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

Ronaldo Silva Virginio Filho
Programa de Engenharia de Sistemas e Computação / COPPE
Universidade Federal do Rio de Janeiro
Rio de Janeiro – RJ
virginio@cos.ufrj.br

Marcos José Negreiros Gomes
Universidade Estadual do Ceará
Mestrado Profissional em Computação UECE/CEFET-CE
Fortaleza – CE
negreiro@uece.br

Adilson Elias Xavier
Programa de Engenharia de Sistemas e Computação / COPPE
Universidade Federal do Rio de Janeiro
Rio de Janeiro – RJ
adilson@cos.ufrj.br

Resumo
Apresentamos o desenvolvimento de um algoritmo híbrido para o Problema de Localização Capacitada de Custo Fixo (PLC). No PLC a questão central é localizar um conjunto de facilidade (uma ou mais), minimizando o custo global de localizá-las. O algoritmo está baseado no problema de transporte e em uma meta-heurística gulosa, o GRASP, para obter um conjunto solução de facilidades localizadas. Avaliamos duas estratégias GRASP, usando Alfa Fixo e Alfa Variado, onde esta última a denominamos de GRASP Adaptativo (o mesmo que reativo). Para testar os resultados foi utilizada a biblioteca de instâncias do PLC a OR-Library, onde obtivemos resultados que se mantiveram com GAP na média a 1% da solução ótima.
Palavras-Chaves: Localização Capacitada, Meta-heurística, Problema de Transporte, GRASP.

Abstract
In this work an algorithm for the Fixed Cost Capacitated Location Problem (FCCLP), which objective is 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 transport problem and a meta-heuristic GRASP to encompass the 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 a gap from the optimum solution in the average below 1% using our method.
Keywords: Capacitated Location, Meta-heuristics, Transport Problem, GRASP.

Introdução

Este trabalho tem como objetivo a definição de um algoritmo 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. Os problemas de localização vem sendo bastante estudados por diversos pesquisadores. Alguns desses estudos apresentam bons resultados. Considerando apenas aqueles trabalhos relacionados ao emprego da meta-heurística GRASP na resolução de problemas similares de localização, pode-se destacar: [RESENDE, 2001], [RESENDE e WERNEK, 2004], [NEGREIROS et al, 2005].

Este 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 e Relaxação Lagrangeana 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 subproblema a ser resolvido dentro do modelo de solução meta-heurística desenvolvida. O experimento realizado é descrito, sendo mostrado todo o funcionamento do algoritmo híbrido desenvolvido. Na seção 4 são apresentados os resultados computacionais que foram colocados em uma tabela comparando os valores obtidos com a aplicação desenvolvida e as instâncias da OR-Library para o PLC. Na seção 5 é apresentada a conclusão do trabalho.

Problema de Localização Capacitada de custo fixo

Modelo

O problema de Localização Capacitada de Custo Fixo busca definir um conjunto de facilidades (de um conjunto de candidatas) 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.

Este problema é bastante estudado pela literatura, tendo merecido diversos trabalhos sobre sua solução via métodos exatos e heurísticos, [NEGREIROS et al 2005]. Este problema é dito ser do tipo NP-Árduo, uma vez que somente instâncias de pequeno porte $(n+m)$, onde *n* é o número de clientes e *m* o número de facilidades candidatas, podem ser resolvidas em tempo aceitável. Instâncias de grande porte (500 x 1000) também foram resolvidas, [BEASLEY, 1988].

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

(PLC) Minimizar $\sum_{j=1}^{m} f_j y_j + \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_{ij} c_{ij} x_{ij}$ (1.1)

Sujeito a:

$\sum_{j=1}^{m} x_{ij} = 1,\ i = 1,...,n.$ (1.2)

$x_{ij} \le y_j,\ i = 1,...,n \text{ e } j = 1,...,m.$ (1.3)

$\sum_{i=1}^{n} q_i x_{ij} \le Q_j y_j,\ j = 1,...,m$ (1.4)

$x_{ij} \ge 0,\quad i = 1,...,n \text{ e } j = 1,...,m.$ (1.5)

$y_j \in \{0,1\},\ j = 1,...,m$ (1.6)

onde,

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 ($/uq) 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) (dxq) para enviar um produto do cliente *i* à facilidade *j*, *ou seja:* $c_{ij} = d_{ij}.q_i$;

