Comunicado Técnico 52
Dezembro, 2003
Campinas, SP

ISSN 1677-8464

Novos Algoritmos para Resolução do Problema de Roteamento

Marcelo Gonçalves Narciso[1]
Luiz Antônio Nogueira Lorena[2]

O projeto Análise de Redes com Sistemas de Informações Geográficas - ARSIG (Arsig, 2004) tem como uma de suas metas a espacialização de soluções de roteamento de veículos em uma dada área (rural ou urbana). Foram feitos diversos algoritmos para este tema, bem como a integração destes com os Sistemas de Informação Geográfica (SIG), através da linguagem Avenue (Lorena et al., 2001) para o SIG ArcView (ArcView, 2004) e da inserção dos algoritmos no código do Spring (Inpe, 2004).

Roteamento de veículos é um problema clássico de Otimização Combinatória e suas aplicações são inúmeras. Como exemplos podem ser citados: roteamento de ônibus urbano, transporte de grãos do produtor aos centros consumidores, portos, silos, transporte de gado, etc. O objetivo é transportar uma determinada carga de um local qualquer para outros pontos de demanda, de tal forma que o trajeto seja mínimo, minimizando assim custos e tempo de entrega, e voltar ao ponto de origem. No domínio agropecuário, problemas de transporte tais como a distribuição de grãos para silos e portos, transporte de produtos agrícolas do produtor para o consumidor, etc. são muito comuns. Sem um planejamento adequado, o transporte poderá encarecer o produto para o consumidor final. Se o trajeto for minimizado, haverá redução de custos, e isto irá se refletir no preço dos produtos para o consumidor.

Um possível modelo a ser considerado em transporte de produtos é o problema do caixeiro viajante (PCV). Suponha que um caixeiro (que pode representar um ou mais caminhões de entrega de produtos), a partir de sua cidade de origem, precise visitar um conjunto de n - 1 cidades (que representa postos distribuidores de produtos, silos, etc.). Cada cidade deverá ser visitada somente uma vez. Após visitar todas as n - 1 cidades, então o caixeiro deverá voltar a sua cidade inicial. O caixeiro poderá escolher a primeira cidade a ser visitada, a segunda, e assim por diante. Entretanto, caso o caixeiro queira economizar tempo e energia (ou ainda gasolina, recursos, etc.), ele deverá procurar o menor caminho de tal forma que todas as cidades sejam visitadas somente uma vez e então retornar para a sua cidade de origem. Se o caixeiro sabe a distância entre as cidades, então ele pode encontrar o caminho mais curto para a sua viagem simplesmente por comparar todas as maneiras possíveis de se visitar as n - 1 cidades e voltar. Entretanto, admitindo-se que exista sempre um caminho ligando uma cidade a outra, à medida que o número de cidades vai crescendo, o número de maneiras de se visitar todas as cidades, que é dado por (n 1)!, cresce também.

Embora seja difícil encontrar soluções ótimas para o *PCV*, quando o número de cidades é muito grande, com o uso de heurísticas (métodos não exatos) é possível encontrar soluções viáveis que sejam muito próximas do ótimo sem que seja necessário enumerar todas as soluções possíveis.

[1] *Dr. em Computação Aplicada, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: narciso@cnptia.embrapa.br)*
[2] *Ph.D. em Pesquisa Operacional, Pesquisador Associado do Laboratório de Computação e Matemática Aplicada do Instituto Nacional de Pesquisas Espaciais (INPE), Caixa Postal 515 - 12245-970 - São José dos Campos, SP. (e-mail: lorena@lac.inpe.br)*

