Algoritmo Transgenético para o Problema da Árvore Geradora Mínima Multicritério

Sílvia Maria Diniz Monteiro, Elizabeth Ferreira Gouvêa Goldbarg
Depto de Informática e Matemática Aplicada, DIMAp, UFRN,
59072-970, Natal, RN
silvinha_treze@yahoo.com.br, beth@dimap.ufrn.br

Marco César Goldbarg
Depto de Informática e Matemática Aplicada, DIMAp, UFRN,
59072-970, Natal, RN
gold@dimap.ufrn.br

Resumo: *No problema da Árvore Geradora Mínima Multicritério (AGM-mc), um vetor de custos é associado a cada aresta do grafo e o objetivo é encontrar as árvores geradoras Pareto-ótimas ou eficientes. O caminho hamiltoniano de custo mínimo é redutível ao problema da AGM bi-objetivo, o que mostra que a AGM-mc tem solução NP-árdua. Vários problemas reais podem ser modelados pela AGM-mc, como, por exemplo, o projeto de uma rede de infra-estrutura para distribuição de gás no qual se deve levar em conta não apenas o custo da instalação dos dutos, mas também a confiabilidade da conexão. Este trabalho apresenta um algoritmo transgenético para solução da AGM-mc. O algoritmo desenvolvido foi testado para um conjunto de 63 instâncias, compostas por grafos de 50 até 1000 vértices. A comparação foi realizada com um algoritmo memético e um transgenético que fazem parte do estado da arte do problema. Utilizaram-se os indicadores de qualidade epsilon unário e epsilon binário aditivo. O teste estatístico empregado foi o Mann Whitney. As comparações realizadas mostraram que o algoritmo transgenético desenvolvido é bastante competitivo tanto no que se refere ao tempo de execução quanto à qualidade de solução, tendo dominado os algoritmos da literatura em várias instâncias.*

Palavras-chave: Otimização Combinatória, Árvore Geradora Mínima Multicritério, Metaheurísticas, Algoritmos Evolucionários, Algoritmos Transgenéticos, Projeto de Redes.

Introdução

Grande parte dos problemas reais envolvem a maximização ou minimização de vários critérios ao mesmo tempo. Por exemplo, para projetar uma rede de distribuição de gás, pode ser desejável levar em conta o custo da instalação dos dutos, a confiabilidade da rede, o lucro obtido pelo atendimento de cada cliente, questões ambientais, etc. Em geral, os critérios são conflitantes e/ou a otimização de cada objetivo em particular não garante que a solução ótima considerando todos os objetivos seja encontrada. Assim, o tomador de decisões deve escolher a solução com melhor relação de compromisso entre os critérios. Neste contexto, os problemas mono-objetivo podem ser representações muito simplistas dos problemas reais.

A Otimização Multicritério (OMc) permite que os vários critérios sejam levados em conta simultaneamente, o que a torna capaz de representar os problemas de maneira mais realista. O problema geral de otimização multi-objetivo pode ser expresso por:

$$\text{Minimizar } C(x) = (c^1(x), \ldots, c^m(x)),$$

sujeito a $x \in X$, de maneira que se atendam as p restrições de igualdade ou desigualdade $r_i(x)$, com $i=1, \ldots, p$.

Considere x um vetor solução e X um conjunto finito de soluções viáveis. A função C(x) mapeia o conjunto de soluções viáveis X em R^m, onde $m > 1$ é o número de objetivos.

Em vez de uma solução ótima, obtêm-se um conjunto de soluções ótimas, que não são piores que nenhuma outra solução quando todos os critérios são levados em conta simultaneamente. Estas soluções são chamadas de Pareto-ótimas ou soluções eficientes. Fazem parte do conjunto de Pareto ótimo todas as soluções de X que não são dominadas por nenhuma outra. Pelo conceito de Dominância de Pareto, uma solução x domina uma solução y se, e somente se, para todo i variando de 1 a m, $c^i(x) \leq c^i(y)$ e $x \neq y$.

As soluções eficientes podem ser suportadas ou não suportadas. Soluções eficientes suportadas são aquelas que podem ser obtidas com uma ponderação dos objetivos de maneira que a soma dos pesos seja igual a um. As soluções não-suportadas são linearmente independentes, o que significa que ela não pode ser obtida por meio de nenhuma ponderação no vetor de objetivos em que a soma dos pesos seja 1. Encontrar soluções não-suportadas é um desafio ainda maior para os pesquisadores.