Variáveis:

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

$y_j = \begin{cases} 1, & \text{Se o cliente } i \text{ é atendido pela facilidade } j \\ 0, & \text{Caso Contrário} \end{cases}$

No modelo do PLC temos na função objetivo (1.1) a necessidade de encontrar a solução de menor custo de localização e de taxa-custo global do transporte, que atende às restrições de demanda do cliente são integralmente atendidas (1.2), as restrições (1.3) consideram que a demanda no cliente *i* só

pode ser atribuída à facilidade *j* desde que *j* esteja localizada, a restrição (1.4) assegura que a demanda do cliente *i* não está designada a facilidade candidata a localização se esta facilidade não for selecionada, e as restrições (1.5) e (1.6) garantem a necessidade da seleção única entre clientes-facilidades e facilidades candidatas (restrições de integralidade e não negatividade). O modelo PLC acima proposto pode ser visto em [DASKIN, 1995] e [LABBÉ & LOUVEAUX, 1997].
Alguns métodos importantes, entre exatos e heurísticos e exatos, são descritos na literatura para o PLC [GALVÃO, 2004]. [BEASLEY, 1993]. Em seguida apresentamos uma breve descrição de alguns desses métodos.

Métodos heurísticos e exatos
Dentre os métodos heurísticos, os construtivos generalizados para o PLC são o ADD e DROP, [JACOBSEN, 1983]. Ambos são heurísticas gulosas. Sumariamente, temos:
ADD [KUEHN & HAMBURGER, 1963]
A construção da solução para o PLC utilizando o método ADD funciona 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 na primeira iteração e 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 F é removida do conjunto F e passa a pertencer ao conjunto L e uma nova iteração é realizada enquanto existir facilidades no conjunto F e $c_{tot} > c_{it}$, caso falso, as iterações param e o resultado final é c_{tot} e L.
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. Seja F um conjunto de facilidades removidas do conjunto L, sendo L o conjunto das facilidades localizadas. Uma facilidade de L é escolhida para sair do conjunto L na primeira iteração e 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 e 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 c_{tot} e L.
Heurísticas Lagrangeanas [BEASLEY, 1993]
Usa o problema de transporte como base de solução de escolhas aleatórias de conjuntos de facilidades que cobrem o conjunto de clientes, e vai atualizando os multiplicadores a partir de seleções orientadas de candidatos, [CORNUEJOLS et al 1991] *ap ud* [BEASLEY, 1993].
Já para os métodos exatos, dentre os importantes métodos, temos:
[BEASLEY, 1988] Descreve um algoritmo exato para solução do problema de localização capacitada de custo fixo utilizando Relaxação Lagrangeana.
[DASKIN, 1995] Também demonstra um método exato para o PLC baseado em relaxação Lagrangeana e Branch & Bound.

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 selecionar 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, utilizamos 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 construímos uma estratégia gulosa que permite gerar soluções viáveis para o PLC. Após um número de execuções e com a melhor solução gulosa entre as selecionadas, fazemos trocas locais que nos conduzem ao resultado final do processo. Nessa linha de raciocínio, temos então as seguintes metodologias a atender.

O Problema de Transporte Clássico

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

$$Z=\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad (2.1)$$

Sujeito a:

$$\sum_{j=1}^{n} x_{ij} \le a_i\ ,\ i=1,...,n\ . \qquad (2.2)$$

$$\sum_{i=1}^{m} x_{ij} \ge b_J\ ,\ j=1,...,m \qquad (2.3)$$

$$x_{ij} \ge 0\ ,\quad i=1,...,n\ ;\ j=1,...,m \qquad (2.4)$$

Onde,

c_{ij} = custo de distribuição 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, devemos ter o total armazenado igual ao total da produção. Isso pode ser definido pela equação (2.5).

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \qquad (2.5)$$

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* x n) variáveis de decisão.

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. No caso utilizamos o método de Ford-Fulkerson, [SYSLO, DEO e KOWALIK, 1988].

Meta-heurística GRASP

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

GRASP

f(s*) ← +∞
for i = 1,…,N **do**
  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'*
end-for
onde,
f(s*) é o valor da função objetivo, sobre a solução ótima s*;
f(s') é o valor da solução corrente obtida;
N é o número de iterações;

Algoritmo 1: Algoritmo genérico do GRASP.