Embrapa

Para exemplificar como é difícil enumerar todas as soluções possíveis para depois escolher a melhor, suponha um computador muito veloz, capaz de fazer 1 bilhão de adições por segundo. Isso parece uma velocidade imensa. Com efeito, no caso de 20 cidades, o computador precisa apenas de 19 adições para dizer qual o comprimento de uma rota e então será capaz de calcular $10^9/19 = 53$ milhões de rotas por segundo. Contudo, essa imensa velocidade representa quase nada frente à imensidão do número 19! de rotas que precisará examinar. Com efeito, o valor de 19! é 121 645 100 408 832 000 (ou , aproximadamente, 1.2×10^{17} em notação científica). Conseqüentemente, ele precisará de

$$1.2 \times 10^{17} / (53 \text{ milhões }) = 2.3 \times 10^9 \text{ segundos}$$

para completar sua tarefa, o que equivale a cerca de 73 anos. O problema é que a quantidade (n - 1)! cresce muito rapidamente, sendo que a partir de um certo número n de cidades o computador torna-se incapaz de executar o que for exigido em tempo hábil. A Tabela 1 ilustra o que foi mencionado.

Tabela 1. Tempo para calcular a melhor rota em função de n cidades.

n	*Rotas por segundo*	*(n - 1)!*	*Tempo para o cálculo total*
5	250 milhões	24	insignificante
10	110 milhões	362 880	0.003 s
15	71 milhões	87 bilhões	20 min.
20	53 milhões	1.2×10^{17}	73 anos
25	42 milhões	6.2×10^{23}	470 milhões de anos

Na Tabela 1 conclui-se que um método para enumerar todas as soluções e então computar a melhor solução não é viável. Uma proposta para a resolução deste problema é encontrar uma heurística que forneça em geral uma solução de bom nível (muito perto do ótimo) e então tentar melhorá-la ou aceitar a solução obtida. Para saber se a solução é ou não de bom nível, existem métodos para calcular limites (inferior e superior) para uma solução ótima. De uma maneira geral, se os limites são muito próximos (menos do que 1% de diferença) e o valor da solução está dentro destes limites, então esta solução é considerada, no mínimo, quase ótima. Os limites são soluções aproximadas para PCV, podendo ser viáveis (atende a todas as restrições) ou não.

Uma das formas de se calcular o limite inferior é dado pelo método de geração de colunas (MGC) a ser mostrado neste trabalho. Este método fornece um limite inferior para a solução ótima. O limite inferior é uma solução que não atende a todas as restrições do PCV, daí o seu valor final ser menor do que o ótimo. Este método não é novo, mas foi feita uma adaptação que melhora seu desempenho, sendo esta a justificativa deste trabalho

O limite superior será fornecido por uma heurística, cuja solução atende a todas as restrições do problema, mas não necessariamente é uma solução ótima, porém é viável (factível).

A resolução do PCV pode ser espacializada, tornando-se assim fácil para o usuário verificar a solução. Porém, faz-se necessário uma integração de um Sistema de Informação Geográfica (SIG) com o algoritmo para solução de um PCV. Esta integração foi feita no projeto ARSIG (Arsig, 2004). Porém, o mais difícil é o desenvolvimento de algoritmos eficientes para a solução de um PCV, tal que o algoritmo seja executado em um curto espaço de tempo e a solução tenha um bom nível (próximo à solução ótima).

Feitas estas considerações, este trabalho mostra uma nova heurística de boa qualidade para resolver o PCV e também uma heurística para se obter limites inferiores e, conseqüentemente, medir quão perto a solução viável obtida está perto do ótimo, no pior caso.

Formulação do Problema do Caixeiro Viajante

Seja um grafo não direcionado $G = (V,E)$ com n vértices, onde $V = \{1,2,\ldots,n\}$ e o conjunto de arcos $E = \{(i,j)$ tal que $i,j \in V$ e seja c_{ij} o custo não negativo associado ao arco i-j. O problema do caixeiro viajante pode ser matematicamente definido como

$$v(PCV) = min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

$$\text{sujeito a } \sum_{j \in V} x_{ij} = 1 \quad i \in V \qquad (1)$$

(PCV)

$$\sum_{i \in V} x_{ij} = 1 \quad j \in V \qquad (2)$$

$$\sum_{i \in V} \sum_{j \in V} x_{ij} \leq |S| - 1, \quad \forall S \subset V, \; S \neq \emptyset \qquad (3)$$

$$x_{ij} = 0 \text{ ou } 1, i,j \in V, \qquad (4)$$

onde $x_{ij} = 1$ se o caminho i-j pertence a solução e $x_{ij} = 0$, se o caminho i-j não pertence a solução.

