considerar as facilidades em K_0, K_1

e K_2. Então, o número mínimo de facilidades abertas para soluções descendentes de *v*, é dado pelo seguinte modelo:

(4-14) $\min \sum_{i \in K_1 \cup K_2} y_i$

S.a

(4-15) $\sum_{i \in K_1 \cup K_2} a_i y_i \geq \sum_{j \in J} b_j$

(4-16) $y_i = 1, \quad \forall i \in K_1$

(4-17) $y_i \in \{0,1\}, \quad \forall i \in K_2$

Se as facilidades, pertencentes a K_2, não tiverem a mesma capacidade, para solucionar o modelo, há que ordená-las decrescentemente. Seja uma ordenação das facilidades, tal que, $a_1 \bullet a_2 \bullet ... \bullet a_n$, então, $f\_min_{K0,K1,K2}$ será dado por $k+|K_1|$, onde k é o menor valor, tal que, $\sum_{i \in K_1} a_i + \sum_{i=1}^{k} a_i \geq \sum_{j \in J} b_j$.

4.4.5 Determinação do número mínimo, $f\_min_{K0,K1,K2}$, de facilidades que devem ser abertas em um nó *v* quando $w(K_1)$ viável

A partir do momento em que $w(K_1)$ passa a ser viável, tem-se outra forma para obtenção do número mínimo de facilidades a serem abertas em um nó *v*. Sabe-se que $\Delta_i(K_1)$ representa a maior vantagem, com relação aos custos, que a facilidade $i \in K_2$ pode apresentar quando aberta. Seja *LS*, um limite superior do *PLC*, e $s = w(K_1) + \sum_{i \in K_1} f_i$, então, o número mínimo de facilidades abertas para soluções descendentes de *v*, é dado pelo seguinte modelo:

(4-18) $\min \sum_{i \in K_1 \cup K_2} y_i$

S.a

(4-19) $s + \sum_{i \in K_2} \Delta_i(K_1) y_i < LS$

(4-20) $y_i = 1, \quad \forall i \in K_1$

(4-21) $y_i \in \{0,1\}, \quad \forall i \in K_2$

Para solucionar o modelo, há que se ordenar $\Delta_i(K_1)$ crescentemente. Seja uma ordenação das facilidades, pertencentes a K_2, tal que, $\Delta_1(K_1) \bullet \Delta_2(K_1) \bullet ... \bullet \Delta_n(K_1)$,

então, $f\_min_{K0,K1,K2}$ será dado por $k+|K_1|$, onde k é o menor valor, tal que, $s + \sum_{i=1}^{k} \Delta_i(K_1) < LS$.

Cabe também lembrar que a situação em que esta poda é aplicada é, tal que, nem o *O-Teste* nem o *C-Teste* podem ser aplicados. Neste caso, tem-se $\Delta_i(K_1) < 0$.

Os números mínimo e máximo de facilidades abertas, obtidos aqui, alimentam o problema *PLCRL* da seção 3.3.

4.5 Subida na árvore (Backtracking) e Condições de Parada

A subida na árvore (*Backtracking)* em um nó v é feito quando não existe mais necessidade de continuar explorando as soluções descendentes de v. Na exploração da árvore de busca pode ocorrer *Backtracking* em um nó v em 3 situações:

1. v é uma folha, ou seja, quando o conjunto $K_2 = \varnothing$, indicando que não existem mais soluções descendentes de v para serem exploradas.
2. $LI_v \geq LS_{min}$, ou seja, não é mais necessário explorar as soluções descendentes de v, porque elas fornecem um valor maior ou igual ao melhor valor encontrado.
3. Todos os nós descendentes de v já foram explorados, ou seja, quando todas as soluções descendentes de v já foram encontradas de forma explícita (folhas) ou implícita (podas).

A subida na árvore é feita até que seja encontrada uma sub-árvore que ainda não foi explorada, ou seja, encontrar um subproblema que ainda não foi resolvido. A condição de parada do método é quando a subida levar até o nó raiz da árvore. Neste caso não existe mais nenhum subproblema a ser resolvido, ou seja, todas as soluções descendentes do problema original foram encontradas de forma explícita ou implícita.

4.6 Estratégia de busca na árvore

A busca na árvore será realizada utilizando-se uma combinação de busca em profundidade e busca em largura. A busca em profundidade é o ato de descer de um nó v até um nó que forneça uma condição de *backtracking*. A busca em profundidade é necessária porque, no método, as soluções viáveis são encontradas principalmente em nós folhas. Contudo, a estratégia de se optar somente por busca em profundidade pode levar o algoritmo a dispender muito tempo em ótimos locais. Isso porque, a busca em

profundidade, após a escolha de uma das sub-árvores, realiza sua exploração por completo, fazendo com que o método só tome uma sub-árvore, em um nível menor, depois de ter encontrado todas as soluções, de forma implícita ou explícita, referentes à sub-árvore escolhida.

A Figura 15 ilustra um exemplo em que a busca em profundidade pode não funcionar da melhor forma possível. Supondo que em uma ramificação se opte por descer, primeiro, por um lado que na verdade não contém a solução ótima (o que evidentemente é desconhecido neste instante). Na Figura 15, os nós pretos representam os nós explorados na primeira busca em profundidade e as linhas tracejadas representam sub-árvores que precisam ser exploradas. Suponha que no segundo nó (indicado pela seta na Figura 15) foi descido pelo lado que na verdade não contém a solução ótima, isto é, optou-se pelo dado direito, porém a solução ótima estaria no lado esquerdo. Se a busca for realizada, exclusivamente, em profundidade, antes serão explorados todos os nós da sub-árvore à direita, o que pode não ser interessante.

Outra consideração importante, é a tendência de soluções geradas por folhas serem próximas, como é o caso dos nós *hachurados* na Figura 15, serem muito semelhantes. O ideal seria somente explorar todas as folhas próximas caso a solução ótima esteja próxima, caso contrário, deve-se explorar uma folha e ir para outra que esteja bem distante. Essa é a idéia da combinação entre busca e profundidade e busca em largura. Inicialmente, o método procura explorar as diversas partes da árvore, para depois procurar refinar a solução em buscas locais, mas somente nos lugares que se mostraram atraentes. Com isso, é possível encontrar a solução ótima mais rapidamente, pois é possível reverter um suposto erro, de escolha de lado. Assim sendo, é possível realizar mais podas com os limites inferiores, pois os mesmos necessitam de limites superiores cada vez melhores.

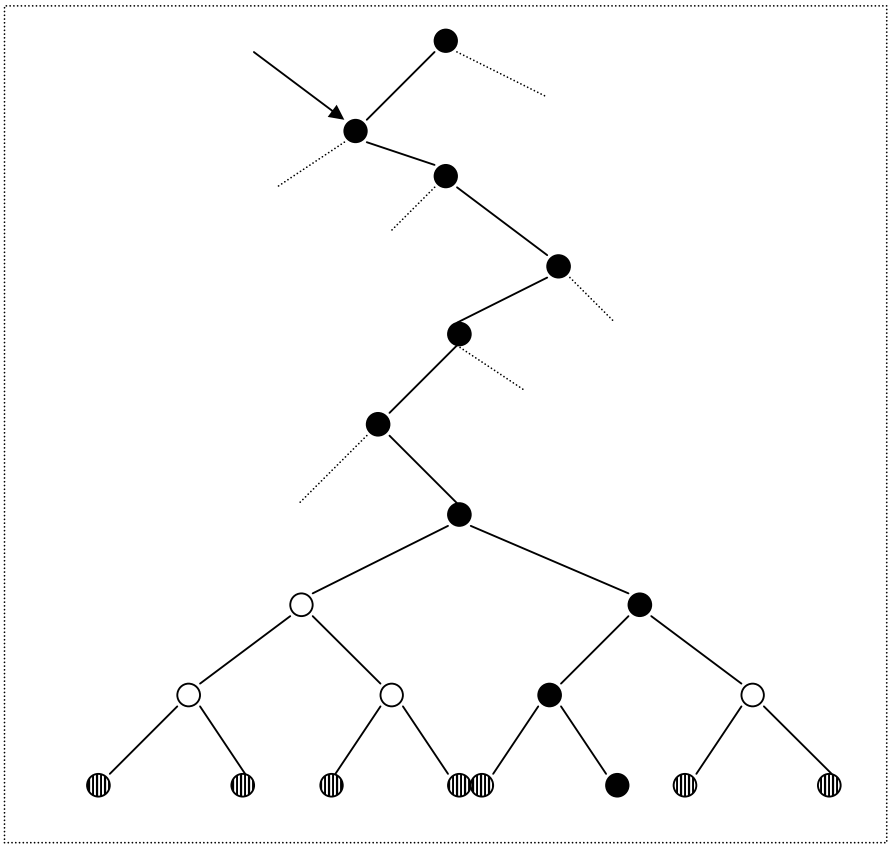


Figura 15 – Funcionamentdo da busca em profundidade

A idéia do método consiste em, primeiramente, descer na árvore até se chegar a uma condição de *backtracking*, ou seja, realizar uma busca em profundidade. Após essa busca em profundidade, teoricamente, qualquer nó ramificado pode ser considerado, ou seja, pode ser explorada qualquer sub-árvore que ainda não sofreu podas. Por exemplo, sejam os nós da Figura 16 os explorados na primeira busca em profundidades e as linhas pontilhadas as sub-árvores que devem ser exploradas. No método aqui proposto pode-se tomar qualquer sub-árvore, representada pelas linhas pontilhadas, para ser explorada.

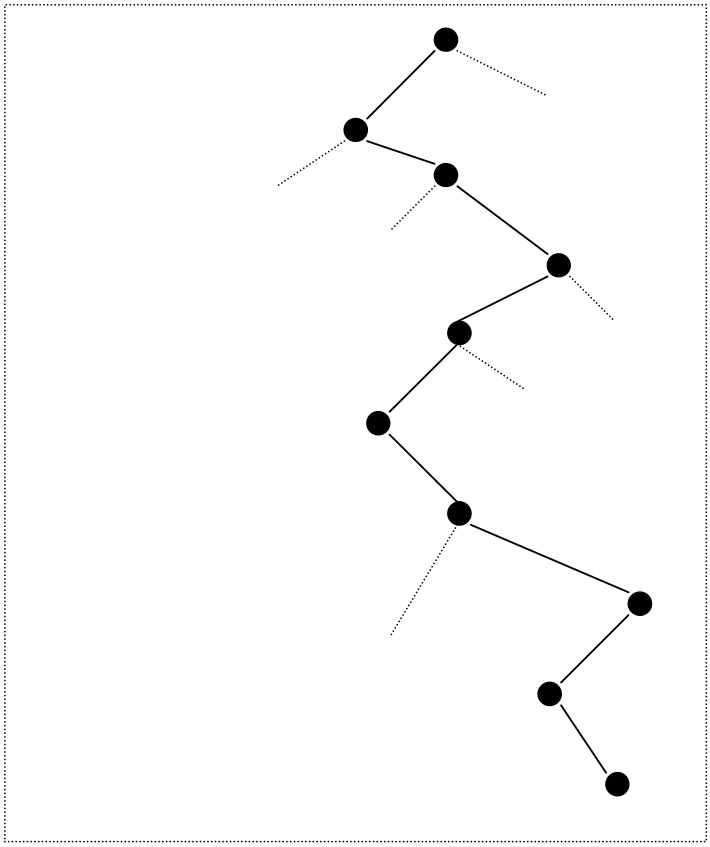


Figura 16 – primeira busca em pofundidade

Depois da escolha da sub-árvore, realiza-se, novamente, uma única busca em profundidade, ou seja, desce-se pela árvore até que ocorra um novo *backtracking*. Após esta busca, teoricamente, qualquer sub-árvore é candidata, novamente, a ser explorada. Então, apenas é explorada uma parte da sub-árvore escolhida. A Figura 17 fornece um exemplo de árvore de busca gerada com busca em profundidade e buscas em largura. Os diversos tipos de nós dizem respeito ao nível de busca que será visto na seção 4.6.1. As linhas pontilhadas representam as sub-árvores que necessitam serem exploradas.

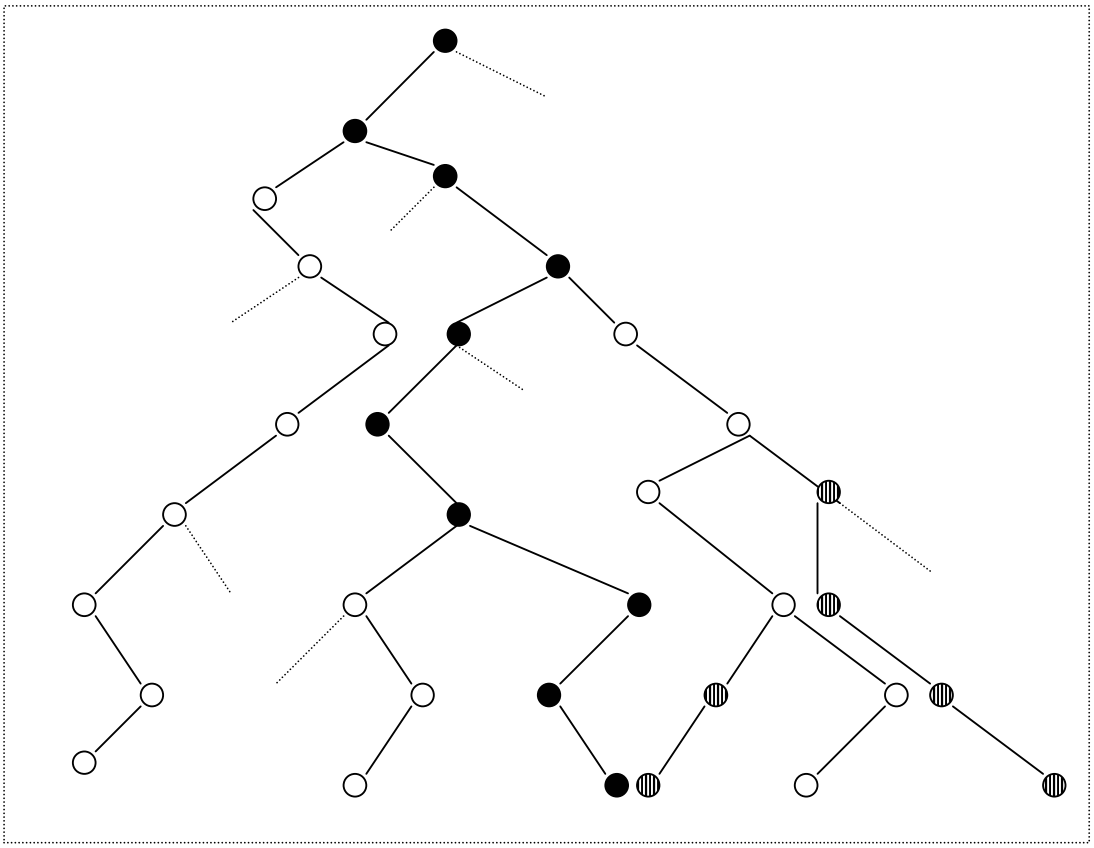


Figura 17 – busca em largura e em profundidade

4.6.1 Nível de busca

Para o bom entendimento do nível de busca será introduzido o algoritmo geral de busca. Seja α a estrutura que armazena os nós que foram ramificados e precisam ser explorados, então, um algoritmo genérico para o método pode ser descrito da seguinte forma:

Algoritmo 15: Busca na árvore

Fazer nó raiz $K_1 = K_0 = \varnothing$ e $K_2 = I$;
Calcular limites inferiores (*PLCRL* e *PLCRI*);
Realizar uma busca em profundidade considerando o nó raiz e armazenando os nós não explorados em α;
Atualizar limite superior;
Enquanto α não estiver vazio ***fazer***
Inicio
    Retira um nó *p* de α;
    Realizar uma busca em profundidade considerando o nó *p* e armazenando os nós não explorados em α;
    Atualizar limite superior, caso necessário;
Fim
----- Fim **`Algoritmo 15`** ***-----***

O **nível de busca** de um nó *v* é definido da seguinte forma: todo nó gerado pela primeira busca em profundidade tem nível de busca 0. Seja $t \in \alpha$ um nó de nível de busca 0. Os nós gerados por uma única busca em profundidade, a partir deste, têm nível de busca 1, e assim por diante. Na Figura 17, os nós em preto têm nível de busca 0, os nós brancos têm nível de busca 1 e os hachurados têm nível de busca 2.

4.6.2 Estratégia de escolha da sub-árvore a ser explorada

A partir das seguintes observações/conclusões:

- Optar somente por busca em profundidade pode levar o algoritmo a dispender muito tempo em ótimos locais;
- As soluções geradas por folhas próximas são muito parecidas;
- Provavelmente, a solução ótima do problema encontra-se em um nível de busca relativamente pequeno;

Foi definida a estratégia de retirada de um nó $v \in \alpha$. Seja α_k o conjunto de nós de nível de busca *k*. Seja um nó $v \in \alpha_k$ escolhido para fazer um busca em profundidade, fazendo $\alpha_k \Downarrow \alpha_k - \{v\}$.Os nós gerados, nesta busca, terão nível de busca *k+1* e serão colocados num conjunto α_{k+1}. A escolha de nós de α_{k+1}, para realizar uma nova busca em profundidade, só acontecerá quanto $\alpha_k = \varnothing$, ou seja, um nó de um certo nível só é escolhido quando todos os nós, de nível inferior, já tiverem sido selecionados. A primeira busca em profundidade será realizada no nó raiz. Os nós resultantes desta busca são colocados em α_0, A seguir, retira-se um nó de α_0 realizando uma nova busca em profundidade. Os nós daí resultantes serão colocados em α_1. Nós de α_1 só serão escolhidos se $\alpha_0 = \varnothing$. Se $\alpha_0 \bullet \varnothing$ então um novo nó de α_0 será retirado, para nova busca em profundidade, e assim por diante.

Foi visto como a escolha é feita levando em conta os níveis de busca. Seja *k* um nível de busca onde os nós estão sendo escolhidos. Se $\alpha_k = \varnothing$, passa-se para o nível de busca *k+1*. Se $\alpha_k \bullet \varnothing$ e $|\alpha_k| = 1$ a escolha é trivial. Se $\alpha_k \bullet \varnothing$ e $|\alpha_k| > 1$, então, uma escolha tem que ser feita. A seguir, serão descritos alguns princípios gerais que orientam esta busca. Cabe lembrar, que para todos os nós $v \in \alpha_k$, se tem $\Delta^{O/C}(v)$, no caso de K_1 viável ou $\Delta^{CMin}(v)$, no caso de K_1 não ser viável. Estes valores foram calculados para fazer as ramificações. O principio geral, que norteia a busca em largura,

é oposto da busca em profundidade. Isto, leva a uma compensação e um equilíbrio na árvore gerada. Se na busca em profundidade, em um nó v, para o qual K_1 é viável, pega-se uma facilidade *p*, tal que, $\Delta^{O/C}(v) = \Delta_p^{O/C}(v) = \max \Delta_i^{O/C}(v)$, fazendo $y_p=1$ e $y_p=0$, na busca em largura pega-se o nó *v*, tal que, $\Delta^{O/C}(v) = \min_{t \in \alpha_k} \Delta^{O/C}(t)$. O nó *v* é aquele para o qual a ramificação feita é mais dúbia, mais sujeita a possíveis questionamentos. Portanto, é razoável que a busca em largura consista em justamente escolher este nó para realizar uma busca em profundidade. O mesmo tipo de raciocínio se aplica a um nó *w*, para o qual K_1 é inviável. Neste caso, na busca em profundidade, pega-se uma facilidade *p*, tal que, $\Delta^{CMin}(v) = \Delta_p^{CMin}(v) = \min \Delta_i^{CMin}(v)$, fazendo $y_p=1$ e $y_p=0$, na busca em largura pega-se o nó *w*, tal que, $\Delta^{CMin}(w) = \max_{t \in \alpha_k} \Delta^{CMin}(t)$. O nó *w* é aquele para o qual a ramificação feita é mais dúbia, mais sujeita a possíveis questionamentos. Portanto, é razoável que a busca em largura consista em justamente escolher este nó para realizar uma busca em profundidade.

Definida a estratégia utilizando os resultado de $\Delta^{O/C}(v)$ e de $\Delta^{CMin}(v)$, ainda resta um impasse. Muitas das vezes em α_k existirá nós com valores para $\Delta^{O/C}(v)$, quando K_1 for viável, e nós com valores somente para $\Delta^{CMin}(v)$, quando K_1 inviável. Para resolver este impasse, será dado prioridade para os nós com valores para $\Delta^{CMin}(v)$, isto é, eles serão considerados antes do nós com valores para $\Delta^{O/C}(v)$. A escolha foi definida, desta forma, porque os valores de $\Delta^{O/C}(v)$ fornecem parâmetros, para decisão da ramificação, mais precisos do que $\Delta^{CMin}(v)$ (isso pode ser visto na seção 4.3). Com isso, α_k será dividido em dois conjuntos α_{k1} e α_{k2}, representando, respectivamente, os nós com valores para $\Delta^{O/C}(v)$ e os nós com valores para $\Delta^{CMin}(v)$. Conseqüentemente, α_{k+1} também será dividido em dois conjuntos, α_{k1+1} e α_{k2+1}. Cada conjunto é representado através de uma lista de prioridade, implementada através de um *heap*. No *heap* será realizar duas operações, a inserção e a remoção de um nó, cuja complexidade de ambas é *O(log n)*, mais detalhes podem ser vistos em Szwarcfiter e Markenzon [38]. A exploração da árvore é representada pelo Algoritmo 16.

Algoritmo 16: Exploração da árvore de busca

Passo 1

Enquanto $\alpha_{k2} \neq \varnothing$ ***fazer***

Inicio

Tomar *v*, tal que, $\Delta^{CMin}(v) = \max_{t \in \alpha_k} \Delta^{CMin}(t)$

Realizar uma busca em profundidade para o nó *v*, inserindo os nós ramificados em α_{k1+1} ou α_{k2+1}, conforme viabilidade de K_1.

Fim

Passo 2

Enquanto $\alpha_{k1} \neq \varnothing$ ***fazer***

Inicio

Tomar *v*, tal que, $\Delta^{O/C}(v) = \min_{t \in \alpha_k} \Delta^{O/C}(t)$

Realizar uma busca em profundidade para o nó *v*, inserindo os nós ramificados em α_{k1+1};

Fim

Passo 3

Se $\alpha_{k1+1} \neq \varnothing$ ou $\alpha_{k2+1} \neq \varnothing$ ***então***

Inicio

$\alpha_{k1} \Downarrow \alpha_{k1+1}$ *e* $\alpha_{k2} \Downarrow \alpha_{k2+1}$

$\alpha_{k1+1} \Downarrow \varnothing$ *e* $\alpha_{k2+1} \Downarrow \varnothing$

ir para o passo 1

Fim

Senão

parar

----- Fim Algoritmo 16 -----

O critério de parada, se aplica quando forem explorados todos os nós de um nível de busca *k* e não houver nenhuma ramificação, deixando $\alpha_{k1+1} \neq \varnothing$ e $\alpha_{k2+1} = \varnothing$.

4.6.3 Busca em profundidade

Uma vez escolhido um nó *v* pertencente a α é realizada uma busca em profundidade. A busca em profundidade é o ato de descer a partir de um nó *v*, neste caso *v* é retirado de α conforme seção 4.6.2, até um nó que forneça uma condição de *backtracking*. O nó *v* representa uma configuração de K_0, K_1 e K_2 e o ato de descer implica na aplicação de:

- Testes de redução: *O-Teste, C-Teste, O-RITeste, C-RITeste, O-RLTeste* e *C-RLTeste*.
- Limites inferiores: $PLCRL_{K0,K1,K2}$ e $PLCRI_{K0,K1,K2}$.

- Podas utilizando número mínimo e máximo de facilidades.
- Regras de ramificação.

Será mostrado, na seqüência, em que momento essas técnicas serão aplicadas no método desenvolvido.

4.6.3.1 Aplicação dos limites inferiores

No Capítulo 3, foram desenvolvidos 2 limites inferiores o $PLCRI_{K0,K1,K2}$ e o $PLCRL_{K0,K1,K2}$. Antes de começar o processo de busca os dois limites são calculados, considerando os conjuntos $K_0 = K_1 = \varnothing$ e $K_2 = I$. O $PLCRL_{K0,K1,K2}$ é atualizado da seguinte forma:

- Toda vez que uma facilidade tem seu valor fixado em zero ou um é atualizado o valor do $PLCRL_{K0,K1,K2}$, utilizando o Algoritmo 9 para abertura de uma facilidade e o Algoritmo 10 para o fechamento de uma facilidade.
- O Algoritmo 8, ou seja, o algoritmo que utiliza otimização por subgradiente, é aplicado com cautela, pois seu custo computacional é caro. A atualização do $PLCRL_{K0,K1,K2}$ através do Algoritmo 8 pode ocorrer em dois casos: quando um nó p for retirado de α ou correr uma ramificação. Contudo, somente ocorre se o $PLCRI_{K0,K1,K2}$ está distante do LS_{mim} (melhor limite superior encontrado). Quando $PLCRI_{K0,K1,K2}$ estiver próximo do LS_{min} significa que o valor do limite inferior esta próximo do valor da solução ótima. Essa informação indica também que o valor do $PLCRI_{K0,K1,K2}$ está próximo do $PLCRLI_{K0,K1,K2}$ ou é melhor do que este, caso estejam distantes um do outro. Sabe-se que para os nós descendentes do nó v, associado aos conjuntos K_o, K_1, K_2, os valores do *PLCRI* sempre serão maiores ou iguais (melhores ou iguais) do que os valores do $PLCRLI_{K0,K1,K2}$. Isso indica, para os nós descendentes de v, que o valor do *PLCRI* sempre estará próximo do valor do *PLCRL* ou será melhor. Sendo a atualização do *PLCRI* muito mais barata do que a atualização do *PLCRL* com otimização por subgradiente, este último será abandonado para os descendentes de v quando $PLCRI_{K0,K1,K2}$ estiver próximo de LS_{min}. A medida de proximidade utilizada é quando o intervalo entre o $PLCRI_{K0,K1,K2}$ e o LS_{min} for menor ou igual a 0,4%. Então, o Algoritmo 8 será executado quanto um nó p for retirado de α ou correr uma ramificação e a seguinte condição for satisfeita:

  $PLCRI_{K0,K1,K2} * 100 / LS_{min} < 99.6$

Por ser mais eficiente do que o $PLCRL_{K0,K1,K2}$ o $PLCRI_{K0,K1,K2}$ é calculado com mais freqüência. O $PLCRI_{K0,K1,K2}$ é calculado nas seguintes condições:

- Toda vez que um nó *p* for retirado de α.
- Após a realização dos testes de redução *O-* e *C-Teste*, no caso de sucesso dos mesmos.
- Após a realização dos testes de redução *O-* e *C-RITeste*, no caso de sucesso dos mesmos.
- Após a realização dos testes de redução *O-* e *C-RLTeste*, no caso de sucesso dos mesmos.
- Após cada ramificação.

4.6.3.2 Aplicação dos testes de redução

Os testes de redução são aplicados em cada nó. Primeiramente são aplicados o *C-RLTeste* e o *O-RLTest*e, segundo o Algoritmo 11. Depois são aplicados o *C-RITeste* e *O-RITeste*, segundo o Algoritmo 2. Por último são aplicados o *O-Teste* e *C-Teste*, conforme Algoritmo 1. O *O-RLTeste* e o *C-RLTeste* são os testes mais baratos, computacionalmente falando, então é interessante que eles sejam aplicados primeiro. Isso porque, caso se abra ou feche uma facilidade, esta não será considerada nos outros testes, que são mais caros, computacionalmente falando. Quando ao Algoritmo 2 e ao Algoritmo 1, não existe nenhuma preferência, a priori, de qual deve ser executado primeiro. A ordem foi dada empiricamente.

4.6.3.3 Aplicação das podas utilizando número mínimo e máximo de facilidades

A verificação das podas utilizando o número mínimo e máximo de facilidades é realizada a cada facilidade aberta ou fechada. Contudo, a obtenção do número mínimo e máximo de facilidades, calculados conforme seção 3.3.1 e seção 4.4, é realizada nas seguintes condições:

- Antes de começas a exploração da árvore com os conjuntos $K_0 = K_1 = \varnothing$ e $K_2 = I$.
- Após a aplicação da regra de ramificação.

4.6.3.4 Aplicação das regras de ramificação

As regras de ramificação somente serão aplicadas após a aplicação de todas as podas, conforme visto na seção 4.2, caso não sejam satisfeitas as condições de *backtracking*. Caso $w(K_1)$ seja viável será aplicada a *O/C-HeurísticaLI* caso contrário, é aplicada a *CMin-HeurísticaLI*.

4.6.4 Estudo da aplicabilidade das estratégias definidas para encontrar um bom limite superior

Traçadas as estratégias de ramificação e busca na árvore, as principais responsáveis pela obtenção dos limites superiores, será realizado um estudo objetivando validar as estratégias estabelecidas, bem como, e obter um parâmetro para uma possível parada do algoritmo, antes mesmo, de se ter garantias que a solução encontrada é ótima.

O estudo foi realizado, tendo como base os problemas de Cornuejols, Sridharan e Thizy [20], para identificar em quais níveis de busca existe a maior probabilidade de encontrar o nó que contém a solução ótima. Os problemas gerados por Cornuejols, Sridharan e Thizy [20] são divididos em categorias Estas são determinadas pelas relações entre a quantidade de oferta e a quantidade de demanda, ou seja, por $\frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j} =$ *1, 1.5, 3, 5 e 10*. Cada categoria é dividida em 6 grupos de dimensão |I|x|J| iguais a 8x25, 16x25, 25x25, 16x50, 33x50 e 50x50, com 5 testes cada. Os resultados são mostrados na Tabela 20 com as dimensões dos problemas, facilidades x clientes, embaixo de cada grupo. O índice indica o número do problema gerado (de 1 a 5).

Categoria.índice	Grupo1 8x25	Grupo2 16x25	Grupo3 25x25	Grupo4 16x50	Grupo5 33x50	Grupo6 50x50	*Média*
1.1	1	3	0	1	1	2	***1,33***
1.2	1	0	1	0	2	0	***0,67***
1.3	1	0	2	1	0	2	***1,00***
1.4	0	0	2	1	3	0	***1,00***
1.5	0	1	4	0	3	0	***1,33***
2.1	0	2	7	1	3	4	***2,83***
2.2	2	0	1	0	1	5	***1,50***
2.3	1	2	1	0	0	0	***0,67***
2.4	0	2	2	2	2	0	***1,33***
2.5	0	0	3	0	4	0	***1,17***
3.1	0	1	0	1	5	1	***1,33***
3.2	0	0	2	1	5	1	***1,50***
3.3	0	2	2	0	0	2	***1,00***
3.4	0	1	2	1	1	0	***0,83***
3.5	0	1	1	0	1	2	***0,83***
5.1	1	0	0	0	1	1	***0,50***
5.2	0	1	2	0	0	4	***1,17***
5.3	1	3	1	2	0	5	***2,00***
5.4	0	0	0	2	1	1	***0,67***
5.5	0	0	1	0	5	1	***1,17***
x.1	0	0	0	0	0	3	***0,50***
x.2	0	2	2	1	0	1	***1,00***
x.3	2	2	4	0	2	0	***1,67***
x.4	0	0	4	0	1	2	***1,17***
x.5	0	0	9	3	0	3	***2,50***
Média	***0,4***	***0,92***	***2,12***	***0,68***	***1,64***	***1,6***	***1,23***

Tabela 20 – Nível de busca em que se encontra a solução ótima

Os resultados mostraram que a solução ótima encontra-se, freqüentemente, em um nível de busca relativamente baixo. Na maioria dos casos a solução ótima foi encontrada até o nível de busca 5, a média geral foi 1,23. Houve somente dois casos onde ela foi encontrada acima do nível de busca 5: estes estão representados em células cinzas na Tabela 20.

Esses resultados, de certa forma, consolidam as estratégias de busca e as regras de ramificação. Estas regras, por serem resultados de heurísticas, tem um certo grau de incerteza. A partir do momento que se vai explorando os nós com maior nível de busca, este grau de incerteza tende a diminuir: isso porque ss pressupõe que as heurísticas tenham maior probabilidade de escolher o rumo certo do que o errado, isto é, maior probabilidade de escolher a ramificação, a ser explorada, que contenha o valor da solução ótima. Quando estamos tratando de nós em níveis de busca mais elevados

significa que já foram realizadas algumas alterações no rumo da busca na árvore. No exemplo da Figura 18, existem nós com três níveis de busca diferentes, os de nível 0, em preto, os de nível 1, em branco, e os de nível 2, hachurados. Quando se chega ao nó folha *hachurado*, podemos assumir que já foram feitas duas alterações no rumo da busca, os representados pelas setas. Assim a probabilidade da solução ótima ser encontrada é maior, pois já foram realizadas duas alterações. Com isso, a solução não deve estar em um nó de grau elevado, caso contrário, as heurísticas não estariam trazendo bons resultado.

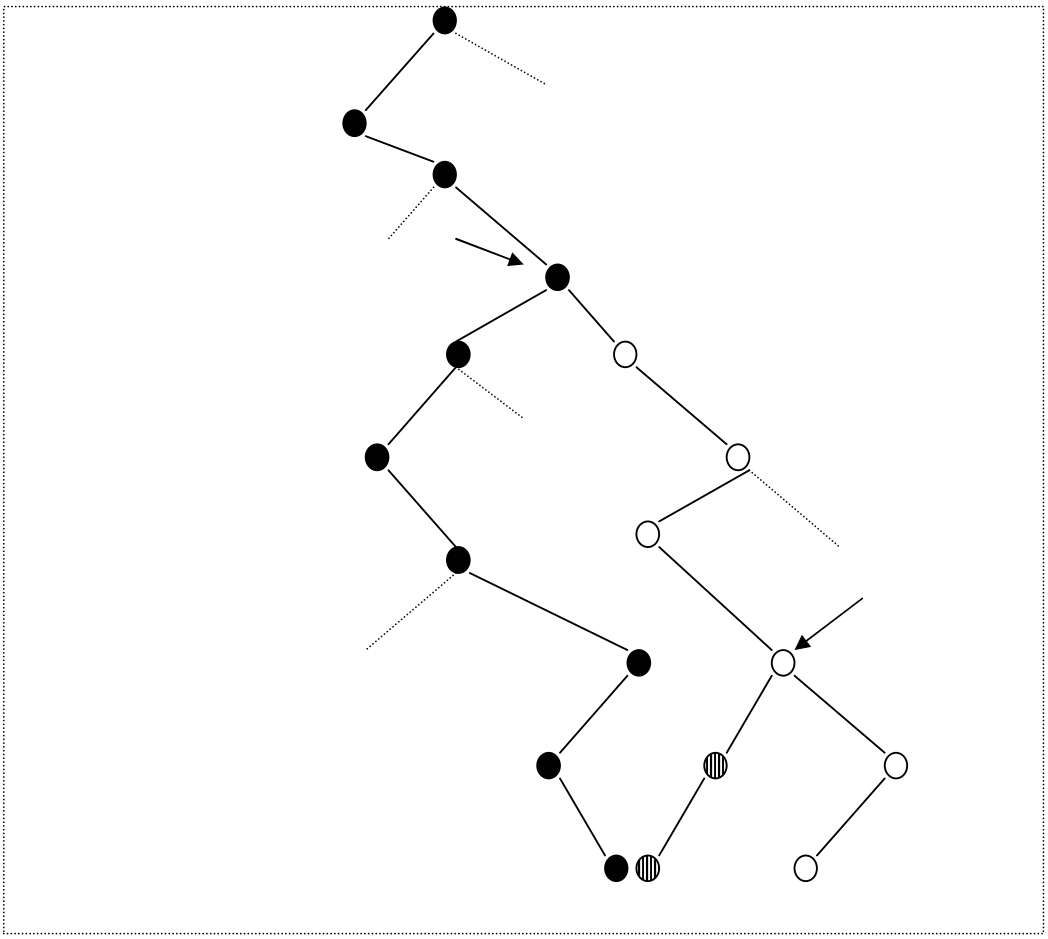


Figura 18 – níveis de busca da árvore

4.7 Algoritmo ε-ótimo

Embora o método tenha sido desenvolvido para encontrar a solução ótima do *PLC*, pode-se adaptá-lo tornando-o um algoritmo ε-ótimo, ou seja, um algoritmo onde é garantido que o erro máximo gerado seja conhecido. Seja $LI_{K0,K1,K2}$, o limite inferior gerado no nó *v*, LS_{min} a melhor limite superior encontrado e ε um valor determinado pelo usuário, então se $\frac{LS_{\min} - \mathrm{LI}_{K_0,K_1,K_2}}{\mathrm{LI}_{K_0,K_1,K_2}} \leq \varepsilon$ o processo de enumeração no nó *v* é parado. Neste caso, ocorre backtracking, ou seja, uma poda é gerada. Ela substitui, a poda realizada pelo *LI*, contudo não garante a otimalidade. É garantido que o valor ótimo estará no máximo a (ε*100)% da solução ótima. Com isso, caso se queira uma solução com um erro de 0,5% o valor de ε dever ser 0,005.

O parâmetro ε dependerá do grau de precisão que o usuário deseja da solução, quanto maior for ε, provavelmente mais rápida a solução será gerada. Em casos extremos, quando o usuário necessitar de um tempo de resposta pequeno, poderá definir um ε grande, obter a resposta e, após isso, tentar resolver com um ε menor. Caso a resposta desse processamento não tenha sido gerada a tempo, poderá utilizar a solução gerada com ε grande, tendo, com isso, uma solução onde se sabe qual é o erro máximo gerado.

4.8 Algoritmo ótimo e ε-ótimo

Foram vistos diversos algoritmos e técnicas para atacar o *PLC*. A idéia, desta seção, é juntar tudo para fornecer a estrutura geral do algoritmo ótimo e ε-ótimo. Como visto, anteriormente, a única diferença ente o ótimo e ε-ótimo é a forma do corte utilizando no limite inferior. Para o ε-ótimo é introduzido um erro ε no processo corte. Assim se a condição $\frac{LS_{\min} - \mathrm{LI}_{K_0,K_1,K_2}}{\mathrm{LI}_{K_0,K_1,K_2}} \leq \varepsilon$ for satisfeita é realizado um corte. Será utilizada esta mesma condição para o corte do algoritmo ótimo, contudo com $\varepsilon = 0$, isto é, erro zero.

4.8.1 Testes de redução

Os testes de redução abrem e fecham facilidades, fazem isto retirando uma facilidade de K_2 e colocando a em K_0, no fechamento, e em K_1, na abertura. Os algoritmos que aplicam os testes de redução são Algoritmo 1, Algoritmo 2, e Algoritmo

11. Como visto na seção 4.6.3.1 toda vez que uma facilidade for aberta ou fechada deve ser atualizado o limite inferior *PLCRL*. Da mesma forma, toda vez que uma facilidade for aberta ou fechada é verificado se ocorreu um corte através do número máximo e mínimo de facilidades abertas na solução ótima, conforme seção 4.6.3.3. Assim sendo, devem ser incorporado nos algoritmos de redução algumas funcionalidades. Essas funcionalidades serão incorporadas após o fechamento e após a abertura de uma facilidade por algum teste de redução.

a) Fechamento de uma facilidade

Seja *f_min* o número mínimo de facilidades abertas na solução ótima para um nó *v* e seus descendentes, então, após algum teste de redução realizar o fechamento de uma facilidade *k*, ou seja, fazer $K_0 = K_0 \cup \{k\}$ e $K_2 = K_2 - \{k\}$ deve-se executar o Algoritmo 17.

Algoritmo 17: atualização do *PLCRL* e Corte com *f_min*

Atualizar $PLCRL_{K0,K1,K2}$ conforme Algoritmo 10

Atualizar $LI_{K0,K1,K2}$ caso necessário

Se $\frac{LS_{\min} - \mathrm{LI}_{K_0,K_1,K_2}}{\mathrm{LI}_{K_0,K_1,K_2}} \leq \xi$ ***Então Parar*** *{backtracking}*

Se $|K_1 \cup K_2| = f\_min$ ***Então***

Inicio

$K_1 = K_1 \cup K_2$

$K_2 = \varnothing$

Fim

----- Fim Algoritmo 17 -----

b) Abertura de uma facilidade

Seja *f_max* o número máximo de facilidades abertas na solução ótima para um nó *v* e seus descendentes, então, após algum teste de redução realizar a abertura de uma facilidade *k*, ou seja, fazer $K_1 = K_1 \cup \{k\}$ e $K_2 = K_2 - \{k\}$ deve-se executar o Algoritmo 18.

Algoritmo 18: atualização do *PLCRL* e Corte com *f_max*

Atualizar $PLCRL_{K0,K1,K2}$ conforme Algoritmo 9

Atualizar $LI_{K0,K1,K2}$ caso necessário

Se $\frac{LS_{\min} - \mathrm{LI}_{\mathrm{K}_0,\mathrm{K}_1,\mathrm{K}_2}}{\mathrm{LI}_{\mathrm{K}_0,\mathrm{K}_1,\mathrm{K}_2}} \leq \xi$ ***Então Parar*** *{backtracking}*

Se $|K_1| = f\_max$ ***Então***

Inicio

$K_0 = K_2$

$K_2 = \varnothing$

Fim

----- Fim Algoritmo 18-----

4.8.2 Primeiro limite superior

Os limites inferiores são calculados, primeiramente, no nó raiz. Nesse ponto ainda não foi encontrado um limite superior. Contudo para o cálculo do *PLCRL* é necessário que se tenha um limite superior. Como esse objetivo, foi desenvolvida uma forma simples de se obter um limite superior. Seja $a_1 \bullet c_2 \bullet \ldots \bullet c_n$, onde n = |I|, uma ordenação das capacidades das facilidades. Tomar as primeiras *k* facilidades que fazem $\sum_{i=1}^{k} a_k \geq \sum_{j \in J} b_j$, ou seja, que tornem o problema viável, e abri-las. Seja *w(K)* o problema de transporte, onde $K \subseteq I$, o Algoritmo 19 encontra o primeiro limite superior. O valor encontrado é colocado em LS_{min} (melhor limite superior encontrado).

Algoritmo 19: primeiro limite superior

Ordenar os valores de a_i, $\forall i \in I$, em ordem decrescente.

Seja *K* o menor conjunto de facilidades tal que $\sum_{i=1}^{k} a_k \geq \sum_{j \in J} b_j$

Calcular *w(K)*

Fazer $LS_{min} = w(K) + \sum_{i \in K} f_i$

----- Fim Algoritmo 19-----

4.8.3 Algoritmo de busca em profundidade

A seção 4.6.3 fornece explicações de como é realizado a busca em profundidade. Deste modo, será apresentado o algoritmo para realização da busca em profundidade, Algoritmo 20.

Algoritmo 20: Busca em profundidade

Passo 1: *{Cálculo dos limites inferiores}*

Calcular $PLCRI_{K0,K1,K2}$

Atualizar $LI_{K0,K1,K2}$ caso necessário

Se $\frac{LS_{\min} - \mathrm{LI}_{K_0,K_1,K_2}}{\mathrm{LI}_{K_0,K_1,K_2}} \leq \xi$ ***Então Parar*** *{backtracking}*

Se $PLCRI_{K0,K1,K2} * 100 / LS_{min} < 99.6$ ***Então***

Inicio

Calcular $PLCRL_{K0,K1,K2}$ conforme **Algoritmo 8** *{atualização do PLCRL}*

Atualizar $LI_{K0,K1,K2}$ caso necessário

Fim

Se $\frac{LS_{\min} - \mathrm{LI}_{K_0,K_1,K_2}}{\mathrm{LI}_{K_0,K_1,K_2}} \leq \xi$ ***Então Parar*** *{backtracking}*

Passo 2: *{Aplicação do C-RLTeste e o O-RLTeste}*

Aplicar os testes de redução *C-RLTeste* e o *O-RLTeste*, segundo o Algoritmo 11, incorporado do Algoritmo 17 e Algoritmo 18.

Se $K_2 = \varnothing$ **Então**

Ir para o Passo 6

Calcular $PLCRI_{K0,K1,K2}$

Atualizar $LI_{K0,K1,K2}$ caso necessário

Se $\frac{LS_{\min} - \mathrm{LI}_{K_0,K_1,K_2}}{\mathrm{LI}_{K_0,K_1,K_2}} \leq \xi$ ***Então Parar*** *{backtracking}*

Passo 3: *{Aplicação do C-RITeste e o O-RITeste}*

Aplicar os testes de redução *C-RITeste* e o *O-RITeste*, segundo o Algoritmo 2, incorporado do Algoritmo 17 e Algoritmo 18.

Se $K_2 = \varnothing$ **Então**

Ir para o Passo 6

Calcular $PLCRI_{K0,K1,K2}$

Atualizar $LI_{K0,K1,K2}$ caso necessário

Se $\frac{LS_{\min} - \mathrm{LI}_{\mathrm{K_0,K_1,K_2}}}{\mathrm{LI}_{\mathrm{K_0,K_1,K_2}}} \leq \xi$ ***Então Parar*** *{backtracking}*

Passo 4: *{Aplicação do C-Teste e o O-Teste}*

Aplicar os testes de redução *C-Teste* e o *O-Teste*, segundo o **Algoritmo 1**, incorporado do Algoritmo 17 e Algoritmo 18.

Se $K_2 = \varnothing$ **Então**

Ir para o Passo 6

Calcular $PLCRI_{K0,K1,K2}$

Atualizar $LI_{K0,K1,K2}$ caso necessário

Se $\frac{LS_{\min} - \mathrm{LI}_{\mathrm{K_0,K_1,K_2}}}{\mathrm{LI}_{\mathrm{K_0,K_1,K_2}}} \leq \xi$ ***Então Parar*** *{backtracking}*

Passo 5: *{Ramificação}*

Se $w(K_1)$ viável ***Então***

Inicio

Ramificar facilidade *i* fornecida pela *O/C-HeuristicaLI*

Inserir nó ramificado em α_{k1+1}

Calcular número máximo de facilidades ($f\_max_{K0,K1,K2}$) segundo seção 4.4.2

Calcular número máximo de facilidades ($f\_min_{K0,K1,K2}$) segundo seção 4.4.5

Fim

Senão

Inicio

Ramificar facilidade *i* fornecida pela *CMin-HeurísticaLI*

Inserir nó ramificado em α_{k2+1}

Calcular número máximo de facilidades ($f\_max_{K0,K1,K2}$) segundo seção 4.4.2

Calcular número máximo de facilidades ($f\_min_{K0,K1,K2}$) segundo seção 4.4.4

Fim

Se *i* foi fechada pela aplicação da heurística ***Então***

atualização do *PLCRL* e Corte com *f_min,* segundo Algoritmo 17

Senão

atualização do *PLCRL* e Corte com *f_max,* segundo Algoritmo 18

Se $K_2 = \varnothing$ **Então**

Ir para o Passo 6

Senão

Ir para o Passo 1

Passo 6: cálculo e atualização do limite superior

Se $w(K_1)$ viável ***Então***

Inicio

Calcular $w(K_1)$

$LS \Downarrow w(K_1) + \Sigma_{i \in K1} f_i$

Se $LS < LS_{min}$ ***Então***

$LS_{min} \Downarrow LS$

Fim

----- Fim Algoritmo 20-----

4.8.4 Algoritmo geral

O algoritmo que será apresentado na seqüência é o algoritmo geral, isto é, ele é o responsável pela resolução do *PLC*. Nele é realizado o processo de inicialização, onde é calculado o número mínimo e máximo de facilidades abertas para o problema original o primeiro limite superior, os dois limites inferiores *PLCRI* e *PLCRL*, e é colocado o nó raiz em α_{k2}. Depois desta inicialização é realizado o processo de busca na árvore, o qual foi apresentado na seção 4.6

Algoritmo 21: Algoritmo Geral
Passo 1: *{inicialização}*

Calcular LS_{min} segundo Algoritmo 19

Fazer nó raiz $K_1 = K_0 = \varnothing$ e $K_2 = I$;

Calcular o $f\_min_{K0,K1,K2}$ segundo seção 4.4.3

Calcular o $f\_max_{K0,K1,K2}$ segundo 4.4.1

Calcular $PLCRL_{K0,K1,K2}$ segundo Algoritmo 8

Calcular $PLCRI_{K0,K1,K2}$

$LI_{K0,K1,K2} = max(PLCRI_{K0,K1,K2}, PLCRL_{K0,K1,K2})$ *{atualizar LI}*

inserir raiz em α_{k2}

Passo 2: *{processo de busca}*

Enquanto $\alpha_{k2} \neq \varnothing$ ***fazer***
Inicio

Tomar v, tal que, $\Delta^{CMin}(v) = \max_{t \in \alpha_k} \Delta^{CMin}(t)$

Realizar uma busca em profundidade para o nó v, conforme Algoritmo 20.

Fim

Passo 3:

Enquanto $\alpha_{k1} \neq \varnothing$ ***fazer***
Inicio

Tomar v, tal que, $\Delta^{O/C}(v) = \min_{t \in \alpha_k} \Delta^{O/C}(t)$

Realizar uma busca em profundidade para o nó v, conforme Algoritmo 20.

Fim

Passo 4:
Se $\alpha_{k1+1} \neq \varnothing$ ou $\alpha_{k2+1} \neq \varnothing$ ***então***
Inicio

$\alpha_{k1} \Downarrow \alpha_{k1+1}$ *e* $\alpha_{k2} \Downarrow \alpha_{k2+1}$
$\alpha_{k1+1} \Downarrow \varnothing$ *e* $\alpha_{k2+1} \Downarrow \varnothing$
ir para o passo 2

Fim
Senão
parar
----- Fim Algoritmo 21-----

4.9 Resultados computacionais

Para que possa ser verificado se o método desenvolvido para resolução do Problema de Localização Capacitado é eficiente, serão realizadas comparações com os métodos encontrados na literatura e com o *software* comercial XPRESS, o qual tem

mostrado ser um dos melhores *softwares* para resolução de problemas de programação inteira. Vale ressaltar que o XPRESS é um *software* para resolver problemas gerais de programação inteira. Essa característica o coloca em certa desvantagem. De qualquer forma, é um *software* comercial implementado utilizando as melhores técnica de programação matemática e computação.

Para que as comparações possam ser realizadas serão utilizados problemas encontrados na literatura além de um gerador de problemas, o qual será apresentado na seção 4.9.1. Vale destacar que também serão utilizados problemas novos, isso porque, será explorado, de forma mais abrangente, as características do *PLC* e suas dimensões. Isso é necessário para que se possa atingir os objetivos propostos, os quais são resolver:

- ¬ Problemas com dimensões maiores que as dos encontrados na literatura, principalmente no que se refere ao número de facilidades, isto é, ao número de variáveis binárias (0 ou 1).
- ¬ Problemas não Euclidianos.
- ¬ Problemas com muitos ótimos locais.

O algoritmo para resolução do *PLC* foi implementado em C++ sob a sistema operacional Linux Conectiva 8. O compilador utilizado foi o g++ versão 2.95.3.

4.9.1 Gerador de problemas

Os dados dos problemas serão gerados de forma aleatória. Contudo, para que o processo não fique somente sob responsabilidade do sistema, o gerador terá vários parâmetros. Com isso se poderá obter problemas com características diferentes. Serão gerados problemas não euclidianos em relação aos custos de transporte. Estes tipos de problemas tendem, intuitivamente, a terem um maior grau de dificulade.

4.9.1.1 Funções

O gerador de problemas foi baseado no gerador de problemas de Cornuejols, Sridharan e Thizy [20]. No entanto, foram adicionadas novas formas de obtenção de alguns dados, as quais foram obtidas de formas empíricas, baseadas em experiências adquiridas. O gerador terá as seguintes funções:

4.9.1.1.1 Geração dos dados da demanda

As demandas são geradas aleatoriamente, conforme Cornuejols, Sridharan e Thizy [20], entre 5 e 35, isto é, $b_j \ \forall j \in J$ varia no intervalo [5,35].

4.9.1.1.2 Geração dos dados de capacidade

As ofertas são geradas aleatoriamente respeitando duas restrições: a primeira restrição a ser atendida é a viabilidade do problema, ou seja, $\sum_{i \in I} a_i \geq \sum_{j \in J} b_j$, e a segunda restrição e a relação $r = \frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j}$, a qual será especificada pelo usuário.

4.9.1.1.2.1 Geração das capacidades de cada facilidade

A geração das capacidades é realizada em dois passos. No primeiro passo as capacidades serão geradas aleatoriamente entre 10 e 160. Isto é, $a_i \ \forall i \in I$ varia, uniformente, no intervalo [10,160]. No segundo passo as capacidades são ajustadas para atender a relação $r = \frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j}$. Esse ajuste é realizado aumentado as capacidades ou diminuindo-as de 1 em 1, em ordem crescente relativo ao número da facilidade, até que a relação seja satisfeita.

4.9.1.1.3 Geração dos dados de custo fixo

Os custos fixos serão gerados após a obtenção das capacidades. Seja U[a,b] a distribuição uniforme do intervalo [a,b], então os custos fixos são obtidos através da seguinte fórmula:

$$f_i = (U[0,90] + U[100,110]\sqrt{a_i})q \qquad \forall i \in I$$

,onde q é fornecido pelo usuário. Esse parâmetro pode fazer com que a diferença entre os custos fixos e variáveis seja grande ou pequena. A função foi obtida em Cornuejols, Sridharan e Thizy [20], exceto o parâmetro q. Assim, quanto $q=1$, a geração do custo fixo é igual ao gerado por Cornuejols, Sridharan e Thizy [20].

4.9.1.1.4 Geração dos dados de custo de transporte

Para geração dos custos de transporte são inicialmente gerados pontos, representando clientes e facilidades, uniformemente em um quadrado unitário. Seja d_{ij} a

distância euclidiana da facilidade *i* para o cliente *j*, então $c_{ij} = dij +/- 0,2 * r$, onde *r* é um número gerado aleatoriamente e uniformemente no intervalo [0,1]. Isso significa dizer que o custo de transporte será até 20% maior ou até 20% menor do que a distância Euclidiana, fazendo, com isso, que o problema gerado seja não Euclidiano. Caso $c_{ij} < 0$ faz-se $c_{ij} = d_{ij}$. Finalmente, $c_{ij} = c_{ij} * t$, onde t é um parâmetro fornecido pelo usuário. Esse parâmetro faz com que a diferença entre os c_{ij} seja pequena ou grande. Cornuejols, Sridharan e Thizy [20] utilizam a distância Euclidiana.

