Metaheurísticas GRASP + VNS para a solução de Problemas de Otimização Combinatória

Mozar Baptista da Silva
Lúcia Maria de Assumpção Drummond
Luiz Satoru Ochi

Programa de Pós Graduação em Ciência da Computação – Instituto de Computação
Universidade Federal Fluminense
Endereço para correspondência: Rua Capitão Zeferino, 56/808 , Icarai, Niterói, RJ, 24220-230
E mail:mozar@compuland.com.br, {lucia,satoru}@dcc.ic.uff.br

Abstract

We present metaheuristics based on concepts of Greedy Randomized Adaptive Search procedures (GRASP) and Variable Neighborhood Search (VNS) to solve generalizations of the well – known Traveling Salesman Problem where only a subset of the cities are visited. A comparison of the proposed algorithms with the Tabu Search proposed by Voss [8] is presented, too.

Resumo

Apresentamos neste artigo, algoritmos metaheuristicos baseados em conceitos dos métodos GRASP e VNS para a solução de generalizações do clássico Problema do Caixeiro Viajante onde apenas um subconjunto do conjunto das cidades é visitada. Adicionalmente fornecemos resultados computacionais comparando o desempenho dos algoritmos propostos com um método Tabu Search proposto por Voss [8].

palavras – chave: metaheurísticas, algoritmos adaptativos, otimização combinatória.

.

1. Introdução

As chamadas metaheurísticas ou heurísticas inteligentes têm se mostrado um bom caminho para a solução de problemas NP-Completos e NP-Hard na área de otimização combinatória.

Dentre as metaheurísticas mais conhecidas, podemos citar: Algoritmos Genéticos (AG) [7]; Redes Neurais (RN)[7]; *Scatter Search* (SS) [4]; *Tabu Search*(TS) [4]; GRASP [1]; VNS [3] e *Ant Colony Systems (ACS) [7] e Simulated Annealing* [7]. A principal característica destas metaheurísticas, é a de possuírem ferramentas que procuram superar as armadilhas dos ótimos locais quando ainda estamos distantes de um ótimo global em problemas de otimização.

Outra característica das metaheurísticas, é a sua adaptabilidade. Podemos dizer que as metaheurísticas procuram não ser tão rígidas como os métodos exatos de otimização mas

também não recaindo numa flexibilidade às vezes caótica das heurísticas de busca convencionais.

Dentre as metaheurísticas mais usadas em problemas de otimização, as que têm obtido os melhores resultados são: *Tabu Search (TS)*, GRASP e VNS.

Neste trabalho, propomos basicamente dois algoritmos: Um baseado em conceitos de GRASP e o outro em VNS, além de versões híbridas conjugando GRASP com VNS.

Para analisar o desempenho dos algoritmos propostos, tomamos como base uma generalização do Problema do Caixeiro Viajante (TSP) onde somente um subconjunto do conjunto das cidades é visitado, O *Traveling Purchaser Problem* (TPP) [5, 6, 8].

Neste trabalho, os resultados são comparados com o algoritmo Tabu Search, proposto por Voss [8], por ter na literatura os melhores resultados para o TPP.

Na seção 2 descrevemos o TPP e apresentamos uma nova formulação deste problema como um problema de programação linear inteira e na seção 3, apresentamos um resumo dos métodos TS, GRASP e VNS. Na seção 4 apresentamos os algoritmos propostos e na seçao 5 descrevemos os resultados computacionais e as conclusões.

2 – O *Traveling Purchaser Problem* (TPP)

Uma das generalizações do TSP onde somente parte do conjunto dos nós é visitado numa solução válida, é conhecida na literatura como *The Traveling Purchaser Problem* (TPP) [6]. O TPP é um problema NP-Hard e pode ser descrito da seguinte forma:

- Existe um conjunto de n lojas $N = \{ 1, 2, ..., n\}$ mais uma origem $s = \{0\}$.
- Existe um conjunto de m produtos $K = \{ 1, 2, ..., m\}$ que devem ser adquiridos nestas n lojas.
- Existe uma matriz $D = (d_{kj})$ tal que $k \in K$, $j \in N$, onde d_{kj} representa o custo do produto k na loja j.
- Existe uma matriz $C = (c_{ij})$ tal que $i, j \in N \cup \{0\}$, onde c_{ij} representa o custo de se locomover de i para j.
- Assumimos também que cada produto está disponível em pelo menos uma loja; que na origem s = 0 não existem nenhum produto disponível; e que o viajante pode passar por uma loja mais de uma vez se necessário sem adquirir nenhum produto.

