MÉTODO PRÁTICO PARA RESOLUÇÃO DO PROBLEMA DE LOCALIZAÇÃO DE FACILIDADES CAPACITADO

José Lassance de Castro Silva
Universidade Federal do Ceará, lassance@lia.ufc.br

Osvaldo Moreira de Oliveira Filho
Universidade Federal do Ceará, Osvaldo@ufc.br

Resumo

Neste trabalho abordamos o problema de localização de facilidades capacitado, com custos lineares de transporte e fixos de instalação, utilizados na resolução de um problema similar no projeto GASLOG. Usaremos alguns métodos simples, práticos e computacionalmente baratos para instalar as facilidades, resolvendo, em seguida, o problema de transporte associado às facilidades instaladas. Posteriormente, analisaremos, os resultados computacionais obtidos com a aplicação do método aos problemas encontrados na literatura e específico, observando sua viabilidade prática.
Palavras chaves: Problema de Localização de Facilidades, Problema de Transporte, Heurística.

Abstract

In this work, we examine the capacitated plant location problem with linear trasportation and fixed location costs, used in the resolution of a similar problem in the GASLOG project. We will use some methods simple, practical and cheap computationaly to install the plants, solving after the transport problem associated to the plants installed. We analyze the computational results obtained with the application of the method to the problems found in the literature and specific, observing, then, your practical viability.
Key-Words: The Capacitated Plant Location Problem, The Transport Problem, Heuristic.

1. Introdução

O Problema de Localização de Facilidades Capacitado é de extrema importância para muitas organizações, já que a instalação de facilidades, como fábricas ou depósitos, implica diferentes custos de distribuição de produtos, afetando seus lucros. Tais custos estão relacionados à instalação de facilidades e ao transporte do produto ou serviço considerado, e geralmente variam de acordo com o porte e a localização das facilidades. Várias questões associadas ao problema de localização de facilidades têm surgido na literatura, conforme podemos encontrar em Francis, McGinnis e White [1], Aikens [2] e Sridharan [3], entre outros.

Considere um conjunto de facilidades $I = \{1, \ldots, m\}$ e um conjunto de consumidores (ou clientes) $J = \{1, \ldots, n\}$. Sejam a_i a capacidade (oferta) associada à facilidade $i \in I$, b_j a demanda total associada ao consumidor $j \in J$, f_i o custo fixo de instalação da facilidade $i \in I$, e c_{ij} o custo linear de transporte unitário da facilidade $i \in I$ ao consumidor $j \in J$. O Problema de Localização de Facilidades Capacitado, denominado daqui em diante de *PLC*, consiste em encontrar um conjunto de facilidades $P \subseteq I$ que atenda à demanda total de todos os consumidores em J, de modo a minimizar o custo total. Sendo assim, o PLC pode ser formulado como um modelo matemático tendo a seguinte forma:

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

Sujeito a:

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

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

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

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

onde x_{ij} é a quantidade de unidades do produto enviada da facilidade $i \in I$ para o consumidor $j \in J$, e y_i representa uma variável binária que indica a instalação ou não da facilidade $i \in I$ ($y_i = 1$ ou $y_i = 0$, respectivamente). A função objetiva (1), a qual se quer minimizar, representa o custo total de instalação da facilidade e de transporte. A restrição (2) assegura que nenhum consumidor é atendido por uma facilidade que não foi instalada e que sua oferta não ultrapasse a quantidade disponível na mesma. A restrição (3) indica a exigência de que a demanda de cada consumidor seja satisfeita. A restrição (4) representa a não negatividade das variáveis x_{ij}, para qualquer facilidade $i \in I$ e qualquer consumidor $j \in J$. Finalmente, a restrição (5) exige a integralidade (binária) das variáveis y_i para toda facilidade $i \in I$.

O PLC é um problema de natureza combinatória classificado como NP-difícil, conforme descreve Sridharan [3]. Devido à sua natureza, a obtenção computacional da solução ótima para instâncias de médio e grande porte é, algumas vezes, intratável, por exigir uma grande quantidade de recursos computacionais. O tempo de resolução para essas instâncias se torna, impraticável, devido principalmente à complexidade do problema. Por esses motivos, os algoritmos exatos, aqueles que determinam a solução ótima do problema, nem sempre conseguem resolvê-lo em tempo hábil. Daí o fato de utilizarmos algoritmos aproximativos, aqueles que encontram boas soluções num tempo computacional aceitável, na resolução do PLC. Por sua vez, esses algoritmos não garantem determinar a solução ótima, mas apresentam boas soluções na prática, e não se consegue mostrar a existência de um limite, para a razão entre o valor da solução obtida e o valor da solução ótima. Apesar disso, suas implementações costumam computar, em tempo hábil e utilizando uma quantidade de memória razoável, uma boa solução viável para ser aplicada ao problema.

Algoritmos para resolver o PLC de maneira exata podem ser encontrados em Bartezzaghi, Colorni e Palermo [4], Baker [5], Beasley [6], Van Roy [7], e Valiati e Bornstein [8]. Algoritmos aproximativos podem ser examinados em Bornstein e Azlán [9], Bornstein e Mateus [10], Jacobsen [11] e Beasley [12].