4.9.1.1.5 Gravação dos dados

Os dados do problema gerado serão armazenados em um arquivo *ascii* conforme a especificado abaixo, onde *n* é o número de facilidades, *m* é o número de clientes, a_i é a capacidade da facilidade *i*, f_i é o custo fixo da facilidade *i*, b_j é a demanda do cliente *j*, e c_{ij} é o custo de transporte, de uma unidade, da facilidade *i* para o cliente *j*.

Formato do arquivo:

n m
a_1 f_1
a_2 f_2
.
.
.
a_n f_n
b_1 c_{11} c_{21} ... c_{n1}
b_2 c_{12} c_{22} ... c_{n2}
.
.
.
b_m c_{1m} c_{2m} ... c_{nm}

4.9.2 Geração de dados dos problemas testes

Para que o método desenvolvido seja validado é necessária a geração de problemas testes com diversas características. Os problemas foram gerados utilizando-se o gerador definido na seção 4.9.1, e diferenciam-se através das seguintes características:

1. Diferentes relações entre os custos de transporte, desde uma pequena diferença entre eles a uma grande diferença. Para isso t = 10, 20, 500 e 5000.

2. Diferentes dimensões de problemas. As dimensões utilizadas são 200x200, 300x300, 500x500, onde o primeiro número representa a quantidade de facilidades e o segundo a quantidade de centros de demanda.
3. Diferentes relações entre a quantidade de capacidade e a quantidade de demanda. As relações consideradas são $\frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j} = 1.5, 2, 5, 10$ e 20.
4. Diferentes relações entre custo fixo e custo variável, onde $q=1, 5$ e 20.

Para cada combinação das características citadas foram gerados três problemas diferentes. Para facilitar a compreensão o nome do problema terá o seguinte formato: plc_r_IxJ_t_q-n.dat, onde:

r = relações entre a quantidade de demanda e a quantidade de oferta (1.5, 2, 5, 10 e 20).

I = número de centros de oferta (200, 300 e 500).

J = numero de centros de demanda (200, 300 e 500).

t = diferença máxima entre os custos de transporte (10, 500, 5000).

q = 1, 5 e 20

n = número do problema (1,2 e 3).

4.9.3 Resultados Computacionais de problemas da literatura

Seráo apresentados, primeiramente, os resultados computacionais para os problemas da literatura formulados por Kühn & Hamburger [32] (problemas disponibilizados na OR-Library), Beasley [11] (problemas disponibilizados na OR-Library), Cornuejols, Sridharan e Thizy [20] e Ardal [3]. Inicialmente, será mostrado o comportamento do método proposto quando se pára a exploração da árvore nos nós de nível de busca igual a três, isto é, o comportamento do método nas primeiras buscas em profundidades. Assim ter-se-á parâmetros da solução gerada quando for necessário parar o método antes do mesmo garantir que a solução ótima foi encontrada. Pode-se considerar esse processo com um método heurístico. Os resultados serão comparados com os melhores métodos heurísticos encontrados na literatura.

4.9.3.1 Critérios de comparação

a) Algoritmos heurísticos

O algoritmo proposto será parado quando explorados todos os nós do nível de busca 3. Esse critério do parada fornece um resultado heurístico para o método.

Assim sendo, serão realizadas comparações com algoritmos heurísticos da literatura. Dois são os critérios de comparação utilizados, a qualidade de solução (distância em relação a solução ótima) e o tempo de execução. Em relação ao tempo de execução é difícil afirma alguma coisa com precisão, pois as máquinas utilizadas para resolução dos diversos métodos são muito diferentes uma das outras. De qualquer forma, será utilizado, como parâmetro de comparação entre alguns computadores, o número de operações de Pontos Flutuantes por segundo. Esse critério também é questionável, pois são retirados de diversas fontes. A Tabela 21 mostra o computador, o número de operações de Pontos Flutuantes (MFLOP) e a referência.

b) Algoritmos exatas

São utilizados dois critérios de comparação, o número de nós explorados na árvore de busca e o tempo de execução. O número de nós explorados, no método proposto, é contado quando é aplicada uma regra de ramificação. Assim, o número de nós explorados é inicializado como 1 (exploração do nó raiz) e a cada ramificação é somado 2 a este número. Provavelmente, este critério ou um critério parecido é utilizado pelo XPRESS e pelos métodos que serão utilizados para comparação. A principal diferença é que em cada método é realizado um processamento diferente no nó, isto faz com que este critério de comparação não seja tão representativo em alguns casos. Provavelmente, alguns dos métodos, que serão utilizados para comparação, não computam os nós ramificados, que ao retirar da estrutura de armazenamento tem o limite inferior maior do que o limite superior. Outra característica, destes métodos, é que alguns fazem um pré-processamento. Neste pré-processamento algumas vezes é garantido a otimalidade, isso faz que o método aponte zero nós explorados. Como dito anteriormente, em relação ao tempos de execução é difícil afirma alguma coisa categoricamente. Para algumas máquinas foi conseguido o número de operação de Pontos Flutuantes, os quais são mostrados na Tabela 21.

Computador	**MFLOP**	**Referência**
Cray-1S	800	Fox [24]
Pentium III 700 Mhz	176	http://homepage.virgin.net/roy.longbottom/whetstone%20results.htm
Cray X-MP	900	Fox [24]
Sparc ELC	2,5	Compernolle [19]

Tabela 21 – MFLOP de alguns computadores

4.9.3.2 Comportamento do método quando se pára a exploração nos nós de nível de busca igual a três

A máquina utilizada para obtenção dos resultados foi uma máquina Pentium III, 700 Mhz com 256 Mega bytes de memória principal. Vale observar que é uma máquina defasada para os dias de hoje.

A Tabela 22 mostra os resultados obtidos, para os problemas de Kühn & Hamburger. Z^h representa a solução gerada pelo método proposto quando se pára a exploração da árvore de busca nos nós com nível de busca igual a três. Mostra-se o erro em relação à solução ótima, o erro em relação ao limite inferior (LI), o qual é obtido pelo método, e o tempo em segundos obtido para encontrar Z^h. A solução ótima do problema é representada por Z^*. A comparação é realizada com o algoritmo heurístico desenvolvido por Beasley [12], onde é fornecido o tempo em segundos e a qualidade da solução obtida Z_{UB} (solução viável) pelo seu método em relação à solução ótima Z*. Beasley utilizou um Cray X-MP/28 para obtenção de seus resultados.

Problema	**nXm**	**Algoritmo Proposto**			**Beasley**	
		$100*((Z^h - LI)/LI)$	**$100*((Z^h - Z^*)/Z^*)$**	**Seg.**	**$100*((Z_{ub} - Z^*)/Z^*)$**	**Seg.**
41	16x50	0,00	0,00	0,18	0	0,97
42		0,00	0,00	0,17	0	0,57
43		0,00	0,00	0,21	0	0,58
44		0,00	0,00	0,22	0	0,59
51		0,00	0,00	0,22	0	0,80
61		0,00	0,00	0,08	0	0,46
62		0,00	0,00	0,15	0	0,44
63		0,00	0,00	0,18	0	0,71
64		0,00	0,00	0,15	0	0,36
81	25x50	0,00	0,00	0,49	0	2,27
82		0,00	0,00	0,41	0	1,74
83		0,00	0,00	0,38	0	1,90
84		0,00	0,00	0,82	0	2,86
91		0,00	0,00	0,13	0	1,23
92		0,00	0,00	0,55	0	1,11
93		0,00	0,00	0,39	0	1,17
94		0,00	0,00	0,45	0,06	1,25
111	50x50	0,00	0,00	0,60	0	2,74
112		0,00	0,00	0,85	0	3,11
113		0,00	0,00	1,24	0	2,71
114		0,00	0,00	1,65	0,44	4,22
121		0,00	0,00	0,33	0	2,37
122		0,00	0,00	0,66	0	1,72
123		0,00	0,00	0,77	0	1,47
124		0,33	0,00	1,38	0,17	1,91
Média		***0,01***	***0,00***	***0,51***	***0,03***	***1,57***

Tabela 22 – Solução heurística (problemas teste de Kühn & Hamburger)

Para os problemas de Kühn & Hamburger o método proposto obteve um erro médio de 0,01%, enquanto o proposto por Beasley 0,03%. Somente em dois problemas o método proposto não conseguiu atingir a solução ótima, quando o mesmo é parado no término da exploração dos nós de nível de busca igual a três. Fazendo a equivalência entre as máquinas, utilizando a Tabela 21, o Pentium III levou em média 0,51 segundos para resolver os problemas enquanto o Cray X-MP levou 8,03 segundos.

A Tabela 23 mostra os resultados obtidos, para os problemas de Beasley [10]. Z^h representa a solução gerada pelo método proposto quando se pára a exploração da árvore de busca nos nós com nível de busca igual a três. Mostra-se o erro em relação à solução ótima, o erro em relação ao limite inferior (LI), o qual é obtido pelo método, e o tempo em segundos obtido para encontrar Z^h. A solução ótima do problema é representada por Z^*. A comparação é realizada com o algoritmo heurístico desenvolvido por Beasley [12] , onde é fornecido o tempo em segundos e a qualidade da solução obtida Z_{UB} (solução viável) pelo seu método em relação à solução ótima Z*. Beasley utilizou um Cray X-MP/28 para obtenção de seus resultados.

Problema	**nXm**	**Algoritmo Proposto**			**Beasley**	
		$100*((Z^h - LI)/LI)$	**$100*((Z^h - Z^*)/Z^*)$**	**Seg.**	**$100*((Z_{ub} - Z^*)/Z^*)$**	**Seg.**
A1	100x1000	0,14	0,00	138,60	0,24	73,65
A2	100x1000	0,00	0,00	115,52	0,45	65,49
A3	100x1000	0,00	0,00	111,98	0,00	29,87
A4	100x1000	0,00	0,00	49,09	0,00	16,58
B1	100x1000	0,00	0,00	54,96	0,40	131,84
B2	100x1000	0,29	0,00	335,74	0,14	111,78
B3	100x1000	0,56	0,36	374,28	0,91	96,14
B4	100x1000	0,00	0,00	150,95	0,00	48,92
C1	100x1000	0,08	0,00	93,10	0,00	105,74
C2	100x1000	0,26	0,00	301,25	0,00	83,42
C3	100x1000	0,12	0,00	102,34	0,44	86,89
C4	100x1000	0,00	0,00	37,60	0,00	59,11
Média		***0,12***	***0,03***	***155,45***	***0,22***	***75,79***

Tabela 23 - Solução heurística (problemas teste de Beasley)

Para os problemas de Beasley [12] o método proposto obteve um erro médio de 0,03% enquanto o proposto por Beasley 0,22%. Somente em um problema o método proposto não conseguiu atingir a solução ótima, quando o mesmo é parado no término da exploração dos nós de nível de busca igual a três. Até o erro médio relativo ao limite inferior foi menor do que o erro gerado pelo método de Beasley. Fazendo a equivalência

entre as máquinas, utilizando a Tabela 21, o Pentium III levou em média 155,45 segundos para resolver os problemas enquanto que o Cray X-MP 387,56 segundos.

A Tabela 25 mostra os resultados obtidos, para os problemas de Cornuejols, Sridharan e Thizy [20]. Eles dividiram os problemas em grupos, os quais são determinadas pelas relações entre a quantidade de demanda e a quantidade de oferta, as relações são $\frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j}$ = *1, 1.5, 3, 5 e 10*. Cada grupo é dividido em 6 subgrupos de dimensão |I|x|J| iguais a 8x25, 16x25, 25x25, 16x50, 33x50 e 50x50, com 5 testes cada, totalizando os 30 testes de cada grupo. Z^h representa a solução gerada pelo método proposto quando se pára a exploração da árvore de busca nos nós com nível de busca igual a três. Mostra-se o erro em relação à solução ótima, o erro em relação ao limite inferior (LI), o qual é obtido pelo método, e o tempo em segundos obtido para encontrar Z^h. A solução ótima do problema é representada por Z^*. A comparação é realizada com os algoritmos heurísticos desenvolvido por Jacobsen [30] (VSM e DROP-HI) e por Cornuejols, Sridharan e Thizy [20] (H). A solução obtida por cada método heurístico é denominada de Z_{UB}. Os tempos computacionais desses métodos são mostrados na Tabela 25, onde é apresentado a média para cada dimessão. Os tempos computacionais para as heurística H, VSM e DROP-HI foram obtidos utilizando um computador Dec. 20.

Problema	**nxm**	**Algoritmo proposto**			**Thizy**	**Jacobsen**	
					H	**VSM**	**DROP-HI**
		$100*((Z^h - LI)/LI)$	**$100*((Z^h - Z^*)/Z^*)$**	**Seg.**	**$100*((Z_{ub} - Z^*)/Z^*)$**	**$100*((Z_{ub} - Z^*)/Z^*)$**	**$100*((Z_{ub} - Z^*)/Z^*)$**
thiz1-a1.dad	8x25	0,00	0,00	0,07	0,00	0,00	17,27
thiz1-a2.dad		0,00	0,00	0,10	0,00	5,73	5,73
thiz1-a3.dad		0,00	0,00	0,07	0,00	6,22	11,14
thiz1-a4.dad		0,00	0,00	0,06	0,00	13,13	13,13
thiz1-a5.dad		0,00	0,00	0,09	0,00	8,82	8,82
thiz1-b1.dad	16x25	1,69	0,00	0,28	0,00	15,69	15,69
thiz1-b2.dad		0,76	0,00	0,14	0,00	6,23	20,80
thiz1-b3.dad		0,00	0,00	0,16	0,00	13,30	13,30
thiz1-b4.dad		0,00	0,00	0,13	0,00	10,92	15,20
thiz1-b5.dad		1,18	0,00	0,51	0,00	0,29	5,36
thiz1-c1.dad	25x25	0,00	0,00	0,20	0,00	5,95	30,01
thiz1-c2.dad		0,00	0,00	0,28	0,00	10,02	20,96
thiz1-c3.dad		0,00	0,00	0,30	0,00	9,18	26,96

thiz1-c4.dad		0,00	0,00	0,24	0,00	15,01	38,25
thiz1-c5.dad		0,86	0,00	0,42	0,00	12,57	26,36
thiz1-d1.dad	16x50	0,00	0,00	0,30	0,00	15,30	17,97
thiz1-d2.dad		0,00	0,00	0,19	0,00	17,75	23,71
thiz1-d3.dad		0,00	0,00	0,34	0,00	4,08	24,42
thiz1-d4.dad		0,00	0,00	0,38	0,00	2,77	16,78
thiz1-d5.dad		0,00	0,00	0,24	0,00	10,48	22,86
thiz1-e1.dad	33x50	1,17	0,00	1,55	0,00	25,52	26,24
thiz1-e2.dad		0,83	0,00	1,35	0,00	19,35	24,10
thiz1-e3.dad		0,00	0,00	0,40	0,00	18,58	28,02
thiz1-e4.dad		0,95	0,00	1,63	0,00	5,77	26,10
thiz1-e5.dad		1,33	0,12	1,30	0,00	11,72	29,98
thiz1-f1.dad	50x50	0,27	0,00	1,18	0,00		27,63
thiz1-f2.dad		0,00	0,00	0,75	0,00		26,08
thiz1-f3.dad		0,39	0,00	1,14	0,00		24,44
thiz1-f4.dad		0,00	0,00	0,67	0,00		28,88
thiz1-f5.dad		0,00	0,00	0,75	0,00		27,58
thiz2-a1.dad	8x25	0,00	0,00	0,07	0,00	0,00	24,05
thiz2-a2.dad		2,80	0,00	0,11	0,00	0,82	8,40
thiz2-a3.dad		0,00	0,00	0,09	0,00	4,27	14,40
thiz2-a4.dad		0,00	0,00	0,06	0,00	0,00	35,71
thiz2-a5.dad		0,00	0,00	0,07	0,00	19,54	21,84
thiz2-b1.dad	16x25	0,00	0,00	0,16	0,00	10,36	41,98
thiz2-b2.dad		0,00	0,00	0,17	0,00	13,00	22,36
thiz2-b3.dad		0,00	0,00	0,16	0,00	8,75	38,64
thiz2-b4.dad		2,22	0,00	0,40	0,00	14,30	20,91
thiz2-b5.dad		0,00	0,00	0,11	0,00	18,59	44,04
thiz2-c1.dad	25x25	1,93	0,48	0,62	0,00	4,82	34,68
thiz2-c2.dad		1,08	0,00	0,30	0,00	9,56	29,89
thiz2-c3.dad		0,00	0,00	0,22	0,00	20,25	40,33
thiz2-c4.dad		0,94	0,00	0,27	0,00	13,30	37,25
thiz2-c5.dad		1,59	0,05	0,49	0,00	6,76	41,36
thiz2-d1.dad	16x50	1,83	0,00	0,38	0,00	13,19	27,92
thiz2-d2.dad		0,00	0,00	0,33	0,00	5,93	23,11
thiz2-d3.dad		0,00	0,00	0,19	0,00	11,05	25,08
thiz2-d4.dad		1,18	0,00	0,81	0,00	14,12	14,12
thiz2-d5.dad		0,00	0,00	0,17	0,00	7,61	41,27
thiz2-e1.dad	33x50	0,30	0,00	1,24	0,00	10,84	46,67
thiz2-e2.dad		0,87	0,00	1,48	0,00	11,81	32,34
thiz2-e3.dad		1,11	0,00	0,78	0,00	19,98	47,39
thiz2-e4.dad		0,45	0,00	0,93	0,00	14,38	28,48
thiz2-e5.dad		0,60	0,00	1,14	0,00	12,82	39,08
thiz2-f1.dad	50x50	0,24	0,00	0,84	0,00		36,27
thiz2-f2.dad		0,87	0,01	2,48	0,01		41,60
thiz2-f3.dad		0,00	0,00	0,58	0,00		45,38
thiz2-f4.dad		0,00	0,00	0,64	0,00		39,02
thiz2-f5.dad		0,00	0,00	0,62	0,00		42,78
thiz3-a1.dad	8x25	0,00	0,00	0,07	0,00	0,00	12,85
thiz3-a2.dad		0,00	0,00	0,06	0,00	12,10	22,81
thiz3-a3.dad		0,00	0,00	0,09	0,00	0,00	3,92
thiz3-a4.dad		0,00	0,00	0,06	0,00	8,97	16,11

thiz3-a5.dad		0,00	0,00	0,05	0,00	10,38	30,23
thiz3-b1.dad	16x25	0,00	0,00	0,23	0,00	2,90	39,29
thiz3-b2.dad		0,00	0,00	0,11	0,00	9,65	39,18
thiz3-b3.dad		2,11	0,00	0,26	0,00	3,13	39,35
thiz3-b4.dad		0,00	0,00	0,13	0,00	0,00	60,09
thiz3-b5.dad		0,00	0,00	0,17	0,00	0,00	39,47
thiz3-c1.dad	25x25	0,00	0,00	0,17	0,00	16,97	62,02
thiz3-c2.dad		0,00	0,00	0,33	0,09	0,69	53,54
thiz3-c3.dad		0,00	0,00	0,25	0,00	1,56	53,93
thiz3-c4.dad		1,42	0,00	0,49	0,00	17,82	52,53
thiz3-c5.dad		0,00	0,00	0,25	0,00	0,00	61,60
thiz3-d1.dad	16x50	0,23	0,00	0,47	0,00	6,90	33,88
thiz3-d2.dad		1,21	0,00	0,54	0,00	0,63	15,29
thiz3-d3.dad		0,13	0,00	0,33	0,00	0,00	26,03
thiz3-d4.dad		0,00	0,00	0,40	0,00	3,35	26,83
thiz3-d5.dad		0,00	0,00	0,19	0,00	0,00	41,84
thiz3-e1.dad	33x50	1,94	0,00	2,34	0,00	11,40	53,72
thiz3-e2.dad		2,14	0,10	2,52	0,00	12,37	54,61
thiz3-e3.dad		0,89	0,00	1,00	0,00	8,38	47,45
thiz3-e4.dad		0,00	0,00	0,71	0,00	13,02	40,10
thiz3-e5.dad		0,00	0,00	0,91	0,00	2,35	53,66
thiz3-f1.dad	50x50	0,81	0,00	2,25	0,00	4,20	64,06
thiz3-f2.dad		0,49	0,00	2,85	0,00		62,71
thiz3-f3.dad		0,88	0,00	4,73	0,00	11,71	44,40
thiz3-f4.dad		0,28	0,00	2,09	0,00		49,08
thiz3-f5.dad		0,70	0,00	3,47	0,00		44,47
thiz5-a1.dad	8x25	0,00	0,00	0,07	0,00	0,00	3,92
thiz5-a2.dad		0,00	0,00	0,05	0,00	3,98	10,64
thiz5-a3.dad		0,00	0,00	0,05	0,00	0,00	18,88
thiz5-a4.dad		0,00	0,00	0,09	0,00	0,00	11,90
thiz5-a5.dad		0,00	0,00	0,01	0,00	0,00	23,92
thiz5-b1.dad	16x25	0,00	0,00	0,10	0,00	0,00	59,50
thiz5-b2.dad		0,00	0,00	0,17	0,00	1,61	49,81
thiz5-b3.dad		0,00	0,00	0,26	0,00	4,55	37,36
thiz5-b4.dad		0,00	0,00	0,13	0,00	3,18	25,06
thiz5-b5.dad		0,00	0,00	0,16	0,00	0,00	40,53
thiz5-c1.dad	25x25	3,09	0,00	0,33	0,00	3,33	52,94
thiz5-c2.dad		1,40	0,00	0,38	0,00	13,31	32,80
thiz5-c3.dad		3,15	0,00	0,25	0,07	7,11	59,22
thiz5-c4.dad		0,00	0,00	0,25	0,00	0,00	64,85
thiz5-c5.dad		0,00	0,00	0,30	0,00	2,22	43,07
thiz5-d1.dad	16x50	0,00	0,00	0,20	0,00	0,00	30,36
thiz5-d2.dad		1,56	0,00	0,38	0,00	2,37	28,91
thiz5-d3.dad		0,00	0,00	0,28	0,00	0,00	31,11
thiz5-d4.dad		4,62	0,37	0,49	0,00	0,00	22,01
thiz5-d5.dad		0,88	0,00	0,41	0,00	1,76	25,55
thiz5-e1.dad	33x50	0,45	0,00	0,79	0,00	0,00	50,15
thiz5-e2.dad		0,00	0,00	0,56	0,00	5,46	68,73
thiz5-e3.dad		0,00	0,00	0,61	0,00	0,00	52,80
thiz5-e4.dad		0,00	0,00	0,64	0,00	1,93	69,54
thiz5-e5.dad		3,92	1,67	2,37	0,00	3,78	45,31

thiz5-f1.dad	50x50	0,80	0,00	2,11	1,13	8,56	64,41
thiz5-f2.dad		2,09	0,08	3,90	1,24	0,92	79,34
thiz5-f3.dad		2,57	0,05	5,70	0,01	17,38	58,11
thiz5-f4.dad		0,00	0,00	1,56	0,42	0,26	67,37
thiz5-f5.dad		1,92	0,00	4,66	2,02	1,37	52,14
thizx-a1.dad	8x25	0,00	0,00	0,06	0,00	0,00	17,68
thizx-a2.dad		0,00	0,00	0,03	0,00	0,00	24,90
thizx-a3.dad		0,00	0,00	0,01	0,00	0,00	22,09
thizx-a4.dad		0,00	0,00	0,06	0,00	2,10	3,32
thizx-a5.dad		0,00	0,00	0,07	0,00	0,00	1,91
thizx-b1.dad	16x25	0,00	0,00	0,09	0,00	0,00	22,37
thizx-b2.dad		0,00	0,00	0,11	0,00	0,00	35,20
thizx-b3.dad		0,00	0,00	0,08	0,00	0,00	51,02
thizx-b4.dad		0,00	0,00	0,03	0,00	0,00	58,71
thizx-b5.dad		0,00	0,00	0,01	0,00	0,00	34,97
thizx-c1.dad	25x25	0,00	0,00	0,14	0,00	0,00	48,54
thizx-c2.dad		1,02	0,00	0,16	0,00	5,39	64,99
thizx-c3.dad		4,21	1,35	0,29	0,00	3,05	41,00
thizx-c4.dad		2,70	0,00	0,43	0,00	1,55	51,62
thizx-c5.dad		5,04	3,04	0,34	0,00	8,41	80,28
thizx-d1.dad	16x50	0,08	0,00	0,23	0,00	0,00	18,02
thizx-d2.dad		2,12	0,00	0,26	0,00	0,00	7,52
thizx-d3.dad		0,00	0,00	0,30	0,00	0,00	19,63
thizx-d4.dad		0,00	0,00	0,04	0,00	0,00	21,89
thizx-d5.dad		1,97	0,00	0,78	0,00	0,00	27,11
thizx-e1.dad	33x50	0,25	0,00	0,86	0,00	10,50	62,96
thizx-e2.dad		0,00	0,00	0,59	0,00	0,00	50,59
thizx-e3.dad		1,39	0,00	1,05	0,00	0,00	37,67
thizx-e4.dad		1,99	0,00	0,72	0,00	7,98	48,65
thizx-e5.dad		0,00	0,00	0,29	0,00	0,94	58,59
thizx-f1.dad	50x50	1,60	0,00	1,96	0,00	1,07	77,06
thizx-f2.dad		0,00	0,00	0,86	0,00	10,22	81,07
thizx-f3.dad		3,42	0,00	1,47	0,00	0,00	76,16
thizx-f4.dad		0,66	0,00	1,47	0,10	0,00	75,74
thizx-f5.dad		3,34	0,00	2,62	0,22	5,62	63,11
Média		***0,65***	***0,05***	***0,68***	***0,04***	***5,53***	***36,27***

Tabela 24 - Solução heurística (problemas teste de Thizy)

Dimensões nxm	**Algoritmo proposto**	**Thizy**	**Jacobsen**	
		H	**VSM**	**DROP-HI**
	Seg.	**Seg.**	**Seg.**	**Seg.**
8x25	0,06	1,70	0,60	0,10
16x25	0,17	3,90	2,80	0,20
25x25	0,31	9,00	9,60	0,50
16x50	0,35	9,60	5,70	0,50
33x50	1,11	23,00	41,00	1,70
50x50	2,05	37,00	>92,00	3,70

Tabela 25 – Tempos computacionais Solução heurística (problemas teste de Thizy)

Para os problemas de Cornuejols, Sridharan e Thizy o método proposto obteve um erro médio de 0,05%. Somente o método proposto por Cornuejols, Sridharan e Thizy obteve um resultado um pouco melhor. De qualquer forma, isso não quer dizer que o método desenvolvido é pior do que o de Cornuejols, Sridharan e Thizy para esses conjuntos de problema. Isso porque, poder-se-ia perfeitamente estabelecer outro critério de parada, pois os tempos são muitos pequenos (menores do que 1 segundo), e explorar mais nós da árvore de busca.

A Tabela **26** mostra os resultados obtidos, para os problemas de Ardal [3]. Ela gerou 12 problemas, de dimensões 75x100, baseados no gerador de problemas de Cornuejols, Sridharan e Thizy. Z^h representa a solução gerada pelo método proposto quando se pára a exploração da árvore de busca nos nós com nível de busca igual a três. Mostra-se o erro em relação à solução ótima, o erro em relação ao limite inferior (LI), o qual é obtido pelo método, e o tempo em segundos obtido para encontrar Z^h. A solução ótima do problema é representada por Z^*. Não é realizada nenhuma comparação com outro método heurístico por não se ter conhecimento de alguém que usou essa base de testes. Ardal executou os testes somente para o algoritimo ótimo, na seção 4.9.3.3, será mostrado seus resultados juntamente com os resultados do algoritmo ótimo que foi proposto.

Problema	nXm	Algoritmo Proposto		
		$100*((Z^h - LI)/LI)$	$100*((Z^h - Z^*)/Z^*)$	Seg.
karen1-1.dad	75x100	0,41	0,11	16,73
karen1-2.dad	75x100	0,59	0,00	22,9
karen1-3.dad	75x100	0,13	0,00	2,42
karen1-4.dad	75x100	0,35	0,00	4,35
karen1-5.dad	75x100	0,13	0,00	3,42
karen2-1.dad	75x100	0,00	0,00	2,09
karen2-2.dad	75x100	0,28	0,00	2,92
karen2-3.dad	75x100	0,33	0,00	3,89
karen2-4.dad	75x100	0,40	0,10	28,55
karen2-5.dad	75x100	0,65	0,00	34,09
karen3-1.dad	75x100	0,56	0,00	32,56
karen3-2.dad	75x100	0,76	0,00	26,53
karen3-3.dad	75x100	0,62	0,00	9,18
karen3-4.dad	75x100	0,27	0,00	11,28
karen3-5.dad	75x100	0,98	0,00	51,46
Média		***0,43***	***0,01***	***16,82***

Tabela 26 Solução heurística (problemas teste de Ardal)

Como para os problemas anteriores, o método proposto demonstrou fornecer soluções com excelente qualidade. O método proposto obteve um erro médio de 0,01%. Somente em dois problemas o método proposto não conseguiu atingir a solução ótima, quando o mesmo é parado no término da exploração dos nós de nível de busca igual a três.

4.9.3.3 Solução ótima para os problemas da literatura

Será apresentado, nesta seção, os resultado obtidos pelo algoritmo exato proposto, isto é, será apresentado os resultado quando o algoritmo proposto explora toda a árvore de busca de forma explicita ou implícita. A máquina utilizada para obtenção dos resultados foi uma máquina Pentium III, 700 Mhz com 256 Mega bytes de memória principal. Para que possa ser verificado que o método utilizado para resolução do *PLC* é eficiente, serão realizadas comparações com o *software* comercial XPRESS 14.05. O XPRESS é rodado na mesma máquina: Assim é possível ter uma comparação real entre o método desenvolvido e o XPRESS.

A Tabela 27 mostra os resultados obtidos pelo método proposto, pelos métodos de Beasley [11], Van Roy [45], Baker B. M. e Baía A. P. [8] (os resultados apresentados, em seu artigo, na tabela 5) e pelo XPRESS sobre os problemas de Kühn

& Hamburger. Todos estes métodos encontram a solução ótima para o problema. As medidas mostradas são o número de nós explorados e o tempo em segundos para os métodos encontrarem a solução ótima. Beasley utilizou um Cray-1S, já Van Roy utilizou um MV8000 (***na*** indica os problemas não avaliados por Van Roy). Baker utilizou um CCI-PC 486DX (50 mhz) para encontrar seus resultados. A média Roy é referente somente aos problemas que o mesmo obteve resultados, ou seja, são eliminados os problemas onde consta ***na***, por exemplo, os problemas 51, 111, 112, 113, etc.

Problema	**nXm**	**Algoritmo Proposto**		**Beasley**		**Van Roy**		**XPRESS**		**Baker**	
		Nós	**Séc.**	**Nós**	**Seg.**	**Nós**	**Seg.**	**Nós**	**Seg.**	**Nós**	**Seg.**
41	16x50	1	0,19	0	1,2	-	0,24	1	0,30	3	6,81
42		1	0,18	0	1,0	-	0,25	1	0,40	2	7,47
43		1	0,22	13	1,9	3	1,50	7	0,50	2	7,80
44		3	0,21	0	0,9	-	0,97	7	0,50	2	7,31
51		3	0,22	0	1,2	na	na	7	0,30	10	3,68
61		3	0,08	0	0,3	-	0,38	1	0,10	0	1,09
62		1	0,15	0	0,5	-	0,39	1	0,10	0	0,98
63		7	0,18	7	1,1	6	1,92	1	0,30	30	4,61
64		1	0,15	0	0,3	15	2,71	3	0,20	2	4,23
71		3	0,08	0	0,2	-	0,25	1	0,10	0	0,93
72		7	0,08	0	0,2	2	1,33	1	0,10	0	0,99
73		9	0,08	0	0,2	2	1,43	1	0,20	0	1,15
74		3	0,08	0	0,1	2	1,37	1	0,30	0	1,05
81	25x50	15	0,49	33	3,8	-	0,33	1	0,20	68	11,92
82		7	0,41	5	1,8	12	2,26	10	0,30	20	9,28
83		9	0,39	9	1,8	26	3,53	1	0,20	16	10,27
84		25	0,84	41	4,0	118	6,37	17	0,40	60	16,75
91		9	0,13	0	0,8	-	0,31	1	0,10	0	1,59
92		5	0,56	0	0,9	-	1,66	1	0,20	20	5,38
93		13	0,40	23	2,4	10	3,09	33	0,30	30	6,43
94		11	0,45	57	3,1	41	4,93	43	0,30	128	13,45
101		9	0,12	0	0,4	38	2,46	1	0,10	0	1,59
102		11	0,13	0	0,4	5	5,47	1	0,20	0	2,26
103		15	0,14	0	0,5	34	7,55	1	0,30	0	4,73
104		5	0,14	0	0,2	132	26,44	18	0,40	0	1,70
111	50x50	9	0,60	3	2,4	na	na	5	0,20	12	12,41
112		9	0,86	37	4,3	na	na	21	0,50	100	26,59
113		27	1,32	129	8,7	na	na	344	2,00	304	48,00
114		41	1,68	151	8,8	na	na	454	2,70	298	56,96
121		29	0,32	0	1,5	na	na	1	0,20	0	6,48
122		13	0,67	11	2,3	na	na	190	0,90	38	16,97
123		17	0,78	49	3,8	na	na	1144	5,60	74	17,19
124		55	1,73	175	7,5	na	na	1636	8,60	602	60,52
131		29	0,27	0	0,8	na	na	1	0,20	0	5,82
132		1	0,48	0	0,6	na	na	94	0,60	0	11,54
133		31	0,27	0	0,6	na	na	869	4,20	0	8,46
134		17	0,27	0	0,7	na	na	853	4,40	0	6,64
Média		***12,30***	***0,41***	***20,1***	***1,9***			***156,03***	***0,99***	***49,22***	***11,11***
Média Roy		***7,25***	***0,25***			***18,58***	***3,21***				

Tabela 27 - Solução exata (problemas teste de Kühn & Hamburger)

É possível afirmar, para os problemas de Kühn & Hamburger, que o algoritmo proposto foi o que obteve em média o menor número de nós explorados. Também que é, em média, mais de 2 vezes mais rápido do que o XPRESS. Em alguns casos o XPRESS foi mais rápido que o método proposto, contudo, somente para os problemas onde o tempo de execução são muito baixos. Em nenhum desse problemas, o tempo do método proposto foi superior a 0,67 segundos. Fazendo a equivalência entre as máquinas, utilizando a Tabela 21, o Pentium III levou em média 0,41 segundos para resolver os problemas enquanto que o Cray-1S 8,64 segundos.

A Tabela 28 mostra os resultados obtidos pelo método algoritmo proposto, pelo método desenvolvido por Beasley [11] (algoritmo exato para resolução do *PLC*) e pelo XPRESS, para problemas gerados aleatoriamente por Beasley [11]. As medidas mostradas são o número de nós explorados e o tempo em segundos para os métodos encontrarem a solução ótima. Para o XPRESS foi utilizada uma terceira medida, o *gap* entre a melhor solução encontrada e o limite inferior do problema. Isso porque, não foi possível obter a solução ótima para os problemas. Beasley utilizou um Cray-1S para obtenção dos resultados.

Problema	**nXm**	**Algoritmo Proposto**		**Beasley**		**XPRESS**		
		Nós	**Seg.**	**Nós**	**Seg.**	**Nós**	**Seg.**	**gap**
A1	100x1000	45	140,04	43	87,4	>164630	>70000	17,50
A2	100x1000	15	111,77	39	79,9	>207175	>117326	26,41
A3	100x1000	15	109,83	27	59,3	>122748	>46930	29,90
A4	100x1000	3	47,35	-	28,1	>131226	>45540	35,56
B1	100x1000	15	52,30	7	74,2	>113345	>50400	31,77
B2	100x1000	143	385,78	309	321,6	>179560	>78026	38,36
B3	100x1000	151	447,94	257	244,9	>218347	>81360	40,14
B4	100x1000	25	151,69	47	89,5	>183142	>73819	54,65
C1	100x1000	37	102,41	59	150,6	>201501	>92690	39,42
C2	100x1000	145	398,29	237	294,7	>258150	>88168	36,85
C3	100x1000	23	109,73	45	108,7	>239582	>86060	45,48
C4	100x1000	7	37,70	7	87,7	>259968	>91778	54,93
Média		***52,00***	***174,57***	***89,7***	***135,5***			

Tabela 28 - Solução exata (problemas teste de Kühn & Hamburger)

É possível afirmar, para os problemas em questão, que o algoritmo proposto foi o que teve em média o menor número de nós explorados. Também que é muito mais eficiente do que o XPRESS. Em nenhum caso o XPRESS conseguiu chegar a solução ótima, pois o mesmo teve seu processamento abortado porque o tempo de computação

estava muito alto. Fazendo a equivalência entre as máquinas, utilizando a Tabela 21, o Pentium III levou em média 174,57 segundos para resolver os problemas enquanto que o Cray-1S 615,90 segundos.

A Tabela 29 mostra os resultados obtidos pelo método proposto e pelo XPRESS para os problemas gerados por Cornuejols, Sridharan e Thizy [20] . As medidas mostradas são o número de nós explorados e o tempo em segundos para os métodos encontrarem a solução ótima.

Problema	**nXm**	**Algoritmo Proposto**		**XPRESS**	
		Nós	**Seg.**	**Nós**	**Seg.**
thiz1-a1.dad	8x25	7	0,07	21	0,2
thiz1-a2.dad		21	0,09	21	0,2
thiz1-a3.dad		5	0,07	7	0,2
thiz1-a4.dad		1	0,07	1	0,1
thiz1-a5.dad		25	0,09	45	0,3
thiz1-b1.dad	16x25	53	0,30	267	0,5
thiz1-b2.dad		19	0,15	13	0,2
thiz1-b3.dad		27	0,16	113	0,6
thiz1-b4.dad		17	0,14	53	0,2
thiz1-b5.dad		187	0,56	735	0,8
thiz1-c1.dad	25x25	11	0,20	55	0,2
thiz1-c2.dad		33	0,28	48	0,2
thiz1-c3.dad		39	0,31	67	0,2
thiz1-c4.dad		31	0,24	75	0,2
thiz1-c5.dad		83	0,44	293	0,6
thiz1-d1.dad	16x50	13	0,30	65	0,6
thiz1-d2.dad		1	0,19	17	0,4
thiz1-d3.dad		13	0,34	87	0,8
thiz1-d4.dad		41	0,38	79	1,3
thiz1-d5.dad		7	0,25	27	0,5
thiz1-e1.dad	33x50	159	2,20	505	2,8
thiz1-e2.dad		107	1,53	295	1,6
thiz1-e3.dad		1	0,41	31	0,4
thiz1-e4.dad		257	2,05	876	4,2
thiz1-e5.dad		147	1,90	618	2,8
thiz1-f1.dad	50x50	81	1,19	158	0,8
thiz1-f2.dad		31	0,75	61	0,4
thiz1-f3.dad		81	1,17	567	2,3
thiz1-f4.dad		3	0,67	21	0,3
thiz1-f5.dad		1	0,75	1	0,1
thiz2-a1.dad	8x25	9	0,07	0	0,2
thiz2-a2.dad		47	0,11	77	0,2
thiz2-a3.dad		21	0,09	0	0,2
thiz2-a4.dad		5	0,06	0	0,4

thiz2-a5.dad		5	0,06	60	0,4
thiz2-b1.dad	16x25	19	0,18	17	0,6
thiz2-b2.dad		33	0,18	0	0,6
thiz2-b3.dad		17	0,16	75	0,6
thiz2-b4.dad		121	0,41	0	0,4
thiz2-b5.dad		3	0,12	110	0,3
thiz2-c1.dad	25x25	245	1,00	0	1,43
thiz2-c2.dad		73	0,32	175	0,6
thiz2-c3.dad		17	0,22	51	0,5
thiz2-c4.dad		35	0,26	117	0,4
thiz2-c5.dad		93	0,57	503	1,2
thiz2-d1.dad	16x50	35	0,42	112	1,1
thiz2-d2.dad		21	0,34	0	0,8
thiz2-d3.dad		1	0,19	0	0,3
thiz2-d4.dad		111	0,81	0	0,8
thiz2-d5.dad		1	0,18	1	0,1
thiz2-e1.dad	33x50	75	1,25	462	3,7
thiz2-e2.dad		165	1,58	623	3,9
thiz2-e3.dad		31	0,84	138	1,1
thiz2-e4.dad		111	0,95	194	2,1
thiz2-e5.dad		103	1,32	550	4,3
thiz2-f1.dad	50x50	33	0,90	233	1
thiz2-f2.dad		361	3,20	1872	12,4
thiz2-f3.dad		9	0,58	45	0,4
thiz2-f4.dad		1	0,63	33	0,7
thiz2-f5.dad		1	0,62	3	0,2
thiz3-a1.dad	8x25	3	0,06	0	0,4
thiz3-a2.dad		5	0,06	9	0,2
thiz3-a3.dad		11	0,08	0	0,2
thiz3-a4.dad		15	0,06	0	0,2
thiz3-a5.dad		1	0,05	0	0,2
thiz3-b1.dad	16x25	31	0,23	0	0,2
thiz3-b2.dad		7	0,11	0	0,2
thiz3-b3.dad		39	0,29	0	0,2
thiz3-b4.dad		15	0,13	0	0,2
thiz3-b5.dad		37	0,17	0	0,2
thiz3-c1.dad	25x25	1	0,17	0	0,5
thiz3-c2.dad		49	0,33	0	0,5
thiz3-c3.dad		23	0,26	0	0,8
thiz3-c4.dad		117	0,55	0	0,4
thiz3-c5.dad		23	0,24	0	0,6
thiz3-d1.dad	16x50	35	0,49	145	0,5
thiz3-d2.dad		53	0,55	167	0,9
thiz3-d3.dad		19	0,35	25	0,4
thiz3-d4.dad		31	0,40	101	1,1
thiz3-d5.dad		3	0,19	11	0,2
thiz3-e1.dad	33x50	215	2,86	764	6,8
thiz3-e2.dad		349	4,14	1059	7,1
thiz3-e3.dad		73	1,19	307	2,3
thiz3-e4.dad		35	0,72	141	0,9
thiz3-e5.dad		61	0,92	241	1,4

thiz3-f1.dad	50x50	123	2,39	612	3,9
thiz3-f2.dad		199	3,12	814	5,5
thiz3-f3.dad		553	6,05	1855	11,4
thiz3-f4.dad		161	2,28	660	3,8
thiz3-f5.dad		341	3,60	659	3,9
thiz5-a1.dad	8x25	11	0,07	1	0,1
thiz5-a2.dad		7	0,04	1	0,1
thiz5-a3.dad		3	0,06	1	0,1
thiz5-a4.dad		13	0,09	1	0,1
thiz5-a5.dad		3	0,00	1	0,1
thiz5-b1.dad	16x25	3	0,10	1	0,1
thiz5-b2.dad		9	0,17	1	0,1
thiz5-b3.dad		35	0,27	3	0,6
thiz5-b4.dad		7	0,13	1	0,2
thiz5-b5.dad		7	0,16	1	0,3
thiz5-c1.dad	25x25	57	0,41	15	1
thiz5-c2.dad		63	0,43	15	1
thiz5-c3.dad		23	0,26	10	0,6
thiz5-c4.dad		29	0,26	10	0,5
thiz5-c5.dad		21	0,30	15	1
thiz5-d1.dad	16x50	1	0,20	5	0,2
thiz5-d2.dad		31	0,41	214	2,2
thiz5-d3.dad		13	0,29	95	1
thiz5-d4.dad		41	0,59	254	3,6
thiz5-d5.dad		27	0,42	7	0,8
thiz5-e1.dad	33x50	35	0,79	13	1
thiz5-e2.dad		23	0,58	3	1
thiz5-e3.dad		21	0,60	7	1,1
thiz5-e4.dad		13	0,64	3	1,6
thiz5-e5.dad		185	3,26	215	7,5
thiz5-f1.dad	50x50	127	2,45	451	3,3
thiz5-f2.dad		337	6,71	1403	13,6
thiz5-f3.dad		847	13,02	6017	41,7
thiz5-f4.dad		67	1,58	158	1,9
thiz5-f5.dad		583	10,36	1958	14,1
thizx-a1.dad	8x25	3	0,05	0	0,1
thizx-a2.dad		3	0,03	1	0,1
thizx-a3.dad		7	0,01	1	0,1
thizx-a4.dad		5	0,07	0	0,1
thizx-a5.dad		7	0,07	25	0,2
thizx-b1.dad	16x25	5	0,09	0	0,3
thizx-b2.dad		9	0,11	0	4
thizx-b3.dad		5	0,08	0	0,2
thizx-b4.dad		1	0,03	33	0,2
thizx-b5.dad		1	0,01	3	0,1
thizx-c1.dad	25x25	3	0,14	9	0,2
thizx-c2.dad		11	0,17	0	0,4
thizx-c3.dad		39	0,40	155	0,9
thizx-c4.dad		29	0,50	199	1
thizx-c5.dad		43	0,47	0	0,5
thizx-d1.dad	16x50	15	0,24	101	0,6

thizx-d2.dad		13	0,27	183	0,9
thizx-d3.dad		11	0,30	135	0,8
thizx-d4.dad		3	0,05	23	0,2
thizx-d5.dad		15	0,79	97	0,6
thizx-e1.dad	33x50	23	0,85	283	2,1
thizx-e2.dad		1	0,58	203	1,3
thizx-e3.dad		31	1,06	379	2,5
thizx-e4.dad		19	0,77	385	2,9
thizx-e5.dad		3	0,29	31	0,6
thizx-f1.dad	50x50	35	2,08	341	3,5
thizx-f2.dad		15	0,89	55	1,11
thizx-f3.dad		85	2,52	631	6,8
thizx-f4.dad		35	1,50	598	7,1
thizx-f5.dad		361	7,98	7066	73
Média		***62,73***	***0,89***	***267,07***	***2,22***

Tabela 29 - Solução exata (problemas teste de Thizy)

O método proposto mostrou-se, em média, mais de 2,5 vezes mais rápido do que o XPRESS para os problemas de Cornuejols, Sridharan e Thizy [20]. Também teve o número de nós explorados menor do que o XPRESS. Dos 150 problemas o XPRESS foi mais rápido em apenas 12 problemas, são eles: thizx-d5.dad, thiz5-b2.dad, thiz2-f5.dad, thiz2-f3.dad, thiz2-d5.dad, thiz2-d4.dad, thiz2-b4.dad, thiz1-f5.dad, thiz1-f4.dad, thiz1-c4.dad, thiz1-c3.dad e thiz1-c2.dad. Vale observar que todos os problema, em que o XPRESS foi mais rápido, são de fácil resolução. Em nenhum deles o método proposto levou mais de 1 segundo para resolver, o mais demorado foi thiz2-d4.dad, o qual levou 0,81 segundos para ser resolvido. Dentre todos os problemas, da Tabela 29, o mais demorado foi thiz5-f3.dad, o qual necessitou de 12,87 segundos para ser solucionado.

A Tabela 30 mostra os resultados obtidos pelo método proposto, pelo método de Ardal [3] e pelo XPRESS para os problemas gerados por Ardal [3]. As medidas mostradas são o número de nós explorados e o tempo em segundos para os métodos encontrarem a solução ótima. Ardal utilizou uma máquina SUN Sparc ELC para obtenção dos resultados.

Problema	nXm	Algoritmo Proposto		XPRESS		Ardal	
		Nós	seg.	nós	seg.	Nós	seg.
karen1-1.dad	75x100	3231	36,79	4202	99,3	251	6851
karen1-2.dad	75x100	6143	88,08	11427	265,4	1253	33186
karen1-3.dad	75x100	65	2,47	271	7,4	17	712
karen1-4.dad	75x100	1943	20,47	5333	133,1	195	5514
karen1-5.dad	75x100	77	3,50	352	7,5	5	345
karen2-1.dad	75x100	11	2,16	297	5,8	9	234
karen2-2.dad	75x100	179	2,98	566	16	49	1590
karen2-3.dad	75x100	133	4,03	511	17	43	1225
karen2-4.dad	75x100	4839	52,52	9314	302,4	77	2240
karen2-5.dad	75x100	1713	35,90	5311	160,1	43	1407
karen3-1.dad	75x100	2637	59,03	336	12	207	5869
karen3-2.dad	75x100	1047	36,42	1534	52	127	5249
karen3-3.dad	75x100	211	9,40	1161	36,2	53	2888
karen3-4.dad	75x100	271	11,57	416	15	79	2988
karen3-5.dad	75x100	3923	113,48	9224	295	387	21318
Média		***1761,53***	***31,92***	***3350,33***	***94,95***	***186,33***	***6107,73***

Tabela 30 - Solução exata (problemas teste de Ardal)

O método proposto mostrou-se, em média, mais de 3 vezes mais rápido do que o XPRESS para os problemas de Ardal. Também teve o número de nós explorados menor do que o XPRESS. Diferentemente do todos os problemas e métodos anteriores o método de Ardal obteve vantagem em relação aos nós explorados. De qualquer forma, o método proposto demonstrou ser muito eficiente na resolução dos problemas propostos por Ardal, com média de 31,92 segundos. Fazendo a equivalência entre as máquinas, utilizando a Tabela 21, o Pentium III levou em média 31,92 segundos para resolver os problemas enquanto que a Sun Sparc ELC 86,76 segundos. Para os problemas de Ardal o número de nós explorado foi grande, contudo os tempos computacionais são pequenos. O grande número de nós explorados, provavelmente, decore porque os problemas teste têm muitos ótimos locais, pois os custos variáveis são semelhantes. Contudo, o problema de transporte, para estes problemas, é de fácil resolução, fazendo com que o tempo de exploração de um nó seja muito rápido.

4.9.3.4 Solução ε-ótima para os problemas da literatura

Será apresentado, nesta seção, os resultado obtidos pelo algoritmo ε-ótimo proposto. A máquina utilizada para obtenção dos resultados foi uma máquina Pentium III, 700 Mhz com 256 Mega bytes de memória principal. Vale observar que é uma máquina defasada para os dias de hoje. Para que possa ser verificado que o método

utilizado para resolução do *PLC* é eficiente, os resultado foram comparados com o método de Barahona e Chudak [9]. O ε-ótimo não foi aplicado aos probemas da seção 4.9.3.2, pelo fato de não se justifica a sua aplicação, já que o algoritmo ótimo resolve todos os problemas de forma eficiente.

A Tabela 31 mostra resultados do algoritmo ε-ótimo, comparando-os com os resultados do método heurístico de Barahona e Chudak [9]. Os problemas testes são obtidos conforme Cornuejols, Sridharan e Thizy [20]. Os problemas testes de Barahona e Chudak [9] não são exatamente iguais aos gerados para resolução pelo método proposto. Contudo, eles seguem a mesma estruturação dada em Cornuejols, Sridharan e Thizy [20]. Cada linha da Tabela 31 representa a média de 5 problemas testes. A coluna representa as dimensões do problema, a coluna 2 o fator $\frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j}$ = *1, 1.5, 3, 5, 10*, a coluna 3 representa o tempo, em segundo, para o algoritmo ε-ótimo encontrar a solução (). Por último, a coluna 4 representa o erro gerado pelo algoritmo de Barahona e Chudak [9] e a coluna 5 o tempo, em segundo, para o mesmo alcançar a solução. Barahona e Chudak [9] utilizaram um computador IBM RISC 6000/7043P-240 para obtenção dos resultados.

		Método proposto (ε-ótimo ≤ 0,5%)	Heurística Barahona	
|I|x|J|	**fator**	**Seg.**	**Erro(%)**	**seg**
300x300	1.5	29,65	0,87	497
300x300	2	24,96	0,68	240
300x300	3	84,11	0,74	241
300x300	5	109,10	0,85	274
300x300	10	2612,48	0,93	310
500x500	1.5	84,36	0,13	2306
500x500	2	80,20	0,65	758
500x500	3	74,52	0,67	730
500x500	5	70,80	0,86	752
500x500	10	10398,41	1,25	1694
800x800	1.5	174,03	0,65	5541
800x800	2	220,06	0,50	2191
800x800	3	218,25	0,64	1709
800x800	5	300,74	0,68	1891
800x800	10	1090,83	1,02	3462
1000x1000	1.5	229,91	1,07	9311
1000x1000	2	371,50	0,73	3291
1000x1000	3	337,52	0,46	2605
1000x1000	5	329,40	0,67	3354
1000x1000	10	3781,85	0,93	3495
Média		***1031,13***	***0,75***	***2232,6***

Tabela 31 - Solução ε-ótimo (problemas teste gerados pelo gerador de Thizy)

A Tabela 31 mostra que o método proposto apresentou excelentes resultados para problemas de grande dimensões. O algoritmo ε-ótimo conseguiu resolver problemas, considerado difíceis pela literatura, com 1000 variáveis binárias e 1 milhão de variáveis reais, muito rapidamente. Além de sua eficiência o mesmo garante que o erro máximo gerado não é maior do que 0,5%. Não foi possível encontrar um parâmetro de comparação para a máquina que Barahona e Chudak [9] utilizaram. Em www.spec.org é possível realizar uma comparação entre a máquina IBM RISC 6000 7043-260 200 MHz e a Pentium III 700 MHz. Essa comparação informa que o Pentium III 700 Mhz = **1,18** IBM RISC 6000 7043-260 200 MHz. Acredita-se que a máquina IBM RISC 6000/7043P-240 166MHz não tenha um desempenho muito menor do que a IBM RISC 6000 7043-260 200 MHz. Com isso, além do método de Barahona e Chudak

[9] fornecerem um erro maior do que o método proposto, os tempos computacionais indicam vantagens na maioria dos problemas para o método proposto (exceto para a maioria dos problemas de fator 10).