Na seção 2, discute-se brevemente o problema da Árvore Geradora Mínima Multicritério. A seguir,

apresentam-se os algoritmos transgenéticos. A seção 4 contém detalhes da implementação realizada. Na seção 5, descrevem-se os experimentos computacionais e os resultados obtidos. A seguir, são tecidas considerações finais sobre o trabalho desenvolvido.

O problema da AGM-mc

O problema da Árvore Geradora Mínima (AGM) consiste em encontrar uma árvore geradora, de custo mínimo, em um grafo com arestas valoradas. Assim, dado um grafo *G (V, E)* com *n* vértices e *m* arestas, o problema da AGM é determinar um subgrafo *G' (V, E')* de *G*, em forma de árvore, com *n* vértices e $m = n - 1$ arestas, de maneira que a soma dos custos das arestas de *G'* seja minimizada. Este problema apresenta solução polinomial e possui diversas aplicações, principalmente, na área de projeto de redes de infra-estrutura.

No problema da AGM com múltiplos critérios (AGM-mc), um vetor de pesos não negativos $w_{ij} = (w_{ij}^1, ..., w_{ij}^k)$, $k > 1$, é associado a cada aresta $(i, j) \in E$. Considere S o conjunto de todas as possíveis árvores geradoras $T = (V_T, E_T)$ de G e $W = (W^1, ..., W^k)$ onde $W^q = \sum_{(i,j)\in E_T} w_{ij}^q, q = 1, ..., k$. Neste caso, deseja-se encontrar o subconjunto S^* de S, de tal forma que $T^* \in S^*$ se, e somente se, $\nexists T \in S$ tal que $T \prec T^*$.

O caminho hamiltoniano mínimo é redutível à AGM bi-objetivo, o que demonstra que a AGM-mc tem solução NP-árdua. Isso significa que os algoritmos exatos existentes, para uma entrada suficientemente grande, podem exigir um tempo computacional inviável para encontrar a solução. Por isso, uma abordagem via programação metaheurística se mostra mais apropriada.

Dentre os algoritmos evolucionários, os algoritmos transgenéticos tem se destacado por fornecer soluções de qualidade em um tempo computacional pequeno quando comparado a outras abordagens evolucionárias da literatura. Assim, esta metaheurística foi a escolhida para solucionar a AGM-mc.

Algoritmos Transgenéticos

Algoritmos evolucionários são caracterizados pela existência de uma população de possíveis soluções a qual evolui de acordo com operadores de manipulação que promovem variabilidade e seleção [7]. Tais algoritmos são inspirados em processos evolutivos e mecanismos biológicos e realizam uma busca estocástica no espaço de soluções de problemas de otimização. A Transgenética Computacional (TC) apóia sua metáfora na cooperação como principal estratégia evolucionária. Sua inspiração é a transferência horizontal de genes e a evolução endossimbiótica [3].

A teoria endossimbiótica serial é uma teoria evolucionária que baseia sua formulação na união de indivíduos de naturezas distintas – de diferentes espécies – para a constituição de saltos adaptativos ou a formação de espécies híbridas ou novas [8]. A transferência horizontal de genes corresponde à aquisição de código genético, por organismos, de forma distinta da herança, também chamada de transferência vertical [4]. A transferência horizontal é uma importante força evolutiva, uma vez que permite que organismos tenham acesso à genes de outras espécies, que jamais poderiam ser herdados. Um conhecido veículo de transferência horizontal é o plasmídio. Um plasmídio é uma partícula genética móvel, DNA circular, que pode se replicar independente dos cromossomos. Eles podem ser naturais ou construídos artificialmente. No último caso são chamados de plasmídios recombinantes.

O contexto evolucionário dos algoritmos transgenéticos está em uma célula hospedeiro que co-evolve com seus endossimbiontes. Portanto, nos algoritmos transgenéticos, existe uma população de cromossomos, os quais representam soluções do problema sendo solucionado. Tais cromossomos são pensados como os endosimbiontes. A célula hospedeiro mantém informações próprias que se referem ao problema e ao método de busca. Tais informações são utilizadas por agentes para modificar os cromossomos endosimbiontes. Estes agentes são chamados de vetores transgenéticos [4] e se baseiam em vetores naturais de transferência horizontal.