Neste trabalho estamos interessados em mostrar como foi realizada a pesquisa no desenvolvimento de uma das tarefas do Projeto GASLOG, onde realizamos uma aplicação do PLC. O projeto visa desenvolver um sistema computacional de gestão integrada da distribuição do gás natural (GASLOG), de aperfeiçoamento contínuo e crescente abrangência, para aplicação em cidades das regiões Norte e Nordeste do Brasil; este sistema permitirá a agregação de módulos desenvolvidos com diversas finalidades, tais como:

- Identificar a localização espacial ótima (em termos de custos de distribuição mínimos) de fontes primárias e secundárias de gás natural no âmbito dos estados das regiões Norte e Nordeste a partir da rede dutoviária existente;
- Monitorar a rede de gás natural, via rastreamento digital com sensores, medindo em tempo real parâmetros como volume, temperatura, pressão etc visando o controle operacional e a otimização da segurança na distribuição;
- Rastrear, qualificar e quantificar demandas potenciais de gás natural com vistas a subsidiar a elaboração de programações de oferta e planos de distribuição otimizada de gás natural para pequenas, médias e grandes unidades de consumo (residencial e empresarial) no âmbito das Regiões Metropolitanas e principais cidades do interior dos Estados das Regiões Norte e Nordeste;

· Utilizar uma base de dados para consulta geral sobre atores envolvidos com a distribuição e uso do gás natural na região Norte e Nordeste do Brasil, utilizando ambientes de Internet e Intranet.

Além disso, no GASLOG pretende-se também elaborar os planos de expansão das redes primária e secundária de distribuição do gás natural nas cidades da Região através da montagem da base digital georeferenciada de dados para suporte à gestão otimizada da distribuição do gás natural. Portanto vamos mostrar uma técnica de resolução para o PLC, que pode ser aplicada na resolução do problema específico do GASLOG, que utiliza a Internet como ferramenta de interatividade com o usuário.

A organização deste trabalho segue à: Na Seção 2, apresentaremos a técnica, desenvolvida por nós, aplicada na resolução do PLC. Na seção 3 descrevemos uma heurística usada na resolução do problema de transporte. Na Seção 4, descrevemos os experimentos computacionais através dos problemas gerados pela literatura. As conclusões estão apresentadas na seção 5, enquanto a bibliografia será apresentada na Seção 6.

2. Resolução do PLC

No PLC, diremos que uma facilidade $i \in I$ está *aberta* se ela foi instalada, ou seja, se $y_i = 1$, e que está *fechada* se ela não foi instalada, ou seja, se $y_i = 0$. Seja I_0 e I_1 uma partição de I ($I_0 \cup I_1 = I$ e $I_0 \cap I_1 = \varnothing$), tal que $I_0 = \{i \in I \mid y_i = 0\}$ e $I_1 = \{i \in I \mid y_i = 1\}$, ou seja, I_0 e I_1 são os conjuntos das facilidades fechadas e abertas, respectivamente. O vetor $y \in \{0, 1\}^m$ associado à partição, é tal que $y_i = 0$ se e somente se $i \in I_0$, e $y_i = 1$ se e somente se $i \in I_1$. Observe ainda que y é exatamente o vetor característico referente à instalação das facilidades.

Dado um PLC, e fixando o conjunto de facilidades de modo a obter uma partição $I_0 \cup I_1$ conforme descrito, obteremos o seguinte modelo matemático associado ao Problema de Transporte (*PT*):

$$(\text{PT}) \qquad \min \sum_{i \in I_1} \sum_{j \in J} c_{ij} x_{ij} \qquad (6)$$

Sujeito a:

$$\sum_{j \in J} x_{ij} \le a_i \quad , \forall i \in I_1 \qquad (7)$$

$$\sum_{i \in I_1} x_{ij} = b_j \quad , \forall j \in J \qquad (8)$$

$$x_{ij} \ge 0 \quad , \forall i \in I_1, \forall j \in J \qquad (9) \ ,$$

Um PT pode ser resolvido de forma geral através do método simplex ou uma de suas variações tal como o método *modi*, desenvolvido por Dantzig, descrito em Hadley [13]. A proposição abaixo apresenta um resultado importante para o desenvolvimento do nosso método, cuja a demonstração não é apresentada por ser facilmente verificada.

Proposição: O conjunto de facilidades abertas I_1, é capaz de suprir a demanda total do sistema ($\sum_{i \in I_1} a_i \ge \sum_{j \in J} b_j$), se e somente se a partição $I_0 \cup I_1 = I$ gera, um PT P_t, cuja solução, juntamente com a partição, corresponde a uma solução viável do PLC associado com valor da função objetivo igual ao valor ótimo de P_t acrescido de $\sum_{i \in I_1} f_i y_i$.

Assim sendo, podemos resolver o PLC da seguinte forma:

1. Seja $\wp$ a família de todas as possíveis partições distintas de I em facilidades abertas e fechadas.
2. Para cada partição $P \in \wp$ capaz de suprir a demanda total do sistema, resolva o PT associado ao PLC e a P, guardando sempre a melhor solução obtida para o PLC.

O procedimento acima é exato, porém exponencial e difícil de ser executado na prática. Entretanto, em vez de usarmos $\wp$, podemos tomar um subconjunto R de $\wp$ de tamanho razoável e aplicar o procedimento acima, para diminuir a complexidade, e tentar encontrar sua solução ótima. A escolha de R influi diretamente no tempo e espaço de memória necessária para a resolução do problema. A seguir, será abordado um método para a construção de um R de tamanho fixo.

