Algoritmos Genéticos Aplicados ao Problema de Roteamento de Veículos

Milton Roberto Heinen[1] **e Fernando Santos Osório**[1]

[1]Universidade do Vale do Rio dos Sinos (UNISINOS)
Computação Aplicada – PIPCA
CEP 93022-000 – São Leopoldo - RS - Brasil

mheinen@turing.unisinos.br, fosorio@unisinos.br

Resumo. *Este artigo trata do problema de roteamento de veículos, que é uma tarefa muito importante na maioria dos sistemas de transporte e logística. Embora o problema de roteamento de veículos ainda não possua uma solução exata em tempo polinomial, existem algumas soluções em tempo polinomial que podem aproximar o valor ótimo. Neste artigo serão descritas três heurísticas de aproximação para este problema: a Heurística de Clark e Wright, a Heurística de Mole e Jameson e os Algoritmos Genéticos. Diversos experimentos foram realizados com cada uma das heurísticas, e foram realizas comparações relativas ao tempo de execução e a qualidade das soluções encontradas.*

1. Introdução

O roteamento de veículos é um problema presente na maioria das empresas de transporte, logística e distribuição. Seu objetivo é determinar, dentre todas as possíveis rotas alternativas, qual é a que representa o menor custo, ou seja, qual é a solução ótima [Goldbarg and Luna 2000]. Aparentemente, é fácil descobrir qual a melhor solução, basta calcular o custo de todas as possíveis combinações e selecionar a que apresentar o menor custo. Para um pequeno conjunto de locais a serem visitados (nodos), isto é perfeitamente viável, mas a medida que este conjunto cresce, a solução vai se tornando cada vez mais difícil do ponto de vista computacional. Isto ocorre porque o número de combinações possíveis se torna muito grande, fazendo com que o cálculo possa demorar até vários séculos dependendo do número de nodos.

Em termos de complexidade computacional, os Problemas de Roteamento de Veículos (PRV), assim como a maioria dos problemas combinatoriais, são classificados como sendo NP-Completos [Cormen et al. 2002], ou seja, a complexidade de tempo é não polinomial. Até o momento, não foi possível encontrar nenhuma solução de tempo polinomial para problemas da classe NP-Completo, assim sendo, atualmente não existe nenhuma solução exata para o problema de roteamento de veículos em tempo polinomial – $O(n^c)$ [Karp 1975]. Mas devido a sua grande importância estratégica, soluções aproximadas para o PRV vem sendo estudadas a várias décadas. O objetivo deste artigo é descrever três heurísticas de aproximação para o problema de roteamento de veículos: (i) a heurística de Clark e Wright [Clark and Wright 1964]; (ii) a heurística de Mole e Jameson [Mole and Jameson 1976]; (iii) os Algoritmos Genéticos [Goldberg 1989, Mitchell 1996], que são na verdade meta-heurísticas.

Em nossos trabalhos anteriores [Heinen 2005], um estudo comparativo foi realizado entre as heurísticas de Clark e Wright e a de Mole e Jameson. Neste artigo, será

realizada uma comparação dos resultados destas heurísticas com os resultados obtidos através do uso de Algoritmos Genéticos. O artigo está estruturado da seguinte forma: Inicialmente serão descritos os Algoritmos Genéticos e o PRV do ponto de vista teórico. Em seguida, serão descritas as heurísticas de aproximação selecionadas, abordando suas características e limitações. Após isto serão descritas as implementações realizadas, e os resultados obtidos com cada uma das heurísticas.

2. Algoritmos Genéticos

Os Algoritmos Genéticos (AG) são métodos de busca estocástica baseados na Teoria da Evolução Natural das Espécies [Darwin 1859], criados por John Holland nos anos 60 [Holland 1975]. Os Algoritmos Genéticos trabalham com uma população de soluções iniciais, chamadas cromossomos, que através de diversas operações vão sendo evoluídas até que se chegue a uma solução que melhor atenda a algum critério específico de avaliação [Mitchell 1996]. Para que isto ocorra, a cada geração os cromossomos são avaliados segundo uma função que mede o seu nível de aptidão, chamada de função de *fitness*. Os cromossomos que tiverem o melhor *fitness* são selecionados para darem origem a próxima geração, através de operações como cruzamentos e mutações. Desta forma, a tendência é que a cada geração o conjunto de soluções vá sendo melhorado, até que se chegue a uma solução que atenda aos objetivos desejados [Goldberg 1989].

