Aplicação da metaheurística LNS ao problema probabilístico de localização-alocação de máxima cobertura

Ligia Corrêa de Souza[1], Luiz Antonio Nogueira Lorena[2], Marcos Antonio Pereira[3]
[1]Programa de Mestrado em Computação Aplicada – CAP
Instituto Nacional de Pesquisas Espaciais – INPE
[2]Laboratório Associado de Computação e Matemática Aplicada – LAC
Instituto Nacional de Pesquisas Espaciais – INPE
[3]Departamento de Matemática
Universidade Estadual Paulista – UNESP

{li.correasouza@gmail.com**,** lorena@lac.inpe.br**,** mapereira@feg.unesp.br**}**

Abstract. The facility location problems deal with decisions involving the meeting demand for an individual or a population center from suppliers of products or services, considering logistics. The Queuing Maximal Covering Location-Allocation Model is a facility location problem which seeks to locate facilities to maximize the attendance of demand for given a maximum distance coverage, considering a minimum criterion of good quality. In the probabilistic problem here studied, the facilities must be located so that customers arrive within a reasonable time and also that, once in the queue, the waiting time is not greater than a maximum limit and/or the length of it is not greater than a maximum value to ensure the quality of service. This article discusses heuristic techniques to solve this problem using an algorithm called Large Neighborhood Search. The tests were performed for instances of 30 and 324 points and some results are presented and compared with results found in literature.

Keywords: Location-allocation problems, facility location, Large Neighborhood Search.

Resumo. Os problemas de localização de facilidades tratam de decisões envolvendo o atendimento da demanda de um indivíduo ou de uma população a partir de centros fornecedores de produtos ou serviços, considerando aspectos logísticos. O Problema Probabilístico de Localização-Alocação de Máxima Cobertura é um problema de localização de facilidades em que se busca localizar facilidades de modo a maximizar o atendimento da demanda, para uma dada distância máxima de cobertura, considerando um critério mínimo de qualidade desse atendimento. No problema probabilístico aqui estudado, as facilidades devem ser localizadas de tal maneira que os clientes cheguem dentro de um tempo aceitável e também que, uma vez na fila, o tempo de espera não seja maior que um limite máximo e/ou que o comprimento da mesma não seja maior que um valor máximo para garantir a qualidade no serviço. O presente artigo aborda técnicas heurísticas já existentes para resolver este problema utilizando um algoritmo chamado *Large Neighborhood Search*. Os testes computacionais foram realizados para instâncias de 30 e 324 pontos e alguns resultados são apresentados e comparados com os resultados encontrados na literatura e pelo software CPLEX.

Palavras-chave: Problemas de localização-alocação, localização de facilidades, *Large Neighborhood Search.*

1. Introdução

O Problema de Localização de Máxima Cobertura (MCLP) tem sido estudado vastamente desde sua formulação por Church e ReVelle (1974) e considera um conjunto de pontos de demanda e um conjunto de locais candidatos para a instalação de facilidades. A cada facilidade está associada uma distância de serviço S (também chamada de distância crítica ou de cobertura) de forma que apenas os clientes situados a uma distância menor que S de algum centro serão atendidos. Como o número de centros a serem instalados pode não ser suficiente para atender todos os clientes, o problema busca determinar os locais de instalação das facilidades que atendam a maior parte de demanda existente. No entanto, essa abordagem não considera alguns detalhes do atendimento (como a espera causada por filas) que, em alguns casos, podem incorrer em custos adicionais indiretos. Usualmente, associa-se a formação de filas a um excesso de demanda de um serviço em relação a capacidade de atendimento [Moreira 2007]. Outros problemas podem ser modelados como extensões dessa formulação, de modo a representar com mais fidelidade a realidade, pois para os sistemas que apresentam comportamento de chegada aleatória de clientes, a qualidade do serviço não está relacionada apenas com a cobertura dos clientes. Assim, as facilidades devem ser localizadas de tal maneira que os clientes cheguem dentro de um tempo aceitável e também que, uma vez na fila, o tempo de espera não seja maior que um limite máximo e/ou que o comprimento da mesma não seja maior que um valor máximo, garantindo assim uma qualidade mínima no atendimento [Marianov e Serra 1998].

