UM ALGORITMO GRASP HÍBRIDO PARA O PROBLEMA DE

LOCALIZAÇÃO CAPACITADA DE CUSTO FIXO

Ronaldo Silva Virginio Filho

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

Aprovada por:

Prof. Adilson Elias Xavier, D.Sc

Prof. Marcos José Negreiros Gomes, D.Sc

Prof. Nélio Domingues Pizzolato, D.Sc.

Prof. Nelson Maculan Filho, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

FEVEREIRO DE 2007

VIRGINIO FILHO, RONALDO SILVA

Um Algoritmo GRASP Híbrido para o Problema de Localização Capacitada de Custo Fixo [Rio de Janeiro] 2007

VI, 73, 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e computação, 2007)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Problema de Localização Capacitada
2. GRASP
3. Problema de Transporte
4. Otimização Combinatória
5. Meta-heurística

I. COPPE/UFRJ II. Título (série).

Agradecimentos

Agradeço aos meus familiares pelo apoio e incentivo recebido durante toda minha vida, o que me proporcionou chegar até este momento.

Tenho um agradecimento especial a fazer a dois grandes amigos: Yuri e Pedro, pelo apoio e generosidade que tiveram comigo, sem os quais não teria conseguido seguir no mestrado.

Ao meu orientador e amigo, Marcos Negreiros, não tenho palavras para agradecer seu empenho, incentivo, tempo e paciência gastos comigo, que vêm desde a época de graduação na UECE. Muito obrigado.

Ao meu orientador Adilson Xavier um agradecimento especial por sua dedicação, empenho e por ter aceitado dividir minha orientação à distância com o Professor Marcos Negreiros.

Agradeço a paciência, incentivo e apoio de minha namorada e amigos durantes os dias em que me encontrei tomado pelo estresse, cansaço e mau humor.

Ao Programa de Engenharia de Sistema e Computação – PESC da COPPE/UFRJ por ter me recebido.

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

UM ALGORITMO GRASP HÍBRIDO PARA O PROBLEMA DE LOCALIZAÇÃO CAPACITADA DE CUSTO FIXO

Ronaldo Silva Virginio Filho

Fevereiro/2007

Orientador: Adilson Elias Xavier

Programa: Engenharia de Sistemas e Computação

Apresentamos o desenvolvimento de um algoritmo para o Problema de Localização Capacitada de Custo Fixo (PLC). No PLC a questão central é localizar um conjunto de facilidades (uma ou mais), minimizando o custo global de localizá-las. O algoritmo implementa uma meta-heurística gulosa, o GRASP, para chegar a um conjunto solução de facilidades localizadas e no final, com o objetivo de tentar melhorar a solução, são realizadas trocas entre os elementos do conjunto solução e as facilidades candidatas que não fazem parte da solução final. Avaliamos duas estratégias GRASP, usando Alfa Fixo e Alfa Variado. Esta última estratégia, a denominamos de GRASP Adaptativo (o mesmo que reativo). Para testar os resultados foi utilizada a biblioteca de problemas teste do PLC a OR-Library, onde obtivemos resultados que se mantiveram com GAP em média abaixo de 0,5% da solução ótima.

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

AN HYBRID ALGORITHM FOR THE FIXED COST CAPACITATED LOCATION PROBLEM

Ronaldo Silva Virginio Filho

February/2007

Advisor: Adilson Elias Xavier

Department: Computing and Systems Engineering

In this work a hybrid algorithm for the Fixed Cost Capacitated Location Problem (FCCLP), whose objective is to locate a number of capacitated facilities that cover the total demand of the customers in a geographical region. We use a hybrid algorithm based on the transportation problem and a meta-heuristic GRASP to encompass the solution, in order to identify a solution set for the facilities on addition, exchanges among facilities belonging to the solution set and others non candidate solution are made in the search for the final solution. We evaluate two strategies of GRASP, using fixed alpha and a range of alpha (reactive GRASP). To evaluate the results the OR-Library was used for the instances of the FCCLP, where as result we found an average gap from the optimum solution below 0,5% using our method.

1. INTRODUÇÃO

1.1. Apresentação do problema

O Problema de Localização Capacitada (PLC) busca definir um conjunto de facilidades que atenderão um outro conjunto de clientes geograficamente dispersos, considerando a minimização global dos custos relativos à demanda-distância entre as facilidades e os clientes, e a soma dos custos fixos das facilidades utilizadas.

Os problemas de localização vêem sendo muito estudados pela literatura, e bons resultados foram encontrados e publicados em diversos trabalhos. Considerando aqueles ligados à meta-heurística GRASP com problemas similares de localização, bons resultados também foram reportados, o que nos induziu a realizar a metodologia que aqui apresentamos, [RESENDE, 2001], [RESENDE e WERNEK, 2004], [NEGREIROS, CARVALHO & MACULAN, 2005].

Estes problemas podem ser identificados em redes que são formadas por pontos (nós ou vértices) e linhas (arcos ou arestas) e podem descrever de forma natural vias públicas, conexões de água, telefonia, etc. O termo facilidade pode ser substituído por fábricas, depósitos, escolas, antenas, etc. Em geral, várias facilidades serão localizadas, que por sua vez, serão alocados aos seus clientes. Desta forma tais problemas são também conhecidos como problemas de localização-alocação. A maioria dos problemas de localização de facilidades é considerada de difícil solução, alguns desses problemas pertencem à classe NP-difícil, [ARARAKI, 2002].

1.2. Objetivos do trabalho

Este trabalho tem como objetivo a definição de um algoritmo híbrido para o problema de localização capacitada de custo fixo (PLC) utilizando-se a meta-heurística GRASP em conjunto com algoritmos exatos do problema de transporte, para obtenção de um conjunto solução de facilidades localizadas.

O trabalho proposto apresenta um algoritmo para resolução do PLC que se divide basicamente em duas fases: a primeira encontra uma solução inicial para o PLC e

a segunda busca através de dois métodos diferentes de trocas melhorarem a solução inicial. Um conjunto de problemas teste foi exaustivamente utilizado com objetivo de testar a eficiência do algoritmo proposto. [OR-LIBRARY], [CORNUEJOLS, SRIDHARAN & THIZY, 1991].

O algoritmo tem a característica de ser extremamente rápido na sua execução, além de fornecer soluções de muito boas.

Com o objetivo de verificar o comportamento do método proposto, de modo a garantir a reprodutividade dos ensaios aqui apresentados, foi realizada uma análise da robustez do método, retratando o melhor caso, caso médio e pior caso do ponto de vista da variação do GAP encontrado na solução.

1.3. Estrutura do trabalho

O trabalho está organizado do seguinte modo, na seção 2 é descrito o modelo matemático do problema, sua definição e algumas situações em que pode ser aplicado. Alguns métodos desenvolvidos e bastante utilizados na resolução de problemas de localização como o ADD, DROP, Relaxação Lagrangeana e métodos aproximativos são comentados no trabalho.

Na seção 3, é feita uma análise da meta-heurística GRASP e mostrada a diferença de implementações entre GRASP Simples e Reativo (Adaptativo). O problema de transporte é incluído, pois é considerado como um sub-problema a ser resolvido dentro do modelo de solução meta-heurística desenvolvida. Na seção 4 o algoritmo proposto é descrito em suas duas fases: fase de solução do PLC e fase de trocas. Na seção 5 são apresentados os resultados computacionais obtidos na resolução dos problemas teste da OR-Library para o PLC dos problemas teste apresentados por Thizy. Na seção 6 fazemos à conclusão do trabalho e na seção 7 é apresentada à bibliografia do trabalho.

2. PROBLEMA DE LOCALIZAÇÃO CAPACITADA DE CUSTO FIXO

2.1. Modelo

O modelo para o PLC pode ser colocado genericamente do seguinte modo, [DASKIN, 1995]:

$$\text{(PLC) Minimizar} \quad \sum_{j=1}^{m} f_j y_j + \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_{ij} c_{ij} x_{ij} \qquad [2.1]$$

sujeito a,

$$\sum_{j=1}^{m} x_{ij} = 1, \; i = (1,...,n) \qquad [2.2]$$

$$x_{ij} \leq y_j, \; \forall (i,j) \in \{(1,...,n),(1,...,m)\} \qquad [2.3]$$

$$\sum_{i=1}^{n} q_i x_{ij} \leq Q_j y_j, \; \forall j \qquad [2.4]$$

$$x_{ij} \geq 0, \forall (i,j) \in \{(1,...,n),(1,...,m)\} \qquad [2.5]$$

$$y_j \in \{0,1\}, \; \forall j \in (1,...,m) \qquad [2.6]$$

Onde, são especificados os seguintes parâmetros:

f_j é o custo fixo de instalar uma facilidade (depósito, fábrica) no local j;

q_i - é a demanda do cliente *i*;

d_{ij} - é a distância mínima do cliente *i* à facilidade *j*;

α_{ij} é a taxa de transporte ($/unidade de carga) por unidade de demanda (q) para envio de produto do cliente *i* à facilidade *j* ;

Q_j é a capacidade de uma facilidade candidata;

c_{ij} é o custo total em distância(d)×demanda(q) (d×q) para enviar um produto do cliente *i* à facilidade *j, ou seja:* $c_{ij} = d_{ij}.q_i$;

E as seguintes variáveis:

x_{ij} – Fração da demanda do cliente *i* que é atendida pela facilidade *j*;

$$y_j = \begin{cases} 1, & \text{Se a facilidade j é instalada;} \\ 0, & \text{Caso Contrário.} \end{cases}$$

No modelo do PLC tem-se na função objetivo [2.1] a necessidade de encontrar a solução de menor custo de localização e de taxa-custo global do transporte, que atende às restrições de atribuição única da facilidade ao cliente [2.2]. As restrições [2.3] consideram que a demanda no cliente *i* não pode ser atribuída à facilidade *j* a menos que *j* esteja instalada. A restrição [2.4] assegura que a demanda do cliente i não está designada à facilidade candidata a localização, se esta facilidade não for selecionada, e as restrições [2.5] e [2.6] garantem a necessidade da seleção única entre clientes-facilidades e facilidades candidatas (restrições de integralidade e não negatividade). O modelo PLC acima proposto é conhecido como a formulação forte do problema e pode ser visto em [DASKIN, 1995] e [LABBÉ & LOUVEAUX, 1997].

O problema de Localização Capacitada de Custo Fixo busca definir um conjunto de facilidades (de um conjunto de candidatas) que atenderão um conjunto de clientes geograficamente dispersos, considerando a minimização global dos custos relativos à demanda-distância entre as facilidades e os clientes, e a soma dos custos fixos das facilidades utilizadas ou instaladas.

O PLC pode ser aplicado em diversas áreas, tais como distribuição geográfica de escolas em uma cidade para que possa ser atendido o maior número de crianças com o menor custo, localização de antenas transmissoras de sinal para celular, distribuição de agências bancárias pela cidade, localização de ambulâncias entre outras. [BARCELOS, PIZZOLATO & LORENA, 2004], [PIZZOLATO & FRAGA, 1997].

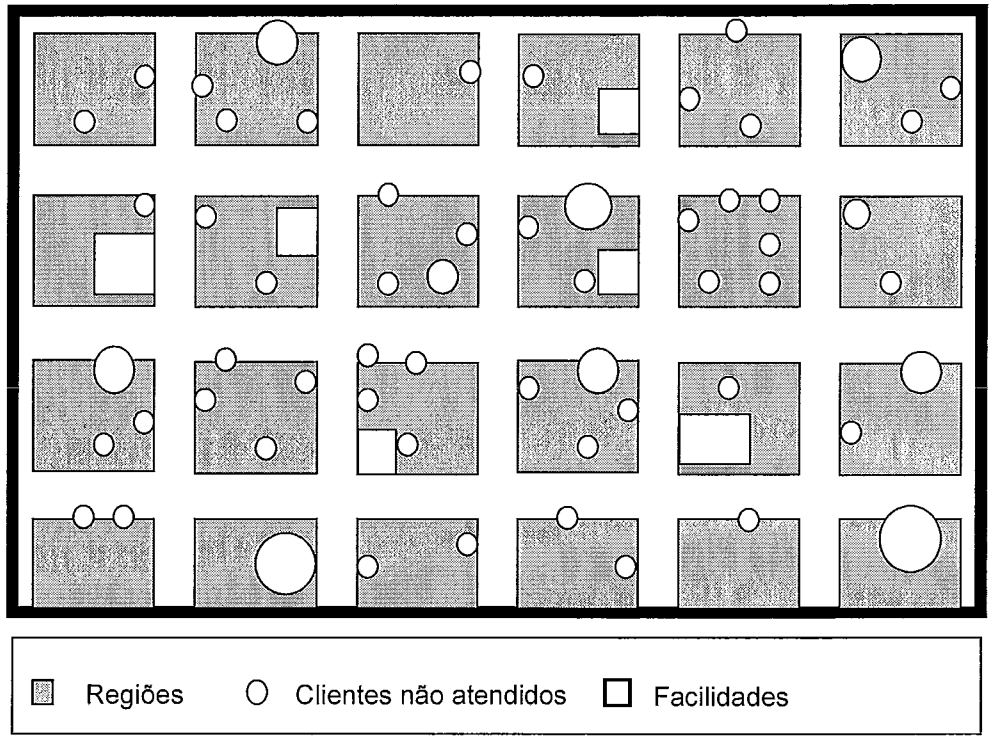

Figura 1: a) Exemplo de uma configuração inicial para o problema de localização capacitada de custo fixo.

A configuração da figura 1.a) mostra um conjunto de facilidades candidatas a atender a demanda dos clientes espalhados pelas várias regiões de forma a minimizar o custo desta alocação. Na figura, as bolas e os retângulos representam os clientes e as facilidades candidatas respectivamente, sendo a representação diferenciada dos tamanhos dos objetos uma ilustração das diferentes capacidades das facilidades e das demandas que os clientes podem assumir.

Nas figuras 1.b) – 1.e) são ilustradas as seqüências de facilidades que são abertas ao longo de um processo de solução do problema. Na figura 1.e) todos os clientes foram atendidos pelas facilidades que entraram para o conjunto de localizadas. As facilidades candidatas não atendidas permaneceram de branco, enquanto as facilidades localizadas e seus respectivos clientes atendidos estão marcados com outra cor.

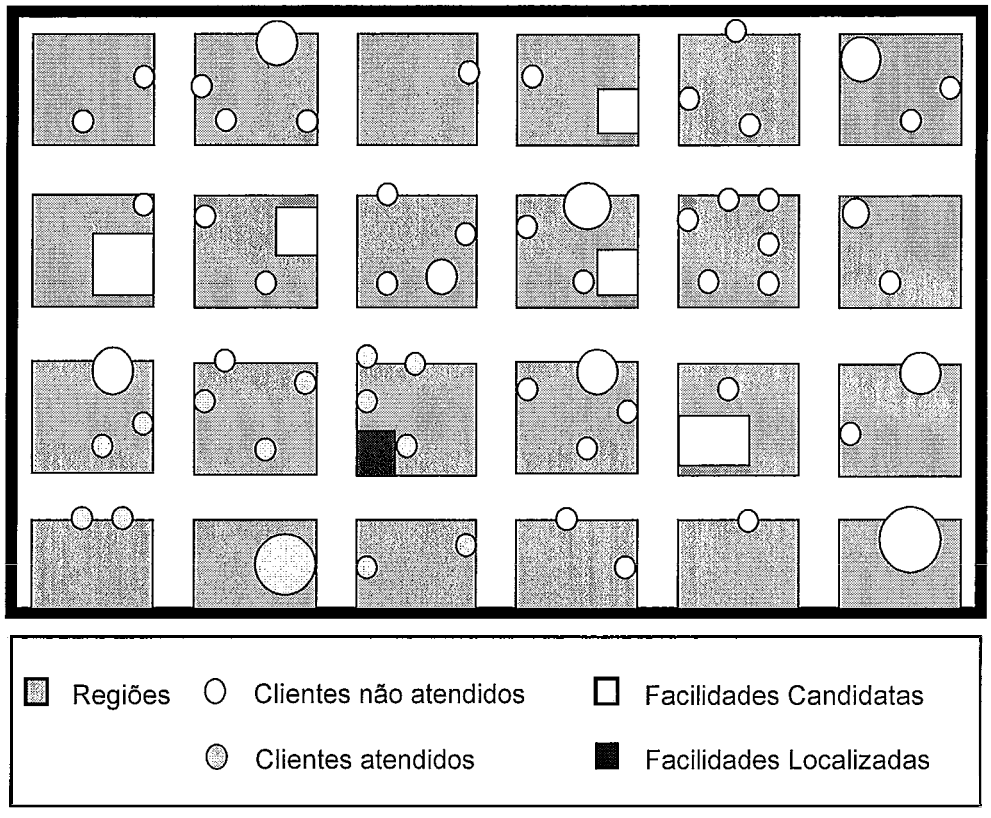

Figura 1: b) Estado após a primeira facilidade localizada.