Instalando Facilidades

Cada membro de $\wp$ corresponde a um único vetor característico y referente à instalação ou não das facilidades. Vamos, então, construir dez vetores denominados de y^1, y^2, ..., y^{10}, onde cada vetor desse representa um elemento de R. A seguir apresentamos a construção dos dez vetores:

$\boldsymbol{y^1}$: Seja $\langle a_{k_1},...,a_{k_m} \rangle$ a ordem não-decrescente dos valores das capacidades das facilidades. Inicialmente, seja $y^1 = \mathbf{0}$. Faça, então, $y^1_{k_i}=1$, $\forall$ $1 \le i \le p$, tal que $\sum_{i=1}^{p} a_{k_i} \ge \sum_{j=1}^{n} b_j$ e $\sum_{i=1}^{p-1} a_{k_i} < \sum_{j=1}^{n} b_j$, ou seja, instale as facilidades de menor capacidade até que o conjunto das facilidades abertas seja capaz de suprir a demanda total dos clientes. Este vetor dá prioridade às facilidades de pequeno porte, que geralmente têm custo fixo de instalação menor que as facilidades de grande porte.

$\boldsymbol{y^2}$: Seja a mesma ordem não-decrescente $\langle a_{k_1},...,a_{k_m} \rangle$ obtida durante a construção do vetor anterior. Inicialmente, seja $y^2 = \mathbf{0}$. Faça, então, $y^2_{k_i} = 1$, $\forall$ $p \le i \le m$, tal que $\sum_{i=p}^{m} a_{k_i} \ge \sum_{j=1}^{n} b_j$ e $\sum_{i=p-1}^{m} a_{k_i} < \sum_{j=1}^{n} b_j$, ou seja, instale as facilidades de maior capacidade até que o conjunto das facilidades abertas seja capaz de suprir a demanda total dos clientes. Este vetor dá prioridade às facilidades de grande porte, que tendem a suprir a demanda total do sistema rapidamente, com a abertura de poucas facilidades.

$\boldsymbol{y^3}$: Seja $\langle f_{k_1},..., f_{k_m} \rangle$ a ordem não-decrescente dos custos fixos de instalação das facilidades. Inicialmente, seja $\boldsymbol{y^3} = \mathbf{0}$. Faça, então, $y^3_{k_i} = 1$, $\forall$ $1 \le i \le p$, tal que $\sum_{i=1}^{p} a_{k_i} \ge \sum_{j=1}^{n} b_j$ e $\sum_{i=1}^{p-1} a_{k_i} < \sum_{j=1}^{n} b_j$, ou seja, instale as facilidades de menor custo fixo de instalação até que o conjunto das facilidades abertas seja capaz de suprir a demanda total do sistema. Só levaremos este vetor em consideração se ele não coincidir com um dos vetores utilizados anteriormente.

***y*⁴**: Seja a mesma ordem não-decrescente $\langle f_{k_1},..., f_{k_m} \rangle$ obtida durante a construção do vetor y^3 anterior. Inicialmente, seja $y^4 = \mathbf{0}$. Faça, então, $y^4_{k_i} = 1$, $\forall\ p \le i \le m$, tal que $\sum_{i=p}^{m} a_{k_i} \ge \sum_{j=1}^{n} b_j$ e $\sum_{i=p-1}^{m} a_{k_i} < \sum_{j=1}^{n} b_j$, ou seja, instale as facilidades de maior custo de instalação até que o conjunto das facilidades abertas seja capaz de suprir a demanda total dos clientes. Este vetor dá prioridade a abertura de poucas facilidades de grande porte, que tendem a suprir a demanda total do sistema rapidamente com base no custo de instalação. Este vetor será levado em consideração se não coincidir com os vetores descritos anteriormente.

***y*⁵**: Seja $w_i = f_i / a_i$, $\forall\ i=1, ..., m$, o custo fixo por produto da facilidade *i*. Seja $\langle w_{k_1},..., w_{k_m} \rangle$ a ordem não-decrescente dos custos fixos por produto das facilidades. Inicialmente, seja $\boldsymbol{y^5} = \mathbf{0}$. Faça, então, $y^5_{k_i} = 1$, $\forall\ 1 \le i \le p$, tal que $\sum_{i=1}^{p} a_{k_i} \ge \sum_{j=1}^{n} b_j$ e $\sum_{i=1}^{p-1} a_{k_i} < \sum_{j=1}^{n} b_j$, ou seja, instale as facilidades de menor custo fixo de instalação por produto até que o conjunto das facilidades abertas seja capaz de suprir a demanda total do sistema.

***y*⁶**: Seja a mesma ordem não-decrescente $\langle w_{k_1},..., w_{k_m} \rangle$ obtida durante a construção do vetor anterior. Inicialmente, seja $y^6 = \mathbf{0}$. Faça, então, $y^6_{k_i} = 1$, $\forall$ p $\le i \le m$, tal que $\sum_{i=p}^{m} a_{k_i} \ge \sum_{j=1}^{n} b_j$ e $\sum_{i=p-1}^{m} a_{k_i} < \sum_{j=1}^{n} b_j$, ou seja, instale as facilidades de maior custo fixo por produto até que o conjunto das facilidades abertas seja capaz de suprir a demanda total do sistema. Este vetor, assim como $\boldsymbol{y^2}$, também dá prioridade às facilidades de grande porte, que tendem a suprir a demanda total do sistema rapidamente, com a abertura de poucas facilidades, entretanto aqui leva-se em consideração o custo fixo por produto.