Os vetores transgenéticos λ *(I,* Φ*)* interagem com a população de soluções viáveis por meio de uma cadeia de informação *I* e de um método de manipulação Φ, o qual representa a ação do vetor sobre o indivíduo. Há muitos tipos de vetores transgenéticos e de possíveis ações. Neste trabalho, foram utilizados os vetores plasmídio e plasmídio recombinante. Um vetor é dito plasmídio quando sua cadeia de informação é traduzida no formato genético – uma subcadeia de DNA – e seu método utiliza dois procedimentos:

- p_1 – Ataque: Define o critério de avaliação que estabelece quando um cromossomo *S* é suscetível à manipulação de um vetor λ.
- p_2 – Transcrição: Se Ataque (*S*, λ) = "verdadeiro", o procedimento define como a informação *I*, transportada pelo vetor λ, será transferida para o cromossomo *S*.

A cadeia de informações dos plasmídios é obtida das informações da célula hospedeira. No caso dos plasmídios recombinantes, a cadeia de informação

pode ser parcialmente ou integralmente gerada por um método construtivo.

Um agente plasmídio atua infiltrando sua cadeia de informações nos indivíduos da população, de acordo com o especificado no procedimento p_2.

Transgenético para a AGM-mc

O pseudo-código referente ao algoritmo transgenético proposto para a AGM-mc é exibido na figura 1. Inicialmente, uma população de cromossomos é gerada. Os indivíduos são codificados com a representação direta proposta por Raidl [9]. O algoritmo executa um número fixo de iterações (#numGer). A população tem tamanho fixo #tamPop e é gerada em duas fases.

```
(01) gere a população inicial (P={C1,...,C#tamP})
(02) crie o arquivo (G_A, P)
(03) para i = 1 até #numGer
(04)    crie os plasmídios (#numPlas)
(05)    para j = 1 até #tamPop
(06)       t ← random (1, #numPlas)
(07)       Cresultado ← plasmídio (Cj, t)
(08)       se (melhor (Cresultado, Cj))
(09)          Cj ← Cresultado
(10)       senão cont[j] ← cont[j] + 1
(11)       se (cont[j] = 2)
(12)          Cresultado ← plasmídioRecomb (Cj)
(13)          cont[j] = 0
(14)          se (melhor (Cresultado, Cj))
(15)             Cj ← Cresultado
(16)       fim_se_linha_11
(17)       atualize (G_A, Cj)
(18)    fim_for_j
(19) fim_for_i
```

Figura 1. Pseudo-código para o algoritmo proposto

Na primeira fase, no máximo #popIni indivíduos são obtidos pelo rmc-Kruskal [10], um método guloso randômico baseado no algoritmo de Kruskal para AGM com apenas um objetivo. No rmc-Kruskal, o vetor w_{ij} de pesos das arestas é substituído pelo valor resultante do produto interno $v.w_{ij}$., onde v é um vetor de escalarização. Inspirado na fase construtiva do GRASP [2], a cada passo de decisão, cria-se uma lista restrita de candidatos, LRC, e um elemento desta lista é escolhido randomicamente para ser adicionado à solução. Nesta fase, se forem gerados dois indivíduos iguais, apenas um deles permanece na solução. Os outros indivíduos da população são gerados usando o método RandomWalk [9].

O algoritmo armazena um arquivo de soluções não-dominadas *G_A*. No início, este arquivo é composto pelas soluções não dominadas da população inicial. O número máximo de soluções não-dominadas neste arquivo é limitado em 300. O arquivo *G_A* é gerenciado de acordo com a proposta de Knowles [5]. Assim, o espaço de objetivos é representado por uma estrutura de grid multi-dimensional dividida em células em que uma posição é associada a cada solução presente no grid. As soluções não-dominadas sempre são adicionadas ao arquivo. Se houver menos de 300 soluções, a nova solução é adicionada. Caso o arquivo já esteja cheio, escolhe-se, aleatoriamente, uma solução da região mais populosa do grid para ser retirada do arquivo e dar lugar à nova solução.