4.9.4 Solução ótima e ε-ótima para problemas não Euclidianos

Os problemas que serão apresentados, nesta seção, foram gerados conforme seção 4.9.1. O próprio nome do problema informa os parâmetros utilizados (veja seção 4.9.1). Procurou-se fazer, além da geração de problemas não Euclidianos, uma grande variação nos dados dos problemas testes. Não foi possível gerar as soluções de todos os problemas idealizados, por falta de tempo e infra-estrutura. De qualquer forma, os resultados gerados fornecem parâmetros para avaliação da potencialidade do método. Os problemas testes foram solucionados em 4 computadores diferentes. As tabelas, que apresentam os resultados computacionais, foram divididas em função do computador utilizado. Os problemas são resolvidos utilizando o algoritimo ótimo e o algoritmo ε-ótimo, ambos exploram toda a árvore de busca implicitamente ou explicitamente. A diferença é que o ε-ótimo realiza um corte quando o limite superior estiver a uma determinada distância do limite inferior, conforme visto na seção 4.7.

A Tabela 32 mostra os resultados obtidos pelos algoritmos ótimo e ε-ótimo para problemas com $|I| = 200$ e $|J| = 200$. Para o ε-ótimo utilizou-se o erro máximo de 0,5%. A primeira coluna mostra o problema gerado; a segunda coluna mostra o número de facilidades abertas na solução ótima; a terceira coluna mostra o número de nós explorados para obtenção da solução ótima; a quarta coluna mostra o tempo, em segundos, para o algoritmo ótimo resolver o problema; a quinta coluna mostra o erro gerado pelo algoritmo ε-ótimo, em relação à solução ótima; e a sexta coluna mostra o tempo, em segundos, para o algoritmo ε-ótimo resolver o problema teste. Os algoritmos foram executados em um computador Intel(R) Pentium(R) 4 CPU 1.80GHz. As células da Tabela 32, não preenchidas, indicam que não foi gerado resultados computacionais para os problemas

Problema	Algoritmo ótimo			Algoritmo ε-ótimo	
	abertas	Nós	seg.	erro (%)	seg.
plc_1,5_200x200_10_1-1.dat	57	1	3,40	0,00	4,31
plc_1,5_200x200_10_1-2.dat	60	1409	169,14	0,24	6,19
plc_1,5_200x200_10_1-3.dat	57	1041	147,12	0,19	3,57
plc_1,5_200x200_10_20-1.dat	57	3	10,52	0,00	11,24
plc_1,5_200x200_10_20-2.dat	61	2061	247,22	0,42	10,63
plc_1,5_200x200_10_20-3.dat	57	867	123,72	0,20	14,14
plc_1,5_200x200_10_5-1.dat	57	17	4,21	0,11	4,22
plc_1,5_200x200_10_5-2.dat	60	3235	326,82	0,12	7,00
plc_1,5_200x200_10_5-3.dat	57	939	133,62	0,20	4,68
plc_1,5_200x200_5000_1-1.dat	162	7	8,46	0,16	6,47
plc_1,5_200x200_5000_1-2.dat	165	11	14,80	0,34	5,44
plc_1,5_200x200_5000_1-3.dat	165	11	14,54	0,34	5,69
plc_1,5_200x200_5000_20-1.dat	57	93	13,30	0,04	4,09
plc_1,5_200x200_5000_20-2.dat	60	129	10,46	0,00	4,73
plc_1,5_200x200_5000_20-3.dat	57	931	161,00	0,24	6,76
plc_1,5_200x200_500_1-1.dat	71	817	298,28	0,03	5,75
plc_1,5_200x200_500_1-2.dat	79	6951	3515,53	0,07	53,32
plc_1,5_200x200_500_1-3.dat	75	15363	6420,49	0,07	117,87
plc_1,5_200x200_500_20-1.dat	57	93	13,25	0,04	4,14
plc_1,5_200x200_500_20-2.dat	60	129	10,25	0,00	4,79
plc_1,5_200x200_500_20-3.dat	57	931	161,71	0,24	6,85
plc_10_200x200_10_1-1.dat	15	309	57,63	0,21	4,95
plc_10_200x200_10_1-2.dat	19	18521	8161,98	0,12	1187,75
plc_10_200x200_10_1-3.dat	15	6101	3812,58	0,08	1147,83
plc_10_200x200_10_20-1.dat	15	114801	31740,99	0,12	20181,85
plc_10_200x200_10_20-2.dat					
plc_10_200x200_10_20-3.dat					6,30
plc_10_200x200_10_5-1.dat	15	71943	10657,46		
plc_10_200x200_10_5-2.dat	15	159	13,70	0,03	3,68
plc_10_200x200_10_5-3.dat	15	155	10,70	0,10	3,78
plc_10_200x200_5000_1-1.dat	83	9	6,37	0,20	3,57
plc_10_200x200_5000_1-2.dat	76	3	4,29	0,03	5,59
plc_10_200x200_5000_1-3.dat	79	9	7,34	0,00	7,29
plc_10_200x200_5000_20-1.dat	16	2987	985,73	0,04	197,42
plc_10_200x200_5000_20-2.dat	16	2709	929,80	0,04	58,57
plc_10_200x200_5000_20-3.dat	16	1111	323,06	0,19	5,44
plc_10_200x200_5000_5-1.dat	38	175	59,87	0,00	27,61
plc_10_200x200_5000_5-2.dat					4,59
plc_10_200x200_5000_5-3.dat	35	57	22,85	0,14	467,34
plc_10_200x200_500_1-1.dat	29	651	167,03	0,00	40,00
plc_10_200x200_500_1-2.dat	28	449	104,69	0,14	41,37
plc_10_200x200_500_1-3.dat	27	1551	355,41	0,03	76,21
plc_10_200x200_500_20-1.dat	16	2987	992,82	0,04	193,24
plc_10_200x200_500_20-2.dat	16	2709	921,29	0,04	59,60
plc_10_200x200_500_20-3.dat	16	1111	320,55	0,19	5,84
plc_10_200x200_500_5-1.dat	19	2085	682,49	0,00	247,36
plc_10_200x200_500_5-2.dat	19	35601	10408,35	0,00	2688,46
plc_10_200x200_500_5-3.dat	20	61859	17254,17	0,00	4147,79
plc_20_200x200_10_1-1.dat	9	891	416,24	0,02	106,54

plc_20_200x200_10_1-2.dat	9	2045	757,17	0,00	179,67
plc_20_200x200_10_1-3.dat	9	823	391,22	0,00	79,93
plc_20_200x200_10_20-1.dat	13	47613	20066,98	0,02	15134,25
plc_20_200x200_10_20-2.dat	13	42059	11693,78	0,09	5130,09
plc_20_200x200_10_20-3.dat	16	29559	12614,21	0,05	9486,31
plc_20_200x200_10_5-1.dat	17	5793	1393,93	0,26	3023,12
plc_20_200x200_10_5-2.dat	9	12923	1929,75	0,07	329,22
plc_20_200x200_10_5-3.dat	24	13725	904,47	0,21	9,49
plc_20_200x200_5000_1-1.dat	72	7	14,17	0,00	2,68
plc_20_200x200_5000_1-2.dat	68	3	6,93	0,28	7,84
plc_20_200x200_5000_1-3.dat	67	13	15,87	0,00	12,08
plc_20_200x200_5000_20-1.dat	9	1969	939,59	0,07	365,87
plc_20_200x200_5000_20-2.dat	9	50467	18924,45	0,00	7983,33
plc_20_200x200_5000_20-3.dat	9	803	336,23	0,00	147,96
plc_20_200x200_5000_5-1.dat					48,39
plc_20_200x200_5000_5-2.dat					3,96
plc_20_200x200_5000_5-3.dat					43,52
plc_20_200x200_500_1-1.dat	23	1515	366,11	0,00	321,62
plc_20_200x200_500_1-2.dat	21	75	22,89	0,00	7,92
plc_20_200x200_500_1-3.dat	23	967	269,90	0,00	416,22
plc_20_200x200_500_20-1.dat	9	1969	940,91	0,07	369,72
plc_20_200x200_500_20-2.dat	9	50467	18979,25	0,00	7971,30
plc_20_200x200_500_20-3.dat	9	803	352,39	0,00	147,80
plc_20_200x200_500_5-1.dat	13	8295	2745,71	0,00	1699,46
plc_20_200x200_500_5-2.dat	11	4371	1148,80	0,00	750,37
plc_20_200x200_500_5-3.dat	11	1897	512,04	0,00	362,69
plc_2_200x200_10_1-1.dat	45	6219	730,76	0,40	10,63
plc_2_200x200_10_1-2.dat	46	251	21,56	0,18	3,47
plc_2_200x200_10_1-3.dat	44	533	76,64	0,00	3,75
plc_2_200x200_10_20-1.dat	44	1143	154,48	0,30	11,66
plc_2_200x200_10_20-2.dat	46	1391	78,85	0,24	4,67
plc_2_200x200_10_20-3.dat	44	127	20,71	0,01	10,93
plc_2_200x200_10_5-1.dat	44	1161	154,16	0,25	7,28
plc_2_200x200_10_5-2.dat	46	1175	61,58	0,02	4,11
plc_2_200x200_10_5-3.dat	44	75	6,10	0,01	4,29
plc_2_200x200_5000_1-1.dat	145	39	34,24	0,02	12,78
plc_2_200x200_5000_1-2.dat	132	47	38,07	0,14	4,90
plc_2_200x200_5000_1-3.dat	137	7	10,78	0,14	5,14
plc_2_200x200_5000_20-1.dat	45	381	38,48	0,10	4,02
plc_2_200x200_5000_20-2.dat	46	65	9,66	0,02	3,99
plc_2_200x200_5000_20-3.dat	45	959	85,42	0,15	3,77
plc_2_200x200_5000_5-1.dat					45,43
plc_2_200x200_5000_5-2.dat					63,01
plc_2_200x200_5000_5-3.dat					44,39
plc_2_200x200_500_1-1.dat	64	18897	8684,73	0,12	90,38
plc_2_200x200_500_1-2.dat	64	59949	33670,69	0,02	262,79
plc_2_200x200_500_1-3.dat	62	81183	39280,92	0,03	349,40
plc_2_200x200_500_20-1.dat	45	381	38,62	0,10	4,14
plc_2_200x200_500_20-2.dat	46	65	9,45	0,02	4,03
plc_2_200x200_500_20-3.dat	45	959	85,89	0,15	3,81
plc_2_200x200_500_5-1.dat	49	6573	851,30	0,01	4,72

plc_2_200x200_500_5-2.dat	49	5521	927,52	0,04	4,33
plc_2_200x200_500_5-3.dat	49	17617	4437,99	0,22	5,06
plc_5_200x200_10_1-1.dat	25	2095	218,95	0,04	18,05
plc_5_200x200_10_1-2.dat	25	1249	92,76	0,03	3,55
plc_5_200x200_10_1-3.dat	25	4131	547,85	0,30	3,85
plc_5_200x200_10_20-1.dat	25	70629	10555,66	0,00	204,06
plc_5_200x200_10_20-2.dat	25	337	11,78	0,05	3,58
plc_5_200x200_10_20-3.dat	25	343743	74141,24	0,12	3712,28
plc_5_200x200_10_5-1.dat	27	38379	7995,77	0,13	142,88
plc_5_200x200_10_5-2.dat	25	201	12,54	0,08	3,43
plc_5_200x200_10_5-3.dat	25	297689	82393,90	0,16	2602,39
plc_5_200x200_5000_1-1.dat	91	49	32,29	0,00	4,12
plc_5_200x200_5000_1-2.dat	93	237	173,54	0,15	3,93
plc_5_200x200_5000_1-3.dat	97	9	9,38	0,13	4,03
plc_5_200x200_5000_20-1.dat	25	2949	423,39	0,02	24,64
plc_5_200x200_5000_20-2.dat					135,76
plc_5_200x200_5000_20-3.dat					39,37
plc_5_200x200_5000_5-1.dat					119,87
plc_5_200x200_5000_5-2.dat					227,56
plc_5_200x200_5000_5-3.dat					132,29
plc_5_200x200_500_1-1.dat	43	118059	38480,05	0,07	641,44
plc_5_200x200_500_1-2.dat	44	839	303,94	0,06	31,72
plc_5_200x200_500_1-3.dat	49	18679	6218,07	0,14	551,90
plc_5_200x200_500_20-1.dat	25	2949	412,28	0,02	24,71
plc_5_200x200_500_20-2.dat	24	4753	1126,61	0,02	34,39
plc_5_200x200_500_20-3.dat	31	463	97,40	0,30	4,45
plc_5_200x200_500_5-1.dat	29	2437	645,49	0,15	5,18
plc_5_200x200_500_5-2.dat	29	3375	1140,46	0,06	89,05
plc_5_200x200_500_5-3.dat	28	3733	1032,28	0,05	105,69
Média		***15337,78***	***4444,58***	***0,09***	***748,80***

Tabela 32 – Solução ótima e ε-ótima para problemas não Euclidianos de dimensões 200x200

A Tabela 33 coloca lado a lado, os resultados do algoritmo ótimo proposto com os resultado do XPRESS, para alguns problemas de 200x200, apresentados na Tabela 32, Para o algoritmo proposto são apresentados o número de nós e o tempo em segundos para se explorar toda a árvore de busca. Para o Xpress são apresentados o número de nós explorados e o tempo em segundos para se explorar toda a árvore de busca ou até ele ter consumido uma quantidade enorme de processamento, sem terminar de explorar a árvore de busca. Para os casos onde o Xpress foi parado, pelo tempo excessivo de processamento, são apresentados os intervalos entre o limite inferior e superior e a quantidade de nós ativos (nós que ainda deveriam ser explorados).

Problema	Algoritmo ótimo proposto		XPRESS			
	Nós	Seg.	Nós	seg.	gap	nós ativos
plc_1,5_200x200_10_1-1.dat	1	3,4	229	6,4	0	
plc_1,5_200x200_10_1-2.dat	1409	169,14	3976	284,2	0	
plc_1,5_200x200_10_1-3.dat	1041	147,12	4977	345,8	0	
plc_1,5_200x200_500_1-1.dat	817	298,28	823	63,3	0	
plc_1,5_200x200_500_1-2.dat	6949	3515,53	1452	1342	0	
plc_1,5_200x200_500_1-3.dat	15363	6420,49	4016	310,4	0	
plc_10_200x200_10_1-1.dat	309	57,63	49578	8709	0	
plc_10_200x200_10_1-2.dat	18521	8161,98	>498854	>100770	0,57	269048
plc_10_200x200_10_1-3.dat	6101	3812,58	364796	45879,6	0	
plc_20_200x200_10_1-1.dat	891	416,24	>1149235	>109303	1,78	544277
plc_20_200x200_10_1-2.dat	2043	757,17	>931244	>104800	2,46	378300
plc_20_200x200_10_1-3.dat	822	391,22	>1797292	>157861	1,91	654070
plc_2_200x200_10_1-1.dat	6217	730,76	15538	1335,4	0	
plc_2_200x200_10_1-2.dat	251	21,56	718	108,6	0	
plc_2_200x200_10_1-3.dat	531	76,64	2033	166,5	0	
plc_5_200x200_10_1-1.dat	2093	218,95	11268	1556,7	0	
plc_5_200x200_10_1-2.dat	1249	92,76	6087	749,8	0	
plc_5_200x200_10_1-3.dat	4131	547,85	70197	12532,3	0	

Tabela 33 – Solução ótima para problemas não Euclidianos de dimensões 200x200

A Tabela 34 mostra os resultados obtidos pelos algoritmos ótimo e ε-ótimo para problemas com $|I| = 300$ e $|J| = 300$. Para o ε-ótimo utilizou-se o erro máximo de 0,5%. A primeira coluna mostra o problema gerado; a segunda coluna mostra o número de facilidades abertas na solução ótima; a terceira coluna mostra o número de nós explorados para obtenção da solução ótima; a quarta coluna mostra o tempo, em segundos, para o algoritmo ótimo resolver o problema; a quinta coluna mostra o erro gerado pelo algoritmo ε-ótimo, em relação à solução ótima; e a sexta coluna mostra o tempo, em segundos, para o algoritmo ε-ótimo resolver o problema teste. Os algoritmos foram executados em um computador Intel(R) Xeon(TM) MP CPU 1.50GHz. As células da Tabela 34, não preenchidas, indicam que não foi gerado resultados computacionais para os problemas.

Problema	Algoritmo ótimo			Algoritmo ε-ótimo	
	abertas	Nós	seg.	erro (%)	seg.
plc_1,5_300x300_10_1-1.dat	84	33	26,55	0,02	33,74
plc_1,5_300x300_10_1-2.dat	88	1	22,19	0,00	36,8
plc_1,5_300x300_10_1-3.dat	89	4683	2024,33	0,07	49,39
plc_1,5_300x300_10_20-1.dat	84	1137	949,56	0,01	98,24
plc_1,5_300x300_10_20-2.dat	89	555	467,93	0,36	93,73
plc_1,5_300x300_10_20-3.dat	89	5265	2739,12	0,16	70,89
plc_1,5_300x300_10_5-1.dat	84	359	128,15	0,04	29,37
plc_1,5_300x300_10_5-2.dat	88	101	48,04	0,21	32,65
plc_1,5_300x300_10_5-3.dat	88	5807	4343,34	0,23	33,83
plc_1,5_300x300_5000_1-1.dat					42,22
plc_1,5_300x300_5000_1-2.dat					43,08
plc_1,5_300x300_5000_1-3.dat					35,7
plc_1,5_300x300_5000_20-1.dat					29,26
plc_1,5_300x300_5000_20-2.dat					39,95
plc_1,5_300x300_5000_20-3.dat					38,17
plc_1,5_300x300_5000_5-1.dat					31,55
plc_1,5_300x300_5000_5-2.dat					33,73
plc_1,5_300x300_5000_5-3.dat					27,51
plc_1,5_300x300_500_1-1.dat					36,56
plc_1,5_300x300_500_1-2.dat					28,88
plc_1,5_300x300_500_1-3.dat					29,18
plc_1,5_300x300_500_20-1.dat	84	1	25,20	0,00	21
plc_1,5_300x300_500_20-2.dat	88	1169	857,28	0,24	29,75
plc_1,5_300x300_500_20-3.dat	89	987	141,73	0,27	20,66
plc_1,5_300x300_500_5-1.dat	88	62397	48594,82	0,02	27,76
plc_1,5_300x300_500_5-2.dat	90	3013	3220,88	0,05	43,81
plc_1,5_300x300_500_5-3.dat	89	3453	3709,33	0,04	28,94
plc_10_300x300_10_1-1.dat	23	901	2018,10	0,24	71,47
plc_10_300x300_10_1-2.dat	23	475	648,35	0,21	25,47
plc_10_300x300_10_1-3.dat					182,79
plc_10_300x300_10_20-1.dat					112,53
plc_10_300x300_10_20-2.dat					33,52
plc_10_300x300_10_20-3.dat					28,2
plc_10_300x300_10_5-1.dat					57,34
plc_10_300x300_10_5-2.dat					22,69
plc_10_300x300_10_5-3.dat					25,57
plc_10_300x300_5000_1-1.dat					10761,16
plc_10_300x300_500_1-1.dat					10784,29
plc_10_300x300_500_1-2.dat					29855
plc_10_300x300_500_1-3.dat					17783,92
plc_10_300x300_500_20-1.dat					71,43
plc_10_300x300_500_20-2.dat					733,09
plc_10_300x300_500_20-3.dat					29,75
plc_20_300x300_10_1-1.dat					347,48
Média		**1922,32**	**4115,58**	**0,13**	**1636,18**

Tabela 34 – Solução ótima e ε-ótima para problemas não Euclidianos de dimensões 300x300

A Tabela 35 mostra os resultados obtidos pelos algoritmos ótimo e ε-ótimo para problemas com $|I| = 500$ e $|J| = 500$. Para o ε-ótimo utilizou-se o erro máximo de 0,5%. A primeira coluna mostra o problema gerado; a segunda coluna mostra o número de facilidades abertas na solução ótima; a terceira coluna mostra o número de nós explorados para obtenção da solução ótima; a quarta coluna mostra o tempo, em segundos, para o algoritmo ótimo resolver o problema; a quinta coluna mostra o erro gerado pelo algoritmo ε-ótimo, em relação à solução ótima; e a sexta coluna mostra o tempo, em segundos, para o algoritmo ε-ótimo resolver o problema teste. Os algoritmos foram executados e um computador Intel(R) Xeon(TM) CPU 3.40GHz . As células da Tabela 35, não preenchidas, indicam que não foi gerado resultados computacionais para os problemas.

Problema	Algoritmo ótimo			Algoritmo ε-ótimo	
	abertas	Nós	seg.	erro (%)	seg.
plc_1,5_500x500_10_1-1.dat	144	223	102,36	0,38	12,18
plc_1,5_500x500_10_1-2.dat	141	699	186,16	0,38	12,25
plc_1,5_500x500_10_1-3.dat	140	727	510,50	0,34	16,08
plc_1,5_500x500_10_20-1.dat	146	489	147,44	0,50	65,52
plc_1,5_500x500_10_20-2.dat	142	703	469,03	0,24	90,68
plc_1,5_500x500_10_20-3.dat	141	791	285,49	0,42	66,13
plc_1,5_500x500_10_5-1.dat	144	215	146,27	0,39	16,82
plc_1,5_500x500_10_5-2.dat	141	225	248,51	0,37	23,93
plc_1,5_500x500_10_5-3.dat	141	819	561,52	0,33	15,53
plc_1,5_500x500_500_1-1.dat					22,21
plc_1,5_500x500_500_1-2.dat					23,34
plc_1,5_500x500_500_1-3.dat					21,15
plc_1,5_500x500_500_20-1.dat	144	191	79,59	0,00	16,19
plc_1,5_500x500_500_20-2.dat	141	809	775,98	0,16	20,38
plc_1,5_500x500_500_20-3.dat	142	593	158,33	0,41	14,99
plc_1,5_500x500_500_5-1.dat					19,14
plc_1,5_500x500_500_5-2.dat					20,33
plc_1,5_500x500_500_5-3.dat					21,60
plc_10_500x500_10_1-1.dat					73,80
plc_10_500x500_10_1-2.dat					15,36
plc_10_500x500_10_1-3.dat					14,91
plc_10_500x500_10_20-1.dat					15,10
plc_10_500x500_10_20-2.dat					19,56
plc_10_500x500_10_20-3.dat					17,64
plc_10_500x500_10_5-1.dat					17,97
plc_10_500x500_10_5-2.dat					20,92
plc_10_500x500_10_5-3.dat					95,03
plc_20_500x500_10_1-1.dat	22	5007	5804,78	0,17	259,62
plc_20_500x500_10_1-2.dat	22	27683	33555,94	0,25	1863,10
plc_20_500x500_10_1-3.dat					5183,77
plc_20_500x500_10_20-1.dat					1155,65
Média		***2798,14***	***3073,71***	***0,31***	***298,42***

Tabela 35 – Solução ótima e ε-ótima para problemas não Euclidianos de dimensões 500x500

Da mesma forma que a Tabela 35, a Tabela 36 mostra os resultados obtidos pelos algoritmos ótimo e ε-ótimo para problemas com $|I| = 500$ e $|J| = 500$. As tabelas foram divididas porque dos problemas foram executados em máquinas diferentes. Para o ε-ótimo utilizou-se o erro máximo de 0,5%. A primeira coluna mostra o problema gerado; a segunda coluna mostra o número de facilidades abertas na solução ótima; a terceira coluna mostra o número de nós explorados para obtenção da solução ótima; a quarta coluna mostra o tempo, em segundos, para o algoritmo ótimo resolver o problema; a quinta coluna mostra o erro gerado pelo algoritmo ε-ótimo, em relação à

solução ótima; e a sexta coluna mostra o tempo, em segundos, para o algoritmo ε-ótimo resolver o problema teste. Os algoritmos foram executados em um computador AMD Opteron(tm) Processor 252. As células da Tabela 36, não preenchidas, indicam que não foi gerado resultados computacionais para os problemas.

Problema	**Algoritmo ótimo**			**Algoritmo ε-ótimo**	
	abertas	**nós**	**seg.**	**erro (%)**	**seg.**
plc_2_500x500_10_1-1.dat	112	3161	548,63	0,00	12,43
plc_2_500x500_10_1-2.dat	110	17563	15634,68	0,40	11,31
plc_2_500x500_10_1-3.dat	110	9	14,12	0,00	9,88
plc_2_500x500_10_20-1.dat	111	3085	365,94	0,00	21,47
plc_2_500x500_10_20-2.dat	120	293	54,32	0,00	20,73
plc_2_500x500_10_20-3.dat	119	273	51,67	0,00	47,39
plc_2_500x500_10_5-1.dat	112	1757	96,44	0,13	13,48
plc_2_500x500_10_5-2.dat					10,28
plc_2_500x500_10_5-3.dat	110	1005	72,92	0,02	12,57
plc_2_500x500_500_1-1.dat					34,01
plc_2_500x500_500_1-2.dat					17,34
plc_2_500x500_500_1-3.dat					1432,98
plc_2_500x500_500_20-1.dat	115	46237	29024,68	0,28	11,01
plc_2_500x500_500_20-2.dat	111	4711	1546,89	0,01	11,02
plc_2_500x500_500_20-3.dat	123	189	36,20	0,31	11,79
plc_2_500x500_500_5-1.dat					13,54
plc_2_500x500_500_5-2.dat					11,91
plc_2_500x500_500_5-3.dat					12,57
plc_5_500x500_10_1-1.dat					10,35
plc_5_500x500_10_1-2.dat					9,82
plc_5_500x500_10_1-3.dat					23,39
plc_5_500x500_10_20-1.dat					27,98
plc_5_500x500_10_20-2.dat					26,04
plc_5_500x500_10_20-3.dat					11,70
plc_5_500x500_10_5-1.dat					9,90
plc_5_500x500_10_5-2.dat					10,89
plc_5_500x500_10_5-3.dat					9,56
plc_5_500x500_500_1-2.dat					74717,46
plc_5_500x500_500_20-1.dat					2,29
plc_5_500x500_500_20-2.dat					2,37
plc_5_500x500_500_20-3.dat	61	38649	7495,56	0,43	5,39
plc_5_500x500_500_5-1.dat					4,79
plc_5_500x500_500_5-2.dat					5,63
plc_5_500x500_500_5-3.dat	64	81519	53538,01	0,13	4,74
Média		***15265,46***	***8344,62***	***0,13***	***2252,88***

Tabela 36 – Solução ótima e ε-ótima para problemas não Euclidianos de dimensões 500x500

Os problemas resolvidos nesta seção destacam-se por se diferenciarem dos problemas encontrados na literatura. As principais diferenças estão nas características não Euclidianas e nas dimensões dos problemas resolvidos. Estas características tornam o problema muito mais difícil de ser resolvido. Outro fato, também importante, são as diversas variações realizadas nos dados dos problemas. Com isso, e pelos resultados computacionais apresentados, o método mostrou-se capaz de resolver problemas anda não explorados com complexidade elevada e de grandes dimensões. A variação nos dados dos problemas indicam que o método desenvolvido é bastante robusto. Essa robustez, provavelmente, deve-se ao fato do método atacar o problema de diversas formas: quatro testes de redução, dois limites inferiores, heurísticas, busca em largura e profundidade, entre outras.

Capítulo 5

5 Conclusões

Apesar de que na maioria das vezes os testes de redução não conseguirem determinar uma solução ótima de um *PLC*, as principais aplicabilidades deles são em relação à redução do problema original e ao direcionamento das ramificações, no *Branch and Bound*. O principal problema da falta de sucesso do *O-Teste* é quando $K_1 \cup K_2$ é grande demais, o que gera uma insensibilidade do custo de transporte ao retirar-se uma facilidade, ocasionando assim a falha do teste. O principal problema da falta de sucesso do *C-Teste* é quando K_1 é muito pequeno, o que torna muito sensível o custo de transporte quando é inserida uma facilidade, ocasionando assim a falha do teste. O sucesso de um teste influência positivamente no sucesso do outro.

O fato de estar sendo desenvolvido um método para um problema não polinomial, traz a necessidade de que ocorram cortes eficientes. Com esse objetivo, dois problemas que geram limites inferiores foram desenvolvidos no Capítulo 3. São eles o *PLCRL* e o *PLCRI*. O primeiro utiliza-se da relação Lagrangeana nas restrições de capacidades e de demanda e o segundo da relaxação da restrição de integridade. Para se obter uma boa solução para o *PLCRL* é utilizada otimização por subgradientes, já para se resolver o *PLCRI* é utilizado o algoritmo de transporte desenvolvido no Anexo 1. O limite inferior *PLCRL* é uma ótima ferramenta para realização de cortes e normalmente é mais robusto, em termos da qualidade da solução, do que o *PLCRI*, contudo é mais caro computacionalmente. O *PLCRI* demonstrou ser um bom limite inferior nos casos onde as diferenças entre os custos variáveis são pequenas (esses tipos de problemas tendem a gerarem muitos ótimos locais). Isso porque, os resultados computacionais mostraram que o *PLCRI* ficou muito próximo do valor da solução ótima. Além de o *PLCRI* ser um limite inferior, o mesmo pode obter a solução ótima do *PLC*. Vale observar que, embora o *PLCRL* normalmente forneça um limite mais próximo do ótimo do que o *PLCRI*, a tendência é que a medida que forem sendo fixados valores a K_0 e K_1 a diferença entre os dois limites vá caindo.

O *PLCRL* é calculado no inicio do algoritmo *Branch and Bound,* utilizando otimização por subgradientes. Após esse cálculo o mesmo é atualizado toda vez que

uma facilidade for fixada em zero ou em um. Novas otimizações por sub gradientes somente são calculadas em ramificações caso o valor do *PLCRI* esteja muito distante do melhor limite superior encontrado ou, o valor do *PLCRI,* esteja muito distante do *PLCRL*. O *PLCRI*, por ser mais barato, computacionalmente falando, é calculado no inicio do algoritmo, a cada nova ramificação e após os testes de redução serem realizados, se os mesmos tiverem fixado o valor de pelo menos uma variável.

O *PLCRL* e o *PLCRI* além de serem limites inferiores do *PLC*, fornecendo maneiras para que se possam realizar cortes, fornecem também insumos para a realização de testes de redução. Foram desenvolvidos quatro testes de redução, o *O-RLTeste* e *C-RLTeste* que utilizam informações do *PLCRL* e o *O-RITeste* e *C-RITeste* que utilizam informações do *PLCRI*. O *O-RLTeste*, *C-RLTeste*, *O-RITeste* e *C-RITeste* abrem/fecham facilidades, da mesma forma que o *O-Teste* e *C-Teste*, reduzindo a dimensão do problema. O *O-RLTeste*, *C-RLTeste* demonstraram bons resultados de forma geral, conforme resultados computacionais apresentados na seção 3.5. *O-RITeste* e *C-RITeste* são bastante poderosos quando o *PLCRI* estiver próximo da solução ótima.

O *O-RLTeste* e o *C-RLTeste* por serem testes baratos, computacionalmente falando, são aplicados em todos o nós da árvore de busca. Já o *O-RITeste* e o *C-RITeste,* por terem a dependência de *PLCRI* e não serem tão baratos quanto o *O-RLTeste* e o *C-RLTeste,* são aplicados a partir de um indicativo, pré-estabelecido, de possível sucesso dos mesmos, isto é, possível abertura ou fechamento de facilidades na aplicação dos testes.

Um importante fato observado foi que os testes *O-RITeste, C-RITeste, O-RITeste* e *C-RITeste* atacam um conjunto parecido de facilidades, isto é, abrem em fecham um conjunto parecido. Contudo eles diferem do *O-Teste* e do *C-Teste*, esses por sua vez abrem e fecham um conjunto diferente de facilidades do que os testes de redução baseados em limites inferiores. Esse fato é importante para que ocorram muitas podas na árvore de busca.

O *O-Teste* e o *C-Teste* abrem/fecham facilidades *a priori*. Contudo para o bom desempenho eles necessitam da abertura/fechamento das facilidades *a priori*. Este ciclo vicioso é quebrado pelo método *Branch and Bound*. O método *Branch and Bound*

desenvolvido utiliza-se: dos testes de redução *O-Teste*, *C-Teste, O-RLTeste*, *C-RLTeste*, *O-RITeste* e *C-RITeste*; dos limites inferiores *PLCRL*, esse baseado na relaxação lagrangeana, e *PLCRI*, esse baseado na relaxação da restrição de integridade; e dos números mínimo e máximo de facilidades abertas na solução ótima para fazer podas na árvore de busca. Quando os testes de redução falham o processo de ramificação (*branching*) abre e fecha as facilidades até que os mesmos voltem a funcionar ou ocorrer uma poda. Utilizou-se uma combinação entre busca em profundidade e busca em largura como estratégia de exploração da árvore. As regras de *branching* foram baseadas em heurísticas decorrentes dos testes de redução *O-Teste* e *C-Teste* e dos limites inferiores desenvolvidos. As regras, juntamente com a estratégia de busca, fornecem uma boa estratégia de descida na árvore, proporcionando uma solução viável próxima da ótima no início da exploração. Isso permite que o método funcione como uma heurística. O limite inferior do problema, que é calculado utilizando relaxação Lagrangeana e relaxação da restrição de integridade, juntamente com a melhor solução viável encontrada pelo método, fornecem uma medida da qualidade da solução, proporcionando assim ao usuário a possibilidade de parar a execução do método antes mesmo de ser encontrado a solução ótima.

As regras de *branching* utilizadas são baseadas na *O-* e *C-Heurística* desenvolvidas por Jacobsen [30]. A *O/C-Heurística* procura sanar as falhas das regras de Jacobsen fazendo um balanceamento entre as duas heurísticas para a tomada de uma decisão mais precisa quando K_1 for viável. A *O/C-Heurística* leva em conta a mínima e máxima sensibilidade dos custos de transporte, quando uma facilidade é aberta, e as comparam com o custo fixo. Para o caso de K_1 ser inviável foram desenvolvidas a *CMax-* e a *CMin-Heurística* que implica no fechamento/abertura de uma determinada facilidade. Além do *O-* e *C-Teste* fornecerem informações relativas a tendência quanto a uma facilidade ser aberta ou fechada, através das heurísticas *O/C-Heurística*, *CMin-Heurística* e *CMax-Heuristica*, os limites inferiores também fornecem tais informações. Assim sendo, foram incorporadas as informações fornecidas pelos limites inferiores as estratégias de ramificação. O *PLCRI* fornece a informação de quais facilidades devem ser consideradas nas regras de ramificação. São elas $\{i \in I \mid 0 < y_i < 1\}$. O *PLCRL* fornece um peso para as regras de ramificação. Como as informações dos limites inferiores as heurísticas *O/C-Heurística*, *CMin-Heurística* e *CMax-Heuristica*

transformam se, respectivamente em, *O/C-HeurísticaLI*, *CMin-HeurísticaLI* e *CMax-HeurísticaLI*.

O objetivo principal do trabalho foi resolver problemas de grande porte e obter a solução ótima ou muito próxima dela em um tempo viável. Como o *PLC* é um problema típico de planejamento, normalmente, não é necessário a obtenção de uma solução em segundos ou minutos. Com dito anteriormente, na maioria dos casos, é muito mais valioso se ter uma solução com boa qualidade em 48 horas do que ter uma solução com qualidade discutível em alguns segundos. Com isso, o foco do trabalho foi encontrar a melhor solução, ou muito próxima dela, para o tempo viável da instituição em questão. Isto é, um método que encontre a solução ótima para o problema ou realize o máximo de esforço possível, considerando o tempo viável da instituição, fornecendo uma solução com boa qualidade. Quando o método não fornecer a solução ótima o mesmo fornecerá um parâmetro de qualidade, a máxima distância que a solução atual está da ótima.

Além de o método funcionar como algoritmo ótimo e heurístico o mesmo pode ser utilizado com um algoritmo ε-ótimo. Isto é, pode-se estabelecer a máxima distância que a solução, a ser fornecida pelo método, tenha da solução ótima. O algoritmo ε-ótimo é muito importante para os problemas que tenham muitas soluções próximas da ótima. Para esses problemas é difícil o sucesso das podas, tornado o método ótimo ineficiente. Também pode ser utilizada nos problemas em que os dados tenham uma margem de erro em seu levantamento. Talvez não seja interessante buscar o ótimo de um problema onde nem os dados do mesmo têm essa precisão.

Os resultados computacionais demonstraram que o algoritmo conseguiu encontrar rapidamente uma solução próxima da ótima nas primeiras buscas em profundidade, sendo melhor, na maioria dos casos, em termos de qualidade, do que as soluções obtidas pelos melhores métodos heurísticos encontrados na literatura. Uma das características importantes, dessa comparação da solução gerada através de "heurística", é que o método foi testado sobre todos os problemas da literatura a que se teve acesso, diferentemente da maioria das heurísticas que são normalmente apresentadas, as quais são testadas apenas em uma base de problemas.

O algoritmo ótimo desenvolvido não teve problemas para resolver os problemas encontrados na literatura, as soluções foram obtidas muito rapidamente. O método, para os problemas de Kühn & Hamburger [32], obteve a média de 0,43 segundos, para os problemas de Beasley [11], obteve a média de 172,83 segundos, para os problemas de Cornuejols, Sridharan e Thizy [20], obteve a média de 30,87 segundos e para os problemas de Ardal [3], obteve a média de 31,56 segundos. Em todas as bases de problemas o método proposto foi, em média, mais rápido do que o XPRESS, sendo que no mínimo foi 2 vezes mais rápido. Nos problemas maiores, os de Bealesy, o XPRESS foi extremamente ineficiente. Para esses problemas o XPRESS sempre foi parado antes de encontrar a solução ótima, porque seu tempo de computação estava alto demais. Em alguns poucos problemas o XPRESS foi mais rápido que o método proposto, contudo somente para os problemas muito fáceis de serem resolvidos. Exceto para os problemas gerados por Ardal [3] e para seu método, o método proposto explorou, em média, menos nós do que os outros métodos comparados. Da mesma forma, que na solução gerada pelas primeiras buscas em profundidade, o método ótimo foi testado sobre todos os problemas da literatura que se teve acesso, diferentemente da maioria dos outros métodos, os quais são testados apenas em uma ou duas bases de problemas teste. Em relação a comparação de tempos computacionais, com os outros métodos da literatura, não é possível afirmar algo preciso. Contudo, as referências utilizadas indicam que o método proposto é mais rápido, em média, do que os algoritmos de Ardal [3] e Beasley [11].

Um dos resultados mais expressivos foi alcançando resolvendo problemas teste de grandes dimensões, considerados difíceis pela literatura, gerados conforme Cornuejols, Sridharan e Thizy [20]. Foram resolvidos problemas testes com até mil variáveis binárias e um milhão de variáveis contínuas. Os problemas foram resolvidos com o algoritmo ε-ótimo, onde o erro máximo admitido foi 0,5%. De uma forma geral, conforme Tabela 31, o algoritmo proposto apresentou resultados melhores do que o algoritmo de Barahona e Chudak [9].

Dado que o método proposto apresentou, para os problemas encontrados na literatura, uma solução heurística com qualidade nas primeiras buscas em profundidade e ser foi muito eficiente na resolução ótima e ε-ótima dos problemas , conclui-se que os

objetivos 1, 3, 4 e parte do 2 e 5, apresentados na seção 1.3, foram concluídos com sucesso.

O algoritmo proposto resolveu, de forma ótima e ε-ótima, problemas com características ainda não exploradas na literatura, aumentado com isso a tratabilidade do problema. Os problemas resolvidos são não Euclidianos, com até 500 variáveis binárias e 250.000 variáveis contínuas. O método se mostrou bastante robusto, isso porque, foram realizadas diversas variações nos dados dos problemas resolvidos. Os tempos computacionais são eficientes, considerando a complexidade e as dimensões dos problemas resolvidos. Com isso, os objetivos 2 e 5, apresentados na seção 1.3, são atingidos por completo. Logo os objetivos desse trabalho foram alcançados.

O Problema de Transporte Capacitado (*PTC*) é gerado diversas vezes no método de resolução do Problema de Localização Capacitado (*PLC*). É possível considerar que, no método desenvolvido para se resolver o *PLC*, o *PTC* é uma das principais ferramentas. O *PTC* foi utilizado na realização de quatro testes de redução e no cálculo de um dos limites inferiores. Por causa dessa dependência, foi desenvolvido, no Anexo 1, um método para resolução do *PTC*. O método utiliza o Primal Simplex para encontrar a solução do problema e uma árvore geradora para representar uma solução básica. Sáo utilizadas as variáveis do problema dual para facilitar o cálculo do custo reduzido. Também é armazenada a informação da altura dos nós na árvore geradora. Com essa informação é possível detectar mais facilmente o ciclo, formado pela inclusão do arco de entrada. Provavelmente a mudança mais significativa, em relação os métodos encontrados na literatura, foi a inclusão da lista de prioridade, utilizando a estrutura de dados *heap*, para armazenar os custos reduzidos negativos. Essa estrutura facilitou a procura por uma variável para entrar na base.

As comparações, realizadas entre o método desenvolvido para resolução do *PTC* e o XPRESS, mostraram que o método desenvolvido superou o XPRESS em todos os testes realizados. Nesse estudo, provavelmente, o fato mais importante é que o XPRESS só ficou próximo do método desenvolvido nos problemas muito fáceis. Nos problemas difíceis a diferença foi significativa, chegando, o método desenvolvido ser 70,85 vezes mais rápido do que o XPRESS. Os resultados do algoritmo desenvolvido de pós-otimização, para o *PTC*, também foram bastante satisfatórios. Isso permitiu, a aplicação,

de forma eficiente, dos testes de redução e limites inferiores, que utilizam o *PTC*, no método do Capítulo 4.

No Anexo 2 é apresentado um método, para atacar o *PLC*, que utiliza a técnica de decomposição, a qual está baseada na formação de grupos, os quais representam um subproblema do problema principal. Na verdade, é um algoritmo geral para resolver o problema e vários algoritmos baseados neste algoritmo geral, ou seja, várias especializações do algoritmo geral. Nenhum desses algoritmos garante a otimalidade da solução gerada, ou seja, são métodos considerados heurísticos. Os resultados computacionais mostram que os métodos geram soluções excelentes para problemas que abrem, na solução ótima, um número de facilidade maior ou igual a 20% do total de facilidades candidatas.

No Anexo 3 é apresentado outro método que utiliza a técnica de decomposição para resolver o problema. A decomposição utilizada é baseada na decomposição de Benders. A solução gerada pelo método é ótima. Os resultados computacionais mostraram, que apesar do método apresentar uma melhora significativa em relação à decomposição de Benders pura, o método não é muito eficiente. O principal problema apresentado foi a dificuldade de se resolver o problema mestre, o qual fornece o limite inferior. Além de ser um problema difícil de ser resolvido evolui com o limite inferior vagarosamente. Um resultado interessante, que embora não tenha sido mostrado nos resultados computacionais, foi que a solução ótima via limite superior foi encontrada muito rapidamente. Com isso, abrem-se novas frentes da pesquisa, as quais estão listadas no próprio anexo.

5.1 Principais contribuições

5.1.1 Principal contribuição

Desenvolvimento de um método ótimo para resolução do Problema de Localização Capacitado, o qual demonstrou ser eficiente e robusto. O método resolveu os problemas da literatura sem nenhuma dificuldade. Além disso, o método mostrou-se capaz de resolver, em tempo viável, problemas com dificuldades que ainda não tinham sido exploradas. As dificuldades introduzidas são combinações das seguintes características:

- Problemas com dimensões maiores aos encontrados na literatura, principalmente no que se refere ao número de facilidades, isto é, ao número de variáveis binárias (0 ou 1).
- Problemas não Euclidianos.
- Problemas com muitos ótimos locais.
- Problemas variando diversas características dos dados do problema.

Outra característica importante do método é que pode funcionar como um método heurístico e um método ε-ótimo, quando o método heurístico além de fornecer uma solução próxima da ótima fornece também um parâmetro de qualidade, a máxima distância que a solução fornecida está da ótima.

Provavelmente, a robustez do método é devido ao fato dele ser resolvido se utilizando diversas técnicas, tais como, quatro Testes de redução, dois limites inferiores, regras de ramificação sólidas, estratégia de busca que realiza uma combinação entre busca em profundidade e busca em largura, cortes com número mínimo e máximo, entre outras.

5.1.2 Outras contribuições importantes

- Estudo da aplicabilidade do *O-Teste* e *C-Teste* (dificuldades de fechamento do *gap*, generalização das dificuldades da aplicação dos testes, influência de um teste no outro).
- Desenvolvimento de algoritmos que utilizam o *O-Teste* e *C-Teste* para obtenção da solução ótima do *PLC*
- Adequação do *O-Teste* e *C-Teste* a um algoritmo *Branch and Bound*.
- Desenvolvimento e resolução do limite inferior utilizando a relaxação da restrição integridade (*PLCRI*).
- Adaptação do *PLCRI* ao *Branch and Bound*.
- Adaptação do limite inferior desenvolvido por Beasley [12], que utiliza Relaxação Lagrangeana, para contemplar o número mínimo e máximo de facilidades.
- Adaptação do método de resolução do limite inferior desenvolvido por Beasley [12], que utiliza Relaxação Lagrangeana, para contemplar o número mínimo e máximo de facilidades.

- Mudança dos parâmetros do limite inferior desenvolvido por Beasley [12], que utiliza Relaxação Lagrangeana, para se adequar ao método *Branch and Bound*.
- Desenvolvimento de algoritmos, Algoritmo 9 e Algoritmo 10, para a atualização do *PLCRL*. Esses algoritmos facilitam a aplicação do *PLCRL* no método *Branch and Bound*.
- Adaptação do *PLCRL* ao *Branch and Bound*.
- Desenvolvimento do modelo, a partir do *PLCRL*, para encontrar o número mínimo e máximo de facilidades abertas na solução ótima do *PLC*.
- Método de resolução do modelo, baseado no PLCRL, para encontrar o número mínimo e máximo de facilidades abertas na solução ótima do *PLC*.
- Desenvolvimento e dos testes *O-RITeste* e *C-RITeste*.
- Método de resolução dos testes *O-RITeste* e *C-RITeste*.
- Adaptação dos testes *O-RITeste* e *C-RITeste* ao *Branch and Bound*.
- Método de resolução dos testes *O-RLTeste* e *C-RLTeste*.
- Adaptação dos testes *O-RLTeste* e *C-RLTeste* ao *Branch and Bound*.
- Estudo da aplicabilidade dos *O-RLTeste, C-RLTeste, O-RLTeste, C-RITeste, O-Teste* e *C-Teste* no primeiro nó do *Branch and Bound*.
- Desenvolvimento de modelos, a partir do *O-Teste* e *C-Teste*, para encontrar o número mínimo e máximo de facilidades abertas na solução ótima do *PLC*.
- Método de resolução dos modelos, baseados no *O-Teste* e *C-Teste*, para encontrar o número mínimo e máximo de facilidades abertas na solução ótima do *PLC*.
- Adaptação das podas que utilizam o número mínimo e máximo de facilidades, abertas na solução ótima, ao *Branch and Bound*.
- Desenvolvimento das heurísticas *O/C-HeurísticaLI*, *CMin-HeurísticaLI* e *CMax-HeurísticaLI*. Essas heurísticas são aplicadas nas regras de ramificação.
- Desenvolvimento de uma estratégia de busca que utiliza uma combinação da busca em profundidade com a busca em largura. É importante destacar que essa estratégia pode ser utilizada em qualquer *Branch and Bound*.
- Embora não tenha sido testado, o método desenvolvido para resolução do *PLC* resolve o problema p-mediana. Para isso basta fixar o número mínimo e máximo de facilidades abertas na solução ótima com p e mudar a primeira solução viável gerada.

5.1.3 Contribuições dos anexos

- Desenvolvimento de um método de transporte muito eficiente.
- A principal novidade no método de transporte foi a inserção do heap para ajudar na busca da variável a entrar na base.
- Adequação do *PTC* ao *PLC*. Métodos para inserção e remoção de centros de oferta, possibilitando com isso a aplicação do *PTC* no *Branch and Bound*.
- Algoritmo de pós-otimização para o método de transporte desenvolvido.
- Desenvolvimento de um método ótimo baseado na decomposição de Benders. Embora o método não tenha apresentado resultados significativos, se comparado com o método do Capítulo 4, ele mostrou ser mais eficiente do que o método puro de Benders. Também abriu novas linhas para pesquisa.
- Desenvolvimento de heurísticas que utilizam análise de agrupamento. Vários métodos de agrupamento foram desenvolvidos, sendo que algumas deles demonstraram ótimos resultados, comparando-se com métodos heurístico da literatura, principalmente para problemas que abrem mais de 20% das facilidades na solução ótima.

5.2 Trabalhos futuros

Talvez a primeira coisa que deveria ser realizado seriam testes em problemas reais. Isso porque os testes que foram realizados podem ser considerados testes de estresse, pois procuram explorar problemas difíceis. Normalmente, os problemas reais têm a tendência de serem mais fáceis do que os problemas que foram gerados. Será que realmente são? Assim sendo, seria interessante a aplicação do método para responder essa questão e verificar quais são as limitações, do método, em relação a esses problemas.

Outro ponto que seria importante desenvolver é a aplicação de um terceiro limite inferior, o qual ainda não foi desenvolvido. Não é possível saber se ele fornecerá um ganho para o método, contudo, pela experiência adquirida, acredita-se que em alguns casos se obterá bons resultados. O desejo é desenvolver um método eficiente, se possível, para o seguinte problema:

(5-1) $$(PLC)\ \min \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i$$

S.a.

(5-2) $$\sum_{j\in J} x_{ij} \le a_i y_i, \quad \forall i \in I$$

(5-3) $$\sum_{i\in I} x_{ij} = b_j, \quad \forall j \in J$$

(5-4) $$x_{ij} \ge 0, \quad \forall i \in I, \forall j \in J$$

(5-5) $$y_i \ge 0, \quad \forall i \in I$$

(5-6) $$y_i \le 1, \quad \forall i \in I$$

(5-7) $$\frac{x_{ij}}{b} \le y_i, \quad \forall i \in I, \forall j \in J$$

(5-8) $$x_{ij}\ \text{int}, \quad \forall i \in I, \forall j \in J$$

O problema é parecido com o *PLCRI* desenvolvido na seção 3.1, acrescido das restrições (5-7) e (5-8).

Embora não tenha sido mencionado no trabalho, foi desenvolvido um método, baseado no método do Capítulo 4, para resolver o Problema de Localização não Capacitado (*PLNC*). O método foi desenvolvido para resolver o problema gerado pela Relaxação Lagrangeana da restrição de capacidade. Isto é para resolver um limite inferior do *PLC*. Esse limite inferior não foi utilizado porque embora o limite gerado tenha boa qualidade o seu custo computacional é muito caro para aplicá-lo no *Branch and Bound*. Embora o método não tenha sido utilizado como limite inferior o mesmo mostrou-se muito eficiente para resolução do *PLNC*. Com isso, vale finalizar o trabalho testando o método e melhorando-o.

Provavelmente o principal trabalho para o futuro é tornar o método para resolução do *PLC* distribuído. Embora omitido, um método que rodasse de forma distribuída nunca deixou de ser um dos objetivos. A idéia inicial foi desenvolver um

método que rodasse seqüencialmente de forma robusta e eficiente, mas que tivesse características que pudessem ser executadas paralelamente. Assim é o método desenvolvido para resolver o *PLC*. As características paralelas são inúmeras, sendo a principal, em primeira análise, a possibilidade de se explorar todos os nós pertencentes a α, nós ainda não explorados, de forma independente. A única relação dos nós de α, não absolutamente necessária, mas que ajudaria muito nas podas, é o melhor limite superior encontrado. Essa relação é simples e não requer grande *overhead* na comunicação. Existem outras características paralelas tais como: cada teste de redução é independente um do outro, por exemplo, a aplicação do *O-Teste* para a facilidade *i* não tem relação com a aplicação do *O-Teste* para facilidade *j*; os limites inferiores são independentes um do outro, isto é, pode-se calcular o *PLCRI* e o *PLCRL* paralelamente. A atualização do número mínimo e máximo de facilidades abertas na solução ótima só depende da mudança do *status* de uma facilidade.

Pela experiência adquirida acredita-se que para um bom desempenho seja necessário paralisar somente a exploração dos nós. Isto é, vai-se distribuindo os nós pertencentes a α para cada máquina e ou processador ocioso. Nesse caso, existirão dois tipos de comunicação: o primeiro, bastante simples, é a informação que uma solução melhor foi encontrada. Para isso, basta enviar um valor do tipo real para todas as máquinas envolvidas. O segundo tipo de comunicação é a distribuição balanceada dos nós a serem processados. Essa distribuição é o grande desafio do desenvolvimento do método distribuído, isso porque em um nó existem muitas informações para serem transmitidas, não podendo, com isso, ser desenvolvida uma política qualquer distribuição, pois poderá acarretar no congestionamento da rede/barramento e num grande *overhead* de processamento.

Vale observar que, hoje, normalmente, o custo de processamento é muito barato em relação ao ganho que se pode obter com uma possível melhora da qualidade da solução do PLC. Normalmente o investimento fica na casa de mil e o beneficio, a médio e longo prazo, na casa de milhões. Hoje, é fácil encontrar empresas de médio e grande porte com diversos *cluster* com mais de 2000 máquinas com excelente capacidade de processamento em cada uma delas. Esse tipo de informação motiva o desenvolvimento de um método distribuído.

Anexo 1

6 Problema de transporte

Neste capítulo será apresentado o método desenvolvido para resolver o Problema de Transporte Capacitado (*PTC*). O *PTC* já foi amplamente discutido na literatura. Existem vários métodos para resolvê-lo, tanto utilizando fluxos em redes, quanto o método Simplex. O leitor pode obter conhecimento sobre o assunto através das seguintes bibliografias: Goldbarg e Luna [26], Ahuja, Magnanti e Orlin [35], Chvátal [18], Jacobsen [29] e Ahrens e Finke [2].

Foi dedicado atenção especial ao *PTC*, porque é uma das maiores ferramentas na resolução do *PLC*. O *PTC* é utilizado para a realização de quatro testes de redução, dois dos quais serão vistos no Capítulo 2 e os outros dois, no Capítulo 3. Outro subproblema que utiliza o *PTC* é o cálculo de um dos limites inferiores utilizados na resolução do *PLC*, a ser visto no Capítulo 3. Isso mostra que o método desenvolvido para resolver o *PLC* é altamente dependente do *PTC*. Um método ineficiente de resolução do *PTC* pode comprometer a eficiência computacional do algoritimo proposto parra a resolução do PLC. O objetivo principal deste trabalho é mostrar como o problema foi resolvido e fazer comparações com o *software* comercial XPRESS.