***y*⁷**: É o vetor gerado aleatoriamente da seguinte forma: Inicialmente, seja $\boldsymbol{y^7} = \mathbf{0}$**;** Escolha aleatoriamente uma componente *k* em $\boldsymbol{y^7}$ tal que se $y^7_k = 0$, então faça $y^7_k = 1$, e, enquanto $\sum_{i=1}^{m} a_i y_i < \sum_{j=1}^{n} b_j$, repita esta mesma operação para outro valor de *k*, entre 1 e *m*. A construção deste vetor tem por finalidade produzir uma solução totalmente aleatória, tentando avaliar uma solução que não foi determinada nos passos anteriores, levando em consideração apenas o atendimento da demanda total do sistema.

***y*⁸**: É o vetor gerado aleatoriamente à direita de *t* dado pela seguinte forma: Inicialmente, selecionamos um valor *t*, entre 1 e *m*; seja $\boldsymbol{y^8 = 0}$**;** Para todo $k = t, t+1, .., m$ faça $y^8_k = 1$, enquanto $\sum_{i=1}^{m} a_i y_i < \sum_{j=1}^{n} b_j$. A construção deste vetor tem por finalidade produzir uma solução aleatória abrindo somente as facilidades *i*, com *i* entre *t* e *m*, levando em consideração apenas o atendimento da demanda total do sistema. Caso as *m-t* facilidades abertas não atendam a demanda total dos clientes então abriremos a facilidade 1, caso seja necessário a facilidade 2, e assim por diante, até que a demanda total seja suprimida pelas facilidades abertas.

y^9: É o vetor gerado aleatoriamente à esquerda de t dado pela seguinte forma: Sejam t, o número selecionado entre 1 e m dado na construção do vetor y^8, e **y^9=0;** Para todo k=t, t-1, .., 1, faça y_k^9=1 enquanto $\sum_{i=1}^{m} a_i y_i < \sum_{j=1}^{n} b_j$. A construção deste vetor tem por finalidade produzir uma solução aleatória abrindo somente as facilidades i, com i entre 1 e t, levando em consideração apenas o atendimento da demanda total do sistema. Caso as t facilidades abertas não atendam a demanda total dos clientes então abriremos a facilidade m, caso seja necessário a facilidade m-1, e assim por diante, até que a demanda total seja suprimida pelas facilidades abertas.

y^{10}: É o vetor gerado aleatoriamente à esquerda e à direita de t alternadamente, até que a demanda total dos clientes sejam atendida pelas facilidades abertas.

Construídos os vetores acima, devemos resolver o PT associado a cada um deles. Durante a pesquisa, tentamos resolver o PT através do algoritmo *Modi*, descrito em Hadley [13], entretanto este método não funcionou bem para problemas de médio e grande porte. Constatamos que em alguns problemas, a memória RAM existente e disponível, nas máquinas utilizadas, não foi suficiente, visto que este algoritmo é complexo e recursivo. Assim, desenvolvemos uma heurística, descrita a seguir, para realizar esta delicada tarefa. Nossa heurística foi desenvolvida com base no método desenvolvido por Silva e Soma [14].

3. Heurística Permutacional aplicada ao Problema de Transporte

Um problema de otimização combinatória permutacional pode ser definido por um terno (S, g, n), onde S é o conjunto de todas as soluções viáveis (soluções que satisfazem as restrições do problema, com $|S|=n!$), g é uma função ou um procedimento que avalia cada solução $s \in S$ e n é uma instância do problema. Podemos representar s como uma permutação de n elementos distintos, ou seja, s=< a_1 a_2 ... a_n >. N(s) é chamada *a vizinhança de s* e contém todas as soluções que podem ser alcançadas de s por um simples movimento. Aqui, o significado de um movimento é aquele de um operador que transforma uma solução para uma outra com pequenas modificações.

O procedimento da Heurística Permutacional aplicada ao Problema de Transporte, denominada de *HP*, consiste basicamente em dividir o conjunto de soluções viáveis S em n vizinhanças N(s_i) distintas entre si, e cada uma destas vizinhanças em quatro subvizinhança N(s_{ij})⊂N(s_i), $1 \le j \le 4$. A permutação s_i, que gera N(s_i), tem o elemento i na primeira posição da permutação, e para toda permutação s em N(s_i), s também inicia com o elemento i. Desta forma N(s_i)∩N(s_k)=∅, $1 \le i,k \le n$, com $i \ne k$. Exceto s_{i1}= s_i, as quatro permutações s_{i1}, s_{i2}, s_{i3} e s_{i4} que produzirão as quatro subvizinhanças N(s_{i1}), N(s_{i2}), N(s_{i3}) e N(s_{i4}), respectivamente, são obtidas da troca de posições dos elementos de s_i, da seguinte forma:

1°) A permutação s_{i2} mantém a primeira posição de s_i e inverte as (n-1) posições restantes de s_i;
2°) A permutação s_{i3} mantém a primeira posição de s_i e troca seqüencialmente 2 a 2 as demais posições adjacentes de s_i;
3°) A permutação s_{i4} mantém a primeira posição de s_{i3} e inverte as (n-1) posições restantes de s_{i3}.

A subvizinhança N(s_{ij}) é formada por todas as permutações que são obtidas de s_{ij} trocando de posição 2 a 2 todos os elementos de s_{ij}, a partir da segunda posição, e mantendo a ordem dos outros elementos intactos. Assim, o número de permutações em cada subvizinhança é [(n-1)×(n-2)/2 + 1] que implica em [4×[(n-1)×(n-2)/2 +1]] permutações em cada vizinhança e um total de [2 × n × (n-1) × (n-2) + 4n] permutações geradas para o problema.