O objetivo do TPP, é gerar uma trajetória fechada passando pela origem s = 0 e por um subconjunto $J \subseteq N$, de modo que a soma dos custos de compra dos m produtos e do percurso seja minimizada.

Embora o TPP possua várias aplicações práticas principalmente nas áreas de roteamento e escalonamento, este problema ainda não é muito explorado pela literatura afim. A maioria das técnicas existentes para o TPP propõe o uso de algoritmos heurísticos convencionais e de nosso conhecimento só existem um método TS proposto por Voss[8] além de um algoritmo genético paralelo proposto por Ochi e outros [5].

2.1 - Formulação Matemática Proposta

Propomos uma formulação matemática para o TPP onde se pressupõe um grafo completo e usando distâncias euclidianas.

Seja G = G(M,E) um grafo direcionado associado com o TPP. Definem-se as seguintes variáveis:

- x_{ij} = 1, se o arco é utilizado, e = 0, caso contrário;
- y_{kj} = 1 se o item k foi comprado no mercado j; 0 caso contrario;
- T = (t_{ij}): 0 < i, j < n matriz que representa o custo de mover-se do mercado i para o mercado j;
- P = (p_{kj}): 1 < k < m, 1 < j < n, onde p_{kj} representa o custo do item k no mercado j;
- A = (a_{kj}): 1 < k < m, 1 < j < n = 1, se o item k é oferecido pelo mercado j e = 0 caso contrário;
- F = (f_{ij}): 0 < i,j < n representa o fluxo (numero de items) transportado de i para j, onde $f_{ij} \geq 0$ e inteiro.

$$V(TPP) = \text{minimizar} \sum_{i,j \in M} t_{ij}\, x_{ij} + \sum_{k \in K,\, j \in M} p_{kj}\, y_{kj} \qquad (2.1)$$

sujeito à:

$$\sum_{j \in M} a_{kj}\, y_{kj} = 1, \forall\, k \in K \qquad (2.2)$$

$$\sum_{j \in M} x_{ij} = \sum_{j \in M} x_{ji} = 1, \text{ para } i = m_0 \qquad (2.3)$$

$$\sum_{i \in M - m0} x_{ij} = \sum_{i \in M - m0} x_{ji}, \forall\, j \in M - m_0 \qquad (2.4)$$

$$\sum_{i \in M} x_{ij} \geq \sum_{k \in K} a^{*}_{kj}\, y_{kj}, \forall\, j \in M - m_0, \text{ onde } a^{*}_{kj} = \frac{a_{kj}}{\sum_{p \in K} a_{pj}} \qquad (2.5)$$

$$\sum_{j \in M - m0} f_{m0\, j} = 1 \qquad (2.6)$$

$$\sum_{p \in M} f_{jp} = \sum_{i \in M - m0} f_{ij} + \sum_{k \in K} y_{kj}, \quad \forall\, j \in M - m_0 \qquad (2.7)$$

$$\sum_{j \in M - m0} f_{jm0} = m + 1 \qquad (2.8)$$

$$f_{ij} > x_{ij} - 1, \forall\, i,j \in M \qquad (2.9)$$

$$x_{ij} \geq \frac{f_{ij}}{m+1}, \forall\, i,j \in M$$

$$x_{ij} \in \{0,1\}, \forall\, i,j \in M \qquad (2.10)$$

$$y_{kj} \in \{0,1\}, \forall\, k \in K, \forall\, j \in M \qquad (2.11)$$

$$f_{ij} \geq 0, \text{ e inteiro}, \forall\, i,j \in M \qquad (2.12)$$

Na função objetivo equação 2.1, o primeiro termo, representa o custo de rota e o segundo o custo dos k itens adquiridos. Portanto devemos minimizar cada termo para conseguir atingir o ótimo global.

As equações 2.2 garantem que cada um dos m itens é adquirido em algum mercado.

Para garantirmos que o comprador inicie da origem e retorne a mesma temos as equações 2.3, que para ser mais geral deveria ser ≥ 1 , ao invés de = 1, porque o comprador pode passar várias vezes pela origem. Entretanto como estamos trabalhando com distância euclidiana e grafo completo podemos simplificar essas equações e considerar que o comprador retorne somente uma vez para origem.