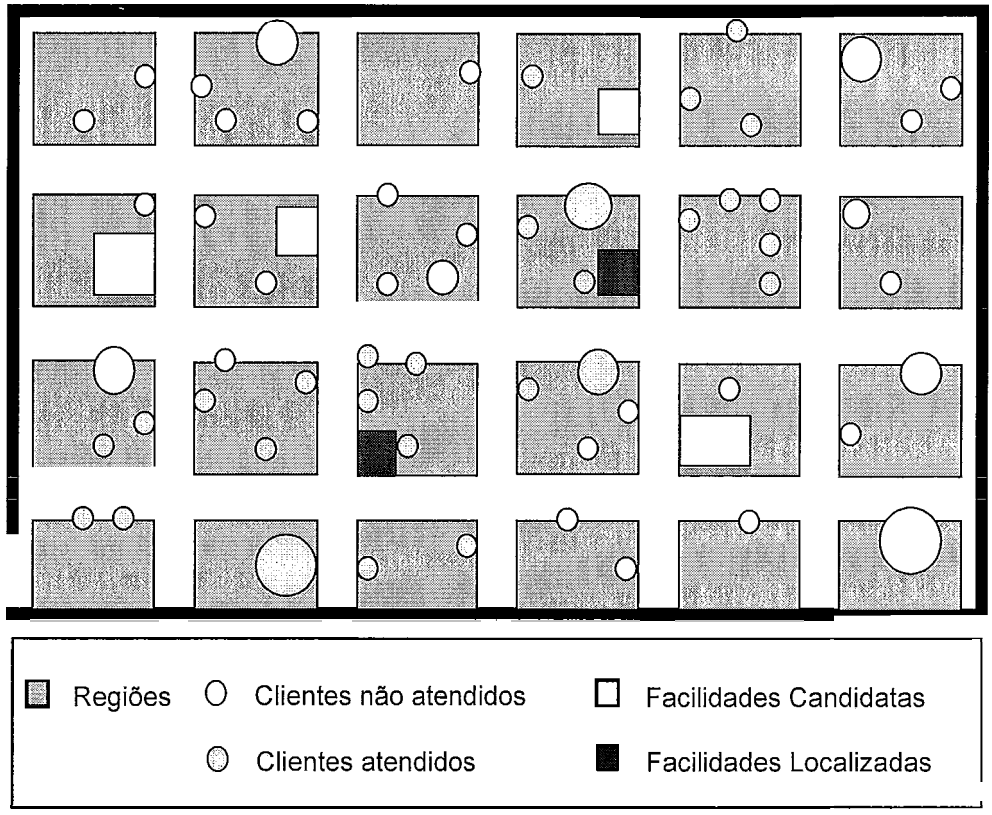

Figura 1: c) Estado após a segunda facilidade localizada.

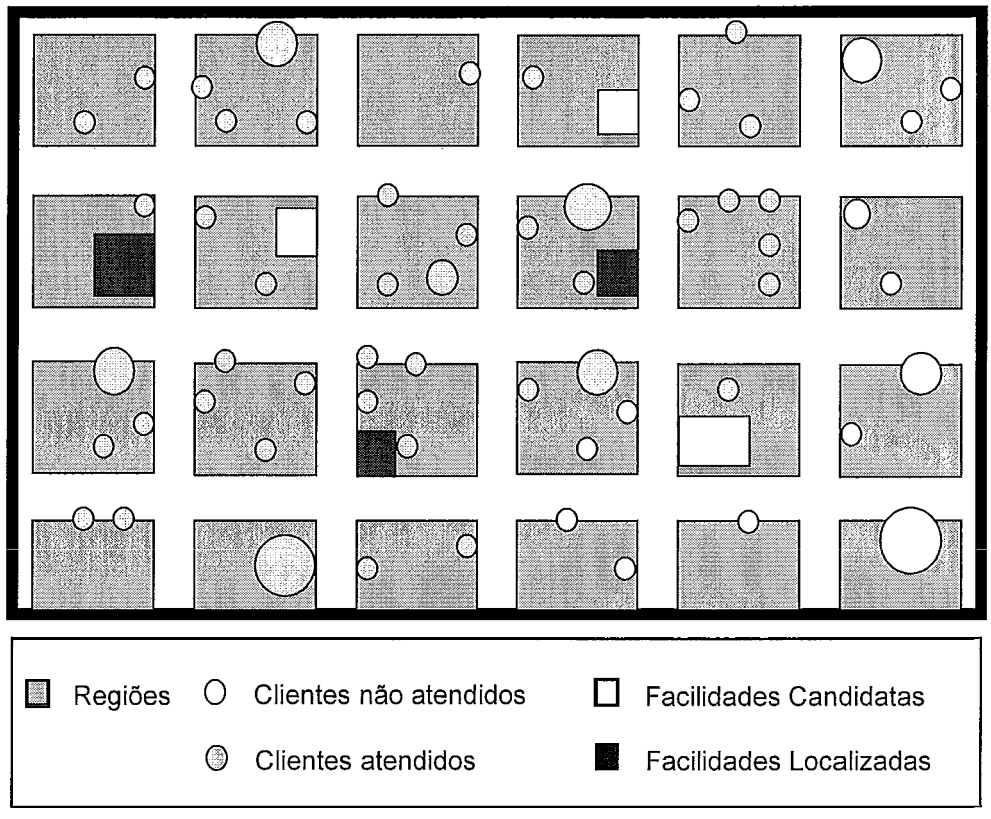

Figura 1: d) Estado após a terceira facilidade localizada.

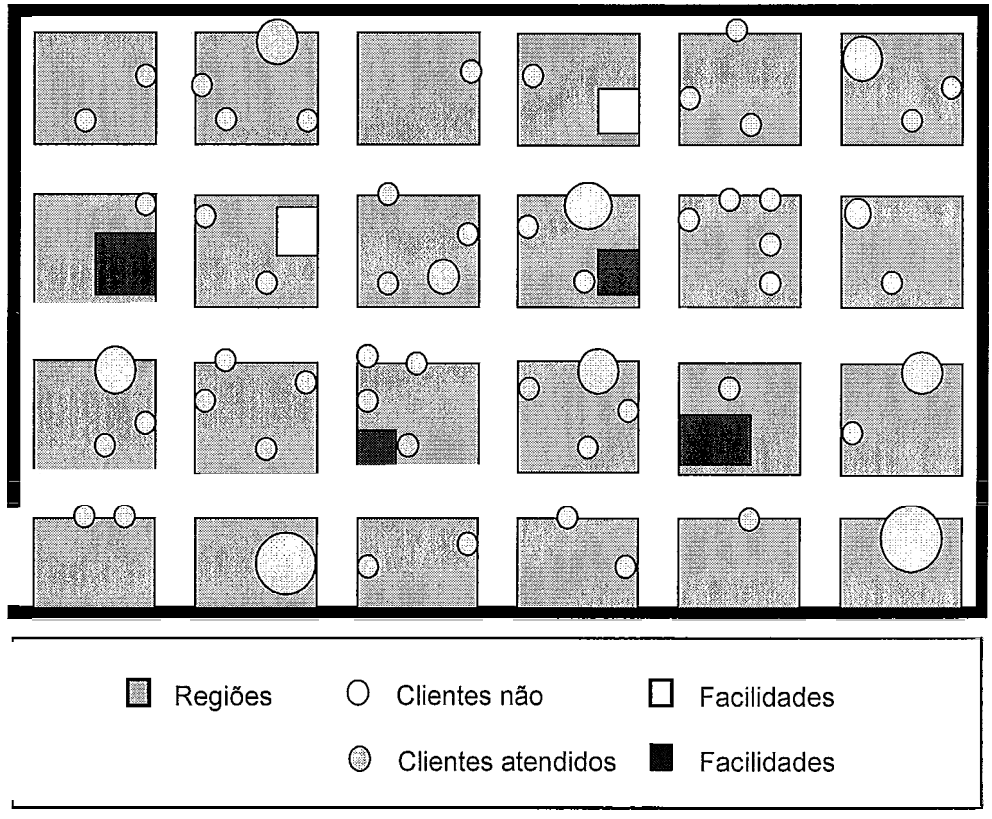

Figura 1: e) Estado final e solução do PLCF.

Este problema é bastante estudado pela literatura, tendo merecido diversos trabalhos sobre sua solução via métodos exatos e heurísticos, [NEGREIROS, CARVALHO & MACULAN, 2005]. Este problema é dito ser do tipo NP-Árduo, uma vez que somente problemas teste de pequeno porte $(n+m)$ podem ser resolvidas em tempo aceitável. Problemas teste de grande porte (500 × 1000) também foram resolvidas, [BEASLEY, 1988].

2.2. Métodos utilizados na resolução do PLC

A literatura apresenta alguns importantes métodos para resolução do PLC, que podem ser classificados em exatos e heurísticos. Dentre os métodos heurísticos, os procedimentos construtivos generalizados para este problema são o ADD e DROP. Ambos são fundamentados em heurísticas gulosas. Sumariamente, são apresentados:

2.2.1. ADD [KUEHN & HAMBURGER, 1963]

A construção da solução para o PLC utilizando o método ADD é efetuada da seguinte maneira: c_{tot} (Custo de localização) inicialmente recebe um valor muito elevado. Seja F um conjunto de facilidades candidatas a pertencer ao conjunto L, sendo L o conjunto das facilidades localizadas. Uma facilidade de F é escolhida para entrar no conjunto L em cada iteração.

Verifica-se se $c_{tot} > c_{it}$ (Custo da iteração), caso verdade c_{tot} recebe o valor de c_{it} e a facilidade escolhida é removida do conjunto F, passando a pertencer ao conjunto L.

Uma nova iteração é realizada enquanto existir facilidades não verificadas no conjunto F e $c_{tot} > c_{it}$, caso falso, as iterações param e o resultado final é c_{tot} e L.

2.2.2. DROP [FELDMAN, LEHRER & RAY, 1966]

A construção da solução para o PLC utilizando o método DROP funciona de maneira semelhante ao método ADD: c_{tot} (Custo de localização) inicialmente recebe o custo para alocação da demanda dos clientes com todas as facilidades candidatas pertencendo ao conjunto de localizadas. Tomando F agora como o conjunto de facilidades removidas do conjunto L, e sendo L o conjunto das facilidades localizadas. Uma facilidade de L é escolhida para sair do conjunto L em cada iteração. Verifica-se se $c_{tot} > c_{it}$ (Custo da iteração), caso verdade c_{tot} recebe o valor de c_{it} e a facilidade escolhida de L é removida do conjunto L e passa a pertencer ao conjunto F. Uma nova iteração é realizada enquanto $|L| \geq 1$ (Número de elementos no conjunto L for maior ou igual a 1) e $c_{tot} > c_{it}$, caso falso, as iterações param e o resultado final é dado pelo custo c_{tot} e o conjunto de facilidades L.

Já para os métodos exatos, dentre os mais importantes tem-se:

2.2.3. [BEASLEY, 1988] Descreve um algoritmo exato para solução do problema de localização capacitada de custo fixo utilizando Relaxação Lagrangeana.

2.2.4. [DASKIN, 1995] Também demonstra um método exato para o PLC baseado em relaxação Lagrangeana e Branch & Bound.

2.2.5. [GALVÃO, 2004] Descreve diferentes modelos e métodos para o problema de localização utilizando algoritmos exatos e heurísticos, entre eles: problemas de localização não-capacitada, problemas de localização com cobertura, modelos de localização hierárquicos e modelos de localização probabilísticos.

A literatura também vem dando destaque aos métodos aproximativos.

2.2.6. Métodos aproximativos

Dado um problema de minimização, um algoritmo é dito ρ-aproximativo, se para qualquer instância do problema, o algoritmo executa em tempo polinomial e a solução de saída tem um custo quando muito $\rho \geq 1$ vezes o custo mínimo, onde ρ é chamada performance garantida ou taxa de aproximação do algoritmo. [ZHANG, CHEN & YE, 2005].

Desde os trabalhos de [SHMOYS, TARDOS & ARDAL, 1997], desenvolvendo algoritmos para o problema de localização de facilidades, que uma considerável atenção nos últimos anos tem sido lançada sobre estes métodos.

A melhor taxa de aproximação mais correntemente conhecida para o problema de localização não-capacitada é 1,517 que se deve a [MAHDIAN, YE & ZHANG, 2002], [MAHDIAN, YE & ZHANG, 2003], o qual é um caso especial do PLC com todas as capacidades $u_i = \infty$.

Todos os algoritmos de aproximação conhecidos para o PLC com garantia de performance limitada usam técnicas baseadas em busca local. [KORUPULO, PLAXTON & RAJARAMAN, 1998] foram os primeiros a mostrar que o algoritmo de busca local proposto por [Kuehn & Hamburger, 1963] tem uma taxa de aproximação igual a 8 para o caso onde todas as capacidades são uniformes, isto é $u_i = u \ \forall i \in f$.

Quando as capacidades não são uniformes, [PÁL, TARDOS & WEXLER, 2001] têm apresentado um algoritmo de busca local com performance garantida entre 4 e 8,53. Seu algoritmo consiste dos três tipos seguintes de operações de melhora: abrir uma facilidade; abrir uma facilidade e fechar uma ou mais facilidades; fechar uma facilidade e abrir uma ou mais facilidades.

A introdução de um novo tipo de operações de troca por [ZHANG, CHEN & YE, 2005], chamada *multi-exchange*, a qual possibilita serem feitas múltiplas trocas simultaneamente, isto é, fecha muitas facilidades e abre muitas facilidades. Este novo tipo de operação habilita a aproximação do PLC com um fator de $3+2\sqrt{2}+\varepsilon$ para uma dada constante qualquer $\varepsilon > 0$. A taxa de aproximação do algoritmo *multi-exchange* está de acordo com o que tem sido provado por [Chudak & Williamson, 1999] para o caso da capacidade uniforme.

3. PROPOSTA DE UM ALGORITMO GRASP HÍBRIDO PARA O PLC

O modelo geral da meta-heurística GRASP considera que um algoritmo construtivo guloso deve ser usado para amostrar um conjunto de possíveis candidatos a pertencer à solução gulosa a cada passo. A seguir, com uma solução gulosa na mão, a meta-heurística sugere trocas locais para melhorar a solução obtida [RESENDE, 2001]. Para resolver as instâncias do PLC, foi utilizada a formulação geral baseada no problema de transporte, que garante que a solução do problema é viável do ponto de vista do transporte, a seguir construída uma estratégia gulosa que permite gerar soluções viáveis para o PLC. Após um número de execuções e com a melhor solução gulosa entre as amostradas, são feitas trocas locais que conduzem ao resultado final do processo.

Nessa linha de raciocínio, são apresentadas as seguintes metodologias.

3.1. O problema de transporte clássico

O problema de transporte pode ser apresentado de forma genérica da seguinte forma:

Minimizar:

$$Z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad [3.1]$$

Sujeito a

$$\sum_{j=1}^{n} x_{ij} \leq a_i , \qquad i = (1,2,...,m) \qquad [3.2]$$

$$\sum_{i=1}^{m} x_{ij} \geq b_J , \qquad j = (1,2,...,n) \qquad [3.3]$$

$$x_{ij} \geq 0 , \qquad i = (1,2,...,m); j = (1,2,...,n) \qquad [3.4]$$

Onde,

c_{ij} = custo de transporte entre a fonte i e o destino/sumidouro j;

x_{ij} = total a ser distribuído da fonte i até o destino/sumidouro j;

a_i = Total produzido pela fonte i;

b_j = total a ser armazenado pelo destino/sumidouro j.

Para que o problema tenha solução, ele deve estar balanceado, ou seja, deve-se ter o total armazenado igual ao total da produção. Isso pode ser definido pela equação [3.5].

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \qquad [3.5]$$

O fato de o problema estar balanceado, faz com que uma das restrições seja redundante. Isto significa que o problema se reduzirá a *(m + n* - 1) restrições e *(m* × n) variáveis de decisão.

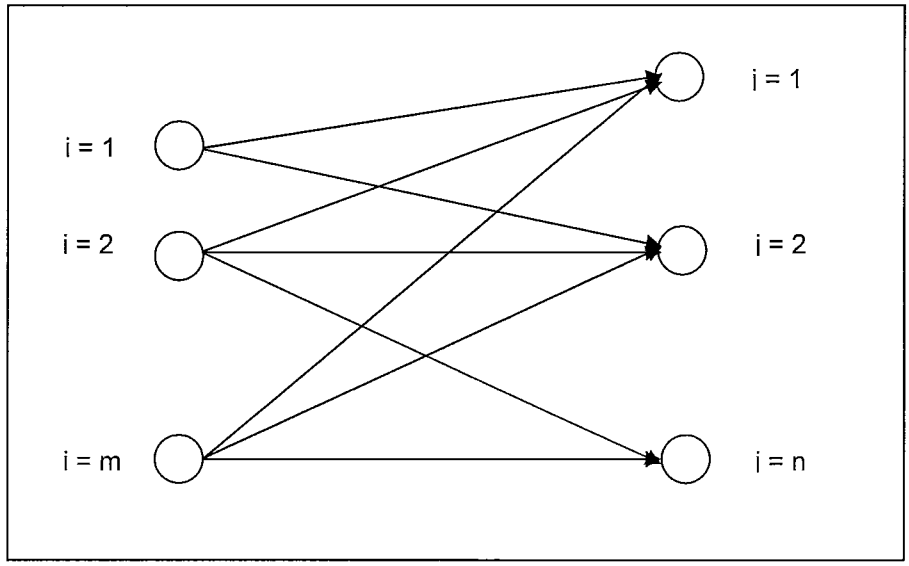

Figura 2: Fluxo de transporte em uma rede.

Como se trata de um problema típico e clássico de programação linear, ele pode ser resolvido utilizando-se o método Simplex convencional, [DANTZIG, 1963]. Entretanto, técnicas específicas para este tipo de problema podem resolvê-lo de forma mais rápida que o Simplex. Neste projeto foi utilizado o método de Ford-Fulkerson (Algoritmo 1), [SYSLO, DEO e KOWALIK, 1988].

Algumas das dificuldades do Problema de Transporte aparecem quando não há balanceamento entre oferta e demanda. Caso isto ocorra, o problema não pode ser resolvido da maneira apresentada. Deve-se então criar uma origem ou destino/sumidouro fictício para que a instância esteja balanceada.

Para o problema inicial, se a produção total for maior que a capacidade total, basta criar um depósito fictício com **capacidade = produção total - capacidade total**, com custos de distribuição nulos. Se a produção total for menor que a capacidade total, basta criar uma fábrica fictícia.

Outra maneira de se resolver o problema seria tratar as restrições pertinentes não mais como equações e sim como inequações.

Soluções múltiplas ocorrem quando, detectada a solução ótima, um dos valores das contribuições for zero. O caminho fechado para a variável x_{ij} correspondente indicará a forma de obtenção da solução alternativa.

O algoritmo 1 é uma das alternativas para resolver o problema de transporte, através de fluxos em redes. O algoritmo foi proposto por Ford & Fulkerson (1962), como uma maneira de oferecer uma alternativa mais rápida do que o tradicional método Simplex tradicional, [DANTIZIG, 1963]. Isso foi conseguido através da estrutura muito particular desse problema.