Para a implementação dos Algoritmos Genéticos, foi selecionada a biblioteca de software GAlib[1], desenvolvida por Matthew Wall do *Massachusetts Institute of Technology* (MIT). A GAlib foi selecionada por ser uma das mais completas, eficientes conhecidas bibliotecas de software para a simulação de Algoritmos Genéticos, e também por ser uma solução gratuita e de código aberto, baseada na linguagem de programação C++

3. Roteamento de veículos

O Problema de Roteamento de Veículos (PRV) é um problema de grande importância estratégica e de difícil solução computacional, e por isso vem sendo estudado há várias décadas por pesquisadores de todo o mundo. Ele pode ser descrito da seguinte forma: dado um depósito de abastecimento e vários clientes ligados ao depósito e entre si através de vários caminhos com custos diferentes, descobrir a melhor rota a ser empreendida por um veículo, que saia do depósito, passe por todos os clientes e retorne com o menor custo possível [Fisher and Jaikumar 1981]. Uma forma muito comum de se representar o PRV é através de grafos ponderados [Cormen et al. 2002], onde os vértices representam o depósito e os clientes, e as arestas representam os caminhos ligando os diversos vértices entre si. Os pesos das arestas representam os custos de se percorrer os caminhos.

3.1. Heurísticas de aproximação

Pelo fato de o PRV não possuir uma solução exata em tempo polinomial, para que ele possa ser solucionado outras estratégias devem ser utilizadas, como por exemplo as heurísticas de aproximação. Estas heurísticas fornecem uma solução aproximada, que é suficiente na maioria dos casos. O foco deste artigo são as heurísticas construtivas, que partem de uma solução inicial que é atualizada a cada passo segundo algum critério de minimização do custo. Assim, a solução vai sendo construída iterativamente, até que se

[1]GAlib – http://www.lancet.mit.edu/ga/

chegue a uma solução aproximada. Pelo fato dos procedimentos de economia e inserção sempre utilizarem a solução de menor custo a cada passo, eles são chamados de algoritmos gulosos [Cormen et al. 2002].

4. Heurística de Clark e Wright

A heurística de Clark e Wright foi um dos primeiros algoritmos a serem propostos para solucionar o problema de roteamento de veículos. Ele foi muito importante em sua época, servindo de base para outros algoritmos mais sofisticados que surgiram mais tarde. O funcionamento do algoritmo ocorre da seguinte forma: Um grafo ponderado representando os clientes, os possíveis caminhos e o depósito é lido a partir de uma matriz de adjacências ou de alguma outra estrutura de armazenamento Em seguida são criadas $n - 1$ rotas iniciais, cada uma delas ligando um dos clientes ao depósito, onde n é o número de nodos do grafo. Após isto, são calculadas as economias para todas as possíveis combinações 2 por 2 de nodos, através da fórmula:

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \tag{1}$$

onde s_{ij} representa a economia obtida ao se unir as rotas que possuem os nodos i e j, e c_{ij} representa o custo de se deslocar do nodo i para o nodo j. As economias são colocadas em uma tabela, que é ordenada de forma decrescente.

Para cada linha da tabela de economias, começando pela maior economia, é verificado se é possível unir as rotas que contém os nodos i e j. As rotas podem ser combinadas se ambos os nodos estiverem em extremos opostos de rotas diferentes, e se a união não violar nenhuma das restrições do modelo (capacidade de carga dos veículos, limite de combustível, janela de tempo, etc). O processo continua com o próximo par de nodos da tabela de economias, até que se chegue ao último ítem da tabela ou até que não hajam mais rotas a serem combinadas.

A complexidade computacional do algoritmo Clark e Wright é $O(n^2)$, ou seja, ele é resolvido em tempo polinomial. Uma desvantagem deste algoritmo é que a combinação de rotas só é realizada quando os nodos estão nos extremos das rotas, o que faz com que os nodos internos sejam excluídos do teste de economia. A principal contribuição deste algoritmo pode ser considerada o fato de ele ter aberto caminho para algoritmos mais poderosos que surgiram após este, como por exemplo o de Mole e Jameson [Mole and Jameson 1976].