No passo 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 randomizada entre as possíveis escolhas. No passo 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, no passo 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 e do método guloso de busca). 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), temos 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 & RESENDE, 1995].
A grande vantagem do GRASP está justamente no aproveitamento do desempenho das heurísticas gulosas [NEGREIROS et al 2005], geralmente polinomiais de baixo índice. 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 maiores.
O modo no processo de escolha de uma facilidade candidata na etapa 1 do GRASP, que no caso do problema de localização capacitada de custo fixo, é a de aleatorização da facilidade candidata a entrar na solução *s*', pode ocorrer de duas maneiras neste trabalho, as quais são chamadas de GRASP Simples e Adaptativo, cujas formas de escolha da facilidade candidata e diferenças serão explicadas a seguir. 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,

Escolha Aleatória GRASP Simples

Repeat
i := **Random** (|F|);
Until verifica se i está no intervalo α solicitado;
Return f_i;

Algoritmo 2: Busca na lista de candidatos com alfa simples

O algoritmo 2 ilustra o modo como o GRASP escolhe uma facilidade candidata a fazer parte da solução *s*', que funciona da seguinte forma, primeiro é encontrado um índice randomizado entre os valores 0 à $|F|$, este passo é repetido até que o índice randomizado 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 Reativo (Adaptativo), ou "Reative GRASP", ao invés de um valor de α que é 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].

Monta Vetor GRASP Reativo

for i = 1,…, $|G|$ **do**

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

Algoritmo 3: Montagem de um vetor de Alfas dentro de um intervalo [1,100]

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 4 mostrado abaixo, sendo os passos seguintes à escolha do valor α com o qual se vai trabalhar, praticamente, os mesmos do algoritmo 2.

Escolha Aleatória GRASP Reativo

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

α := G_i;

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

Return f_i;

Algoritmo 4: Seleção de um candidato da lista para um conjunto de valores de alfa dados.

Descrição do Algoritmo Híbrido

Apesar da existência de heurísticas conhecidas para o problema, ADD, DROP e outras, como vimos anteriormente, na montagem da solução GRASP nós desenvolvemos uma nova metodologia para o problema. Nossa metodologia baseia-se em um algoritmo híbrido – utilizando Métodos Exatos (Problema de Transporte) e Meta-heurística (GRASP), que na verdade combinam-se para encontrar soluções próximas à ótima do problema geral de Localização Capacitada, em tempo razoável (polinomial). Passaremos a seguir a relatar o nosso procedimento 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.

1° Passo: 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.

2º Passo: Montar um vetor de custos por facilidade, ordenado ascendentemente pelo valor do custo relativo de transporte por unidade, 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.

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 a 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 nó de oferta com o valor que falta para igualar com a demanda, sendo atribuído 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 nó 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 nó de oferta quanto de demanda que são adicionados, não entram para solução do problema de localização, são apenas usados para balancear a quantidade ofertada e a

demanda. O custo relativo de transporte (3.1) indica quanto é gasto em média para transportar cada unidade ofertada de uma facilidade até o cliente. Esse valor é obtido da seguinte forma:
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.

$$CRT_i = (CT_i + CF_i) / K_i \qquad (3.1)$$

3º Passo: Encontra uma solução para o problema de transporte aplicando-se o GRASP Simples ou o GRASP Adaptativo. No 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 α, sendo que este valor é fornecido pelo usuário no GRASP Simples. 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.
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. Executa o algoritmo de transporte de acordo com o caso entre o valor de oferta e demanda.
Atribui o custo do transporte adicionado ao custo fixo das facilidades localizadas e o custo fixo da facilidade candidata a uma variável auxiliar que guarda o custo da iteração e depois 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.
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.
Verifica se toda a demanda já foi atendida, caso tenha sido então uma solução já foi encontrada e continua o processo de inserção de localidades até que o custo total com as facilidades localizadas aumente, senão volta ao item 1 deste passo. A figura 1 ilustra a melhoria gradual da solução à medida que vão sendo localizadas novas facilidades, utilizando o procedimento que elaboramos. Mantém o processo de inserção de facilidades, continuando o cálculo de custo variável e fixo via Problema de Transporte, até que a inclusão de uma nova facilidade aumente o custo global de localização, descartando-a e concluindo o processo de adição de facilidades.