Deve ser enfatizado que essa escolha do algoritmo de Ford & Fulkerson foi determinada única e exclusivamente pela disponibilidade de código aberto na internet, poupando assim os esforços inerentes à implementação do resolvedor do problema de transporte.

Registre-se, outrossim, a existência de novos algoritmos para o problema de transporte com complexidade computacional mais baixa. Evidentemente, o uso dessas alternativas tornaria o algoritmo proposto mais rápido. Deve-se, entretanto, ser observado que a finalidade precípua do trabalho de dissertação é analisar e

validar a opção GRASP híbrida como alternativa para resolver o problema de localização capacitada de custo fixo.

Tem-se como opção alternativa para resolver o problema de transporte o método chamado de *push-relabel* que serviu de base para implementações mais eficientes computacionalmente, como o algoritmo de Goldberg. Este possui uma implementação simples com complexidade $O(V^2E)$, onde V é o conjunto de vértices e E o conjunto de arestas.

O método *push-relabel* executa as operações básicas sem aplicar nenhuma ordem, entretanto é possível melhorar sua eficiência apenas escolhendo uma ordem para aplicação das operações e gerenciando adequadamente as estruturas de dados da rede. Esta nova abordagem deu origem ao algoritmo *relabel-to-front*, que possui uma complexidade $O(V^3)$, o que é assintoticamente tão bom quanto o *push-relabel*, e melhor em redes densas. [CORMEN & LEISERSON, 2002].

Algoritmo 1: Algoritmo de Ford & Fulkerson (1962).

Rotina A (Rotulamento)

Passo 1:

Inicialização das variáveis do dual.

$u_i = 0 \qquad i = 1,2,...,m$

$v_j = \min_{1 \le i \le m} c_{ij} \qquad i = 1,2,...,m;\ j = 1,2,...,n$

$x_{ij} = 0, \qquad i = 1,2,...,m;\ j = 1,2,...,n$

Passo 2:

Designar Rótulos $\left(0, \varepsilon_i = a_i - \sum_{j=1}^{n} x_{ij}\right)$ para todas as linhas de i para qual $\sum_{j=1}^{n} x_{ij} < a_i$

Passo 3:

Selecionar uma linha i rotulada e não percorrida, e percorrê-la para todas as colunas j não rotuladas semelhantes à célula (i, j). Rotular a coluna $(i, \delta_j = \varepsilon_i)$. Repetir até que todas as linhas rotuladas tenham sido percorridas.

Passo 4:

Selecionar uma coluna j rotulada e não percorrida, e percorrê-la para todas as linhas i não rotuladas semelhantes a $x_{ij} > 0$. Rotular com $(j, \varepsilon_i = \min\{x_{ij}, \delta_j\})$ linhas semelhantes. Repetir até que todas as colunas previamente rotuladas tenham sido percorridas.

Passo 5:

Repetir os passos 3 e 4 até que cada uma das colunas seja rotulada para qual $\sum_{i=1}^{m} x_{ij} < b_j$ (Ir para o passo 6, nova descoberta realizada), ou não podem mais ser designados rótulos (sem descoberta, ir para o passo 8).

Rotina B (Troca Fluxo)

Passo 6:

Existe uma coluna k com (l, δ_k) semelhante a $\sum_{i=1}^{m} x_{ik} < b_k$. Alternadamente adicione e subtraia de x_{ij} o valor $\varepsilon = \min\left\{\delta_k, b_k - \sum_{i=1}^{m} x_{ik}\right\}$ ao longo do caminho iniciando com o elemento x_{1k}, e indicado pelos primeiros elementos dos rótulos. Este processo termina quando é alcançado um rótulo com o primeiro elemento igual a zero.

Passo 7:

Se toda a demanda da coluna tiver sido satisfeita, então o algoritmo termina, caso contrário, descarta o rótulo antigo e retorna a rotina A.

Rotina C (Troca Variável Dual)

Passo 8:

O processo de rotulação que não resulta em ganho de fluxo, fazer I e J ser o conjunto de índices de linhas e colunas rotuladas, respectivamente.

Passo 9:

Calcula uma nova variável dual viável como segue:

$$\delta = \min\left\{c_{ij} + u_i - v_j : i \in I, j \in \bar{J}\right\}$$

$$\bar{u}_i = \begin{cases} u_i, & i \in I \\ u_i + \delta, & i \in \bar{I} \end{cases}$$

$$\bar{v}_i = \begin{cases} v_i, & j \in J \\ v_i + \delta, & j \in \bar{J} \end{cases}$$

onde $\bar{I}$ e $\bar{J}$ são conjuntos complementares de I e J com respeito aos conjuntos $\{1,2,...,m\}$ e $\{1,2,...,n\}$, respectivamente.

Passo 10: Retorna ao passo 2.

Neste algoritmo alternam-se trocas entre as soluções primal e dual. Portanto, o algoritmo pode ser classificado como um método primal-dual. Nesta implementação computacional o algoritmo manipula uma tabela de transporte de variáveis x_{ij} com m × n células. Células da tabela para qual $-u_i + v_j = c_{ij}$ são chamadas admissíveis, outras células são inadmissíveis.

O cálculo avança por uma seqüência de passos de rotulação (rotina A). São usados os rótulos (l, ε_i) para linhas e (l, δ_j) para colunas. Caso seja atingido um ganho de fluxo, este é aumentado pela rotina B. Se o resultado não aumenta, as variáveis dual u_i, v_j são modificadas pela rotina C. O algoritmo termina quando é obtido um fluxo que satisfaça as restrições de demanda primal, [SYSLO, DEO & KOWALIK, 1983].

3.2. Meta-heurística GRASP

A meta-heurística GRASP (Procedimento Guloso Adaptativo de Busca – ou Greedy Randomised Adaptative Search Procedure) pode ser colocada genericamente como segue, [FEO & RESENDE, 1995], [RIBEIRO & POGGI DE ARAGÃO, 1998]:

Algoritmo 2: Algoritmo genérico do GRASP

f(s*) ← +∞
Para i := 1 **até** N **faça**
Inicio
[1] Construir uma solução *s* usando um algoritmo guloso aleatorizado
[2] Aplicar um procedimento de **busca local** a partir de s, obtendo a solução s'
[3] if f(s') < f(s*) **then** s* ← s'
Fim

onde,
f(s*) é o valor da função objetivo, sobre a solução ótima até então obtida s*;
f(s') é o valor da solução corrente obtida;
N é o número de iterações;

Na etapa [1] da meta-heurística, uma solução é construída a partir de um procedimento guloso, onde a etapa de seleção construtiva deste algoritmo é

aleatorizada, ou seja, a próxima escolha gulosa é gerada aleatoriamente (usando um método de aleatorização com distribuição conhecida) entre as possíveis escolhas. Na etapa [2], uma melhoria é testada dentro da solução *s'* construída, utilizando procedimentos de troca ou refinamentos genéricos de solução, os quais podem ser procedimentos específicos relativos ao problema em análise. Por fim, na etapa [3], mantém-se a melhor solução atingida até o momento e continua-se o processo enquanto não for atingido o número de iterações.

A qualidade da solução encontrada é intrínseca à técnica de amostragem repetitiva no espaço de busca (depende do gerador randômico usado para a instância específica). Cada iteração GRASP age como se estivesse obtendo uma amostra de uma distribuição desconhecida, onde a média e variância da amostragem dependem das restrições impostas na criação da lista restrita de candidatos. Quando uma solução não é aleatorizada no passo [1], ou simplesmente a seleção é sempre constante (como no método guloso convencional), tem-se um algoritmo GRASP puramente guloso, que encontra a mesma solução em todas as iterações. Já quando se faz uma amostragem de candidatos à medida que se evolui o procedimento, os resultados obtidos podem ser ruins em algumas iterações, porém em outras podem ser muito bons devido às muitas possibilidades de amostragem do espaço de vizinhança de soluções, [FEO e RESENDE, 1995].

A grande vantagem do GRASP está justamente no aproveitamento do desempenho das heurísticas gulosas [NEGREIROS, CARVALHO & MACULAN, 2005], geralmente polinomiais de grau baixo. A facilidade de paralelização e a pouca dependência a parâmetros de amostragem e ajuste (espaço de aleatorização e número de iterações) desta meta-heurística são outros importantes pontos de vantagem, os quais podem ser explorados para atingir desempenhos melhores.

Uma maneira de se fazer a escolha da facilidade candidata na etapa [1] do PLC, é fazer a aleatorização da facilidade candidata a entrar na solução corrente S'. Este caso pode ocorrer através de duas formas distintas de GRASP: o GRASP Simples e o GRASP Adaptativo. A seguir são apresentadas.

Algoritmo 3: Busca na lista de candidatos com alfa simples

Escolha Aleatória GRASP Simples

Onde,

$|F|$ É o número de facilidades candidatas,

F É o conjunto de facilidades candidatas,

i Índice aleatorizado,

f_i Facilidade escolhida,

α Intervalo de aceitação do índice escolhido.

Repeat

$i :=$ **Random** (|F|);

Until $i \leq \alpha$;

Return f_i;

O algoritmo 3 ilustra o modo como o GRASP escolhe uma facilidade candidata a fazer parte da solução *s'*. Funciona da seguinte forma: primeiro é encontrado um índice aleatoriamente entre os valores 1 a $|F|$, este passo é repetido até que o índice aleatório esteja no intervalo α desejado, pois no GRASP simples o valor de α é fornecido pelo usuário. Após obter o índice, é retornada a facilidade candidata escolhida f_i para pertencer a *s'*.

No GRASP Adaptativo (também chamado de Reativo), ou "Reative GRASP", ao invés de um valor de α escolhido pelo usuário para definir uma faixa da quantidade das facilidades candidatas, é montado um conjunto de α's aleatoriamente dentro de um intervalo de valores para alfa, que será utilizado na hora da escolha aleatória do GRASP Adaptativo, [DELMAIRE, DÍAZ & FERNÁNDEZ, 1999].

Algoritmo 4: Montagem de um vetor de Alfas dentro de um intervalo [1,100]

Monta Vetor GRASP Reativo

Onde,

$|G|$ É o número de intervalos α,

G É o conjunto de intervalos α,

G_i É o intervalo α para o índice i,

α Intervalo de aceitação do índice escolhido.

for i = 1,..., $|G|$ **do**

G_i := **Random** ([1,100]); // gera aleatoriamente valores α de 1% a 100%

A escolha da facilidade candidata a entrar na solução s' tem agora um passo a mais, que é a escolha randômica do valor α que será utilizado dentre os vários valores que foram gerados ao montar o conjunto G. Esta escolha é a primeira parte do algoritmo 5 mostrado abaixo, sendo os passos seguintes à escolha do valor α com o qual se vai trabalhar, praticamente, os mesmos do algoritmo 3.

Algoritmo 5: Seleção de um candidato da lista para um conjunto de valores de α dados.

Escolha Aleatória GRASP Reativo

Onde,

$|G|$ É o número de intervalos α,

G É o conjunto de intervalos α,

G_i É o intervalo α para o índice i,

α Intervalo de aceitação do índice escolhido.

$|F|$ É o número de facilidades candidatas,

F É o conjunto de facilidades candidatas,

i Índice aleatorizado,

f_i Facilidade escolhida,

i := **Random**($|G|$); // Função de aleatorização com distribuição conhecida

$\alpha := G_i$;

i := **Random**([1, $|F| * \alpha$]);

Return f_i;

4. ALGORITMO HÍBRIDO PARA O PROBLEMA DE LOCALIZAÇÃO CAPACITADA DE CUSTO FIXO

Apesar da existência de heurísticas conhecidas para o problema como: ADD, DROP e outras, como visto anteriormente, na montagem da solução GRASP foi desenvolvida uma nova metodologia para o problema. A metodologia baseia-se em um algoritmo híbrido – utilizando Métodos Exatos (Problema de Transporte) e Meta-heurística (GRASP), que na verdade se combinam para encontrar soluções próximas à ótima do problema geral de Localização Capacitada de Custo Fixo, em tempo razoável (polinomial).

A metodologia desenvolvida é composta de duas fases: Uma inicial que resolve o (PLCF) construindo uma solução, e uma outra fase de melhora da solução inicial encontrada. A fase inicial deu origem ao algoritmo *localização capacitada* e a fase de melhora da solução é composta de dois algoritmos de trocas, *melhora com uma troca* e *multi-trocas.*

A seguir é apresentado de maneira sucinta o algoritmo *localização capacitada* e, logo após, são relatados os nossos procedimentos de construção da solução inicial passo a passo. O algoritmo a seguir segue basicamente a mesma linha do algoritmo ADD, incluindo novas facilidades candidatas à medida que se evolui no processamento das soluções.

È assumida a hipótese de que todos os dados do problema de localização capacitada pertencem à classe dos números inteiros. Assim, devido à propriedade de unimodularidade da matriz do problema, todas as soluções básicas são inteiras.

Algoritmo 6: Algoritmo Localização Capacitada

Inicializações;
Montar vetor custo por facilidade candidata;
Ordenar vetor custo por facilidade candidata
Monta vetor com valores alfa para ser usado no GRASP Adaptativo;

Repita
Limpa matriz de custos;
Guarda estado das facilidades candidatas;

Para *i:= 1* ***até*** *NrIterações* ***faça***
Inicio
Limpa vetores de trabalho;
Se *graspsimples* ***Então***
Obter índices de inclusão e remoção pelo método grasp simples
Senão
Obter índices de inclusão e remoção pelo método grasp adapt.;

Remover facilidade do conj. de candidatas;

Ajusta vetores de trabalho para o problema de transporte;
Obter caso de transporte a ser executado;
// 0 indica Oferta menor que demanda;
// 1 indica oferta igual a demanda;
// 2 indica oferta maior que demanda;
Calcula problema de transporte;
Obter custo total de localização;

Se *AuxCustoIteracao < CustoIteracao* ***então***
Inicio
CustoIteracao:=AuxCustoIteracao;
Guarda índice da facilidade a entrar no conj. de localizadas;
Guarda índice remoção da facilidade a sair do conj. de Cand.;
Guarda matriz de custos;
Fim;
Fim;

Restaura estado das facilidades candidatas;

Se *CustoIteracao < CustoGlobal* ***Então***
Inicio
Adicionar uma facilidade ao conj. de localizadas;
Remover facilidade do conj. de candidatas;
CustoGlobal:=CustoIteracao;
Fim;
Até *(Demanda atendida)* ***and*** *(Nr. Tentativas de inclusões de facilidades atendida);*

Passo 1:

Inicialização dos conjuntos: facilidades candidatas, facilidades localizadas, demandas dos clientes, custo fixo de facilidades, custo de transporte por unidade do cliente até a facilidade, conjunto dos valores de α para o GRASP adaptativo. Inicialização das variáveis: número de facilidades candidatas, número de clientes, número de iterações, valor de alfa, custo da iteração, custo global, índice da facilidade localizada e índice da facilidade a ser removida. Onde demandas e ofertas são tornadas inteiras (haja vista as limitações do método de Ford & Fulkerson).

Passo 2:

Montar um vetor de custos por facilidade com os seguintes dados para cada facilidade candidata: custo de transporte, custo relativo de transporte por unidade, índice da facilidade, índice da facilidade para remoção.

O custo de transporte indica o custo mínimo para se transportar toda a capacidade ofertada por uma facilidade até seus clientes, que foram escolhidos com base na aplicação do problema de transporte, utilizando o algoritmo de Ford-Fulkerson.

Para aplicação do algoritmo de transporte é preciso assegurar o balanceamento entre oferta e demanda, ou seja, a quantidade total ofertada tem que ser igual à quantidade total da demanda. Para garantir esta condição verifica-se em qual dos três casos encontram-se a quantidade de oferta e demanda:

Se a Oferta < Demanda, então é criado um vértice de oferta com o valor que falta para igualar com a demanda, sendo atribuído um custo muito alto para o transporte destas unidades até os clientes.

Se a Oferta = Demanda, então já está balanceada e pode-se aplicar o algoritmo.

Se a Oferta > Demanda, então é criado um vértice de demanda com o valor que falta para igualar com a oferta, sendo atribuído custo muito alto para o transporte destas unidades das facilidades até estes novos clientes.

É importante ressaltar que tanto o vértice de oferta, quanto de demanda que são inseridos, não entram para solução do problema de localização, são apenas usados para balancear a quantidade ofertada e a demanda.

O custo relativo de transporte indica quanto é gasto em média para transportar cada unidade ofertada de uma facilidade até o cliente. Esse valor é obtido da seguinte forma:

Custo Relativo de Transporte

CRT_i = Custo relativo para a facilidade candidata *i* atender cada unidade de demanda atendida pela facilidade.

CT_i = Custo de transporte para a facilidade candidata *i* atender a demanda dos clientes selecionados na execução do problema de transporte.

CF_i = Custo fixo de instalação da facilidade candidata *i*.

K_i = Capacidade de atendimento da facilidade candidata *i*.

$$CRT_i = (CT_i + CF_i) / K_i$$

Após montar o vetor de custos por facilidade, este é ordenado ascendentemente pelo valor do custo relativo de transporte por unidade, para que facilite o processo de escolha dos índices das facilidades candidatas na execução do GRASP.

O índice da facilidade indica qual facilidade está sendo analisada, este índice é usado mais à frente na busca das facilidades candidatas a entrar na solução, pois com a ordenação do vetor de custos por facilidade e remoção de facilidades candidatas que passaram a ser localizadas, é necessário guardar este valor, assim como o índice da facilidade para remoção, que indica a posição do índice da facilidade candidata no vetor de custos por facilidade.

Passo 3:

Montar um vetor de tamanho definido com valores alfa aleatoriamente variando em um intervalo [min, máx] que serão usados no GRASP Adaptativo.

Passo 4:

Inicio da fase de repetições com a execução de um procedimento que limpa a matriz de custos resultante da fase da execução do problema de transporte e guarda o estado das facilidades candidatas;

Passo 5:

Inicio da fase de iterações com a execução de um procedimento que limpa os vetores de trabalho com os valores da oferta, demanda e custo.

