1 CAPÍTULO

O MÉTODO SIMULATED ANNEALING APLICADO EM LOCALIZAÇÃO E ROTEAMENTO

Ferreira, Kamyla Maria [1 *];
Queiroz, Thiago Alves de [2]

[1] Unidade de Matemática e Tecnologia, Universidade Federal de Goiás – Regional Catalão
[2] Unidade de Matemática e Tecnologia, Universidade Federal de Goiás – Regional Catalão

*** email: kamylamaaria@gmail.com**

Resumo: Neste trabalho é investigado o problema de roteamento de veículos integrado com a localização de facilidades, os quais são considerados simultaneamente na tomada de decisão. O problema apresenta capacidade limitada para o depósito e veículos, com o objetivo de minimizar o custo para determinar a localização dos depósitos e o atendimento dos clientes por meio de rotas a serem realizadas. Para tanto, é apresentada uma heurística baseada no método de recozimento simulado, que inclui alguns operadores de vizinhança e de busca local. Experimentos computacionais mostraram que a heurística é competitiva e permite possibilidades de melhorias para lidar com instâncias maiores.

Palavras-chave: Problema de Localização e Roteamento; Recozimento Simulado; Heurística

Ferreira, Kamyla Maria; Queiroz, Thiago Alves de; "O MÉTODO SIMULATED ANNEALING APLICADO EM LOCALIZAÇÃO E ROTEAMENTO", p. 8-24 . In: Seminário de Pesquisa, Pós-Graduação e Inovação da Regional Catalão (2. : 2014 : Goiás) **Coletânea Interdisciplinar em Pesquisa, Pós-Graduação e Inovação - Volume 4 : Ciências Exatas e Tecnológicas**. Anais [livro eletrônico] / organizado por Adriana Freitas Neves, Idelvone Mendes Ferreira, Maria Helena de Paula, Petrus Henrique Ribeiro dos Anjos. São Paulo: Blucher, 2015. ISBN: 978-85-8039-115-2, DOI 10.5151/978859788580391152-V4_Cap1

1. Introdução

Várias empresas buscam minimizar o custo total de distribuição de mercadorias, realizando um planejamento logístico que melhor estabeleça a localização de depósitos e depois rotas para distribuir mercadorias em relação aos seus clientes. Aplicações práticas em problemas de logística podem ser encontradas em Queiroz e Miyazawa (2013, 2014) e Gonçalves e Queiroz (2014).

De acordo com Tuzun e Burke (1999) existem duas maneiras para realizar a distribuição de produtos aos clientes, sendo que na primeira cada veículo atende somente uma demanda, ou seja, a necessidade de um único cliente é suficiente para encher um caminhão. Já na segunda, há mais de um cliente, uma vez que é necessário

diversas demandas para completar a capacidade do veículo. O problema descrito inicialmente é referenciado no contexto do Problema de Localização e Roteamento (PLR), conforme Laporte e Norbert (1981).

O PLR é uma combinação de dois problemas, sendo estes o de localização de instalações e o de roteamento de veículos. Logo, o PLR envolve definir a localização de facilidades, denominados de depósitos, a serem abertos de modo a atender os clientes. Além disso, possui o roteamento de veículos, que tem o objetivo de determinar como será atendida a demanda dos clientes por meio de rotas. Cada rota é realizada por um veículo, partindo de um depósito e finalizando no mesmo depósito de partida. O objetivo é minimizar o custo envolvido para localizar um subconjunto de depósitos mais o custo de estabelecer as rotas entre depósitos e clientes.

O PLR é considerado um problema NP-difícil, uma vez que seus dois subproblemas são reconhecidos como NP-difíceis, conforme aponta Garey e Johnson (1979). Assim, os métodos de solução para o PLR podem ser exatos ou heurísticos. Os exatos retornam a solução ótima, todavia apresentam tempo de execução exponencial e por isso não são utilizados para instâncias grandes. O primeiro algoritmo exato para o PLR foi proposto por Laporte e Norbert (1981), que utilizaram o método branch-and-bound. Existem aplicações também do algoritmo do tipo branch-and-cut por Laporte et al. (1986) e Belenguer et al. (2011).