O congestionamento no atendimento pode ocorrer não só porque a capacidade do centro seja insuficiente, mas também devido à variabilidade no intervalo de chegadas de clientes e no tempo de atendimento. Os modelos tradicionais que tratam desse problema adicionam uma restrição de capacidade que força a demanda por serviço, normalmente constante e menor do que a máxima capacidade do centro, considerando que o número de solicitações de serviço é constante no tempo. Tratando o problema de forma determinística, podem-se ter servidores ociosos ou sobrecarregados [Moreira 2007]. Considerando essa aleatoriedade nos processos de chegada e de atendimento, Marianov e Serra (1998) propuseram modelos que definem uma qualidade mínima no serviço tratando a estocasticidade do problema em restrições adicionais de capacidade. Esses autores definiram então o Problema Probabilístico de Localização-Alocação de Máxima Cobertura (QM-CLAM), fazendo as considerações de estocasticidade da demanda, que busca localizar uma dada quantidade de facilidades com um ou vários servidores, de modo que a população, a uma dada distância de um centro de atendimento, seja servida adequadamente, isto é, que o usuário não fique na fila por um período maior que um dado tempo limite ou que, ao chegar ao centro, não encontre um número de outros clientes acima do previsto, dada uma probabilidade mínima de que isso ocorra. No modelo aqui estudado, considera-se um servidor por centro, ou seja, só existe um posto de atendimento ao cliente em cada facilidade, responsável por seu atendimento integral.

Corrêa e Lorena (2006) resolveram o QM-CLAM usando o Algoritmo Genético Construtivo (AGC) e a Relaxação Lagrangeana. Corrêa *et. al.* (2007) resolveram o QM-CLAM usando uma heurística híbrida chamada *Clustering Search* (CS), que consiste na detecção de regiões promissoras do espaço de busca usando um algoritmo em que soluções semelhantes são agrupadas, após tal medida de similaridade ser definida. Essas regiões promissoras são exploradas utilizando métodos de busca local. Por fim, Corrêa *et. al.*(2009) solucionaram o QM-CLAM utilizando o método de Geração de Colunas (CG).

O propósito deste artigo é o de examinar o Problema Probabilístico de Localização-Alocação de Máxima Cobertura proposto por Marianov e Serra (1998), para um servidor por centro de serviços, e apresentar a solução usando uma heurística conhecida por *Large Neighborhood Search* (LNS) proposta por Shaw (1998). Os resultados são comparados com os obtidos com o CG – que obtém os melhores resultados encontrados na literatura – e com os obtidos pelo *software* CPLEX.

2. Modelo QM-CLAM

A modelagem de problemas de localização de facilidades geralmente utiliza uma representação

em grafos, estabelecendo um conjunto de nós representando os clientes e os locais candidatos para a instalação das facilidades, e um conjunto de arcos representando as possíveis ligações entre clientes e facilidades. No caso geral, considera-se que o conjunto de locais candidatos para a instalação das facilidades é distinto do conjunto de clientes. Neste caso, têm-se dois conjuntos de nós: um para representar os locais potenciais para a instalação das facilidades e outro para representar os clientes. Neste trabalho, será assumido que cada nó que representa um cliente é também um candidato para a instalação de uma facilidade.