Passo 6:

Obtém os índices da facilidade a ser incluída na solução e o seu índice de remoção no vetor de custos montado no passo 2 aplicando se o GRASP Simples ou o GRASP Adaptativo. A diferença entre as duas formas do GRASP está na execução dos itens *a* e *b*, que explicam a forma como são fornecidos e utilizados os valores de α

.

a) O GRASP Simples funciona da seguinte forma: Escolhe-se aleatoriamente o índice de uma facilidade, no conjunto de candidatas, que esteja na faixa dos valores de alfa, sendo que este valor é fornecido pelo usuário no GRASP Simples;

b) Já o GRASP Reativo (Adaptativo) difere do GRASP Simples, pois, primeiro é escolhido aleatoriamente o índice do conjunto dos valores de α com o valor que se vai trabalhar. Depois o funcionamento passa a ser semelhante ao do GRASP Simples, e escolhe-se aleatoriamente o índice de uma facilidade, no conjunto de candidatas, que esteja na faixa dos valores de alfa.

Passo 7:

Remove a cada iteração uma facilidade do conjunto de facilidades candidatas que foi escolhida no passo anterior para entrar na solução da iteração. Essa remoção tem por objetivo não deixar que uma facilidade já testada anteriormente entre na tentativa de melhora, pois seria uma iteração computacionalmente perdida, uma vez que a solução seria a mesma já calculada de outra vez.

Passo 8:

Ajusta vetores de trabalho e verifica em que caso de balanceamento está à facilidade candidata a entrar na solução junto com as facilidades já localizadas, caso haja alguma, e faz o balanceamento se necessário.

Passo 9:

Executa o algoritmo de transporte de acordo com o caso entre o valor de oferta e demanda.

Passo 10:

Obtém o custo total de localização que é composto do custo de transporte mais o custo fixo das facilidades localizadas e o atribui a uma variável auxiliar, que guarda o custo da iteração.

Verifica se esta variável auxiliar é menor que o custo da iteração anterior. Caso seja menor, então o custo da iteração recebe o valor da variável auxiliar. A quantidade de iterações é um valor fornecido pelo usuário. Guarda-se o índice da facilidade localiza e o índice dessa facilidade a ser removido do conjunto de facilidades candidatas do vetor montado no passo 2. Os passos de 5 a 10 formam a fase de iterações, que varia de acordo com o valor fornecido pelo usuário no programa.

Passo 11:

Restaura o conjunto de facilidades candidatas para o seu estado anterior à fase de iterações.

Passo 12:

Verifica se o menor custo das iterações encontrado é menor que o custo global atual, caso seja menor então a facilidade candidata é adicionada ao conjunto das facilidades localizadas, a facilidade é removida do conjunto das candidatas e o custo global recebe o custo da iteração.

Passo 13:

Verifica se toda a demanda já foi atendida e se o número de tentativas de inclusão de facilidades a solução do problema também foi alcançado, caso tenha sido então uma solução já foi encontrada e pára, senão volta ao item 1 deste passo.

Ao fim do passo 12 termina-se a fase de construção de uma solução inicial para o problema de localização capacitada de custo fixo. A seguir são apresentados os dois algoritmos da fase de melhora da solução.

A figura 3 ilustra a melhoria gradual da solução à medida que vão sendo localizadas novas facilidades, utilizando o procedimento que elaboramos.

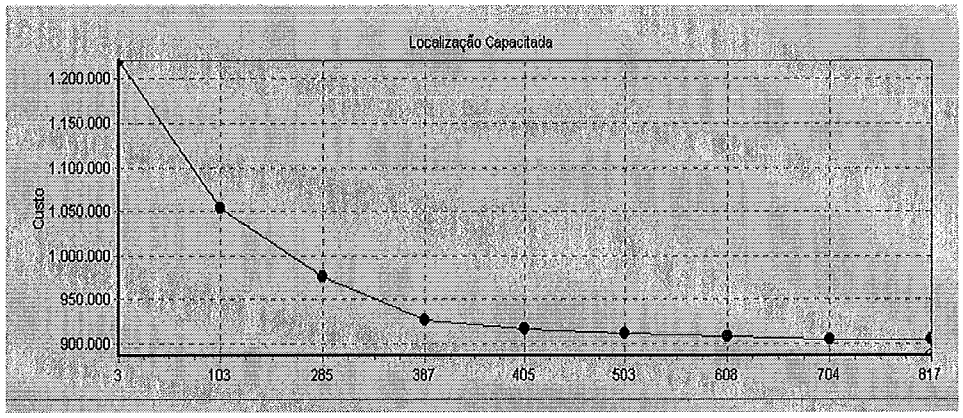

Figura 3: Função de decréscimo de Custo por acréscimo de facilidades localizadas (ADD).

Fazendo-se uma breve análise da complexidade do algoritmo de localização capacitada proposto, tem-se:

m Facilidades;

n Clientes;

Contando as operações mais caras do ponto de vista computacional dentro do laço interno:

$m \times n$ Operações para limpar os vetores de trabalho;

$m \times n$ Operações para ajustar os vetores de trabalho para executar o problema de transporte;

$k(m).n^2$ Operações para execução do problema de transporte, onde $k(m)$ é uma função quadrática do número de facilidades.

O laço interno é executado $k(m)$. Portanto, no laço interno tem-se a seguinte complexidade $O(m^2n^2)$ de pior caso e $\Theta(k^2(m).n^2)$ para o caso médio.

Na estrutura de repetição mais externa temos q iterações para atender a condição de parada.

Finalizando a analise do algoritmo de localização capacitada tem-se uma complexidade de pior caso $O(m^2n^2)$ e uma complexidade de caso médio $\Theta(q.k^2(m).n^2)$.

Algoritmo 7: Algoritmo Melhora com uma troca

O algoritmo melhora com uma troca é o primeiro da fase de melhora a ser executado. Ele tem como objetivo pegar a solução inicial encontrada e fazer trocas entre os elementos das facilidades localizadas e as facilidades candidatas que ficaram de fora da solução, testando se ocorrem melhoras na solução. O algoritmo faz uma troca por vez entre estes conjuntos e é executado para cada facilidade do conjunto de candidatas. O algoritmo funciona da seguinte forma:

```
Para i:=1 até NrCandidatas faça
Inicio
        Limpa matriz de custos;
        Ajusta conj. de facilidades localizadas trocando-se uma pela candidata i;
        Calcula o problema de transporte;

        Se CustoTroca  < CustoSolucao então
        Inicio
                CustoSolucao := CustoTroca;
                Realiza efetivamente a troca entre as facilidades;
                Guarda matriz de custo;
        Fim;
Fim;
```

Passo 1:

Inicia as iterações com a execução de um procedimento que limpa a matriz de custos resultante da fase da execução do problema de transporte.

Passo 2:

Ajusta vetores de trabalho e verifica em que caso de balanceamento está a nova configuração, pela troca da facilidade candidata *i* por uma facilidade localizada escolhida aleatoriamente para sair e faz o balanceamento se necessário.

Passo 3:

Executa o algoritmo de transporte, de acordo com o caso entre o valor de oferta e demanda, e retorna o novo custo da variável custo troca.

Passo 4:

Verifica se o custo com a troca é menor que o custo global anterior, caso seja menor, a facilidade candidata entra na solução no lugar da outra facilidade que estava no conjunto das localizadas e guarda a matriz de custo resultante do problema de transporte. Se for maior mantém o custo global e o conjunto da solução como estava. Repete os passos para cada facilidade candidata em busca de melhorias na solução.

Algoritmo 8: Algoritmo Multi-Trocas

O algoritmo multi-trocas foi desenvolvido usando como base os trabalhos de [PÁL, TARDOS & WEXLER, 2001], que em seu algoritmo implementou os três tipos seguintes de operações de melhora: abrir uma facilidade; abrir uma facilidade e fechar uma ou mais facilidades; fechar uma facilidade e abrir uma ou mais facilidades. Foi utilizado também como referência o trabalho de [ZHANG, CHEN & YE, 2005], o qual introduziu um novo tipo de operação de troca, chamada *multi-exchange*, a qual possibilita serem feitas múltiplas trocas simultaneamente, isto é, fecha muitas facilidades e abre muitas facilidades.

O algoritmo implementa a nova operação proposta em [ZHANG, CHEN & YE, 2005], que em uma operação de troca pode executar uma das três operações descritas anteriormente no trabalho de [PÁL, TARDOS & WEXLER, 2001].

Com o objetivo de buscar melhorar a solução do PLC encontrada com os algoritmos 6 e 7, o algoritmo multi-trocas (algoritmo 8) executa um número de iterações definidas e a cada iteração uma nova operação é escolhida aleatoriamente dentre as três implementadas.

Algoritmo 8: ALGORITMO MULTI-TROCAS

```
Guardar estado da solução inicial do PLCF;
Para i:= 1 até iteracoes faça
Inicio
        Montar vetor custo por facilidade localizada;
        Ordenar vetor custo por facilidade localizada;

        CasoTroca := RandomRange(0,2);
        // 0 - Adicionar uma facilidade
        Se CasoTroca = 0 então
                Adicionar uma facilidade ao conj. de localizadas;

        // 1 - Adicionar varias facilidades e retirar uma localizada
        Se CasoTroca = 1 então
        Inicio
                Adicionar facilidade ao conj. de candidatas;
                Remover facilidade do conj. de localizadas;
                NrLocAdicionadas := RandomRange(1,3);
                Se NrLocAdicionadas > NrCandidatas então
                        NrLocAdicionadas := NrCandidatas;

                Para j:= 1 até NrLocAdicionadas faça
                Inicio
                        Adicionar uma facilidade ao conj. de localizadas;
                        Remover facilidade do conj. de candidatas;
                Fim;
        Fim;

        // 2 - Adicionar uma facilidade e retirar varias localizadas
        Se CasoTroca = 2 então
        Inicio
                NrLocRemovidas := RandomRange(1,5);
                Para j:= 1 até NrLocRemovidas faça
                Inicio
                        Adicionar facilidade ao conj. de candidatas;
                        Remover facilidade do conj. de localizadas;
                Fim;
                Adicionar uma facilidade ao conj. de localizadas;
                Remover facilidade do conj. de candidatas;
        Fim;
Fim;

Limpa vetores de trabalho;
Ajusta vetores de trabalho para o problema de transporte;
Obter caso de transporte a ser executado;

// 0 indica Oferta menor que demanda;
// 1 indica oferta igual a demanda;
// 2 indica oferta maior que demanda;
```

Calcula o problema de transporte;

Obter custo total de localização;
Se *CustoIteracao < CustoGlobal* ***então***
Inicio
    Guardar estado da solução do PLCF;
    CustoGlobal := CustoIteracao;
    LimpaMatriz(XFinal);
    CopiaUltimoCusto(XFinal,X);
    CopiaUltimoCusto(EstadoXFinal,X);
Fim
Senão
Inicio
    Restaura estado da solução anterior a iteração;
    CopiaUltimoCusto(XFinal,EstadoXFinal);
Fim*;*

O algoritmo de multi-trocas usa um modelo aleatório na escolha de um dos três casos de trocas, como se detalha a seguir.

Passo 1:

Na fase de inicialização são guardados os estados das facilidades localizadas e facilidades candidatas para que se possa voltar ao estado anterior se as alterações realizadas não resultarem em melhora na solução do problema.

Passo 2:

Posteriormente, na fase das iterações do algoritmo, é montado um vetor que guarda os custos por unidade transportada para cada facilidade localizada, sendo este custo calculado da seguinte forma:

Custo de Transporte por Facilidades Localizadas

CRT_i = Custo relativo para a facilidade localizada i atender cada unidade de demanda.

CT_i = Custo de transporte para a facilidade localizada *i* atender a demanda dos clientes selecionados na execução do problema de transporte.

CF_i = Custo fixo de instalação da facilidade localizada *i*.

K_i = Capacidade atendida pela facilidade localizada *i*.

$$CRT_i = (CT_i + CF_i) / K_i$$

Depois de calculados esses custos e montado o vetor, é feita uma ordenação ascendente de acordo com o custo por unidade atendida. Este vetor será usado para auxiliar na escolha das facilidades localizadas que sairão da solução do problema por terem a pior relação CRT_i.

Passo 3:

Foram implementados três casos de trocas distintos que serão executados um por vez a cada iteração. É selecionado aleatoriamente um dos casos de troca a ser executado pela iteração:

a) Adiciona uma facilidade ao grupo de localizadas;

   Uma facilidade candidata é incluída na solução e passa a compor o conjunto de facilidades localizadas.

b) Adiciona várias facilidades candidatas ao grupo de localizadas e retira uma do grupo de localizadas;

   Uma facilidade localizada é retirada da solução do problema e volta a compor o conjunto de facilidades candidatas. A facilidade localizada retirada sempre é a última que compõe o vetor que foi montado no passo 2, pois ela é a que menos contribui na solução do problema.

Aleatoriamente é escolhido o número de facilidades que serão incluídas no conjunto de localizadas e iterativamente uma nova facilidade candidata é incluída na solução e passa a compor o conjunto de facilidades localizadas.

c) Adiciona uma facilidade ao grupo de localizadas e retiram várias.

Aleatoriamente é escolhido o número de facilidades que serão excluídas do conjunto de localizadas e iterativamente cada facilidade localizada é excluída da solução e passa a compor novamente o conjunto de facilidades candidatas.

Uma facilidade candidata é incluída na solução e passa a compor o conjunto de facilidades localizadas.

Ao final deste passo, caso o número de iterações tenha sido alcançado, a execução vai para o passo 4, caso contrário, se repete iterativamente de acordo com o número de iterações definida voltando-se ao passo 2.

Passo 4:

Execução de um procedimento que limpa os vetores de trabalho com os valores da oferta, demanda e custo.

Passo 6:

Obtém os índices da facilidade a ser incluída na solução e o seu índice de remoção no vetor de custos montado no passo 2 do algoritmo localização capacitada aplicando-se o GRASP Simples ou o GRASP Adaptativo.

Passo 7:

Ajusta vetores de trabalho e verifica o estado de balanceamento relativo ao problema de transporte da nova solução de facilidades localizadas e, caso haja alguma necessidade, faz o balanceamento.

Passo 8:

Executa o algoritmo de transporte de acordo com o caso entre o valor de oferta e demanda.

Passo 9:

Obtém o custo total de localização que é composto do custo de transporte mais o custo fixo das facilidades localizadas e o atribui a uma variável auxiliar que guarda o custo da iteração.

O custo da iteração é comparado com o valor do custo global da solução e, caso seja menor que o custo global, este recebe o valor do custo da iteração, são guardados os novos estados dos grupos de facilidades localizadas e candidatas, limpa-se a matriz de custos e guarda-se o seu estado; caso contrário, é mantido o custo da solução e volta-se ao estado anterior da iteração para os grupos de facilidades candidatas e localizadas.

Ao final da execução dos algoritmos, são mostrados os resultados dos valores dos custos com e sem as trocas, o conjunto solução das facilidades localizadas com e sem as trocas, e o tempo de execução com e sem as trocas.

5. RESULTADOS

Com o objetivo de analisar o desempenho do algoritmo proposto foi desenvolvido um aplicativo batizado com o nome de *SISLOC*, o qual tem como entrada de dados um problema teste para o PLC no formato da OR-Library e de Thizy, Como mostrados abaixo:

OR-Library

$m \quad n$
$Q_1 \quad f_1$
$Q_2 \quad f_2$
$\vdots$
$Q_m \quad f_m$
q_1
$c_{11} \quad c_{12} \quad c_{13} \quad \ldots \quad c_{1m}$
q_2
$c_{21} \quad c_{22} \quad c_{23} \quad \ldots \quad c_{2m}$
$\vdots$
q_n
$c_{n1} \quad c_{n2} \quad c_{n3} \quad \ldots \quad c_{nm}$

Thizy

$m \quad n$
$Q_1 \quad f_1$
$Q_2 \quad f_2$
$\vdots$
$Q_m \quad f_m$
q_1
$c_{11} \quad c_{12} \quad c_{13} \quad \ldots \quad c_{1n}$
q_2
$c_{21} \quad c_{22} \quad c_{23} \quad \ldots \quad c_{2n}$
$\vdots$
q_m
$c_{m1} \quad c_{m2} \quad c_{m3} \quad \ldots \quad c_{mn}$

Onde,

m - Número de facilidades candidatas;
n - Número de clientes;

Q_i - Capacidade da facilidade i;
f_i - Custo fixo da facilidade i;
q_j - demanda do cliente j;
c_{ij} - custo de transporte (pex. Km) por unidade de produto, enviado da facilidade i para o cliente j.

Os testes foram realizados com problemas teste conhecidos e muito utilizados no PLC: OR-Libray e Thizy, como mostram a Literatura em diversos trabalhos [JACOBSEN, 1983] e [VALIATI, 2006]. Foi utilizado nos testes um computador com processador Semprom 2.4GHz e 1,5 GB de memória RAM.

As tabelas 1 e 2, apresentam os resultados dos experimentos gerados com esta versão de GRASP que foi elaborada, para os casos Simples e Reativo (Adaptativo), e comparada com o resultado dos problemas teste da OR-LIBRARY para o PLC. Na Tabela 1 e 2, são os seguintes os parâmetros utilizados:

1. $|F|$ é o número de facilidades candidatas;
2. $|C|$ é o número de clientes;
3. $|I|$ é o número de facilidades localizadas;
4. GAP é a diferença percentual entre o valor de referência – Solução Ótima (OR-Library) e o obtido pelo algoritmo;
5. Para cada problema teste da OR-Library o aplicativo executou com um valor α igual a 100% para o GRASP Simples, já para o GRASP Adaptativo o valor de α foi definido em torno de [60,100] para todas as instâncias analisadas, como já dito anteriormente.

Os dados mostrados são referentes aos melhores resultados entre 10 execuções para cada instância. O tempo informado é o tempo da instância que obteve o melhor resultado em custo global.