As equações 2.4 representam as equações de conservação de fluxo.

As inequações 2.5 garantem que se pelo menos um item é comprado no mercado j, então o comprador deve passar pelo mercado j.

As próximas restrições são referentes ao fluxo de itens, ou seja, a cada item adquirido, o fluxo é incrementado, portanto se temos um fluxo f_0 ao chegarmos no mercado j e neste mercado é comprado um item k, então ao sairmos do mercado j temos um fluxo $f_j = f_0 + 1$, esta restrição está descrita nas equações 2.7. Na origem compramos um item fictício como mostra a equação 2.6. Ao retornar a origem teremos adquirido todos os itens, inclusive o fictício, portanto o fluxo de chegada na origem é m+1, representado na equação 2.8.

As inequações 2.9, garantem, que se o comprador passou pela aresta (i,j) então temos um fluxo positivo nesta aresta. As inequações 2.10 garantem a volta, ou seja, se temos fluxo na aresta (i,j) então o comprador deve passar pela mesma.

A variável binária a_{kj} pode ser eliminada deste modelo, para tanto quando um item k não for oferecido por um mercado j, o custo deste item neste mercado será M, onde M deverá ser um valor bem alto. Como queremos minizar o custo, este produto não será adquirido neste mercado. Com esta idéia podemos fazer as seguintes simplificação:

As equações 2.2 após a simplificações ficariam da seguinte forma:

$$\sum_{j \in M} y_{kj} = 1, \forall k \in K$$

A simplificação das equações 2.5, resultariam nas seguintes equações:

$$\sum_{i \in M} x_{ij} \geq \sum_{k \in K} \frac{y_{kj},}{|K|} \quad \forall j \in M - m_0$$

3 - Metaheurísticas

Dentre as Metaheurísticas existentes, três em particular têm se destacado pelos resultados que tem obtido em diversas aplicações da área de otimização combinatória: Um deles é o método Tabu Search, desenvolvido por Fred Glover [4]. Nas versões mais sofisticadas, métodos do tipo TS incorporam fases de intensificação e diversificação bem como estratégias de listas tabu dinâmicas que os tem colocado como uma das melhores opções em termos de algoritmos aproximados [7, 8].

Duas Metaheurísticas mais recentes mas que tem produzido resultados altamente promissores, são o *Greedy Randomized Adaptive Search* (GRASP), proposto por Feo e Resende [1] e o *Variable Neighborhood Search* (VNS) proposto por N. Mladenovic [3].

3.1 - *Greedy Randomized Adaptive Search Procedure* (GRASP)

A metaheurística GRASP é um método iterativo, onde cada iteração é composta basicamente por uma etapa de construção seguida de uma etapa de busca local de uma solução. Este par de operações é então repetido um determinado número de vezes.

Na etapa de construção, iterativamente se define elemento a elemento de uma solução através de uma lista de candidatos (LC) previamente ordenados segundo sua aptidão. Dentre estes candidatos, seleciona-se um candidato aleatoriamente. A escolha dos candidatos desta lista é dita adaptativa, já que é guiada por uma função gulosa que leva em conta as informações dos elementos anteriormente selecionados. O GRASP também tem sua componente aleatória, devido a escolha aleatória de um candidato na LC. A solução obtida pelo GRASP na sua fase de construção não é necessariamente um ótimo local, portanto é sempre aconselhável o uso de técnicas de busca local para tentar melhorar a qualidade da atual solução dentro de uma estrutura de vizinhança pré determinada.

A melhor solução obtida até o momento é armazenada e retorna-se à etapa de construção até que um critério de parada seja acionado.

3.2 - *Variable Neighborhood Search* (VNS)

As metaheurísticas conhecida como VNS[3], ao contrário das outras metaheurísticas baseadas em busca local, não segue uma trajetória de busca, mas sim explora a partir de uma solução inicial, uma seqüência crescente de estruturas de vizinhança. Considere N_k , (k = 1, 2, ..., kmáx) um conjunto finito de estruturas de vizinhanças previamente definidos e $N_k(x)$ o conjunto de vizinhanças definidas à partir de uma solução x.