4º Passo: Pegar a solução 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. Para isso foi criado um procedimento que faz uma troca por vez entre estes conjuntos e que funciona da seguinte forma:
Pega-se uma facilidade candidata por vez e executa-se um procedimento que escolhe uma facilidade localizada para sair da solução e uma facilidade candidata a entrar, faz o balanceamento para execução do problema de transporte, e retorna o novo custo.
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. Se for menor mantém o custo global e o conjunto da solução como estava.
Repete os itens a e b para cada facilidade candidata em busca de melhorias na solução.

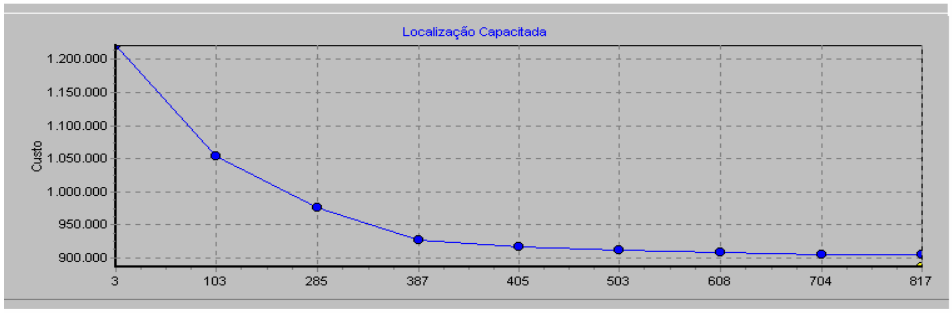

Figura 1: Função de decréscimo de Custo por acréscimo de facilidades localizadas (ADD).

5º Passo: Imprimir o resultado mostrando o valor do custo com e sem as trocas e as facilidades localizadas, o tempo de execução. Os valores do custo antes de serem mostrados no final, são recalculados utilizando-se uma matriz de custo com valores reais, antes estes custos estavam sendo calculados com valores inteiros arredondados, pois, o algoritmo de Ford-Fulkerson trabalha com números inteiros. Com isso o resultados finais ficam mais próximos da realidade.

Resultados Computacionais

Apresentação da Aplicação

Uma aplicação foi desenvolvida implementando o algoritmo para o problema de localização capacitada de custo fixo descrito neste trabalho, utilizando-se o ambiente de programação Delphi 7.0 (linguagem Pascal), sistema operacional Windows 98 (Microsoft) e um computador Athlon XP 1.8 MHZ com 256MB.

O segundo passo está em especificar os números de iterações e corridas que se deseja executar, e selecionar na área “GRASP” qual método será utilizado para resolução do problema. Se a escolha for o GRASP Simples, então é necessário especificar um valor α que será utilizado na seleção das facilidades candidatas, e pode-se escolher ainda os valores de α melhores, com o intervalo [1, α] e o médio, com o intervalo [50-$\alpha/2$,100-$\alpha/2$]. Caso a escolha seja o GRASP Adaptativo, então a área “intervalo” ficará desabilitada, pois não é necessário escolher valores para α, uma vez que é montado um conjunto de α’s de cardinalidade 10 e intervalo [50,100] para cada valor α.

Tabela de Resultados

As tabelas 1 e 2 apresentam os resultados dos experimentos gerados com esta versão de GRASP que elaboramos, para os casos simples e reativo (Adaptativo), e comparamos com os resultados das instâncias da OR-LIBRARY para o PLC. Nas tabelas 1 e 2, são os seguintes os parâmetros utilizados:

$|F|$ é o número de facilidades candidatas;

$|C|$ é o número de clientes;

$|L|$ é o número de facilidades localizadas;

GAP é a diferença percentual entre o valor de referência (OR-Library) e o obtido pelo algoritmo;
OR-Library é o valor de referência ótimo para a instância executada.

Para cada instância da OR-Library o aplicativo executou com um valor α igual a 90% para o GRASP Simples, já para o GRASP Adaptativo o valor de α foi definido no intervalo de [50,100] para todas as instâncias analisadas, como já dissemos anteriormente. Os dados mostrados são referentes ao melhor resultado entre 10 execuções para cada instância. O tempo informado é o tempo da instância que obteve o melhor resultado em custo global.