O modelo MCLP tradicional de Church e ReVelle (1974) não pode ser usado para tratar as restrições de capacidade, pois não comporta as variáveis de localização e alocação. Sem elas não seria possível computar as solicitações de serviços que chegam a um centro, para, consequentemente, poder determinar a ocorrência de um congestionamento. Segundo Pontin *et al.* (2010), a formulação original de Marianov e Serra (1998) é a mais adequada para representar o problema aqui tratado. Nesta formulação, as alocações de clientes a centros são representadas pela matriz binária x_{ij}, onde I é o conjunto dos pontos de demanda e N_i é o conjunto dos pontos das facilidades candidatas que estão dentro da distância de atendimento do cliente i, com $x_{ij} = 1$, se o ponto de demanda i é alocado a facilidade j, e $x_{ij} = 0$, caso contrário. As localizações são representadas pelas variáveis binárias y_j, com $y_j = 1$ se o centro j é selecionado e $y_j = 0$, caso contrário. O parâmetro a_i define a demanda do ponto i. A constante φ indica a probabilidade mínima de serem encontradas no máximo b pessoas na fila ou a probabilidade mínima de que o tempo máximo de espera seja τ minutos. Assume-se também que o tempo de serviço tem uma distribuição exponencial, com taxa média ϕ e que f_i é o produto de uma constante pré-definida pela demanda do ponto i. O modelo de Marianov e Serra (1998) segue como:

$$v(QM - CLAM) = Max \left\{ \sum_{i \in I} \sum_{j \in N_i} a_i x_{ij} \right\} \tag{1}$$

$Sujeito\ a$:

$$\sum_{j \in N_i} x_{ij} \leq 1, \qquad \forall i \in I \tag{2}$$

$$\sum_{j} y_j = p \tag{3}$$

$$x_{ij} \leq y_j, \qquad \forall i \in I \ e \ j \in N_i \tag{4}$$

$$\sum_{i \in I} f_i x_{ij} \leq \phi \sqrt[b+2]{1 - \varphi}, \qquad \forall j \tag{5}$$

$$ou \quad \sum_{i \in I} f_i x_{ij} \leq \phi + \frac{1}{\tau} ln(1 - \varphi), \qquad \forall j \tag{6}$$

$$y_j \ e \ x_{ij} \in \{0,1\}, \qquad \forall i \in I \ e \ \forall j \tag{7}$$

A função objetivo (1) maximiza a demanda alocada aos centros. As restrições (2) impõem que cada ponto de demanda deve ser alocado a, no máximo, uma facilidade. A restrição de cardinalidade (3) define a quantidade de centros a serem abertos. As restrições (4) definem que somente é possível alocar um ponto de demanda i a um centro j se houver um centro instalado em j. As restrições (5) forçam para que haja no máximo b pessoas na fila com, no mínimo, a probabilidade φ. As restrições (6) determinam que o tempo gasto no centro j seja, no máximo τ minutos, com, no mínimo, a probabilidade φ. E, por fim, as restrições (7) definem a natureza binária das variáveis de decisão.

Ao descrever as restrições (5) e (6) foi considerado o modelo de fila M/M/1/∞/FIFO, o que significa que os intervalos entre chegadas estão exponencialmente distribuídos, o tempo de atendimento também está de acordo com uma distribuição exponencial, com apenas um servidor, com fonte de clientes infinita e a disciplina de fila é do tipo "o primeiro a chegar é o primeiro a ser atendido" (*first in - first out – FIFO*). Para as restrições (5) e (6), considera-se que as solicitações de serviços de cada nó de demanda i acontecem de acordo com um processo de

Poisson com taxa f_i. Muitos usuários podem vir de vários pontos de demanda ao mesmo tempo, por isso, a taxa atribuída a uma facilidade é definida como uma superposição de processos de Poisson:

$$\omega_j = \sum_{i \in I} f_i x_{ij} \tag{8}$$

Isso significa que se a variável x_{ij} for 1, o ponto i será alocado ao centro j, e a sua taxa correspondente será incluída no cálculo de ω_j.

Abordagens exatas podem resolver algumas instâncias desse problema, mas devido à natureza combinatória do QM-CLAM, os tempos de processamento necessários para calcular soluções viáveis para o problema podem ser elevados mesmo para instâncias de pequeno porte, o que justifica a pesquisa de métodos alternativos para resolver este problema.