4.1. Heurística de Mole e Jameson

A heurística de Mole e Jameson [Mole and Jameson 1976] é uma heurística bem mais sofisticada que a de Clark e Wright. Entre outras coisas, ela tenta reduzir a principal fragilidade deste último, que é a dos nodos internos não serem candidatos ao teste de economia [Goldbarg and Luna 2000]. O funcionamento do algoritmo ocorre da seguinte forma: Inicialmente o grafo é lido a partir de uma matriz de adjacências, e uma lista de nodos livres (que estão fora de qualquer rota) é criada com todos os nodos do grafo menos o depósito (nodo 0). Em seguida, é selecionado um nodo para ser inserido na rota inicial, segundo algum critério. Na implementação deste algoritmo foi utilizado o critério de selecionar o nodo livre mais próximo do depósito para iniciar uma rota. Após isto, o algoritmo começa a realizar um laço, dentro do qual os nodos vão sendo sucessivamente

inseridos nas rotas. A inserção é realizada segundo dois critérios, o de proximidade e o de economia. O critério de proximidade seleciona o nodo que está mais próximo da rota atual, segundo as distâncias calculadas pela fórmula:

$$e(i,l,j) = c_{il} + c_{lj} - \mu c_{ij} \tag{2}$$

onde c_{il} representa o custo entre os nodos i e l, c_{lj} o custo entre l e j e c_{ij} o custo entre i e j. O nodo que apresentar a menor distância é selecionado para ser inserido na rota, caso isto não viole nenhuma restrição.

O critério de economia visa selecionar qual o melhor local para se inserir o nodo selecionado. Ele é calculado pela fórmula:

$$\sigma(i,l,j) = \lambda c_{0l} - e(i,l,j) \tag{3}$$

onde c_{0l} representa a distância entre o depósito e o nodo l. O local a ser selecionado é o que apresentar a maior economia. Os parâmetros λ e μ, presentes nas Fórmulas 2 e 3, permitem que se altere o comportamento do algoritmo de diversas formas, conforme consta em [Goldbarg and Luna 2000]. Na implementação realizada, foram utilizados os valores de $\lambda = 2$ e de $\mu = 1$. O laço se encerra quando não houverem mais nodos presentes na lista de nodos livres.

A heurística de Mole e Jameson, por ser um modelo mais aperfeiçoado, apresenta diversas vantagens em relação a heurística de Clark e Wright, entre as quais podemos destacar o fato de que o resultado obtido é em geral bem melhor que o obtido com a heurística Clark e Wright. A principal desvantagem do algoritmo é a sua maior complexidade computacional.

4.2. Roteamento genético

No algoritmo genético utilizado para o roteamento de veículos, cada indivíduo da população (genoma) é uma lista de inteiros, onde cada elemento desta lista corresponde a um elemento do grafo, ou seja, um cliente que deve ser visitado. A Tabela 1 mostra um exemplo de genoma para uma rota com 10 clientes. A primeira linha mostra os índices da lista (ordem em que os clientes serão visitados), e a segunda linha mostra o identificador do cliente de cada uma das posições da lista. Na rota da Tabela 1, o primeiro cliente a ser visitado é o 05, em seguida o cliente 10, e assim sucessivamente, até o cliente 03.

Tabela 1. Exemplo de um genoma

Ordem	0	1	2	3	4	5	6	7	8	9
Cliente	05	10	04	02	01	09	07	06	08	03

O tipo de Algoritmo Genético utilizado foi o *GASteadyStateGA* com populações sobrepostas (*overlapping populations*) proposto por [De Jong 1975]. A Tabela 2 descreve os parâmetros do Algoritmo Genético utilizados nas simulações. Uma descrição completa de cada um destes parâmetros pode ser encontrada na documentação da GAlib.

O Algoritmo Genético utilizado opera da seguinte forma: inicialmente, para cada indivíduo é inicializado com rotas aleatórias, mas que passam apenas uma vez em cada cliente (nodo do grafo). Uma verificação é realizada para evitar que dois nodos sem ligação