Tabela 1: Soluções sobre as instâncias menores da OR-Library

GRASP			Simples			Reativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (ms)	$\|L\|$	GAP %	Tempo (ms)	OR-Library
Cap41	16	50	12	0,07	125	12	0,07	125	1040444.375
Cap42	16	50	12	0,06	155	12	0,06	155	1098000.450
Cap43	16	50	12	0,06	161	12	0,06	145	1153000.450
Cap44	16	50	12	0,06	139	12	0,06	143	1235500.450
Cap51	16	50	8	0,03	108	8	0,03	98	1025208.225
Cap61	16	50	11	0,00	172	11	0,00	213	932615.750
Cap62	16	50	9	0,00	101	9	0,00	269	977799.400
Cap63	16	50	7	0,00	62	7	0,00	78	1014062.050
Cap64	16	50	5	0,00	87	5	0,00	95	1045650.250
Cap71	16	50	11	0,10	210	11	0,10	211	932615.750
Cap72	16	50	9	0,10	140	9	0,10	171	977799.400
Cap73	16	50	5	0,00	149	5	0,00	151	1010641.450
Cap74	16	50	4	0,00	53	4	0,00	55	1034976.975
Cap81	25	50	17	0,14	172	16	0,12	335	838499.288
Cap82	25	50	14	0,01	172	14	0,67	172	910889.563
Cap83	25	50	14	0,01	157	13	0,36	296	975889.563
Cap91	25	50	13	0,35	404	13	0,35	205	796648.438
Cap92	25	50	11	0,06	169	11	0,06	185	855733.500
Cap93	25	50	8	0,11	94	8	0,24	152	896617.538
Cap94	25	50	7	0,06	125	7	0,22	153	946051.325
Cap101	25	50	13	0,31	204	13	0,31	198	796648.437
Cap102	25	50	11	0,15	176	11	0,15	163	854704.200
Cap103	25	50	7	0,23	108	7	0,23	122	893782.112

GRASP			Simples			Reativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (ms)	$\|L\|$	GAP %	Tempo (ms)	OR-Library
Cap104	25	50	4	0,00	98	4	0,00	82	928941.750
Cap111	50	50	17	0,11	437	18	0,34	484	826124.713
Cap112	50	50	15	0,01	485	15	0,31	500	901377.213
Cap113	50	50	14	0,09	656	14	0,28	576	970567.750
Cap114	50	50	13	0,16	562	13	0,26	583	1063356.488
Cap121	50	50	13	0,35	321	13	0,35	336	793439.563
Cap122	50	50	11	0,00	250	11	0,06	300	852524.625
Cap123	50	50	9	0,09	250	9	0,26	296	895302.325
Cap124	50	50	7	0,06	344	7	0,22	277	946051.325
Cap131	50	50	13	0,31	273	13	0,31	295	793439.562
Cap132	50	50	11	0,15	237	11	0,15	278	851495.325
Cap133	50	50	7	0,31	170	7	0,31	191	893076.712
Cap134	50	50	4	0,00	138	4	0,00	162	928941.750
Média				**0,0986**	**213**		**0,1678**	**229**	

Na tabela 1, analisando os resultados do algoritmo implementado usando GRASP Simples e Adaptativo, pode-se observar um relativo equilíbrio no desempenho, tanto em tempo de execução quanto ao GAP, para a maioria dos problemas teste. O GRASP Reativo teve em média um GAP de 0,17% e o GRASP Simples em média 0,1%. Já em relação ao tempo de execução o GRASP Simples levou em média 213ms contra 229ms do GRASP Reativo.

Os valores encontrados no geral apresentaram uma qualidade muito boa, como estão mostrados nas tabelas de resultados. A tabela 1 apresenta 24 problemas teste com GAP $\leq$ 0,1%, sendo que destas, 9 apresentam GAP igual a zero %. O pior caso na tabela 1 foi da instância Cap82, que ficou apenas 0,67% acima do valor de referência para o GRASP Reativo, já para GRASP Simples o GAP foi de 0,01%. Para todos os outros casos pode-se dizer que a solução ficou muito próxima do ótimo, pois o GAP ficou abaixo de 0,5% comparado com o valor de referência, fato que pode ser explicado pela precisão utilizada e pelos arredondamentos feitos pela própria máquina.

Durante a realização dos testes, pôde-se observar que quanto mais alto o valor de α do GRASP Simples, mais constantes permaneciam a qualidade das soluções, o mesmo valendo para a faixa de α do GRASP Reativo.

Fazendo-se uma comparação com relação ao tempo de execução da aplicação desenvolvida com a descrita em [BEASLEY, 1988], considerando os testes que encontram o valor ótimo ou que apresentam um GAP $\leq$ 0,1%, tem-se 24 instâncias, como mostrado na tabela 2.

Tabela 2: Tempo sobre as instâncias da OR-Library

Instância	GRASP Simples	GRASP Adaptativo	[BEASLEY, 1988]
	Tempo(s)	Tempo(s)	Tempo(s)
Cap41	0,125	0,125	1,2
Cap42	0,155	0,155	1,0
Cap43	0,161	0,145	1,9
Cap44	0,139	0,143	0,9
Cap51	0,108	0,098	1,2
Cap61	0,172	0,213	0,3
Cap62	0,101	0,269	0,5
Cap63	0,062	0,078	1,1
Cap64	0,087	0,095	0,9
Cap71	0,210	0,211	0,2
Cap72	0,140	0,171	0,2
Cap73	0,149	0,151	0,2
Cap74	0,053	0,055	0,1
Cap82	0,172	0,172	1,8
Cap83	0,157	0,296	1,8
Cap92	0,169	0,185	0,9
Cap94	0,125	0,153	3,1
Cap104	0,098	0,082	0,2
Cap112	0,485	0,500	4,3
Cap113	0,656	0,576	8,7
Cap122	0,250	0,300	2,3
Cap123	0,250	0,296	3,8
Cap124	0,344	0,277	7,5
Cap134	0,138	0,162	0,7
Média	**0,188**	**0,205**	**1,87**

O tempo de execução do algoritmo proposto em média para os problemas teste da tabela 2 é de 0,188s contra 1,87s, ou seja, o nosso algoritmo ficou cerca 9,94 vezes mais rápido na média em relação ao método de [BEASLEY, 1988] para as instâncias menores que são de: 16x50, 25x50 e 50x50. O equipamento que Beasley usou em seus experimentos com o método exato foi um CRAY-1S, onde deve-se considerar o fato deste ser um equipamento de meados dos anos 80.

Na tabela 3 são apresentados os resultados obtidos referentes às instâncias 100x1000 da OR-Library . Analisando-se os resultados pode-se observar que o algoritmo proposto obteve em média um GAP de 0,39% para o GRASP Simples. Sendo que foi alcançado o valor ótimo

para três instâncias, Cap A – IV, Cap B – II e Cap C – III; e o pior resultado foi o da instância Cap B – III, a qual ficou em 1,05% para o GRASP Simples e 0,89% para o GRASP Reativo.

Tabela 3: Soluções sobre as instâncias maiores da OR-Library.

GRASP			Simples			Reativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (s)	$\|L\|$	GAP %	Tempo (s)	OR-Library
Cap A – I	100	1000	7	0,59	107,23	7	0,59	102,42	19.240.822,449
Cap A – II	100	1000	6	0,73	73,13	6	0,73	81,63	18.438046,543
Cap A – III	100	1000	5	0,31	79,33	5	0,31	87,86	17.765201,949
Cap A – IV	100	1000	4	0,00	88,9	4	0,00	89,3	17.160439,012
Cap B – I	100	1000	11	0,39	129,78	11	0,39	129,71	13.656379,578
Cap B – II	100	1000	9	0,04	119,94	9	0,48	146,3	13.361927.449
Cap B – III	100	1000	9	1,05	124,9	8	0,89	156,2	13.198556,434
Cap B – IV	100	1000	7	0,52	149,1	8	0,61	128,7	13.082516,496
Cap C – I	100	1000	11	0,44	120,07	11	0,66	137,33	11.646596,974
Cap C – II	100	1000	10	0,17	106,78	10	0,17	106,67	11.570340,289
Cap C – III	100	1000	9	0,00	106,03	9	0,11	110,36	11.518743,744
Cap C – IV	100	1000	9	0,41	96,84	9	0,74	118,3	11.505767,394
Média				**0,3875**	**108,5**		**0,4733**	**116,2**	

As tabelas 4 e 5, apresentam os resultados dos experimentos gerados com esta versão de GRASP elaborada, para os casos Simples e Reativo (Adaptativo), e comparamos com o resultado dos problemas teste de Thizy para o PLC.

As instâncias teste de Thizy estão divididos em duas partes: a primeira, tabela 4, apresenta as instâncias teste trabalhados em [CORNUEJOLS, SRIDHARAN & THIZY, 1991] de tamanho 8x25, 16x25, 25x25, 16x50, 33x50 e 50x50; a segunda parte, tabela 5, apresenta as instâncias teste trabalhadas em [VALIATI, 2006] de tamanho 200x200, que simulam as características das instâncias apresentadas por Thizy.

Na Tabela 4 e 5, são os seguintes os parâmetros utilizados:

1. $|F|$ é o número de facilidades candidatas;
2. $|C|$ é o número de clientes;
3. $|L|$ é o número de facilidades localizadas;
4. GAP é a diferença percentual entre o valor de referência Thizy ótimo e o obtido pelo algoritmo;
5. Para cada instância teste Thizy o aplicativo executou com um valor α igual a 100% para o GRASP Simples, já para o GRASP Adaptativo o valor de α foi definido em torno de [60, 100] para todas as instâncias analisadas, como já dito anteriormente.

Tabela 4: Problemas teste originais Thizy.

GRASP			Simples			Adaptativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (ms)	$\|L\|$	GAP %	Tempo (ms)	Thizy-ORI
thiz1-a1.dad	8	25	4	0,00	20	4	1,06	16	9397
thiz1-a2.dad	8	25	5	0,00	22	5	1,42	31	11399
thiz1-a3.dad	8	25	4	0,00	20	4	0,00	15	10426
thiz1-a4.dad	8	25	4	0,00	17	4	0,00	31	10206
thiz1-a5dad	8	25	5	0,00	24	5	3,04	16	12237
thiz1-b1.dad	16	25	9	0,34	56	9	1,46	63	14844
thiz1-b2.dad	16	25	7	0,00	48	7	0,00	47	12355
thiz1-b3.dad	16	25	8	1,29	77	8	2,32	63	15534
thiz1-b4.dad	16	25	8	0,00	44	8	0,83	63	14827
thiz1-b5.dad	16	25	10	1,45	47	10	3,93	62	14226
thiz1-c1.dad	25	25	11	0,00	94	11	0,73	93	16197
thiz1-c2.dad	25	25	12	0,28	94	12	0,61	109	18593
thiz1-c3.dad	25	25	12	0,00	94	12	1,04	125	16803
thiz1-c4.dad	25	25	10	0,00	78	10	0,04	79	15833
thiz1-c5dad	25	25	13	0,15	94	13	1,32	94	16904
thiz1-d1.dad	16	50	8	0,00	79	8	0,00	63	19377
thiz1-d2.dad	16	50	7	0,00	63	7	0,00	78	18749
thiz1-d3.dad	16	50	8	0,24	63	8	3,41	78	21115
thiz1-d4.dad	16	50	8	0,02	63	8	1,14	78	20227
thiz1-d5.dad	16	50	7	0,00	63	7	3,24	62	19416
thiz1-e1.dad	33	50	16	0,78	235	16	1,26	281	27189
thiz1-e2.dad	33	50	16	0,08	312	16	0,41	328	27891

GRASP			Simples			Adaptativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (ms)	$\|L\|$	GAP %	Tempo (ms)	Thizy-ORI
thiz1-e3.dad	33	50	15	0,00	182	15	0,00	175	28194
thiz1-e4.dad	33	50	16	1,19	251	16	1,26	328	29080
thiz1-e5dad	33	50	16	0,63	315	16	0,82	343	28963
thiz1-f1.dad	50	50	23	0,00	965	23	0,00	531	33647
thiz1-f2.dad	50	50	24	0,60	500	24	0,94	1015	36237
thiz1-f3.dad	50	50	24	0,00	784	24	0,00	765	36195
thiz1-f4.dad	50	50	23	0,00	436	23	0,00	442	34073
thiz1-f5.dad	50	50	23	1,26	890	23	1,26	968	34609
thiz2-a1.dad	8	25	3	0,00	15	3	0,00	15	9052
thiz2-a2.dad	8	25	4	0,00	16	4	0,18	16	10298
thiz2-a3.dad	8	25	4	0,00	31	4	0,00	31	11179
thiz2-a4.dad	8	25	3	0,00	32	3	0,00	15	8898
thiz2-a5dad	8	25	3	0,00	15	3	0,00	15	9033
thiz2-b1.dad	16	25	5	0,49	47	5	1,30	31	10415
thiz2-b2.dad	16	25	6	0,55	94	6	0,55	62	12820
thiz2-b3.dad	16	25	5	0,00	31	5	5,91	46	10838
thiz2-b4.dad	16	25	6	0,67	62	6	0,82	31	12751
thiz2-b5.dad	16	25	5	0,00	32	5	0,00	47	11045
thiz2-c1.dad	25	25	9	0,48	78	9	0,82	94	14660
thiz2-c2.dad	25	25	9	0,00	78	9	1,53	94	15623
thiz2-c3.dad	25	25	8	1,56	62	8	1,56	78	15354
thiz2-c4.dad	25	25	8	0,00	63	8	0,50	78	14466
thiz2-c5dad	25	25	8	0,14	85	8	0,14	78	14000
thiz2-d1.dad	16	50	6	0,00	75	6	2,78	62	17830
thiz2-d2.dad	16	50	5	0,00	88	5	0,00	63	16756
thiz2-d3.dad	16	50	6	0,00	67	6	1,64	62	17181

GRASP			Simples			Adaptativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (ms)	$\|L\|$	GAP %	Tempo (ms)	Thizy-ORI
thiz2-d4.dad	16	50	7	1,16	94	7	1,53	84	18444
thiz2-d5.dad	16	50	5	0,00	70	5	0,00	62	16910
thiz2-e1.dad	33	50	10	0,87	370	10	1,40	160	22570
thiz2-e2.dad	33	50	12	1,63	392	12	1,63	281	24362
thiz2-e3.dad	33	50	10	0,00	285	10	0,00	234	23591
thiz2-e4.dad	33	50	12	0,24	313	12	1,22	313	24718
thiz2-e5dad	33	50	11	0,38	235	11	1,35	229	23962
thiz2-f1.dad	50	50	17	0,77	438	17	0,77	719	27905
thiz2-f2.dad	50	50	16	0,59	375	16	2,30	813	28352
thiz2-f3.dad	50	50	15	0,00	437	15	0,00	672	27624
thiz2-f4.dad	50	50	16	0,00	438	16	0,00	750	27117
thiz2-f5.dad	50	50	16	0,00	531	16	0,35	428	29136
thiz3-a1.dad	8	25	2	0,00	11	2	0,00	0	5058
thiz3-a2.dad	8	25	2	0,00	16	2	0,00	0	4489
thiz3-a3.dad	8	25	2	0,00	15	2	0,00	0	4365
thiz3-a4.dad	8	25	2	0,00	12	2	0,00	0	5326
thiz3-a5dad	8	25	2	0,00	13	2	0,00	0	4449
thiz3-b1.dad	16	25	4	1,10	36	4	5,46	31	4818
thiz3-b2.dad	16	25	4	0,00	78	4	0,00	63	5234
thiz3-b3.dad	16	25	4	0,00	34	4	0,00	32	5461
thiz3-b4.dad	16	25	3	0,00	32	3	6,15	15	5267
thiz3-b5.dad	16	25	3	0,00	33	3	0,00	15	5270
thiz3-c1.dad	25	25	5	0,00	73	5	2,59	47	5327
thiz3-c2.dad	25	25	6	0,00	82	6	0,00	78	5783
thiz3-c3.dad	25	25	5	2,39	125	5	5,24	78	6605
thiz3-c4.dad	25	25	5	0,00	67	5	0,22	94	6033

GRASP			Simples			Adaptativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (ms)	$\|L\|$	GAP %	Tempo (ms)	Thizy-ORI
thiz3-c5dad	25	25	5	0,68	63	5	8,26	62	6149
thiz3-d1.dad	16	50	4	0,00	67	4	1,51	47	8135
thiz3-d2.dad	16	50	4	0,00	94	4	1,77	62	8251
thiz3-d3.dad	16	50	4	1,31	79	4	5,11	47	8244
thiz3-d4.dad	16	50	4	2,29	78	4	3,95	31	8438
thiz3-d5.dad	16	50	3	0,00	58	3	0,00	31	7680
thiz2-e1.dad	33	50	7	1,77	125	7	7,44	235	9826
thiz3-e2.dad	33	50	7	0,18	272	7	1,29	140	10291
thiz3-e3.dad	33	50	7	1,71	250	7	1,87	157	10204
thiz3-e4.dad	33	50	7	1,48	157	7	2,35	219	9526
thiz3-e5dad	33	50	7	2,27	281	7	2,27	219	10350
thiz3-f1.dad	50	50	10	2,21	359	10	2,21	344	11214
thiz3-f2.dad	50	50	10	1,55	344	10	3,72	391	11937
thiz3-f3.dad	50	50	11	1,50	547	11	1,50	484	12235
thiz3-f4.dad	50	50	11	2,05	476	11	3,19	359	12242
thiz3-f5.dad	50	50	11	2,55	625	11	4,05	562	11374
thiz5-a1.dad	8	25	2	0,00	12	2	2,49	31	4894
thiz5-a2.dad	8	25	2	0,00	18	2	0,00	28	5582
thiz5-a3.dad	8	25	2	0,00	35	2	0,00	25	4444
thiz5-a4.dad	8	25	1	0,00	8	1	0,00	16	4596
thiz5-a5dad	8	25	1	0,00	12	1	0,00	0	4606
thiz5-b1.dad	16	25	2	0,00	34	2	0,00	32	4449
thiz5-b2.dad	16	25	2	0,00	27	2	1,61	17	4854
thiz5-b3.dad	16	25	2	0,00	78	2	4,87	12	4681
thiz5-b4.dad	16	25	2	0,00	25	2	0,00	16	4880
thiz5-b5.dad	16	25	2	0,00	22	2	0,00	16	4907

GRASP			Simples			Adaptativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (ms)	$\|L\|$	GAP %	Tempo (ms)	Thizy-ORI
thiz5-c1.dad	25	25	3	0,00	48	3	0,00	47	5040
thiz5-c2.dad	25	25	4	0,00	125	3	0,00	33	5267
thiz5-c3.dad	25	25	3	0,00	42	3	0,00	31	4120
thiz5-c4.dad	25	25	3	0,00	49	3	0,00	31	4862
thiz5-c5.dad	25	25	4	5,94	53	4	15,57	41	5086
thiz5-d1.dad	16	50	2	0,00	33	2	0,00	31	6697
thiz5-d2.dad	16	50	2	0,00	40	2	0,00	31	7581
thiz5-d3.dad	16	50	2	0,00	42	2	0,00	31	7834
thiz5-d4.dad	16	50	3	1,65	63	2	0,37	62	8115
thiz5-d5.dad	16	50	2	0,00	46	2	0,00	31	7847
thiz5-e1.dad	33	50	4	1,38	172	4	2,50	109	8133
thiz5-e2.dad	33	50	4	3,49	125	4	6,30	79	7575
thiz5-e3.dad	33	50	4	0,00	125	4	0,00	125	8354
thiz5-e4.dad	33	50	4	2,22	203	4	5,22	172	7929
thiz5-e5dad	33	50	4	0,00	172	4	1,98	78	8364
thiz5-f1.dad	50	50	6	2,77	265	6	2,77	266	9675
thiz5-f2.dad	50	50	6	3,41	453	5	3,41	390	9110
thiz5-f3.dad	50	50	6	3,22	484	6	3,22	297	9098
thiz5-f4.dad	50	50	6	1,77	453	6	5,74	281	9194
thiz5-f5.dad	50	50	7	0,68	472	7	0,68	516	10111
Média				**0,5784**	**160**		**1,5558**	**157**	

Analisando os resultados obtidos na tabela 4 pode-se ver que o algoritmo GRASP Simples obteve um desempenho médio melhor em relação ao GRASP Adaptativo, ficando aquele com GAP médio de 0,58% e este com GAP médio de 1,56%.