A cada iteração, são gerados *#numPlas* plasmídios de dois tipos diferentes. Os plasmídios do primeiro tipo são gerados a partir de informações advindas de árvores geradoras mínimas obtidas com o método rmc-Kruskal. O tamanho da LRC é 0,05*n*, onde *n* é o número de arestas da árvore geradora do grafo, ou seja, o número de vértices menos 1. A informação do segundo tipo de plasmídio é obtida do arquivo de soluções não dominadas. Esta informação é construída escolhendo-se, de forma aleatória, um trecho de uma solução situada em uma região pouco populosa do grid. O tamanho da cadeia de informação armazenada por cada plasmídio é um valor escolhido aleatoriamente, que varia de 0,3*n* a 0,6*n*.

O plasmídio recombinante tem sua cadeia de informação construída com base no método rmc-Kruskal. Quando este vetor transgenético ataca um indivíduo, aplica-se uma escalarização aos vetores de pesos das arestas. Apenas um fragmento entre 0,05*n* e 0,1*n* do cromossomo do indivíduo permanece na solução. Este fragmento é escolhido aleatoriamente, de maneira que as arestas com menor valor com base na escalarização têm maior probabilidade de continuarem na solução. A cadeia de informação do plasmídio recombinado é, então, utilizada para completar a árvore geradora.

O loop principal do algoritmo (linha 03) é executado *#numGer* vezes. Em cada geração, todos os cromossomos da população são atacados por um dos vetores plasmídios criados, o qual é escolhido randomicamente. O operador de transcrição constrói uma árvore com as arestas da cadeia de informação do plasmídio selecionado. A seguir, são inseridas na árvore todas as arestas do indivíduo que não formam ciclos com as arestas do plasmídio. Se necessário, arestas são escolhidas aleatoriamente para serem adicionadas à solução até que uma árvore geradora seja formada.

Associado a cada indivíduo da população, há um contador de falhas, com valor inicial igual a 0. A cada vez que um ataque de plasmídio não consegue melhorar a solução, o contador de falhas para o indivíduo correspondente é incrementado. Caso um número máximo de falhas (no caso, 2) seja

alcançado, o plasmídio recombinado é aplicado e o contador é zerado.

A função que avalia se o ataque dos vetores transgenéticos implicou melhoria, ou não, do indivíduo é a função *melhor*. *melhor* ($C_{resultado}$, C_j) avalia $C_{resultado}$ com sendo melhor que C_j se $C_{resultado}$ domina C_j ou se $C_{resultado}$ é uma solução não-dominada em relação a C_j e às soluções do arquivo global.

Uma solução obtida pelo ataque de um dos vetores transgenéticos mencionados só substitui o indivíduo da população em caso de melhoria.

Experimentos Computacionais

O algoritmo transgenético proposto (MT) foi comparado a um eficiente algoritmo transgenético (RT), proposto por Rocha et al. [11]. Ambos os algoritmos foram implementados em C e testados em uma máquina Pentium IV de 3,2GHz e 1G de RAM, com KUbuntu 7.10, utilizando o gcc. Os algoritmos foram aplicados a 63 instâncias geradas de acordo com o descrito no trabalho de Knowles [5]. Três grupos de 21 instâncias pertencentes às classes *concave*, *correlated* e *anti-correlated* foram geradas como grafos completos com 2 objetivos. Cada classe apresenta uma instância com número de vértices (*nv*) = 50 e duas instâncias com *nv* de 100 até 1000. Para gerar as instâncias *correlated* e *anti-correlated* é necessário informar um fator de correlação β. Para gerar as instâncias concave, precisa-se de dois parâmetros, ζ e η. A tabela 1 mostra os parâmetros utilizados para a geração das instâncias.

Foram realizadas 30 execuções independentes de cada algoritmo para cada instância. O número de gerações do MT, *#numGer*, é 30, o tamanho da população, *#tamPop*, é 150 e *#popIni* é 142. A cada geração, criam-se *#numPlas* igual a 10 plasmídios.

A comparação entre algoritmos de programação multiobjetivo é realizada por meio de indicadores de qualidade. Neste trabalho, adotou-se o indicador de qualidade épsilon binário [6].