A função objetivo do problema v*(PCV)* indica a minimização das distâncias ao se visitar todas as cidades e voltar a cidade de origem. A restrição (1) indica que, a partir da cidade i, deve-se ir a apenas uma cidade j, $j \in V-\{i\}$. Analogamente, a restrição (2) indica que, a partir de uma cidade j, deve-se ir apenas a uma cidade i, $i \in V - \{j\}$. A restrição (3) serve para eliminar ciclos dentro de todo o trajeto. Desta forma, se existe um conjunto S com k cidades ($k < n$), estas k cidades não serão interligadas de forma a existir um ciclo entre elas. O único ciclo que pode existir é o trajeto começando de uma cidade inicial, percorrendo as outras n-1 cidades e voltando à cidade inicial. Esta restrição também pode ser escrita como

$$\sum_{i \in S} \sum_{j \in V-S} x_{ij} \geq 1, \; \forall S \subset V, \; S \neq \emptyset \qquad (5)$$

A restrição (4) indica que, para $xij = 1$, o arco que liga i a j é escolhido para fazer parte da solução. Se o valor for 0, então o arco que liga i a j não é escolhido para o trajeto.

O *(PCV)* é *simétrico* quando $c_{ij} = c_{ji}$, para todo $i,j \in V$. Sendo assim, o *(PCV)* simétrico pode ser definido como um grafo não direcionado completo $G = (V,E)$ com n vértices, $V = \{1,2,...,n\}$, onde E é o conjunto de todos os arcos conhecidos de i a j ou j a i, e $c_{ij} = c_{ji}$. Desta forma, o *(PCV)* simétrico, doravante chamado de *(PCVS)*, pode ser definido como

$$v(PCV) = min \sum_{i \in V} \sum_{j > V} c_{ij} x_{ij}$$

$$\text{sujeito a} \sum_{j<i} x_{ij} + \sum_{j<i} x_{ij} = 2 \quad i \in V \qquad (6)$$

(PCV)

$$\sum_{i \in V} \sum_{j \in S,\ \forall S \subset V,\ S \neq \varnothing} x_{ij} \leq |S| - 1, \qquad (7)$$

$$x_{ij} = 0 \text{ ou } 1, \quad i,j \in V,\ j > i \qquad (8)$$

A expressão (7) pode ser também escrita como

$$\sum_{i \in V} \sum_{\substack{j \in S \\ j > i}} x_{ij} + \sum_{i \in V} \sum_{\substack{j \in S \\ j > i}} x_{ij} \geq 2, \ \forall S \subset V,\ S \neq \varnothing \qquad (9)$$

Mais detalhes sobre o PCV, veja Lawler et al. (1997).

Método de Geração de Colunas - um Novo Enfoque

Neste trabalho está-se examinando uma abordagem combinada da relaxação Lagrangeana/surrogate (Narciso & Lorena, 2002) e o método tradicional de geração de colunas (Wolsey & Nemhauser, 2001). A técnica de geração de colunas pode ser aplicada a problemas lineares de grandes dimensões, no caso de não se dispor de todas as colunas "a priori", ou quando se pretende resolver um problema utilizando a decomposição de Dantzig-Wolfe (Dantzig & Wolfe, 1960), onde as colunas correspondem aos pontos extremos do conjunto convexo de soluções factíveis do problema. Neste caso, o algoritmo para resolução alterna entre um subproblema e um problema mestre restrito. A partir de um conjunto inicial de colunas, resolve-se o problema mestre, obtendo-se as variáveis duais que serão utilizadas pelo subproblema para determinar novas colunas a serem consideradas no problema mestre. A relaxação Lagrangeana/surrogate é usada para gerar novas colunas e resulta em uma aceleração e estabilização do método de geração de colunas. São apresentados resultados computacionais para instâncias da (TSPLIB, 2004) comparando os métodos de geração de colunas (tradicional e o combinado com a relaxação Lagrangeana/surrogate).