3. LNS

A metaheurística *Large Neighborhood Search* (LNS) foi proposta por [Shaw 1998]. A ideia principal do LNS é que a larga vizinhança permite que a metaheurística navegue no espaço de solução facilmente, mesmo se a instância é restrita. Nessa metaheurística, a vizinhança é definida implicitamente por um método de destruição e um método de reparação. Tais métodos realizam uma alternância entre uma solução inviável e uma solução viável: a operação de destruir cria uma solução inviável que é trazida de volta em forma viável pela heurística de reparação. A vizinhança $N(x)$ de uma solução x é então definida como o conjunto de soluções que podem ser alcançadas pela primeira ao aplicar o método de destruir e, em seguida, o método de reparo. O método de destruir contém um elemento aleatório de tal modo que diferentes partes da solução são destruídas em cada chamada do método.

No algoritmo 1, mostra-se o pseudo-código do LNS. A variável x^b é a melhor solução observada durante a busca, x é a solução atual e x^t é uma solução temporária que pode ser descartada ou promovida para ser a solução atual. A função $d(.)$ é o método de destruição que retorna a solução x parcialmente destruída enquanto $r(.)$ é o método de reparo que devolve uma solução viável construída a partir da destruída.

Algoritmo 1: LNS

```
1:  Entrada: uma solução x viável
2:  x^b = x;
3:  Repita
4:      x^t = r(d(x))
5:      Se aceitar x^t então
6:          x = x^t
7:      Fim se
8:      Se v(x^t) < v(x^b) então
9:          x^b = x^t
10:     Fim se
11:     Até que o critério de parada seja atingido
12: Retorne x^b
```

Na linha 2 uma solução aleatória é gerada e armazenada na variável x^b. Na linha 4 os métodos de destruição e reparação são aplicados gerando a solução x^t. Na linha 5, a nova solução é avaliada, e uma heurística (função de aceitação) determina se esta solução deve ser a nova solução corrente (linha 6) ou se deve ser rejeitada. A linha 8 verifica se a nova solução é melhor do que a melhor solução conhecida e então atualiza-se a melhor solução na linha 9, se necessário, sendo que $v(x)$ denota o valor objetivo da solução x. O processo das linhas $(4-10)$ é repetido até que um determinado critério de parada seja alcançado. Na linha 12 a melhor solução encontrada é retornada.

4. LNS aplicado ao QM-CLAM

A solução de localização do problema é apresentada na forma de um vetor binário, sendo que o tamanho do vetor corresponde ao número de locais candidatos para a instalação das facilidades. O elemento do vetor é *1* quando a facilidade "posição mais um" é selecionada, e *0* caso contrário. Supondo que o número de facilidades candidatas seja *10* e que o número de facilidades a serem instaladas seja *3*, uma possível solução de localização é mostrada na Figura 1, sendo que as facilidades instaladas seriam *3*, 5 e *10*:

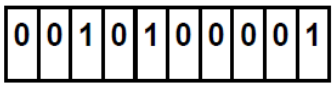

Figura 1. Representação de uma solução.

Uma solução aleatória é gerada para inicializar o algoritmo. Para esta solução, geramos aleatoriamente p posições do vetor solução, inicializado com o valor 0 em todas as posições, para serem iguais a 1. Desse modo, temos uma solução de localização inicial que respeita a restrição de cardinalidade e determina sua viabilidade. Após a geração da solução de localização, calculam-se as variáveis de alocação considerando-se as demais restrições do problema. Inicialmente, aloca-se cada cliente a todas as facilidades capazes de atendê-lo, de acordo com o raio de cobertura estabelecido. A seguir, eliminam-se as alocações múltiplas, se houverem, onde um cliente esteja alocado a mais de uma facilidade. Para tal eliminação, considera-se o cliente com maior número de alocações e o deixa alocado somente à facilidade menos congestionada, repetindo esse processo até que não haja alocações múltiplas. Se ainda assim as restrições de congestionamento (5) e (6) forem violadas para alguma facilidade, utiliza-se o CPLEX OPTIMIZER 12.1 (IBM, 2009) para encontrar as alocações de modo que as mesmas não excedam a capacidade. Desse modo, o CPLEX é acionado dentro do método de reparação, garantindo a viabilidade da solução reparada.