O teste estatístico de Mann-Whithney (U-test) é utilizado para verificar a significância dos resultados obtidos com o indicador. Este teste, também chamado de Mann-Withney-Wilcoxon, é um teste não-paramétrico usado para verificar a hipótese nula de que duas amostras vêm da mesma população [1].

Os *p-values* resultantes são mostrados na tabela 2, separados por classe e por instância. O nível de significância adotado é 0,05. Assim, um valor menor que 0,05 indica que o algoritmo correspondente exibe um valor melhor de acordo com este indicador de qualidade (épsilon binário).

Tabela 1. Parâmetros das instâncias

Id	Conc		Corr	Anticorr
	Z	H	β	β
50	0,03	0,125	0,7	-0,7
100_1	0,01	0,02	0,3	-0,3
100_2	0,02	0,1	0,7	-0,7
200_1	0,05	0,2	0,3	-0,3
200_2	0,08	0,1	0,7	-0,7
300_1	0,03	0,1	0,3	-0,3
300_2	0,05	0,125	0,7	-0,7
400_1	0,025	0,125	0,3	-0,3
400_2	0,04	0,2	0,7	-0,7
500_1	0,02	0,1	0,3	-0,3
500_2	0,03	0,15	0,7	-0,7
600_1	0,0016	0,1	0,125	-0,125
600_2	0,002	0,02	0,95	-0,95
700_1	0,0014	0,03	0,35	-0,35
700_2	0,001	0,008	0,7	-0,7
800_1	0,00125	0,035	0,45	-0,45
800_2	0,0015	0,03	0,05	-0,05
900_1	0,0011	0,009	0,15	-0,15
900_2	0,002	0,01	0,85	-0,85
1000_1	0,001	0,2	0,4	-0,4
1000_2	0,0005	0,1	0,9	-0,9

Tabela 2. p-values para as instâncias testadas

Id	Conc		Corr		Anticorr	
	MT	RT	MT	RT	MT	RT
50	**0**	1	**0**	1	**0**	1
100_1	**0**	1	**0**	1	**0**	1
100_2	**0**	1	**0**	1	**0**	1
200_1	**0**	1	**0**	1	**0**	1
200_2	**0**	1	**0**	1	**0**	1
300_1	**0**	1	**0**	1	**0**	1
300_2	**0**	1	**0**	1	**0**	1
400_1	**0**	1	**0**	1	0,26	0,74
400_2	**0**	1	**0**	1	**0,03**	0,97
500_1	**0**	1	**0**	1	**0**	1
500_2	**0**	1	**0**	1	**0**	1
600_1	**0**	1	**0**	1	**0**	1
600_2	**0,01**	0,99	**0**	1	**0**	1
700_1	0,61	0,39	**0**	1	**0**	1
700_2	**0**	1	**0**	1	**0**	1
800_1	0,18	0,82	**0**	1	**0**	1
800_2	0,86	0,14	**0**	1	**0**	1
900_1	**0**	1	**0**	1	**0**	1
900_2	**0**	1	**0**	1	**0**	1
1000_1	0,24	0,76	**0**	1	**0**	1
1000_2	0,25	0,75	**0**	1	**0**	1

Pode-se observar, na tabela 2, que o algoritmo proposto foi melhor que o RT para a maioria das instâncias em todas as classes.

No caso das instâncias *concave*, o MT só não foi melhor que o RT, segundo o indicador de qualidade épsilon binário, para as instâncias 700_1, 800_1, 800_2, 1000_1 e 1000_2, de um conjunto de 21 instâncias. Mesmo nessas instâncias, o RT não conseguiu ser melhor que o MT. Para todas as outras 16 instâncias, o MT foi melhor que o RT, de acordo com o épsilon binário.

O algoritmo MT, proposto neste trabalho, foi melhor que o RT em todas as instâncias da classe *correlated*, pelo indicador de qualidade épsilon binário. Este indicador também considera o MT melhor que o RT para todas as instâncias da classe *anti-correlated*, com exceção da instância 400_1. Se o nível de significância fosse reduzido para 0,01, o MT só deixaria de ser melhor que o RT para uma instância, a *anti-correlated* 400_2.

Os tempos de execução (em segundos, desconsiderando as casas decimais) dos dois algoritmos em análise são exibidos na tabela 3. O tempo do MT é significativamente menor que o tempo do RT.