Foi observado que das 120 instâncias teste da tabela 4 à solução ótima foi obtida em 71 delas, dando cerca de 59% de soluções ótimas. Apenas 27 instâncias teste apresentaram GAP superior a 1% o que equivale a 22,5% das instâncias. O pior resultado obtido na tabela 4 refere-se à instância thiz5-c5.dad, a qual quando executada com o algoritmo GRASP Adaptativo obteve um GAP de 15,57%, mas ao ser executada pelo algoritmo GRASP Simples esse GAP caiu para 5,94%.

Com relação ao tempo de execução os dois algoritmos se mantiveram equivalentes, sendo em média executado em 160ms para o GRASP Simples e 157ms para o GRASP Adaptativo. Algumas instâncias obtiveram a solução ótima sem que fosse gasto ao menos 1ms do processador.

Acredita-se ser possível melhorar ainda mais as soluções obtidas, principalmente para o algoritmo GRASP Adaptativo, uma vez que os testes realizados foram feitos de forma "batch", ou seja, os parâmetros de entrada foram fixados (os mesmos) para todas as instâncias teste.

Tabela 5: Soluções comparadas às instâncias de Thizy

GRASP			Simples			Adaptativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (s)	$\|L\|$	GAP %	Tempo (s)	Thizy-Valiati
plc_1.5_200x200_10_1-1	200	200	57	1,56	10,43	57	1,78	11,28	53940,87109
plc_1.5_200x200_10_1-2	200	200	60	1,55	42,14	60	1,36	15,56	54328,54297
plc_1.5_200x200_10_1-3	200	200	57	1,71	12,62	57	1,69	11,31	52644,40625
plc_1.5_200x200_10_5-1	200	200	57	0,43	11,60	57	0,55	11,64	263215,4688
plc_1.5_200x200_10_5-2	200	200	60	0,58	16,68	60	0,58	16,16	264608,1563
plc_1.5_200x200_10_5-3	200	200	57	0,47	11,72	57	0,41	9,26	256818,1719
plc_1.5_200x200_10_20-1	200	200	58	0,16	28,34	57	0,43	11,56	1047887,625
plc_1.5_200x200_10_20-2	200	200	60	0,54	17,96	60	0,56	17,03	1051450,25
plc_1.5_200x200_10_20-3	200	200	57	0,20	11,30	57	0,29	10,81	1022341,375
plc_1.5_200x200_500_1-1	200	200	72	6,23	117,92	72	6,29	119,38	117885,0625
plc_1.5_200x200_500_1-2	200	200	80	4,94	149,83	79	6,01	146,75	119786,8984
plc_1.5_200x200_500_1-3	200	200	72	4,47	127,63	72	4,40	125,23	114615,2344
plc_1.5_200x200_500_20-1	200	200	57	0,19	125,08	57	0,21	119,38	1127256,375
plc_1.5_200x200_500_20-3	200	200	57	0,17	135,98	57	0,36	131,69	1099914,875
plc_2_200x200_10_1-1	200	200	44	1,54	9,46	44	1,70	8,53	47764,66406
plc_2_200x200_10_1-2	200	200	46	1,85	39,34	46	1,94	9,63	47568,30469
plc_2_200x200_10_1-3	200	200	44	1,86	29,05	44	2,01	8,86	46550,28906
plc_2_200x200_10_5-1	200	200	44	0,33	29,13	44	0,46	8,03	231654,8281
plc_2_200x200_10_5-2	200	200	46	0,66	50,98	46	0,89	10,66	230880,1875
plc_2_200x200_10_5-3	200	200	44	0,83	28,41	44	0,94	8,69	225185,9063
plc_2_200x200_10_20-1	200	200	44	0,14	9,14	44	0,14	8,47	920967,125
plc_2_200x200_10_20-2	200	200	46	0,52	48,81	46	0,63	12,06	917817,875
plc_2_200x200_10_20-3	200	200	44	0,44	27,02	44	0,80	8,75	895003,375
plc_2_200x200_500_1-1	200	200	64	10,06	105,03	64	10,21	101,53	110155,3516

GRASP			Simples			Adaptativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (s)	$\|L\|$	GAP %	Tempo (s)	Thizy-Valiati
plc_2_200x200_500_1-2	200	200	65	8,16	103,63	65	8,21	98,25	110010,5625
plc_2_200x200_500_1-3	200	200	65	10,68	101,53	65	11,42	106,83	104981,8828
plc_2_200x200_500_20-1	200	200	44	0,66	111,05	44	0,53	104,91	1007329
plc_2_200x200_500_20-2	200	200	46	1,03	123,28	46	1,29	119,36	1001728,313
plc_2_200x200_500_20-3	200	200	44	0,88	128,66	44	0,94	126,08	983115,3125
plc_5_200x200_10_1-1	200	200	24	2,95	29,58	24	3,01	11,73	35587,71875
plc_5_200x200_10_1-2	200	200	25	2,56	20,30	25	2,64	10,36	35316,625
plc_5_200x200_10_1-3	200	200	25	2,36	22,36	25	3,62	9,71	34962,34766
plc_5_200x200_10_5-1	200	200	25	0,91	25,05	24	0,95	11,93	169471,125
plc_5_200x200_10_5-2	200	200	25	1,49	13,14	25	1,62	12,67	167976,875
plc_5_200x200_10_5-3	200	200	25	1,32	10,54	25	1,52	11,28	165750,6563
plc_5_200x200_10_20-1	200	200	24	0,63	10,55	24	0,65	12,84	670560,8125
plc_5_200x200_10_20-2	200	200	25	1,21	32,16	25	1,29	15,47	665280,9375
plc_5_200x200_10_20-3	200	200	25	1,48	10,36	25	1,47	9,77	655364,875
plc_5_200x200_500_1-1	200	200	49	14,09	41,86	46	12,48	42,48	99756,16406
plc_5_200x200_500_1-2	200	200	50	17,82	49,80	49	16,54	47,61	98521,85938
plc_5_200x200_500_1-3	200	200	50	19,87	41,27	50	18,30	45,20	94833,75781
plc_5_200x200_500_20-1	200	200	24	4,53	61,14	24	4,57	66,64	769630,875
plc_5_200x200_500_20-2	200	200	25	8,24	52,25	25	8,36	63,30	728310,875
plc_5_200x200_500_20-3	200	200	25	4,71	64,53	25	3,93	63,11	748563,625
plc_10_200x200_10_1-1	200	200	15	4,62	9,14	15	4,94	8,41	27144,45508
plc_10_200x200_10_1-2	200	200	16	4,63	24,73	16	6,86	10	28339,84766
plc_10_200x200_10_1-3	200	200	15	2,41	16,81	15	2,82	9,17	27953,60938
plc_10_200x200_10_5-1	200	200	15	0,74	10,19	15	1,46	9,20	132271,3125
plc_10_200x200_10_5-2	200	200	15	2,69	8,38	15	2,93	8,53	125514,7891
plc_10_200x200_10_5-3	200	200	15	1,31	7,54	15	1,99	9,14	128657,5

GRASP			Simples			Adaptativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (s)	$\|L\|$	GAP %	Tempo (s)	Thizy-Valiati
plc_10_200x200_10_20-1	200	200	15	0,96	21,58	15	1,51	8,97	520145,2813
plc_10_200x200_10_20-2	**200**	**200**	**16**	**0,00**	**11,10**	**16**	**0,06**	**11,03**	**536308,28509**
plc_10_200x200_10_20-3	200	200	15	1,06	11,03	15	1,13	12,70	505952,6563
plc_10_200x200_500_1-1	200	200	37	13,89	16,22	32	11,76	16,16	105777,9375
plc_10_200x200_500_1-2	200	200	34	15,26	20,42	36	15,20	19,63	101288,4063
plc_10_200x200_500_1-3	200	200	37	17,38	26,03	35	15,40	21,45	93879,67188
plc_10_200x200_500_20-1	200	200	15	6,68	33,99	15	6,49	34,63	638221,25
plc_10_200x200_500_20-2	200	200	16	7,48	41,75	16	7,85	35,52	634302,625
plc_10_200x200_500_20-3	200	200	15	4,65	37,17	15	3,32	31,19	628111,375
plc_20_200x200_10_1-1	200	200	9	4,91	24,09	9	5,77	6,14	22666,37109
plc_20_200x200_10_1-2	200	200	9	4,71	14,98	9	5,53	8,11	22554,96094
plc_20_200x200_10_1-3	200	200	9	4,05	13,19	9	4,38	21,91	22213,05859
plc_20_200x200_10_5-1	200	200	9	3,49	13,84	9	4,44	7,98	100217,0859
plc_20_200x200_10_5-2	200	200	9	3,36	12,64	9	4,54	8,23	99463,58594
plc_20_200x200_10_5-3	200	200	24	1,95	27,27	24	2,29	10,94	159936,625
plc_20_200x200_10_20-1	200	200	9	3,42	15,42	9	5,15	5,86	390859,2813
plc_20_200x200_10_20-2	200	200	9	3,56	18,21	9	4,79	8,89	386867,4063
plc_20_200x200_10_20-3	200	200	9	3,54	12,74	9	4,41	8,02	381483,8125
plc_20_200x200_500_1-1	200	200	27	7,56	10,03	26	8,31	10,27	121446,5703
plc_20_200x200_500_1-2	200	200	25	10,31	14,55	27	10,99	10,23	113551,8125
plc_20_200x200_500_1-3	200	200	25	10,60	9,97	27	9,82	10,41	112696,8906
plc_20_200x200_500_20-1	200	200	9	12,35	18,39	9	7,58	12,16	544666,6875
plc_20_200x200_500_20-2	200	200	9	7,58	21,66	9	8,31	17,74	540511,25
plc_20_200x200_500_20-3	200	200	9	8,65	18,24	9	7,41	15,92	530943,5625
Média				**4,3353**	**37,70**		**4,3976**	**31,63**	

Na tabela 5 são apresentados os resultados para as instâncias teste retiradas de [VALIATI, 2006] de tamanho 200x200, que simulam as instâncias originais do trabalho de [CORNUEJOLS, SRIDHARAN & THIZY, 1991], cujo tamanho é de 8x25, 16x25, 25x25, 16x50, 33x50 e 50x50.

Os resultados entre os dois algoritmos propostos mantiveram-se equivalentes quanto ao GAP, ficando os algoritmos GRASP Simples com GAP de 4,34% e o GRASP Adaptativo com GAP 4,4%.

O tempo de execução mostrou uma pequena diferença entre os algoritmos, que em média executou mais rápido para o GRASP Adaptativo com 32 segundos, já para o GRASP Simples o tempo foi de 38 segundos.

O problema plc_10_200x200_10_20-2 não apresentava solução ótima, foi resolvido pelo nosso algoritmo GRASP Simples e obteve um custo igual a 536308,28509 e para o GRASP Adaptativo um GAP 0,06% em relação ao nosso algoritmo.

A tabela 5 apresentou dentre os casos de teste analisados anteriormente nas outras tabelas o pior desempenho em média, este fato pode ser explicado basicamente por dois motivos: O tamanho das instâncias teste que são instâncias grandes para o PLC, e a execução de forma "batch" das instâncias teste. A solução obtida por nosso algoritmo depende muito da entrada de parâmetros feita pelo usuário, que neste caso se manteve constante para todos os problemas.

Com o objetivo de mostrar como a intervenção do usuário no problema teste tem fundamental importância em nosso algoritmo, foi construída a tabela 6 que mostra o resultado da execução de maneira "batch" para o GRASP Simples e logo abaixo o resultado das mesmas instâncias teste com intervenção do usuário na entrada de parâmetros.

Foi observada uma significativa melhora na solução das instâncias teste e acredita-se ser possível melhorar ainda mais, pois o ajuste de parâmetros de entrada para o GRASP é fundamental para o sucesso do algoritmo. Esta dependência é de certa

forma danosa, quando o método é utilizado por pessoas não especializadas, porém acredita-se que com o uso da ferramenta, o analista do PLC poderá rapidamente absorver a melhor calibração de parâmetro para cada uma das instâncias que vier a resolver.

Tabela 6: Comparativo da execução "batch" com a execução com intervenção do usuário nos parâmetros.

GRASP			Simples			Adaptativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (s)	$\|L\|$	GAP %	Tempo (s)	Thizy-Valiati
Batch									
plc_10_200x200_500_1-1	200	200	**37**	**13,89**	**16,22**	32	11,76	16,16	105777,9375
plc_10_200x200_500_1-2	200	200	**34**	**15,26**	**20,42**	36	15,20	19,63	101288,4063
plc_10_200x200_500_1-3	200	200	**37**	**17,38**	**26,03**	35	15,40	21,45	93879,67188
plc_10_200x200_500_20-1	200	200	**15**	**6,68**	**33,99**	15	6,49	34,63	638221,25
plc_10_200x200_500_20-2	200	200	**16**	**7,48**	**41,75**	16	7,85	35,52	634302,625
plc_10_200x200_500_20-3	200	200	**15**	**4,65**	**37,17**	15	3,32	31,19	628111,375
Com intervenção									
plc_10_200x200_500_1-1	200	200	**75**	**7,18**	**97,84**	32	11,76	16,16	105777,9375
plc_10_200x200_500_1-2	200	200	**67**	**8,72**	**49,67**	36	15,20	19,63	101288,4063
plc_10_200x200_500_1-3	200	200	**77**	**4,64**	**129,28**	35	15,40	21,45	93879,67188
plc_10_200x200_500_20-1	200	200	**15**	**4,57**	**8,02**	15	5,65	7,33	638221,25
plc_10_200x200_500_20-2	200	200	**17**	**4,52**	**21,09**	16	5,51	19,26	634302,625
plc_10_200x200_500_20-3	200	200	**15**	**3,38**	**11,5**	15	3,02	10,13	628111,375

A tabela 7 apresenta um comparativo de tempo com algumas instâncias teste nas quais foi obtido um GAP $\leq$ 2% com relação ao valor ótimo.

Tabela 7: Tempo para os problemas teste de Thizy.

Instância	GRASP Simples	GRASP Adaptativo	[THIZY]
	Tempo(s)	Tempo(s)	Tempo(s)
Plc_1,5_200x200_10_1-2	42,14	15,56	169,14
Plc_1,5_200x200_10_1-3	12,62	11,31	147,12
Plc_1.5_200x200_10_5-2	16,68	16,16	326,82
Plc_1.5_200x200_10_5-3	11,72	9,26	133,82
Plc_1.5_200x200_10_20-2	17,96	17,03	247,22
Plc_1.5_200x200_10_20-3	11,30	10,81	123,72
Plc_2_200x200_10_1-1	9,46	8,53	730,76
Plc_5_200x200_10_5-3	10,54	10,54	82393,9
Plc_5_200x200_10_20-1	10,55	12,84	10555,66
Plc_5_200x200_10_20-3	10,36	9,77	74141,24
Plc_10_200x200_10_5-1	10,19	9,20	10657,46
Plc_10_200x200_10_20-1	21,58	8,97	31740,99
Média	**15,43**	**11,67**	**17614**

Analisando-se os tempos de execução da tabela 7, pode ser visto que nossa aplicação conseguiu resolver as instâncias do problema com valores muito bons utilizando um computador com processador Sempron 2.4 Mhz.

O desempenho do algoritmo proposto para o PLC foi avaliado nas suas duas variações implementadas: o GRASP Simples e GRASP Adaptativo, com relação ao GAP (distância da solução obtida em relação ao ótimo conhecido) e ao tempo de execução.

Para verificar o comportamento do método proposto, de modo a garantir a reprodutividade dos ensaios aqui realizados, foi realizada uma análise da robustez do método da seguinte maneira: cada instância teste foi executada 100 vezes, tendo como meta alcançar o valor ótimo para as instâncias teste, ou seja, obter um GAP o mais próximo de 0% e no menor tempo possível.