Os passos de um VNS básico são: Seleção de um conjunto *kmáx* de estruturas de vizinhanças N_k , (k = 1, 2, ..., *kmáx*); escolha de uma solução inicial *x* e um critério de parada; Partindo de uma solução inicial, analisamos a primeira vizinhança $N_1(x)$; caso numa vizinhança N_k não tenhamos atualizado a melhor solução, passamos a analisar a próxima vizinhança N_{k+1}, caso contrário, se em N_k a melhor solução obtida até o momento for atualizada, retornamos a vizinhança N_1. O critério de parada é acionado quando no último $N_{kmáx}$ não tivermos obtido nenhuma melhoria.

3.3 - Busca Tabu

A busca tabu, foi inicialmente proposta por Glover [4]. Ela pode ser descrita como uma heurística de alto nível para resolver problemas de otimização, desenvolvida para guiar heurísticas de busca local, a evitar a armadilha do ótimo local. A maioria das metaheurísticas são técnicas adaptativas de busca que visam, uma exploração inteligente do espaço de busca. Para conseguir seus objetivos, a Busca Tabu utiliza duas técnicas para direcionar a pesquisa. A primeira é a lista tabu, que guarda as soluções recentemente visitadas, para tentar evitar a formação de ciclos. A segunda é o uso de memórias que auxiliaram na busca de regiões do espaço de soluções pouco exploradas e dificultando a pesquisa em regiões muito exploradas.[9,10].

A lista tabu, normalmente, contém os atributos dos movimentos reversos que devem ser proibidos ou desestimulados (movimentos Tabu). Esses movimentos permanecem por um certo tempo na lista tabu. Esse tempo é chamado de prazo tabu. Um movimento tabu pode ser realizado caso ele satisfação ao critério de aspiração. O critério de aspiração, poderia ser, por

exemplo, aceitar um movimento, que gere uma solução melhor, que a melhor solução encontrada até o momento, mesmo tendo ele o status de tabu [11].

A eficiência dos algoritmos Tabu dependem dos seguintes parâmetros:

- definição da vizinhança,
- a geração da solução inicial,
- a gerência da lista tabu,
- critério de aspiração
- critério de parada.

Nos implementamos dois algoritmos, baseados nos algoritmos propostos por Voss [8], O primeiro usa a gerência chamada de *Cancellation Sequence Method* (CSM) e o Segundo algoritmo usa a gerência *Reverse Elimination Method* (REM).

4 - Algoritmos Propostos

Propomos neste trabalho, basicamente três metaheurísticas para o TPP, mas que facilmente podem ser adaptados para outras generalizações do TSP onde apenas parte do conjunto de nós é visitado numa solução.

As metaheurísticas GRASP e VNS possuem em comum o fato de exigirem um algoritmo de construção de uma solução e algoritmos de busca local. Desta forma, descrevemos a seguir, um conjunto de algoritmos de construção e de busca local para serem utilizadas dentro das metaheurísticas GRASP e VNS aqui propostos.

4.1 - Algoritmos de Construção para o TPP

1. **ADD**: A cada passo deste algoritmo, uma nova loja (nó) que fornece a melhor economia é inserida na solução parcial. O algoritmo pára quando todos os produtos forem adquiridos.
2. **DROP**: Este algoritmo inicia com uma solução que contém todas as lojas (nós) como no TSP e a cada passo, é selecionado uma loja (nó) para ser retirado da atual solução, se este procedimento produzir uma economia e se na solução remanescente, tivermos todos os m produtos a serem adquiridos. O algoritmo pára quando nenhuma economia for observada com a retirada de um nó.
3. **ADDGENI**: Este algoritmo difere do ADD em relação ao cálculo da economia . Aqui o critério de insersão é baseado na heurística GENI [2] ao invés do cálculo das economias usadas no ADD.
4. **DROPGENI**: Usa o procedimento de remoção de nós do GENI[2] dentro do algoritmo DROP.
5. **RandomADD**: A cada passo, a cada passo do algoritmo ADD, seleciona-se uma lista dos k melhores candidatos (nós) a serem inseridos naquele momento, e destes seleciona-se um nó aleatoriamente.
6. **RandomDROP**: Similar ao anterior, agora usando como base o algoritmo DROP.

4.2 - Algoritmos de Busca Local para o TPP

1. **ADDSearch**: Aplica o ADD sobre uma solução inicial até que nenhuma economia seja obtida.