Entretanto, a maioria das aplicações deste problema são resolvidas através de métodos heurísticos, que geram uma solução aceitável em tempo polinomial, mas nem sempre ótima. Aplicações práticas de algumas heurísticas são evidenciadas em Yu et al. (2010) baseada no método de Recozimento Simulado (Simulated Annealing), Derbel et al. (2010) e Jarboui et al. (2013) que propuseram algoritmos de Busca de Vizinhança.

Este trabalho tem intuito de propor uma abordagem para o PLR usando o método Recozimento Simulado e tendo como base o trabalho de Yu et al. (2010). Desenvolve-se um algoritmo a partir da proposta de Yu et al. (2010) para obter soluções eficientes ou até mesmo ótimas dentro de um tempo computacional aceitável. A Seção 2 apresenta formalmente o problema e como a solução é representada no contexto do Recozimento Simulado. A heurística é formalizada e discutida em detalhes na Seção 3. Os resultados computacionais são expressos na Seção 4, enquanto a Seção 5 traz as conclusões e direções para trabalhos futuros.

2. Caracterização do Problema e Heurística Proposta

O PLR é caracterizado em um grafo não-orientado G = (V, E), sendo V o conjunto de vértices (ou nós), que representa todos os possíveis locais para os depósitos I = {1, 2, ..., M} e os clientes J = {M+1, M+2, ..., M+N}, e E sendo o conjunto de arestas que ligam dois nós de V, com exceção de arestas para depósito-depósito. Para cada aresta {i, j} existe um custo c_{ij} associado. Cada cliente j $\in$ J tem demanda d_j que

deve ser atendida completamente por um único veículo. Cada depósito i $\in$ I tem capacidade W_i e custo de abertura O_i.

Além disso, existe um conjunto K de veículos idênticos, com capacidade Q. Para cada depósito aberto há um custo fixo F_i associado a este. Um veículo só pode executar uma única uma rota e cada rota inicia e finaliza no mesmo depósito, de forma que a carga total na rota não pode ultrapassar a capacidade do veículo. Também, pode-se ter mais de um veículo associado a um depósito, porém não se pode exceder a sua capacidade. O objetivo é minimizar o custo incorrido da escolha dos depósitos a serem instalados e das rotas que vão ser executadas.

2.1. Representação da solução

A solução do PLR é representa por um vetor contendo os depósitos em I, os clientes em J e pelo Nfictício. O Nfictício representa a quantidade de zeros que é necessária para separar as rotas, de modo que a capacidade do veículo não seja excedida. O Nfictício é calculado por $\sum_i (d_i/Q)$ e representa o número mínimo de rotas (ou veículos) necessários para atender a demanda dos clientes. Todavia, neste trabalho, devido às comparações com os resultados de Jarboui et al. (2013), assumiu-se que o depósito possui uma única rota associada.

Assim, na representação da solução, o vetor começa por um depósito e é seguido por seus clientes, caso o depósito seja aberto. Caso o depósito esteja fechado, então ele é seguido de outro depósito ou por dois zeros consecutivos. A Tabela 1 ilustra a solução inicial gerada pela estratégia desse trabalho, considerando a instância coord20-5-1 de Prins et al. (2006).

Tabela 1. Solução inicial da instância coord20-5-1.

1	21	20	8	12	9	2	18	10	22	14	25	23	7	15	3	4	24	6	11	16	17	19	13	5

Fonte: Elaborada pela Autora.

A Tabela 1 apresenta uma instância com 20 clientes e 5 depósitos, sem zeros fictícios. Os depósitos são representados pelos números naturais de 1 a 5, enquanto os clientes seguem dos naturais de 6 até 25.

3. Formalização da Heurística

A heurística desenvolvida para resolver o PLR é baseada na estratégia de Recozimento Simulado e segue o trabalho de Yu et al. (2010). O Recozimento Simulado foi proposto inicialmente por Metropolis et al. (1953) como um método para otimização

termodinâmica, mas posteriormente foi expandido por Kirkipatrick et al. (1982) para problemas de otimização em geral.