O *PTC* consiste em atender aos centros de demanda, a partir de um conjunto de centros de oferta, de forma a minimizar o custo de transporte. Cada centro de oferta tem uma capacidade máxima de atendimento, a qual não pode ser ultrapassada. Cada centro de demanda deve ser atendido por completo, ou seja, toda sua demanda deve ser suprida.

Inicialmente, foi implementado o método desenvolvido por Jacobsen [29] para resolução do *PTC*. O algoritmo utiliza o método primal Simplex para encontrar a solução e uma árvore geradora para representar a solução básica correspondente. O método não demonstrou grande eficiência.

O método de Jacobsen [29] tem um grande limitador: o cálculo dos custos reduzidos. A cada iteração do método Simplex, é necessário calcular todos os custos reduzidos. Para sanar esse problema, foi realizado uma combinação entre o método de Jacobsen [29] e o método de Ahrens e Finke [2]. Nesse novo método desenvolvido são utilizadas as variáveis do problema dual para se calcular o custo reduzido. Com isso, a cada iteração do método Simplex, só é necessária a atualização das variáveis do problema dual que pertencem à subárvore afetada com a variável que entrou na base.

O segundo método para resolução do *PTC* teve um melhora significativa em relação ao primeiro. Testes realizados, considerando o tempo de execução, mostraram que o segundo método obteve um desempenho superior, em mais de três vezes, em relação ao primeiro método. Embora o objetivo de melhorar o desempenho do método tenha sido alcançado, os resultados ainda não eram satisfatórios. Comparações realizadas entre o método desenvolvido e o XPRESS mostraram que o XPRESS chegou a ter um tempo de execução superior em 10 vezes, em relação ao segundo método.

Foi dada seqüência à pesquisa sobre uma possível melhora de eficiência do algoritmo de transporte. Identificou-se que, embora o segundo método tenha melhorado a eficiência em relação ao cálculo do custo reduzido, ainda era muito custoso, computacionalmente falando, encontrar a variável para entrar na base (isto é, encontrar um custo reduzido com um valor negativo, mesmo utilizando a tática de se colocar na base a primeira variável encontrada associada a um custo reduzido negativo). A partir dessa observação, foi desenvolvido o método que será apresentado neste capítulo, onde foi acrescentada uma lista de prioridades, utilizando a estrutura de dados *heap*, para armazenar os custos reduzidos negativos, facilitando, assim, a procura por uma variável para entrar na base.

6.1 Modelagem matemática do Problema de Transporte Capacitado

O Problema de Transporte Capacitado (*PTC*) consiste em, dado um conjunto de centros de oferta I, em que cada centro tem uma capacidade a_i, e um conjunto de centros de demanda J, em que cada um tem necessidade b_j, encontrar uma solução que atenda todas as necessidades dos centros de demanda, de forma a minimizar o custo total. O

custo é composto pelo custo de transporte c_{ij}, que é o custo de transporte de uma unidade do centro de oferta *i* para o centro de demanda *j*. O problema de transporte capacitado pode ser formulado como:

(6-1) $$\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

s.a.

(6-2) $$\sum_{j \in J} x_{ij} \leq a_i, \quad \forall i \in I$$

(6-3) $$\sum_{i \in I} x_{ij} = b_j, \quad \forall j \in J$$

(6-4) $$x_{ij} \geq 0, \quad \forall i \in I, \forall j \in J$$

onde x_{ij} é a quantidade enviada de *i* para *j*. As restrições (1-2) asseguram que o total de demanda atendida pelo centro de oferta *i* não ultrapasse a sua capacidade. As restrições (1-3) asseguram que a demanda de cada centro de demanda seja atendida. As restrições (1-4) asseguram que as quantidades transportadas não sejam negativas.

6.2 Árvore geradora

A teoria dos matróides estabeleceu a conexão entre a solução básica de modelos de programação linear sobre grafos, isto é, de modelos de fluxo em redes, e as árvores geradoras do grafo. Será utilizada, neste trabalho, uma árvore geradora para a representação de uma solução básica. A cada iteração do método simples a árvore geradora será atualizada para representar a nova base. Existem várias estruturas para se representar uma árvore geradora. No método desenvolvido, foi utilizada a estrutura *threaded index*, apresentado em Jacobsen [29], incluindo, na estrutura, os dados de altura de cada nó da árvore.

6.2.1 Estrutura da árvore geradora

Seja *T* uma árvore geradora e *S* o conjunto de nós de *T*, onde $|S| = n$ nós. A árvore geradora *T* será representada através de três índices: predecessor, linha e altura. Cada índice terá tamanho *n* e eles representaram, respectivamente, o pai de cada nó da árvore, o sucessor, no percurso da árvore, de cada nó da árvore e a altura de cada nó da árvore. O índice predecessor será representado por *P*, onde *P(i)* indica o pai do nó *i* na

árvore, $\forall i \in S$. O índice linha será representado por *L*, onde *L(i)* indica o sucessor, no percurso da árvore, do nó $i, \forall i \in S$. O índice altura será representado por *A*, onde *A(i)* indica a altura do nó *i* na árvore, $\forall i \in S$.

A título de exemplo, a árvore da Figura 19 será representada através dos índices *P*, *L* e *A* mostrados na Tabela 37.

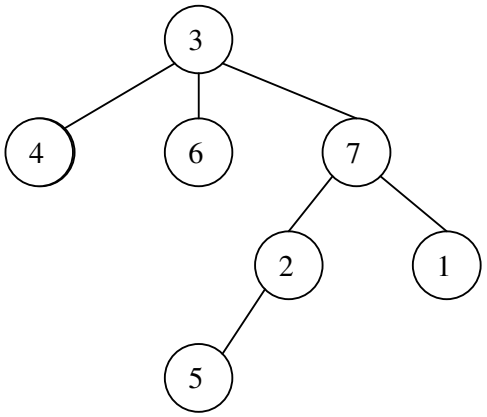


Figura 19 – Árvore geradora

	1	**2**	**3**	**4**	**5**	**6**	**7**
P(i)	7	7	0	3	2	3	*3*
L(i)	0	5	4	6	1	7	*2*
A(i)	*2*	*2*	*0*	*1*	*3*	*1*	*1*

Tabela 37 – valores do índices

O predecessor do nó *raiz* , *P(raiz)*, sempre terá valor 0. No exemplo da Figura 19 *P(3) = 0*. O índice *L* estabelece o recurso para que a árvore seja percorrida em pré-ordem. De maneira geral, o algoritmo de percurso em pré-ordem de uma árvore realiza os seguintes passos:

- Visita o nó raiz;
- Percorre todas as sub-árvores em pré-ordem, começando pela sub-árvore mais à esquerda até a mais à direita.

O leitor pode tomar mais conhecimento do algoritmo de busca em pré-ordem, sob uma árvore, em Szwarcfiter e Markenzon [38]. A definição formal do índice linha pode ser encontrada em Glover et al. [27]. A Figura 20 mostra o índice linha, através das setas pontilhadas, sob a árvore mostrada na Figura 19.

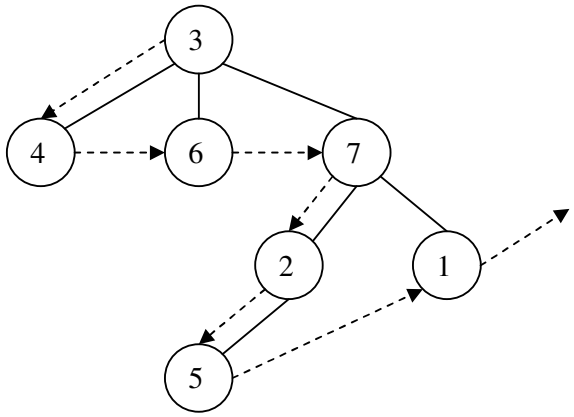


Figura 20 – indice linha na árvore

Para que o índice linha seja mais bem definido, serão introduzidos os índices *sucessor* e *irmão*, onde sucessor, chamado de *S(i)*, representa o filho mais à esquerda de cada nó *i*, e irmão, chamado de *I(i)*, representa o irmão mais à direita de cada nó *i*. Quando um nó *i* não possui filho, *S(i) = 0* e quando um nó *i* não possui irmão à direita, *I(i) = 0*. Considerando a árvore do exemplo da Figura 19, a Tabela 38 representa os índices sucessor e irmão.

I	**1**	**2**	**3**	**4**	**5**	**6**	***7***
S(i)	0	5	4	0	0	0	*2*
I(i)	*0*	*1*	*0*	*6*	*0*	*7*	*0*

Tabela 38 – valores do indices filho e irmão

O índice linha pode ser explicado utilizando os índices *S* e *I* através do Algoritmo 22.

Algoritmo 22: encontrarLinha

Passo 1:

Iniciar fazendo *i* igual a nó raiz

Passo 2:

Se *S(i)* ≠ *0* ***então*** fazer *L(i)* ⇓ *S(i)* ir para o passo 5

Passo 3:

Se *I(i)* ≠ *0* ***então*** fazer *L(i)* ⇓ *I(i)* ir para o passo 5

Passo 4:

Fazer *k* = *i* e ir para cima da árvore, utilizando *k* ⇓ *P(k)*, até que seja encontrado um *j* = *B(k)*, irmão de *k*, com *L(j)* indefinido.

Se *j* não foi encontrado ***então*** fazer *L(i)* ⇓ *0* e parar

Senão colocar *L(i)* ⇓ *j* ir para o passo 5

Passo 5:

Fazer *i* ⇓ *T(i)* ir para o passo 2.

----- Fim Algoritmo 22 -----

6.3 Método para resolução do PTC

O método desenvolvido resolve o seguinte problema (*PTC'*):

(6-5) $$\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

S.a.

(6-6) $$\sum_{j \in J} x_{ij} = a_i, \quad \forall i \in I$$

(6-7) $$\sum_{i \in I} x_{ij} = b_j, \quad \forall j \in J$$

(6-8) $$x_{ij} \geq 0, \quad \forall i \in I, \forall j \in J$$

A diferença entre o *PTC'* e o *PTC* está na restrição de capacidade. A restrição (6-6) é de igualdade, indicando que toda oferta, de cada centro de oferta, deve ser transportada. O problema que se deseja resolver é o *PTC*.Assim, é necessário que o mesmo seja transformado no *PTC'*. Para realizar essa transformação basta criar um centro de demanda virtual *v*, onde sua demanda será $b_v = \sum_{i \in I} a_i - \sum_{j \in J} b_j$ e $c_{iv} = 0 \ \forall i \in I$.

Isso significa que todas as ofertas não enviadas aos centros de demanda do *PTC* serão enviadas para o centro virtual, *v* com custo 0, não alterando com isso o valor da solução do *PTC*.

6.3.1 Estrutura da solução básica

A solução básica viável do primal sempre satisfaz as restrições de (6-6) a (6-8). Em Dantzig [21], mostra-se a correspondência de 1 para 1 entre a solução básica do primal e a árvore geradora do grafo bipartido $K_{I,J}$, onde um arco da árvore geradora representa uma variável na base. O exemplo da Figura 21 mostra um árvore geradora com três centros de demanda (D_1,D_2,D_3) e quatro centros de oferta (O_1,O_2,O_3,O_4). No exemplo, as variáveis x_{13}, x_{23}, x_{33}, x_{32}, x_{31}, x_{42} estão na base. A árvore geradora será composta por $|I| + |J|$ nós, sendo que cada nó representa um centro de oferta ou de demanda.

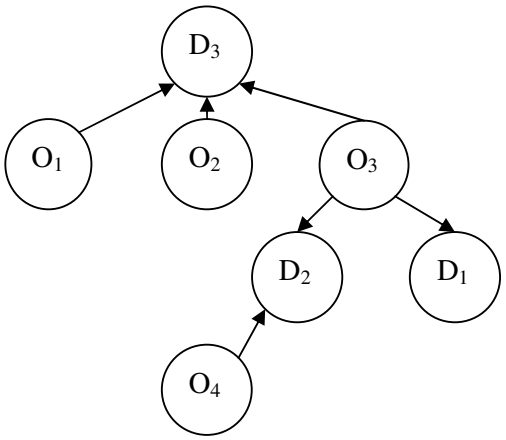


Figura 21 – exemplo de árvore geradora

A árvore geradora representa as variáveis na base. É necessário também representar o valor das mesmas. A Figura 21 mostra, através das setas, a direção do fluxo na árvore, sempre saído de um centro de oferta para um centro de demanda. O valor do fluxo representa o valor da variável na base. Será utilizado o elemento *X* para representar o fluxo nos arcos. *X* terá tamanho *n*, onde *n* é o número de nós da árvore *T*. $X(k)\ \forall\, k \in T$ é definido como:

- Se *k* for raiz $X(k) = 0$
- Se *k* representa um centro de oferta $X(k) = x_{kP(k)}$
- Se *k* representa um centro de demanda $X(k) = x_{P(k)\,k}$

Uma solução básica do *PTC'* será representada através dos índices *P*, *L*, *A* e do elemento *X*. Todos são vetores unidimensionais de tamanho $n = |I|+|J|$. As $|I|$ primeiras

posições representarão os centros de oferta e as |*J*| últimas posições representarão os centros de demanda. Então, para se verificar, por exemplo, o predecessor do centro de demanda *j*, utiliza-se a expressão *P(j+|I|)* e para o predecessor do centro de oferta *i*, utiliza-se *P(i)*.

6.3.2 Primeira solução básica viável

O método Simplex que será desenvolvido parte de uma solução básica viável. Como esta solução ainda não está disponível, é necessário encontrá-la. Assim sendo, precisa-se de um procedimento, como a fase 1 do método Simplex, que encontre a solução inicial. Vários métodos foram propostos na literatura para se obter a solução inicial do PTC'. Três deles forma implementados: o NWCR (*north west corner rule*), o SMALC (*minimum entry method*) e o RME (*row minimum entry*). Será mostrado somente o NWCR, o qual está sendo utilizado pelo método desenvolvido, pois foi o que obteve resultados significativamente melhores do que os outros. O leitor pode conhecer os outros métodos em Jacobsen [29].

Para o desenvolvimento de método NWCR, será introduzido o vetor auxiliar *Q*. *Q(k)* representa a capacidade restante, oferta ainda não transportada, do centro de oferta *k*, se $k \in I$. Caso contrário, *Q(k)* representa a demanda que falta para atender o centro de demanda *j*. Seja *T* a árvore geradora a ser construída, com |*I*| + |*J*| nós, o Algoritmo 23 calcula a solução inicial, onde a variável *Custo* representa o custo da solução encontrada.

Algoritmo 23: Encontrar a Solução Básica Viável NWCR

{ Inicialização }
$\forall$ k $\in$ T X(k) $\Downarrow$ 0; P(k) $\Downarrow$ 0; L(k) $\Downarrow$ 0.
$\forall$ k $\in$ T ***Se*** k $\leq$ I ***então*** Q(k) $\Downarrow$ a_k { k $\in$ I }
Senão Q(k) $\Downarrow$ b_k { k $\in$ J }
i = 1; j $\Downarrow$ 1
Custo $\Downarrow$ 0

```
{ Construção da árvore que representa a solução básica viável }
Enquanto (i + j) ≤ |T| fazer
Inicio
        c = c_ij;
        Se Q(i) = Q(j+|I|) então
        Inicio
                Se j = |J| então
                Inicio
                        pai ⇓ j+|I|
                        filho ⇓ i
                        i ⇓ i + 1
                Fim
                Senão
                        pai ⇓ i
                        filho ⇓ j+|I|
                        j ⇓ j + 1
                Inicio
                Fim
        Fim
        Senão
        Inicio
                Se Q(i) < Q(j+|I|) então
                Inicio
                        pai ⇓ j+|I|
                        filho ⇓ i
                        i ⇓ i + 1
                Fim
                Senão
                Inicio
                        pai ⇓ i
                        filho ⇓ j+|I|
                        j ⇓ j + 1

                Fim
        Fim
        Custo ⇓ Custo + c * Q(filho)
        X(filho) ⇓ Q(filho)
        Q(filho) ⇓ 0
        Q(pai) ⇓ Q(pai) – Q(filho)
        P(filho) ⇓ pai
        Se L(pai) ≠ 0 então
                L(filho) ⇓ L(pai)
        L(pai) ⇓ filho
Fim
----- Fim Algoritmo 23 -----
```

Exemplo de aplicação do NWCR

Seja o Problema de Transporte Capacitado *P* da Tabela 39, onde a coluna a_i representa a capacidade de cada centro de oferta, a linha b_j representa a demanda de

cada centro de demanda e o restante da matriz representa o custo de transporte, de uma unidade, do centro de oferta *i* para um centro de demanda *j*. As Tabelas 5 a 14 mostram as iterações do algoritmo NWCR. Antes da aplicação do NWCR, deve-se transformar o problema da Tabela 39 no *PTC'*, como apresentado na seção 6.3. A Tabela 40 mostra o problema após a criação do centro de demanda virtual D_6.

Centros Demanda/ Centros Oferta	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	***a_i***
O_1	5	10	30	40	28	*15*
O_2	25	30	6	7	40	*10*
O_3	15	15	40	60	17	*20*
O_4	37	29	20	25	8	*15*
b_j	*10*	*10*	*10*	*10*	*10*	-

Tabela 39 – Dados do *PTC*

Centros Demanda/ Centros Oferta	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	***a_i***
O_1	5	10	30	40	28	0	*15*
O_2	25	30	6	7	40	0	*10*
O_3	15	15	40	60	17	0	*20*
O_4	37	29	20	25	8	0	*15*
b_j	*10*	*10*	*10*	*10*	*10*	*10*	-

Tabela 40 – Dados do *PTC* transformado

Inicialização

i = 1; j =1

Custo = 0

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	0	0	0	0	0	0	0	0	0	*0*
L(i)	0	0	0	0	0	0	0	0	0	*0*
X(i)	0	0	0	0	0	0	0	0	0	*0*
Q(i)	*15*	*10*	*20*	*15*	*10*	*10*	*10*	*10*	*10*	*10*

Tabela 41 – índices

Final da iteração 1

i = 1; j =2

c = 5

Custo = 50

Pai = 1

Filho = 5

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	0	0	0	0	1	0	0	0	0	*0*
L(i)	5	0	0	0	0	0	0	0	0	*0*
X(i)	0	0	0	0	10	0	0	0	0	*0*
Q(i)	*5*	*10*	*20*	*15*	*0*	*10*	*10*	*10*	*10*	*10*

Tabela 42 – índices iteração 1

Final da iteração 2

i = 2; j =2

c = 10

Custo = 100

Pai = 6

Filho = 1

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	0	0	0	1	0	0	0	0	*0*
L(i)	5	0	0	0	0	1	0	0	0	*0*
X(i)	5	0	0	0	10	0	0	0	0	*0*
Q(i)	*0*	*10*	*20*	*15*	*0*	*5*	*10*	*10*	*10*	*10*

Tabela 43 – índices iteração 2

Final da iteração 3

i = 2; j =3

c = 30

Custo = 250

Pai = 2

Filho = 6

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	0	0	0	1	2	0	0	0	*0*
L(i)	5	6	0	0	0	1	0	0	0	*0*
X(i)	5	0	0	0	10	5	0	0	0	*0*
Q(i)	*0*	*5*	*20*	*15*	*0*	*0*	*10*	*10*	*10*	*10*

Tabela 44 – índices iteração 3

Final da iteração 4

i = 3; j =3

c = 6

Custo = 280

Pai = 7

Filho = 2

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	7	0	0	1	2	0	0	0	*0*
L(i)	5	6	0	0	0	1	2	0	0	*0*
X(i)	5	5	0	0	10	5	0	0	0	*0*
Q(i)	*0*	*0*	*20*	*15*	*0*	*0*	*5*	*10*	*10*	*10*

Tabela 45 – índices iteração 4

Final da iteração 5

i = 3; j =4

c = 40

Custo = 480

Pai = 3

Filho = 7

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	7	0	0	1	2	3	0	0	*0*
L(i)	5	6	7	0	0	1	2	0	0	*0*
X(i)	5	5	0	0	10	5	5	0	0	*0*
Q(i)	*0*	*0*	*15*	*15*	*0*	*0*	*0*	*10*	*10*	*10*

Tabela 46 – índices iteração 5

Final da iteração 6

i = 3; j =5

c = 60

Custo = 1080

Pai = 3

Filho = 8

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	7	0	0	1	2	3	3	0	*0*
L(i)	5	6	8	0	0	1	2	7	0	*0*
X(i)	5	5	0	0	10	5	5	10	0	*0*
Q(i)	*0*	*0*	*5*	*15*	*0*	*0*	*0*	*0*	*10*	*10*

Tabela 47 – índices iteração 6

Final da iteração 7

i = 4; j =5

c = 17

Custo = 1165

Pai = 9

Filho = 3

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	7	9	0	1	2	3	3	0	*0*
L(i)	5	6	8	0	0	1	2	7	3	*0*
X(i)	5	5	5	0	10	5	5	10	0	*0*
Q(i)	*0*	*0*	*0*	*15*	*0*	*0*	*0*	*0*	*5*	*10*

Tabela 48 – índices iteração 7

Final da iteração 8

i = 4; j =6

c = 8

Custo = 1205

Pai = 5

Filho = 9

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	7	9	0	1	2	3	3	4	*0*
L(i)	5	6	8	9	0	1	2	7	3	*0*
X(i)	5	5	5	0	10	5	5	10	5	*0*
Q(i)	*0*	*0*	*0*	*10*	*0*	*0*	*0*	*0*	*0*	*10*

Tabela 49 – índices iteração 8

Final da iteração 9

i = 5; j =6

c = 0

Custo = 1205

Pai = 10

Filho = 4

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	7	9	10	1	2	3	3	4	*0*
L(i)	5	6	8	9	0	1	2	7	3	*4*
X(i)	5	5	5	10	10	5	5	10	5	*0*
Q(i)	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*10*

Tabela 50 – índices iteração 9

Na Tabela 50, a condição de parada do algoritmo de construção da árvore geradora, que representa uma solução básica viável, foi atingida, isto é, $(i+j) > |T|$. A Figura 22 mostra graficamente como ficou a árvore.

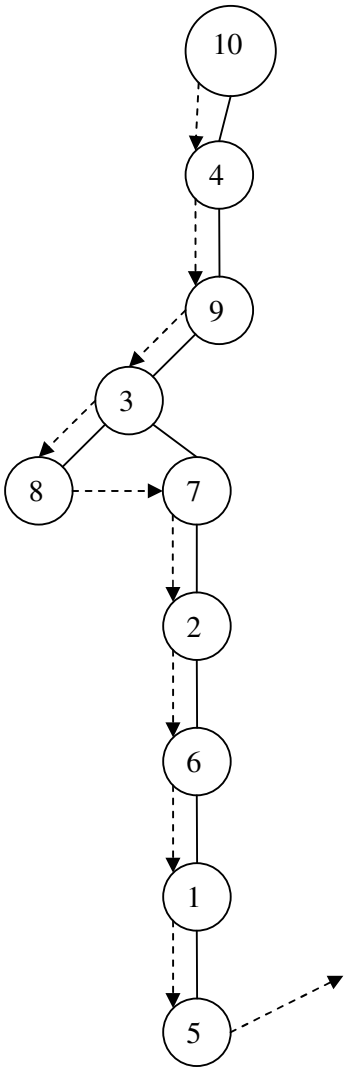


Figura 22 – árvore após a aplicação do Algoritmo 23

Vale observar que se for necessário o acréscimo do centro de demanda virtual, o mesmo será a raiz da árvore gerada pelo algoritmo NWCR. De qualquer forma, será visto, na seção 6.3.3, o algoritmo que procura o nó raiz da árvore.

6.3.3 Cálculo do custo reduzido

Para encontrar a variável que irá entrar na base, é necessário realizar o cálculo do custo reduzido. Para isso, será utilizado o problema dual do *PTC'*, que tem a seguinte formulação:

(6-9) $$(DPTC')\max \sum_{i\in I} a_i y_i \sum_{j\in J} b_j y_j$$

S.a.

(6-10) $$y_i + y_j \le c_{ij} \forall i \in I, \forall j \in J$$

Utilizando o teorema da folga complementar, o qual pode ser visto em Puccini A. L. e Pizzolato [34], o reduzido associado à variável x_{ij} pode ser obtido como:

(6-11) $$d_{ij} = c_{ij} - y_i - y_j \forall i \in I, \forall j \in J$$

Sabe-se também, pelo teorema da folga complementar, que:

(6-12) $$x_{ij} d_{ij} = 0 \forall i \in I, \forall j \in J \ .$$

Assim, para toda variável x_{ij} básica, $d_{ij}=0$. A partir da aplicação do algoritmo NWCR é possível saber quais são as variáveis básicas da solução encontrada. Para que seja possível calcular as variáveis duais e, por conseqüência, o custo reduzido, como mostrado em Jacobsen [29], é necessário encontrar a raiz da árvore geradora. O Algoritmo 24, que deve ser executado após o término do algoritmo NWCR, encontra o nó raiz da árvore. O algoritmo, simplesmente, realiza uma pesquisa no índice *P* para encontrar *P(i) = 0*.

Algoritmo 24: Encontrar Raiz da Árvore Geradora

```
k ⇓ 1
encontrou ⇓ falso
Enquanto não encontrou faça
Inicio
        Se P(k) = 0 então
        Inicio
                raiz = k
                Encontrou = verdadeiro
        Fim
        Senão
                k ⇓ k+1
Fim
----- Fim Algoritmo 24 -----
```

Tendo a raiz em mãos e a árvore gerada pelo algoritmo NWCR, é fácil realizar o cálculo das variáveis duais. Para isso, como mostrado em Jacobsen [29], basta fazer $y_{raiz}=0$, e utilizar as equações (6-11) e (6-12).

Pela equação (6-12), tem-se que $d_{kP(k)}=0$. Então pela equação (6-11), tem-se $y_k = c_{kP(k)} - y_{P(k)}$, para $k \in I$ ou $d_{P(k)k}=0$ e $y_k = c_{P(k)k} - y_{P(k)}$ para $k \in J$. Como $y_{raiz}=0$, basta percorrer a árvore geradora, da raiz para as folhas, para calcular as variáveis duais. O Algoritmo 25 calcula as variáveis duais e a altura da árvore. Vale lembrar que somente é possível obter as variáveis duais após a obtenção da primeira solução viável, pois só no término do NWCR é possível encontrar o nó raiz da árvore. Os valores das variáveis duais serão armazenados no vetor Y de tamanho $n = |I|+|J|$, onde as primeiras $|I|$ posições de Y representarão $y_i \quad \forall i \in I$, e as $|J|$ últimas posições representarão $y_j \quad \forall j \in J$.

Algoritmo 25: Calcular Variáveis Duais e Altura

```
pai ⇓ raiz   e  filho ⇓ L(pai)
A(pai) ⇓ 0
Se (pai ≤ |I|) então
Inicio
        i ⇓ pai; j ⇓ filho - |I|
        Y(pai) ⇓ cij
Fim
Senão
Inicio
        i ⇓ filho; j ⇓ pai - |I|
        Y(pai) ⇓ cij
Fim
Enquanto filho ≠ 0 fazer
Inicio
        A(filho) ⇓ A(pai) + 1;
        Se (filho ≤ |I|) então
        Inicio
                i ⇓ filho; j ⇓ pai - |I|
                Y(filho) ⇓ cij – Y(pai)
        Fim
        Senão
        Inicio
                i ⇓ pai; j ⇓ filho - |I|
                Y(filho) ⇓ cij – Y(pai)
        Fim
        filho ⇓ L(filho)
        pai ⇓ P(filho)
Fim
----- Fim Algoritmo 25 -----
```

Encontrados os valores das variáveis duais é possível calcular o custo reduzido, das variáveis que não estão na base, utilizando a equação (6-11). Com isso, pode-se encontrar possíveis variáveis para entrar na base e melhorar a solução do *PTC'*.

A Tabela 51 mostra as variáveis duais do problema da Tabela 40, após terem sido executados os algoritmos NWCR e **EncontrarRaiz**. A Tabela 52 mostra os custos reduzidos: os negativos estão nas células em cinza.

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
Y(i)	-45	-25	9	0	50	55	31	51	8	*0*
A(i)	*7*	*5*	*3*	*1*	*8*	*6*	*4*	*4*	*2*	*0*

Tabela 51 – valores das variáveis duais e altura dos nós

Centros Demanda/ Centros Oferta	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	***D_6***
O_1	0	0	44	34	42	*45*
O_2	0	0	0	-19	57	*25*
O_3	-44	-49	0	0	0	*-9*
O_4	*-13*	*-26*	*-11*	*-26*	*0*	*0*

Tabela 52 – valores dos custos reduzidos

6.3.4 Encontrar a variável para entrar na base

Encontrar a variável para entrar na base é o mesmo que encontrar o arco para entrar na árvore geradora. Uma variável para entrar na base deve ter o custo reduzido associado negativo. A Tabela 52 mostra que as variáveis x_{24}, x_{31}, x_{32}, x_{36}, x_{41}, x_{42}, x_{43} e x_{44} têm custos reduzidos negativos, portanto, podem entrar na base. Normalmente, os métodos encontrados na literatura optam por uma das duas opções abaixo listadas para escolha da variável a entrar na base, sendo a segunda a mais utilizada.

1. Encontrar a variável com menor custo reduzido negativo para entrar na base, no caso da Tabela 52, a variável x_{32}.
2. Encontrar a primeira variável com custo reduzido negativo, no caso da Tabela 52, a variável x_{24}.

Encontrar a variável para entrar na base, utilizando o método Simplex, no *PTC*, pode ser uma das tarefas mais custosas em termos computacionais, porque o número de variáveis, normalmente, é muito maior que o número de restrições. Um problema com 1000 centros de ofertas e 1000 centros de demanda gera um milhão de variáveis. Assim

sendo, o espaço de busca para se encontrar um custo reduzido negativo, tem dimensão de um milhão. No pior caso, para se encontrar um custo reduzido negativo, tem-se a ordem de um milhão de operações e, no caso médio, quinhentas mil operações. Foi com base nessas informações que foi feita a principal mudança no método de resolução do *PTC*, em relação aos métodos encontrados na literatura.

Foi introduzida uma lista de prioridades *H*, utilizando a estrutura de dados *heap*, para ajudar na busca pela variável a entrar na base. Vale lembrar que a inserção e a remoção em uma lista de prioridades, utilizando *heap*, tem $O(log_2\ n)$ Szwarcfiter e Markenzon [38]. Embora a idéia tenha sido simples, o ganho de performance foi significativo, como será visto na seção 6.5, onde o método passou de 10 vezes mais lento, em alguns casos, a 70 vezes mais rápido, em alguns casos, do que o *software* comercial XPRESS.

A lista de prioridades tem a função de armazenar as variáveis com os custos reduzidos negativos. Após o cálculo das variáveis duais, conforme mostrado na seção 6.3.3, são calculados todos os custos reduzidos, colocando-se na lista de prioridades aqueles com valores negativos. A partir desse momento, o espaço de busca por uma variável a entrar na base passa ser a lista de prioridades, até que a mesma fique vazia. Ficando a lista de prioridades vazia, novamente são calculados todos os custos reduzidos, colocando-se na lista de prioridades aqueles que têm valor negativo. Caso não seja encontrado nenhum custo reduzido com valor negativo, para que a variável associada seja inserida na lista de prioridades, significará que a condição de parada do método Simplex foi atingida, isto é, a solução ótima foi encontrada. Nem sempre a lista de prioridades pode trazer benefícios. Caso o número de custos reduzidos negativos seja próximo do número de variáveis não básicas pode ser que não se tenha beneficio algum ou até alguma perda de performance. A principal motivação para introdução da lista de prioridades é porque, normalmente, o número de custos reduzidos negativos é muito menor do que o número de variáveis não básicas.

Assim que uma lista de prioridades for construída, seja após o cálculo das variáveis duais, conforme mostrado na seção 6.3.3, seja após a lista ficar vazia, os valores dos custos reduzidos são reais. Isso significa que é possível pegar uma variável da lista e colocar na base, pois a mesma tem custo reduzido negativo, satisfazendo a

condição suficiente para se entrar na base. Nesse caso, a variável a entrar na base será a primeira da lista, porque possui o menor custo reduzido negativo.

Após a entrada na base da primeira variável contida na lista de prioridades, não é mais possível garantir que as variáveis contidas na mesma continuem com o valor do custo reduzido negativo. Contudo, é na lista de prioridades que será realizada a busca por uma variável a entrar na base, uma vez que a primeira variável da lista de prioridades, após a primeira remoção, é a que tinha o segundo menor custo reduzido na iteração anterior. É muito difícil que uma mudança na base tenha tornado seu custo reduzido positivo. Isso vale para a segunda variável da lista, a terceira e assim por diante. É claro que há uma tendência muito maior de que o custo reduzido das últimaas variáveis da lista de prioridades tenha se tornado positivo. Porém, quando chegar a hora de sua verificação, isto é, de se calcular o seu custo reduzido atual, o tamanho da lista será pequeno.

O cálculo do custo reduzido de uma variável tem *O(1)*, já que as variáveis duais serão sempre atualizadas, conforme será mostrado na seção 6.3.6.3. Assim, para verificar se a variável retirada da lista de prioridades continua com custo reduzido negativo, basta calcular a expressão (6-11).

Na seqüência será apresentado o Algoritmo 26, para a construção da lista de prioridades *H*, sendo que o primeiro elemento em H, será sempre a variável com o menor custo reduzido. É utilizada a função *inserir(H, e)*, para inserir um elemento na lista. O elemento é composto pelo valor do índice $i \in I$, índice $j \in J$, e o custo reduzido, representados respectivamente por e_i, e_j e e_{cr}.

Algoritmo 26: Construir Lista de Prioridades

$\forall$ i $\in$ I $\forall$ j $\in$ J ***fazer***
Inicio
    $e_{cr} \Downarrow c_{ij} - Y(i) - Y(j+|I|)$
    Se $e_{cr} < 0$ ***então***
    Inicio
        $e_i \Downarrow$ i
        $e_j \Downarrow$ j
        inserir(H,e)
    Fim
Fim
----- Fim Algoritmo 26 -----

Será mostrado o algoritmo que encontra a variável para entrar na base, o Algoritmo 27. É importante lembrar que, caso não exista variável com custo reduzido negativo, a condição de parada do algoritmo Simplex foi atingida. A função *vazia(H)* retorna verdadeiro, se a lista *H* estiver vazia, ou falso, caso contrário. A função *remover(H)* retira o primeiro elemento da lista *H*, retornando-o. As variáveis *i* e *j* representam os índices do centro de oferta e centro de demanda da variável x_{ij} que entrará na base.

Algoritmo 27: Encontrar Arco de Entrada

encontrou ⇓ falso
Enquanto não encontrou e não vazia(H) ***Fazer*** *{ laço 1 }*
Inicio
    e ⇓ remover(H)
    i ⇓ e_i; j ⇓ e_j
    cr ⇓ c_{ij} – Y(i) – Y(j+|I|)
    Se cr < 0 ***então***
        encontrou ⇓ verdadeiro
Fim
Se não encontrou ***então***
Inicio
    Construir Lista de Prioridades
    Se vazia(H) ***então***
        Arco de saída não encontrado
        Parar algoritmo Simplex, ótimo encontrado
    Senão
    Inicio
        e ⇓ remover(H)
        i ⇓ e_i; j ⇓ e_j
    Fim
Fim
----- Fim Algoritmo 27 -----

O laço 1 realiza a busca por um custo reduzido negativo na lista de prioridades *H*. Caso a busca não tenha sucesso, isto é, caso a lista de prioridades *H* fique vazia, o algoritmo de construção da lista de prioridades (ContruirListaPrioridade) será executado. Isso fará com que sejam calculados todos os custos reduzidos, e a lista *H* terá as variáveis com custo reduzido negativo. Caso *H* esteja vazia, é porque não existe nenhuma variável para entrar na base, ocasionando o final do método Simplex. Caso contrário, a primeira variável da lista *H* será inserida na base, a qual tem o menor custo reduzido negativo.

6.3.5 Encontrar a variável para sair da base

A entrada da variável x_{ij} na base significa que o arco *(i,j)* irá pertencer à árvore geradora, mais precisamente o arco *(i,j+|I|)*. Será utilizada a notação (i, j) no lugar de (i,j+|I|) durante as explicações. Com a entrada do arco *(i,j)* na árvore, gera-se um ciclo na mesma, o qual deve ser eliminado com a saída de um arco que pertence ao ciclo. O primeiro passo, então, é identificar o ciclo formado pela inclusão do arco (*i,j*). O arco de saída é o arco em que seu fluxo se torna zero, primeiro, quando é aumentado o fluxo no arco (*i,j*). O leitor pode verificar esta afirmação na teoria de fluxo em redes, a qual pode ser encontrada facilmente na literatura. Pode-se citar Goldbarg e Luna [26], Ravinfra, Magnanti e Orlin [35], Jacobsen [29].

Em cada nível da árvore geradora se encontra um determinado tipo de centro, isto é, no nível *k* da árvore geradora, há nós que representam centros de oferta ou nós que representam centros de demanda. Os níveis são alternados; se o nível *k* representar nós de centros de oferta, o nível *k+1* representará nós de centros de demanda, ou vice-versa. É fácil verificar que essas afirmativas são verdadeiras, pois, caso contrário, ter-se-ia centros de demanda atendendo centros de demanda e/ou centros de oferta atendendo centros de oferta. Assim, e considerando que o fluxo sempre sai de um centro de oferta para um centro de demanda, o fluxo no ciclo, quando aumentado o fluxo no arco *(i,j)*, irá, alternadamente, aumentar e diminuir. A Figura 23 mostra a direção do fluxo do exemplo representado na Figura 22. Se o arco (4,5) for acrescentado, conforme mostrado, em cinza tracejado, na Figura 24 (lembrando que na Figura 24 os centros de oferta são representados com números *j+|I|*), os arcos pontilhados ((4,9), (3,7), (2,6) e (1,5)) e tracejados ((3,9), (2,7) e (1,6)) pertencem ao ciclo formado pelo acréscimo. Nos arcos pontilhados o fluxo será diminuído, enquanto que nos arcos tracejados o fluxo será aumentado, quando for acrescentado o fluxo no arco cinza tracejado (4,1). Sendo assim, o arco que sairá da base será um arco pontilhado, aquele que chegar a zero primeiro; no exemplo, qualquer um entre (4,9), (3,7), e (2,6).

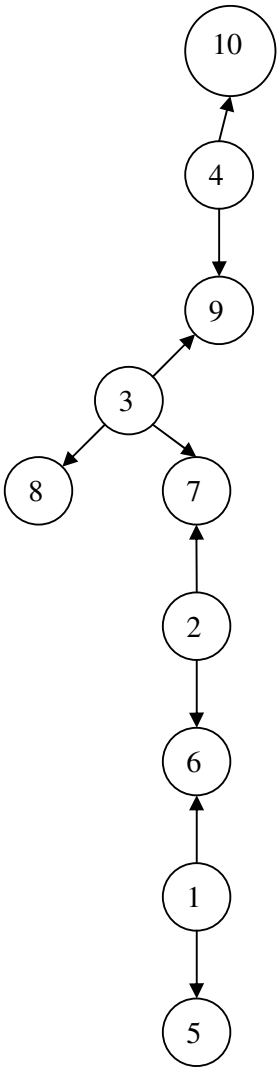


Figura 23 – Direção dos fluxos

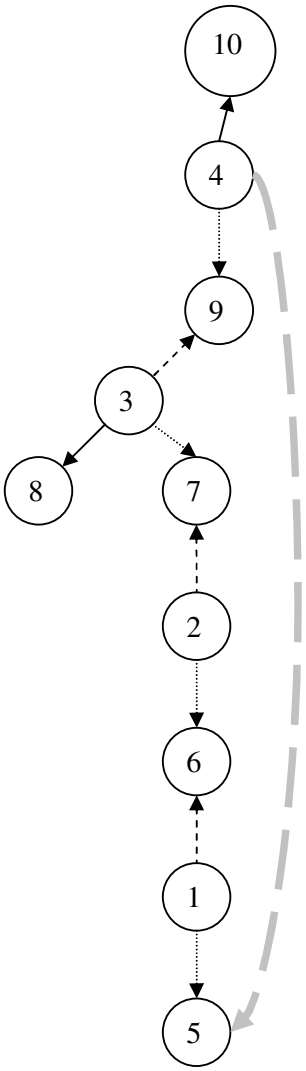


Figura 24 – Acrescimo do arco (4,5)

O Algoritmo 28 encontra o ciclo formado com a inserção do arco (*i,j*), o arco que sairá da base e o fluxo que deve ser acrescentado e diminuído nos arcos. Dados *i* e *j*, para encontrar o ciclo, o algoritmo vai subindo na árvore, a partir dos nós *i* e *j*+|*I*|, até encontrar um nó *v* comum ao dois caminhos, identificando, com isso, o ciclo. A variável *v1* subirá na sub-árvore do nó *i* e a variável *v2* subirá na sub-árvore do nó *j*+|*I*|. Quando a variável *v1* for um centro de oferta, verifica-se o quanto se pode diminuir o fluxo *X(v1)*, ou seja, quanto se pode diminuir a variável $x_{v1\ P(v1-|I|)}$, pois a mesma terá seu fluxo diminuído, quando o arco (*i,j*) for acrescentado. Quando a variável *v2* for um centro de demanda, verifica-se o quanto se pode diminuir o fluxo *X(v2)*, ou seja, o quanto se pode

diminuir a variável $x_{P(v2)\ v2+|I|}$, pois a mesma terá seu fluxo diminuído, quando o arco (*i,j*) for acrescentado. A variável *filho* representa uma das extremidades do arco que sairá da base, a outra extremidade será dada por *P(filho)*. A variável *fluxo* representará o valor da variável x_{ij}, isto é, o valor que passará no arco *(i,j)*.

Algoritmo 28: Encontrar Arco de Saída

```
v1 ⇓ i
v2 ⇓ j + |I|
fluxo ⇓ ∞;
Enquanto v1 ≠ v2 fazer
Inicio
        Se A(v1) > A(v2) então
        Inicio
                Se v1 ≤ |I| então
                        Se X(v1) < fluxo então
                        Inicio
                                fluxo = X(v1);
                                filho = v1;
                        Fim
                v1 = P(v1);
        Fim
        Senão
        Inicio
                Se v2 ≥ |I| então
                        Se X(v2) < fluxo então
                        Inicio
                                fluxo = X(v2);
                                filho = v2;
                        Fim
                v2 = P(v2);
        Fim
Fim
----- Fim Algoritmo 28 -----
```

A condição *A(v1) > A(v2)* verifica se a busca pelo ciclo subirá pela sub-árvore do nó *i* ou *j*. Caso *A(v1) > A(v2)* seja verdadeira, o algoritmo subirá pela sub-árvore de *i*; caso contrário, subirá pela sub-árvore de *j*+|*I*|.

6.3.6 Atualização da árvore geradora (geração da nova solução básica)

Encontrado o arco de entrada e o arco de saída, é necessário atualizar a árvore geradora, os fluxos e as variáveis duais da nova base. Para atualização da árvore, o

primeiro passo da atualização é verificar se o arco a ser inserido será representado na árvore por $P(i)=j$ ou $P(j)=i$.

6.3.6.1 Ordem do arco de entrada

A inclusão do arco (i,j) forma um ciclo na árvore. Com isso, haverá dois caminhos para se chegar de i a j ou vice-versa: um é formado pelo caminho existente na árvore, e o outro é formado pelo próprio arco (i,j). Seja v o nó, de menor altura, comum aos dois caminhos, o qual no final do Algoritmo 28 é igual a $v1$ e $v2$, então v pode ter três valores distintos, são eles:

1. $v = i$
2. $v = j$
3. $v \neq i$ e $v \neq j$

A Figura 25 mostra, de uma maneira geral, um exemplo do primeiro caso $v = i$. Na árvore existem dois caminhos de i para j ou vice-versa, o próprio arco (i,j), em cinza, e o segundo o caminho já existente na árvore, representado pelos arcos pontilhados. Sabe-se que o arco a sair da base deve ser um arco pontilhado. Com isso, na atualização da árvore, deve ser feito $P(j) = i$; caso contrário, a árvore ficaria desconexa. É fácil ver, nesse caso, que a condição $A(v1) > A(v2)$ do Algoritmo 28 sempre será falsa. Então, a variável *filho*, do mesmo algoritmo, será um centro de demanda. Pode-se, também, concluir que a variável *filho* está no caminho que liga v a j, porém, em uma altura menor ou igual à de j.

A Figura 26 mostra, de uma maneira geral, um exemplo do segundo caso $v = j$. Na árvore existem dois caminhos de i para j ou vice-versa, o próprio arco (i,j), em cinza, e o segundo o caminho já existente na árvore, representado pelos arcos pontilhados. Sabe-se que o arco a sair da base deve ser um arco pontilhado. Com isso, na atualização da árvore, deve ser feito $P(i) = j$; caso contrário, a árvore ficaria desconexa. É fácil ver, nesse caso, que a condição $A(v1) > A(v2)$ do Algoritmo 28 sempre será verdadeira. Então, a variável *filho*, do mesmo algoritmo, será um centro de oferta. Pode-se, também, concluir que a variável *filho* está no caminho que liga v a i, porém, em uma altura menor ou igual à de i.

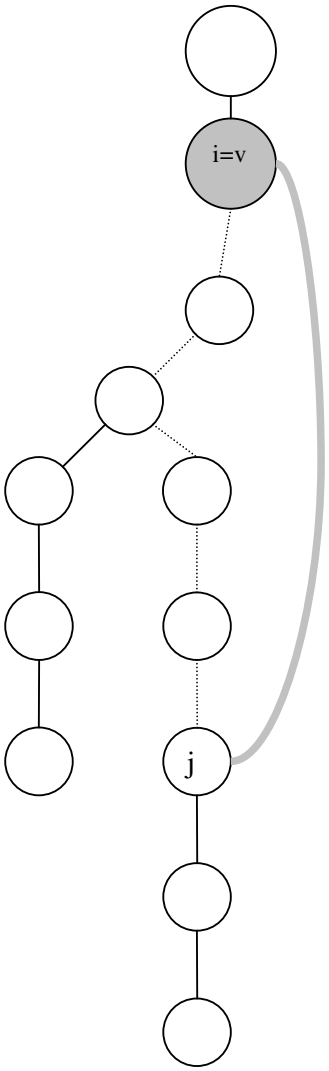


Figura 25 – exemplo caso *v = i*

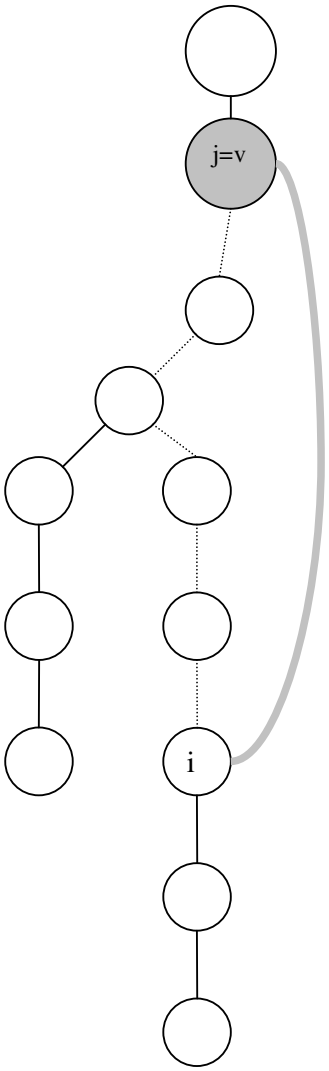


Figura 26 – exemplo caso *v = j*

A Figura 27 mostra, de uma maneira geral, um exemplo do terceiro caso $v \neq i$ e $v \neq j$. Na árvore existem dois caminhos de *i* para *j* ou vice-versa, o próprio arco *(i,j)*, em cinza, e o segundo o caminho já existente na árvore, representado pelos arcos tracejados e pontilhados. Sabe-se que arco a sair da base deve ser um arco tracejado ou pontilhado. Com isso, se um arco tracejado sair da base, deve ser feito *P(j) = i*; caso contrário, a árvore ficaria desconexa. Se um arco pontilhado sair da base, deve ser feito *P(i) = j*/ caso contrário, a árvore ficaria desconexa. Olhando para a condição *A(v1) > A(v2)* do Algoritmo 28, verifica-se que se um arco tracejado sair da base, a variável *filho*, do mesmo algoritmo, será um centro de demanda e estará no caminho que liga *v* a *j*, porém,

em uma altura menor ou igual à de *j*. Se um arco pontilhado sair da base, a variável *filho*, do Algoritmo 28, será um centro de oferta e estará no caminho que liga *v* a *i*, porém, em uma altura menor ou igual à de *i*.

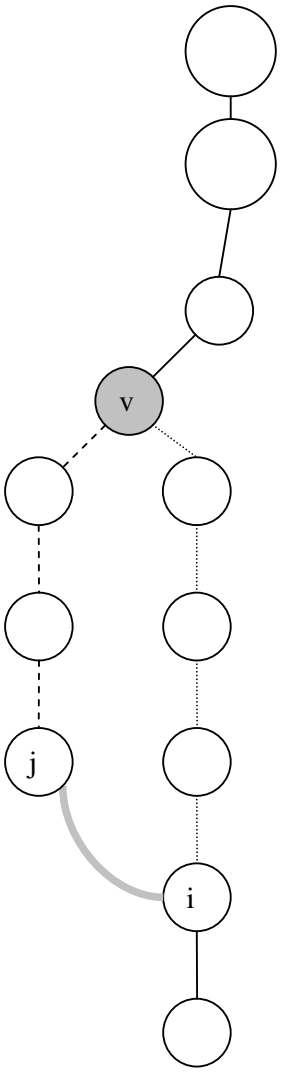


Figura 27 - Exemplo caso $v \neq i$ e $v \neq j$

As considerações realizadas sobre a ordem do arco *(i,j)* a entrar na base podem ser resumidas da seguinte forma:

- Se $v = i$, então $P(j) = i$ e a variável *filho* é um centro de demanda que está no caminho de *v* a *j*, porém, com altura menor ou igual a *j*.

- ¬ Se $v = j$, então $P(i) = j$ e a variável *filho* é um centro de oferta que está no caminho de v a i, porém, com altura menor ou igual a i.
- ¬ Se $v \neq i$, $v \neq j$ e o arco de saída estiver no caminho v a j, então $P(j) = i$ e a variável *filho* é um centro de demanda que está no caminho de v a j, porém, com altura menor ou igual a j
- ¬ Se $v \neq i$, $v \neq j$ e o arco de saída estiver no caminho v a i, então $P(i) = j$ e a variável *filho* é um centro de oferta que está no caminho de v a i, porém, com altura menor ou igual a i.

Então, para saber qual a ordem do arco de entrada *(i,j)*, na árvore geradora, basta olhar para a variável *filho* do Algoritmo 28. Se a variável *filho* for um centro de oferta, deve-se fazer $P(i) = j$; caso contrário, deve-se fazer $P(j) = i$. Outra conclusão é que a variável *filho* sempre estará entre o caminho que liga v a k, onde k é o filho no arco de entrada *(i,j)*.

Seja *(i,j)* o arco de entrada, o Algoritmo 29 identifica a maneira que o mesmo será representado na árvore geradora, fornecendo o arco (n_1, n_2) tal que, na árvore geradora, $P(n_2) = n_1$.

Algoritmo 29: Encontra ordem do Arco de Entrada

Se *filho* ≤ |I| ***então***
Inicio
$n_1 \Downarrow$ i
$n_2 \Downarrow$ j
Fim
Senão
Inicio
$n_1 \Downarrow$ j
$n_2 \Downarrow$ i
Fim
----- Fim Algoritmo 29 -----

6.3.6.2 Atualização do fluxo

É necessário atualizar o fluxo dos arcos que pertencem ao ciclo formado pela entrada do arco (i,j). Embora o Algoritmo 28 encontre o ciclo e identifique quais arcos devem ser aumentados e quais devem ser diminuídos, não é possível atualizar os fluxos no mesmo, porque somente após o seu término é identificado o fluxo que deve passar pelo ciclo. Sendo assim, o algoritmo de atualização do fluxo dos arcos, Algoritmo 30, é

quase idêntico ao Algoritmo 28, onde ao invés de verificar o maior fluxo passante pelo ciclo, atualiza os fluxos.

Algoritmo 30: Atualizar Fluxos

```
v1 ⇓ i
v2 ⇓ j + |I|
Enquanto v1 ≠ v2 fazer
Inicio
        Se A(v1) > A(v2) então
        Inicio
                Se v1 ≤ |I| então
                        X(v1) ⇓ X(v1) – fluxo
                Senão
                        X(v1) ⇓ X(v1) + fluxo
                v1 = P(v1);
        Fim
        Senão
        Inicio
                Se v2 ≥ |I| então
                        X(v2) ⇓ X(v2) – fluxo
                Senão
                        X(v2) ⇓ X(v2) + fluxo
                v2 = P(v2);
        Fim
Fim
----- Fim Algoritmo 30 -----
```

6.3.6.3 Atualização da árvore e das variáveis duais

Após ser identificado qual arco entrará na base, qual sairá, qual a ordem do arco de entrada e ter sido atualizado o fluxo dos arcos, deve-se atualizar a árvore geradora e as variáveis duais. Seja (n_1,n_2) o arco de saída, tal que $n_1 = P(n_2)$, seja (q_1,q_2) o arco de entrada, tal que $q_1 = P(q_2)$, e seja v o nó, de menor altura, do ciclo formado pela inclusão do arco (q_1,q_2), que no final do Algoritmo 28 é igual a *v1* e *v2*. O Algoritmo 31 mostra como é realizada a atualização da árvore geradora.

Além da inclusão do arco (q_1,q_2) na árvore geradora, deve ser atualizada a sub-árvore de n_2, a qual deve passar a ser sub-árvore de q_2, isto é, todos os nós onde n_2 é antecessor, inclusive o mesmo, devem ser descendentes de q_2. Essa atualização é necessária porque a conexão de n_2 na árvore, depois da saída do arco (n_1,n_2), passa a ser através de q_2 Na seção 6.3.6.1 foi identificado que n_2 está no caminho de v a q_2, porém, com uma altura menor ou igual a v. Com isso, a idéia básica do algoritmo de atualização da árvore é ir trocando, a partir de q_2 até n_2, a ordem do pai e filho, ou seja, o filho passa

a ser pai e o pai passa a ser filho. Nessa troca é realizada a atualização de toda a sub-árvore do nó envolvido, atualizando as variáveis duais, as alturas dos nós e as direções dos fluxos. Mais informações sobre a atualização da árvore geradora podem ser encontras em Jacobsen [29]. A Figura 28 mostra a inclusão do arco (q_1,q_2), em cinza, e a retirada do arco (n_1,n_2). O Algoritmo 31 transforma a árvore da Figura 28 na árvore da Figura 29.