De acordo com Ropke e Pisinger (2006), a destruição da solução deve ser feita considerando o problema estudado. Duas implementações do LNS foram consideradas para este problema. Na primeira (LNS1), destruímos 50% das facilidades instaladas, pois a quantidade de facilidades utilizadas nos testes computacionais é pequena e, portanto, faz-se necessário destruir metade ou mais do vetor solução para que o mesmo mude em todos os casos testados. O método de destruição escolhido foi o método aleatório. Escolhemos aleatoriamente, entre as posições que têm valor igual a 1, uma posição do vetor solução, e o substituímos por 0, tornando assim a solução inviável, pois desrespeita a restrição de cardinalidade que define a quantidade de centros. Na segunda (LNS2), destruímos a solução da seguinte forma: deixamos 20% da solução com valores iguais a 0, p valores iguais a 1 e o restante com valores iguais a 2. Quando o valor é igual a 2, significa que aquele ponto não é demanda nem facilidade, ou seja, é um ponto que ainda deve ser definido, o que torna a solução incompleta e, portanto, inviável. Gera-se outra solução aleatória também incompleta para recombinar com a solução anteriormente destruída.

O método de reparo recupera a viabilidade da solução anteriormente destruída e, assim como para o método de destruição, tem-se a liberdade de escolhê-lo. O método de reparo da solução, implementado primeiro neste trabalho, também é aleatório, não garantindo que a solução reparada será melhor do que a solução anterior. A reparação em LNS2 segue as regras para a combinação das duas soluções, sendo que tais regras foram retiradas de Corrêa (2008) e baseiam-se no Algoritmo Genético Construtivo (Furtado, 1998) como segue na Tabela 1.

Tabela 1. Recombinação das soluções no método de reparação.

valor na posição da 1ª solução	valor na posição da 2ª solução	resultado
2	2	2
1	1	1
0 ou 2	0	0
1	2	1
0	2	0
0 ou 2	1	1 ou 0 (aleatoriamente)
1	0	1 ou 0 (aleatoriamente)

De acordo com Pisinger e Ropke (2009), a função de aceitação pode ser implementada de diferentes maneiras, como por exemplo utilizar a ideia do *Simulated Annealing* (SA). Esta idéia foi aplicada ao trabalho da seguinte forma: utilizou-se uma temperatura, inicialmente alta que decai com o tempo, a fim de aceitar soluções de piora com maior probabilidade (que utiliza a temperatura em seu cálculo) no início da busca e aceitar soluções de piora com menor probabilidade no fim da busca.

5. Resultados computacionais

O LNS foi testado na rede de *30* vértices fornecida em Marianov e Serra (1998) e na rede de *324* pontos, sendo que as distâncias são euclidianas, obtidas de uma base de dados geográficos da cidade de São José dos Campos-SP, acrescidas da população fictícia em cada ponto de demanda e que estão disponíveis em http://www.lac.inpe.br/~lorena/instancias.html. Para a rede de 30 pontos foram realizados testes apenas para as restrições de comprimento de fila, sendo: raio de cobertura (S) igual a 1,5 milhas, tempo médio de atendimento igual a 20 minutos, taxa de chamada diária igual a 0,015 vezes a demanda do ponto, todos definidos em Marianov e Serra (1998). Para a rede de 324 pontos considerou-se o teste com o LNS1 apenas, sendo: raio de cobertura igual a 250m, tempo de atendimento igual a 15 minutos; taxa de chamada igual a 0,01 vezes a demanda do ponto. O valor de φ é calculado da seguinte forma para a rede de 30 pontos: como a taxa de chegada é diária, temos que a taxa de atendimento também deve ser e, portanto, devemos saber quantos clientes podem ser atendidos por dia em cada facilidade, ou seja, temos $\phi = \frac{24*60}{20} = 72$.