3.1. Solução Inicial

A solução inicial é gerada por meio do algoritmo da Figura 3. Em linhas gerais, faz-se a atribuição dos clientes aos depósitos por meio de um método guloso, ou seja, para um novo cliente j, seleciona-se o depósito ou cliente i já alocado tal que c_{ij} é mínimo. Esse processo é repetido até que todos os clientes estejam em alguma rota (atribuídos a algum depósito).

```
1   Enquanto (os clientes não forem todos inseridos) Faça
2      Para (i=0 até número de clientes)
3              Para (j=0 até número de depósitos)
4                 menor = mínimo da distância do depósito i até o cliente j;
5                 depot=i;
6                 cl=j;
7              sol[depot][cl]=menor;
8       Enquanto (capacidade do depósito não for violada) Faça
9              sol[depot][cl] = menor;
10               menor=10000000;
11               Para (l=0 até número de clientes)
12                    menor=menor distância do cliente cl até o cliente l;
13                    cl=l;
14              capacidade do depósito=-1;
15   Para (i=0 até número de depósitos)
16           X[ii]=i+1;
17           Para (j=0 até número de clientes)
18                    Se (sol[i][j] diferente vazio)
19                           depot++;
20                           cliente=j;
21           Se (depot for igual a 1)
22                 X[ii]= cliente;
23                 sol[i][cliente];
24            Se (depot maior ou igual 2)
25                 X[ii]= cliente;
26                 sol[i][cliente];
27                 depot--;
28                 Enquanto (depot não tiver vazio) Faça
29                      Para (l=0 até clientes)
30                           cl=1000000;
31                           Se (sol[i][l] for diferente 0 e sol[i][l]<cl)
32                               cl=sol[i][l];
33                               cliente=l;
```

```
34                      X[ii]=cliente;
35                      sol[i][cliente];
36                      depot--;
```

Figura 3. Rotina para gerar a solução inicial.
Fonte: Elaborada pela Autora.

3.2. Operação de Troca

A rotina de troca é um operador de vizinhança que realiza a troca de posições entre depósitos e/ou clientes selecionados no vetor solução. Também, o valor zero, associado ao Nfictício, pode ser considerado na hora da troca. Em resumo, escolhe-se duas posições aleatórias i e j do vetor solução. A operação de troca considera quatro casos distintos, conforme ilustra o algoritmo da Figura 4.

```
1   i=rand()%tamanho do vetor;
2   j=rand()%tamanho do vetor;
3   Se (X[i] e X[j] são clientes)
4          o cliente na posição i é transferido para j e vice-versa;
5   Se (X[i] e X[j] são depósitos)
6            o depósito na posição i é transferido para j e vice-versa;
7     Se ((X[i] é depósito e X[j] é cliente) ou (X[i] é cliente e X[j]é
depósito))
8         Se (X[i] é depósito e X[j] é cliente)
9                      auxcl=X[j];
10                      auxdep=X[i];
11          Senão
12                      auxcl=X[i];
13                      auxdep=X[j];
14          Para (l=0 até tamanho do vetor X)
15                Se (X[l] = auxdep)
16                   X[l]=0;
17                   os clientes de auxdep são transferidos para o depósito
anterior;
18                   aux1 =depósito anterior a auxdep;
19          Para (l=0 até tamanho do vetor X)
20                Se (X[l]=auxcl)
21                   X[l]=auxdep;
22          Para (l=0 até o tamanho do vetor X)
23                Se (X[l]==aux1)
24                    aux2=X[l];
25                    X[l]=auxdep;
26                    Enquanto (l for menor que o tamanho do vetor X) Faça
27                              l++;
28                              aux1=X[l];
```

```
29                              X[l]=aux2;
30                              aux2=aux1;
```

Figura 4. Rotina para a operação de Troca.
Fonte: Elaborada pela Autora.

As trocas que ocorrem quando os valores aleatórios caem nas posições que contenham um depósito e um cliente, ou vice-versa, ampliam o espaço de busca para se obter uma solução melhor. Com isso, ocasionam uma maior influência na diversificação da solução.