Tabela 2. Parâmetros do Algoritmo Genético

Parâmetro	Descrição	Valor
gaNpCrossover	Probabilidade de cruzamento (p_c)	1,00
gaNpMutation	Probabilidade de mutação (p_m)	0,30
gaNpopulationSize	Tamanho da população (s_p)	500
gaNnGenerations	Número de gerações (n_g)	1000
gaNpReplacement	Taxa de substituição (p_r)	0,90

entre si (arestas) sejam colocados em posições consecutivas. A cada geração, os indivíduos são avaliados através de uma função que mede a qualidade da solução representada por este indivíduo. Esta função, chamada de função de *fitness*, calcula o comprimento total da rota a ser percorrida, com base nas distâncias entre os elementos consecutivos da rota. Os indivíduos que se mostrarem mais aptos, ou seja, aqueles que tiverem o menor *fitness*, serão escolhidos para comporem a próxima geração, através das operações de cruzamento e mutação. O esquema de seleção utilizado foi o *stochastic remainder sampling selector* (*GASRSSelector*), que segundo [Goldberg 1989] possui um desempenho superior ao esquema tradicional da roleta (*GARouletteWheelSelector*).

O operador de mutação utilizado realiza, com uma probabilidade p_m, uma troca entre dois elementos da lista, fazendo com que a ordem de visita dos mesmos seja alterada. Esta troca é realiza verificando se os elementos alterados possuem ligação com os novos vizinhos. Se a rota resultante for inválida, ela é corrigida de forma que todos os elementos consecutivos tenham ligação entre si. O operador de cruzamento opera sobre dois indivíduos. Ele divide, com uma probabilidade p_c, ambas as rotas em um ponto e as mistura, gerando dois novos indivíduos. Se alguma das rotas tiverem elementos repetidos, estes são substituídos por um dos elementos que estiverem faltando, selecionado aleatoriamente. Estas operações são repetidas várias vezes, gerando uma nova população de tamanho $s_p \times p_r$, que é misturada com a população original. Em seguida, os elementos com os piores *fitness* são descartados da população, de forma que ela volte novamente ao tamanho original.

Todo este processo descrito acima é repetido n_g vezes, e ao final o melhor indivíduo da população é salvo. Este indivíduo representará a rota mais econômica gerada pelo algoritmo genético. Quando $n_g \rightarrow \infty$, a solução encontrada irá convergir para a solução ideal. As desvantagens dos Algoritmos Genéticos é que eles exigem um tempo muito maior de execução e/ou maior poder computacional que as demais heurísticas, e só garantem a solução ótima no infinito. As principais vantagens são: (i) tendo poder computacional e tempo suficiente, a solução encontrada será melhor que a solução obtida com os outros algoritmos; (ii) os Algoritmos Genéticos podem tirar proveito de arquiteturas multiprocessadas e *clusters*.

5. Resultados

Para que fosse possível obter um melhor entendimento das heurísticas, bem como para que fosse possível realizar experimentos com as três heurísticas, elas foram implementadas utilizando a linguagem de programação C++. Nas três implementações, a leitura do grafo é realizada através de um arquivo texto cujo nome é informado através da linha de comando. Este arquivo, além da matriz de adjacência, possui outras informações necessá-

rias para se implementar as restrições do modelo. As restrições implementadas foram: (i) Limite de combustível; (ii) Tempo máximo para se percorrer as rotas; e (iii) Capacidade máxima dos veículos. A medida que as rotas vão sendo calculadas, os programas exibem informações na tela relativas ao processamento realizado, de forma que o algoritmo possa ser compreendido facilmente. Ao final, são mostradas as rotas calculadas e a duração, o consumo de combustível e a ocupação dos veículos para cada uma delas.

Para que fosse possível avaliar o desempenho dos três algoritmos, foram geradas matrizes de adjacência aleatórias com 50, 75, 100, 150, 250, 500, 750 e 1000 nodos. Cada matriz de adjacência foi testada com os três algoritmos, e os resultados obtidos são mostrados na Tabela 3. Os custos estão expressos em quilômetros (comprimentos das trajetórias encontradas) e os tempos em segundos. Todos os experimentos foram realizados no mesmo computador, que possui um *clock* de 1.54GHz, 512MB de memória RAM e sistema operacional Linux.

Tabela 3. Comparação entre as heurísticas

	Clark e Wright		Mole e Jameson		Genético	
Tamanho	Tempo	Custo	Tempo	Custo	Tempo	Custo
50	0,00s	652,66	0,01s	521,40	16,91s	265,18
75	0,00s	1038,34	0,06s	587,46	74,56s	314,30
100	0,00s	1168,36	0,15s	704,19	128,25s	449,51
150	0,01s	1409.57	0,60s	849,73	215,36s	764,19
250	0,05s	1810,79	3,49s	1133,57	566,02s	1607,05
500	0,31s	2520,76	39,73s	1734,83	3168,57s	3744,18
1000	2,07s	3715,19	445,51s	2807,60	34913,63s	16988,64

Para os Algoritmos Genéticos, foram realizados cinco experimentos distintos para cada tamanho de entrada, e as médias dos resultados obtidos nestes experimentos são mostradas nas duas últimas colunas da Tabela 3. Isto se deve à natureza estocástica desta metodologia, que torna os resultados obtidos diferentes a cada experimento.