Vários problemas foram montados, variando-se os parâmetros p, b *e* φ para cada rede. Os resultados dessa abordagem foram comparados aos obtidos pelo uso do *software* comercial CPLEX *12.1* (ILOG, 2009) e aos resultados de Corrêa *et. al.*(2009), que utilizaram Geração de Colunas (CG). Os nomes dos problemas foram codificados do seguinte modo: quantidade de facilidades para instalar, quantidade de pessoas na fila, tipo de restrição ('0' para restrições de quantidade de pessoas na fila ou '1' para restrições de tempo de atendimento) e probabilidade mínima de encontrar b pessoas na fila ou probabilidade mínima de esperar τ minutos para ser atendido.

Os problemas foram resolvidos em um computador Intel Core i5 2,4GHz com 4GB RAM. Os programas para o acionamento do CPLEX e para o LNS foram codificados em C, utilizando o ambiente Dev-C++ com compilador GCC do Projeto GNU. Os resultados são apresentados nas Tabelas 2 e 3. As tabelas são divididas em quatro colunas: o nome do problema; a solução do CPLEX; a solução do LNS1 e 2; e a melhor solução (CG) encontrada em Corrêa *et. al.* (2009). Na coluna referente ao CPLEX são fornecidos: a solução inteira viável e o *gap* obtidos do CPLEX, com o tempo de todos os problemas limitado em *3600* segundos (*1* hora). O valor de *Gap Cplex* igual a zero define que o valor ótimo foi obtido. Esse valor é calculado por: 100*(limite superior–limite inferior)/(limite inferior), e fornece, em porcentagem, o *gap* entre esses limitantes. Na coluna do LNS1, são apresentados a melhor solução, a média das soluções em 10 execuções com o tempo computacional limitado em 3600 segundos ou 1000 avaliações da função objetivo que não piore a mesma, sendo que o tempo médio das 10 execuções é apresentado, e o *Gap* LNS. O *Gap* LNS é calculado por: 100*[(melhor solução encontrada pelo CPLEX ou de Corrêa *et. al.*(2009)) – (Melhor solução do LNS)]/(Melhor solução do LNS), e fornece, em porcentagem, o *gap* entre esses limitantes. Na coluna LNS2 são apresentados a melhor solução encontrada e seu tempo de processamento considerando apenas uma execução do algoritmo, limitado em 3600 segundos.

Para a rede de 30 pontos, o LNS1 encontrou o valor ótimo para 60% das instâncias (valores em destaque na tabela) e, do restante, conseguiu valores próximos aos melhores valores encontrados pelo CPLEX e na literatura. Para as instâncias em que o LNS1 não encontrou o ótimo, o *Gap* LNS1 ficou entre 0% e 4%, com tempos computacionais abaixo de 9 segundos. Em 39% desses casos, o LNS1 obteve um tempo computacional pelo menos 284 vezes menor. Em 80% dos casos, o CPLEX encontrou a solução mais rápido que o LNS1. As instâncias que não apresentam valores na coluna CG não foram resolvidas pelo artigo comparado. Podemos verificar

também que, para os casos em que o CG foi testado, o LNS1 foi mais rápido do que o CG em 65% dos casos, sendo que encontrou o mesmo valor em 40% desses casos. Para o 60% restante, o LNS1 encontrou um valor com *GAP* inferior a 4%. Ainda considerando esta rede, o LNS2 encontrou soluções em menos tempo que o LNS1 em 84% dos casos, mas encontrou o valor ótimo em apenas 21% dos casos.

Tabela 2. Resultados para instâncias de 30 pontos.