3.3. Operação de Inserção

A operação de inserção consiste em inserir um elemento de uma posição imediatamente antes de outro elemento de outra posição. Escolhe-se duas posições aleatórias, i e j, e considera-se quatro casos para a inserção, sendo que o zero também é considerado como elemento válido. A Figura 5 apresenta o algoritmo para a operação de inserção.

```
1    i=rand()%tamanho do vetor;
2    j=rand()%tamanho do vetor;
3    Se (X[i] e X[j] são ambos clientes)
4         auxcliente=X[i];
5         X[i]=0
6         Para (l=0 até tamanho do vetor X)
7                Se (X[l]=X[j])
8                    o cliente auxcliente é inserido antes de X[l];
9    Se (X[i] e X[j] são depósitos)
10        auxdeposito=X[i]
11        X[i]=0;
12          os clientes atendidos por auxdeposito são transferidos para o
depósito anterior;
13   Se  (X[i] é um cliente e X[j] é depósito)
14         auxcliente=X[i];
15         X[i]=0;
16         Para (l=0 até tamanho do vetor X)
17               Se (X[l]==X[j])
18                   auxcliente é inserido antes do depósito X[l];
19   Se (X[i] é um depósito e X[j] é um cliente)
20         auxcliente=X[i];
21         auxdeposito=X[j];
22         X[i]=0 e seus clientes são transferidos para o depósito anterior;
23        Para (l=0 até o tamanho do vetor X)
```

```
24              Se (X[l]==auxcliente)
25                 auxdeposito é inserido após X[l];
```

Figura 5. Rotina para a operação de Inserção.
Fonte: Elaborada pela Autora.

3.4. Operação de Inversão

A operação de inversão é aplicada para melhorar as rotas que consideram os zeros fictícios como clientes. Ela realiza a inversão de uma sequência de clientes e depósitos, a partir de um determinado elemento, ou seja, escolhe-se aleatoriamente duas posições e a sequência entre essas posições passa a ser considerada em ordem inversa. O algoritmo na Figura 6 ilustra a ideia dessa operação.

```
1   i=rand()%tamanho do vetor;
2   j=rand()%tamanho do vetor;
3     Se (| j-i |>3)
4         Se (o número de posições entre i e j for par)
5            Para (l=0 até (número de posições entre i e j dividido por 2))
6                  aux1 = X[i];
7                  aux2 = X[j];
8                  X[i] = aux2;
9                  X[j] = aux1;
10                 i--;
11                 j++;
12       Senão
13           Para (l=0 até (número de posições entre i e j menos 1, dividido
por 2))
14                aux1 = X[i];
15                aux2 = X[j];
16                X[i] = aux2;
17                X[j] = aux1;
18                i--;
19                j++;
```

Figura 6. Rotina para a operação de inversão.
Fonte: Elaborada pela Autora.

3.5. Busca Local

A rotina de busca local consiste em realizar operações de troca ou de inserção. Para o caso da troca, o algoritmo está ilustrado na Figura 7, e consiste em testar todas as trocas entre os elementos do vetor, exceto pela primeira posição que é um depósito.

```
1   XX= X;
2   Enquanto (i< que o tamanho do vetor) Faça
3              Para (l=1 até o tamanho do vetor X)
4                    Se (XX[l] diferente XX[i])
5                        os valores são trocados de posições;
6                        Se (custo XX for melhor que X)
7                             auxX=XX;
8                             XX=X;
9                        Senão
10                             XX=X;
11                       i++;
```

Figura 7. Operação de troca na Busca Local.
Fonte: Elaborada pela Autora.

A operação de inserção na busca local é apresentada no algoritmo da Figura 8. No caso da inserção, considera-se inserir cada elemento em todas as posições do vetor solução. A melhor solução passa a ser considerada como solução final após aplicação dessa operação.

```
1   XX= X;
2   Enquanto (i < que o tamanho do vetor X) do
3             Para (l=1 até o tamanho do vetor X)
4                Se (XX[l] é diferente XX[i] e l é diferente de 0)
5                      XX[l] é inserido antes de XX[i];
6                    Se (custo XX for melhor que X)
7                        auxX=XX;
8                        XX=X;
9                    Senão
10                        XX=X;
11            i++;
```

Figura 8. Operação de inserção na Busca Local.
Fonte: Elaborada pela Autora.