Na tabela 1, analisando os resultados do algoritmo implementado usando GRASP Simples e Adaptativo, pode-se observar uma relativa melhora no desempenho, tanto em tempo de execução quanto ao GAP, para a maioria das instâncias que utilizaram o GRASP Reativo. Na tabela 2 observa-se que em média o GRASP Reativo encontrou soluções um pouco melhores do que as encontradas no

GRASP Simples, sendo que ambas soluções ficaram com GAP abaixo de 0,75% na média. Destaca-se que as instâncias testadas na tabela 2 são relativamente grandes e que algumas soluções alcançaram o valor ótimo dentre elas: Cap A – IV, Cap B – I e Cap C – III.
Os valores encontrados no geral apresentaram uma qualidade muito boa como estão mostrados nas tabelas de resultados, sendo que o pior valor, instância Cap93, ficou apenas 2,52% acima do valor de referência, e em outros casos pode-se dizer que a solução ótima foi alcançada, pois o valor ficou 0,00% comparado com o valor de referência, fato que pode ser explicado pela precisão utilizada (simples) e os arredondamentos feitos pela própria máquina.

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

GRASP			Simples			Reativo			
Instância	$\|F\|$	$\|C\|$	$\|L\|$	GAP%	Tempo (s)	$\|L\|$	GAP %	Tempo (s)	OR-Library
Cap41	16	50	12	0,31	0,691	12	0,31	0,682	1040444.375
Cap42	16	50	12	0,08	0,699	12	0,06	0,688	1098000.450
Cap43	16	50	12	0,08	0,696	12	0,06	0,688	1153000.450
Cap44	16	50	12	0,06	0,698	12	0,06	0,690	1235500.450
Cap51	16	50	7	0,58	0,359	7	0,58	0,336	1025208.225
Cap61	16	50	10	0,14	0,272	10	0,14	0,280	932615.750
Cap62	16	50	9	0,00	0,270	9	0,00	0,269	977799.400
Cap63	16	50	5	0,16	0,257	5	0,16	0,241	1014062.050
Cap64	16	50	5	0,00	0,258	5	0,00	0,240	1045650.250
Cap71	16	50	10	0,24	0,200	10	0,24	0,198	932615.750
Cap72	16	50	9	0,10	0,205	9	0,10	0,201	977799.400
Cap73	16	50	5	0,00	0,149	5	0,00	0,151	1010641.450
Cap74	16	50	3	0,37	0,137	4	0,00	0,148	1034976.975
Cap81	25	50	17	0,65	0,976	16	0,12	0,973	838499.288
Cap82	25	50	13	0,93	0,955	14	0,86	0,956	910889.563
Cap83	25	50	12	1,88	0,971	12	1,88	0,971	975889.563
Cap91	25	50	12	1,09	0,571	13	1,01	0,521	796648.438
Cap92	25	50	10	1,17	0,515	10	1,17	0,488	855733.500
Cap93	25	50	7	2,52	0,434	7	0,73	0,393	896617.538
Cap94	25	50	6	1,51	0,390	6	1,51	0,378	946051.325
Cap101	25	50	12	0,68	0,417	13	0,55	0,409	796648.437
Cap102	25	50	9	1,19	0,360	10	0,24	0,358	854704.200
Cap103	25	50	6	2,41	0,249	6	0,66	0,258	893782.112
Cap104	25	50	3	1,78	0,195	4	0,00	0,208	928941.750
Cap111	50	50	18	0,44	2,119	17	0,20	2,020	826124.713
Cap112	50	50	16	1,33	2,170	15	0,67	2,037	901377.213
Cap113	50	50	15	1,30	2,042	14	0,49	2,181	970567.750
Cap114	50	50	12	0,93	2,230	13	0,62	2,167	1063356.488
Cap121	50	50	14	1,78	1,202	13	0,42	1,096	793439.563
Cap122	50	50	12	2,26	1,119	12	0,38	1,145	852524.625
Cap123	50	50	8	1,15	0,831	9	0,33	0,837	895302.325
Cap124	50	50	8	1,83	0,842	7	0,29	0,757	946051.325
Cap131	50	50	14	1,91	1,122	13	0,39	0,998	793439.562
Cap132	50	50	10	2,16	0,883	10	0,31	0,868	851495.325
Cap133	50	50	6	0,57	0,527	7	0,64	0,628	893076.712
Cap134	50	50	4	0,00	0,438	4	0,06	0,404	928941.750

Durante a realização dos testes pode-se observar que se aumentarmos o valor de α do GRASP Simples para valores maiores que 90% a qualidade das soluções encontradas também aumentam e mantém-se mais constantes ao longo das execuções. Com isso, a solução da instância do Cap93 passa a ter um GAP de 1,59% para um α=95% e 0,17% para um α=100%.