O algoritmo de geração de colunas, após as modificações baseadas no algoritmo da relaxação lagsur (Narciso & Lorena, 2002), é o seguinte:

Algoritmo de Geração de Colunas em função do multiplicador t (AGC (t))

AGC (t)

(i) Escolha um conjunto inicial de colunas (com função objetivo e restrições), o qual formará o problema inicial *(SP-P)*.

Cada coluna é gerada através de uma árvore 1-tree que conecta todos os nós (cidades). Exemplos podem ser encontrados em (Wolsey & Nemhauser, 2001).

(ii) Resolva *(SP-P)* e obtenha a solução dual do problema p_j, $j = 1,...,n$;

(iii) Resolva a relaxação Lagrangean/surrogate usando o multiplicador dual obtido no passo (ii), isto é, $\underset{t \geq 0}{Max}\ v(L_t(p))$. Obtenha o melhor valor de t. Veja Narciso & Lorena (2002) para maiores detalhes.

(iv) Com o resultado de $\underset{t \geq 0}{Max}\ v(L_t(p))$ que é uma árvore mínima (1-tree), adicione este ao problema (SP-P) desde que

$$\sum_{i<j} c_{ij} x_{ij} - \sum_{k \in v} l\,k.t \left(\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} \right) < 0, \quad i = 1,...,n$$

(v) Se a árvore 1-tree é um tour, ou o limite máximo de iterações é encontrado, ou o problema convergiu (resultados da relaxação lagsur não mudam após 30 iterações) então pare.

(vi) Realize testes para remover colunas e retorne ao passo (ii). As colunas a serem removidas são aquelas nas quais seu valor correspondente na função objetivo é maior que a média dos valores das demais colunas. Retirar até 5 colunas por iteração.

A diferença do AGC tradicional e o proposto (AGC(t)) é o fato de existir o multiplicador t (passo 3 e 4 do algoritmo), o qual é calculado conforme a relaxação lagrangeana/surrogate (Narciso & Lorena, 2002). Isto faz com que o algoritmo fique muito mais rápido, em termos de convergência. Esta foi a contribuição deste trabalho. Foram feitos testes com instâncias disponíveis no *site* (TSPLIB, 2004). OS algoritmos foram codificados para C++ e executados em uma Sun Ultra 60. Os resultados obtidos, em termos de número de iterações podem ser observadas na Tabela 2.

Observe que os resultados são praticamente os mesmos, porém o número de iterações com o AGC(t) diminuiu significativamente (conseqüentemente, o tempo de execução diminui).

Tabela 2. Algoritmo com o conceito de Geração de Colunas Tradicional e com a relaxação lagrangeana/surrogate.

Número de cidades	Solução ótima do problema	Número de iterações do AGC	Valor obtido pelo AGC	Número de iterações do AGC (t)	Valor obtido pelo AGC (t)
16	6.859	231,2	6.852,14	158,4	6.852,14
48	10.628	1.144,8	10.596,12	312,6	10.596,12
52	7.526	225,6	7.503,42	185,4	7.503,42
100	21.282	571,2	20.856,36	223,8	20.856,36
225	3.919	760,0	3.762,24	529,2	3.762,24
442	50.778	2.400,0	50.270,22	303,6	50.270,22
1291	50.801	2.400,0	49.276,97	370,8	49.276,97
1304	252.948	2.400,0	240.300,60	634,2	247.889,00
1379	56.638	2.388,0	55.505,24	577,2	55.505,24
1665	62.128	2.400,0	60.264,16	1.800,0	60.885,44
1748	336.556	2.400,0	319.728,20	1.800,0	329.824,90
1889	316.536	2.400,0	300.709,20	1.800,0	310.205,30
2152	64.253	2.400,0	62.967,94	1.800,0	63.610,47
2392	378.032	3.000,0	362.910,70	2.000,0	370.471,40

Algoritmo para Calcular uma Solução Viável para o PCV

A heurística AGC(t) pode ser aproveitada para calcular, a cada iteração, uma solução viável em função do multiplicador obtido no passo (ii). Neste enfoque, a função objetivo do PCV seria

$$V(VCP) = min \sum_{i \in V} \sum_{j \in V} (c_{ij} - p_i + p_j) * x_{ij} \qquad (10)$$