Para cada execução i da instância teste analisada, foi guardado o valor do GAP obtido e o tempo de execução. Os resultados de algumas instâncias teste foram escolhidos e são apresentados, onde são mostrados separadamente em dois gráficos, um com relação ao número da execução e o GAP, e o seguinte com relação ao número da execução e o tempo. O resultado do algoritmo é então apresentado no gráfico, associando um valor do GAP (g_i) da execução, que está ordenado ascendentemente com um valor $e_i = i$ da execução, e gerando pontos no espaço $\Re^2$ da forma $z_i = (g_i, e_i)$, para i=1,2,...,100. [TRINDADE, 2005].

Na análise da robustez, procurou-se selecionar instâncias teste que obtiveram resultados bons e não muito bons. Nosso objetivo foi fazer uma análise ampla do ponto de vista de melhor caso, caso médio e pior caso do método, em termos de desempenho e custo relativo à solução ótima. Dentro dessas características, foram escolhidas as instâncias teste: Cap44_16x50, Cap122_50x50, Thizy-f5_50x50 e Plc_1.5_200x200_10_5-3.

Na instância Cap44_16x50 foram obtidos resultados muito bons para as duas versões do algoritmo. Com o GRASP Adaptativo as 100 execuções se mantiveram muito próximas do GAP 0% e para o GRASP Simples cerca de 70% das execuções mantiveram o GAP próximo de 0%. O tempo de execução das instâncias se manteve praticamente constante para 84% das execuções no GRASP Adaptativo e 72% para o GRASP Simples.

O Cap122_50x50 mostrou uma variação média em suas execuções, nas quais o GRASP Simples manteve 93% de suas execuções abaixo do GAP de 1,5% e o GRASP Simples obteve cerca de 63% de execuções para o mesmo GAP. Nesta instância pode-se observar que o GRASP Simples manteve para todas as execuções uma solução melhor em relação ao GRASP Adaptativo levando-se em conta o GAP, já em relação ao tempo ocorreu o inverso.

A instância Thizy-f5_50x50, embora consiga resultados bons para suas primeiras execuções, mostra uma variação de aproximadamente 14% no GAP entre a primeira execução e a centésima. O tempo se manteve praticamente constante para cada uma das instâncias, mas o GRASP Simples apresentou um desempenho melhor quando comparado com o mesmo tempo de execução para o GRASP Adaptativo.

Na instância Plc_1.5_200x200_10_5-3 cerca de 90% das execuções mantiveram-se com GAP entre 0,5% e 1%, ficando o GRASP Adaptativo com os resultados um pouco melhores do que o GRASP Simples. Com relação ao tempo de execução os dois algoritmos apresentaram tempos muito próximos.

Com a análise de robustez procurou-se mostrar, escolhendo algumas instâncias, que o algoritmo híbrido proposto apresenta uma qualidade muito boa. Embora algumas instâncias tenham apresentado resultados considerados ruins para esta classe de algoritmos, deve-se ter em mente que mesmo uma meta-heurística está sujeita a tais variações em problemas combinatoriais (como é do conhecimento da literatura sobre casos patológicos).

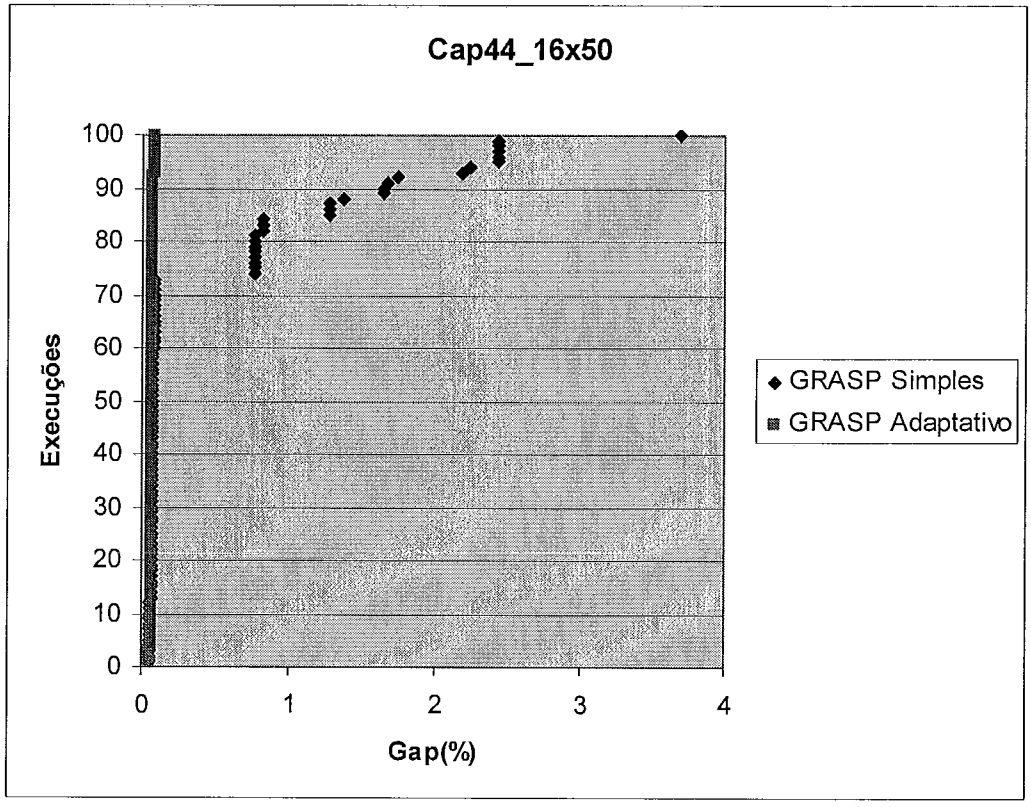

Figura 4: a) Cap44_16x50 Execuções x Gap(%)

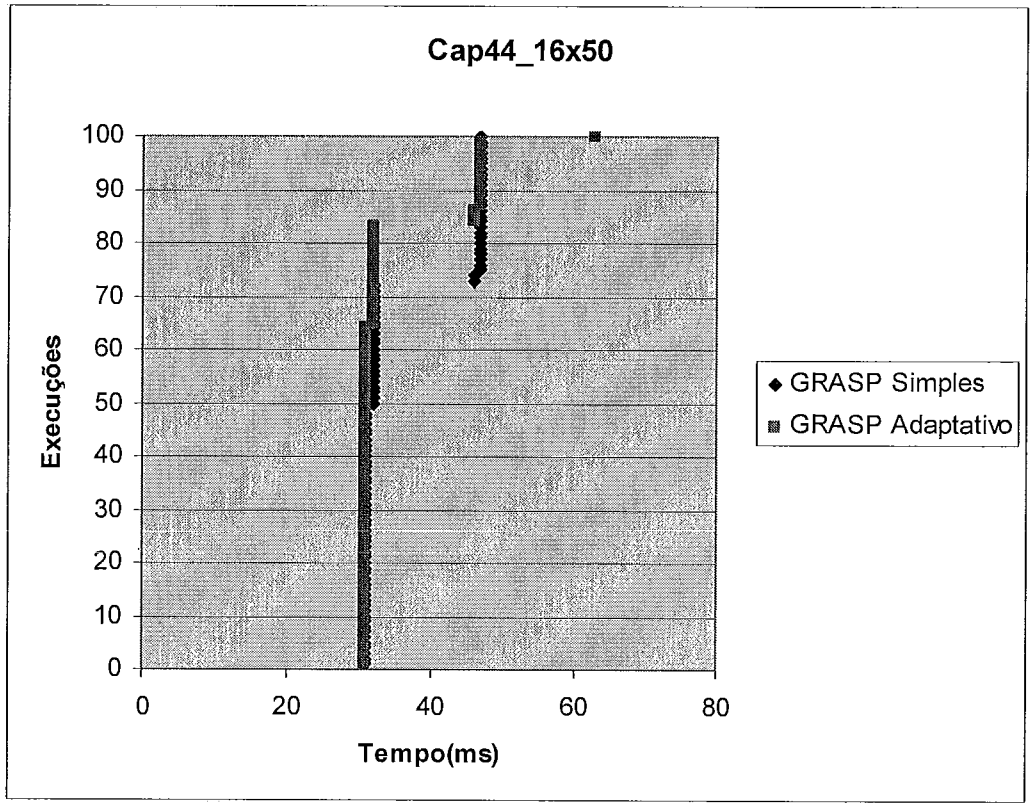

Figura 4: b) Cap44_16x50 Execuções x Tempo(ms)

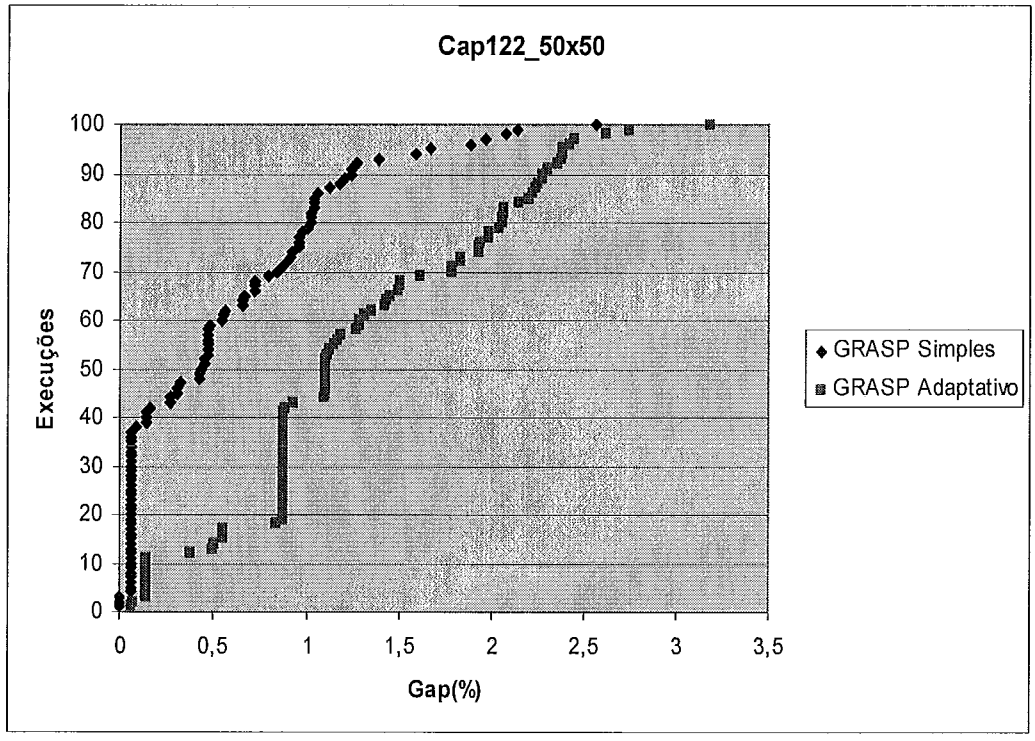

Figura 5: a) Cap122_50x50 Execuções x Gap(%)

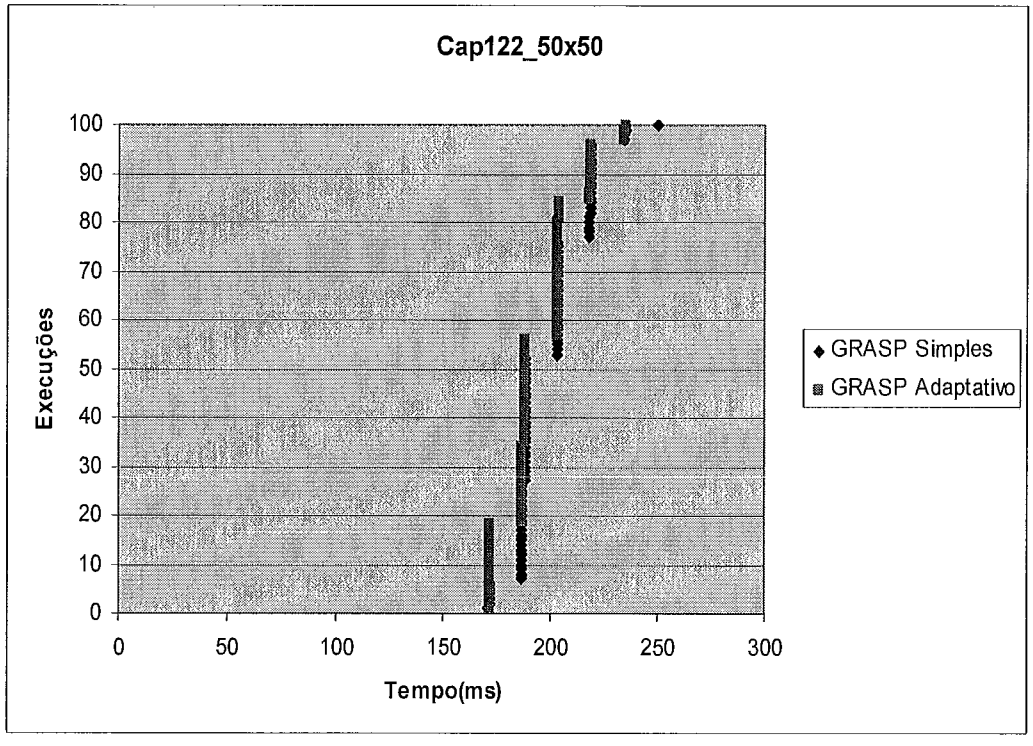

Figura 5: b) Cap122_50x50 Execuções x Tempo (ms)

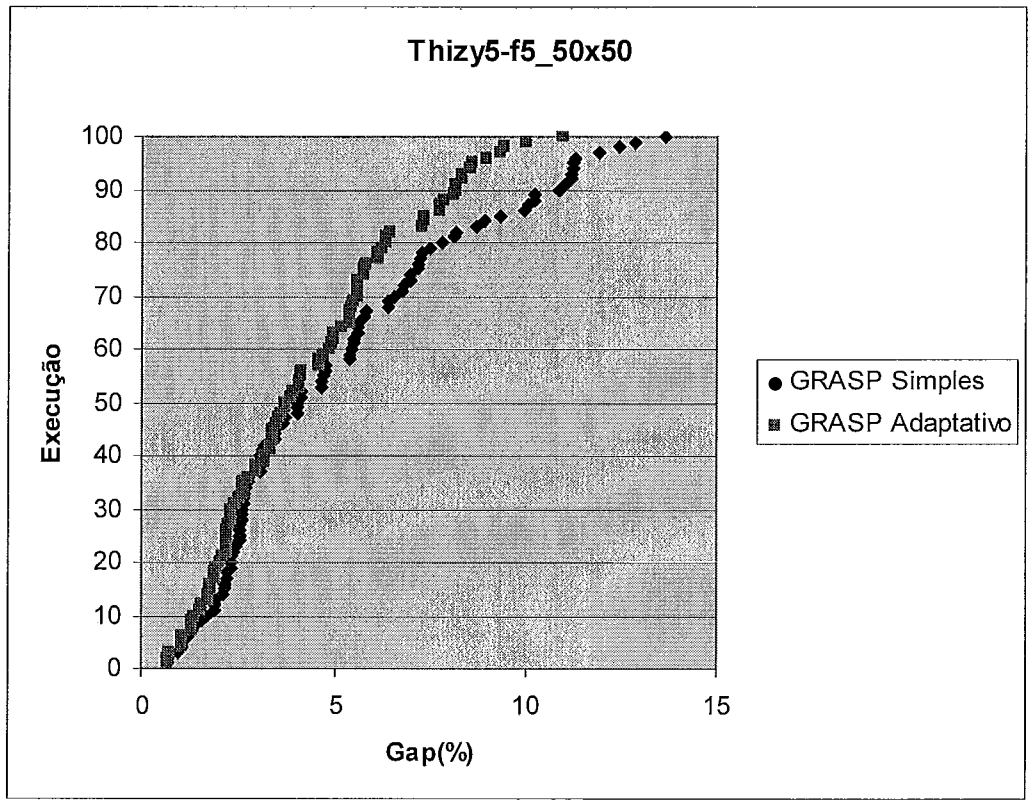

Figura 6: a) Thizy5-f5_50x50 Execuções x Gap(%)

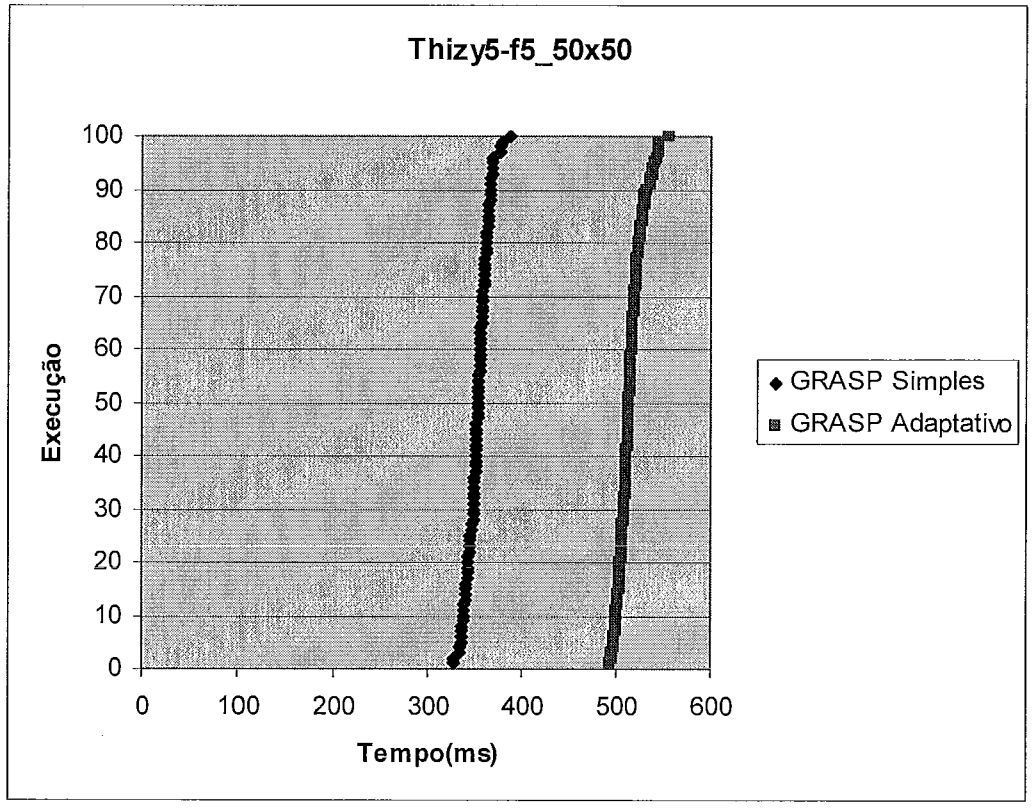

Figura 6: b) Thizy5-f5_50x50 Execuções x Tempo(ms)