2. **DROPSearch**: Aplica o DROP sobre uma solução inicial até que nenhuma economia seja obtida.
3. **ADDDROPSearch**: Aplica alternadamente o ADD e o DROP até que nenhuma melhoria seja observada.
4. **DROPADDSearch**: Remove um nó e depois aplica uma série de ADD, quando nenhuma economia for observada, repete DROP e outra sequência de ADD, até que um critério de parada seja acionado.
5. **SwapSearch**: Permuta a posição de dois nós não adjacentes da solução inicial . Repete este procedimento h vezes, onde h é um parâmetro de entrada.
6. **NeighSearch**: Constrói uma lista de k nós aleatoriamente escolhidas da atual solução. Estes nós são removidos da atual solução e ficam proibidos de retornar durante as k próximas soluções, após o qual eles perdem o status de solução Tabu e passam a ser candidatos viáveis novamente.
7. **VNSSearch**: Executa o NeighSearch num loop: Inicialmente (k=1) nó é retirado da atual solução, numa Segunda etapa, (k=2) nós são retirados simultaneamente até numa h-ésima etapa onde (k=h) nós são retirados simultaneamente, a cada etapa de retirada de nós, são realizadas novas insersões se isso produzir melhorias na atual solução . Este algoritmo trabalho como um VNS.
8. **Híbrido**: Neste algoritmo há uma união de vários algoritmos de busca. Primeiro executa-se uma vez o DROPADDSearch e o ADDDROPSearch, depois executa-se um loop contendo o ADDSearch, DROPSearch e SwapSearch, o critério de parada do loop e até que não haja melhora na soluçao.

4.3 - Algoritmos GRASP, VNS e GRASP+VNS

Para o algoritmo GRASP, temos para a geração da solução inicial os algoritmos: ADDGENI, DROPGENI, RandomADD, RandomDROP, que são algoritmos gulosos para a geração da solução inicial. e para a busca local temos: ADDSearch, DROPSearch, ADDDROPSearch, DROPADDSearch, SwapSearch e Híbrido. Temos então um total de 24 algoritmos do tipo GRASP.

Para o VNS podemos usar todos os algoritmos para a solução inicial e a busca usamos o VNSSearch, usa o NeighSearch que pode utilizar os seguintes algoritmos de busca local : ADDSearch, DROPSearch, ADDDROPSearch, DROPADDSearch, SwapSearch e Híbrido. Temos então um total de 36 algoritmos do tipo VNS.

Podemos também combinar o GRASP com o VNS, formando o VNS+GRASP, que consiste de um algoritmo GRASP, onde na busca local utilizamos um algoritmo VNS. Desta forma podemos combinar os algoritmos de geração de soluções inicial usados no GRASP, com os algoritmos de busca local usados no VNS, teremos então mais 24 algoritmos.

5 - Resultados Computacionais e conclusões

Para avaliar o desempenho dos Algoritmo GRASP e VNS aqui propostos, utilizamos todas as combinações de construção e busca local descritos na etapa anterior. As duas melhores combinações de cada método: GRASP e VNS que obtiveram os melhores resultados em testes preliminares foram adotados para os teste complementares. Além dos algoritmos propostos foram implementadas várias versões do *Tabu Search*, usando fases de intensificação e diversificação, além de listas tabu dinâmica: *Reverse Elimination Method* (REM) e *Cancelation*

Sequential Method (CSM), ambos propostos por S. Voss[8]. As duas melhores combinações (construção e busca local) para GRASP, VNS e TABU foram as seguintes:

- TABU – CSM usando ADDDROP .
- TABU – REM usando DROP
- GRASP1: usando RandomADD na SI e Híbrido na BL
- GRASP2: usando ADDGENI na SI e DROPADDSearch na BL
- VNS1: usando DROPGENI na SI e Híbrido na BL
- VNS2: usando DROP na SI e DROPADD na BL
- GRASPVNS1:usando ADDGENI na SI e Híbrido na BL
- GRASPVNS2: usando RandomADD na SI e Híbrido na BL

Todos os algoritmos TABU, GRASP e VNS usaram como critério de parada um tempo máximo de 600 segundos.

As instâncias geradas aleatoriamente devido a não existência de nosso conhecimento de bibliotecas públicas para o TPP, tiveram as seguintes dimensões: número de produtos: de 50 a 500 e número de lojas: de 50 a 500. Os custos de cada produto e de cada arco (distância) foram geradas aleatoriamente dentro de um intervalo pré estabelecido.