Dada uma permutação qualquer s_1, podemos obter os demais s_i, $2 \le i \le n$, trocando o elemento da primeira posição de s_1 pelo elemento da i-éssima posição, mantendo a ordem dos outros elementos intactos. Sem perdas de generalidades podemos supor s_1=<1 2 3 ... n>, então s_2=<2 1 3 4 5 ... n>, s_3=<3 2 1 4 5 ... n >, ... , s_n=< n 2 3 4 5 ... (n-1) 1 >. A Tabela 1, dada a seguir, mostra as

subvizinhança da vizinhança $N(s_{1j})$, com n=8. A Tabela 2, abaixo, por sua vez, apresenta o Número de Permutações geradas no Problema, na coluna HP, para valores de n entre 10 e 500.

j	s_{1j}	$N(s_{1j})$
1	12345678	13245678, 14325678, 15342678, 16345278, 17345628, 18345672, 12435678, 12543678, 12645378, 12745638, 12845673, 12354678, 12365478, 12375648, 12385674, 12346578, 12347658, 12348675, 12345768, 12345876, 12345687
2	18765432	17865432, 16785432, 15768432, 14765832, 13765482, 12765438, 18675432, 18567432, 18465732, 18365472, 18265437, 18756432, 18745632, 18735462, 18725436, 18764532, 18763452, 18762435, 18765342, 18765234, 18765423
3	13254768	12354768, 15234768, 14253768, 17254368, 16254738, 18254763, 13524768, 13452768, 13754268, 13654728, 13854762, 13245768, 13274568, 13264758, 13284765, 13257468, 13256748, 13258764, 13254678, 13254867, 13254786
4	18674523	16874523, 17684523, 14678523, 15674823, 12674583, 13674528, 18764523, 18476523, 18574623, 18274563, 18374526, 18647523, 18654723, 18624573, 18634527, 18675423, 18672543, 18673524, 18674253, 18674325, 18674532

Tabela 1 – Subvizinhanças da permutação s_1= <12345678>.

n	**HP**	***n***	**HP**
10	1.480	70	657.160
15	5.520	75	810.600
20	13.760	80	986.240
30	48.840	85	1.185.580
40	118.720	90	1.410.120
50	235.400	100	1.940.800
60	410.880	200	15.761.600
65	524.420	500	248.504.000

Tabela 2 – Número de permutações geradas por HP.

A Figura 1, dada a seguir, apresenta detalhes da implementação do nosso procedimento, escrito em linguagem Ansi C. Executamos este programa recursivamente, sem analisar o desempenho da função objetivo, e constatamos que não houve repetição de permutação para valores de $n \geq 8$, em nenhuma vizinhança. A heurística proposta utiliza uma quantidade de recursos computacionais claramente polinomial, mais ainda, no pior caso tem-se $O(n^3)$ de tempo de execução e $O(n)$ de espaço (memória). Limitantes esses, que para os valores encontrados em aplicações práticas são bastante adequados. O procedimento $g(n, s)$ que avalia o desempenho da solução atual s será descrito a seguir.

Podemos modelar o PT como sendo um POCP, onde S é o conjunto de todas as permutações possíveis dos n consumidores. O procedimento g usado para avaliar uma solução s de S leva em consideração a ordem em que os consumidores se encontram na permutação s. Tendo o primeiro consumidor em s prioridade para ser atendida toda a sua demanda, depois o segundo e assim sucessivamente. A oferta do produto nas facilidades será atualizada sempre que for atendida a demanda total de um consumidor. A distribuição do produto para atender a demanda total de um consumidor é realizada seguindo o menor custo de transporte das facilidades com oferta disponível para este consumidor.

Para deixar a execução do procedimento g mais rápida, foram alocados em ordem crescente os dados relativos aos custos de transportes para cada consumidor, no início do programa. A seguir descrevemos a avaliação do desempenho do nosso método, utilizando os problemas testes encontrados na literatura.

```
void HP( n )
{ int i, j, k, t, p1, s;
  int s1[MAXITEMS], s2[MAXITEMS];

  for(i=1; i<=n; i++) s1[i]=i;              /* s1 está associada a solução s_i                */
  for(i=1; i<=n; i++) {                     /* gera as n vizinhanças a partir de s_i          */
     if (i>1) { s1[1]=i; s1[i]=1; }         /* identifica a primeira posição de s_i           */
     for(j=1; j<=4; j++) {                  /* Procedimento para gerar as 4 permutações s_ij  */
       switch (j) {
          case 1: for(k=1; k<=n; k++) s2[k]=s1[k]; break;
          case 2: t=(n+1)/2; for(k=2; k<=t; k++) { s=s2[k]; s2[k]=s2[n+2-k]; s2[n+2-k]=s; } break;
          case 3: for(k=1; k<=n; k++) s2[k]=s1[k]; k=2;
                  while (k<=(n-1)) { s=s2[k]; s2[k]=s2[k+1]; s2[k+1]=s; k=k+2; } break;
          case 4: t=(n+1)/2; for(k=2; k<=t; k++) { s=s2[k]; s2[k]=s2[n+2-k]; s2[n+2-k]=s; } break;
       }              /*  Fim do Switch  */
       g( n, s2);     /* Avalia o desempenho da primeira solução da subvizinhança */

       for(k=2; k<=(n-1); k++) {  /* Procedimento para gerar a subvizinhança a partir de s2=s_ij */
         for(t=k+1; t<=n; t++) {
           s=s2[k]; s2[k]=s2[t]; s2[t]=s;
           g( n, s2);          /* Avalia o desempenho das demais soluções da subvizinhança */
           s2[t]=s2[k]; s2[k]=s;
         }  /*  Fim do for t de k+1 a n    */
       }    /*  Fim do for k de 2 a n-1   */
    }       /*  Fim do for j de 1 a 4      */
    if (i>1) { s1[1]=1; s1[i]=i; }
  }     /* Fim do for i de 1 a n  */
  return;
}
```

Figura 1 – Detalhes da implementação da HP.