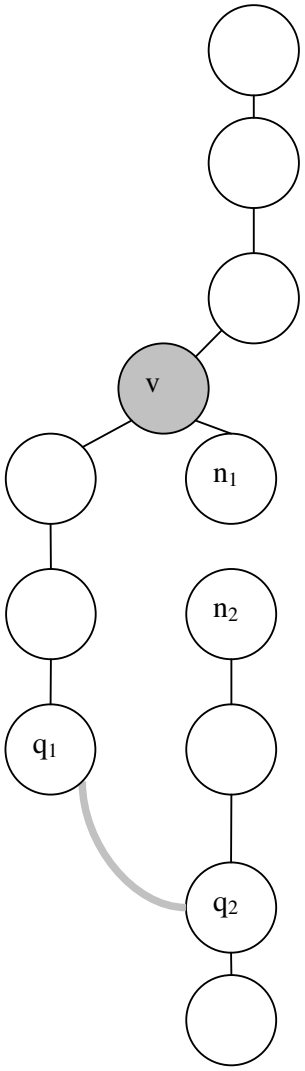


Figura 28 - Inclusão do arco (q_1, q_2)

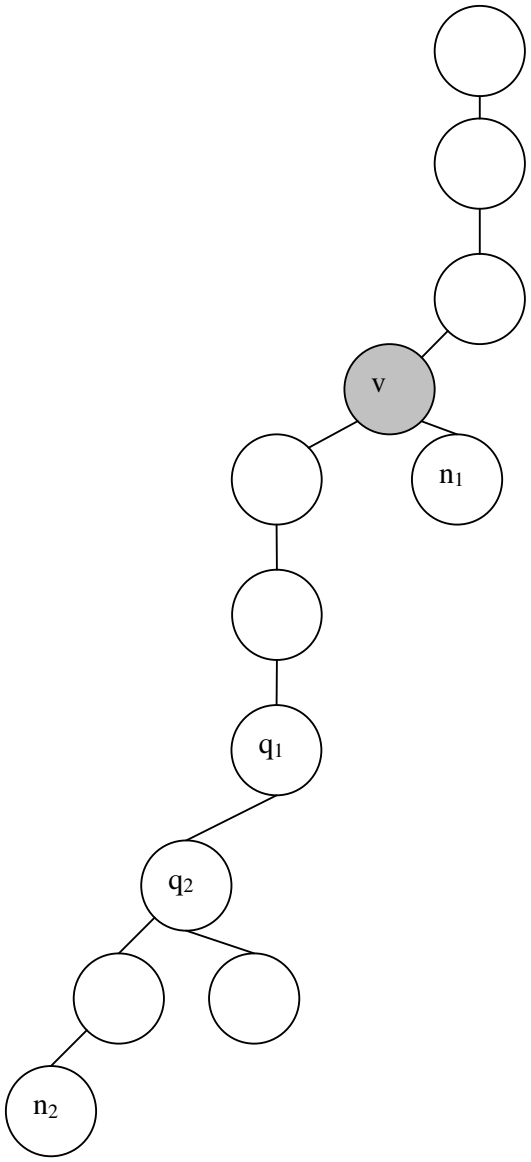


Figura 29 - Transformaçào da árvore após a inclusão do arco (q_1,q_2)

A atualização das variáveis duais e das alturas dos nós é realizada conforme visto na seção 6.3.6.3, contudo, somente nas partes onde a árvore é alterada. O Algoritmo 31 usa como sub-rotina o Algoritmo 32. Esse algoritmo recebe dois parâmetros, o parâmetro nó pai e o parâmetro nó filho. Ele retira o vértice filho da sub-árvore a que o mesmo pertence, colocando-o, com toda sua sub-árvore, como filho do vértice pai.

O primeiro passo do Algoritmo 32 é atualizar a variável dual e a altura do filho. O segundo passo é a identificação das sub-árvores do nó filho. Para a identificação das sub-árvores do vértice filho é utilizado um vetor auxiliar, denominado *M*. O vetor *M* tem tamanho |*I*| + |*J*|, e cada posição representa um nó da árvore. As primeiras |*I*| posições representam os centros de oferta, e as últimas |*J*| posições representam os centros de demanda. O valor de cada posição do vetor *M* será zero, se o nó não pertencer a uma das sub-árvores de filho, ou o valor do nó filho, caso o nó pertença a uma das sub-árvores de filho. *M* será preenchido conforme as sub-árvores forem sendo exploradas. Na identificação das sub-árvores do nó filho é realizada a atualização das variáveis duais e da altura dos nós pertencentes a essa sub-árvore.

Após a atualização das sub-árvores do nó filho, atualiza-se a sub-árvore de onde foi retirado o nó filho e incluído o nó filho como sendo filho de vértice pai. O Algoritmo 31, então, consiste de executar o Algoritmo 32 para cada nó do caminho q_2 até n_2 e atualizar a variável de fluxo do nó, pois a mesma, como visto na seção 6.3.1, contém o valor do fluxo do arco (vértice, P(vértice)).

Algoritmo 31: Atualizar Árvore e Variáveis Duais

pai ⇓ q_1
filho ⇓ q2
F_pai ⇓ fluxo
F_filho ⇓ F(filho)
Fazer
    Retirar Vértice filho e inserir arco (pai, filho) *{* Algoritmo 32:*}*
    { atualiza orientação do fluxo }
    F(filho) ⇓ F_pai
    F_pai ⇓ F_filho
    F_filho ⇓ F(P(filho))
    pai ⇓ filho
    filho ⇓ P(filho)
Enquanto pai ≠ n_2
----- Fim Algoritmo 31 -----

Algoritmo 32: Retirar Vértice filho e inserir arco (pai,filho)

{ atualiza a altura e variável dual de filho }
A(filho) ⇓ A(pai)+1
Se (filho ≤ |I|) ***então***
Inicio
    i ⇓ filho; j ⇓ pai - |I|
    Y(filho) ⇓ c_{ij} – Y(pai)
Fim

Senão
Inicio
    i ⇓ pai; j ⇓ filho - |I|
    Y(filho) ⇓ c_{ij} – Y(pai)
Fim

{inicializa marca da sub-árvore}
M(i) ⇓ 0 ∀ i ∈ T

{ percorre sub-árvore de filho e atualiza a altura e as variáveis duais }
M(filho) ⇓ filho
u1 ⇓ L(filho)
u2 ⇓ L(u1)
Enquanto (v ≠ 0 e M(P(u2)) = filho) ***fazer***
Inicio
    { atualiza a altura e a variável dual de i }
    A(u2) ⇓ A(P(u2))+1
    Se (u2 ≤ |I|) ***então***
    Inicio
        i ⇓ u2; j ⇓ P(u2) - |I|
        Y(u2) ⇓ c_{ij} – Y(P(u2))
    Fim
    Senão
    Inicio
        i ⇓ P(u2); j ⇓ u2 - |I|
        Y(u2) ⇓ c_{ij} – Y(P(u2))
    Fim
    u1 ⇓ u2
    M(u2) ⇓ filho
    u2 ⇓ L(u2)
Fim

{encontra o nó k que tem como o próximo o nó a ser retirado (nó filho)}
k ⇓ P(filho)
Enquanto L(k) ≠ filho ***fazer***
    k ⇓ L(k)

{ atualiza sub-árvore onde foi retirado o vértice filho }
L(k) ⇓ u2

{ inclui o arco (pai, filho) }
L(u1) ⇓ L(pai)
L(pai) ⇓ filho
P(filho) ⇓ pai
----- Fim Algoritmo 32 -----

6.3.7 Método Primal Simplex

Nas seções anteriores foram apresentados todos os passos que o algoritmo Simplex deve realizar. O Algoritmo 33 apresenta o método Simplex, que apenas reproduz os passos apresentados. É importante observar que a condição de parada do método Simplex foi discutida na seção 6.3.4. Ela é atingida quando não existirem mais custos reduzidos negativos, isto é, quando não existirem mais arcos de entrada. A expressão *Custo* ⇓ *Custo + (cr * fluxo)* atualiza o valor da função objetivo, onde *cr* é o custo reduzido do arco de entrada *(cr < 0)* e *fluxo* é o maior fluxo que se pode passar no ciclo formado pela inserção do arco de entrada na árvore. O Algoritmo 27: Encontrar Arco de Entrada identifica o valor de *cr*, e o Algoritmo 28: Encontrar Arco de Saída identifica o valor da variável *fluxo*. A variável *Custo* é inicializada no Algoritmo 23: Encontrar a Solução Básica Viável NWCR. Após a inicialização, seu valor somente é reduzido, pois c*r* sempre será negativo. Quando o método parar a variável *Custo*, terá o valor da solução ótima.

Algoritmo 33 Método Simplex

Encontrar a Solução Básica Viável
Encontrar Raiz da Árvore Geradora
Calcular Variáveis Duais e Altura
Construir Lista de Prioridades
Enquanto Encontrar Arco de Entrada ***Fazer***
Inicio
    Encontrar Arco de Saída
    Custo ⇓ Custo + (cr * fluxo)
    Atualizar Fluxos
    Encontrar ordem do Arco de Entrada
    Atualizar Árvore e Variáveis Duais
Fim
----- Fim Algoritmo 33 -----

6.3.8 Exemplo

Na seção 6.3.2 foi apresentado um exemplo de *PTC*, onde os dados do problema são representados na Tabela 39. Conforme visto na seção 6.3.2, a Tabela 53 mostra os índices *P, L, X* depois do passo Encontrar a Solução Básica Viável, isto é, depois da execução do Algoritmo 23.

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	7	9	10	1	2	3	3	4	*0*
L(i)	5	6	8	9	0	1	2	7	3	*4*
X(i)	*5*	*5*	*5*	*10*	*10*	*5*	*5*	*10*	*5*	*0*

Tabela 53 – índice após encontrar solução básica viável

Conforme visto na seção 6.3.3, a Tabela 54 mostra os índices *Y* e *A* depois dos passos Encontrar Raiz da Árvore Geradora e Calcular Variáveis Duais e Altura, isto é, depois da execução do Algoritmo 24 e do Algoritmo 25.

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
Y(i)	-45	-25	9	0	50	55	31	51	8	*0*
A(i)	*7*	*5*	*3*	*1*	*8*	*6*	*4*	*4*	*2*	*0*

Tabela 54 – valores das variáveis duais e altura dos nós

O próximo passo do algoritmo Simplex é Construir Lista de Prioridades, a qual está representada na Tabela 55.

Posições	**1**	**2**	**3**	**4**	**5**	**6**	**7**	***8***
I	3	4	3	2	4	4	4	*3*
J	2	4	1	4	1	2	3	*6*
cr	*-49*	*-26*	*-44*	*-19*	*-13*	*-26*	*-11*	*-9*

Tabela 55 – Lista de Prioridade

Nesse momento, é inicializado o processo iterativo do método Simplex, lembrando que o *Custo = 1205*, como foi visto na seção 6.3.2. Na seqüência, será mostrado o processo iterativo.

6.3.8.1 Iteração 1

Encontrar Arco de Entrada: (3,2) ◊ fazendo $j + |I| = (3,6)$

Encontrar Arco de Saída: (2,6)

Custo ⇓ 1205 + (-49 * 5) (Custo = 960)

A Figura 30 mostra o arco de entrada, em cinza, e o de saída pontilhado. A Tabela 56 mostra como ficou a lista de prioridades.

A Figura 31 e a Tabela 57 mostram como ficou a árvore após a aplicação dos seguintes passos:

Atualizar Fluxos

Encontrar ordem do Arco de Entrada

Atualizar Árvore e Variáveis Duais

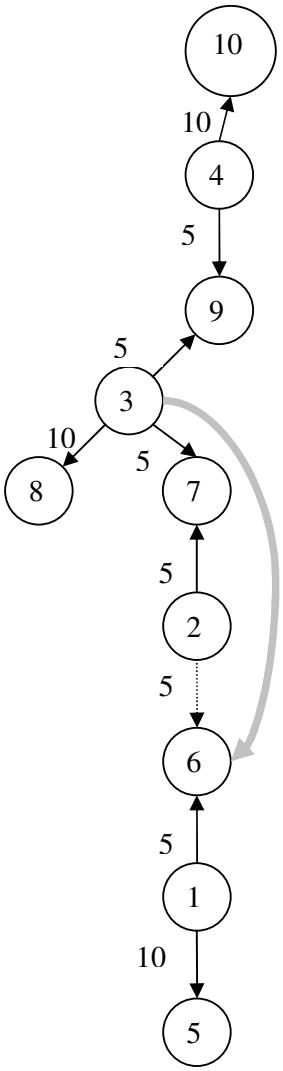


Figura 30 – arco de entrada (3,6) e arco de saída (2,5)

Posições	**1**	**2**	**3**	**4**	**5**	**6**	***7***
I	3	4	4	2	4	3	*4*
J	1	4	2	4	1	6	*3*
cr	*-44*	*-26*	*-26*	*-19*	*-13*	*-9*	*-11*

Tabela 56 – Lista de Prioridade

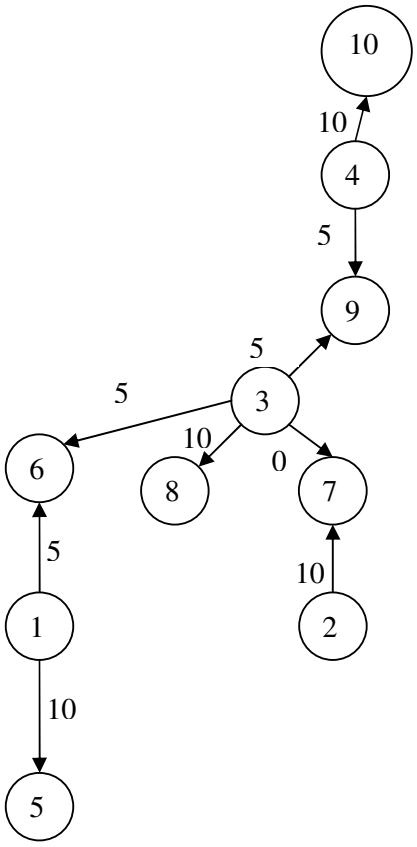


Figura 31 – Árvore após atualização

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	7	9	10	1	3	3	3	4	*0*
L(i)	5	0	6	9	8	1	2	7	3	*4*
X(i)	5	10	5	10	10	5	0	10	5	*0*
Y(i)	4	-25	9	0	1	6	31	51	8	*0*
A(i)	*5*	*5*	*3*	*1*	*6*	*4*	*4*	*4*	*2*	*0*

Tabela 57 – índices após atualização

6.3.8.2 Iteração 2

Encontrar Arco de Entrada: (4,4) ◊ fazendo $j + |I| = (4,8)$

Retira elemento da lista de prioridades (3,1)

Calcula custo reduzido ◊ $cr = 5$, logo não pode entrar na base

Retira elemento da lista de prioridades (4,4)

Calcula custo reduzido ◊ $cr = -26$, logo (4,4) entrar na base ◊ fazendo $j + |I| = (4,8)$

Encontrar Arco de Saída: (4,9)

Custo ⇓ 960 + (-26 * 5) (Custo = 830)

A Figura 32 mostra o arco de entrada, em cinza, e o de saída pontilhado, A Tabela 58 mostra como ficou a lista de prioridades.

A Figura 33 e a Tabela 59 mostram como ficou a árvore após a aplicação dos seguintes passos:

Atualizar Fluxos

Encontrar ordem do Arco de Entrada

Atualizar Árvore e Variáveis Duais

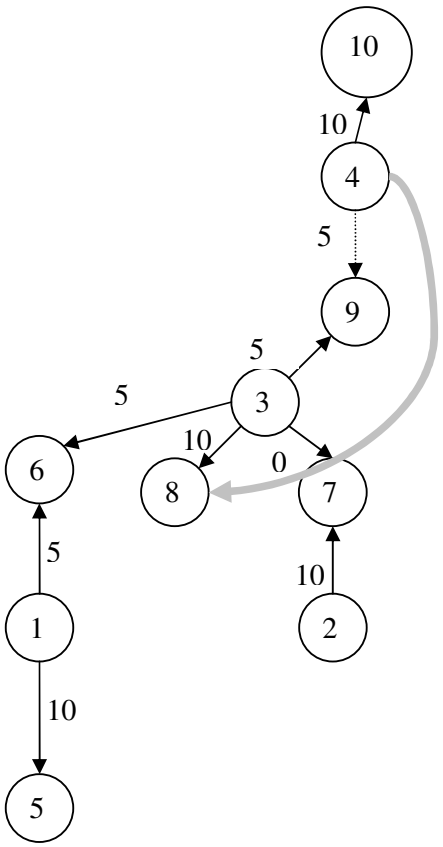


Figura 32 – arco de entrada (4,8) e arco de saída (4,9)

Posições	1	2	3	4	*5*
i	4	2	3	4	*4*
j	2	4	6	3	*1*
cr	*-26*	*-19*	*-9*	*-11*	*-13*

Tabela 58 – Lista de Prioridade

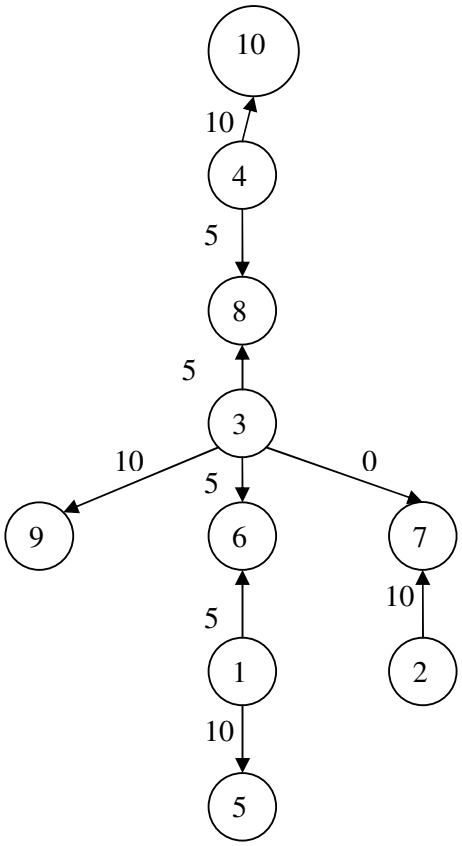


Figura 33 – Árvore após atualização

i	1	2	3	4	5	6	7	8	9	*10*
P(i)	6	7	8	10	1	3	3	4	3	*0*
L(i)	5	0	9	8	7	1	2	3	6	*4*
X(i)	5	10	5	10	10	5	0	5	10	*0*
Y(i)	30	1	35	0	-25	-20	5	25	-18	*0*
A(i)	*5*	*5*	*3*	*1*	*6*	*4*	*4*	*2*	*4*	*0*

Tabela 59 – índices após atualização

6.3.8.3 Iteração 3

Encontrar Arco de Entrada: (2,4) ◊ fazendo $j + |I| = (2,8)$

Retira elemento da lista de prioridades (4,2)

Calcula custo reduzido ◊ *cr = 49*, logo não pode entrar na base

Retira elemento da lista de prioridades (2,4)

Calcula custo reduzido ◊ *cr = -19*, logo (2,4) entrar na base ◊ fazendo $j + |I| = (2,8)$

Encontrar Arco de Saída: (3,8)

Custo ⇓ 830 + (-19 * 5) (Custo = 735)

A Figura 34 mostra o arco de entradas em cinzas e o de saída pontilhado. A Tabela 60 mostra como ficou a lista de prioridades.

A Figura 35 e a Tabela 61 mostram como ficou a árvore após a aplicação dos seguintes passos:

Atualizar Fluxos

Encontrar ordem do Arco de Entrada

Atualizar Árvore e Variáveis Duais

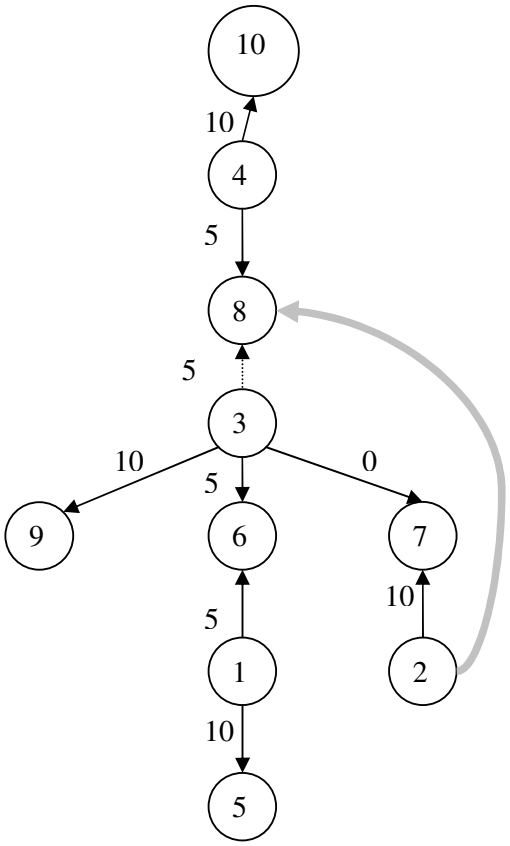


Figura 34 – arco de entrada (2,8) e arco de saída (3,8)

Posições	**1**	**2**	***3***
i	4	4	*3*
j	1	3	*6*
cr	*-13*	*-11*	*-9*

Tabela 60 – Lista de Prioridade

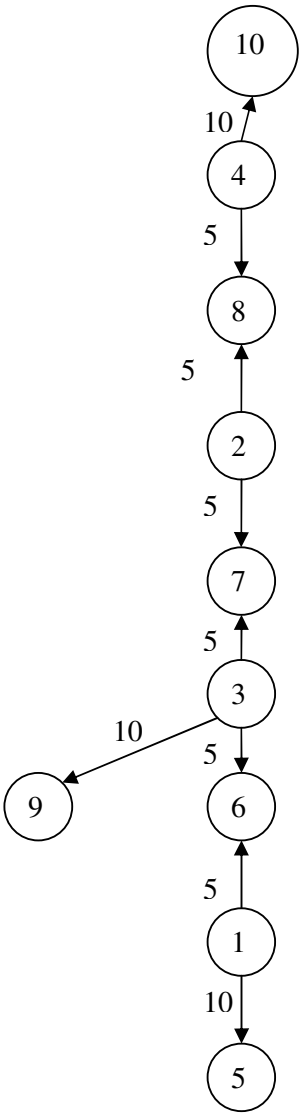


Figura 35 – Árvore após atualização

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	8	7	10	1	3	2	4	3	*0*
L(i)	5	7	9	8	0	1	3	2	6	*4*
X(i)	5	5	5	10	10	5	5	5	10	*0*
Y(i)	11	-18	16	0	-6	-1	24	25	1	*0*
A(i)	*7*	*3*	*5*	*1*	*8*	*6*	*4*	*2*	*6*	*0*

Tabela 61 – índices após atualização

6.3.8.4 Iteração 4

Encontrar Arco de Entrada: (4,7) ◊ fazendo $j + |I| = (4,7)$

Retira elemento da lista de prioridades (4,1)

Calcula custo reduzido ◊ *cr = 43*, logo não pode entrar na base

Retira elemento da lista de prioridades (4,3)

Calcula custo reduzido ◊ $cr = -4$, logo (4,3) entrar na base ◊ fazendo $j + |I| = (4,7)$

Encontrar Arco de Saída: (2,7)

Custo ⇓ 735 + (-4 * 5) (Custo = -715)

A Figura 36 mostra o arco de entrada, em cinza, e o de saída pontilhado. A Tabela 62 mostra como ficou a lista de prioridades.

A Figura 37 e a Tabela 63 mostram como ficou a árvore após a aplicação dos seguintes passos:

Atualizar Fluxos

Encontrar ordem do Arco de Entrada

Atualizar Árvore e Variáveis Duais

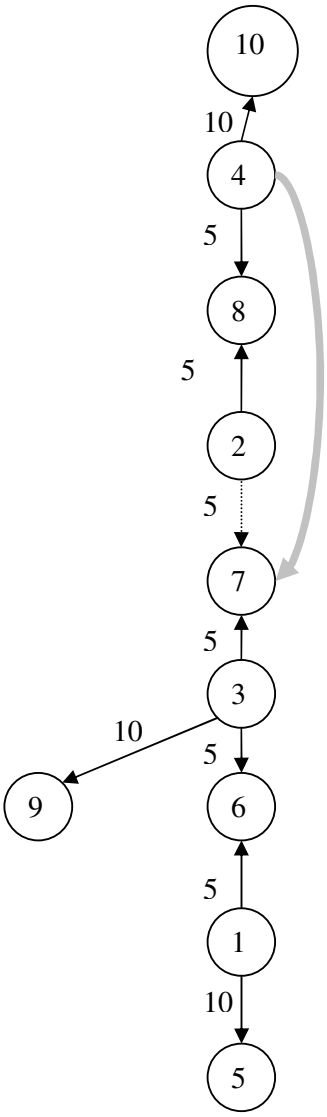


Figura 36 – arco de entrada (4,7) e arco de saída (2,7)

Posições	***1***
i	*3*
j	*6*
cr	*-9*

Tabela 62 – Lista de Prioridade

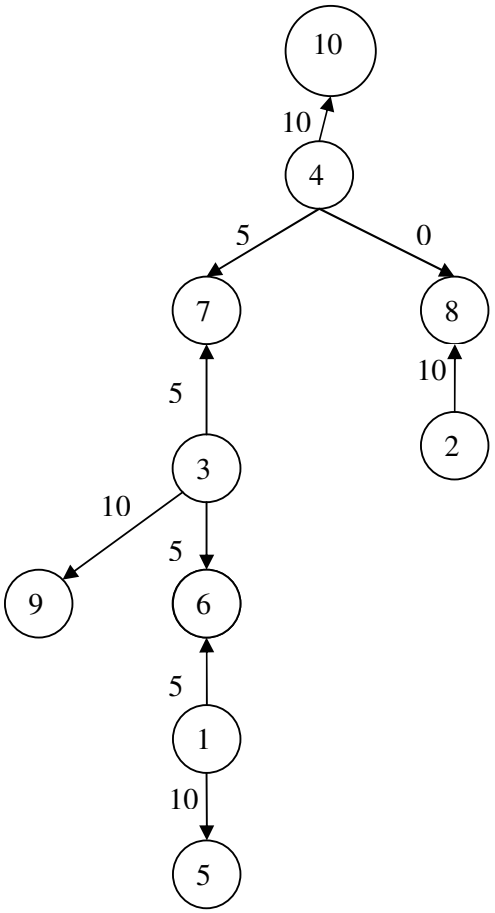


Figura 37 – Árvore após atualização

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	8	7	10	1	3	4	4	3	*0*
L(i)	5	0	9	7	8	1	3	2	6	*4*
X(i)	5	10	5	10	10	5	5	0	10	*0*
Y(i)	15	-18	20	0	-10	-5	20	25	-3	*0*
A(i)	*5*	*3*	*3*	*1*	*6*	*4*	*2*	*2*	*4*	*0*

Tabela 63 – índices após atualização

6.3.8.5 Iteração 5

Encontrar Arco de Entrada: (3,6) ◊ fazendo $j + |I| = (3,10)$

Retira elemento da lista de prioridades (3,6)

Calcula custo reduzido ◊ $cr = -20$, logo (3,6) entrar na base ◊ fazendo $j + |I| = (3,10)$

Encontrar Arco de Saída: (3,10)

Custo ⇓ 715 + (-20 * 5) (Custo = 615)

A Figura 38 mostra o arco de entrada, em cinza, e o de saída pontilhado. Nesse momento, a lista de prioridades se tornou vazia.

A Figura 39 e a Tabela 64 mostram como ficou a árvore após a aplicação dos seguintes passos:

Atualizar Fluxos

Encontrar ordem do Arco de Entrada

Atualizar Árvore e Variáveis Duais

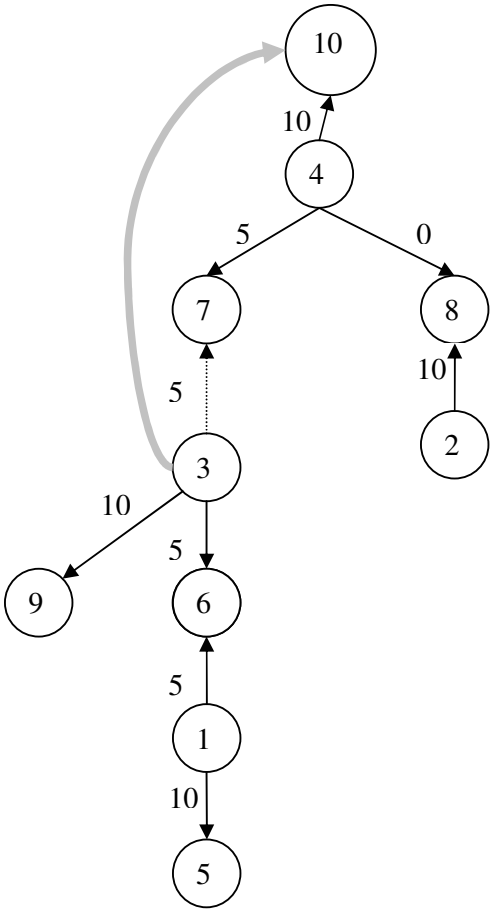


Figura 38 – arco de entrada (3,10) e arco de saída (3,7)

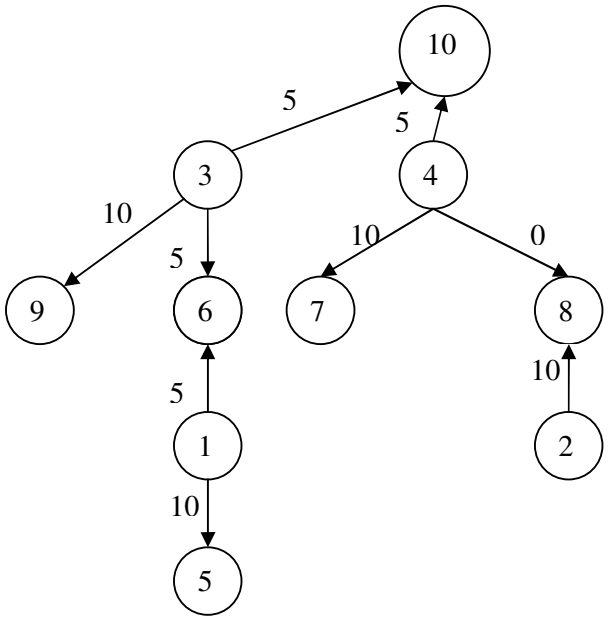


Figura 39 – Árvore após atualização

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	8	10	10	1	3	4	4	3	*0*
L(i)	5	0	9	7	4	1	8	2	6	*3*
X(i)	5	10	5	5	10	5	10	0	10	*0*
Y(i)	-5	-18	0	0	10	15	20	25	17	*0*
A(i)	*3*	*3*	*1*	*1*	*4*	*2*	*2*	*2*	*3*	*0*

Tabela 64 – índices após atualização

6.3.8.6 Iteração 6

Encontrar Arco de Entrada: (4,5) ◊ fazendo $j + |I| = (4,9)$

Nesse momento, a lista de prioridades está vazia. Com isso, conforme o Algoritmo 27, deve ser executado o passo Construir Lista de Prioridades. A Tabela 65 mostra o cálculo dos custos reduzidos e a Tabela 66 mostra a lista de prioridades.

Centros Demanda/ Centros Oferta	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	***D_6***
O_1	0	0	15	20	16	*5*
O_2	33	33	4	0	41	*18*
O_3	5	0	20	35	0	*0*
O_4	*27*	*14*	*0*	*0*	*-9*	*0*

Tabela 65 – Custos Reduzidos

Posições	***1***
i	*4*
j	*5*
cr	*-9*

Tabela 66 – Lista de Prioridade

Retira elemento (4,5) da lista de prioridade para entrar na base ◊ fazendo $j + |I| =$ (4,9)

Encontrar Arco de Saída: (4,10)
Custo ⇓ 615 + (-9 * 5) (Custo = 570)

A Figura 40 mostra o arco de entrada, em cinza, e o de saída pontilhado. Nesse momento, a lista de prioridades se tornou vazia.

A Figura 41 e a Tabela 67 mostram como ficou a árvore após a aplicação dos seguintes passos:

Atualizar Fluxos
Encontrar ordem do Arco de Entrada
Atualizar Árvore e Variáveis Duais

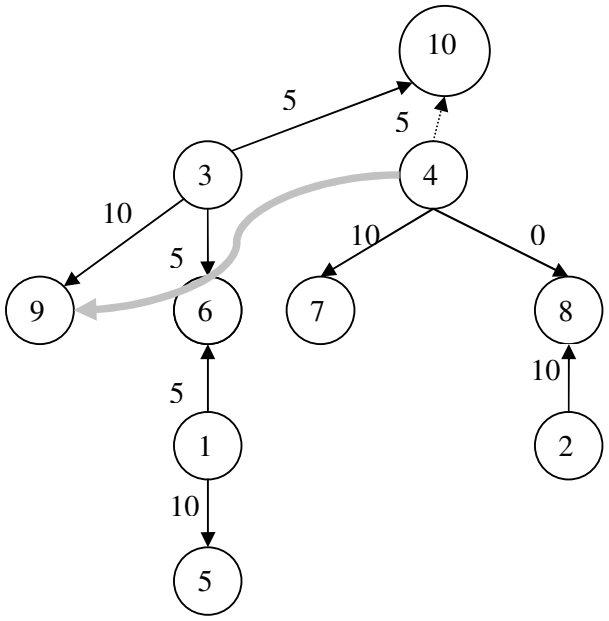


Figura 40 – arco de entrada (4,9) e arco de saída (4,10)

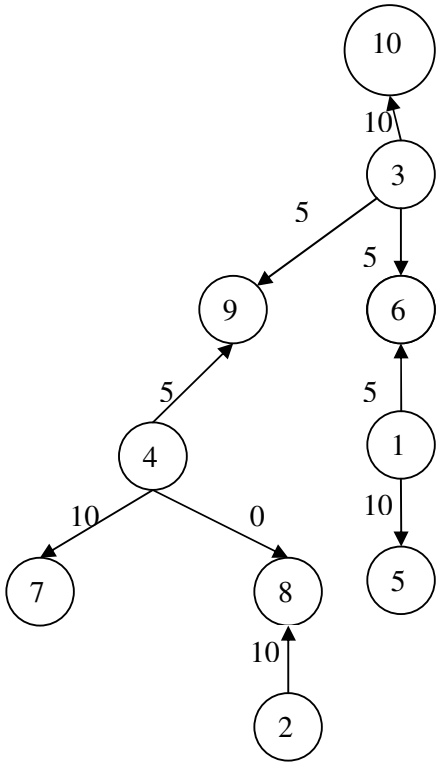


Figura 41 – Árvore após atualização

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***10***
P(i)	6	8	10	9	1	3	4	4	3	*0*
L(i)	5	6	9	7	0	1	8	2	4	*3*
X(i)	5	10	10	5	10	5	10	0	5	*0*
Y(i)	-5	-27	0	-9	10	15	29	34	17	*0*
A(i)	*3*	*5*	*1*	*3*	*4*	*2*	*4*	*4*	*2*	*0*

Tabela 67 – índices após atualização

6.3.8.7 Iteração 7

Encontrar Arco de Entrada:

Nesse momento, a lista de prioridades está vazia. Com isso, conforme o Algoritmo 27, deve executar o passo Construir Lista de Prioridades. A Tabela 68 mostra o cálculo dos custos reduzidos. Como todos os custos são maiores ou iguais a zero, identifica-se, assim, que o algoritmo atingiu a sua condição de parada; portanto, a solução ótima é 570.

Centros Demanda/ Centros Oferta	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	***D_6***
O_1	0	0	6	11	16	*5*
O_2	42	42	4	0	50	*27*
O_3	5	0	11	26	0	*0*
O_4	*36*	*23*	*0*	*0*	*0*	*9*

Tabela 68 – Custos reduzidos

6.4 Adaptação do PTC ao PLC

O algoritmo de resolução do *PLC* utiliza o *PTC* em vários momentos, com propósitos distintos. Embora os propósitos sejam distintos, como realizar testes de reduções e calcular limites inferiores, existe, em comum, a necessidade de se inserir e retirar centros de oferta. Um centro de oferta no *PTC*, como será visto nos próximos capítulos, será uma facilidade aberta para o *PLC*. No algoritmo de resolução do *PLC*, serão necessários a abertura e o fechamento de facilidades. Assim, no algoritmo, haverá a necessidade de inclusão e retirada de centros de oferta no *PTC*. Com a inclusão e/ou retirada de centros de oferta de um *PTC*, a idéia é que o problema não seja reinicializado.

Na inclusão e/ou retirada de centros de oferta, o *PTC* deve ser adaptado às mudanças e resolvido novamente, partindo da solução adaptada, isto é, partindo da solução básica obtida na inclusão e/ou retirada de centros de oferta.

Na resolução do *PLC*, conforme será visto no Capítulo 4, todos os *PTC* utilizados terão, na primeira solução, como centros de oferta, todas as facilidades. Após isso, somente serão acrescentados e retirados centros de oferta, sempre tomando o cuidado para não tornar os problemas inviáveis, ou seja, com a quantidade de oferta menor que a quantidade de demanda. Antes de os *PTC's* serem resolvidos pela primeira vez, será incluído o centro de demanda virtual, como visto na seção 6.3. Isso adequará o problema ao método de solução proposto e fará com que o centro de demanda virtual seja a raiz da árvore geradora, quando for encontrada a primeira solução básica. Detalhes podem ser vistos na seção 6.3.

O vetor *A*, que representa a altura do nó na árvore, será utilizado para identificar se o elemento, no caso a facilidade, pertence ou não ao problema. Caso a facilidade *i* seja um centro de oferta pertencente ao *PTC*, então *A(i)* terá o valor da altura, onde *i* se encontra na árvore; caso contrário, *A(i) = -1*. Com isso, deve-se adaptar o Algoritmo 26 Construir Lista de Prioridades para que o mesmo considere se o centro de oferta pertencer ao problema ou não.

A inclusão de um centro de oferta no *PTC* é muito simples: basta aumentar a demanda do centro de demanda virtual e fazer com que toda a oferta, do centro inserido, seja direcionada para o centro de demanda virtual. Seja *v* o centro de demanda virtual e *i* o centro de oferta a ser incluído, o Algoritmo 34 mostra a inclusão do centro de oferta *i*. É claro que *i* não deve pertencer ao problema.

Algoritmo 34: Inserir Centro de Oferta
$P(i) \Downarrow \; v+|I|$
$L(i) \Downarrow \; L(v+|I|)$
$L(v+|I|) \Downarrow \; i$
$F(i) \Downarrow \; a_i$
$b_{v+|I|} \Downarrow \; b_{v+|I|} + a_i$
$Y(i) \Downarrow \; c_{iv} - Y(v+|I|)$
$A(i) \Downarrow \; A(v+|I|) + 1$
----- Fim Algoritmo 34 -----

Após a inclusão da facilidade *i*, para encontrar a solução ótima, deve ser executado o método Simplex sem o processo de inicialização, isto é, deve ser executado o algoritmo de Pós-Otimização.

Algoritmo 35: Método Simplex Pós-Otimização
Construir Lista de Prioridades
Encontrar Arco de Entrada ***Fazer***
Inicio
    Encontrar Arco de Saída
    Custo ⇓ Custo + (cr * fluxo)
    Atualizar Fluxos
    Encontrar ordem do Arco de Entrada
    Atualizar Árvore e Variáveis Duais
Fim
----- Fim Algoritmo 35 -----

A retirada de um centro de oferta exige um pouco mais de trabalho computacional do que a inclusão de um centro de oferta. O primeiro passo do algoritmo é colocar o centro de oferta *i*, a ser retirado, como filho mais à esquerda do centro de demanda virtual. Com isso, a demanda do centro de demanda $j = P(i)$ será suprida por outros centros de oferta. O segundo passo é fazer com que os centros de demanda, que na árvore são filhos do vértice *i*, sejam atendidos pelos centros de oferta que transportam sua capacidade excedente ao centro de demanda virtual.

Seja *i* o centro de oferta a ser retirado e *v* o centro de demanda virtual, então o nó que representa o centro de oferta *i* pode estar em três posições diferentes na árvore geradora. São elas:

1. Ser o filho mais à esquerda de *v*
2. Ser filho de *v*, mas não o mais à esquerda.
3. Não ser filho de *v*

No primeiro caso, *i* já atende a condição do primeiro passo, não é necessário realizar nada. No segundo caso, somente é necessário trocar o vértice *i* de posição. Já no terceiro caso, é necessário acrescentar o arco (v,i) e retirar, do ciclo formado, o arco que primeiro se tornar zero com o aumento do fluxo em (v,i). O Algoritmo 36 realiza a tarefa de colocar *i* como filho mais à esquerda de *v*. A variável *u1* representa o primeiro

vértice, se existir, no percurso a partir de *i*, que não pertence à sub-árvore de *i*. A variável *u2* representa o vértice anterior, no percurso de *v* a *i* , ao vértice *i*.

Algoritmo 36: Passo 1

```
{ caso 2 }
Se (P(i) = v e L(v) ≠ i) então
Inicio
        u1 = i;
        Enquanto ((P(L(u1)) ≠ v) e (L(u1) ≠ 0) fazer
                u1 ⇓ L(u1)

        u2 ⇓ L(v);
        Enquanto(L(u2) ≠ i) fazer
                u2 ⇓ L(u2)
        L(u2) ⇓ L(u1)
        L(u1) ⇓ L(v)
        L(v) ⇓ i
Fim

{ caso 3 }
Se (P(i) ≠ v ) então
Inicio
        { arco de entrada (i,j) }
        j ⇓ v - |I|
        cr ⇓ c_ij – Y(i) – Y(j+|I|)
        Encontrar Arco de Saída
        Custo ⇓ Custo + (cr * fluxo)
        Atualizar Fluxos
        Atualizar Árvore e Variáveis Duais
Fim
----- Fim Algoritmo 36 -----
```

No segundo passo, devem ser incluídos arcos entre os centros de oferta que atendem o centro de demanda virtual e os centros de demanda atendidos por *i*. Essa inclusão deve terminar quando não existir nenhum centro de demanda atendido por *i*. Quando essa condição for alcançada, faz-se *L(v)* ⇓ *L(i)* e *A(i)* ⇓ *-1*.

6.5 Resultados Computacionais

O *PTC* é utilizado como sub rotina para resolver o *PLC*. Portanto, é preciso assegurar que o método de transporte desenvolvido fornece bons resultados, práticos, em relação ao tempo de processamento.

Para que possa ser verificado que o método utilizado para resolução do problema de transporte é eficiente, serão realizadas comparações com o *software* comercial XPRESS, o qual tem mostrado ser um dos melhores *softwares* para resolução de problemas lineares.

O método desenvolvido para a resolução do *PTC* será chamado de *MPTC* (Método para resolução do Problema de Transporte Capacitado) . Para que a comparação entre o *MPT* e o XPRESS possa ser realizada, são necessários problemas teste. Não foi possível encontrar uma base de problemas testes, por isso foi desenvolvido um gerador de problemas.

6.5.1 Gerador de problemas

Os dados dos problemas serão gerados de forma aleatória. Contudo, para que o processo não fique somente sob responsabilidade do sistema, o usuário poderá fornecer algumas diretivas, possibilitando a geração de problemas mais próximos às necessidades do usuário.

Para que sejam gerados problemas com um certo grau de dificuldade, o gerador gerará problemas não euclidianos em relação aos custos de transporte, pois estes tendem a ser mais difíceis, isto é, são mais complicados de serem resolvidos. Embora essa seja a diretiva, não impede que o usuário possa gerar problemas fáceis. Para isso, deve-se fornecer parâmetros para que o sistema leve à geração de problemas mais fáceis.

6.5.1.1 Funções

A maioria das funcionalidades do sistema foi obtidas de forma empírica, baseada em experiências adquiridas. O sistema terá as seguintes funções:

6.5.1.1.1 Geração dos dados da demanda

As demandas serão geradas aleatoriamente entre um e o número máximo da demanda, a qual será especificada pelo usuário.

6.5.1.1.2 Geração dos dados de oferta

As ofertas serão geradas aleatoriamente, respeitando duas restrições. A primeira restrição a ser atendida é a viabilidade do problema, ou seja, $\sum_{i \in I} a_i \geq \sum_{j \in J} b_j$. A segunda restrição é a relação $\frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j}$, a qual será especificada pelo usuário.

6.5.1.1.2.1 Geração da oferta de cada centro de oferta

Para geração da oferta de cada centro de oferta será criado um vetor *v* com *n* elementos, onde cada elemento v_i do vetor *v* representará a percentagem que o centro de oferta *i* terá em relação à oferta total, ou seja, em relação a $O = \sum_{i \in I} a_i$. Por exemplo, seja 1000 a soma das ofertas; caso o elemento v_i tenha o valor 5, indicará que a capacidade do centro de oferta *i* será igual a 50.

O módulo de geração de demanda será realizado antes do módulo de geração de oferta. Seja *D* o somatório de toda a demanda e *r* a relação oferta/demanda, então *O = D*r*. Seja *n* o número de centros de oferta, para a geração de *v*, utilizaremos basicamente 2 passos:

1. **Inicialização:** Inicialmente $v_i = 100/n$, ou seja, será distribuído igualmente o percentual da oferta total para cada centro de oferta.
2. **Geração aleatória das ofertas:** a idéia básica é fazer um processo iterativo, em que, a cada passo, escolhem-se dois centros de oferta *p* e *k*, sendo *p* para incremento de v_p e *k* para decremento de v_k. Como se deve manter os 100% da oferta total, o valor do incremento e do decremento devem ser iguais. O valor do incremento e decremento, em termos absolutos será chamado de *i_d*. Para encontrar o valor de *i_d* será utilizado um escalar *f*, o qual é denominado de fator de *i_d*. Assim, utilizaremos a seguinte fórmula *i_d = (100/n)/f* para encontrarmos *i_d*. É importante observar que v_i não pode ser negativo. Assim sendo, será incluída no método a seguinte condição: toda vez que $v_k < i\_d$, será recalculado *i_d* fazendo com que o novo *i_d* seja menor do que o

anterior, eliminando, assim, um valor mínimo para v_i. Para isso, será utilizado um novo valor de *f*, o qual será calculado da seguinte forma: *f = f*1.1*. A parada do processo de geração aleatória ocorrerá, quando forem realizadas *3xn* trocas.

6.5.1.1.3 Geração dos dados de custo de transporte

Os custos de transporte serão gerados aleatoriamente entre 0 e o valor máximo de custo de transporte, o qual será especificado pelo usuário.

6.5.1.1.4 Gravação dos dados

Os dados do problema gerado serão armazenados em um arquivo *ascii*, conforme especificado abaixo, onde *n* é o número de centros de oferta, *m* é o número de centros de demanda, b_j é a demanda do centro de demanda *j*, a_i é a oferta do centro de oferta *i* e c_{ij} é o custo de transporte, de uma unidade, do centro de oferta *i* para o centro de demanda *j*.

Formato do arquivo:

n m
b_1
b_2
.
.
.
b_m
a_1
a_2
.
.
.
a_n
c_{11} c_{12} ... c_{1m}
c_{21} c_{22} ... c_{2m}
.
.
.
c_{n1} c_{n2} ... c_{nm}

6.5.2 Geração de dados dos problemas teste

Para que o método desenvolvido seja validado, é necessária a geração de problemas teste com diversas características. Os problemas foram gerados, utilizando-se o gerador definido na seção 6.5.1, diferenciando-se através das seguintes características:

5. Diferentes relações entre os custos de transporte. Desde uma pequena a uma grande diferença entre eles. As diferenças são de no máximo 8, 100, 1000 e 10000.
6. Diferentes dimensões de problemas. As dimensões utilizadas são 100x100, 100x1000, 300x1000, 500x1000, 1000x1000 e 1000x5000, onde o primeiro número representa a quantidade de centros de ofertas e, o segundo, a quantidade de centros de demanda.
7. Diferentes relações entre a quantidade de demanda e a quantidade de oferta. As relações consideradas são $\frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j}$ = *1, 1.5, 3, 5 e 10.*

Para cada combinação das características citadas, foram gerados três problemas diferentes. Para facilitar a compreensão, o nome do problema terá o seguinte formato: pt_r_IxJ_c_30-n.dat, onde:

r = relações entre a quantidade de demanda e a quantidade de oferta (1, 1.5, 3, 5, 10).

I = número de centros de oferta (100, 300, 500 e 1000).

J = nu mero de centros de demanda (100, 1000 e 5000).

c = diferença máxima entre os custos de transporte (8,100,1000 e 10000).

n = número do problema (1,2 e 3).

6.5.3 Resultados

O algoritmo para resolução do *PTC* foi implementado em C++ sob a sistema operacional Linux Conectiva 8. O compilador utilizado foi o g++ versão 2.95.3. Para comparação dos resultados foi utilizado o *software* comercial XpressMP versão 14.05 para *Windows*. Embora os métodos rodem em sistemas operacionais diferentes eles foram rodados em uma mesma máquina, onde foi aplicado o recurso de dual boot, permitindo com isso que a comparação seja real. A máquina utilizada para obtenção dos resultados foi uma máquina Pentium III, 700 mhz com 256 Megabytes de memória principal. Vale observar que é uma máquina defasada para os dias de hoje.

A Tabela 69 mostra os resultados computacionais obtidos, exceto para os problemas com dimensões de $|I| = 1000$ e $|J| = 5000$. Estes problemas não estão na Tabela 70 pelo fato de que o *software* Xpress utilizado travava na execução dos mesmos. Assim sendo os problemas com dimensões de $|I| = 1000$ e $|J| = 5000$ são apresentados na Tabela 70, sem comparação com o Xpress.

A primeira coluna, coluna Problema, da Tabela 69 representa o nome do problema teste, onde no próprio nome é possível identificar as suas características, conforme apresentado na seção 6.5.2. A segunda coluna, coluna MPTC, apresenta o tempo, em segundos, necessário para que o método desenvolvido obtenha a solução ótima do problema. A terceira coluna, coluna Xpress, apresenta o tempo, em segundos, necessário para que o Xpress obtenha a solução ótima do problema. A quarta coluna, coluna Xpress/MPTC, apresenta a quantidade de vezes que o método desenvolvido foi mais rápido do que o Xpress. Isto é, seja s1 o tempo de resolução do método desenvolvido e s2 o tempo de resolução do Xpress, então a quarta coluna é formada por s2/s1.