Tabela 2: 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	0,36	131,9	7	0,1	137,8	19.240.822,449
Cap A – II	100	1000	6	0,73	122,8	6	0,73	120,6	18.438046,543
Cap A - III	100	1000	5	0,33	121,0	5	0,31	110	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	0,24	174,1	11	0,00	177,3	13.656379,578
Cap B – II	100	1000	9	0,38	164,6	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	0,66	192,2	11	0,66	178,6	11.646596,974
Cap C – II	100	1000	10	0,17	138,1	10	0,17	137,8	11.570340,289
Cap C – III	100	1000	9	0,00	143,3	9	0,00	136,4	11.518743,744
Cap C – IV	100	1000	9	0,41	134,5	9	0,74	118,3	11.505767,394

Fazendo-se uma comparação com relação ao tempo de execução da aplicação desenvolvida com a descrita em [BEASLEY, 1988] para algumas instâncias que encontram o valor ótimo, como mostrado na tabela 1, temos:

Tabela 3: Tempo sobre as instâncias da OR-Library

Instância	Aplicação		[BEASLEY, 1988]
	Método	Tempo(s)	Tempo(s)
Cap64	GRASP Simples	0,258	0,3
Cap64	GRASP Adaptativo	0,240	
Cap73	GRASP Simples	0,149	0,2
Cap73	GRASP Adaptativo	0,151	
Cap74	GRASP Adaptativo	0,148	0,1
Cap104	GRASP Adaptativo	0,208	0,2
Cap134	GRASP Simples	0,438	0,7

Analisando-se os tempos de execução da tabela 3 podemos ver que nossa aplicação conseguiu resolver o PLC com tempos computacionais bons utilizando um computador Athlon XP 1.8 Mhz, enquanto que em [BEASLEY, 1988] foi utilizado um computador CRAY-1S, apesar deste ser um equipamento de meados dos anos 80.

Conclusão

Neste trabalho foram apresentados um algoritmo híbrido para encontrar soluções para o Problema de Localização Capacitada de Custo Fixo, e também um sistema de testes desenvolvido para experimentar instâncias da OR-Library.

Os resultados encontrados na resolução de várias instâncias da OR-Library apresentaram-se muito bons, tanto para o GRASP Simples quanto para o GRASP Reativo(Adaptativo), sendo que este último apresentou no geral resultados melhores e com um tempo computacional menor.

Para um trabalho futuro poderá ser feita uma implementação utilizando a idéia do algoritmo DROP anteriormente descrito e também a implementação de uma outra forma de fazer trocas no conjunto solução de facilidades localizadas em busca de uma solução local melhor, utilizando-se técnicas de path-relinking, [WERNECK & RESENDE, 2004].

Bibliografia

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

[BEASLEY, 1993] Beasley, J. – “Lagrangean Heuristics for Location Problems”, EJOR, vol 65, pgs. 383-399.

[CORNUEJOLS et al 1991] Cornuejols, G., Sridharan, R., and Thizy, J.M., "A comparison of heuristics and relaxations for the capacitated plant location problem", *European Journal of Operational Research* 50 pgs 280-297.

[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.

[FELDMAN, LEHRER & RAY, 1966] Feldman, E., Lehrer, F.A & Ray, T.L – “Warehouse location under continuous economies of scale”, Management Sciences, vol. 12(9), pgs 670-684 (1996)

[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.

[KUEHN & HAMBURGER, 1963] Kuehn, AA & Hamburger, M.J. , ``A heuristic program for locating warehouses´´, Management Sciences, vol 11(2), pgs 213-235 (1963)

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

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

[NEGREIROS et al, 2005] Negreiros Gomes, Marcos J., Carvalho, Jorge B. e Maculan, "Uma Avaliação Experimental da Meta-heurística GRASP Aplicada ao Problema de Localização Não Capacitado", Submetido à Revisa Pesquisa Operacional.

[OR-LIBRARY] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/capinfo.html em 06/05/2005.

[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 em 06/05/2005.

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

[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.