Assim, com a variação da função objetivo em função de p, tem-se uma solução nova x_{ij} a cada passo do algoritmo pois p varia. No AGC(t), entre os passos (ii) e (iii), pode ser inserida uma heurística qualquer que resolva o PCV tal que a função objetivo seja função de p, conforme descrito na equação 10. Em cada iteração do AGC(t), seria gerada uma nova solução x_{ij}. A melhor solução (menor valor da função objetivo) obtida após todas as iterações do AGC(t) seria então o resultado final. Qualquer heurística pode ser inserida neste contexto. Certamente, irá produzir um resultado melhor do que a heurística original. Para exemplificar, foram feitos testes com uma heurística fácil de se implementar, chamada de "gulosa", cujo algoritmo está descrito a seguir.

1. *Partindo-se do nó (ou cidade) i = 1, verificar qual é o nó mais próximo de i = 1. Seja j este novo nó. Adicione a distância ij à solução e faça i = j (atualizar i).*

2. *Partindo-se de i, verificar qual é o nó j mais próximo. Este novo nó j não pode ser um nó escolhido anteriormente. Adicione esta nova distância ij à solução e faça i = j.*

3. *Repetir o passo 2 até que não sobre nó algum. Quando não sobrar mais nenhum nó, adicione a distância de j (último nó escolhido) ate o nó inicial (nó 1) à solução.*

A proposta de melhoria deste algoritmo, aproveitando o AGC(t), seria:

1. *Atualizar a função objetivo, após o passo (ii) da heurística AGC(t), tal como a equação (10).*

2. *Partindo-se do nó (ou cidade) i = 1, verificar qual é o nó mais próximo de i = 1. Seja j este novo nó. Adicione a distância ij à solução e faça i = j (atualizar i).*

3. *Partindo-se de i, verificar qual é o nó j mais próximo. Este novo nó não pode ser um nó escolhido anteriormente. Adicione esta nova distância ij à solução e faça i = j.*

4. *Repetir o passo 3 até que não sobre nó algum. Quando não sobrar mais nenhum nó, adicione a distância de j (último nó escolhido) até o nó inicial (nó 1) à solução.*

Seja gulosa2 o algoritmo acima.

A Tabela 3 a seguir mostra os resultados com a heurística gulosa com a função objetivo normal (sem o multiplicador) e com a função objetivo sendo função do multiplicador.

Observe que a diferença relativa, em relação à solução ótima de cada instância foi menor para a heurística que estava em função do multiplicador p. Uma explicação para este fato é que os valores de p vão sendo alterados durante a execução de AGC(t) para obter um valor da relaxação mais perto do ótimo. Isto vai influenciar também na heurística gulosa2, visto que a função objetivo também é função de p.

A heurística gulosa não obtém, via de regra, bons resultados à medida que o número de cidades aumenta. Porém, os resultados obtidos são aceitáveis. Outras heurísticas melhores que a gulosa podem ser inseridas e os resultados obtidos serão melhores. Segundo Cunha et al. (2003) existe um apanhado de heurísticas que podem ser usadas no contexto do multiplicador p e terem um resultado ainda melhor do que sua versão original.

Tabela 3. Heurística gulosa com e sem multiplicador

Número de cidades	Valor da Heurística gulosa	Valor da Heurística gulosa2	Solução ótima do problema	% relativa da Heurística gulosa	% relativa da Heurística gulosa2
16	9.988	6.859	6.859	45,62	0,00
48	12.861	10.653	10.628	21,01	0,24
52	8.962	7.526	7.526	19,08	0,00
100	27.772	22.334	21.282	30,49	4,94
225	4.648	4.192	3.919	18,60	6,97
442	61.925	53.911	50.778	21,95	6,17
1291	60.042	54.921	50.801	18,19	8,11
1304	339.549	273.200	252.948	34,24	8,01
1379	69.106	62.432	56.638	22,01	10,23
1400	28.058	22.571	20.127	39,41	12,14
1665	73.666	68.253	62.128	18,57	9,86
1748	410.344	378.944	336.556	21,92	12,59
1889	388.986	343.701	316.536	22,89	8,58
2152	79.034	72.027	64.253	23,00	12,09
2392	460.671	429.668	378.032	21,86	13,66
3038	172.791	155.066	137.694	25,49	12,62

Assim, o enfoque desta proposta de solução viável é aproveitar qualquer heurística conhecida e inseri-la no algoritmo do AGC(t) para melhorar sua solução, conforme exemplificado com a heurística gulosa.