Algoritmo	Erro médio (%)	Número médio de melhores soluções (%)	Tempo de execução (segundos) (*)
VNS1	0.19	47.22	318.56
VNS2	0.11	44.44	324.97
GRASP1	0,09	52,78	340,72
GRASP2	0,16	47,22	343,55
GRASPVNS1	0.04	63.89	327.31
GRASPVNS2	0.09	55.56	279.50
TABU-CSN	0.41	52.78	234.75
TABU-REM	0.44	47.22	234.19

Tabela 1: Resultados médios entre as 6 melhores versões, onde (*) significa o tempo onde a melhor solução foi encontrada.

Na tabela 1, mostramos os resultados médios obtidos por cada uma das seis versões, nos diversos testes realizados para as diversas instâncias. O resultado mostrado é uma média geral . Na primeira coluna estão os oito algoritmos (os dois melhores de cada técnica) de melhor desempenho, na segunda coluna mostramos a média da margem de erro em relação a melhor solução obtida dentre todos os algoritmos em cada instância, na coluna 3 mostramos o número de vezes em que um método obteve a melhor solução comparando todos os métodos, e na coluna 4, mostramos o tempo de execução médio onde foi encontrado cada melhor solução por cada método, dentro do limite de 600 segundos. Observe que estes valores representam uma média de todos os testes realizados.

Diante dos resultados da Tabela1, verificamos que em termos de erros, os métodos GRASPVNS foram superiores, seguidos pelos métodos VNS, GRASP e TABU. Pela coluna 3, verificamos que na média, GRASPVNS, GRASP e TABU conseguiram os melhores resultados em maior número de vezes. Em função dos tempos, a média dos TS foram melhores seguidos pelo GRASPVNS, GRASP e VNS. Uma observação importante é que as duas versões GRASPVNS que combina GRASP com VNS obtiveram resultados bem expressivos, mostrando que vale a pena investir na combinação das metaheurísticas afim de aproveitar as suas melhores características.. Podemos concluir que as três técnicas apresentaram resultados mais ou menos previsíveis diante da expectativa que existia anteriormente. Com certeza estes resultados

superam resultados de outras metaheurísticas diante do desempenho apresentado pelos três: TS, GRASP e VNS em outras aplicações mostradas pela literatura (veja [1], [5], [8]).

7. Referências

[1] T. Feo and M. G. Resende. Greedy Randomized Adaptive Search Procedures. J. of Global Optimizations 6: 109-13. 1995.

[2] M Gendreau, A Hertz and G. Laporte. New insertion post-optimization procedures for the traveling salesman problem. Op. Res. 40: 1086-1094. 1992.

[3] N. Mladenovic. A variable neighborhood algorithm – a new metaheuristic for optimization combinatorial. Abstract of papers presented at Optimization Days, Montreal 12, 1995.

[4] F. Glover and M Laguna. Tabu Search . Kluwer Academic Pub. 1998.

[5] L. S. Ochi, R. M. V. Figueiredo and L.M. A Drummond. Design and Implementation of a Parallel Genetic Algorithm for the Travelling Purchaser Problem. APPLIED COMPUTING'97/ACM: 257-263 Editors**:** B.Bryant; J.Carroll;D.Oppenhein;J.Hightower; and K.George. Association for Comp. Machinery,Inc., NY.

[6] T. Ramesh. Traveling purchaser problem. Opsearch 18: 78-91. 1981.

[7] C. R. Reeves (editor). Modern Heuristics Techniques for Combinatorial Problems. Blackwell Scientific Publications. 1993.

[8] S. Voss. Dynamic Tabu Search strategies for the Traveling Purchaser Problem. Annals of Op. Research 63 : 253-275. 1998.

[9] Crainic, T. G., Toulouse, M., Gendreau, M., Toward a taxonomy of parallel tabu search heuristics, INFORMS Journal on Computing (1996).

[10] Crainic, T. G., Toulouse, M., Gendreau, M., Toward a taxonomy of parallel tabu search Algorithms,.Research Report CRT-933, Centre de Recherche sur les Transports , Universite de Montreal (1993).

[11] Martins, L. S., Paralelização de Metaheurísticas em ambientes de Memória Distribuída. Tese de Doutorado (1999), DI – PUC/RIO.