Problema	MPTC Seg.	Xpress Seg.	Xpress/ MPTC
pt_1_100x100_8_30-1.dat	0,01	0,4	40,00
pt_1_100x100_8_30-2.dat	0,01	0,3	30,00
pt_1_100x100_8_30-3.dat	0,02	0,3	15,00
pt_1_100x1000_8_30-1.dat	0,42	5,4	12,86
pt_1_100x1000_8_30-2.dat	0,36	5,8	16,11
pt_1_100x1000_8_30-3.dat	0,41	5,3	12,93
pt_1_300x1000_8_30-1.dat	0,72	25,1	34,86
pt_1_300x1000_8_30-2.dat	0,71	26,5	37,32
pt_1_300x1000_8_30-3.dat	0,7	22,9	32,71
pt_1_500x1000_8_30-1.dat	1,03	55,2	53,59
pt_1_500x1000_8_30-2.dat	1,02	46,1	45,20
pt_1_500x1000_8_30-3.dat	1,18	58,5	49,58
pt_1_1000x1000_8_30-1.dat	2,17	150	69,12
pt_1_1000x1000_8_30-2.dat	2,11	149,5	70,85
pt_1_1000x1000_8_30-3.dat	2,46	157,3	63,94
pt_1,5_100x100_8_30-1.dat	0,01	0,2	20,00
pt_1,5_100x100_8_30-2.dat	0,01	0,2	20,00
pt_1,5_100x100_8_30-3.dat	0,01	0,2	20,00
pt_1,5_100x1000_8_30-1.dat	0,37	1,4	3,78
pt_1,5_100x1000_8_30-2.dat	0,41	1,3	3,17
pt_1,5_100x1000_8_30-3.dat	0,41	1,4	3,41
pt_1,5_300x1000_8_30-1.dat	0,67	4,9	7,31
pt_1,5_300x1000_8_30-2.dat	0,64	5,1	7,97

pt_1,5_300x1000_8_30-3.dat	0,62	5,8	9,35
pt_1,5_500x1000_8_30-1.dat	0,83	13,9	16,75
pt_1,5_500x1000_8_30-2.dat	0,87	12,1	13,91
pt_1,5_500x1000_8_30-3.dat	0,86	12,3	14,30
pt_1,5_1000x1000_8_30-1.dat	1,58	46	29,11
pt_1,5_1000x1000_8_30-2.dat	1,57	55,9	35,61
pt_1,5_1000x1000_8_30-3.dat	1,65	45,7	27,70
pt_3_100x100_8_30-1.dat	0,01	0,1	10,00
pt_3_100x100_8_30-2.dat	0,004	0,1	25,00
pt_3_100x100_8_30-3.dat	0,01	0,1	10,00
pt_3_100x1000_8_30-1.dat	0,46	0,9	1,96
pt_3_100x1000_8_30-2.dat	0,49	0,9	1,84
pt_3_100x1000_8_30-3.dat	0,47	0,9	1,91
pt_3_300x1000_8_30-1.dat	0,67	2,9	4,33
pt_3_300x1000_8_30-2.dat	0,69	2,9	4,20
pt_3_300x1000_8_30-3.dat	0,69	3	4,35
pt_3_500x1000_8_30-1.dat	0,89	5,6	6,29
pt_3_500x1000_8_30-2.dat	0,86	4,9	5,70
pt_3_500x1000_8_30-3.dat	0,96	5,2	5,42
pt_3_1000x1000_8_30-1.dat	1,61	37,6	23,35
pt_3_1000x1000_8_30-2.dat	1,68	33,1	19,70
pt_3_1000x1000_8_30-3.dat	1,38	38,9	28,19
pt_5_100x100_8_30-1.dat	0,01	0,1	10,00
pt_5_100x100_8_30-2.dat	0,004	0,1	25,00
pt_5_100x100_8_30-3.dat	0,01	0,2	20,00
pt_5_100x1000_8_30-1.dat	0,67	0,8	1,19
pt_5_100x1000_8_30-2.dat	0,72	0,9	1,25
pt_5_100x1000_8_30-3.dat	0,68	0,8	1,18
pt_5_300x1000_8_30-1.dat	0,77	2,9	3,77
pt_5_300x1000_8_30-2.dat	0,75	2,5	3,33
pt_5_300x1000_8_30-3.dat	0,77	2,5	3,25
pt_5_500x1000_8_30-1.dat	0,87	4,9	5,63
pt_5_500x1000_8_30-2.dat	0,91	5,5	6,04
pt_5_500x1000_8_30-3.dat	1,11	4,9	4,41
pt_5_1000x1000_8_30-1.dat	1,32	31,5	23,86
pt_5_1000x1000_8_30-2.dat	1,29	32	24,81
pt_5_1000x1000_8_30-3.dat	1,55	44,8	28,90
pt_10_100x100_8_30-1.dat	0,01	0,1	10,00
pt_10_100x100_8_30-2.dat	0,01	0,1	10,00
pt_10_100x100_8_30-3.dat	0,01	0,1	10,00
pt_10_100x1000_8_30-1.dat	0,78	0,8	1,03
pt_10_100x1000_8_30-2.dat	0,82	0,9	1,10
pt_10_100x1000_8_30-3.dat	0,77	0,9	1,17
pt_10_300x1000_8_30-1.dat	1,32	2,5	1,89
pt_10_300x1000_8_30-2.dat	1,18	2,6	2,20
pt_10_300x1000_8_30-3.dat	1,1	2,5	2,27
pt_10_500x1000_8_30-1.dat	1,35	4,9	3,63
pt_10_500x1000_8_30-2.dat	1,24	5,2	4,19
pt_10_500x1000_8_30-3.dat	1,27	5,7	4,49
pt_10_1000x1000_8_30-1.dat	2,04	48,9	23,97
pt_10_1000x1000_8_30-2.dat	1,82	31,3	17,20

pt_10_1000x1000_8_30-3.dat	1,63	32,3	19,82
pt_1_100x100_100_30-1.dat	0,03	0,3	10,00
pt_1_100x100_100_30-2.dat	0,03	0,3	10,00
pt_1_100x100_100_30-3.dat	0,03	0,3	10,00
pt_1_100x1000_100_30-1.dat	0,88	12,8	14,55
pt_1_100x1000_100_30-2.dat	0,89	10,5	11,80
pt_1_100x1000_100_30-3.dat	0,86	8,8	10,23
pt_1_300x1000_100_30-1.dat	1,84	26,2	14,24
pt_1_300x1000_100_30-2.dat	1,87	28,2	15,08
pt_1_300x1000_100_30-3.dat	2,04	27,4	13,43
pt_1_500x1000_100_30-1.dat	2,95	64	21,69
pt_1_500x1000_100_30-2.dat	2,93	61,3	20,92
pt_1_500x1000_100_30-3.dat	3,04	45,8	15,07
pt_1_1000x1000_100_30-1.dat	5,35	158,5	29,63
pt_1_1000x1000_100_30-2.dat	5,76	142,3	24,70
pt_1_1000x1000_100_30-3.dat	5,27	163,8	31,08
pt_1,5_100x100_100_30-1.dat	0,02	0,3	15,00
pt_1,5_100x100_100_30-2.dat	0,02	0,1	5,00
pt_1,5_100x100_100_30-3.dat	0,01	0,1	10,00
pt_1,5_100x1000_100_30-1.dat	0,65	1,4	2,15
pt_1,5_100x1000_100_30-2.dat	0,61	1,4	2,30
pt_1,5_100x1000_100_30-3.dat	0,65	1,2	1,85
pt_1,5_300x1000_100_30-1.dat	1,36	8,4	6,18
pt_1,5_300x1000_100_30-2.dat	1,2	8,1	6,75
pt_1,5_300x1000_100_30-3.dat	1,18	12,1	10,25
pt_1,5_500x1000_100_30-1.dat	2,05	17,2	8,39
pt_1,5_500x1000_100_30-2.dat	1,94	15,2	7,84
pt_1,5_500x1000_100_30-3.dat	2,01	12,6	6,27
pt_1,5_1000x1000_100_30-1.dat	3,33	56,5	16,97
pt_1,5_1000x1000_100_30-2.dat	3,59	50,4	14,04
pt_1,5_1000x1000_100_30-3.dat	3,66	53,5	14,62
pt_3_100x100_100_30-1.dat	0,02	0,1	5,00
pt_3_100x100_100_30-2.dat	0,01	0,1	10,00
pt_3_100x100_100_30-3.dat	0,01	0,1	10,00
pt_3_100x1000_100_30-1.dat	0,64	0,8	1,25
pt_3_100x1000_100_30-2.dat	0,55	0,7	1,27
pt_3_100x1000_100_30-3.dat	0,49	0,7	1,43
pt_3_300x1000_100_30-1.dat	1,06	2,8	2,64
pt_3_300x1000_100_30-2.dat	1,15	2,9	2,52
pt_3_300x1000_100_30-3.dat	1,09	3	2,75
pt_3_500x1000_100_30-1.dat	1,7	12,7	7,47
pt_3_500x1000_100_30-2.dat	1,64	9,5	5,79
pt_3_500x1000_100_30-3.dat	1,51	5,8	3,84
pt_3_1000x1000_100_30-1.dat	3,15	32,2	10,22
pt_3_1000x1000_100_30-2.dat	2,83	32,7	11,55
pt_3_1000x1000_100_30-3.dat	2,84	31,7	11,16
pt_5_100x100_100_30-1.dat	0,01	0,1	10,00
pt_5_100x100_100_30-2.dat	0,01	0,1	10,00
pt_5_100x100_100_30-3.dat	0,004	0,1	25,00
pt_5_100x1000_100_30-1.dat	0,54	0,7	1,30
pt_5_100x1000_100_30-2.dat	0,45	0,7	1,56

pt_5_100x1000_100_30-3.dat	0,54	0,7	1,30
pt_5_300x1000_100_30-1.dat	1,06	2,4	2,26
pt_5_300x1000_100_30-2.dat	0,98	2,2	2,24
pt_5_300x1000_100_30-3.dat	0,96	2,3	2,40
pt_5_500x1000_100_30-1.dat	1,49	11,4	7,65
pt_5_500x1000_100_30-2.dat	1,51	4,9	3,25
pt_5_500x1000_100_30-3.dat	1,71	7,3	4,27
pt_5_1000x1000_100_30-1.dat	2,99	38,9	13,01
pt_5_1000x1000_100_30-2.dat	3	33,4	11,13
pt_5_1000x1000_100_30-3.dat	2,85	37,2	13,05
pt_10_100x100_100_30-1.dat	0,01	0,1	10,00
pt_10_100x100_100_30-2.dat	0,004	0,1	25,00
pt_10_100x100_100_30-3.dat	0,01	0,1	10,00
pt_10_100x1000_100_30-1.dat	0,46	0,6	1,30
pt_10_100x1000_100_30-2.dat	0,4	0,7	1,75
pt_10_100x1000_100_30-3.dat	0,41	0,7	1,71
pt_10_300x1000_100_30-1.dat	1,16	1,9	1,64
pt_10_300x1000_100_30-2.dat	1,23	1,9	1,54
pt_10_300x1000_100_30-3.dat	1,11	1,9	1,71
pt_10_500x1000_100_30-1.dat	1,92	13	6,77
pt_10_500x1000_100_30-2.dat	1,38	10,9	7,90
pt_10_500x1000_100_30-3.dat	1,42	5	3,52
pt_10_1000x1000_100_30-1.dat	2,58	35,9	13,91
pt_10_1000x1000_100_30-2.dat	2,77	34,3	12,38
pt_10_1000x1000_100_30-3.dat	2,71	33,3	12,29
pt_1_100x100_1000_30-1.dat	0,03	0,2	6,67
pt_1_100x100_1000_30-2.dat	0,03	0,2	6,67
pt_1_100x100_1000_30-3.dat	0,03	0,2	6,67
pt_1_100x1000_1000_30-1.dat	0,9	4,4	4,89
pt_1_100x1000_1000_30-2.dat	0,96	4,6	4,79
pt_1_100x1000_1000_30-3.dat	0,89	3,9	4,38
pt_1_300x1000_1000_30-1.dat	2,21	26,5	11,99
pt_1_300x1000_1000_30-2.dat	2,21	22,6	10,23
pt_1_300x1000_1000_30-3.dat	2,29	23,6	10,31
pt_1_500x1000_1000_30-1.dat	3,85	57,3	14,88
pt_1_500x1000_1000_30-2.dat	3,73	63,9	17,13
pt_1_500x1000_1000_30-3.dat	3,63	53,1	14,63
pt_1_1000x1000_1000_30-1.dat	9,27	258,5	27,89
pt_1_1000x1000_1000_30-2.dat	9,4	272,3	28,97
pt_1_1000x1000_1000_30-3.dat	8,02	240,6	30,00
pt_1.5_100x100_1000_30-1.dat	0,01	0,1	10,00
pt_1.5_100x100_1000_30-2.dat	0,02	0,1	5,00
pt_1.5_100x100_1000_30-3.dat	0,01	0,2	20,00
pt_1.5_100x1000_1000_30-1.dat	0,61	1	1,64
pt_1.5_100x1000_1000_30-2.dat	0,62	1	1,61
pt_1.5_100x1000_1000_30-3.dat	0,65	1,3	2,00
pt_1.5_300x1000_1000_30-1.dat	1,36	3,7	2,72
pt_1.5_300x1000_1000_30-2.dat	1,31	3	2,29
pt_1.5_300x1000_1000_30-3.dat	1,4	3	2,14
pt_1.5_500x1000_1000_30-1.dat	1,89	7,5	3,97
pt_1.5_500x1000_1000_30-2.dat	1,85	8,2	4,43

pt_1.5_500x1000_1000_30-3.dat	1,85	8,6	4,65
pt_1.5_1000x1000_1000_30-1.dat	10,35	48,5	4,69
pt_1.5_1000x1000_1000_30-2.dat	8,27	40,3	4,87
pt_1.5_1000x1000_1000_30-3.dat	11,44	48,2	4,21
pt_3_100x100_1000_30-1.dat	0,01	0,1	10,00
pt_3_100x100_1000_30-2.dat	0,01	0,1	10,00
pt_3_100x100_1000_30-3.dat	0,01	0,1	10,00
pt_3_100x1000_1000_30-1.dat	0,57	0,8	1,40
pt_3_100x1000_1000_30-2.dat	0,54	0,8	1,48
pt_3_100x1000_1000_30-3.dat	0,55	0,7	1,27
pt_3_300x1000_1000_30-1.dat	1	2,2	2,20
pt_3_300x1000_1000_30-2.dat	1,05	2,1	2,00
pt_3_300x1000_1000_30-3.dat	1,58	2,1	1,33
pt_3_500x1000_1000_30-1.dat	2,1	8,9	4,24
pt_3_500x1000_1000_30-2.dat	1,5	9,4	6,27
pt_3_500x1000_1000_30-3.dat	1,49	4,8	3,22
pt_3_1000x1000_1000_30-1.dat	2,69	29,7	11,04
pt_3_1000x1000_1000_30-2.dat	2,88	30,3	10,52
pt_3_1000x1000_1000_30-3.dat	2,79	34,6	12,40
pt_5_100x100_1000_30-1.dat	0,01	0,1	10,00
pt_5_100x100_1000_30-2.dat	0,004	0,1	25,00
pt_5_100x100_1000_30-3.dat	0,01	0,1	10,00
pt_5_100x1000_1000_30-1.dat	0,46	0,7	1,52
pt_5_100x1000_1000_30-2.dat	0,56	0,7	1,25
pt_5_100x1000_1000_30-3.dat	0,51	0,9	1,76
pt_5_300x1000_1000_30-1.dat	1,16	1,9	1,64
pt_5_300x1000_1000_30-2.dat	0,95	1,9	2,00
pt_5_300x1000_1000_30-3.dat	0,95	2	2,11
pt_5_500x1000_1000_30-1.dat	1,43	4,3	3,01
pt_5_500x1000_1000_30-2.dat	1,22	7,2	5,90
pt_5_500x1000_1000_30-3.dat	1,59	4,3	2,70
pt_5_1000x1000_1000_30-1.dat	2,69	39,5	14,68
pt_5_1000x1000_1000_30-2.dat	2,16	45,3	20,97
pt_5_1000x1000_1000_30-3.dat	2,55	27,4	10,75
pt_10_100x100_1000_30-1.dat	0,01	0,1	10,00
pt_10_100x100_1000_30-2.dat	0,004	0,1	25,00
pt_10_100x100_1000_30-3.dat	0,004	0,1	25,00
pt_10_100x1000_1000_30-1.dat	0,42	0,7	1,67
pt_10_100x1000_1000_30-2.dat	0,37	0,6	1,62
pt_10_100x1000_1000_30-3.dat	0,31	0,6	1,94
pt_10_300x1000_1000_30-1.dat	0,74	2	2,70
pt_10_300x1000_1000_30-2.dat	0,93	2	2,15
pt_10_300x1000_1000_30-3.dat	0,94	1,9	2,02
pt_10_500x1000_1000_30-1.dat	1,21	3,7	3,06
pt_10_500x1000_1000_30-2.dat	1,38	3,3	2,39
pt_10_500x1000_1000_30-3.dat	1,16	3,2	2,76
pt_10_1000x1000_1000_30-1.dat	2,23	35,1	15,74
pt_10_1000x1000_1000_30-2.dat	2,21	30,1	13,62
pt_10_1000x1000_1000_30-3.dat	2,23	29	13,00
pt_1_100x100_10000_30-1.dat	0,03	0,2	6,67
pt_1_100x100_10000_30-2.dat	0,03	0,2	6,67

pt_1_100x100_10000_30-3.dat	0,03	0,2	6,67
pt_1_100x1000_10000_30-1.dat	0,95	4,9	5,16
pt_1_100x1000_10000_30-2.dat	0,93	5,1	5,48
pt_1_100x1000_10000_30-3.dat	0,93	5	5,38
pt_1_300x1000_10000_30-1.dat	2,32	22,6	9,74
pt_1_300x1000_10000_30-2.dat	2,23	18,3	8,21
pt_1_300x1000_10000_30-3.dat	2,25	21,2	9,42
pt_1_500x1000_10000_30-1.dat	3,76	41,5	11,04
pt_1_500x1000_10000_30-2.dat	3,63	51,1	14,08
pt_1_500x1000_10000_30-3.dat	3,67	40,5	11,04
pt_1_1000x1000_10000_30-1.dat	7,98	130,8	16,39
pt_1_1000x1000_10000_30-2.dat	8,15	141	17,30
pt_1_1000x1000_10000_30-3.dat	8,07	180,5	22,37
pt_1.5_100x100_10000_30-1.dat	0,02	0,1	5,00
pt_1.5_100x100_10000_30-2.dat	0,02	0,1	5,00
pt_1.5_100x100_10000_30-3.dat	0,02	0,1	5,00
pt_1.5_100x1000_10000_30-1.dat	0,63	1,1	1,75
pt_1.5_100x1000_10000_30-2.dat	0,64	1,1	1,72
pt_1.5_100x1000_10000_30-3.dat	0,64	1	1,56
pt_1.5_300x1000_10000_30-1.dat	1,25	3,8	3,04
pt_1.5_300x1000_10000_30-2.dat	1,3	3,2	2,46
pt_1.5_300x1000_10000_30-3.dat	1,2	3,7	3,08
pt_1.5_500x1000_10000_30-1.dat	1,86	14,6	7,85
pt_1.5_500x1000_10000_30-2.dat	1,78	15,5	8,71
pt_1.5_500x1000_10000_30-3.dat	2,03	19,3	9,51
pt_1.5_1000x1000_10000_30-1.dat	3,78	39,4	10,42
pt_1.5_1000x1000_10000_30-2.dat	3,71	37,7	10,16
pt_1.5_1000x1000_10000_30-3.dat	3,8	37,8	9,95
pt_3_100x100_10000_30-1.dat	0,02	0,1	5,00
pt_3_100x100_10000_30-2.dat	0,01	0,1	10,00
pt_3_100x100_10000_30-3.dat	0,004	0,1	25,00
pt_3_100x1000_10000_30-1.dat	0,52	0,7	1,35
pt_3_100x1000_10000_30-2.dat	0,54	0,8	1,48
pt_3_100x1000_10000_30-3.dat	0,48	0,8	1,67
pt_3_300x1000_10000_30-1.dat	1,07	2,2	2,06
pt_3_300x1000_10000_30-2.dat	1,05	2,2	2,10
pt_3_300x1000_10000_30-3.dat	1,01	2,3	2,28
pt_3_500x1000_10000_30-1.dat	1,51	11,4	7,55
pt_3_500x1000_10000_30-2.dat	1,4	5,9	4,21
pt_3_500x1000_10000_30-3.dat	1,46	5,8	3,97
pt_3_1000x1000_10000_30-1.dat	2,5	29,1	11,64
pt_3_1000x1000_10000_30-2.dat	2,69	37	13,75
pt_3_1000x1000_10000_30-3.dat	2,52	28,2	11,19
pt_5_100x100_10000_30-1.dat	0,01	0,1	10,00
pt_5_100x100_10000_30-2.dat	0,01	0,1	10,00
pt_5_100x100_10000_30-3.dat	0,004	0,1	25,00
pt_5_100x1000_10000_30-1.dat	0,46	0,7	1,52
pt_5_100x1000_10000_30-2.dat	0,42	0,7	1,67
pt_5_100x1000_10000_30-3.dat	0,51	0,7	1,37
pt_5_300x1000_10000_30-1.dat	0,94	2	2,13
pt_5_300x1000_10000_30-2.dat	0,86	2	2,33

pt_5_300x1000_10000_30-3.dat	0,96	1,9	1,98
pt_5_500x1000_10000_30-1.dat	1,37	3,3	2,41
pt_5_500x1000_10000_30-2.dat	1,33	4,1	3,08
pt_5_500x1000_10000_30-3.dat	1,33	8,9	6,69
pt_5_1000x1000_10000_30-1.dat	1,98	26,7	13,48
pt_5_1000x1000_10000_30-2.dat	2,47	30,2	12,23
pt_5_1000x1000_10000_30-3.dat	2,33	26,7	11,46
pt_10_100x100_10000_30-1.dat	0,01	0,1	10,00
pt_10_100x100_10000_30-2.dat	0,01	0,1	10,00
pt_10_100x100_10000_30-3.dat	0,01	0,1	10,00
pt_10_100x1000_10000_30-1.dat	0,3	0,6	2,00
pt_10_100x1000_10000_30-2.dat	0,37	0,7	1,89
pt_10_100x1000_10000_30-3.dat	0,33	0,7	2,12
pt_10_300x1000_10000_30-1.dat	0,74	1,9	2,57
pt_10_300x1000_10000_30-2.dat	0,88	2,1	2,39
pt_10_300x1000_10000_30-3.dat	0,91	2	2,20
pt_10_500x1000_10000_30-1.dat	1,26	11,2	8,89
pt_10_500x1000_10000_30-2.dat	1,23	6,2	5,04
pt_10_500x1000_10000_30-3.dat	1,27	6	4,72
pt_10_1000x1000_10000_30-1.dat	2,19	27	12,33
pt_10_1000x1000_10000_30-2.dat	2,18	27	12,39
pt_10_1000x1000_10000_30-3.dat	2,19	25,2	11,51
Média	***1,41***	***18,34***	***12,99***

Tabela 69 – resultados computacionais para o método proposto e Xpress

A primeira coluna, coluna Problema, da Tabela 70 representa o nome do problema teste, onde no próprio nome é possível identificar as suas características, conforme apresentado na seção 6.5.2. A segunda coluna, coluna MPTC, apresenta o tempo, em segundos, necessário para que o método desenvolvido obtenha a solução ótima do problema. Como dito anteriormente, não foi possível obter as soluções dos problemas da Tabela 70 através da versão utilizada do Xpress. Para resolução dos problemas de dimensão $|I| = 1000$ e $|J| = 5000$ foram necessários 48Mb de memória.

Problema	MPTC Seg.
pt_1_1000x5000_8_30-1.dat	29,19
pt_1_1000x5000_8_30-2.dat	30,76
pt_1_1000x5000_8_30-3.dat	29,78
pt_1,5_1000x5000_8_30-1.dat	23,51
pt_1,5_1000x5000_8_30-2.dat	23,07
pt_1,5_1000x5000_8_30-3.dat	25,09
pt_3_1000x5000_8_30-1.dat	19,3
pt_3_1000x5000_8_30-2.dat	19,26
pt_3_1000x5000_8_30-3.dat	22,59
pt_5_1000x5000_8_30-1.dat	19,26
pt_5_1000x5000_8_30-2.dat	21,05
pt_5_1000x5000_8_30-3.dat	20,02
pt_10_1000x5000_8_30-1.dat	24,65
pt_10_1000x5000_8_30-2.dat	23,95
pt_10_1000x5000_8_30-3.dat	24,91
pt_1_1000x5000_100_30-1.dat	82,28
pt_1_1000x5000_100_30-2.dat	80,6
pt_1_1000x5000_100_30-3.dat	74,28
pt_1,5_1000x5000_100_30-1.dat	55,36
pt_1,5_1000x5000_100_30-2.dat	55,79
pt_1,5_1000x5000_100_30-3.dat	64,22
pt_3_1000x5000_100_30-1.dat	46,97
pt_3_1000x5000_100_30-2.dat	46,42
pt_3_1000x5000_100_30-3.dat	45,5
pt_5_1000x5000_100_30-1.dat	44,83
pt_5_1000x5000_100_30-2.dat	47,44
pt_5_1000x5000_100_30-3.dat	45,36
pt_10_1000x5000_100_30-1.dat	47,37
pt_10_1000x5000_100_30-2.dat	51,96
pt_10_1000x5000_100_30-3.dat	44,62
pt_1_1000x5000_1000_30-1.dat	139,94
pt_1_1000x5000_1000_30-2.dat	127,04
pt_1_1000x5000_1000_30-3.dat	134,51
pt_1.5_1000x5000_1000_30-1.dat	160,79
pt_1.5_1000x5000_1000_30-2.dat	68,26
pt_1.5_1000x5000_1000_30-3.dat	56,51
pt_3_1000x5000_1000_30-1.dat	54,78
pt_3_1000x5000_1000_30-2.dat	56,46
pt_3_1000x5000_1000_30-3.dat	52,36
pt_5_1000x5000_1000_30-1.dat	41,51
pt_5_1000x5000_1000_30-2.dat	34,91
pt_5_1000x5000_1000_30-3.dat	35,56
pt_10_1000x5000_1000_30-1.dat	30,36
pt_10_1000x5000_1000_30-2.dat	31,83
pt_10_1000x5000_1000_30-3.dat	34,55
Média	***49,97***

Tabela 70 – resultados computacionais para o método proposto

O Gráfico 1 mostra o tempo médio de resolução dos problemas com diferença máxima entre os custos de transporte de 8, 100, 1000 e 10000. Os problemas mais difíceis para serem resolvidos foram os problemas com diferença máxima entre os custos de transporte iguais a 1000, tanto para o MPTC quanto para o Xpress. O comportamento do gráfico mostra que o tempo de resolução vai aumentando conforme for sendo aumentado a diferença entes os c_{ij}, contudo ele aumenta até um determinado momento, depois volta a cair. É importante observar que a maior diferença entre o MPTC e o Xpress é nos problemas mais difíceis de serem resolvidos.

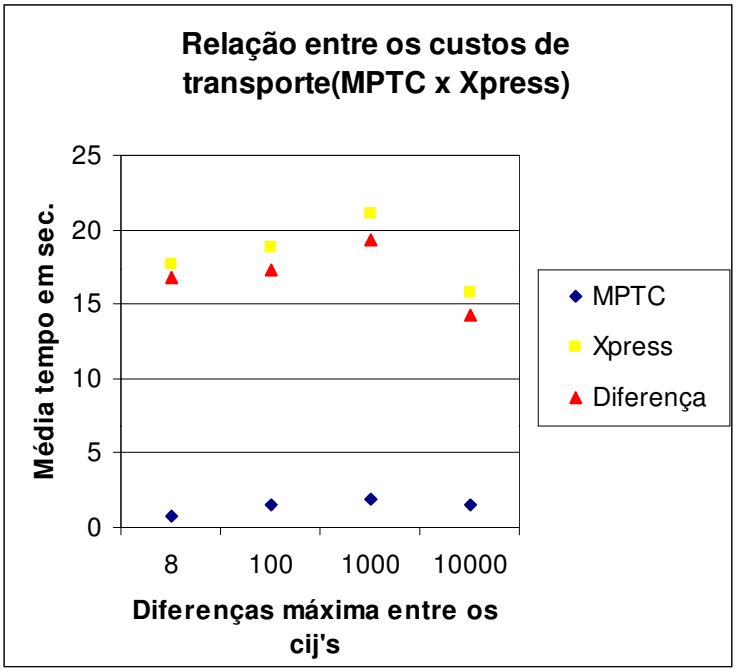


Gráfico 1 - Relação entre os custos de transporte(MPTC x Xpress)

O Gráfico 2 mostra o tempo médio de resolução dos problemas com diferentes relações entre a quantidade de demanda e a quantidade de oferta. As relações consideradas são: $\frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j} = 1, 1.5, 3, 5 \text{ e } 10$. Os problemas mais difíceis para serem resolvidos foram os problemas com $\frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j} = 1$, tanto para o MPTC quanto para o Xpress. O comportamento do gráfico mostra que o tempo de resolução vai reduzindo quando é aumentada a relação oferta/demanda, contudo ele praticamente estabiliza a

partir de $\frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j} = 3$. É importante observar que a maior diferença entre o MPTC e o Xpress são nos problemas mais difíceis de serem resolvidos.

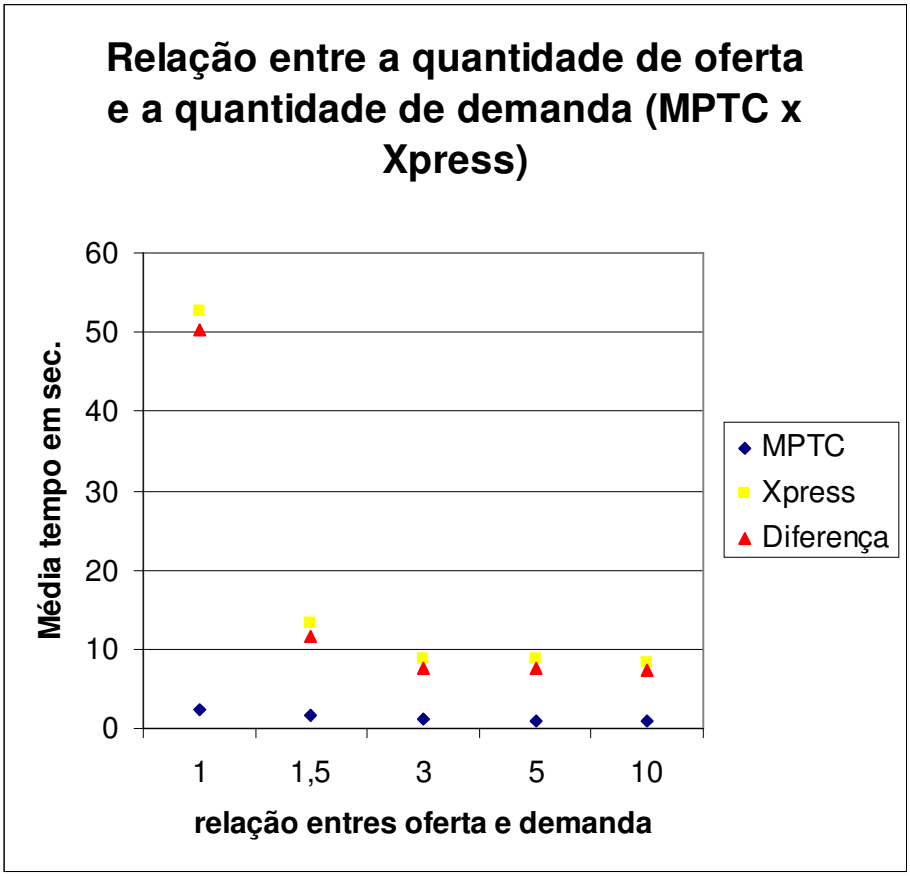


Gráfico 2 - Relação entre a quantidade de oferta e a quantidade de demanda (MPTC x Xpress)

O Gráfico 3 mostra a relação entre o tempo médio de resolução dos problemas e a quantidade de variáveis dos problemas. As quantidades de variáveis consideradas são: 10000, 100000, 300000, 500000, 1000000. Os problemas mais difíceis para serem resolvidos foram os problemas com as maiores dimensões, tanto para o MPTC quanto para o Xpress. O comportamento do gráfico mostra que o tempo de resolução vai aumentando conforme for sendo aumentado as dimensões do problema. O Gráfico 3 pode mascarar alguns resultados, quais sejam, porque para problemas muito pequenos a diferença relativa entre o MPTC e o Xpress pode ser muito grande, ao contrário do que fica explicito no Gráfico 3, devido ao fato dos números lá serem absolutos. Isto faz

"sumir" as diferenças para valores muito pequenos. De qualquer forma, parece q ue a maior diferença entre o MPTC e o Xpress são nos problemas mais difíceis de serem resolvidos.

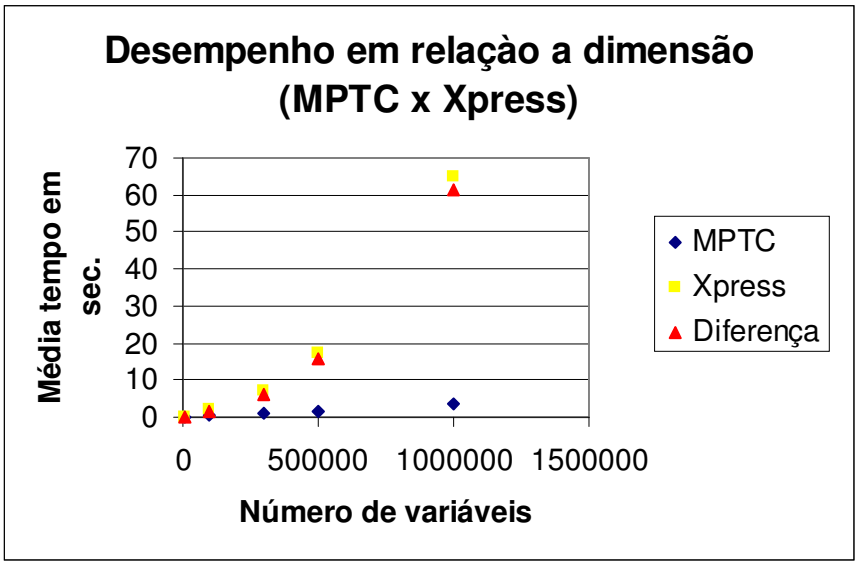


Gráfico 3 - Desempenho em relaçào a dimensão (MPTC x Xpress)

A Tabela 71 mostra os resultados computacionais quando é realizado a pós-otimização. Para realização dos testes foi seguido os seguintes passos:

1. Encontrar a solução do problema
2. Fechar 10 centros de ofertas, escolhidos aleatoriamente
3. Executar o algoritmo de pós-otimização
4. Abrir 5 centros de ofertas fechados no passo 2, escolhidos aleatoriamente
5. Executar o algoritmo de pós-otimização

A primeira coluna da Tabela 71, coluna Problema, representa o nome do problema teste, onde no próprio nome é possível identificar as suas características, conforme apresentado na seção 6.5.2. A segunda coluna, coluna **Fechar 10(seg.)**, apresenta o tempo, em segundos, necessário para que no passo 3 seja encontrada a solução ótima do problema após terem sido fechados 10 centros de oferta, no passo 2. A terceira coluna, coluna **abrir 5(seg**.), apresenta o tempo, em segundos, necessário para que no passo 5 seja encontrada a solução ótima do problema após terem sido abertos 5 centros de oferta, no passo 4. Vale observar que não foi possível utilizar a relação

$\frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j} = 1$, pois caso fosse fechado algum centro de oferta o problemas se tornaria inviável.

Problema	Fechar 10(seg.)	Abrir 5(seg.)
pt_1.5_1000x1000_1000_30-1.dat	0,27	0,10
pt_1.5_1000x1000_1000_30-2.dat	0,35	0,13
pt_1.5_1000x1000_1000_30-3.dat	0,50	0,16
pt_3_1000x1000_10000_30-1.dat	0,13	0,10
pt_3_1000x1000_10000_30-2.dat	0,37	0,06
pt_3_1000x1000_10000_30-3.dat	0,30	0,16
Média	**0,32**	**0,12**

Tabela 71 - resultados computacionais de pós otimização

6.6 Conclusão

O método desenvolvido para resolução do problema de transporte foi baseado no método do Jacobsen [29]. É utilizado o método simplex para resolução do problema. A solução básica é representada através de uma árvore geradora. Para representar a árvore foi utilizada a estrutura *threaded index*, apresentado em Jacobsen [29]. Essa estrutura tem a vantagem de utilizar alocação seqüencial de memória para se representar a árvore, possibilitando, como isso, a utilização de aritmética de ponteiros para se manipular a árvore, trazendo grandes vantagens em termos de eficiência computacional. Para o cálculo do custo reduzido foram incorporadas as idéias apresentadas em Ahrens e Finke [2].

Além da incorporação das idéias apresentadas em Ahrens e Finke [2], foi introduzida no método de Jacobsen [29] a altura de cada nó da árvore, permitindo, com isso, que se detecte o ciclo formado pela inclusão do arco de entrada na árvore mais rapidamente. É através da altura que é detectado se o centro de oferta está sendo considerado ou não no problema, facilitando a adaptação realizada para retirar ou inserir um centro de oferta. Isso é necessário para o desenvolvimento do método de resolução do *PLC* que será apresentado no Capítulo 4. Outra adaptação realizada foi a inclusão do centro de demanda virtual que recebe as ofertas não enviadas para os centros de demanda. A contribuição mais importante foi a inclusão da lista de prioridade para facilitar a busca pela variável a entrar na base. A inclusão da lista foi realizada

porque foi observado que o número de restrições é muito menor do que o número de variáveis, fazendo com que se tenha um número de variáveis muito menor na base do que fora dela.

As comparações realizadas entre o método desenvolvido e o Xpress mostraram que o método desenvolvido superou o Xpress em todos os testes realizados. Em média o método foi quase 13 vezes mais rápido, no máximo chegou a ser 70,85 vezes mais rápido, conforme linha em preto na Tabela 69, e no mínimo foi 1,03 vezes mais rápido, conforme linha em cinza da Tabela 69. Provavelmente o fato mais importante das comparações realizadas é que as maiores diferenças, entre o MPTC e o Xpress, foram, em sua maioria, nos problemas mais "difíceis".

Os resultados do algoritmo de pós-otimização desenvolvido também foram bastante satisfatórios. Isso permite que o *PTC* possa ser aplicado, de forma eficiente, no algoritmo *branch and bound*, desenvolvido no Capítulo 4. Isso porque, no *branch and bound* ora é aberta ora é fechada uma facilidade, a qual corresponde a um centro de oferta no *PTC*. O *PTC* é aplicado nos testes de redução, apresentados no Capítulo 2, e no limite inferior, apresentado no Capítulo 3.

O método desenvolvido demonstrou ser robusto, rodando problemas de um milhão e quinhentos mil variáveis, em um computados defasado, muito rapidamente. A única limitação de memória do método é a suportada pelo sistema operacional. Isto é, o método utilizará, caso necessário, todo a memória principal e virtual disponibilizada pelo sistema operacional. Caso haja algum problema em relação à memória basta aumentá-la no sistema operacional, seja fisicamente, através do acréscimo de mais *chip(s)* de memória, seja logicamente, aumentando a memória virtual.

Anexo 2

7 Métodos de agrupamento

Os métodos que serão apresentados são todos heurísticos, ou seja, métodos cuja solução gerada pode não ser ótima. Deve-se ressaltar que não se está interessado em fazer um algoritmo que seja alguns segundos ou milésimos de segundo melhor, em tempo de execução, do que os algoritmos encontrados na literatura. O objetivo é encontrar uma solução com qualidade em um tempo viável.

Deseja-se resolver o *PLC* dividindo-o em problemas menores, ou seja, em subproblemas. Para alcançar esse objetivo, serão formados grupos compostos de um subconjunto de facilidades e clientes do problema principal.

Os algoritmos que serão apresentados são compostos por três partes. A primeira consiste na formação de grupos. Na segunda, a partir dos grupos formados, resolve-se o Problema de Localização Capacitado de cada grupo. A terceira parte é referente ao refinamento da solução gerada pela segunda parte. Antes de ser explicado como será a formação dos grupos e o processo de geração da solução, serão definidos alguns conceitos.

7.1 Definições

Inicialmente será considerado que cada facilidade $i \in I$ tem uma **distância** para cada cliente $j \in J$, a qual será denominada d_{ij}. É importante observar que a distância d_{ij} não é idêntica ao custo variável c_{ij}. Posteriormente será visto que se pode relacioná-las. Vale ressaltar que o termo distância consiste, de certa forma, de um abuso de linguagem. Isso porque, as distâncias definidas aqui, não atendem as propriedades da teoria de agrupamento.

Um **grupo** x será formado por um conjunto de facilidades, denominado I_x, e um conjunto de clientes, denominado J_x. Assim sendo, o grupo x está associado a um Problema de Localização Capacitado, que será denominado P_x. O problema P_x representa um **subproblema** de *PLC*. G representará um conjunto de grupos.

Define-se D_{xy} como sendo a **distância do grupo *x* para o grupo *y***. É importante observar que D_{yx} pode ser diferente de D_{xy}, pois D_{yx} representa a distância do grupo y para o grupo x.

A **distância entre os dois grupos** *x* e *y* será representada por Δ_{xy}, onde $\Delta_{xy} = \Delta_{yx}$.

Também se define F_{kx}, onde *k* é uma facilidade e *x* um grupo, como sendo a **distância da facilidade *k* ao grupo *x***.

Cada grupo terá uma **consistência**. Essa indicará uma medida de proximidade dos elementos do grupo. Denominaremos C_x a consistência do grupo *x*. Com isso, quanto menor a consistência do grupo, mais próximos são as facilidades dos clientes.

Define-se *V(PLC)* como sendo a solução ótima de *PLC* e $V(P_i)$ a solução ótima do subproblema $P_i \subseteq PLC$.

Deseja-se resolver o *PLC* dividindo-o em $P_1, P_2, P_3, \ldots, P_k$ subproblemas, onde $k \leq |I|$. De uma forma geral gostar-se-ia de resolver cada subproblema, $P_1, P_2, P_3, \ldots, P_k$, separadamente e obter a solução de *PLC* através da união das soluções dos subproblemas. Isto é, gostar-se-ia que $V(PLC) = V(P_1) + V(P_2) + \ldots + V(P_k)$. Como dito anteriormente cada subproblema P_i $i=1,\ldots,k$ está associado a um grupo *i*. Caso a solução de *PLC,* através da união das soluções dos subproblemas, seja ótima, diz-se que os grupos formados são **bem definidos**. Então a expressão bem definidos indica que é dispensável a união de grupos para obtenção da solução ótima de *PLC*. Assim, a união de grupos não alteraria a solução gerada, $V(P_1) + V(P_2) + V(P_3) + \ldots + V(P_k) = V(P_1 \cup P_2) + V(P_3) + \ldots + V(P_k) = V(P_1 \cup P_2 \cup P_3) + V(P_4) + \ldots + V(P_k) = \ldots = V(PLC)$. Será demonstrado esta propriedade para a união de P_1 e P_2, as outras uniões seguem o mesmo raciocínio.

Tese:

$V(P_1) + V(P_2) + V(P_3) + \ldots + V(P_k) = V(PLC) \Rightarrow V(P_1) + V(P_2) + V(P_3) + \ldots + V(P_k) = V(P_1 \cup P_2) + V(P_3) + \ldots + V(P_k)$.

Demonstração:

- Fato 1: $V(P_1)+ V(P_2) \bullet V(P_1 \cup P_2)$ sempre
- Fato 2: $V(P_1)+ V(P_2)+ V(P_3)+...+ V(P_k) \bullet V(PLC)$ sempre

Pela hipótese tem-se:

$V(P_1)+ V(P_2)+ V(P_3)+...+ V(P_k) = V(PLC)$

Utilizando o fato 1:

(i) $V(P_1)+ V(P_2)+ V(P_3)+...+ V(P_k) = V(PLC) \bullet V(P_1 \cup P_2)+ V(P_3) +...+ V(P_k)$

pode-se fazer:

$P_1 \cup P_2 = P'_1, \quad P_3 = P'_2, \quad . \ . \ ., \quad P_k = P'_{k-1}$

Utilizando o fato 2:

(ii) $V(P'_1)+ V(P'_2)+ V(P'_3)+...+ V(P'_{k-1}) \bullet V(PLC) \Rightarrow V(P_1 \cup P_2)+ V(P_3) +...+ V(P_k) \bullet V(PLC)$

De (i) e (ii) Obtém-se:

$V(P_1 \cup P_2)+ V(P_3) +...+ V(P_k) = V(PLC)$

,

Antes de exemplificar cada um desses conceitos será mostrado como pode ser realizado aos cálculos de alguns conceitos.

7.2 Cálculos

Os cálculos que serão apresentados são relativos a algumas metodologias de agrupamento que serão mostradas na seção 7.4.1. Contudo, não representam a totalidade de formas de obtermos a distância entre grupos. Outra formas serão mostradas quando necessário. O principal objetivo das fórmulas abaixo é exemplificar alguns conceitos vistos.

7.2.1 Distância de um grupo para outro

Seja x e y dois grupos distintos. D_{xy} e D_{yx} serão obtidos das seguintes formas:

$$D_{xy} = \frac{\sum_{i \in I_x} \sum_{j \in J_y} d_{ij}}{|I_x| * |J_y|}$$

$$D_{yx} = \frac{\sum_{i \in I_y} \sum_{j \in J_x} d_{ij}}{|I_y| * |J_x|}$$

Caso $J_y = \varnothing$ ou $I_x = \varnothing$ teremos $D_{xy} = \infty$, e se $J_x = \varnothing$ ou $I_y = \varnothing$ então $D_{yx} = \infty$.

7.2.2 Distância entre dois grupos

Seja *x* e *y* dois grupos distintos. O cálculo de Δ_{xy} será realizado da seguinte maneira:

$\Delta_{xy} = (D_{xy} + D_{yx}) / 2$ caso $D_{xy} \neq \infty$ e $D_{yx} \neq \infty$

$\Delta_{xy} = D_{xy}$ caso $D_{xy} \neq \infty$ e $D_{yx} = \infty$

$\Delta_{xy} = D_{yx}$ caso $D_{xy} = \infty$ e $D_{yx} \neq \infty$

$\Delta_{xy} = \infty$ caso $D_{xy} = \infty$ e $D_{yx} = \infty$

Note que diferentemente de D_{xy} tem-se que $\Delta_{yx} = \Delta_{yx}$.

7.2.3 Distância entre uma facilidade e um grupo

Seja *x* um grupo e *k* uma facilidade. Definiremos a distância entre *k* e *x* como:

$$F_{kx} = \frac{\sum_{j \in J_x} d_{kj}}{|J_x|}$$

7.2.4 Consistências

Seja *x* um grupo, então a consistências do grupo será dada por:

$$C_x = \frac{\sum_{i \in I_x} \sum_{j \in J_x} d_{ij}}{|I_x| * |J_x|}$$

7.3 Exemplificação

Seja o Problema de Localização Capacitado *P* abaixo:

Facilidades/ Clientes	**Cliente 1**	**Cliente 2**	**Cliente 3**	**Cliente 4**	**Cliente 5**	$\mathbf{f_i}$	a_i
Facilidade1	5	10	30	40	28	70	*20*
Facilidade2	25	30	6	7	40	70	*20*
Facilidade3	15	15	40	60	17	70	*20*
Facilidade4	37	29	20	25	8	70	*20*
b_j	*10*	*10*	*10*	*10*	*10*	-	-

Figura 42 – Dados do *PLC*

Onde a coluna f_i representa o custo fixo de instalação de cada facilidade, a coluna a_i representa a capacidade de cada facilidade, a linha b_j representa a demanda de cada cliente e o restante da matriz representa o custo de transporte, de uma unidade, da facilidade *i* para um cliente *j*.

Serão formados os seguintes grupos G_1 com $I_1 = \{1\}$ e $J_1\{1\}$, G_2 com $I_2 = \{2\}$ e $J_2 = \{3,4\}$, G_3 com $I_3=\{4\}$ e $J_3 = \{5\}$ e G_4 com $I_4=\{3\}$ e $J_4 = \{2\}$. Todos os grupos são Subproblemas de Localização Capacitado de *P* e todos são viáveis. Para exemplificar será tomado $d_{ij} = c_{ij}$.

A solução ótima do problema *P* é $V(P) = 570$, onde as facilidade *1, 2* e *4* são abertas. A solução de *P* considerando a solução do grupos é: $V(P_1) + V(P_2) + V(P_3) + V(P_4) = 120+200+150+220 = 690$, ou seja, os grupos não estão bem definidos. Segue o cálculo da distância entre um grupo e outro:

$D_{12} = (30+40)/2 = 35$

$D_{21} = 25/1 = 25$

$D_{13} = 28/1 = 28$

$D_{31} = 37/1 = 37$

$D_{14} = 10/1 = 10$

$D_{41} = 15/1 = 15$

$D_{23} = 40/1 = 40$

$D_{32} = (20+25)/2 = 22,5$

$D_{24} = 30/1 = 30$

$D_{42} = (60+40)/2 = 50$

$D_{34} = 29/1 = 29$

$D_{43} = 17/1 = 17$

Segue o cálculo da distância entre os grupos:

$\bullet_{12} = \bullet_{21} = (35+25)/2 = 30$

$\bullet_{13} = \bullet_{31} = (28+37)/2 = 32,5$

$\bullet_{14} = \bullet_{41} = (10+15)/2 = 12,5$

$\bullet_{23} = \bullet_{32} = (40+22,5)/2 = 31,25$

$\bullet_{24} = \bullet_{42} = (30+50)/2 = 80$

$\bullet_{34} = \bullet_{43} = (29+17)/2 = 23$

Pela métrica de cálculo utilizada acima, os grupos que estão mais próximo são 1 e 4. Pode-se uni-los, formando assim os seguintes grupos: G_{14} com $I_1 = \{1,3\}$ e $J_{14}\{1,4\}$, G_2 com $I_2 = \{2\}$ e $J_2 = \{3,4\}$ e G_3 com $I_3=\{4\}$ e $J_3 = \{5\}$.

A solução de P considerando a nova configuração dos grupos é: $V(P_{14}) + V(P_2) + V(P_3) = 220+200+150 = 570 = V(P)$, ou seja, os grupos estão bem definidos.

7.4 Formação dos grupos

O objetivo é resolver o *PLC* através da resolução de subproblemas do *PLC*. O primeiro passo é a obtenção desses subproblemas. Tem-se como objetivo a utilização de 3 estratégias, são elas:

1. A partir de grupos pequenos, ou seja, grupos com poucas facilidades e poucos clientes, comparando-se com o número de facilidades e clientes do *PLC*, realizar a união de grupos próximos. Essa união de grupos pararia quando algum critério de parada for alcançado.
2. A partir de um grupo, formado por todas as facilidades e todos os clientes do *PLC,* ir dividindo-o em grupos menores, até que algum critério de parada seja alcançado.
3. Fixar um número *k* de grupos a serem formados. Cada cliente e cada facilidade devem ser colocados no grupo que esteja mais próximo a eles.

Neste trabalho procura-se explorar parte da estratégia um, as outras duas ficam como propostas de trabalhos futuros.

7.4.1 Estratégia 1

Nessa metodologia, para decomposição do *PLC* em grupos, utilizam-se principalmente 3 módulos, são eles:

1. **Formação dos grupos iniciais:** o módulo é baseado na formação inicial de pequenos grupos, isto é, na formação de um grande número de grupos, proporcionalmente ao tamanho do problema, com poucos elementos em cada grupo.
2. **Cálculo da distância entre os grupos:** esse módulo procura indicar a proximidade entre os grupos. Esta medida fornece um parâmetro para uni-los.
3. **Critérios de parada:** a medida que se vai unindo os grupos esses vão crescendo em número de elementos e a distância entre eles vai aumentando. Com isso, utilizaremos dois critérios de parada.
    a. Quando o agrupamento de dois grupos quaisquer fornecer um número de elementos maior do que um número pré-determinado.
    b. Quando à distância entre quaisquer dois grupos for o suficientemente grande.

O algoritmo geral para formação dos grupos é ilustrado abaixo:

inicio
Formar Grupos Iniciais;
Calcular Distância Entre os Grupos;
Enquanto Critério de parada não for atingido ***faça***
início
Agrupar os grupos *x* e *y* tal que Δ_{xy} seja mínimo;
Atualizar a Distância entre os grupos considerando o novo grupo formado;
fim
fim

Pode-se observar que a única sub-rotina, do algoritmo geral, que está fixa é a de agrupar x e y. Para todas as outras existem várias maneiras diferentes de se obter os resultados desejados. Foram descritos, anteriormente, dois critérios de parada que serão utilizados, contudo, os dois critérios não são fixos. O primeiro critério não especifica o número de elementos aceitável e o segundo não especifica o que é uma distância suficientemente grande. Tanto a formação de grupos iniciais quanto o cálculo de distância entre grupos podem ser obtidos de várias formas. Com isso, utilizando a estratégia 1, existem várias possibilidades para a formação de subproblemas do *PLC*.

Serão vistas nas seções seguintes algumas formas de se obter os subproblemas do *PLC*, detalhando cada sub-rotina.

7.4.1.1 1º Forma

7.4.1.1.1 Formação dos grupos iniciais

Criar $|I|$ grupos, onde cada grupo é composto por uma facilidade $k \in I$ e pelos clientes $j \in J$ tal que $d_{kj} = min_{i \in I}\, d_{ij}$. Com isso, tem-se que cada grupo tem uma única facilidade. Em alguns grupos podemos ter vários clientes e em outros nenhum cliente.

7.4.1.1.2 Cálculo da distância entre os grupos

Como visto anteriormente a distância entre dois grupos x e y é dada por Δ_{xy}. Para facilitar o agrupamento foram criados dois vetores para conter os dados referentes à distância entre os grupos. O vetor *dist* e o vetor *grupoDist*. A posição *i* desses vetores representa o grupo *i*. O elemento *dist(i)* representa a menor distância do grupo *i* para todos os outros grupos e o elemento *grupoDist(i)* o grupo associado a essa distância. A figura 43 da um exemplo dos vetores *dist* e *grupoDist*. No exemplo, o grupo 2 tem a menor distância igual a 5, a qual está associada com o grupo 4, conforme o vetor *distGrup*

posição	1	2	3	4
dist	6	5	6	5
grupoDist	3	4	1	2

figura 43 – valores do vetores dist e grupoDist

Os vetores dist e grupoDist são formados no módulo **Calcular Distância Entre os Grupos.**

Na seqüência será verificado como d_{ij} é obtido. Cada facilidade *k* terá uma lista de prioridade com os valores, *1, 2, 3, ..., m*, onde $m = |J|$. O valor 1 representa a distância para o cliente que tem o menor $c_{kj}\ \forall j \in J$, o valor 2 a distância para o cliente que tem o segundo menor c_{kj}, o valor m representa a distância para o cliente que tem o

maior c_{kj}. No exemplo da seção Exemplificação, considerando a facilidade 4, temos os seguintes valores para d_{4j}:

$d_{41} = 5$

$d_{42} = 4$

$d_{43} = 2$

$d_{44} = 3$

$d_{45} = 1$

Esta forma de se obter d_{ij} tenta unir os cliente com as facilidades que tem maior probabilidade de atendê-lo.

7.4.1.1.3 Critério de parada

Utilizam-se três critérios de parada, qualquer um que seja satisfeito terminará o processo de agrupamento. Os dois primeiros são referentes ao primeiro critério de parada do algoritmo geral. O terceiro é referente ao segundo critério de parada do algoritmo geral. Os critérios de paradas são:

1. o agrupamento de quaisquer 2 grupos formará um grupo com mais de 35 facilidades.
2. o agrupamento de quaisquer 2 grupos formará um grupo com mais de 50% dos clientes.
3. quando as distâncias entre grupos forem maiores do que 60% do maior d_{ij}.

O primeiro e o segundo critério utilizado são justificados pela necessidade de se ter grupos de tamanho médio. Essa necessidade está baseada no fato da existência de um algoritmo eficiente para resolver, de forma a obter a solução ótima, Problemas de Localização Capacitado de tamanho médio, o algoritmo desenvolvido por Valiati [39]. O número máximo de facilidades em um grupo, no caso 35, pode ser estendido até 50. Esse número é justificado pelos resultados computacionais apresentados na seção em Valiati[39].

O terceiro critério é bastante subjetivo. A idéia é ter grupos tão distante um do outro que a união dos mesmo não traria nenhum benefício para a solução dos problemas. Isto é, gostar-se-ia de chegar a grupos bem definidos. Para verificar se dois

grupos, x e y, atendem esse critério de parada, basta verificar se $\Delta_{xy} > d * 0,7$, *onde* $d = max\ d_{ij}\ \forall\ i \in I$ *e* $\forall\ j \in J$.

7.4.1.1.4 Atualização da distância entre os grupos

Toda vez que dois grupos, x e y, forem unidos deve-se atualizar os vetores *dist* e *distGrup.* A atualização será realizada para o novo grupo formado e para os grupos $i \in G$, tal que, *distGrup(i)* = x ou *distGrup(i)* = y. Isso significa que é preciso calcular novamente a distância entre todos os grupos.

Vale observar que não é garantido a viabilidade dos grupos, ou seja, pode ser que existam grupos tais que a soma das capacidades das facilidades seja menor do que a soma das demandas dos clientes. Assim sendo, antes de ser resolvido um Problema de Localização Capacitado associado a um grupo deve-se torná-lo viável. Isso será realizado na seção Resolução do problema.