3.6. Heurística Completa

A heurística utilizada foi obtida do trabalho de Yu et al. (2010), sendo referenciada por SALRP. Ela possui sete parâmetros principais I_{iter}, T_0, T_f, K, P, $N_{non\text{-}improving}$ e α. Os valores iniciais atribuídos para estes parâmetros influenciam na convergência da solução, por isso testes de calibração foram efetuados para determinar quais parâmetros utilizar.

O I_{iter} indica o número de iterações a realizar caso a temperatura não atinja zero, enquanto T_0 representa a temperatura inicial e T_f denota a temperatura final. K é a constante de Boltzmann usada para calcular a probabilidade de se aceitar soluções ruins. P é a penalidade aplicada quando ocorre violação da capacidade do depósito e é calculada a partir do custo de abrir os depósitos, ou seja, por $(\sum_i CD[i])/M/2)$, em que CD[i] é custo para abrir o depósito i. $N_{non\text{-}improving}$ indica o número de reduções consecutivas que podem ser realizas na temperatura e α é um coeficiente que controla o processo de arrefecimento da temperatura.

No início da heurística SALRP, a temperatura T é definida como sendo igual a T_0. Posteriormente, gera-se a solução inicial X para, em seguida, obter uma nova solução Y com base na aplicação da operação de troca, inserção ou inversão. O tipo de operação é escolhido de forma aleatória, em que todos têm a mesma probabilidade de acontecer.

A solução Y é avaliada por meio da função de probabilidade $\exp(-\Delta / K T)$ que vai indicar se está nova solução deve ser aceita ou não, caso ela seja pior do que X. Para Δ usa-se a diferença do valor de solução entre Y e X. A cada iteração executada, a temperatura é reduzida por um fator α, enquanto a solução em X passa pela Busca Local, com operações de troca e inserções. O algoritmo termina quando a temperatura T for igual a T_f ou quando a melhor solução X não for melhorada após $N_{non\text{-}improving}$ reduções de temperatura.

4. Testes Computacionais

Foram realizados testes computacionais a partir de instâncias disponíveis em Prins et al. (2006), Tuzun e Burke (1999) e Barreto (2004). Os parâmetros para rodar a heurística SALRP foram obtidos de Yu et al. (2010) e correspondem a α=0.98, T_0=30, T_f=0.1, K=1/9 e $N_{non\text{-}improving}$=12. O valor de I_{iter} varia de acordo com a instância a ser executada, mas neste trabalho limitou-se o seu valor para atingir no máximo 5000. O computador usado nos testes foi um Intel Core i5 uPro, 4 GB de memória RAM e sistema operacional Linux.

Nas Tabelas 2 a 5 estão os resultados obtidos para as instâncias mencionadas anteriormente. Cada tabela possui o nome da Instância, com a quantidade n de clientes e m de depósitos, o Número de Veículos utilizados para atender a demanda dos clientes, o Tempo computacional gasto em segundos para encontrar a solução, a

Melhor Solução conhecida (obtido de Jarboui et al. (2013)) e o Desempenho da heurística em comparação com a melhor solução, o qual é calculado como: 100 – (((Valor da Solução – Melhor Solução conhecida) ÷ Melhor Solução conhecida) × 100). O número de veículos gastos para realizar a distribuição das mercadorias é calculado como sendo a demanda total atendida por um depósito sobre a capacidade do veículo.

A Tabela 2 apresenta os resultados obtidos para as instâncias de Prins et al. (2006), a qual fornecia 30 instâncias variando entre 20 a 200 clientes e entre 5 a 10 depósitos. Observou-se que quatro instâncias apresentaram soluções igual à melhor conhecida e outras quatro instâncias tiveram desempenho inferior a 80%, sendo a instância com pior valor intitulada 20-5-2. O tempo computacional gasto para rodar as instâncias variou foi de até 580 segundos, sendo que os maiores tempos foram gastos por instâncias que continham muitos clientes.