4. Experimentos Computacionais

A implementação do método proposto foi realizada em linguagem C, padrão ANSI, e executada em um Pentium IV, com 1.8 GHz e 256 MB de memória RAM.

Na Tabela 3, descrita a seguir, apresentamos um quadro resumo, contendo o nome do arquivo teste utilizado com o valor de *m* e *n*, a solução (z) obtida pelo método, a solução ótima (z^*) do problema, o desvio ($(z-z^*)/z^*$) e o tempo gasto (em segundos) para executar o método. Os arquivos de instâncias são os conjuntos de problemas-teste IV – XIII, descritos em Beasley [6]. Tais problemas incluem essencialmente problemas-teste de diversos padrões para o PLC. Entretanto muitos problemas possuem valores constantes e iguais para os custos de instalações f_i, que compromete a diversificação dos dez vetores *y*.

Conforme os dados apresentados na Tabela 3, a seguir, podemos ver que o método foi bastante rápido, tendo aproximadamente 2 segundos de tempo médio para apresentação da solução. Apenas 3 problemas levaram aproximadamente 6 segundos, o maior tempo de execução para determinar a solução. A média do desvio para os 37 problemas executados foi de 0,368, entretanto tivemos 4 casos em que o desvio ficou abaixo de 0,15 e 8 casos em que o desvio ficou entre 0,15 e 0,20. Aplicamos a técnica a três problemas (*n*=50 e *m*=10, 20, 25) simulado na plataforma do GASLOG, usando a internet, cujo o modelo matemático (1 a 5) foi utilizado para determinar a solução ótima. Nesta aplicação o resultado foi satisfatório, com o desvio médio sendo igual a 0,12 e o tempo variou de 7 a 10 segundos. Esta aplicação foi simulada na internet para analisarmos principalmente o fator tempo.

Arquivo (m,n)	Sol. Obtida	Sol. Ótima	Desvio	Tempo
cap41.txt (16,50)	1195363,875	1040444,375	0,149	3
cap42.txt (16,50)	1250363,875	1098000,450	0,139	3
cap43.txt (16,50)	1305363,875	1153000,450	0,132	3
cap44.txt (16,50)	1387863,875	1235500,450	0,123	5
cap51.txt (16,50)	1229040,625	1025208,225	0,199	4
cap61.txt (16,50)	1238485,750	932615,750	0,328	2
cap62.txt (16,50)	1253485,750	977799,400	0,282	2
cap63.txt (16,50)	1222489,250	1014062,050	0,206	2
cap64.txt (16,50)	1252489,250	1045650,250	0,198	2
cap71.txt (16,50)	1248142,375	932615,750	0,338	1
cap72.txt (16,50)	1248142,375	977799,400	0,276	0
cap73.txt (16,50)	1248142,375	1010641,450	0,235	1
cap74.txt (16,50)	1248142,375	1034976,975	0,206	1
cap81.txt (25,50)	1195363,875	838499,288	0,426	3
cap82.txt (25,50)	1250363,875	910889,563	0,373	3
cap83.txt (25,50)	1305363,875	975889,563	0,338	3
cap84.txt (25,50)	1387863,875	1069369,525	0,298	5
cap91.txt (25,50)	1238485,750	796648,438	0,555	3
cap92.txt (25,50)	1253485,750	855733,500	0,465	2
cap93.txt (25,50)	1268485,750	896617,538	0,415	2
cap94.txt (25,50)	1290985,750	946051,325	0,365	1
cap101.txt (25,50)	1248142,375	796648,437	0,567	1
cap102.txt (25,50)	1248142,375	854704,200	0,460	2
cap103.txt (25,50)	1248142,375	893782,112	0,396	1
cap104.txt (25,50)	1248142,375	928941,750	0,344	1
cap111.txt (50,50)	1398972,750	826124,713	0,693	6
cap112.txt (50,50)	1453972,750	901377,213	0,613	5
cap113.txt (50,50)	1479533,750	970567,750	0,524	6
cap114.txt (50,50)	1591472,750	1063356,488	0,497	6
cap121.txt (50,50)	1220415,500	793439,563	0,538	2
cap122.txt (50,50)	1235415,500	852524,625	0,449	3
cap123.txt (50,50)	1250415,500	895302,325	0,397	2
cap124.txt (50,50)	1258157,250	946051,325	0,330	2
cap131.txt (50,50)	1248142,375	793439,562	0,573	3
cap132.txt (50,50)	1248142,375	851495,325	0,466	3
cap133.txt (50,50)	1248142,375	893076,712	0,398	1
cap134.txt (50,50)	1248142,375	928941,750	0,344	4
Média			0,368	2,89

Tabela 3 – Resultados computacionais.