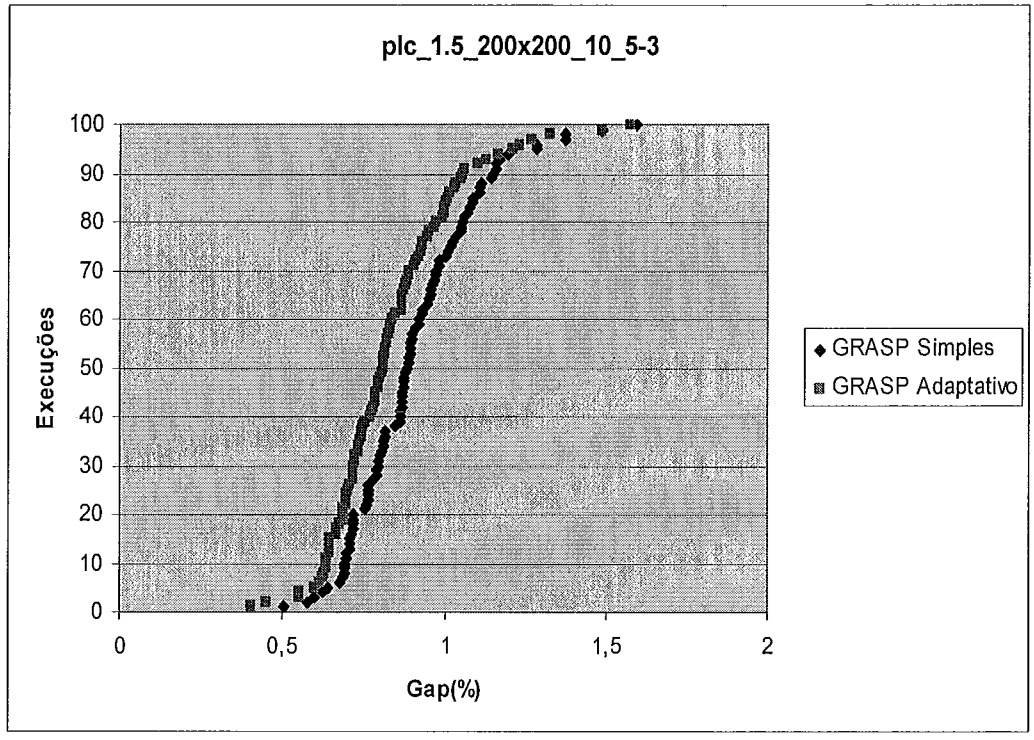

Figura 7: a) plc_1.5_200x200_10_5-3 Execuções x Gap(%)

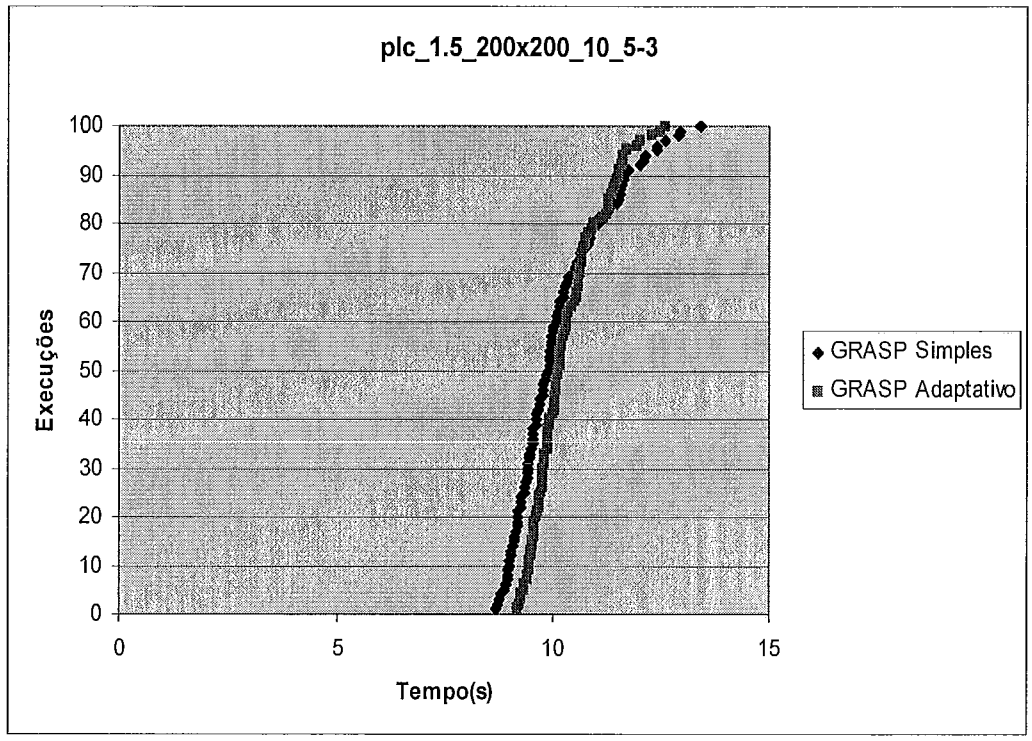

Figura 7: b) plc_1.5_200x200_10_5-3 Execuções x Tempo(s)

6. CONCLUSÕES

O problema de transporte tem sido amplamente estudado na literatura e diversas abordagens ao longo dos anos têm sido usadas para sua resolução. Métodos heurísticos, exatos e aproximativos têm sido usados combinados ou separadamente.

Neste trabalho teve-se a oportunidade de mostrar um novo método híbrido (utilizando meta-heurística e método exato de forma intricada) de resolução para o PLC, onde foi combinado um método exato, o problema de transporte, com uma Meta-heurística, o GRASP.

Nosso trabalho tem a característica inovadora de utilizar um algoritmo guloso, o GRASP, na escolha de facilidades ao construir a solução unida ao problema de transporte, o que deu ao nosso algoritmo uma qualidade muito boa nas suas soluções obtidas. Sua característica polinomial contribui para que seja uma boa escolha para a resolução do problema de localização capacitada em um tempo computacional aceitável.

O algoritmo híbrido desenvolvido está dividido basicamente em duas fases, a primeira constrói uma solução inicial de boa qualidade para o PLC, combinando o problema de transporte com a meta-heurística GRASP. Uma variação no uso do GRASP deu origem a dois algoritmos desenvolvidos que chamamos de GRASP Simples e GRASP Adaptativo.

No GRASP Simples, o intervalo α de facilidades candidatas, que podem entrar na solução é um valor fixo durante a execução do algoritmo. No GRASP Adaptativo foi utilizado um vetor de α's que a cada escolha de uma nova facilidade pode variar.

A segunda fase consiste de duas etapas de trocas, uma chamada de simples troca, em que para cada uma das facilidades candidatas tenta-se efetuar uma troca por vez com o conjunto de facilidades abertas. A segunda etapa da fase de trocas consiste na execução de um algoritmo chamado de multi-trocas, que possui três tipos de operações de melhoria: abrir uma facilidade; abrir uma facilidade e fechar uma ou mais

facilidades; fechar uma facilidade e abrir uma ou mais facilidades. Uma operação de troca pode executar aleatoriamente uma das três operações descritas anteriormente.

O algoritmo se apresentou estável para a maioria das soluções apresentadas, e nas instâncias teste em que se teve a oportunidade de trabalhar melhor a escolha dos parâmetros de entrada, como no caso da OR-LIBRARY, os resultados encontrados foram de ótima qualidade.

Nos problemas teste de Thizy a execução de forma "batch" deixou alguns resultados um pouco a desejar. Foi mostrado que a intervenção do usuário é fundamental para encontrarmos uma boa solução. Contudo, as instâncias teste menores, 8x25, 16x25, 25x25, 16x50, 33x50 e 50x50 de Thizy, apresentaram bons resultados e como foi mostrado para algumas maiores (pex. 200x200) a solução encontrada na execução "batch" pode ser significativamente melhorada caso seja feita uma análise direta (isolando a instância com problema na precisão na resposta, das demais, e assim ajustando os parâmetros do método mais adequados a ela).

Foi feita uma análise de robustez para 4 instâncias: Cap44_16x50, Cap122_50x50, Thizy-f5_50x50 e Plc_1.5_200x200_10_5-3. procurou-se mostrar instâncias boas e não muito boas com o objetivo de apresentar o melhor caso, caso médio e pior caso encontrado pelo algoritmo proposto. Para cada uma das 4 instâncias foram construídos dois gráficos de dispersão, um com o GAP por Execução e outro de Tempo por Execução.

O algoritmo proposto apresentou-se bastante robusto, uma vez que ele consegue manter uma variação de GAP baixa durante muitas execuções. Embora algumas instâncias tenham apresentado resultados ruins para uma meta-heurística que está sujeita a tais variações.

O algoritmo proposto pode ser usado como método de partida para um algoritmo exato, pois apresenta uma solução muito boa servindo como limite superior para a solução exata.

Ademais, a substituição do algoritmo de Ford & Fulkerson por um sucedâneo mais eficiente, de complexidade mais baixa, tornaria todos os tempos consignados neste trabalho ainda melhores.

Este trabalho está servindo de base para um artigo sobre localização capacitada dinâmica de facilidades que está sendo elaborado [NEGREIROS et al, 2007]. Neste problema de localização, existem cenários independentes de custos, ofertas e demandas em um horizonte de planejamento, e deseja-se minimizar o máximo arrependimento por instalação equivocada de facilidades que permanecerão fixas, quando selecionadas, durante todo o horizonte de planejamento.

Tem-se como direção para trabalhos futuros, a possibilidade de paralelização do algoritmo proposto, o desenvolvimento e testes de novos métodos de trocas entre facilidades abertas e o conjunto de facilidades candidatas, que possibilitem melhorar a solução inicial encontrada, e o uso dessa técnica combinada a um método exato ou de relaxação, haja vista sua baixa complexidade média.

7. BIBLIOGRAFIA

[ARARAKI, 2002] Araraki, Reinaldo Gen Ichiro - "Heurística de localização-alocação para problemas de localização de facilidades", INPE/LAC - Tese de Doutorado (2002).

[BARCELOS, PIZZOLATO & LORENA, 2004] Barcelos, F. B.; Pizzolato, N. D. and Lorena,L.A.N. "Localizacao de escolas do ensino fundamental com modelos capacitado e nao-capacitado: caso de Vitoria/ES". Pesquisa Operacional vol. 24 (1): 133 - 149, 2004 - Edicao especial - 60 anos Prof. Roberto Galvão.

[BEASLEY, 1988] Beasley, J. – "An algorithm for solving large capacitated warehouse location problems", EJOR, vcol 33, pgs. 314-325.

[CHUDAK & WILLIAMSON, 1999] Chudak, F. A., D. P. Williamson. 1999. "Improved approximation algorithms for capacitated facility location problems". *Proc. 7th Conf. Integer Programming Combin. Optim. (IPCO)*, 99–113.

[CORMEN & LEISERSON, 2002] CORMEN, Thomas H., LEISERSON, Charles E. Algoritmos: Teoria e Prática. 2ª edição, Rio de janeiro, Editora Campus, 2002

[CORNUEJOLS, SRIDHARAN & THIZY, 1991] 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.

[DANTZIG, 1963] Dantzig, George B., "Linear Programming and Extensions", Princeton University Press.

[DASKIN, 1995] Daskin, M.S. - Network and Discrete Location: Models, Algorithms and Applications, ED. John wiley (1995).

[DELMAIRE, DÍAZ & FERNÁNDEZ, 1999] "Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem", H. Delmaire, J.A. Díaz, E. Fernández, and M. Ortega *INFOR*, 37:194-225, 1999.

[FEO & RESENDE,1995] "Greedy randomized adaptive search procedures", Resende, M.G.C. & Feo,T.A. *Journal of Global Optimization*, 6:109-133, 1995.

[GALVÃO, 2004] "Uncapacitated Facility Location Problems: Contributions." Galvão, R.D. Pesquisa Operacional, v.24, n.1, p.7-38, Janeiro a Abril de 2004.

[GOLDBARG & LUNA, 2000] Goldbarg, Marco Cesar e Luna, Henrique Pacca L. Programação Linear e Otimização Combinatória, ED. Campus (2000).

[JACOBSEN, 1983] Jacobsen, S.K. "Heuristics for the capacitated plant location model", EJOR, vol. 12, pgs. 253-261 (1983).

[KORUPULO, PLAXTON & RAJARAMAN, 1998] Korupolu, M. R., C. G. Plaxton, R. Rajaraman. 1998. "Analysis of a local search heuristic for facility location Problems". *Proc. 9th Annual ACM-SIAM Sympos. Discrete Algorithms* (*SODA*), 1–10.

[KUEHN & HAMBURGER, 1963] Kuehn, A. A., M. J. Hamburger. 1963. "A heuristic program for locating warehouses". *Management Sci.* 9643–666.

[LABBÉ & LOUVEAUX, 1997] Labbé, M. & Louveaux, F.V. "Location Problems", Annoted Bibliographies in Combinatorial Optimization, Ed. Dell´Amico, Maffioli & Martello, JWiley, (1997).

[MAHDIAN, YE & ZHANG, 2002] Mahdian, M., Y. Ye, J. Zhang. 2002. "Improved pproximation algorithms for metric facility location problems". K. Jansen, S. Leonardi, V. Vazirani, eds. *Approximation Algorithms for Combinatorial Optimization, Lecture Notes in Computer Science*, No. 2462. Springer, Berlin, Germany, 229–242.

[MAHDIAN, YE & ZHANG, 2003] Mahdian, M., Y. Ye, J. Zhang. 2003. "A 2-approximation algorithm for the soft-capacitated facility location problem". S. Arora, K. Jansen, Jose D. P. Rolim, Amit Sahai, eds., *Approximation, Randomization, and Combinatorial Optimization, Lecture Notes in Computer Science*, No. 2764. Springer, Berlin, Germany, 129–140.

[NEGREIROS, CARVALHO & MACULAN, 2005] Negreiros Gomes, Marcos J., Carvalho, Jorge B. e Maculan, Nelson F. "Uma Avaliação Experimental da Meta-heurística GRASP Aplicada ao Problema de Localização Não Capacitada", Submetido à Revisa Pesquisa Operacional.

[NEGREIROS, PALHANO & TELES, 2007] Negreiros Gomes, Marcos J., Palhano, Augusto W.C., Virginio, Ronaldo S. F, Teles, Ingrid G. - "O Problema de Localização Capacitada Dinâmica Através de Algoritmos Clássicos"- A ser publicado.

[OR-LIBRARY] http://mscmga.ms.ic.ac.uk/ , acesso via www

[PÁL, TARDOS & WEXLER, 2001] Pál, M., É. Tardos, T. Wexler. 2001. "Facility location with hard capacities". *Proc. 42nd IEEE Sympos. Foundations Comput. Sci. (FOCS)*, 329–338.

[RESENDE, 2001] Resende, M.G.C. "Greedy randomized adaptive search procedures (GRASP)", In C.A. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*, volume 2, pages 373-382. Kluwer Academic Publishers, 2001.

[RIBEIRO & POGGI DE ARAGÃO, 1998] Ribeiro, C.C. & Poggi de Aragão, M.V.- "Meta-heurísticas", Escola Brasileira de Computação, Rio de Janeiro - http://www.inf.puc-rio.br/~poggi

[PIZZOLATO & FRAGA, 1997] Pizzolato, N.D. & Fraga da Silva, H.B. (1997). "The location of public schools: evaluation of practical experiences". *International Transaction in Operations Research*, **4**, 13-22.

[SYSLO, DEO & KOWALIK] Syslo, Maciej M.; Deo, Narsingh e Kowalik, Janusz S. "Discrete Optimization Algorithms", ED. Prentice-Hall (1983).

[SHMOYS, TARDOS & ARDAL, 1997] Shmoys, D. B., É. Tardos, K. I. Aardal. 1997. "Approximation algorithms for facility location problems". *Proc. 29th Annual ACM Sympos. Theory Comput. (STOC)*, 265–274.

[SZWARCFITER & MARKENZON] Szwarcfiter J. L. & Markenzon L., *Estrutura de dados e seus algoritmos*, 2ª edição, Rio de janeiro, editora LTC, 1994.

[TRINDADE, 2005] Trindade, Viviane de Aragão. "Desenvolvimento e Análise Experimental da Metaheurística GRASP para um Problema de Planejamento de Sondas de Manutenção", Dissertação de M.Sc, Programa de Pós-Graduação em Computação – UFF, Niterói, RJ, Brasil, 2005.

[VALIATI, 2006] Valiati D. , "Método ótimo e épsilon-ótimo para resolução do problema de localização capacitado", tese de D.Sc, Engenharia de Sistemas e Computação -COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2006.

[VIRGINIO, NEGREIROS & XAVIER, 2005] Virginio Ronaldo S. F., Negreiros Gomes, Marcos J. & Xavier Adilson E., "Um Algoritmo GRASP Híbrido para o Problema de Localização Capacitada de Custo Fixo". *XXXVII SBPO*, Gramado, RS, Brasil, setembro, 2005.

[WERNECK & RESENDE, 2004] Werneck , R.F. & Resende, M.G.C. , "A hybrid heuristic for the p-median problem", *Journal of Heuristics*, 10:59-88, 2004.

[ZHANG, CHEN & YE, 2005] Zhang, J., B. Chen, Y. Ye. 2005. "A multi-exchange local search algorithm for the capacitated facility location problem", Mathematics of Operations Research, Vol. 30, Nº 2, May 2005, pp. 389-403.