Tabela 2. Resultado para Prins et al. (2006).

Instância (n x m)	Veículo	Tempo (s)	Valor da Solução	Melhor Solução Conhecida	Desempenho (%)
20-5-1 (20x5)	5	0	21.360,00	21.360,00	100,00
20-5-1b (20x5)	3	1	15.662,00	15.662,00	100,00
20-5-2 (20x5)	5	1	22.929,00	14.708,00	44,11
20-5-2b (20x5)	3	7	14.095,00	14.095,00	100,00
50-5-1 (50x5)	13	12	15.080,00	15.076,00	99,97
50-5-1b (50x5)	6	6	15.082,00	15.077,00	99,96
50-5-2 (50x5)	14	18	29.578,00	29.562,00	99,94
50-5-2b (50x5)	7	13	29.581,00	29.562,00	99,93
					(continuação)
Instâncias (n x m)	**Veículo**	**Tempo (s)**	**Valor da Solução**	**Melhor Solução conhecida**	**Desempenho (%)**
50-5-2bbis (50x5)	6	2	17.910,00	17.889,00	99,88
50-5-3 (50x5)	18	13	11.002,00	11.002,00	100,00
50-5-3b (50x5)	6	18	11.007,00	11.002,00	99,95
100-5-1 (100x5)	23	27	133.292,00	133.254,00	99,97
100-5-1b (100x5)	12	27	133.292,00	133.254,00	9997
100-5-2 (100x5)	23	52	97.037,00	97.032,00	99,99
100-5-2b (100x5)	11	35	97.031,00	97.029,00	99,99
100-5-3 (100x5)	27	128	86.638,00	86.610,00	99,96
100-5-3b (100x5)	12	179	86.643,00	86.610,00	99,96
100-10-1 (100x10)	27	159	196.737,00	155.407,00	73,41
100-10-1b (100x10)	13	102	196.756,00	155.412,00	73,39
100-10-2 (100x10)	22	242	141.483,00	141.414,00	99,95
100-10-2b (100x10)	12	104	141.413,00	141.404,00	99,96
100-10-3 (100x10)	26	162	139.792,00	136.499,00	97.58
100-10-3b (100x10)	12	166	139.792,00	136.531,00	97,61
200-10-1 (200x10)	45	546	236.858,00	236.756,00	99,96
200-10-1b (200x10)	22	566	236.858,00	236.764,00	99,96

200-10-2 (200x10)	47	543	390.656,00	276.903,00	58,92
200-10-2b (200x10)	22	240	276.986,00	276.902,00	99,96
200-10-3 (200x10)	45	55	235.135,00	235.110,00	99,98
200-10-3b (200x10)	23	580	235.280,00	235.119,00	99,93
Total:	523	4021	-	**Desempenho Médio:**	94,80

Fonte: Elaborada pela Autora.

A Tabela 3 traz os resultados para as 36 instâncias de Tuzun e Burke (1999), as quais possuem número de clientes e depósitos maiores do que nas instâncias de Prins et al. (2006) e Barreto (2004). Os valores das soluções para estas instâncias apresentaram um desempenho acima de 90%, exceto para uma única instância que obteve 57,40%. O tempo computacional gasto para resolver as instâncias teve variação entre 57 e 453 segundos.

Tabela 3. Resultado para Tuzun e Burke (1999).

Instância (n x m)	Veículos	Tempo (s)	Valor da Solução	Melhor Solução conhecida	Desempenho (%)
111112 (100x10)	11	81	901,00	881,00	97,79
111122 (100x20)	11	89	830,00	817,00	98,41
111212(100x10)	10	57	801,00	785,00	97,96
111222(100x20)	11	136	827,00	818,00	98.90
112112(100x10)	11	105	649,00	632,00	97,31
112122(100x20)	11	212	579,00	574,00	99,13

(continuação)