7.4.1.2 2º Forma

A segunda forma difere da primeira somente na maneira de se obter d_{ij}. Assim, as sub-rotinas **Cálculo da Distância entre os Grupos** e **Atualização da Distância entre os Grupos** fornecerão valores diferentes para os vetores *dist* e *distGrup*.

Da mesma maneira que na 1º Forma não é garantido a viabilidade dos grupos, ou seja, pode ser que existam grupos tais que a soma das capacidades das facilidades seja menor do que a soma das demandas dos clientes. Assim sendo, antes de ser resolvido um Problema de Localização Capacitado associado a um grupo deve-se torná-lo viável. Isso será realizado na seção Resolução do problema.

7.4.1.2.1 Formação dos grupos iniciais

Mesmo que 1º Forma.

7.4.1.2.2 Cálculo da distância entre os grupos

Nessa forma considera-se $d_{ij} = c_{ij}$.

7.4.1.2.3 Critério de parada

Mesmo que 1º Forma.

7.4.1.2.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.3 3º Forma

A terceira forma difere da primeira somente na maneira de se obter d_{ij}. Assim, as sub-rotinas **Cálculo da Distância entre os Grupos** e **Atualização da Distância entre os Grupos** fornecerão valores diferentes para os vetores *dist* e *distGrup*.

Da mesma maneira que na 1º Forma não é garantido a viabilidade dos grupos, ou seja, pode ser que existam grupos tais que a soma das capacidades das facilidades seja menor do que a soma das demandas dos clientes. Assim sendo, antes de ser resolvido um Problema de Localização Capacitado associado a um grupo deve-se torná-lo viável. Isso será realizado na seção Resolução do problema.

7.4.1.3.1 Formação dos grupos iniciais

Mesmo que 1º Forma.

7.4.1.3.2 Cálculo da distância entre os grupos

Nessa forma d_{ij} é obtido com segue:

$$d_{ij} = c_{ij} + \frac{f_i}{a_i}$$

Note que d_{ij} representa o custo de transporte mais o rateio do custo fixo para cada unidade que a facilidade suporta.

7.4.1.3.3 Critério de parada

Mesmo que 1º Forma.

7.4.1.3.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.4 4º Forma

A quarta forma difere da terceira na sub-rotina de formação dos grupos iniciais.

Da mesma maneira que na 1º Forma não é garantido a viabilidade dos grupos, ou seja, pode ser que existam grupos tais que a soma das capacidades das facilidades seja menor do que a soma das demandas dos clientes. Assim sendo, antes de ser resolvido um Problema de Localização Capacitado associado a um grupo deve-se torná-lo viável. Isso será realizado na seção Resolução do problema.

7.4.1.4.1 Formação dos grupos iniciais

Criar $|I|$ grupos, onde cada grupo é composto por uma facilidade $k \in I$ e cada grupo terá um número $t \bullet \left\lfloor \frac{|J|}{|I|} \right\rfloor$ de clientes. Inicialmente coloca-se $p = \left\lfloor \frac{|J|}{|I|} \right\rfloor$ clientes em cada grupo, seguindo a seguinte regra:

- Para cada grupo x, que contém a facilidade *k*, colocam-se os *p* clientes que tem os menores custos de transportes e não pertencem a nenhum grupo.

Os $|J| - \left(\left\lfloor \frac{|J|}{|I|} \right\rfloor * |I|\right)$ clientes restantes são colocados nos outros grupos seguindo a seguinte regra: Seja *j* um cliente que não pertença a nenhum grupo, colocar *j* no grupo que tenha a facilidade *k* tal que $d_{kj} = max_{i \in I} c_{ij}$. Em caso de empate coloca-se a o cliente no grupo que tenha a facilidade de menor índice.

7.4.1.4.2 Cálculo da distância entre os grupos

Mesmo que 3º Forma.

7.4.1.4.3 Critério de parada

Mesmo que 1º Forma.

7.4.1.4.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.5 5º Forma

A quinta forma difere da terceira na sub-rotina de formação dos grupos iniciais.

Da mesma maneira que na 1º Forma não é garantido a viabilidade dos grupos, ou seja, pode ser que existam grupos tais que a soma das capacidades das facilidades seja menor do que a soma das demandas dos clientes. Além de não ser garantido a

viabilidade, aqui, pode existir facilidades que pertençam a mais de um grupo. Assim sendo, antes de ser resolvido um Problema de Localização Capacitado associado a um grupo deve-se torná-lo viável e considerar as facilidades que pertençam a mais de um grupo. Isso será realizado na seção Resolução do problema.

7.4.1.5.1 Formação dos grupos iniciais

Criar $|J|$ grupos, onde cada grupo é composto por um cliente $j \in J$ e pelas 4 facilidades que tenham os menores d_{ij} $i \in$ I. As facilidades restantes, aquelas que não foram inseridas em nenhum grupo, são inseridas cada uma em um grupo isolado, ou seja, em um grupo sem clientes. Com isso, têm-se grupos com um cliente e 4 facilidades e grupos com uma única facilidade.

É importante observar que uma facilidade pode estar em mais de um grupo. Isso será considerado na seção de resolução do problema.

7.4.1.5.2 Cálculo da distância entre os grupos

Mesmo que 3º Forma.

7.4.1.5.3 Critério de parada

Mesmo que 1º Forma.

7.4.1.5.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.6 6º Forma

A sexta forma difere da terceira na sub-rotina de formação dos grupos iniciais.

Da mesma maneira que na 1º Forma não é garantido a viabilidade dos grupos, ou seja, pode ser que existam grupos tais que a soma das capacidades das facilidades seja menor do que a soma das demandas dos clientes. Além de não ser garantido a viabilidade, aqui, pode existir facilidades que pertençam a mais de um grupo, Assim sendo, antes de ser resolvido um Problema de Localização Capacitado associado a um grupo deve-se torná-lo viável e considerar as facilidades que pertençam a mais de um grupo. Isso será realizado na seção Resolução do problema.

7.4.1.6.1 Formação dos grupos iniciais

Seja *w(K)* o problema de transporte para $K = I$, conforme definido na Capítulo 2, e x^* a solução ótima dada pela resolução de *w(K)*. Criar $|J|$ grupos, onde cada grupo é composto por um cliente $j \in J$ e pelas facilidades $i \in I$ tal que $x^*_{ij} > 0$. Aqui, limita-se o número de facilidades a 30, caso esse número seja ultrapassado toma-se as 30 facilidades com os maiores x^*_{ij} . As facilidades restantes, aquelas que não atendem nenhum cliente na solução de *w(K)*, são inseridas cada uma em um grupo isolado, ou seja, em um grupo sem clientes. Com isso, têm-se grupos com um cliente e algumas facilidades e grupos com uma única facilidade.

É importante observar que uma facilidade pode estar em mais de um grupo. Isso será considerado na seção de resolução do problema.

7.4.1.6.2 Cálculo da distância entre os grupos

Mesmo que 3º Forma.

7.4.1.6.3 Critério de parada

Mesmo que 1º Forma.

7.4.1.6.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.7 7º Forma

A partir dessa forma será mudado a maneira de se obter Δ_{xy}, não será mais utilizado d_{ij}. A sétima forma difere da sexta na sub-rotina de Cálculo da distância entre os grupos e na sub-rotina Critério de parada.

7.4.1.7.1 Formação dos grupos iniciais

Mesmo que 6º Forma.

7.4.1.7.2 Cálculo da distância entre os grupos

Nessa forma $\bullet_{xy}$ é obtido com segue:

$$\Delta_{xy} = V(P_x \cup P_y) - (V(P_x) + V(P_y))$$

onde $V(P_x \cup P_y)$ representa o valor da solução do Problema de Localização Capacitado unindo-se os grupos *x* e *y*. $V(P_x)$ e $V(P_y)$ representam respectivamente a solução do Problema de Localização Capacitado do grupo *x* e do grupo *y*. É fácil ver que $\bullet_{xy} \leq 0$, como a solução do problema é dado por $V(P_1)+ V(P_2)+...+ V(P_x)+..+V(P_y)+...+ V(P_k)$, então, com a união do grupo *x* e *y* obteremos uma solução dada por $V(P_1)+ V(P_2)+...+ V(P_x \cup P_y)+...+ V(P_k)$ que fornecerá um valor menor ou igual para o *PLC*, pois $V(P_x \cup P_y) \leq V(P_x) + V(P_y)$. Com isso, serão unidos os grupos com a menor distância, pois fornecerão, num primeiro momento, uma melhor solução para o problema *PLC*. Quando $\bullet_{xy} = 0$, nenhum ganho imediato, na solução do problema *PLC*, é obtido na união dos grupos *x* e *y*.

Os vetores *dist* e *grupoDist* continuarão sendo utilizados, como mostrado na primeira forma. Assim sendo, a rotina de Atualização da distância entre os grupos continua sendo realizada como na primeira forma, claro, agora utilizando a nova metodologia para se obter $\bullet_{xy}$.

7.4.1.7.3 Critério de parada

Tem-se 2 critérios de parada, qualquer um que seja satisfeito terminará o processo de agrupamento. O primeiro é referente ao primeiro critério de parada do algoritmo geral. O segundo é referente ao segundo critério de parada do algoritmo geral. Os critérios de paradas são:

1. o agrupamento de quaisquer 2 grupos formará um grupo com mais de 35 facilidades.
2. quando $\bullet_{xy} = 0$ para quaisquer grupos x e y.

O primeiro critério utilizado é justificado pela necessidade de se ter grupos de tamanho médio. Essa necessidade está baseada no fato da existência de um algoritmo eficiente para resolver, de forma a obter a solução ótima, Problemas de Localização Capacitado de tamanho médio, o algoritmo de Valiati [39]. O número máximo de

facilidades em um grupo, no caso 35, pode ser estendido até 50. Esse número é justificado pelos resultados computacionais apresentados em Valiati [39].

O segundo critério está baseado no fato de que a união entre dois grupos não alteraria a solução do *PLC*. Contudo, isso não garante que os grupos são bem definidos, pois pode existir 3 ou mais grupos que unidos forneça uma solução melhor para o *PLC*.

7.4.1.7.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.8 8º Forma

A oitava forma difere da sétima na sub-rotina de formação dos grupos iniciais. Nesta forma não é garantido a viabilidades dos grupos.

7.4.1.8.1 Formação dos grupos iniciais

Seja *w(K)* o problema de transporte para $K = I$, conforme definido no Capítulo 2, e x^* a solução ótima dada pela resolução de *w(K)*. Criar $|I|$ grupos, onde cada grupo é composto por uma facilidade $i \in I$ e pelos cliente tal que $x^*_{ij} > 0$, sendo que, a demanda do cliente j, b_j, é substituída por x^*_{ij}. Seja $k \in I$ uma facilidade, tal que, $x^*_{kj} = 0 \ \forall j \in J$, então a facilidade k é inserida em um grupo isolado, ou seja, em um grupo sem clientes.

É importante observar que um cliente $j \in J$ pode ser dividido em vários clientes $j_1, j_2, ...,j_k$, onde $b_j = b_{j1} + b_{j2} + .. + b_{jk}$ e cada cliente $j_1, j_2, ...,j_k$ pertence a um grupo diferente. Quando na união de dois grupos existir, um cliente que foi separado, na formação dos grupos iniciais, deve ser unido. Isto é, o cliente passa a ser único, sendo que a demanda desses novos clientes é a soma das demandas dos clientes que foram divididos anteriormente.

7.4.1.8.2 Cálculo da distância entre os grupos

Mesmo que 7º Forma.

7.4.1.8.3 Critério de parada

Mesmo que 7º Forma.

7.4.1.8.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.9 9º Forma

A nona forma difere da oitava na sub-rotina de Cálculo da distância entre os grupos e na sub-rotina Critério de parada.

7.4.1.9.1 Formação dos grupos iniciais

Mesmo que 8º Forma.

7.4.1.9.2 Cálculo da distância entre os grupos

Nessa forma é utilizado o problema de transporte para encontra a distância entre os grupos. Seja x e y dois grupos com seus respectivos conjuntos de cliente e facilidades I_x, J_x e I_y, J_y. Seja também $w(K,T) = min\{\Sigma_{k\in K}\Sigma_{t\in T}\ c_{kt}x_{kt} \mid x \in X(K,T)\}$ o Problema de Transporte associado ao *PLC*, onde $K \subseteq I$ é um sub-conjunto de facilidades e $T \subseteq J$ é um sub-conjunto de clientes e $X(K,T) = \{x \geq 0 | \Sigma_{t\in T}\ x_{kt} \leq a_k, \forall\ k \in K;\ \Sigma_{k\in K}x_{kt} = b_t,\ \forall t \in T\}$ o conjunto de soluções viáveis. Então a distância de um grupo x para um grupo y é definido como $D_{xy} = w(I_x,J_y)$. Assim sendo, $D_{yx} = w(I_y,J_x)$. Ou seja, para o cálculo da distância de um grupo para o outro é utilizado o Problema de Transporte, onde os centros de ofertas são representados pelas facilidades de um grupo e os centros de demanda são representados pelos clientes do outro grupo. A distância entre dois grupos, x e y, é dada por:

$\Delta_{xy} = \Delta_{yx} = (D_{xy} + D_{yx}) / 2$

Quando $w(I_y,J_x)$ e/ou $w(I_x,J_y)$ forem inviáveis será utilizado uma proporção para encontrar a distância entre os grupos. Insere-se, no Problema de Transporte, um centro de oferta, com a oferta igual a demanda restante (dr), $dr = \sum_{i\in I_x} a_i - \sum_{j\in J_y} b_j$, para torná-lo viável. Seja w(.) o novo Problema de Transporte gerado e v(w(.)) o valor da solução ótima de w(.). Para encontramos D_{xy}, supondo que $w(I_x,J_y)$ seja inviável, utiliza-se a seguinte fórmula:

$$D_{xy} = \frac{v(w(.))}{\sum_{i\in I_x} a_i} * \sum_{j\in J_y} b_j$$

Para encontrar D_{yx}, supondo que $w(I_y,J_x)$ seja inviável, utiliza-se a seguinte fórmula:

$$D_{yx} = \frac{v(w(.))}{\sum_{i \in I_y} a_i} * \sum_{j \in J_x} b_j$$

7.4.1.9.3 Critério de parada

Tem-se 2 critérios de parada, qualquer um que seja satisfeito terminará o processo de agrupamento. O primeiro é referente ao primeiro critério de parada do algoritmo geral. O segundo é referente ao segundo critério de parada do algoritmo geral. Os critérios de paradas são:

1. o agrupamento de quaisquer 2 grupos formará um grupo com mais de 35 facilidades.
2. quando $\Delta_{xy} > \frac{w(I,J)}{\frac{|I|}{8}}$ para quaisquer grupos *x* e *y*.

O primeiro critério utilizado é justificado pela necessidade de termos grupos de tamanho médio. Essa necessidade está baseada no fato da existência de um algoritmo eficiente para resolver, de forma a obter a solução ótima, Problemas de Localização Capacitado de tamanho médio, o algoritmo de Valiati [39]. O número máximo de facilidades em um grupo, no caso 35, pode ser estendido até 50. Esse número é justificado pelos resultados computacionais apresentados em Valiati [39].

O segundo critério é bastante subjetivo. A idéia é ter grupos tão distante um do outro que não teria nenhum benefício a junção desses. Isto é, gostar-se-ia de chegar a grupos bem definidos. O critério foi escolhido através de diversos testes.

7.4.1.9.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.10 10º Forma

A décima forma difere da nona na sub-rotina de Cálculo da distância entre os grupos e na sub-rotina Critério de parada. Contudo a diferença é pequena, somente na formação do Problema de Transporte no que se refere ao custo de transporte.

7.4.1.10.1 Formação dos grupos iniciais

Mesmo que 8º Forma.

7.4.1.10.2 Cálculo da distância entre os grupos

A única coisa que muda, com relação a 9º forma, é a substituição do custo de transporte c_{ij}, utilizando em $w(K,T) = min\{\Sigma_{k\in K}\Sigma_{t\in T}\, c_{kt}x_{kt} \mid x \in X(K,T)\}$, por $c_{ij} + \frac{f_i}{a_i}$. Logo $w(K,T) = min\{\Sigma_{k\in K}\Sigma_{t\in T}\, (c_{kt}+(f_k/a_k))^* x_{kt} \mid x \in X(K,T)\}$. O restante continua igual a 9º forma.

7.4.1.10.3 Critério de parada

Mesmo que na 9º Forma, contudo utilizando $w(K,T) = min\{\Sigma_{k\in K}\Sigma_{t\in T}\, (c_{kt}+(f_k/a_k))^* x_{kt} \mid x \in X(K,T)\}$.

7.4.1.10.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.11 11º Forma

A décima primeira forma difere da décima na sub-rotina de Formação dos grupos iniciais. Essa rotina é a mesma utilizada na 6º forma.

Da mesma maneira que na 6º Forma, não é garantido a viabilidade dos grupos e pode-se ter facilidades que pertençam a mais de um grupo. Isso será considerado na seção Resolução do problema.

7.4.1.11.1 Formação dos grupos iniciais

Mesmo que 6º Forma.

7.4.1.11.2 Cálculo da distância entre os grupos

Mesmo que 10º Forma.

7.4.1.11.3 Critério de parada

Mesmo que 9º Forma.

7.4.1.11.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.12 12º Forma

A décima segunda forma difere da décima primeira na sub-rotina Cálculo da distância entre os grupos e na sub-rotina Critério de parada.

Da mesma maneira que na 6º Forma, não é garantido a viabilidade dos grupos e pode-se ter facilidades que pertençam a mais de um grupo. Isso será considerado na seção Resolução do problema.

7.4.1.12.1 Formação dos grupos iniciais

Mesmo que 6º Forma.

7.4.1.12.2 Cálculo da distância entre os grupos

Nessa forma é utilizado o problema de transporte para encontra a distância entre os grupos. Seja *x* e *y* dois grupos com seus respectivos conjuntos de cliente e facilidades I_x, J_x e I_y, J_y. Seja $w(K,T) = min\{\Sigma_{k\in K}\Sigma_{t\in T} (c_{kt}+(f_k/a_k))* x_{kt} \mid x \in X(K,T)\}$ o Problema de Transporte associado ao *PLC*, onde $K \subseteq I$ é um sub-conjunto de facilidades e T $\subseteq J$ é um sub-conjunto de clientes e $X(K,T) = \{x \geq 0 | \Sigma_{t\in T} x_{kt} \leq a_k, \forall k \in K; \Sigma_{k\in K} x_{kt} = b_t, \forall t \in T\}$ o conjunto de soluções viáveis. Notem que o custo de transporte do *PLC*, c_{ij}, foi substituído por $c_{ij} + \frac{f_i}{a_i}$

Defini-se a distância entre os grupos *x* e *y* como $\Delta_{xy} = \Delta_{yx} = w(I_x \cup I_y, J_x \cup J_y)\frac{\sum_{i\in I_x\cup I_y} a_i}{\sum_{j\in J_x\cup J_y} b_j}$. Ou seja, para o cálculo da distância entre grupos é utilizado o Problema de Transporte, onde os centros de ofertas são representados pelas facilidades dos dois grupos e os centros de demanda, também, são representados pelos clientes dos dois grupos. A expressão $\frac{\sum_{i\in I_x\cup I_y} a_i}{\sum_{j\in J_x\cup J_y} b_j}$ é utilizada para tentar não favorecer os grupos com muitas facilidades.

Quando $w(I_x\cup I_y, J_x\cup J_y)$ for inviável utiliza-se uma aproximação para encontrar a distância entre os grupos. Inseri-se, no Problema de Transporte, um centro de oferta, com a oferta igual a demanda restante (*dr*), $dr = \sum_{i\in I_x\cup I_y} a_i - \sum_{j\in J_x\cup J_y} b_j$, para torná-lo

viável. Seja *k* esse novo centro de oferta e $j \in J_x \cup J_y$ um centro de demanda, então o custo de transporte de *k* para *j* é definido como segue: $c_{kj} = \frac{\sum_{i \in I_x \cup I_y} c_{ij}}{|I_x| + |I_y|}$

7.4.1.12.3 Critério de parada

Tem-se 2 critérios de parada, qualquer um que seja satisfeito terminará o processo de agrupamento. O primeiro é referente ao primeiro critério de parada do algoritmo geral. O segundo é referente ao segundo critério de parada do algoritmo geral. Os critérios de paradas são:

1. o agrupamento de quaisquer 2 grupos formará um grupo com mais de 35 facilidades.
2. quando $\Delta_{xy} > \left(w(I,J) * \frac{\sum_{i \in I} a_i}{\sum_{j \in J} b_j} \right) * 2$ para quaisquer grupos *x* e *y*.

O primeiro critério utilizado é justificado pela necessidade de se ter grupos de tamanho médio. Essa necessidade está baseada no fato da existência de um algoritmo eficiente para resolver, de forma a obter a solução ótima, Problemas de Localização Capacitado de tamanho médio, o algoritmo de Valiati [39]. O número máximo de facilidades em um grupo, no caso 35, pode ser estendido até 50. Esse número é justificado pelos resultados computacionais apresentados em Valiati [39].

O segundo critério foi obtido através de testes, visando uma boas distribuição entre os grupos.

7.4.1.12.4 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.13 13º Forma

Nessa forma é considerado inicialmente somente as facilidades no processo de agrupamento, ou seja, serão formados grupos que conterão somente facilidades. Embora, inicialmente, um grupo seja composto somente de facilidades, quando é realizado o cálculo da distância entre eles, os clientes são considerados. Os clientes servem com parâmetros de distância entre as facilidades. Após o termino do

agrupamento das facilidades inseri-se os clientes nos grupos que são mais próximos, formando assim, grupos com facilidades e clientes, os quais representam um subproblema do *PLC*. Com isso, insere-se mais uma sub-rotina no algoritmo geral, a de inserção de clientes nos grupos. Nessa forma, utiliza-se um pós-processamento nos grupos formados. O pós-processamento está baseado na troca de elementos entre os grupos. Assim sendo, o algoritmo de formação dos grupos tem a seguinte forma:

inicio
Formar Grupos Iniciais;
Calcular Distância Entre os Grupos;
Enquanto Critério de parada não for atingido ***faça***
início
Agrupar x e y tal que Δ_{xy} seja mínimo;
Atualizar a Distância entre os grupos considerando o novo grupo formado;
Fim
Inserir Clientes nos Grupos;
Pós-processamento;
Fim

Da mesma maneira que na 1º Forma não é garantido a viabilidade dos grupos, ou seja, pode ser que existam grupos tais que a soma das capacidades das facilidades seja menor do que a soma das demandas dos clientes. Assim sendo, antes de ser resolvido um Problema de Localização Capacitado associado a um grupo deve-se torná-lo viável. Isso será realizado na seção Resolução do problema.

7.4.1.13.1 Formação dos grupos iniciais

Criar $|I|$ grupos, onde cada grupo é composto por uma facilidade $i \in I$. Os clientes não são inseridos nesta fase, portanto, nenhum grupo possui clientes.

7.4.1.13.2 Cálculo da distância entre os grupos

Seja d_{ij} a distância da facilidade i para o cliente j. Seja $k, h \in I$, defini-se a distância entre as facilidades k e h da seguinte forma:

$$\mathrm{df}_{\mathrm{kh}} = \mathrm{df}_{\mathrm{hk}} = \sum_{j \in J} | d_{kj} - d_{hj} |$$

Sejam x e y dois grupos, defini-se a distância entre eles da seguinte forma:

$$\Delta_{\mathrm{xy}} = \frac{\sum_{k \in I_x} \sum_{h \in I_y} df_{kh}}{| I_x | * | I_y |}$$

A distância de um cliente *j* para uma facilidade *i* é obtida da seguinte forma:

$$d_{ij} = c_{ij} + \frac{f_i}{a_i}$$

7.4.1.14 Critério de parada

Tem-se 2 critérios de parada, qualquer um que seja satisfeito terminará o processo de agrupamento. O primeiro é referente ao primeiro critério de parada do algoritmo geral. O segundo é referente ao segundo critério de parada do algoritmo geral. Os critérios de paradas são:

1. o agrupamento de quaisquer 2 grupos formará um grupo com mais de 35 facilidades.
2. quando $\Delta_{xy} > (\max_{i \in I, j \in J} d_{ij} - \min_{i \in I, j \in J} d_{ij}) * |J| * 0.3$ para quaisquer grupos *x* e *y*.

O primeiro critério utilizado é justificado pela necessidade de termos grupos de tamanho médio. Essa necessidade está baseada no fato da existência de um algoritmo eficiente para resolver, de forma a obter a solução ótima, Problemas de Localização Capacitado de tamanho médio, o algoritmo de Valiati [39]. O número máximo de facilidades em um grupo, no caso 35, pode ser estendido até 50. Esse número é justificado pelos resultados computacionais apresentados em Valiati [39].

O segundo critério é bastante subjetivo. A idéia é se ter grupos tão distante um do outro que não teria nenhum benefício a junção desses. Isto é, gostar-se-ia de chegar a grupos bem definidos. O critério foi escolhido através de diversos testes.

7.4.1.14.1 Atualização da distância entre os grupos

Mesmo que 1º Forma.

7.4.1.14.2 Inserir Clientes nos Grupos

O objetivo é colocar cada cliente no grupo que esteja mais apto a atendê-lo. Utiliza-se a seguinte fórmula para medir a aptidão (proximidade) de um cliente *j* para com um grupo *x*.

$$B_{jx} = \frac{\sum_{i \in I_x} d_{ij}}{|I_x|}$$

O cliente *j* será inserido no grupo *y* tal que $B_{jy} = \min_{x \in G} B_{jx}$.

7.4.1.14.3 Pós-processamento

Inicialmente toma-se cada facilidade $i \in I$ e calcula-se a proximidade de *i* em relação a cada grupo $x \in G$. A proximidade de uma facilidade *i* para um grupo *x* é obtida da seguinte maneira:

$$F_{ix} = \frac{\sum_{k \in I_x} \sum_{j \in J_x} |d_{ij} - d_{kj}|}{|I_x|}$$

Seja *y* o grupo em que *i* pertença, e seja *p* o grupo tal que $F_{ip} = min_{x \in G} F_{ix}$. Caso $y \neq p$ retira-se a facilidade *i* do grupo *y* e a insere no grupo *p*, esse processo é chamado de troca facilidades. Caso exista alguma troca, realiza-se o mesmo processo para os clientes, ou seja, para cada cliente $j \in J$ calcula-se a proximidade de *j* em relação a cada grupo $x \in G$. A proximidade de um cliente *j* para um grupo *x* é obtida da mesma maneira que foi obtida na sub-rotina Inserir Clientes nos Grupos, isto é:

$$B_{jx} = \frac{\sum_{i \in I_x} d_{ij}}{|I_x|}$$

Seja *y* o grupo em que *j* pertença, e seja *p* o grupo tal que $B_{ip} = min_{x \in G} B_{ix}$. Caso $y \neq p$ retira-se o cliente *i* do grupo *y* e o insere no grupo *p*, esse processo é chamado de troca de clientes. Caso exista alguma troca de clientes, inicializa-se novamente o processo de troca de facilidades e clientes. Esse processo é repetido no máximo 20 vezes ou até não haver troca de clientes ou facilidades.

7.5 Resolução do problema

Inicialmente, para solucionar o *PLC*, será resolvido, para cada grupo *x*, o Problema de Localização Capacitado associado, P_x. Com a junção das soluções dos subproblemas uma solução viável para o *PLC* será gerada, a qual será representada por y' e x' e denominada $v'(PLC)$. Será visto como a solução $v'(PLC)$ é obtida ainda nesta seção.

No processo de agrupamento não é garantido, na maioria das formas, que os grupos sejam viáveis, ou seja, pode ser que existam grupos tais que a soma das capacidades das facilidades seja menor do que a soma das demandas dos clientes. Assim sendo, antes de ser resolvido um grupo deve-se torná-lo viável.

Para se alcançar a viabilidade de um grupo *x*, será inserindo facilidades nesse grupo até torná-lo viável. A primeira facilidade a ser inserida no grupo *x* será a facilidade *k* tal que $F_{kx} = min_{i \in I, i \notin Ix} F_{ix}$. A segunda segue a mesma regra e assim por diante, até que o grupo se torne viável. Esta regra, para obtenção da viabilidade, faz com que uma facilidade possa pertencer a mais de um grupo. Isto é, se existir algum grupo inviável é necessário ter pelo menos uma facilidade em mais de um grupo. Assim sendo, deve-se fazer um controle rigoroso dos dados das facilidades que pertencem a mais de um grupo. Esse caso ocorre, em algumas formas de agrupamento, com isso, deve-se fazer com que essas formas se enquadre no processo de controle dos dados das facilidades.

Seja *x* um grupo, quando o P_x é resolvido de se fazer atualizações/modificações nos grupos que ainda não foram resolvidos. Seja y^+ e x^+ a solução de P_x e $i \in I_k$, então as atualizações/modificações são as seguintes:

- Atualizações:

  caso $y_i^+ = 1$, então $\forall k \in G$, tal que $k \bullet x$, *k* ainda não resolvido e $i \in I_k$, fazer para facilidade $i \in I_k$ $a_i = a_i - \sum_{j \in J_x} x_{ij}^+$ e $f_i = 0$. Isto é, reduzir a capacidade, considerado a capacidade, que a facilidade *i*, utilizou no grupo *x* e colocar o custo fixo da facilidade *i* como zero, pois este custo já foi considerado no grupo *x*.

- Modificações

  caso $y_i^+ = 1$, então $\forall k \in G$, tal que $k \bullet x$, *k* ainda não resolvido e $i \notin I_k$, inserir a facilidade *i* em I_k com $a_i = a_i - \sum_{j \in J_x} x_{ij}^+$ e $f_i = 0$. Isto é, inserir a facilidade *i* no grupo *k* com a capacidade restante, depois do seu atendimento ao grupo *x,* e colocar o custo fixo da facilidade *i* como zero, pois este custo já foi considerado no grupo *x*.

Caso exista algum grupo x tal que $J_x = \varnothing$ as facilidade $i \in I_x$ serão incluídas no grupo y tal que $F_{iy} = min_{k \in G, k \bullet x} F_{ik}$.

Os passos para resolução do problema são os seguintes:

- **Passo 0**
  $Z_g = 0$;
- **Passo I**
  Tomar o grupo x que tenha maior número de facilidades, torná-lo viável e resolver o Problema de Localização Capacitado (*PLC*) associado, ou seja, resolver P_x . Fazer $Z_g = Z_g + v(P_x)$, onde $v(P_x)$ é o valor da solução ótima de P_x.
- **Passo II (atualização dos grupos)**
  Realizar as atualizações e modificações.

- **Passo III (viabilidade)**
  Como uma facilidade pode pertencer a mais de um grupo, a resolução de um pode levar a inviabilidade do outro. Isso porque a capacidade da facilidade comum aos dois grupos pode reduzir quando essa for aberta na resolução de um deles. Assim sendo, deve-se tornar os grupos que perderam a viabilidade, viáveis novamente.
- **Passo IV**
  Caso exista um grupo que ainda não tenha sido resolvido ir para Passo I.
  Caso contrário $v'(PLC) = Z_z$.

7.6 Pós-processamento na resolução do problema

As facilidades abertas na solução dada por $v'(PLC)$ são as que melhor se enquadraram em cada grupo. Caso os grupos sejam bem definidos, a solução, seria ótima. Contudo, saber se os grupos são bem definidos consiste num problema de complexidade elevada. Sendo a união dos grupos e os critérios de parada, de certa forma, subjetivos, os grupos gerados podem não se enquadram na expressão bem definidos. Para tentar sanar esta possível distorção, dando assim uma solução melhor para o *PLC*, será realizado um pós-processamento sobre a solução obtida na resolução dos grupos. Serão mostrados, na seqüência, os tipos de processamento desenvolvidos, a maioria deles consistem em resolver mais um Problema de Localização Capacitado. Esse problema, na maioria das vezes, será formado pelas facilidades $\forall i \in I$ tal que $y'_i = 1$ mais algumas facilidades $i \in I$ tal que $y'_i = 0$, não abertas na resolução dos grupos, que tenham boa probabilidade de serem abertas na solução ótima do *PLC*. Será visto a quantidade e a forma de se obter estas facilidades na seqüência. Não foi testado esse processo para todas as formas de agrupamento, contudo, com o que foi testado, já

fornece um bom parâmetro para a análise. Segue os tipos de processamentos desenvolvidos:

7.6.1 Resolução do Problema de Transporte

Esse é o mais simples, o único em que não é resolvido um Problema de Localização Capacitado. Aqui simplesmente é resolvido *w(K)*, definido no Capítulo 2, para $K = \{\forall i \in I \mid y'_i=1\}$. A solução do *PLC* ficaria $v(P) = w(k) + \sum_{i \in I} f_i y'_i$.

7.6.2 Resolução do Problema de Localização Capacitado sem inclusão de facilidades

É formado um Problema de Localização Capacitado com o conjunto de facilidades $I_{pos} = \{\forall i \in I \mid y'_i=1\}$ e com o conjunto de clientes *J*.

7.6.3 Resolução do Problema de Localização Capacitado com inclusão de facilidades

É resolvido um Problema de Localização Capacitado com o conjunto de facilidades $I_{pos} = \{\forall i \in I \mid y'_i=1\}$ mais algumas que são acrescentada e com o conjunto de clientes *J*. O número de facilidades a serem acrescentadas depende do tamanho de I_{pos}, o aumento será de 40% do seu tamanho. Esse critério foi adotado pelo fado de que o método por Valiati [39], geralmente, é eficiente para problemas que abrem muitas facilidades na solução ótima, mais do que 40% pode tornar o método ineficiente. Foram consideramos 5 métodos para acrescentar facilidades, os quais serão mostrados a seguir:

7.6.3.1 Acrescentar facilidades considerando o C-Teste

Seja $K=\{\forall i \in I \mid y'_i=1\}$, realiza-se o C-teste, $\Delta_i(K)$, para $\forall i \in I$ tal que $y'_i=0$. As facilidades que participarão do aumento serão aquela com melhores valores de $\Delta_i(K)$, considerando a abertura de facilidades, ou seja, as que tiverem os menores valores.

7.6.3.2 Acrescentar facilidades considerando o O-Teste

Realiza-se o O-Teste, $\Delta_i(I - \{i\})$, para $\forall i \in I$ tal que $y'_i=0$. As facilidade que participarão do aumento serão aquelas com melhores valores de $\Delta_i(I-\{i\})$, considerando a abertura de facilidades, ou seja, as que tiverem os menores valores.

7.6.3.3 Acrescentar facilidades considerando o C-Teste, custo fixo e capacidade

Metade da facilidades são acrescentadas utilizando o mesmo critério que a seção Acrescentar facilidades considerando o C-Teste. A outra metade é obtida pela razão entre o custo fixo e a capacidade, tomamos as facilidades com menores g_i, onde $g_i = \frac{f_i}{a_i} \ \forall i \in I.$

7.6.3.4 Acrescentar facilidades considerando o C-Teste e o fechamento de facilidades

Seja $K=\{\forall i \in I \mid y'_i=1\}$, realiza-se o C-teste, $\Delta_i(K-\{l\})$, para $\forall i \in I$ tal que $y'_i=0$ e $\forall l \in I$ tal que $y'_i=1$. Ou seja, é simulado o fechamento de uma das facilidades abertas pela resolução dos grupos e simulado o C-Teste para todas as facilidades que foram fechadas na resolução dos grupos. Obtém-se, para cada facilidade $i \in I$ tal que $y'_i=0$ a possibilidade dela ser aberta na solução ótima $p_i = \min_{l \in I | y_i'=1} \Delta_i(K-\{l\})$. Quanto menor for o valor de p_i maior a possibilidade de a facilidade i ser aberta, com isso, são incluídas as facilidades com os menores valores de p_i.

7.6.3.5 Acrescentar facilidades considerando o Problema de Transporte com uma única facilidade

Seja $C = min_{i \in I, j \in J} c_{ij}$, e seja $w(K_i)$ o Problema de Transporte, conforme definido no Capítulo 2, onde $K_i = \{i,p\}$ sendo que $i \in I$, e p um centro de oferta com capacidade $a_p = \sum_{j \in J} b_j - a_i$ e seus custos variáveis $c_{pj} = C \ \forall j \in J$. Resolve-se $w(K_i) \ \forall i \in I$ tal que $y'_i=0$ e são acrescentas as facilidades com os menores $w(K_i)$.

7.7 Resultados computacionais

Utilizou-se a linguagem C++ para implementação dos algoritmos apresentados. Os testes foram rodados sobre os problemas de Kühn & Hamburger [32], Beasley [11] e Cornuejols, Sridharan e Thizy [20].

Será mostrado, nessa seção, apenas os resultados mais significativos. Forma obtidos resultados excelentes para a 3º, 6º e 13º formas, são para essas formas e a 10º

que serão apresentados os resultados computacionais. A 10º forma foi incluída para servir de referência de uma forma que não obteve bons resultados. Em relação ao pós-processamento da solução gerada pelo agrupamento, serão mostrados os métodos apresentados nas seções 7.6.1, 7.6.2, 7.6.3.1 e 7.6.3.2, pois os resultados dos outros métodos, seção 7.6.3.3, 7.6.3.4 e 7.6.3.5, são muito parecidos com os resultados da seção 7.6.3.1, assim sendo, é evitado a poluição excessiva dos resultados apresentados.

As Tabela 72 e Tabela 73 apresentam os resultados computacionais da 3º forma e dos pós-processamentos apresentados nas seções 7.6.1, 7.6.2, 7.6.3.1 e 7.6.3.2, Os problemas da Tabela 72 são os propostos por Cornuejols, Sridharan e Thizy [20] e os problema da tabela Tabela 73 são os propostos por Kühn & Hamburger [32] (cap41 a cap134) e por Beasley [11] (capa1 a capc4). Segue a explicação do que representa cada coluna:

- A coluna n representa o número de facilidades do problema.
- A coluna abertas representa o número de facilidades abertas na solução ótima.
- A coluna %abertas representa o percentual de facilidades abertas na solução ótima em relação ao número total de facilidades
- A coluna E(Zgrupo) representa o erro percentual gerado pela solução do agrupamento. Seja Z^* a solução ótima do problema e Z a solução gerada pelo agrupamento, então o erro é obtido da seguinte maneira $E(Zgrupo)=100*(Z-Z^*)/ Z^*$.
- A coluna E(7.6.1) representa o erro percentual gerado pela solução do pós-processamento da seção 7.6.1. Seja Z^* a solução ótima do problema e Z a solução gerada pelo pós-processamento, então o erro é obtido da seguinte maneira $E(7.6.1)=100*(Z-Z^*)/ Z^*$.
- A coluna E(7.6.2) representa o erro percentual gerado pela solução do pós-processamento da seção 7.6.27.6.1. Seja Z^* a solução ótima do problema e Z a solução gerada pelo pós-processamento, então o erro é obtido da seguinte maneira $E(7.6.2)=100*(Z-Z^*)/ Z^*$.
- A coluna E(7.6.3.1) representa o erro percentual gerado pela solução do pós-processamento da seção 7.6.3.17.6.1. Seja Z^* a solução ótima do problema e Z a solução gerada pelo pós-processamento, então o erro é obtido da seguinte maneira $E(7.6.3.1)=100*(Z-Z^*)/ Z^*$.

- A coluna E(7.6.3.2) representa o erro percentual gerado pela solução do pós-processamento da seção 7.6.3.27.6.1. Seja Z^* a solução ótima do problema e Z a solução gerada pelo pós-processamento, então o erro é obtido da seguinte maneira $E(7.6.3.2)=100*(Z-Z^*)/ Z^*$.

É apresentado no final das tabelas Tabela 72 e Tabela 73 a média dos erros gerados e a média dos erros gerados para problemas que abrem, na solução ótima, um número maior ou igual a 20% do total de facilidades (média • 20%).

Os resultados computacionais apresentados nas tabelas Tabela 72 e Tabela 73 são muito bons, principalmente os da coluna **E(**7.6.3.2**)**. Nessa coluna para os problemas de Kühn & Hamburger [32] os resultados são melhores do que o método de Beasley [12] e para problemas que abrem, na solução ótima, um número maior ou igual a 20% do total de facilidades só é gerado um único erro, considerando-se os problemas de Kühn & Hamburger [32], Beasley [11] e Cornuejols, Sridharan e Thizy [20]. O erro foi gerado no problema cap114 e mesmo assim foi baixíssimo de 0,04% .

Problema	N	Abertas	%Abertas	E(Zgrupo)	E(7.6.1)	E(7.6.2)	E(7.6.3.1)	E(7.6.3.2)
thiz1-b1	16	9	56,25	0,000	0,000	0,000	0,000	0,000
thiz1-b2	16	7	43,75	0,000	0,000	0,000	0,000	0,000
thiz1-b3	16	8	50	0,000	0,000	0,000	0,000	0,000
thiz1-b4	16	8	50	0,000	0,000	0,000	0,000	0,000
thiz1-b5	16	10	62,5	0,000	0,000	0,000	0,000	0,000
thiz1-c1	25	11	44	0,000	0,000	0,000	0,000	0,000
thiz1-c2	25	12	48	0,000	0,000	0,000	0,000	0,000
thiz1-c3	25	12	48	0,000	0,000	0,000	0,000	0,000
thiz1-c4	25	10	40	0,000	0,000	0,000	0,000	0,000
thiz1-c5	25	13	52	0,000	0,000	0,000	0,000	0,000
thiz1-d1	16	8	50	0,000	0,000	0,000	0,000	0,000
thiz1-d2	16	7	43,75	0,000	0,000	0,000	0,000	0,000
thiz1-d3	16	8	50	0,000	0,000	0,000	0,000	0,000
thiz1-d4	16	8	50	0,000	0,000	0,000	0,000	0,000
thiz1-d5	16	7	43,75	0,000	0,000	0,000	0,000	0,000
thiz1-e1	16	16	100	0,000	0,000	0,000	0,000	0,000
thiz1-e2	33	16	48,48	0,000	0,000	0,000	0,000	0,000
thiz1-e3	33	15	45,45	0,000	0,000	0,000	0,000	0,000
thiz1-e4	33	16	48,48	0,000	0,000	0,000	0,000	0,000
thiz1-e5	33	16	48,48	0,000	0,000	0,000	0,000	0,000
thiz1-f1	50	23	46	0,000	0,000	0,000	0,000	0,000
thiz1-f2	50	24	48	0,000	0,000	0,000	0,000	0,000
thiz1-f3	50	24	48	0,000	0,000	0,000	0,000	0,000
thiz1-f4	50	23	46	0,000	0,000	0,000	0,000	0,000

thiz1-f5	50	23	46	0,000	0,000	0,000	0,000	0,000
thiz2-b1	16	5	31,25	0,000	0,000	0,000	0,000	0,000
thiz2-b2	16	6	37,5	0,000	0,000	0,000	0,000	0,000
thiz2-b3	16	5	31,25	0,000	0,000	0,000	0,000	0,000
thiz2-b4	16	6	37,5	0,000	0,000	0,000	0,000	0,000
thiz2-b5	16	5	31,25	0,000	0,000	0,000	0,000	0,000
thiz2-c1	25	9	36	0,000	0,000	0,000	0,000	0,000
thiz2-c2	25	9	36	0,000	0,000	0,000	0,000	0,000
thiz2-c3	25	8	32	0,000	0,000	0,000	0,000	0,000
thiz2-c4	25	8	32	0,000	0,000	0,000	0,000	0,000
thiz2-c5	25	8	32	0,000	0,000	0,000	0,000	0,000
thiz2-d1	16	6	37,5	0,000	0,000	0,000	0,000	0,000
thiz2-d2	16	5	31,25	0,000	0,000	0,000	0,000	0,000
thiz2-d3	16	6	37,5	0,000	0,000	0,000	0,000	0,000
thiz2-d4	16	7	43,75	6,430	6,116	6,116	0,239	0,000
thiz2-d5	16	5	31,25	0,000	0,000	0,000	0,000	0,000
thiz2-e1	33	10	30,30	0,000	0,000	0,000	0,000	0,000
thiz2-e2	33	12	36,36	0,000	0,000	0,000	0,000	0,000
thiz2-e3	33	10	30,30	0,000	0,000	0,000	0,000	0,000
thiz2-e4	33	12	36,36	0,000	0,000	0,000	0,000	0,000
thiz2-e5	33	11	33,33	0,000	0,000	0,000	0,000	0,000
thiz2-f1	50	17	34	0,000	0,000	0,000	0,000	0,000
thiz2-f2	50	16	32	0,000	0,000	0,000	0,000	0,000
thiz2-f3	50	15	30	0,000	0,000	0,000	0,000	0,000
thiz2-f4	50	16	32	0,000	0,000	0,000	0,000	0,000
thiz2-f5	50	16	32	0,000	0,000	0,000	0,000	0,000
thiz3-b1	16	4	25	0,000	0,000	0,000	0,000	0,000
thiz3-b2	16	4	25	0,000	0,000	0,000	0,000	0,000
thiz3-b3	16	4	25	0,000	0,000	0,000	0,000	0,000
thiz3-b4	16	3	18,75	0,000	0,000	0,000	0,000	0,000
thiz3-b5	16	3	18,75	0,000	0,000	0,000	0,000	0,000
thiz3-c1	25	5	20	0,000	0,000	0,000	0,000	0,000
thiz3-c2	25	6	24	0,000	0,000	0,000	0,000	0,000
thiz3-c3	25	5	20	0,000	0,000	0,000	0,000	0,000
thiz3-c4	25	5	20	0,000	0,000	0,000	0,000	0,000
thiz3-c5	25	5	20	0,000	0,000	0,000	0,000	0,000
thiz3-d1	16	4	25	0,000	0,000	0,000	0,000	0,000
thiz3-d2	16	4	25	0,000	0,000	0,000	0,000	0,000
thiz3-d3	16	4	25	0,000	0,000	0,000	0,000	0,000
thiz3-d4	16	4	25	0,000	0,000	0,000	0,000	0,000
thiz3-d5	16	3	18,75	0,000	0,000	0,000	0,000	0,000
thiz3-e1	33	7	21,21	0,000	0,000	0,000	0,000	0,000
thiz3-e2	33	7	21,21	0,000	0,000	0,000	0,000	0,000
thiz3-e3	33	7	21,21	0,000	0,000	0,000	0,000	0,000
thiz3-e4	33	7	21,21	0,000	0,000	0,000	0,000	0,000
thiz3-e5	33	7	21,21	0,000	0,000	0,000	0,000	0,000
thiz3-f1	50	10	20	0,000	0,000	0,000	0,000	0,000
thiz3-f2	50	10	20	0,000	0,000	0,000	0,000	0,000
thiz3-f3	50	11	22	0,000	0,000	0,000	0,000	0,000
thiz3-f4	50	11	22	0,000	0,000	0,000	0,000	0,000
thiz3-f5	50	11	22	0,000	0,000	0,000	0,000	0,000

thiz5-b1	16	2	12,5	0,000	0,000	0,000	0,000	0,000
thiz5-b2	16	2	12,5	0,000	0,000	0,000	0,000	0,000
thiz5-b3	16	3	18,75	0,000	0,000	0,000	0,000	0,000
thiz5-b4	16	2	12,5	0,000	0,000	0,000	0,000	0,000
thiz5-b5	16	2	12,5	0,000	0,000	0,000	0,000	0,000
thiz5-c1	25	3	12	0,000	0,000	0,000	0,000	0,000
thiz5-c2	25	4	16	0,000	0,000	0,000	0,000	0,000
thiz5-c3	25	3	12	0,000	0,000	0,000	0,000	0,000
thiz5-c4	25	3	12	0,000	0,000	0,000	0,000	0,000
thiz5-c5	25	4	16	0,000	0,000	0,000	0,000	0,000
thiz5-d1	16	2	12,5	0,000	0,000	0,000	0,000	0,000
thiz5-d2	16	2	12,5	0,000	0,000	0,000	0,000	0,000
thiz5-d3	16	2	12,5	0,868	0,166	0,166	0,166	0,000
thiz5-d4	16	3	18,75	0,000	0,000	0,000	0,000	0,000
thiz5-d5	16	2	12,5	0,000	0,000	0,000	0,000	0,000
thiz5-e1	33	4	12,12	0,000	0,000	0,000	0,000	0,000
thiz5-e2	33	4	12,12	0,000	0,000	0,000	0,000	0,000
thiz5-e3	33	4	12,12	0,000	0,000	0,000	0,000	0,000
thiz5-e4	33	4	12,12	0,000	0,000	0,000	0,000	0,000
thiz5-e5	33	4	12,12	0,000	0,000	0,000	0,000	0,000
thiz5-f1	50	6	12	0,000	0,000	0,000	0,000	0,000
thiz5-f2	50	6	12	0,000	0,000	0,000	0,000	0,000
thiz5-f3	50	7	14	0,000	0,000	0,000	0,000	0,000
thiz5-f4	50	6	12	0,000	0,000	0,000	0,000	0,000
thiz5-f5	50	7	14	3,392	3,333	3,333	3,333	3,333
thizx-b1	16	1	6,25	11,594	11,594	11,594	11,594	0,000
thizx-b2	16	1	6,25	0,000	0,000	0,000	0,000	0,000
thizx-b3	16	1	6,25	0,000	0,000	0,000	0,000	0,000
thizx-b4	16	1	6,25	0,000	0,000	0,000	0,000	0,000
thizx-b5	16	1	6,25	15,514	15,514	15,514	15,514	0,000
thizx-c1	25	2	8	0,000	0,000	0,000	0,000	0,000
thizx-c2	25	2	8	0,000	0,000	0,000	0,000	0,000
thizx-c3	25	2	8	0,000	0,000	0,000	0,000	0,000
thizx-c4	25	2	8	0,000	0,000	0,000	0,000	0,000
thizx-c5	25	2	8	0,000	0,000	0,000	0,000	0,000
thizx-d1	16	2	12,5	0,000	0,000	0,000	0,000	0,000
thizx-d2	16	2	12,5	0,000	0,000	0,000	0,000	0,000
thizx-d3	16	2	12,5	0,000	0,000	0,000	0,000	0,000
thizx-d4	16	1	6,25	0,000	0,000	0,000	0,000	0,000
thizx-d5	16	1	6,25	0,000	0,000	0,000	0,000	0,000
thizx-e1	33	2	6,06	0,000	0,000	0,000	0,000	0,000
thizx-e2	33	2	6,06	0,000	0,000	0,000	0,000	0,000
thizx-e3	33	3	9,09	0,000	0,000	0,000	0,000	0,000
thizx-e4	33	2	6,06	0,000	0,000	0,000	0,000	0,000
thizx-e5	33	2	6,06	0,000	0,000	0,000	0,000	0,000
thizx-f1	50	3	6	4,721	4,356	4,356	4,356	4,356
thizx-f2	50	3	6	0,000	0,000	0,000	0,000	0,000
thizx-f3	50	3	6	5,593	5,093	5,093	5,093	3,007
thizx-f4	50	3	6	3,329	3,316	3,316	3,316	3,316
thizx-f5	50	4	8	2,049	1,379	1,379	1,379	1,379
Média				***0,428***	***0,407***	***0,407***	***0,36***	***0,123***

Média • 20%				***0,089***	***0,085***	***0,085***	***0,000***	***0,003***

Tabela 72 – Resultados computacionais (problemas teste de Thizy)

Problema	N	Abertas	%Abertas	E(Zgrupo)	E(7.6.1)	E(7.6.2)	E(7.6.3.1)	E(7.6.3.2)
cap41	16	13	81,25	0,000	0,000	0,000	0,000	0,000
cap42	16	12	75	0,000	0,000	0,000	0,000	0,000
cap43	16	12	75	0,000	0,000	0,000	0,000	0,000
cap44	16	12	75	0,000	0,000	0,000	0,000	0,000
cap51	16	8	50	0,000	0,000	0,000	0,000	0,000
cap61	16	11	68,75	0,000	0,000	0,000	0,000	0,000
cap62	16	9	56,25	0,000	0,000	0,000	0,000	0,000
cap63	16	7	43,75	0,000	0,000	0,000	0,000	0,000
cap64	16	5	31,25	0,000	0,000	0,000	0,000	0,000
cap71	16	11	68,75	0,000	0,000	0,000	0,000	0,000
cap72	16	9	56,25	0,000	0,000	0,000	0,000	0,000
cap73	16	5	31,25	0,000	0,000	0,000	0,000	0,000
cap74	16	4	25	0,000	0,000	0,000	0,000	0,000
cap81	25	17	68	0,000	0,000	0,000	0,000	0,000
cap82	25	14	56	0,000	0,000	0,000	0,000	0,000
cap83	25	14	56	0,000	0,000	0,000	0,000	0,000
cap84	25	13	52	0,028	0,000	0,000	0,000	0,000
cap91	25	15	60	0,013	0,000	0,000	0,000	0,000
cap92	25	11	44	0,089	0,089	0,089	0,089	0,000
cap93	25	8	32	0,000	0,000	0,000	0,000	0,000
cap94	25	7	28	0,000	0,000	0,000	0,000	0,000
cap101	25	15	60	0,013	0,000	0,000	0,000	0,000
cap102	25	11	44	0,089	0,089	0,089	0,089	0,000
cap103	25	8	32	0,089	0,089	0,089	0,000	0,000
cap104	25	4	16	0,018	0,000	0,000	0,000	0,000
cap111	50	17	34	0,117	0,117	0,117	0,117	0,000
cap112	50	15	30	0,079	0,079	0,079	0,000	0,000
cap113	50	14	28	0,420	0,414	0,414	0,083	0,000
cap114	50	13	26	0,741	0,261	0,261	0,261	0,049
cap121	50	15	30	0,232	0,211	0,108	0,108	0,000
cap122	50	11	22	0,701	0,682	0,000	0,000	0,000
cap123	50	9	18	0,996	0,324	0,324	0,257	0,000
cap124	50	7	14	0,803	0,167	0,167	0,104	0,000
cap131	50	15	30	0,232	0,211	0,108	0,108	0,000
cap132	50	11	22	0,702	0,682	0,000	0,000	0,000
cap133	50	8	16	0,788	0,104	0,104	0,104	0,000
cap134	50	4	8	0,713	0,056	0,000	0,000	0,000
capa1	100	7	7	2,743	0,823	0,823	0,291	0,078
capa2	100	6	6	1,973	0,000	0,000	0,000	0,000
capa3	100	5	5	2,932	1,917	1,917	1,917	1,500
capa4	100	4	4	4,957	3,837	1,861	1,861	1,086
capb1	100	11	11	3,004	1,755	1,755	1,747	0,289
capb2	100	9	9	3,133	2,482	1,568	1,568	0,541
capb3	100	8	8	4,899	2,660	2,660	2,660	0,604

capb4	100	7	7	1,091	0,554	0,554	0,554	0,000
capc1	100	11	11	0,360	0,057	0,057	0,057	0,057
capc2	100	10	10	0,849	0,544	0,194	0,194	0,113
capc3	100	9	9	0,639	0,505	0,505	0,198	0,198
capc4	100	9	9	0,884	0,618	0,618	0,207	0,207
Média				***0,701***	***0,394***	***0,295***	***0,257***	***0,096***
Média Cap41 a Cap134				***0,186***	***0,097***	***0,053***	***0,001***	***0,036***
Média • 20%				***0,111***	***0,091***	***0,042***	***0,001***	***0,027***

Tabela 73 – Resultados computacionais (problemas teste de Kühn & Hamburger e Bealey)

A *T*abela 74 apresenta os resultados computacionais da 6º forma, 10º forma e 13º forma. Os problemas da *T*abela 74 são os propostos por Cornuejols, Sridharan e Thizy [20]. Segue a explicação do que representa cada coluna:

- A coluna n representa o número de facilidades do problema.
- A coluna abertas representa o número de facilidades abertas na solução ótima.
- A coluna %abertas representa o percentual de facilidades abertas na solução ótima em relação ao número total de facilidades
- A coluna E(forma6) representa o erro percentual gerado pela solução do agrupamento da 6º forma. Seja Z^* a solução ótima do problema e Z a solução gerada pelo agrupamento, então o erro é obtido da seguinte maneira $E(forma6)=100*(Z-Z^*)/ Z^*$.
- A coluna E(forma13) representa o erro percentual gerado pela solução do agrupamento da 13º forma. Seja Z^* a solução ótima do problema e Z a solução gerada pelo agrupamento, então o erro é obtido da seguinte maneira $E(forma13)=100*(Z-Z^*)/ Z^*$.
- A coluna E(forma10) representa o erro percentual gerado pela solução do agrupamento da 10º forma. Seja Z^* a solução ótima do problema e Z a solução gerada pelo agrupamento, então o erro é obtido da seguinte maneira $E(forma10)=100*(Z-Z^*)/ Z^*$.