Tabela 3. Tempos de execução

Id	Conc		Corr		Anticorr	
	MT	RT	MT	RT	MT	RT
50	0	4	0	3	0	12
100_1	1	6	1	7	2	22
100_2	1	7	1	6	2	28
200_1	5	19	6	25	6	60
200_2	5	15	6	23	6	76
300_1	13	39	13	48	14	93
300_2	13	34	13	47	14	125
400_1	25	64	25	74	26	134
400_2	25	67	24	75	25	169
500_1	40	98	41	109	41	178
500_2	40	94	40	98	41	213
600_1	59	137	59	138	59	190
600_2	59	134	54	133	58	251
700_1	84	185	84	184	84	291
700_2	84	181	81	180	82	297
800_1	109	236	108	235	107	354
800_2	110	235	110	230	110	264
900_1	141	293	140	298	140	362
900_2	140	289	133	290	135	399
1000_1	150	337	175	356	172	470
1000_2	172	354	165	342	169	456

Considerações Finais

Este trabalho propõe um algoritmo transgenético para solução da Árvore Geradora Mínima Multicritério. O algoritmo desenvolvido foi comparado a um eficiente algoritmo transgenético da literatura para o problema, proposto por Rocha et al. [11]. Para a comparação, utilizou-se o indicador de qualidade épsilon binário. Os testes estatísticos de Mann-Withney utilizados mostraram que, segundo este indicador de qualidade, o algoritmo desenvolvido tem melhor desempenho que o da literatura para as três classes de instâncias apresentadas. O tempo de computação obtido com esta implementação foi significativamente menor que o do algoritmo com o qual se comparou.

Referências

[1] W. J. Conover, "Practical Nonparametric Statistics" (3rd Ed.), John Wiley & Sons, 2001.

[2] T. A. Feo, M. G. C. Resende, "Greedy Randomized Adaptive Search Procedures", *Journal of Global Optimization* 6, 1995, pp. 109-133.

[3] M. C. Goldbarg, L. B. Bagi, E. F. G. Goldbarg. "Algoritmo Transgenético para o Problema do Caixeiro Comprador Capacitado". In: 1st Workshop of Computational Intelligence, 2006, Ribeirão Preto. *Proceedings of the International Joint Conference IBERAMIA/SBIA/SBRN 2006 - 1st Workshop on Computational Intelligence*, 2006. v. 1. p. 23-28.

[4] E F G. Golbarg, M C. Goldbarg, C C. Schmidt. "A hybrid transgenetic algorithm for the prize collecting steiner tree problem". *Journal of Universal Computer Science*. 2008. accepted for publication.

[5] J. D. Knowles, "Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization", PhD Thesis. Department of Computer Science, University of Reading, Reading, UK, 2002.

[6] J. D. Knowles, D. W. Corne, "A Comparison of Encodings and Algorithms for Multiobjective Spanning Tree Problems", P*roceedings of the 2001 Congress on Evolutionary Computation* (CEC01), 2002, pp. 544-551.

[7] Z. Michalewicz, D. B. Fogel. "How to solve it: Modern heuristics". Springer, 2000.

[8] H. J. Morowitz. "Beginning of Cellular Life". New Haven, Conn, Yale University Press, 1992.

[9] G. R. Raidl, "An Efficient Evolutionary Algorithm for the Degree-constrained Minimum Spanning Tree Problem", *Proceedings of the 2000 Congress on Evolutionary Computation* (CEC 2000), IEEE Press, 2000, pp. 104-111.

[10] D. A. M. Rocha, E. F. G. Goldbarg, M. C. Goldbarg, "A Memetic Algorithm for the

Biobjective Minimum Spanning Tree Problem", in: *6th European Conference on Evolutionary Computation in Combinatorial Optimization*, 2006, Budapeste, Lecture Notes in Computer Science 3906 Heidelberg , Springer Berlin, 2006, pp. 222-233.

[11] D. A. M. Rocha, E. F. G. Goldbarg, M. C. Goldbarg. "A New Evolutionary Algorithm for the Bi-objective Minimum Spanning Tree". In: *ISDA'07 Seventh International Conference on Intelligent Systems Design and Applications*, 2007, Rio de Janeiro. Proceedings of ISDA'07. Danvers: IEEE Computer Society, 2007. v. 1. p. 735-740.