Problema	CPLEX			LNS1 (destruição/reparação aleatória)				LNS2 (baseado AGC)		CG	
	Sol	Gap (%)	tempo (s)	melhor solução	Média	Gap (%)	tempo (s)	melhor sol	tempo (s)	melhor sol	tempo (s)
2_0_0_85	3700	0,24	0,19	**3700**	3644	0	4,558	**3700**	3,343	3700	0,53
3_0_0_85	5390	0,13	0,11	5330	5324	1,1257	5,222	5320	1,094	5390	28,03
4_0_0_85	5470	0	0,03	**5470**	5386	0	2,226	5390	1,175	-	-
2_0_1_85	5100	0	0,39	5090	4965	0,1965	2,302	4950	1,1413	5100	5,55
3_0_1_85	5390	0,19	0,25	**5390**	5353	0	1,265	5260	1,109	5390	3,77
4_0_1_85	5470	0	0,16	**5470**	5391	0	1,446	5390	1,209	-	-
2_0_2_85	5210	1,36	0,47	**5210**	4966	0	1,749	4930	1,957	5210	4,77
3_0_2_85	5390	0,16	0,06	**5390**	5363	0	1,34	5330	1,321	5390	1,88
4_0_2_85	5470	0	0,02	**5470**	5389	0	1,262	5390	1,254	-	-
2_0_0_95	2140	0	0,14	**2140**	2105	0	4,955	**2140**	4,876	-	-
5_0_0_95	5330	0,38	3600	5130	4964	3,8986	5,926	5070	3,569	5330	6,91
6_0_0_95	5410	0,37	3600	5390	5348	0,3711	4,006	5320	2,405	5410	42,41
2_0_1_95	3520	0,47	0,36	**3520**	3519	0	2,842	**3520**	3,424	-	-
3_0_1_95	5270	0,19	1286,7	5240	5149	0,5725	4,53	4950	4,294	5270	14,75
4_0_1_95	5390	0,19	1,23	**5390**	5375	0	1,83	5150	4,745	5390	30,88
5_0_1_95	5470	0	0,01	**5470**	5402	0	1,702	5390	1.264	-	-
2_0_2_95	4520	13,24	0,36	**4520**	4468	0	3,6	4520	2,35	4520	0,70
3_0_2_95	5390	0,12	0,06	**5390**	5053	0	2,059	**5390**	4,975	5390	11,45
4_0_2_95	5470	0	0,02	**5470**	5386	0	1,83	5330	2.854	-	-

Tabela 3. Resultados para instâncias de 324 pontos.

Problema	CPLEX			LNS1(destruição/reparação aleatória)				CG	
	solução	Gap (%)	tempo (s)	melhor solução	média	Gap (%)	tempo (s)	melhor solução	tempo (s)
10_0_0_85	37177	0,01	85,97	37060	36904	0,3228	193,347	37180	275,11
10_0_1_85	51000	0,02	3600	50003	49961	1,9549	187,12	51000	507,81
10_0_2_85	59739	0,01	96,7	59178	58609	0,9407	235,5	59740	604,48
10_0_0_95	21460	0,01	223,17	21297	21145	0,7596	324,423	21460	65,20
10_0_1_95	35360	0,02	3600	35280	34636	0,2262	436,912	35360	47,33
10_0_2_95	45390	0,01	3600	44708	43768	1,5025	448,907	45390	327,84
20_0_0_85	74352	0,01	3600	72624	72345	2,332	235,67	74358	2630,44
20_0_0_95	42918	0,01	311,17	42300	42246	1,4445	40,149	42920	2425,52
20_0_1_95	70719	0,02	3600	68859	67996	2,6315	383,33	70720	1067,41
20_0_2_95	90768	0,03	3600	87568	86435	3,5361	327,653	90778	2369,98
10_1_40_85	27699	0,01	397,66	27453	27076	0,8917	389,9	27700	238,39
10_1_41_85	29360	0,03	3600	28872	28120	1,6621	435,87	29360	145,88
10_1_42_85	30950	0,02	3600	30519	30489	1,3926	478,128	30950	42,91
10_1_48_90	26920	0,01	115,16	26905	26870	0,0557	43,579	26920	144,72
10_1_49_90	28329	0,01	398,07	27830	27596	1,7649	278,43	28330	315,70
10_1_50_90	29680	0,02	3600	29549	29459	0,4414	324,8	29680	81,81
20_1_40_85	55394	0,02	3600	53852	52829	2,789	297,46	55397	8457,66