Pelos resultados mostrados na Tabela 3, pode-se perceber que o algoritmo de Mole e Jameson é bem mais lento que o de Clark e Wright, e a diferença do tempo de processamento se torna maior a medida que cresce o número de nodos. Isto se deve a maior complexidade computacional deste algoritmo. Em termos de qualidade da solução obtida, pode-se notar claramente que o algoritmo de Mole e Jameson é superior ao de Clark e Wright, fornecendo rotas em média 35% mais econômicas. Isto se deve à heurística mais elaborada, que torna a solução mais próxima do ótimo.

Já os Algoritmos Genéticos foram mais lentos que as outras heurísticas em todos os testes realizados, mas os resultados obtidos foram melhores que as outras técnicas para grafos com menos de 200 nodos. Já para entradas maiores, as soluções encontradas com os Algoritmos Genéticos deixaram bastante a desejar, sendo inclusive piores que as obtidas com as outras técnicas. Para melhorar o desempenho obtido, seria necessário aumentar o tamanho da população s_p e o número máximo de gerações n_g, o que implicaria em tempos maiores de execução.

É importante destacar que embora sejam mais lentos que as outras heurísticas,

os Algoritmos Genéticos são muito mais rápidos que a solução exata. Em um grafo de tamanho 14, por exemplo, os Algoritmos Genéticos levaram apenas 0,24 segundos para encontrar a solução ótima, enquanto que a solução exata, que testa todas as combinações possíveis, levou 4260,23 segundos.

6. Conclusões

Este artigo teve como objetivo descrever o problema de roteamento de veículos, bem como algumas heurísticas de aproximação para este problema. Os algoritmos de Mole e Jameson e de Clark e Wright possuem diversas similaridades entre si, principalmente pelo fato de ambos se basearem na maior economia para a construção das rotas. O algoritmo de Mole e Jameson, apesar de ser mais lento, é o mais indicado para ser utilizado, principalmente se o número de nodos não for muito elevado. Se este número for muito grande, pode ser mais vantajoso utilizar o algoritmo de Clark e Wright, devido a sua menor complexidade computacional. Já os Algoritmos Genéticos são mais aplicados para entradas de tamanho menor, mas conduzem a uma solução melhor se houver poder computacional suficiente. Como perspectivas futuras, pretende-se testar a solução baseada em Algoritmos Genéticos em um *cluster* composto de oito computadores, que em breve estará disponível em nossos laboratórios.

Referências

Clark, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Opns. Res.*, (12):568–581.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2002). *Algoritmos - Teoria e Prática*. Campus, Rio de Janeiro, RJ, Brazil, 2 edition.

Darwin, C. (1859). *Origin of Species*. John Murray, London, UK.

De Jong, K. A. (1975). *An Analysis of the Bahavior of a Class of Genetic Adaptative Systems*. Doctoral thesis, Univ. Michigan, Ann Arbor, MI.

Fisher, M. and Jaikumar, R. (1981). The lagrangean relaxation method for solving integer programming problem. *Mam. Sci.*, (27):01–18.

Goldbarg, M. C. and Luna, H. P. (2000). *Otimização Combinatória e Programação Linear - Modelos e Algoritmos*. Campus, Rio de Janeiro, Brazil.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Heinen, M. R. (2005). Análise e implementação de algoritmos para o roteamento de veículos. In *Anais do IV Simpósio de Informática da Região Centro do RS (SIRC/RS)*, pages 1–8, Santa Maria, RS, Brazil. UNIFRA Editora.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Univ. Michigan Press, Ann Arbor, MI.

Karp, R. M. (1975). On the computational complexity of combinatorial problems. *Neworks*, (5):45–68.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.

Mole, R. H. and Jameson, R. S. (1976). A sequencial routing-building algorithm employing a generalised savings criterion. *Opl. Res Q*, (27).