A Figura 2, dada na página a seguir, ilustra uma simulação do problema com 50 clientes e 20 possíveis centros de distribuição de gás. A ilustração foi gerada dentro da plataforma desenvolvida para a internet do GASLOG. Foram plotados sobre o mapa digital da Cidade de Fortaleza, pontos amarelos e verdes para facilidades e clientes, respectivamente. A Figura 3, mostra a distribuição do gás natural para os 50 clientes, sendo que para esta solução apenas 8 das 20 facilidades foram instaladas.

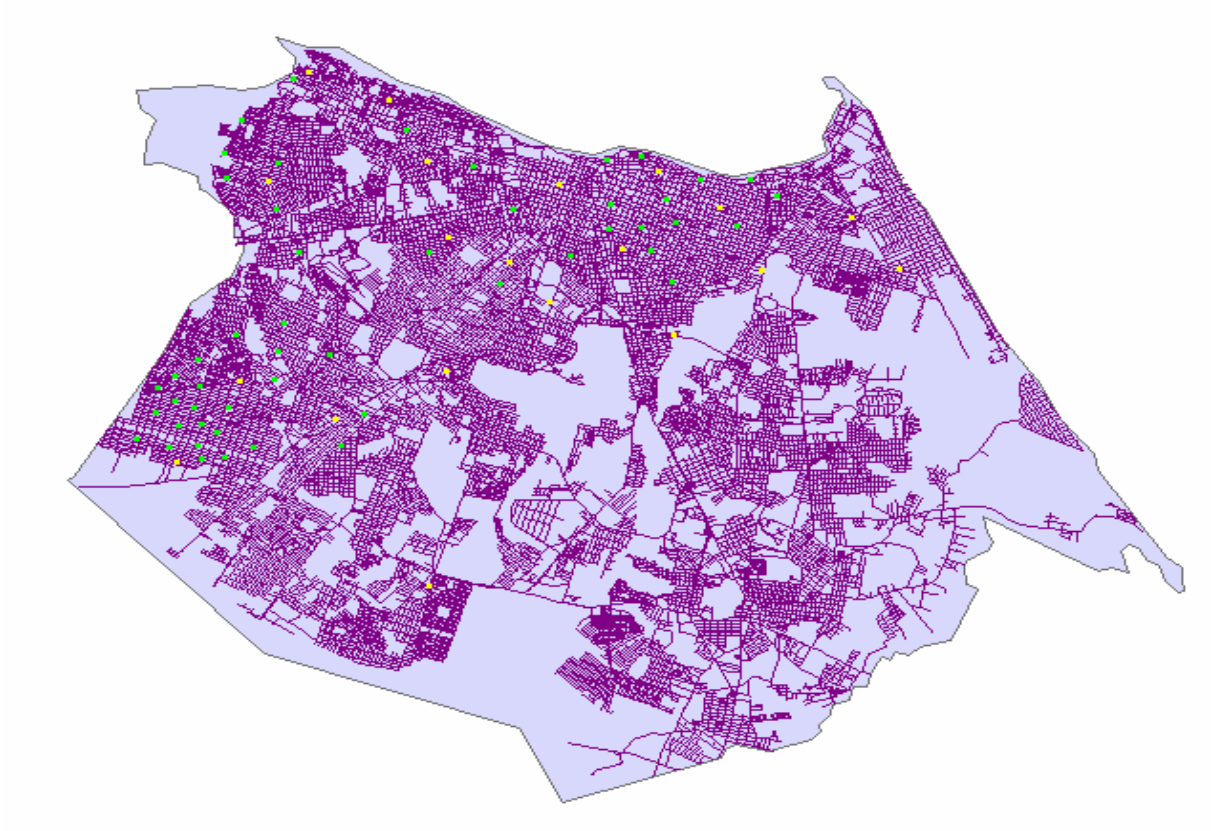

Figura 2 – Apresentação gráfica de um problema simulado no GASLOG com m=20 e n=50.