Para a rede de 324 pontos, o LNS1 não encontrou o valor ótimo para nenhuma instância com o critério de parada utilizado, mas seu *GAP* ficou abaixo de 3,6%. O CPLEX não conseguiu comprovar o ótimo para nenhuma instância e em 59% dos casos atingiu o tempo máximo

estipulado. Para estes últimos casos, o LNS1 teve um tempo computacional pelo menos 7 vezes menor. Em apenas menos de 24% dos casos o CPLEX teve tempo computacional menor do que o atingido pelo LNS1. Em 59% dos casos o LNS1 encontrou uma solução, com *GAPs* inferiores a 4%, mais rapidamente que o método CG. Para os demais casos em que o LNS1 foi mais lento que o CG os *GAPs* foram inferiores a 2%.

6. Conclusão

Este artigo apresentou uma abordagem para a solução do Problema Probabilístico de Localização-Alocação de Máxima Cobertura utilizando a metaheurística *Large Neighborhood Search*, sendo que a aplicação desta metaheurística é inédita para problemas de localização de facilidades. Para grande parte das instâncias, o LNS foi capaz de encontrar soluções ótimas aproximadas em tempo reduzido, o que valida os resultados do LNS para o QM-CLAM. Os resultados mostram que a abordagem é competitiva para a resolução deste problema, em tempos computacionais razoáveis com convergência rápida, e pode ser aplicado a outros problemas de localização. A utilização de outras metaheurísticas bem como de outros métodos de reparação e destruição podem ser considerados em trabalhos futuros para a solução do QM-CLAM, além do aprimoramento dos métodos de reparação aqui utilizados, considerando instâncias maiores deste modelo. Além disto, esta abordagem pode ser aplicada ao problema com *m* centros de serviços.

Referências

Church, R., e C. ReVelle, (1974), The Maximal Covering Location Problem. Papers of the Regional science Association: 32, 101 – 118.

Corrêa, F. A. (2008), Relaxações e método de decomposição para alguns problemas de localização de facilidades modelados em grafos. São José dos Campos: Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, INPE, 134f.

Corrêa, F. A. *et. al*. (2007), Heurística híbrida com detecções de regiões promissoras aplicada ao problema probabilístico de localização-alocação de máxima cobertura. SBPO.

Corrêa, F. A. e Lorena, L. A. N. (2006), Aplicação da relaxação Lagrangeana e do algoritmo genético construtivo na solução do problema probabilístico de localização-alocação de máxima cobertura. Revista Gestão & Produção: v.13, n.2, 233-244.

Corrêa, F. A. *et. al*. (2009), A decomposition approach for the probabilistic maximal covering location-alocation problem. Computers & Operations Research: 36, 2729-2739.

Furtado, J. C. (1998), Algoritmo genético construtivo na otimização de problemas combinatorais de agrupamentos. 112 p. Tese (Doutorado em Computação Aplicada) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos-SP, Brasil.

IBM ILOG CPLEX 12.1 Reference Manual, 884 p. Copyright by IBM, 2009.

Marianov, V. e Serra, D. (1998), Probabilistic maximal covering location-allocation models for congested systems. Journal of Regional Science: v. 38, n. 3, p. 401-424.

Moreira, D. A. (2007), Pesquisa Operacional-Curso Introdutório. São Paulo: Thomson Learning.

Pisinger, D. e Ropke, S. (2009), Large Neighborhood Search. In: Handbook of Metaheuristics, por M. Gendreau e J. Y. Potvin, 2ª ed. Forthcoming.

Pontin, V. M. et al. (2010), Análise de modelos matemáticos para o Problema Probabilístico de Localização-Alocação de Máxima Cobertura. Cadernos do IME – Série Estatística: v.28, 1-14.

Ropke, S. e Pisinger, D. (2006), An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science: 40(4): 455-472.

Shaw, P. (1998), Using constraint programming and local search methods to solve vehicle routing problems. In: CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming). Lecture Notes in Compputer Science, v.1520, p.417-431.