Resolução do Problema de Roteamento de Veículos com Frota Heterogênea via GRASP e Busca Tabu.

Camila Leles de Rezende, Denis P. Pinheiro, Rodrigo G. Ribeiro

camilalelesproj@yahoo.com.br, denisppinheiro@yahoo.com.br, rodrigogribeiro@yahoo.com.br

Cic270 – Inteligência Computacional para Otimização

Prof. Dr. Marcone Jamilson Freitas Souza

DECOM - Departamento de Computação, ICEB – Instituto de Ciências Exatas e Biológicas

UFOP – Universidade Federal de Ouro Preto

Resumo: *Neste trabalho apresenta-se um método heurístico, baseado em GRASP e Busca Tabu, para resolver o Problema de Roteamento de Veículos. Uma solução inicial é gerada pela fase de construção GRASP seguida de uma busca local, obtendo-se a solução final aplicando-se uma Busca Tabu na solução resultante da fase GRASP. Resultados computacionais são apresentados para um conjunto de problemas-teste encontrados na literatura. O método proposto é de fácil entendimento e implementação, requer a manipulação de poucos parâmetros, produz soluções de boa qualidade rapidamente e é capaz de melhorar essas soluções quando lhe é dado um tempo de processamento mais elevado.*

Palavras-chave: Roteamento de Veículos, GRASP, Busca Tabu, Grafos, Metaheurísticas.

1 - Introdução

O Problema de Roteamento de Veículos (PRV) pode ser definido como segue. Dado um conjunto de cidades (ou consumidores), cada qual com uma demanda q_i por um produto, e um depósito com veículos de capacidade Q, encontrar as rotas para os veículos minimizando os custos de transporte.

Uma grande quantidade de aplicações práticas do PRV pode ser encontrada na literatura. Por exemplo, Brown & Graves (1981), Fisher et al. (1982), Bell et al. (1983), Evans & Norback (1985), Golden & Watts (1987) mostram aplicações nas indústrias de petróleo, químicas, alimentícias e de bebidas.

O interesse no PRV é parcialmente devido à sua importância prática, mas também à sua dificuldade. Como uma generalização do Problema do Caixeiro Viajante (PCV), o PRV pertence à classe de problemas NP-Difícil (LENSTRA, 1981), portanto não existem algoritmos em tempo polinomial para encontrar soluções ótimas. Os algoritmos exatos existentes raramente conseguem resolver problemas envolvendo mais do que 50 consumidores (RENAUD & BOCTOR, 2002).

Devido ao limitado sucesso dos métodos exatos, os esforços de pesquisa têm sido direcionados no desenvolvimento de heurísticas para lidar com problemas de maior porte. Exemplos de heurísticas bem sucedidas para resolver PRV são os algoritmos baseados em Busca Tabu de Taillard (1993), Osman (1993) e Gendreau et al. (1994), e a heurística de pétalas de Renaud et al.(1996).

Para uma revisão das mais importantes heurísticas clássicas e modernas para o PRV veja Cordeau et al. (2002) e Laporte (1992). Uma bibliografia do PRV pode ser obtida em Laporte & Osman (1995).

Neste trabalho propomos um método de duas fases para a resolução do PRV, que combina a fase de construção GRASP, com um refinamento usando Busca Tabu, para varrer um espaço de soluções afim de melhorar a solução obtida durante a fase de construção GRASP. Esta estratégia é baseada em uma função de avaliação que procura minimizar as distâncias percorridas e o número de veículos utilizados, ou seja, diminuir os custos das viagens.

O método proposto foi testado usando-se um conjunto de instâncias clássicas encontradas na literatura. Resultados computacionais demonstram a eficiência do método na obtenção de soluções finais de qualidade próxima aos melhores valores encontrados na literatura.

2 - Definição Formal do Problema do Roteamento de Veículos com Frota Heterogênea

Seja $G = (V, E)$ um grafo não direcionado, onde $V = \{v_0, v_1, ..., v_n\}$ é o conjunto dos vértices e $E = \{(v_i, v_j): v_i, v_j \in V, i < j\}$ é o conjunto de arestas. O vértice v_0 representa o depósito, sendo este a base de uma frota de veículos idênticos de capacidade Q, enquanto os vértices remanescentes correspondem às cidades ou consumidores. Cada consumidor v_i tem uma demanda não negativa q_i e $q_0 = 0$. Neste trabalho supõe-se que existe um número ilimitado de veículos no depósito.

A cada aresta (v_i, v_j) está associada uma distância não negativa c_{ij} que representa a distância entre os consumidores.

O Problema de Roteamento de Veículos consiste em determinar o conjunto de rotas que deverão ser feitas pelos veículos minimizando os custos de transporte, dado pela distância e respeitando as seguintes condições:

(a) Cada rota começa e termina no depósito;

(b) Toda cidade de $V\backslash\{v_0\}$ é visitada somente uma vez por somente um veículo;

(c) A demanda total de qualquer rota não deve superar a capacidade Q de um veículo.