Instâncias (n x m)	Veículos	Tempo (s)	Valor da Solução	Melhor Solução conhecida	Desempenho (%)
112212 (100x10)	11	74	389,00	359,00	91,64
113112 (100x10)	11	94	587,00	563,00	95,74
113122(100x20)	11	73	628,00	600,00	95,33
113212 (100x10)	11	142	636,00	446,00	57,40
113222 (100x20)	11	177	453,00	449,00	99,11
121112 (200x10)	20	367	1.119,00	1.055,00	93,93
121122 (200x20)	21	305	1.141,00	1.045,00	90,81
121212 (200x10)	20	274	1.146,00	1.103,00	96,10
121222 (200x20)	20	284	1.150,00	1.061,00	91,61
122112 (200x10)	21	380	793,00	751,00	94,41
122122 (200x20)	20	410	805,00	778,00	96,53
122212 (200x10)	20	403	509,00	491,00	96,33
122222 (200x20)	21	440	495,00	473,00	95,35
123112 (200x10)	21	296	812,00	758,00	92,88
123122 (200x20)	20	310	897,00	819,00	90,48
123212 (200x10)	21	386	514,00	483,00	93,58
123222 (200x20)	21	453	519,00	511,00	98,43
131112 (300x10)	15	81	1.058,00	1.021,00	96,38

131122 (300x20)	15	147	1.001,00	963,00	96,05
131212(300x20)	16	120	1.014,00	959,00	94,26
131222 (300x20)	15	156	995,00	968,00	97,21
132112 (300x10)	15	189	659,00	632,00	95,73
132122 (300x20)	15	197	672,00	649,00	96,46
132212 (300x10)	15	241	412,00	403,00	97,77
132222 (300x20)	16	239	369,00	363,00	98,34
133112 (300x10)	15	209	704,00	686,00	97,38
133122 (300x20)	15	166	620,00	611,00	98,53
133212 (300x10)	16	206	435,00	428,00	98,36
133222 (300x20)	15	204	505,00	484,00	95,66
Total:	561	7818	-	**Desempenho Médio:**	94,90

Fonte: Elaborada pela Autora.

Por fim, a Tabela 4 tem os resultados das instâncias fornecida por Barreto (2004), com número de clientes variando entre 21 a 150. Nesses resultados fica evidente que o algoritmo retornou boas soluções, uma vez que obteve quatro valores iguais aos melhores conhecidos na literatura. Além disso, as demais instâncias apresentaram desempenho superior a 90%, exceto para a Min92, que teve desempenho de 82,34%. Ademais, o tempo gasto para encontrar as soluções limitou-se em 27 segundos, muito inferior aos demais casos.

Tabela 4. Resultado para Barreto (2004).

Instância (n x m)	**Veículos**	**Tempo (s)**	**Valor da Solução**	**Melhor Solução conhecida**	**Desempenho (%)**
Christofides69-(50x5)	6	1	451,56	450,90	99,85
Christofides69-(50x5)	9	10	581,31	568,29	97,70
Christofides69-(100x10)	8	10	687,23	677,47	98,56
Daskin95-(88x8)	6	12	312,96	296,82	94,56
Daskin95-(150x10)	12	27	41.621,99	40.033,25	96,03
Gaskell67-(21x5)	5	0	370,73	370,73	100
Gaskell67-(22x5)	3	0	519,96	519,96	100
Gaskell67-(29x5)	4	0	434,26	434,26	100
Gaskell67-(32x5)	5	0	441,71	441,22	99,89
Gaskell67-(32x5) bis	3	1	444,45	441,22	99,27
Gaskell67-(36x5)	4	0	410,89	410,00	99,78
Min92-(27x5)	4	1	2484	2.484,04	100
Min92-(134x5)	11	17	5.778,7	4.911,32	82,34
Total:	80	79	-	**Desempenho Médio:**	97,54

Fonte: Elaborada pela Autora.

5. Conclusões

Este trabalho discutiu a heurística de Yu et al. (2010) para a versão capacitada do Problema de Localização e Roteamento, fornecendo rotinas para calcular operações de vizinhança baseadas em troca, inserção e inversão, além de uma busca local.