Considerações Finais

A respeito de heurística para se obter solução viável, existem várias propostas na literatura. Porém, nenhuma pode garantir a solução ótima. O método geral sugerido neste trabalho é para melhorar o desempenho de uma heurística qualquer conhecida da literatura. Quanto melhor for a heurística usada, melhores serão os resultados obtidos. O custo para isto é um tempo maior para resolução, que é compensado pela qualidade da solução fornecida.

A respeito do cálculo do limite inferior, o mérito está em diminuir o tempo de estabilização (número de iterações diminui) usando a relaxação lagrangeana/surrogate, voltada para o PCV ou PCVS. O tempo consumido pela resolução da relaxação lagrangeana/surrogate (lagsur), neste enfoque, é muito pequeno comparado à economia de iterações que é obtida. Assim é vantajoso usar a relaxação lagsur para diminuir o numero de iterações do AGC tradicional.

Referências Bibliográficas

ARCVIEW. Disponível em: < http://www.esri.com/ > .Acesso em: 2 jan.2004.

ARSIG - Análise de Redes com Sistemas de Informações Geográficas. Disponível em: < http://www.lac.inpe.br/~lorena/ArsigIndex.html > . Acesso em: 2 jan. 2004.

CUNHA, C. B. da; BONASSER, U. de O.; ABRAHÃO, F.T. M. Experimentos computacionais com heurísticas de melhorias para o problema do caixeiro viajante. Trabalho apresentado ao 16º Congresso de Pesquisa e Ensino em Transportes, Natal, 2002. Disponível em: < www.ptr.usp.br/docentes/cbcunha/files/ 2-opt_TSP_Anpet_2002_CBC.pdf > . Acesso em: 12 out. 2003.

DANTZIG, G. B., WOLFE, P. Decomposition principle for linear programs. Operations Research, Baltimore, v. 8, p.101-111, 1960.

INPE. Spring. Disponível em: < http://www.dpi.inpe.br/spring > . Acesso em: 2 jan. 2004.

LAWLER, E. L.; LENSTRA, J. K.; RINNOOY KAN, A. H. G., SHMOYS, D. D. The traveling salesman problem: a guided tour of combinatorial optimization. New York: John Wiley, 1997. 476 p.

LORENA, L. A. N.; SENNE, E. L. F.; PAIVA, J. A. C.; PEREIRA, M. A. Integração de modelos de localização a sistemas de informações geográficas. Gestão & Produção, São Carlos, v. 8, n. 2, p.180-195, 2001.

NARCISO, M. G.; LORENA, L.A.N. Using logical surrogate information in Lagrangean relaxation: an application to symmetric traveling salesman problems. European Journal of Operational Research, Amsterdam, v. 138, p. 473-483, 2002.

TSPLIB. Disponível em: *ftp://ftp.zib.de/pub/Packages/mp-testdata/tsp/tsplib/tsp/*. Acesso em: 2 jan. 2004.

WOLSEY, A. L.; NEMHAUSER, G. L. Integer and combinatorial optimization. New York: John Wiley, 2001. 760 p.

Comunicado Técnico, 52

Ministério da Agricultura, Pecuária e Abastecimento

Governo Federal

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Endereço: Caixa Postal 6041 - Barão Geraldo
13083-970 - Campinas, SP
Fone: (19) 3789-5743
Fax: (19) 3289-9594
e-mail: sac@cnptia.embrapa.com.br

1ª edição on-line - 2003

Expediente

Supervisor editorial: *Ivanilde Dispato*
Normalização bibliográfica: *Maria Goretti Gurgel Praxedis*
Editoração eletrônica: *Área de Comunicação e Negócios*