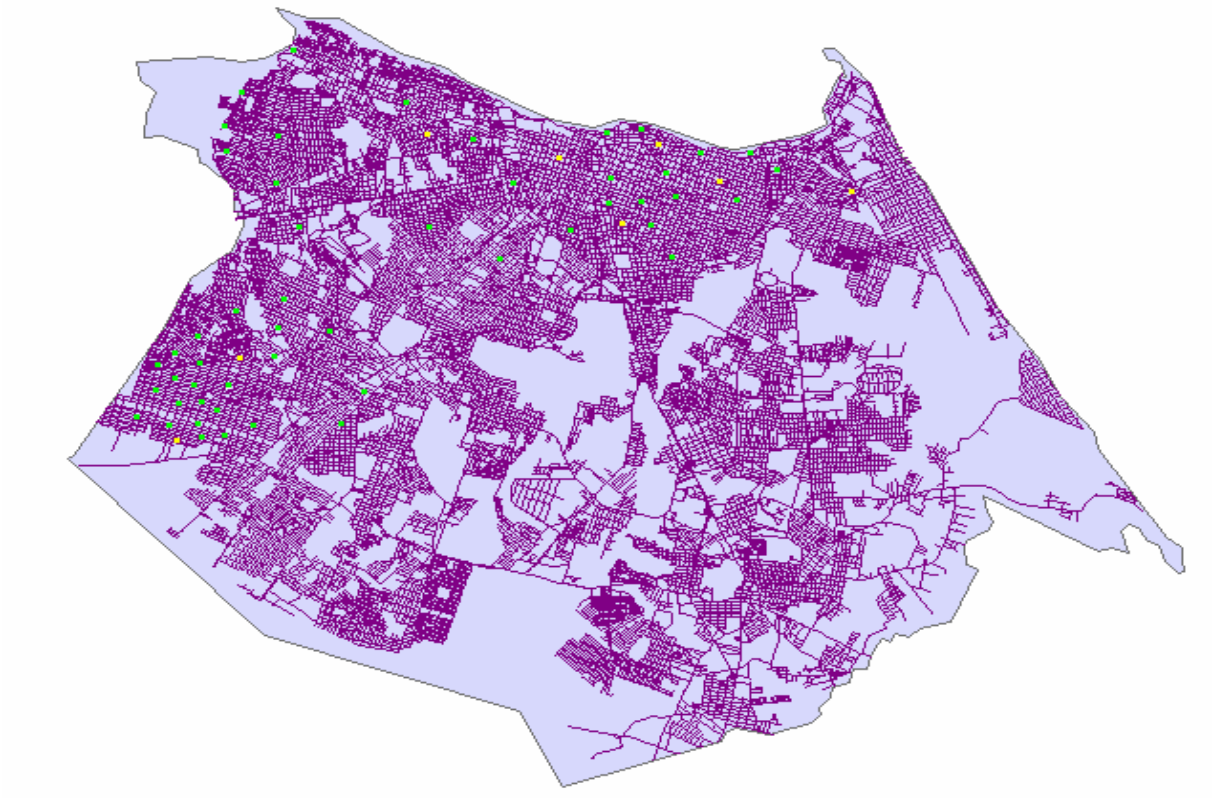

Figura 2 – Solução apresentada para o problema simulado no GASLOG com m=20 e n=50.

5. Conclusões

Constatamos que os resultados obtidos nos nossos experimentos computacionais com problemas encontrados na literatura não foram considerados satisfatórios, pois o desvio médio ficou acima das nossas expectativas, entretanto conseguimos resultados satisfatórios em relação ao tempo de execução. Quando a técnica foi aplicada nos problemas simulados para o projeto GASLOG foram alcançados bons resultados tanto para a solução obtida quanto para o tempo de execução da simulação. Visto que, nos problemas aplicados (plataforma GASLOG) tratamos com um volume de dados considerado via internet.

Verificamos que, na literatura, não encontramos um método que descrevesse o Problema de Transporte como um Problema de Otimização Combinatória Permutacional, outro fato importante constatado na nossa pesquisa.

Como futuros trabalhos sugerimos o processamento distribuído da heurística permutacional e o aumento do número de vetores *y* a serem gerados, levando em considerações outros fatores, sem comprometer o tempo de execução.

Agradecimentos
Os autores agradecem apoio financeiro do CNPq (processo 302176/03-9) e da Universidade Federal do Ceará.

6. Referências Bibliográficas

[1] L. Francis, L. F. McGinnis e J. A. White, *Location Analysis*, European Journal of Operational Research, v. 12 pg. 220-252, 1983.
[2] C.H. Aikens, *Facility Location Models for Distribution Planning*, European Journal of Operational Research, v. 22 pg. 263-279, 1985.
[3] R. Sridharan, *A Survey of the Capacitated Plant Location Problem*, Technical Report, Graduate School of Industrial Administration, Carnegier-Mello University, 1984.
[4] E. Bartezzaghi, A. Colorni e P. C. Palermo, *A Tree Search Algorithm for Plant Location Problems*, European Journal of Operational Research, v. 7 pg. 371-379, 1981.
[5] B. M. Baker, *A Partial Dual Algorithm for the Capacitated Warehouse Location Problem*, European Journal of Operational Research, v. 23 pg. 48-56, 1986.
[6] J. E. Beasley, *An Algorithm for Solving Large Capacitated Warehouse Location Problems*, European Journal of Operational Research, v. 33 pg. 314-325, 1988.
[7] T. J. Van Roy, *A Cross Decomposition Algorithm for Capacitated Facility Location*, Operations Research, v. 34 pg. 145-163, 1986.
[8] D. Valiati e C. T. Bornstein, *Método Capacitado para Resolver o Problema de Localização Capacitado Utilizando Testes de Redução e Relaxação Lagrangeana*, anais do XXXIII Simpósio Brasileiro de Pesquisa Operacional, pg. 1179-1190, 2001.
[9] C. T. Bornstein e H. B. Azlán, *The use of Reduction Tests and Simulated Annealing for the Capacitated Plant Location Problem*, Location Science, v. 6 pg. 145-149, 1991.
[10] C. T. Bornstein e G. R. Mateus, *Dominance Criteria for the Capacitated Warehouse Location Problem*, Operational Research Society, v. 42 pg. 145-149, 1991.
[11] S. K. Jacobsen, *Heuristics for the Capacitated Plant Location Model*, European Journal of Operational Research, v. 12 pg. 253-261, 1983.
[12] J. E. Beasley, *Langrangean Heuristics for Location Problems*, European Journal of Operational Research, v. 65 pg. 383-399, 1993.
[13] Hadley, G. Linear Programming. Addison-Wesley Publishing Company. USA 1967.
[14] J. L. C. Silva e N. Y. Soma. Uma Heurística para Problemas de Otimização Combinatória Permutacional. *Proceedings of the XXXIII SBPO*, Campos do Jordão-SP, Brazil, 2001.