A heurística foi codificada em linguagem C e testada em três conjuntos de instância da literatura. Percebeu-se que o algoritmo tem melhor desempenho em instâncias que apresentam um menor número de clientes, principalmente quando se compara o tempo computacional gasto. Na média de desempenho, destaca-se que a heurística ficou acima dos 95% quando se analisa todo o conjunto de instâncias, demonstrando um resultado satisfatório para aplicações voltadas para o interesse real.

Alguns pontos para trabalhos futuros incluem combinar esse heurística com uma Busca em Vizinhança Variável e considerar novos operadores de vizinhança.

6. Agradecimentos

Os autores agradecem o apoio financeiro recebido das agências CNPq (471351/2012-1), FAPEG e FAPESP (2014/16906-1).

The Simulated Annealing Method Applied in Location and Routing

Abstract: In this work we investigated the vehicle routing problem integrated with the facility location, in which both decisions are taken simultaneously. This integrated problem considers limited capacity for the depots and vehicles, with the objective to minimize the overall cost to locate depots and serve customers by routes from these depots. Therefore, we present a simulated annealing based heuristic, which includes neighborhood operations and a local search routine. Numerical tests shown that the heuristic is competitive and allows solving large sized instances after improvements.

Keywords: Location-Routing Problem; Simulated Annealing; Heuristic.

Referências bibliográficas

BARRETO, S. S. **Análise e Modelização de Problemas de Localização-Distribuição.** 2004. 357 f. Tese de Doutorado – Universidade de Aveiro, Aveiro, Portugal, 2004.

BELENGUER, J. M. et al. **A Branch-and-Cut method for the Capacitated Location-Routing Problem.** . Computers and Operations Research, v. 38, n. 6, p. 931–941, 2011.

DERBEL, H. et al. **An iterated local search for solving a location-routing problem.** Electronic Notes in Discrete Mathematics, v. 36, n. 1, p. 875-882, 2010.

DUHAMEL, C. et al. **A GRASPxELS approach for the capacited location-routing problem.** Computers and Operations Research, v. 37, n. 11, p. 1912-1923, 2009.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability: A Guide to the theory of NP-Completeness.** San Francisco: Freeman, 1979.

GONÇALVES, R. F.; QUEIROZ, T. A. **The Knapsack Problem with Three Practical Constraints.** Procedia Computer Science, v. 29, p. 2192-2200, 2014.

JARBOUI, B. et al. **Variable neighborhood search for location routing.** Computers and Operations Research, v. 40, n. 1, p. 47-57, 2013.

KIRKPATRCK, S.; GELATT, C. D.; VECCHI, M. P. **Optimization by simulated annealing, Science,** v. 220, p. 671-680, 1982.

LAPORTE, G.; NORBERT, Y. **An exact algorithm for minimizing routing and operating costs in depot location.** European Journal of Operational Research, v. 6, n. 2, p. 224-226, 1981.

LAPORTE, G.; NORBERT, Y.; ARPIN, D. **An exact algorithm for solving a capacitated location-routing problem.** Annals of Operations Research, v. 6, n. 9, p. 293-310, 1986.

METROPOLIS, N. et al. **Equations of state calculations by fast computing machines. Journal of Chemical Physics**, v. 21, p.1087-1092, 1953.

PRINS, C.; PRODHON, C. CALVO, R. W. **Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking.** 4OR: A Quarterly Journal of Operations Research, v. 4, n. 3, p. 221–238, 2006.

QUEIROZ, T. A.; MIYAZAWA, F. K. **Two-dimensional strip packing problem with load balancing, load bearing and multi-drop constraints.** International Journal of Production Economics, v. 145, p. 511-530, 2013.

QUEIROZ, T. A.; MIYAZAWA, F. K. **Order and static stability into the strip packing problem.** Annals of Operation Research, In press, 2014, DOI: 10.1007/s10479-014-1634-2.

TUZUN, D.; BURKE, L. I. **A two-phase tabu search approach to the location routing problem.** European Journal of Operational Research, v. 116, n.1, p. 87-99, 1999.

V. F. Yu, S.-W. Lin, W. Lee e C.-J. Ting. **A simulated annealing heuristic for the capacitated location routing problem.** Computers & Industrial Engineering, 58:288–299, 2010.