3 - Estratégias de Solução

A seguir apresentamos algumas definições dos métodos utilizados para chegarmos a uma solução deste problema.

3.1 - Metaheurísticas

As metaheurísticas são métodos de busca local destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada a uma função de avaliação a qual tem que ser modelada para cada problema específico .Contrariamente às heurísticas convencionais, as metaheurísticas são de caráter geral têm condições de escapar de ótimos locais.

As metaheurísticas, assim como os métodos de busca local tradicionais, diferenciam-se entre si basicamente pelas seguintes características:

a) critério de escolha de uma solução inicial;

b) definição da vizinhança $N(s)$ de uma solução s;

c) critério de seleção de uma solução vizinha dentro de $N(s)$;

d) critério de término;

3.2 - GRASP (Greedy Randomized Adaptative Search Procedure)

GRASP (Procedimento de Busca Adaptativa Gulosa e Randômica) é um método iterativo, proposto por Feo e Resende, que consiste de duas fases: uma fase de construção, onde uma solução é gerada, e uma fase de busca local, na qual um ótimo local da vizinhança da solução construída é explorado. A melhor solução encontrada em

todas as iterações, é retornada como resultado do procedimento. A figura abaixo mostra o pseudo-código do procedimento GRASP:

```
procedimento GRASP(f(.); g(.);N(.);GRASPmax; s)
1 f* ← infinito ;
2 para (Iter = 1; 2; ... ; GRASPmax) faça
3       Construcao(g(.); alfa; s)*;
4       BuscaTabu(f(.);N(.); s)*;
5       se (f(s) < f*) então
6           s* ← s;
7           f* ← f(s);
8       fim-se;
9 fim-para;
10 s ← s*;
11 Retorne s;
fim GRASP
```

Figura 1: Algoritmo GRASP

** Parâmetros globais*

Na fase de construção, uma solução é iterativamente construída, elemento por elemento. A cada iteração dessa fase os elementos candidatos a serem incluídos na solução são colocados em uma lista C de candidatos, seguindo o seguinte critério de ordenação: distância da última cidade inserida na solução (ou depósito) para a cidade candidata a ser inserida. Esse processo de seleção é baseado em uma função adaptativa gulosa $g : C \to \Re$, que estima o benefício de seleção de cada um dos elementos. Essa heurística é adaptativa porque os benefícios associados com a escolha de cada elementos são atualizados de modo a refletir a escolha do elemento anterior. O componente aleatório do método reside em selecionar um candidato aleatoriamente a partir de um número restrito da lista de candidatos. Esse conjunto restrito de candidatos recebe o nome de LCR (Lista de Candidatos Restrita). Essa técnica permite uma maior variabilidade das soluções geradas por cada iteração GRASP. Seja $\alpha \in [0,1]$ um dado

parâmetro para o procedimento de construção. O parâmetro $\alpha \in [0,1]$ controla o nível de gulosidade/aleatoriedade da construção. Pelo pseudocódigo do método de construção GRASP, mostrado a seguir, notamos que quando α se aproxima de zero, a solução tende a ser mais gulosa (o valor zero leva a soluções puramente gulosas) e quando esse parâmetro se aproxima de um, a solução tende a ser aleatória (o valor um leva a soluções puramente aleatórias).

Procedimento Construção(g(.) , α , s)
S = 0
Inicialize o conjunto C de candidatos
Enquanto (C $\neq 0$)
Tmin = {Min g(t)/ $t \in c$ }
Tmax = {Max g(t)/ $t \in c$ }
LCR = $\{ t \in C / g(t) \leq T\min + \alpha(T\max - T\min) \}$
T = Selecione aleatoriamente um elemento de LCR
$s = s \cup \{t\}$
Atualize o conjunto C de canditatos
Fim-enquanto
Retorne s;
fim-construção

Figura 2 : Construção GRASP

3.3 - Busca Tabu

A Busca Tabu é um procedimento adaptativo que utiliza uma lista para guiar um método de descida a continuar a exploração do espaço de soluções mesmo na ausência de movimentos de melhora, evitando assim, a formação de ciclos (retorno numa solução previamente visitada).

Mais especificamente a Busca Tabu começa a partir de uma solução s, e explora um subconjunto V da vizinhança N(s) desta solução. O elemento s' com o menor valor de todos os elementos pertencentes a N(s), passa a ser a nova solução corrente, mesmo que esta piore a solução atual. Esse critério faz com que o algoritmo não "caia em ótimos locais" . Porém essa estratégia pode fazer o algoritmo "ciclar", ou seja, voltar a um ótimo local previamente visitado. Para evitar que isso aconteça existe uma lista tabu, que contém até n últimas soluções (movimentos de retorno a solução anterior) geradas. Quando o tamanho desta lista chega a n, o primeiro elemento da lista, é removido, para inserir-se o novo. Isso irá garantir que por n iterações o algoritmo não irá ciclar.