Apresenta-se no final da *T*abela 74 a média dos erros gerados e a média dos erros gerados para problemas que abrem, na solução ótima, um número maior ou igual a 20% do total de facilidades (média $\bullet 20\%$).

Notem que a 10º forma, embora tenha um método de cálculo de distância "sofisticado", não apresenta resultado bons. Já a 6º e a 13º formas apresentam ótimos resultados. Perceba que essas são melhores do que a 3º forma considerando todos os problemas e piores considerando os problemas que abrem, na solução ótima, um número maior ou igual a 20% do total de facilidades.

Problema	|I|	Abertas	%abertas	E(Forma6)	E(forma13)	E(forma10)
thiz1-b1	16	9	56,25	1,98	0,00	0,45
thiz1-b2	16	7	43,75	0,66	0,00	2,47
thiz1-b3	16	8	50	0,00	0,00	5,77
thiz1-b4	16	8	50	0,00	0,00	6,04
thiz1-b5	16	10	62,5	1,86	0,00	0,74
thiz1-c1	25	11	44	0,00	0,00	8,59
thiz1-c2	25	12	48	0,00	0,00	2,89
thiz1-c3	25	12	48	0,10	0,00	1,74
thiz1-c4	25	10	40	0,00	0,00	4,09
thiz1-c5	25	13	52	0,20	0,00	2,43
thiz1-d1	16	8	50	0,00	0,00	1,38
thiz1-d2	16	7	43,75	0,00	0,00	9,14
thiz1-d3	16	8	50	0,00	0,00	4,32
thiz1-d4	16	8	50	0,00	0,00	4,68
thiz1-d5	16	7	43,75	0,00	0,00	0,70
thiz1-e1	16	16	100	0,25	0,00	2,91
thiz1-e2	33	16	48,48	0,03	0,00	3,59
thiz1-e3	33	15	45,45	0,00	0,00	3,62
thiz1-e4	33	16	48,48	0,00	0,00	4,01
thiz1-e5	33	16	48,48	0,00	0,12	2,91
thiz1-f1	50	23	46	0,00	0,00	5,95
thiz1-f2	50	24	48	0,00	0,00	2,11
thiz1-f3	50	24	48	0,00	0,33	2,77
thiz1-f4	50	23	46	0,00	0,00	1,93
thiz1-f5	50	23	46	0,00	0,10	5,08
thiz2-b1	16	5	31,25	0,00	0,00	10,03
thiz2-b2	16	6	37,5	0,00	0,00	3,21
thiz2-b3	16	5	31,25	0,00	0,00	9,49
thiz2-b4	16	6	37,5	0,00	0,00	3,12
thiz2-b5	16	5	31,25	0,00	0,00	8,73
thiz2-c1	25	9	36	0,00	0,00	5,67
thiz2-c2	25	9	36	0,00	0,00	4,10
thiz2-c3	25	8	32	0,00	0,00	5,00
thiz2-c4	25	8	32	0,00	0,00	5,90
thiz2-c5	25	8	32	0,00	0,00	2,12

thiz2-d1	16	6	37,5	0,00	0,00	8,18
thiz2-d2	16	5	31,25	0,00	0,00	8,55
thiz2-d3	16	6	37,5	0,00	0,00	8,49
thiz2-d4	16	7	43,75	2,65	0,00	6,90
thiz2-d5	16	5	31,25	0,00	0,00	21,30
thiz2-e1	33	10	30,30	0,00	0,00	3,52
thiz2-e2	33	12	36,36	0,00	0,00	6,86
thiz2-e3	33	10	30,30	0,00	0,00	5,27
thiz2-e4	33	12	36,36	0,00	0,00	5,08
thiz2-e5	33	11	33,33	0,00	0,00	5,43
thiz2-f1	50	17	34	0,00	0,00	6,26
thiz2-f2	50	16	32	0,00	0,00	0,93
thiz2-f3	50	15	30	0,00	0,00	6,55
thiz2-f4	50	16	32	0,00	0,00	2,29
thiz2-f5	50	16	32	0,00	5,49	5,38
thiz3-b1	16	4	25	0,00	0,00	14,20
thiz3-b2	16	4	25	0,00	0,00	2,85
thiz3-b3	16	4	25	0,13	0,00	12,71
thiz3-b4	16	3	18,75	1,14	0,00	17,60
thiz3-b5	16	3	18,75	0,00	0,00	7,89
thiz3-c1	25	5	20	0,00	0,00	11,47
thiz3-c2	25	6	24	2,11	0,00	3,44
thiz3-c3	25	5	20	0,00	0,00	8,13
thiz3-c4	25	5	20	0,00	0,00	8,09
thiz3-c5	25	5	20	0,00	0,00	8,34
thiz3-d1	16	4	25	0,00	0,00	5,59
thiz3-d2	16	4	25	0,00	0,00	11,95
thiz3-d3	16	4	25	0,00	0,00	8,08
thiz3-d4	16	4	25	0,00	0,00	11,89
thiz3-d5	16	3	18,75	0,00	0,00	13,03
thiz3-e1	33	7	21,21	0,00	0,00	3,68
thiz3-e2	33	7	21,21	0,00	0,00	1,62
thiz3-e3	33	7	21,21	0,00	0,00	7,96
thiz3-e4	33	7	21,21	0,00	0,00	6,81
thiz3-e5	33	7	21,21	0,00	0,00	6,67
thiz3-f1	50	10	20	0,00	0,00	5,16
thiz3-f2	50	10	20	0,00	0,00	5,53
thiz3-f3	50	11	22	0,00	0,00	7,66
thiz3-f4	50	11	22	0,00	0,14	5,14
thiz3-f5	50	11	22	0,00	1,67	3,46
thiz5-b1	16	2	12,5	0,00	0,00	24,57
thiz5-b2	16	2	12,5	0,00	0,00	15,74
thiz5-b3	16	3	18,75	4,55	0,00	8,25
thiz5-b4	16	2	12,5	0,00	0,00	10,29
thiz5-b5	16	2	12,5	0,00	0,00	18,30
thiz5-c1	25	3	12	0,00	0,00	10,20
thiz5-c2	25	4	16	0,00	0,00	0,42
thiz5-c3	25	3	12	0,00	0,00	9,20
thiz5-c4	25	3	12	0,00	0,00	8,52
thiz5-c5	25	4	16	0,00	0,00	11,84
thiz5-d1	16	2	12,5	0,00	0,00	18,96

thiz5-d2	16	2	12,5	0,00	0,00	5,17
thiz5-d3	16	2	12,5	7,33	0,00	7,72
thiz5-d4	16	3	18,75	0,00	0,00	1,29
thiz5-d5	16	2	12,5	0,00	0,00	14,57
thiz5-e1	33	4	12,12	0,00	0,00	8,82
thiz5-e2	33	4	12,12	0,00	0,00	13,80
thiz5-e3	33	4	12,12	0,00	0,00	8,34
thiz5-e4	33	4	12,12	0,00	0,00	8,35
thiz5-e5	33	4	12,12	0,00	0,00	6,24
thiz5-f1	50	6	12	0,00	0,00	6,43
thiz5-f2	50	6	12	0,00	0,00	11,75
thiz5-f3	50	7	14	0,00	0,00	6,07
thiz5-f4	50	6	12	0,00	1,04	19,11
thiz5-f5	50	7	14	4,35	2,10	12,88
thizx-b1	16	1	6,25	11,59	0,00	20,82
thizx-b2	16	1	6,25	0,00	0,00	11,61
thizx-b3	16	1	6,25	0,00	0,88	19,87
thizx-b4	16	1	6,25	17,80	0,00	26,29
thizx-b5	16	1	6,25	23,11	0,00	22,36
thizx-c1	25	2	8	0,00	0,00	14,57
thizx-c2	25	2	8	0,00	0,00	25,75
thizx-c3	25	2	8	0,00	0,00	15,62
thizx-c4	25	2	8	0,00	0,00	28,82
thizx-c5	25	2	8	0,00	0,00	12,69
thizx-d1	16	2	12,5	0,00	0,00	12,50
thizx-d2	16	2	12,5	0,00	0,00	14,96
thizx-d3	16	2	12,5	0,00	0,00	12,73
thizx-d4	16	1	6,25	0,00	0,00	11,85
thizx-d5	16	1	6,25	0,00	0,00	12,00
thizx-e1	33	2	6,06	0,00	0,00	6,81
thizx-e2	33	2	6,06	0,00	0,00	11,61
thizx-e3	33	3	9,09	0,00	0,00	11,65
thizx-e4	33	2	6,06	0,00	0,00	12,40
thizx-e5	33	2	6,06	0,00	0,00	13,95
thizx-f1	50	3	6	4,72	6,68	10,40
thizx-f2	50	3	6	0,00	0,00	19,06
thizx-f3	50	3	6	5,40	7,14	14,79
thizx-f4	50	3	6	3,33	9,75	11,86
thizx-f5	50	4	8	1,61	0,75	15,38
Média				***0,76***	***0,29***	***8,84***
Média • 20%				***0,14***	***0.10***	***5,68***

***T*abela 74 – Resultados computacionais (problemas teste de Thizy)**

7.8 Análises e propostas

Os métodos apresentados, utilizando a estratégia 1, mostraram ser muito bons para problemas que abrem, na solução ótima, um número de facilidades maior ou igual a 20% do total. As formas que se destacaram foram a 3º, a 6º e a 13º.

As 5º, 6º, 7º, 11º e 12º formas, trabalham inicialmente com $|J|$ grupos. Como nos problemas, geralmente, o número de clientes sempre é maior do que o número de facilidades, o tempo computacional despedido foi bem maior do que nas formas que trabalham inicialmente com $|I|$ grupos. Além de estas formas serem mais lentas não acrescentaram em nada na qualidade da solução, comparando-se com as outras formas.

É possível classificar as formas apresentadas pelo modo de cálculo da distância entre os grupos. São três os modos:

1. Baseados em d_{ij}: 1º, 2º, 3º, 4º, 5º, 6º, 7º e 13º formas.
2. Baseados no Problema de Transporte: 9º, 10º, 11º e 12º formas.
3. Baseados no Problema de Localização Capacitado: 7º e 8º formas.

A complexidade do cálculo vai do modo 1, o menos complexo, ao modo 3, o mais complexo. Embora o modo 1 tenha o cálculo da distância mais simples de todos, foi esse que apresentou os melhores resultados. O modo 3, que parecia a principio ser o melhor, embora lento, não teve bons resultados. Esse demonstrou ser um modo muito cego e guloso. Outro fato importante está no custo entre um cliente e uma facilidade, quando este é considerado para o cálculo da distância entre grupos. Os melhores resultados se apresentaram quando foi utilizado $c_{ij} + \frac{f_i}{a_i}$ em vez de c_{ij}, isso tanto para as formas que utilizam d_{ij}, como base para o cálculo da distância, quanto para as formas que utilizam o Problema de Transporte.

Na fase de resolução do problema a ordem de resolução dos grupos pode alterar a solução final. Foi utilizado dois critérios para a escolha de qual grupo seria solucionado primeiro. O primeiro critério escolhe o grupo com melhor consistência e o segundo o grupo com maior número de facilidades. O segundo critério demonstrou-se melhor, foi este o utilizado nos resultados computacionais apresentados.

Na fase de pós-processamento da solução, o método que apresentou os melhores resultados foi o dada seção 7.6.3.2. Embora, nesse método existe a necessidade de se resolver um *PLC*, em nenhum momento pesou no tempo computacional da solução do problema.

Tem-se os seguintes objetivos futuros:

- Aplicar novas "métricas" na fase de agrupamento. "Métricas" que podem ser utilizadas são estatísticas, resolução do Problema não Capacitado entre outras;
- Mesclar os modos de cálculo de distância, procurando aplicar cada um em momentos apropriados;
- Explorar as estratégias 1 e 2;
- Aplicar os métodos que mais se destacam em problemas maiores, dos que foram testados até então, e em problemas não euclidianos.
- Realizar um rearranjo dos elementos nos grupos, utilizando um processamento parecido ao procedimento pós-processamento da 13º forma. Esse rearranjo pode ser tanto durante a formação dos grupos como após.
- Desenvolver uma heurística, sem a utilização de grupos, caso não for possível obter uma solução com qualidade para os problemas que abram menos do que 20% das facilidades na solução ótima. Esta heurística visaria esses tipos de problemas. Assim, mesmo que o usuário não tenha uma estimativa do número de facilidades que serão abertas, ele pode rodar os dois métodos e utilizar a melhor solução.

Anexo 3

8 Anexo 2: Método de Benders aplicado ao Problema de Localização Capacitado

Será proposto um método de resolução exata que utiliza três tipos de decomposições do problema original, duas decomposições primal e uma dual. O método é baseado na decomposição de Benders e utiliza o algoritmo desenvolvido por Valiati [39] para resolver um subproblema do problema original. Procura-se fazer cortes mais precisos no problema mestre de Benders. Com isso, a tendência é se obter a solução ótima do problema com o acréscimo de poucas restrições ao problema mestre. Assim sendo, além da necessidade de se resolver um número pequeno de problemas mestre, esse fica de fácil resolução.

Será mostrado na seção 8.1 o método de Benders para a resolução de um problema genérico de programação inteira mista. Na seção 8.2 será apresentado o método de Benders específico para o *PLC* e na seção 8.3 uma variação do método de Benders, esta variação utiliza o algoritmo desenvolvido por Valiati [39] para resolver subproblemas do *PLC*.

8.1 Método de Benders

Benders, no início da década dos anos 60, introduziu em sua tese de doutorado uma nova maneira de decompor problemas de programação matemática [13]. Várias variantes do método de Benders têm sido implementadas visando solucionar modelos aplicados ao planejamento energético, a programação não linear em estruturas especiais, problemas de localização, etc. Será apresentado nas próximas seções seu método para resolução de problemas de programação inteira mista.

8.1.1 Definições e Generalidades

Seja o Problema de programação inteira mista:

(8-1) $(P)\ \min z = cy + dx$

S.a.

(8-2) $Ay + Dx \geq b$

(8-3) $x \geq 0$

(8-4) $y \in \{0,1\}^n$

onde $c^T \in R^n$, $d^T \in R^m$, $y \in R^n$, $x \in R^m$, $b \in R^m$.

Fixando um valor para cada variável inteira *y*, obtendo assim $y = \bar{y} \in \{0,1\}^n$, *P* ficaria para $\bar{y}$ da seguinte forma:

(8-5) $(Py)\ \min z = c\bar{y} + dx$

S.a.

(8-6) $Dx \geq b - A\bar{y}$

(8-7) $x \geq 0$

Se equação (8-6) e equação (8-7) não formarem um conjunto vazio, isto é, existe $\bar{x}$ tal que $\bar{x} \geq 0$ e $D\bar{x} \geq b - A\bar{y}$, logo $(\bar{y}, \bar{x})$ será uma solução viável de *P*.

O dual de *Py* tem a seguinte forma:

(8-8) $(D)\ \max z = c\bar{y} + u(b - A\bar{y})$

S.a.

(8-9) $uD \leq d$

(8-10) $u \geq 0$

onde $c^T \in R^m$. Lembrando que $c\bar{y}$ é uma constante.

Seja $U=\{ u \geq 0 \mid uD \leq d \}$ um conjunto poliédrico convexo. Pode-se observar que *U* independe de $\bar{y}$, ou seja, a viabilidade de *U* independe dos valores atribuídos às

variáveis *y*. Sejam u^p, *p=1,2,...,P* os vértices de *U* e v^s, s=1,2,...,S os raios extremos de *U*. Isto quer dizer que $\forall u \in U$, é possível escrever.

$u = \sum_{p=1}^{P} \lambda_p u^p + \sum_{s=1}^{S} \mu_s v^s$, onde $\sum_{p=1}^{P} \lambda_p = 1, \lambda_p \geq 0, p = 1,2,....,P$ e $\mu_s \geq 0, s = 1,2,...,S.$

De acordo com a teoria da dualidade em programação linear pode-se constatar que:

1. Se $U=\varnothing$ então *Py* será vazio ou ilimitado, ocorrendo o mesmo para *P*, pois para quaisquer valores atribuídos às variáveis *y, U* será vazio.

2. Se *D* admitir uma solução ótima ao menos uma será um vértice de *U*.

3. Se *D* para $y=\bar{y}$ não possuir solução ótima limitada, isto é, existe v^s para o qual $v^s(b-Ay)>0$, neste caso *Py* será vazio e não existirá $(\bar{y}, x)$ viável para *P*. Isto fará com que apenas as soluções $y \in \{0,1\}$ tais que $v^s(b-Ay) \bullet 0$, para *s=1,2,...,S*, sejam interessantes.

Para ilustrar o item 3 toma-se a função objetivo de *D*, isto é, $t = c\bar{y} + u(b - A\bar{y})$. Se v^s é um raio extremo de *U* para o qual $v^s(b-Ay)>0$ então seja $\hat{u} = \bar{u} + \mu v^s$ para $\mu_s \geq 0$ onde $\bar{u} \in D$. Sabe-se que $\hat{u} \in U$ e seja $u = \bar{u}$ na função objetivo de D: $t = c\bar{y} + \hat{u}(b - A\bar{y})$ ou ainda $t = c\bar{y} + (\bar{u} + \mu v^s)(b - A\bar{y}) = c\bar{y} + \bar{u}(b - A\bar{y}) + \mu v^s (b - A\bar{y})$. Se $\mu \to \infty$ então $t \to \infty$, pois
$v^s(b-Ay)>0$.

Supondo $U \bullet \varnothing$, é possível transformar *D* no seguinte problema:

(8-11) $(D') : \max_{p=1,2,...,P} \{u^p (b - Ay)\}$

S.a.

(8-12) $v^s (b - Ay) \leq 0, \qquad s=1,2,...,S$

(8-13) $y \in \{0,1\}^n$

Só os vetores *y* satisfazendo (8-12) e (8-13) farão *(y,x)* viáveis em *P*. Assim sendo, *P* poderá ser escrito da seguinte maneira:

(8-14) $\min_{y \in \{0,1\}^n} cy + \max_{p=1,...,P} \{u^p (b - Ay) \mid v^s (b - Ay) \leq 0, s = 1,2,...,S\}$

Seja $z' = cy + \max_{p=1,...,P} \{u^p (b - Ay) \mid v^s (b - Ay) \leq 0, s = 1,2,...,S\}$, então

$z' \geq cy + u^p (b - Ay), \qquad p=1,2,...,P$

$v^s (b - Ay) \leq 0, \qquad s=1,2,...,S$

logo:

(8-15) $(P) : \min z = z`$

S.a.

(8-16) $z' \geq cy + u^p (b - Ay), \qquad p=1,2,...,P$

(8-17) $v^s (b - Ay) \leq 0, \qquad s=1,2,...,S$

(8-18) $y \in \{0,1\}^n$

O problema *P* tornou-se um problema de programação linear 0-1 mais uma variável contínua *z'*. O método de enumeração de Balas pode ser modificado para resolver este problema misto, bem particular.

8.1.2 Algoritmo de Benders

O problema mestre de Benders *R* é o problema *P* relaxado da seguinte maneira:

(8-19) $(R): \min z = z`$

S.a.

(8-20) $z' \geq cy + u^p(b - Ay), \qquad p=1,2,...,P' \bullet P$

(8-21) $v^s(b - Ay) \leq 0, \qquad s=1,2,...,S' \bullet S$

(8-22) $y \in \{0,1\}^n$

A idéia geral do algoritmo é que em cada iteração seja introduzidas novas restrições a *R*. é claro que *V(R)* • *V(P)*, isto é, o valor ótimo de *R* é um limite inferior do valor ótimo de *P*. As soluções *(x,y)* viáveis de *P* que serão encontradas no algoritmo fornecem limites superiores para o ótimo de *P*. O algoritmo é apresentado a seguir:

Passo 1: {inicialização}
Tomar uma instância de y, $\bar{y} \in \{0,1\}^n$
$z^{sup} \leftarrow \infty$
$z^{inf} \leftarrow -\infty$

Passo 2:
Resolver *D*
Se *U=*$\varnothing$ ***Então***
PARE (*P* será vazio ou terá uma solução ótima ilimitada)
Se *D* ilimitado ***Então***
Ir para o Passo 4.(um raio extremo v^s de *U* foi encontrado)
Senão (um vértice u^p de *U* foi encontrado)
Se $z^{sup} \geq c\bar{y} + u^p(b - A\bar{y})$ ***Então***
$z^{sup} \leftarrow c\bar{y} + u^p(b - A\bar{y})$

Passo 3:
Acrescentar ao problema *R* a restrição $z' \geq c\bar{y} + u^p(b - A\bar{y})$
Ir para o Passo 5

Passo 4:
Acrescentar ao problema *R* a restrição $v^s(b - A\bar{y}) \leq 0$

Passo 5:
Resolver *R* (*y'* é a solução ótima de *R*)
Se *R=*$\varnothing$ ***Então***
PARE (*P* também será vazio)
$Z^{inf} \leftarrow V(R)$

Passo 6:
Se $z^{inf} < z^{sup}$ ***Então***
Inicio
$\bar{y} \leftarrow y'$
Ir para o Passo 2
Fim
Senão
Inicio
$\bar{y} \leftarrow y'$
Resolver $P\bar{y}$ (que fornecerá $\bar{x}$)
$(\bar{y}, \bar{x})$ é a solução ótima de *P*
Fim

8.1.3 Convergência

Benders mostrou que este algoritmo é finito e ótimo. Será verificada isso de uma maneira bastante intuitiva. Quando é resolvido *D* para uma instância $\bar{y}$ de *y*, acrescenta-se em *R* ou uma restrição informando que $\bar{y}$ não pode ser considerada, pois o problema fica inviável, ou uma restrição para que uma nova instância $\bar{y}'$ de *y* seja gerada. Quando é resolvido *R* gera-se uma instância $\bar{y}$ de *y*. Se $\bar{y}$ for igual a uma instância $\hat{y}$ gerada anteriormente, então $\bar{y}$ é uma instância ótima. Para provar, será suposto que $\bar{y}$ não é uma instância da solução ótima. Seja $z^{inf} = V(R)$ associado a $\bar{y}$, como $\bar{y}$ não está associado a solução ótima $z^{inf} < z^{sup}$. Sabemos que $z^{sup} \bullet V(P\ \hat{y})$, mas $V(P\ \hat{y}) = c\hat{y} + u^p(b - A\hat{y}) \leq V(R) = z^{inf}$, chega-se a uma contradição $z^{inf} \geq z^{sup}$. Logo, se $\bar{y} = \hat{y}$ foi atingido uma instância ótima para *y*. Assim sendo, *R* sempre gerará um $\bar{y}$ diferentes dos anteriores caso este não seja ótimo, como o numero de instância de *y* é finita, $O(2^n)$, o algoritmo converge para a solução ótima em tempo finito. Vale observar que, caso *D* gere uma restrição redundante, a solução de *R* será um vetor $\bar{y}$ que já foi gerado anteriormente, logo é obtido a solução ótima.

8.2 Utilizando o Método de Benders para resolver o Problema de Localização Capacitado

Agora será utilizado o método de Benders para resolver o *PLC*. Seja o problema conforme definido na seção 1.1. Toma-se $y = \overline{y} \in \{0,1\}$, assim sendo, *PLC* fica da seguinte maneira:

(8-23) $$(PLCy) : \min \sum_{i\in I}\sum_{i\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i \overline{y}_i$$

S.a.

(8-24) $$\sum_{j\in J} x_{ij} \le a_i \overline{y}_i, \quad \forall i \in I$$

(8-25) $$\sum_{i\in I} x_{ij} \le b_j, \quad \forall j \in J$$

(8-26) $$x_{ij} \ge 0, \quad \forall i \in I, \forall ji \in J$$

Note que $f_i \overline{y}_i$ e $a_i \overline{y}_i$ são constantes. O subproblema *PLCy* é o Problema de Transporte, que pode ser resolvido, por exemplo, pelo método do Anexo 1. Assim sendo, o trabalho computacional será simples adições e comparações, veja Ahrens e Finke [1]. Se (8-24) e (8-25) não formarem um conjunto vazio então $(\overline{y}, \overline{x})$ será uma solução viável de *PLC*. O problema dual de *PLCy* e definido como:

(8-27) $$(PLCD) : \max \sum_{i\in I} f_i \overline{y}_i - \sum_{i\in I} \mu_i a_i \overline{y}_i + \sum_{j\in J} \alpha_j b_j$$

S.a.

(8-28) $$\mu_i \alpha_j \ge c_{ij}, \quad \forall i \in I, \forall j \in J$$

(8-29) $$\mu_i \ge 0, \quad \forall i \in I$$

Seja $U=\{ \mu_i \ge 0 \mid \mu_i \alpha_j \ge c_{ij}, \forall i \in I, \forall j \in J\}$, então se *PLCD* admitir uma solução ótima ela será um vértice de *U*. Para que o *PLCD* tenha solução ótima *PLCy* deve admitir solução viável e limitada. Para isso, basta que $\sum_{i\in I} a_i \overline{y}_i \ge \sum_{j\in J} b_j$. Desta forma, *PLC* pode ser escrito como:

(8-30) $(PLCM\quad): \min_{y\in\{0,1\}} \max_{p=1,2,...,P} \sum_{i\in I} (f_i - \mu^p a_i) y_i + \sum_{j\in J} \alpha^p b_j$

S.a.

(8-31) $\sum_{i\in I} a_i y_i \geq \sum_{j\in J} b_j$

onde (μ^p, α^p), para *p=1,...,P* são os vértices de *U*.

Seja $z' = \max_{p=1,2,...,P} \sum_{i\in I} (f_i - \mu^p a_i) y_i + \sum_{j\in J} \alpha^p b_j$,

então

$$z' \geq \sum_{i\in I} (f_i - \mu^p a_i) y_i + \sum_{j\in J} \alpha^p b_j, \quad p=1,2,...,P,$$

logo:

(8-32) $(PLCM\quad): \min \quad z = z'$

S.a.

(8-33) $z' \geq \sum_{i\in I} (f_i - \mu^p a_i) y_i + \sum_{j\in J} \alpha^p b_j, \qquad p=1,2,...,P$

(8-34) $\sum_{i\in I} a_i y_i \geq \sum_{j\in J} b_j$

(8-35) $y \in \{0,1\}^n$

8.2.1 Algoritmo de Benders para o Problema de Localização Capacitado

O problema mestre de Benders para o *PLC* pode ser modelado da seguinte forma:

(8-36) $(PLCM\ ) : \min\ z = z'$

S.a.

(8-37) $z' \geq \sum_{i \in I} (f_i - \mu^p a_i) y_i + \sum_{j \in J} \alpha^p b_j, \qquad p=1,2,...,P' \bullet P$

(8-38) $\sum_{i \in I} a_i y_i \geq \sum_{j \in J} b_j$

(8-39) $y \in \{0,1\}^n$

O algoritmo geral de Benders sofre algumas modificações. Essas são referentes ao resultado do dual, pois agora ele sempre será viável. O algoritmo é dado a seguir.

Passo 1: {inicialização}
Tomar uma instância de y, $\overline{y} \in \{0,1\}^n$
Se não existir tal instância ***Então***
PARE (*PLC* inviável)
$z^{sup} \leftarrow \infty$
$z^{inf} \leftarrow -\infty$

Passo 2:
Resolver *PLCD*
Se $z^{\sup} > \sum_{i \in I} (f_i - \mu^p a_i) y_i + \sum_{j \in J} \alpha^p b_j$ ***Então***
$z^{\sup} \leftarrow \sum_{i \in I} (f_i - \mu^p a_i) y_i + \sum_{j \in J} \alpha^p b_j$

Passo 3:
Acrescentar ao problema *PLCM* a restrição $z' \geq \sum_{i \in I} (f_i - \mu^p a_i) y_i + \sum_{j \in J} \alpha^p b_j$

Passo 4:
Resolver *PLCM* ($\hat{y}$ é a solução ótima de *PLCM*)
Se *PLCM* vazio ***Então***
PARE (*PLC* também será vazio)
$Z^{inf} \leftarrow V(PLCM)$

Passo 5:
Se $z^{inf} < z^{sup}$ ***Então***
Inicio
$\overline{y} \leftarrow \hat{y}$
Ir para o Passo 2
Fim
Senão
Inicio
$\overline{y} \leftarrow \hat{y}$
Resolver *PLC* $\overline{y}$ *(que fornecerá* $\overline{x}$ *)*
$(\overline{y}, \overline{x})$ é a solução ótima de *PLC*
Fim

8.3 Método Proposto

O método é bastante simples, seu objetivo é tentar minimizar o número de problemas mestres *PLCM* a serem resolvidos. Como conseqüência elimina-se a possibilidade de que *PLCM* torne-se muito complexo.

Para que se possa entender melhor o método que será proposto será analisado um pouco mais o que o método de Benders realiza. *PLCM* gera um limite inferior z^{inf} para o *PLC* e ao mesmo tempo indica um conjunto de facilidades $\overline{y}$ que fornecem uma solução viável para o problema, ou seja, $\sum_{i \in I} a_i y_i \geq \sum_{j \in J} b_j$. Assim sendo, resolve-se o *PLCD* para $\overline{y}$. Este passo fornece dois resultados. O primeiro é um limite superior z^{sup} para o *PLC*, pois *V(PLCD)=V(PLC* $\overline{y}$ *)*. Este resultado além de fornecer uma solução viável $(\overline{y}, \overline{x})$ para o problema, fornece juntamente com z^{inf} a possibilidade de verificação se a solução ótima foi alcançada. O segundo resultado é a geração de uma restrição para o *PLCM*. Essa restrição faz com que a solução gerada pelo *PLCM* forneça um limite inferior melhor ou igual aos gerados anteriormente. Também restringe a geração de um $\overline{y}$ igual a um gerado anteriormente, a não ser que $\overline{y}$ seja a solução ótima.

No novo método deseja-se melhorar o limite superior e gerar cortes que reduzam o espaço de busca do *PLCM*. O objetivo é resolver o *PLCM* mais rapidamente e obter a solução ótima resolvendo um número menor de *PLCM* do que o método de Benders apresentado na seção 8.2. Só o fato de se ter um número menor de *PLCM* para resolver implica que o *PLCM* será menos complexos que o método de Benders puro.

Antes de prosseguir será definido dois conjuntos $A=\{i|\ \overline{y}_i=1\}$ e $F=\{i|\ \overline{y}_i=0\}$, onde $\overline{y}$ é a solução gerada pelo *PLCM*. A idéia do método consiste em formar um conjunto $P=A \cup H$, onde $H \subset F$, e resolver o *PLC* considerando somente as facilidades contidas em *P*. Esse problema será chamado de PLC_p, o qual tem a seguinte formulação:

(8-40) $$(PLC_p) = \min \sum_{i\in P}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i$$

S.a

(8-41) $$\sum_{j\in J} x_{ij} \le a_i y_i, \quad \forall i \in P$$

(8-42) $$\sum_{i\in P} x_{ij} = b_j, \quad \forall j \in J$$

(8-43) $$x_{ij} \ge 0, \quad \forall i \in P, \forall j \in J$$

(8-44) $$y_i \in \{0,1\}, \quad \forall i \in P$$

É fácil de ver que $V(PLC_p) \le V(PLC_{\overline{y}})$, então teremos num limite superior melhor ou igual ao gerado pela solução obtida considerando $PLC_{\overline{y}}$. Outra conseqüência da resolução do PLC_p é que não é preciso resolver $PLC_{\overline{y}}$ tal que $A \subseteq P$. Isso porque, $V(PLC_p) \le V(PLC_{\overline{y}})$ para qualquer $PLC_{\overline{y}}$ tal que $A \subseteq P$.

Assim sendo, o *PLCM* não deve gerar soluções $\overline{y}$ tal que $A \subseteq P$. Para isso acrescenta-se o seguinte corte no *PLCM*

(8-45) $$\sum_{i\in I, i\notin P} y_i \ge 1,$$

ou seja, pelo menos uma facilidade que não pertença a *P*, na solução gerada por *PLCM*, deve estar aberta.

É possível observar que no método apresentado na seção 8.2, as únicas solução que não eram viáveis para o *PLCM*, eram as que violavam a restrição de viabilidade do *PLC*, isto é, $\sum_{i\in I} a_i y_i \ge \sum_{j\in J} b_j$. Com esse novo corte as soluções *y* tal que $y_i = 0\ \forall i \notin P$ não são viáveis, reduzindo assim o espaço de busca. Então o corte além de reduzir o espaço de busca do *PLCM,* pode fornecer um limite inferior maior.

Outro corte que será acrescentado é o de limite superior no *PLCM*, isto é,

(8-46) $V(PLC_p) \geq z'$,

com isso, tem-se mais uma maneira de fazer corte no *PLCM*, facilitando sua resolução.

A grande motivação para resolver o PLC_P é ter um método que forneça uma solução rápida para problemas de pequeno e médio porte, e para problemas em que a maioria das facilidades seja aberta na solução ótima. Este método é apresentado em Valiati [39].

No Capítulo 2 foi apresentado técnicas de *Add/Drop* nomeadas respectivamente de *O-Teste* e *C-Teste*. Nelas definem-se critérios de abertura e fechamento de facilidades. Caso algum critério seja válido a decisão tomada será ótima, ou seja, caso seja aberta ou fechada uma determinada facilidade este será o *status* que a facilidade terá na solução ótima. Na seção 4.3 define-se três heurísticas com intuito de acertar o *status* de uma facilidade na solução ótima (aberta ou fechada), são elas ***CMax-Heurística*** e ***CMin-Heurística O/C-Heurística***.

As técnicas de *Add/Drop* podem ser utilizadas para resolver o *PLCM*. Antes de começar a resolver o *PLCM* pode-se fixar alguns valores para algumas facilidades. Sejam K_0 e K_1 respectivamente os conjuntos de facilidades fechadas e abertas aplicando o *C-Teste* e o *O-Teste*, então sempre que for necessário resolver o *PLCM* as facilidades pertencentes a K_0 e K_1 serão fixadas a priori. As heurísticas podem ser utilizadas como critério para colocar os elementos de *F* em *P*.

É importante que a primeira instância $\overline{y}$ de *y* gerada seja de boa qualidade. Pois, assim sendo, tem-se uma probabilidade maior de se chegar à solução ótima mais rapidamente. Isso porque, se for gerada a primeira instância de boa qualidade, ter-se-á em mãos um bom limite superior, permitindo com que a solução do *PLCM* seja encontrada mais rapidamente.

Pode ser utilizada a primeira busca em profundidade do método apresentado no em Valiati [39] para encontrar o primeiro $\overline{y}$. Esta, além de fornecer uma boa

aproximação para problema, de forma rápida, fornece quais facilidades podem ser fixadas pelas técnicas de *Add/Drop*, ou seja, fornece os conjuntos K_0 e K_1, reduzindo assim o *PLCM*. Outra maneira de gerar a primeira instância para *y* é resolvendo o PLC_P, onde *P* e obtido através do *O-Teste*, *C-Teste* e da *CMax-Heurística*, *CMin-Heurística* e *O/C-Heurística*. Vale observar, que na construção de *P* deve-se tomar cuidado com a dimensão(não muito grande) e viabilidade do problema.

A restrição $\sum_{i\in I, i\notin P} y_i \geq 1$ faz com que nunca se repita uma solução $\overline{y}$ e possibilita a ocorrência de $z^{inf} > z^{sup}$. Nesse caso a solução é ótima. Contudo com a inserção da restrição $V(PLC_p) \geq z'$, juntamente com $\sum_{i\in I, i\notin P} y_i \geq 1$ pode-se ter *PLCM* inviável, nesse caso a z^{sup} representará a solução ótima, pois caso retirássemos as restrições $V(PLC_p) \geq z'$ teríamos $z^{inf} > z^{sup}$.

8.3.1 Algoritmo

Será apresentado a seguir um esboço do método que esta sendo proposto.

Passo 1: {inicialização}
Tomar uma instância de boa qualidade de y, $\overline{y} \in \{0,1\}^n$
Se não existir tal instância ***então***
PARE (*PLC* inviável)
$z^{sup} \leftarrow \infty$
$z^{inf} \leftarrow -\infty$

Passo 2:
Resolver *PLCD*
Se $z^{sup} > \sum_{i\in I} (f_i - \mu^p a_i)\overline{y}_i + \sum_{j\in J} \alpha^p b_j$ ***Então***
Inicio
$z^{sup} \leftarrow \sum_{i\in I} (f_i - \mu^p a_i)\overline{y}_i + \sum_{j\in J} \alpha^p b_j$
$y^* = \overline{y}$
Fim

Passo 3:
Acrescentar ao problema *PLCM* a restrição $z' \geq \sum_{i\in I} (f_i - \mu^p a_i) y_i + \sum_{j\in J} \alpha^p b_j$

Passo 4:
Resolver *PLCM* ($\hat{y}$ é a solução ótima de *PLCM*)
Se *PLCM* inviável ***Então***
Inicio
Resolver PLC_{y^*} *(que fornecerá* x^**)*
(y^*, x^*) é a solução ótima de *PLC*
PARE (solução ótima encontrada)
Fim
$Z^{inf} \leftarrow V(PLCM)$

Passo 5:
Se $z^{inf} < z^{sup}$ ***Então***
Inicio
$P = A \cup \{i \in F \mid i$ *tenha grande probabilidade de ser aberta na solução ótima}*
Resolver PLC_P { $\tilde{y}$ é a solução ótima de PLC_P*)*
Acrescenta ao *PLCM* $\sum_{i \in I, i \notin P} y_i \geq 1$
Se $V(PLC_P) < z^{sup}$ ***Então***
Inicio $y^* \leftarrow \tilde{y}$
$y^* \leftarrow \tilde{y}$
$z^{sup} = V(PLC_P)$
Acrescentar ao PLCM V (PLC_p) $\geq z'$
Fim
$\overline{y} \leftarrow \hat{y}$
Ir para o Passo 2
Fim
Senão
Inicio
Resolver PLC_{y^*} *(que fornecerá* x^**)*
(y^*, x^*) é a solução ótima de *PLC*
Fim

8.3.2 Resultados computacionais

Foi utilizado o *Branch and Bound* do XPRESS para resolver o *PLCM*. Não foi dado tanta importância em gerar uma primeira instância $\overline{y}$ de boa qualidade, tomou-se as facilidades com os menores índices tal que satisfizessem a restrição de viabilidade do *PLC*. Foi utilizado o $\Delta_i(I)$ como critério de inserção de facilidades em *P*. Mesmo não sendo esse um dos melhores critérios de aumento de *P* e não ter sido gerado a primeira instância $\overline{y}$ de boa qualidade acredita-se que será fornecido um bom parâmetro para a avaliação.

Foram utilizados todos os grupos de problemas gerados por Cornuejols, Sridharan e Thizy [20] . Contudo, praticamente, somente foi tomado os problemas de dimensões 16x25 (16 facilidades por 25 clientes), em alguns casos foi considerado os problemas de 25x25 e de 8x25. Os resultados computacionais são mostrados na Tabela 75, onde a média não considera o problema 1C1. O computador utilizado foi um Pentium III 700 MHz.

Problema	**Dimensão (nxm)**	**Proposto**	***Benders***
1A1	8x28	4.4	*1.7*
1A2		19.4	*5.9*
1A3		9.0	*5.0*
1A4		2.7	*0.7*
1A5		21.4	*6.5*
1B1	16x25	81.0	*4760.0*
1B2		12.1	*198.7*
1B3		16.4	*11025.9*
1B4		4.7	*1446.2*
1B5		13.5	*90981.7*
1C1	25x25	174027	*> 6 dias*
2B1	16x25	123.3	*1133.38*
2B2		282.1	*13686.6*
2B3		272.2	*2251.4*
2B4		180.2	*13742.3*
2B5		73.7	*430.5*
3B1	16x25	1093.6	*1471.7*
3B2		311.4	*694.7*
3B3		920.8	*1374.5*
3B4		30.2	*62.6*
3B5		306.2	*437.4*
5B1	16x25	37.2	*24.5*
5B2		61.3	*62.7*
5B3		136.0	*174.5*
5B4		89.9	*92.1*
5B5		51.6	*25.6*
XB1	16x25	15.0	*21.7*
XB2		12.9	*6.6*
XB3		15.7	*20.3*
XB4		9.3	*6.4*
XB5		5.7	*4.3*
XC5	25x25	479.5	*702.7*
XC2		761	*1103.6*
Média		***168,64***	***4428,61***

Tabela 75 – Resultados computacionais – solução exata (problemas teste de Thizy)

8.3.3 Análises e propostas

O método apresentado em Valiati [39] demonstrou ser muito eficiente para problemas de pequeno e médio porte. Embora não tenha sido testado em toda a sua abrangência, também demonstrou ser eficiente para problemas que abrem a maioria de suas facilidades na solução ótima. Assim sendo, provavelmente não se terá grandes problemas para resolver PLC_p. A maior dificuldade está na resolução dos *PLCM*. Por isso, foi introduzido os cortes (8−6) e (8−7) e tentado reduzir o limite superior resolvendo o PLC_P.

Os resultados computacionais mostraram o que estava-se esperando em relação ao método da seção 8.2, ou seja, demonstrou, no geral, ser bem mais eficiente. Contudo, o método está longe de ser algo eficiente. Mesmo com os cortes realizados, o problema *PLCM* é de difícil resolução. É importante observar que quanto maior a categoria dos problemas de Cornuejols, Sridharan e Thizy menor é o número de facilidades abertas na solução ótima. Talvez isso explique o fato do método proposto ser pior para a maioria dos problemas XB, contudo ele volta a ser melhor para os problemas XC. Outra coisa a observar é que o método proposto é pior para problemas pequenos os de dimensões 8x16.

Os resultados mostraram que o método chega à solução ótima, via limite superior, muito antes do que via limite inferior. Como só é garantido a otimalidade, do método, quando o limite inferior for igual ou ultrapassar o superior tem-se que esperar o crescimento do limite inferior. Porém, esse crescimento só se dá com a resolução do *PLCM*, o qual é de difícil resolução. Com isso o método fica ineficiente.

Uma das possíveis hipóteses de melhoria seria o desenvolvimento de um método mais eficiente para resolução do *PLCM*. Talvez seja possível encontrar algumas propriedades para o problema que o torne de "fácil" res olução.

Outra possibilidade, mais concreta, seria o desenvolvimento de um método que forneça um limite superior para *PLCM*. Seja *V'(PLCM)* o valor do limite superior para o *PLCM*, então, só é preciso resolver *PLCM*, de forma exata, quando o $V'(PLCM) \bullet z^{sup}$. Assim é possível se certificar da parada do método ou não.

Pode-se também não resolver o PLC_p a cada interação. O PLC_p poderia ser resolvido somente para conjuntos *P* com um determinado grau de distinção.

Também é possível utilizar a idéia da inclusão do PLC_P, ou seja, da resolução de subproblemas do *PLC*, em um método que explore a estrutura Primal e Dual do problema. Por exemplo, poderia ser utilizado Relaxação Lagrangeana nas restrições de capacidade **(1-2)** e ou nas restrições de demanda **(1-3)** para representar o Dual de PLC. No Capítulo 3 é apresentado a Relaxação Lagrangeana nas restrições de capacidade e nas restrições de demanda. Acrescentando a restrição de viabilidade do PLC $\sum_{i \in I} a_i y_i \geq \sum_{j \in J} b_j$ no problema dual, esse poderia funcionar com o problema mestre de Benders, com a vantagem de fornecer um limite inferior muito próximo ao ótimo, como visto em Cornuejols, Sridharan e Thizy [20]. Seja *DPLC* a estrutura dual de *PLC*, então o método teria a seguinte estrutura:

Tomar uma instância de boa qualidade de y, $\overline{y} \in \{0,1\}^n$
Enquanto limite Inferior < Limite Superior ***fazer***
Inicio
    Encontrar o conjunto *P*
    Resolver PLC_P
    Atualizar Limite Superior caso necessário
    Gerar o corte (8-6) em *DPLC*
    Resolver *DPLC*
    Atualizar Limite Inferior caso necessário
Fim

9 Bibliografia

1. Aikens C. H., "Facility Location Models for Distribution Planning", *European Journal of Operational Research*,v. 22 pg. 263-279, 1985.
2. Ahrens J. H. e Finke G., "Primal Transportation and Transshipment Algorithms", *Zeitshrift for Operations Research*, v.24 pg. 1-23, 1980
3. Ardal K., "Capacitated Facility location: Separation algorithms and computational experience", *Mathematical Programming*, v 81 pg. 149-175, 1998.
4. Barcelo J., e Casanovas J., "A heuristic Lagrangean Algorithm for the Capacitated Plant Location Problem", *European Journal of Operational Research*, v.15 pg.211-126, 1984.
5. Barcelo J., *Computational Experiments with Location Problems Using Automatic Constraint Generation*, Report RR85/06, Universitat Politecnica de Barcelona, Dept. d'Inve stigacion Operativa i Estadistica, Facultat d'Infomatica, Jordi Girona Salgado, 31, 08034 Barcelona, Spain, 1985.
6. Baker B. M., "Linear Relaxations of the Capacitated Warehouse Location Problem", *European Journal of Operational Research Society*, v. 33 pg. 475-479, 1982.
7. Baker B. M., "A Partial Dual Algorithm for the Capacitated Warehouse Location Problem", *Journal of Operational Research*, v.23 pg.48-56, 1986.
8. Baker B. M. e Baía A. P., *Improvements to Lagrangean Relaxation Algorithms for the Capacitated Warehouse Location Problem,* IFORS 93 Lisboa, 1993
9. Barahona F., Chudak F. A.,"Near-optimal solutions to large-scale facility location problems", *Discrete Optimization*, v. 2, pg 35-50, 2005.
10. Bartezzaghi E., Colorni A. e Palermo P. C., "A Tree Search Algorithm for the Plant Location Problems", *European Journal of Operational Research*, v. 7, pg 371-379, 1981.
11. Beasley J. E., "An Algorithm for Solving Large Capacited Warehouse Location Problems", *European Journal of Operational Research*, v.33 pg.314-325, 1988.
12. Beasley J. E., "Lagrangean Heuristics for Location Problems", *European Journal of Operational Research*, v.65 pg.383-399, 1993.
13. Benders J. F., "Partitioning Procedures for Solving Mixed Variables Programming Problems", *Numerische Mathematik*, v. 4 pg. 238-252, 1962.

14. Bitran G. R., Chandru V., Sempolinski D. E. e Shapiro J. F., "Inverse Optimization: an Application to the Capacitated Plant Location Problem", *Management Science* v. 27, n. 10, pg. 1120-1141, 1981
15. Bornstein C. T. e Azlán H. B., "The Use of Reduction Tests and Simulated Annealing for the Capacitated Plant Location Problem", *Location Science*, v.6, pg.67-81, 1998.
16. Bornstein C. T. e Mateus G. R., "Dominance Criteria for the Capacitated Warehouse Location Problem", *Jornal of the Operational Research Society*, v.42 pg.145-149, 1991.
17. Campêlo Neto M. B., *Testes de Redução e Heurística ADD/DROP para o Problema de Localização Capacitado*, tese M.Sc, Engenharia de Sistemas e Computação COPPE/UFRJ, Rio de Janeiro, RJ, Brasil,1993.
18. Chvátal V., *Linear Programming*, 1ª edição, New York, W. H. Freeman and Company, 1983.
19. Compernolle V. D., Speech Recognition: A New Perspective For Applications, ESAT Kard. Mercierlaan 94 3001 Heverlee, atholieke Universiteit Leuven, Belgium.
20. Cornuejols G., Sridharan R. e Thizy J. M., "A Comparison of Heuristics and Relaxations for the Capacitated Plant Location Problem", *European Journal of Operational Research*, v.50, pg. 280-297, 1991.
21. Dantzig G. B., *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.
22. Domschke W. e Drexl A., "Location and Layout Planning: An International Bibliography", *Lecture Notes in Economics and Mathematical Systems*, v. 238, Springer-Verlag, Berlin,1985.
23. Domschke W. e Drexl A., "ADD-Heuristics - Starting Procedures for Capacitated Plant Location Models", *European Journal of Operational Research*, v. 21, pg. 47-53, 1985.
24. Fox G. C., "Prospects For Parallel Computing", *Northeast Architectures Center*, College Place Syracuse University, New York 13244-4110, 1992.
25. Francis L., McGinnis L. F. e White J. A., *Locational Analysis,* European Journal of Operational Research, v. 12, pg.220-252, 1983.
26. Goldbarg M. C. e Luna H. P. L., *Otimização Combinatória e Programação Linear*, 1ª edição, Rio de Janeiro, editora Campus, 2000.

27. Glover F., Klingman D. e Stutz J., "Augmented threaded index method for network optimization", *INFO, Canad. J. Operational Res. and Information Processing*, v 12, 293-298, 1974
28. Guignard-Spielberg M. e Kim S., *A Strong Lagrangean Relaxation for Capacitated Plant Location Problem*, Report 56, Department of Statistics, the Wharton School, University of Pennsylvania, PA, 1983.
29. Jacobsen S. K., "On the use of tree-indexing methods in transportation algorithms", *European Journal of Operational Research*, pag 54-65, 1978.
30. Jacobsen S. K., "Heuristics for the Capacitated Plant Location Model", *European Journal of Operational Research*, v.12 pg. 253-261, 1983.
31. Hopcrof J. E., Ullman J. D. e Motwani R., *Introdução à Teoria de Autômatos Linguagens e Computação,* 2ª edição, Rio de Janeiro, editora Campus, 2003.
32. Kühn A. A. e Hamburger M. J., "A Heuristic Program for Locating Warehouses", *Mgmt Sci*, v.9 pg. 643-666, 1963.
33. Muller G., Jones P. C., Lowweeer T. J., 'Specially Structured Uncapacitated Facility Location Problems", *Operation Research*, v. 43, n. 4, pg. 661-669, July-August, 1995.
34. Puccini A. L. e Pizzolato N. D., *Programação Linear*, 2ª edição, editora Livros Técnicos e Científicos, Rio de Janeiro, 1991.
35. Ahuja R., Magnanti T. L. e Orlin J. B., *Network Flows Theory, Algorithms and Applications*, 2ª edição, editora Prentice-Hall, New Jersey, 1993.
36. Senne E. L. F., LORENA L. N. "A Lagrangean/Surrogate approach to facility location problems", EURO-TIMS Congress, Barcelona, 1997.
37. Sridharan R., *A Survey of the Capacitated Plant Location Problem,* Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University, 1984.
38. Szwarcfiter J. L. e Markenzon L., *Estrutura de dados e seus algoritmos*, 2ª edição, Rio de janeiro, editora LTC, 19.
39. Valiati D. , *Testes de Redução e Algoritmo Branch and Bound para o Problema de Localização Capacitado,* tese M.Sc, Engenharia de Sistemas e Computação - COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1999.
40. Valiati D. , *Métodos para resolução do Problema de Localização Capacitado,* exame de qualificação da tese de D.Sc, Engenharia de Sistemas e Computação - COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2003.

41. Valiati D. e Bornstein C. T., "A Branch and Bound Algorithm for the Capacitated Plant Location Problem Based on ADD/DROP Procedures", *ISOLDE VII - VIIIth International Symposium on Locational Decisions*, Coimbra & Estoril, Portugal, June 23-29,1999.
42. Valiati D. e Bornstein C. T., "Um Algoritmo Branch and Bound para o Problema de Localização Capacitado", *XXI CNMAC*, Santos, SP, Brasil, setembro, 1999.
43. Valiati D. e Bornstein C. T., "Testes de Redução para o Problema de Localização Capacitado", *XXXII SBPO*, Viçosa, MG, Brasil, outubro, 2000.
44. Valiati D. e Bornstein C. T., "Método Exato para Resolver o Problema de Localização Capacitado Utilizando Testes de Redução e Relaxação Lagrangeana", *XXXII SBPO*, Campos do Jordão, SP, Brasil, novembro, 2001.
45. Van Roy T. J., "A Cross Decomposition Algorithm for the Capacitated Facility Location"*, Operations Research*, v.34, pg.145-163, 1986.
46. Wolsey L.A., "Fundamental Properties of Certain Discrete Location Problem", in Thisse J.-F. e Zoller H. G. (eds). *Location Analysis of Public Facilities*, North-Holland pg.331-355, 1983.