A lista reduz sensivelmente a possibilidade de ciclagem, porém, ela pode proibir a visita a soluções não explorada. Para evitar esse problema, existe uma função de aspiração, que dentro de certas circunstâncias aceita um vizinho tabu (vizinho que está na lista tabu). A função de aspiração A, é tal que para cada s' ela retorna um valor A(v), onde v é o valor atual da função objetivo, e, A(v) é o valor que algoritmo espera alcançar aplicando-se A, a solução s'. Em nossa implementação de busca tabu, adotamos como função de aspiração : **A (f(s*)) = f(s*) – 1**. Ou seja, só aceita-se esse vizinho se ele conduzir a uma melhora na solução.

Dois critérios são utilizados para interromper o processo: Quando f(s*) chega a um determinado valor limite, ou o algoritmo chega a um número máximo de iterações sem melhora na solução. Em nossa abordagem adotamos o segundo critério. Abaixo segue o pseudocódigo da busca tabu.

```
procedimento BT(f(.);N(.);A(.); |V|; fmin; |T|;BTmax; s)
1  s* ← s;                {Melhor solução obtida até então}
2  Iter ← 0;              {Contador do número de iterações}
3  MelhorIter ← 0;        {Iteração mais recente que forneceu s?}
4  T ← 0 ;;               {Lista Tabu}
5  Inicialize a função de aspiração A;
6  enquanto (f(s) > fmin e Iter -MelhorIter < BTmax) faça
7      Iter ← Iter + 1;
8      Seja s´ ← s © m o melhor elemento de V contido N(s) tal que
           o movimento m não seja tabu (m não pertence T) ou
           s´ atenda a condição de aspiração (f(s´) < A(f(s)));
9      T ← T - {movimento mais antigo} + {movimento que gerou s´};
10     Atualize a função de aspiração A;
11     s ← s´;
12     se (f(s) < f(s*)) então
13         s* ← s;
14         MelhorIter ← Iter;
15     fim-se;
16 fim-enquanto;
17 s ← s*;
18 Retorne s;
fim BT;
```

4 – Resultados Obtidos

Veículo	Capacidade	Custo
A	20	20
B	30	35
C	40	50
D	70	120
E	120	225

4.1 – Problema 1. 20 primeiras cidades do arquivo vrp8.txt.

	Literatura		**Resultados obtidos**	
	Fo	**Veículos**	**Fo**	**Veículos**
Melhor conhecido	**965**	**AB^2CE^2**	**929**	**CE^3**

- **Instância 2:**

Veículo	Capacidade	Custo
A	60	1000
B	80	1500
C	150	3000

Resultados:

	Literatura		Resultados Obtidos	
	Fo	Veículos	Fo	Veículos
Melhor Resultado	6446	A^6	7953	BC^2

4.2 – Problema 2. 50 cidades do arquivo vrp8.txt.

Veículo	Capacidade	Custo
A	50	100
B	100	250
C	160	450

	Literatura		Resultados Obtidos	
	Fo	Veículos	Fo	Veículos
Melhor resultado	2640	A^6B^5	3125	ABC^4

Veículo	Capacidade	Custo
A	60	1000
B	80	1500
C	150	3000

Veículos utilizados no problema 2.

4.2 – Problema 2. 20 primeiras cidades do arquivo vrp8.txt

	Literatura	**Resultados obtidos**
Melhor conhecido	$\mathbf{A^6}$	$\mathbf{A^5B}$
Melhor obtido	$\mathbf{A^6}$	$\mathbf{A^8}$

6 – Conclusão

O problema de Roteamento de Veículos além de possuir grande aplicabilidade no mundo real possui uma grande complexidade para sua resolução computacional. O clássico Problema do Caixeiro Viajante(PCV), pode ser resolvido como uma instancia do PRV, fazendo uma transformação polinomial adequada nos dados. O PCV pode ser considerado um subproblema do PRV, para isso, basta considerarmos o depósito contendo apenas um caminhão e este possuindo capacidade infinita e custo de percurso deste caminhão igual a um. A partir desta afirmação concluímos que o nosso algoritmo que resolve o PRV pode ser utilizado para resolver o PCV, bastando apenas aplicar uma transformação adequada na solução do PRV para o PCV.

7 – Blibliografia

- GOLDEN, B et al. The Fleet Size and Mix Vehicle Routing Problem. Computers and Operations Reserach, 11 : 49-66, 1984.
- BEASLEY, J. Operations Research Library; In http://www.ms.ic.ac.uk. Acesso em 01/11/2003.
- Notas de aula de Inteligência Computacional. Prof° Marcone Jamilson Freitas Souza. In http://www.decom.ufop.br/prof/marcone. Acesso em 20/11/2003.
- Notas de aula de Algortmos e Estruturas de Dados III. Profª. Lucília Camarão e Prof° Elton Silva.
- Trabalho de Projeto Orientado. Problema de Roteamento de Veículos com Frota Homogênea (*Vehicle Routing Problem*). Dárlinton B. F. Carvalho.