Algoritmo Imunológico Artificial CLONALG e Algoritmo Genético Aplicados ao Problema do Caixeiro Viajante

Simone S. F. Souza **Ruben Romero**
Departamento de Engenharia Elétrica, Faculdade de Engenharia de Ilha Solteira (FEIS)
Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP), Ilha Solteira, SP, BRASIL
E-mails: simonefrutuoso.mat@gmail.com, ruben@dee.feis.unesp.br

Palavras-chaves: *Problema do Caixeiro Viajante, Sistemas Imunológicos Artificiais, Algoritmo de Seleção Clonal, Algoritmo Genético.*

Resumo: *Neste artigo apresenta-se uma comparação entre o algoritmo de seleção clonal e o algoritmo genético, quando aplicados na resolução do problema do caixeiro viajante (PCV). Tanto os algoritmos evolutivos, como os imunológicos são abordagens eficientes para a solução de problemas de otimização combinatorial. Desta forma, visando obter parâmetros comparativos (benchmarking) entre os métodos, ambos foram codificados na mesma linguagem (Matlab) e executados no mesmo computador, sendo testados com as mesmas instancias do problema. Para avaliar os métodos, utilizou-se 15 instâncias do PCV simétrico da base de dados TSPLIB, das quais a solução ótima é conhecida, possibilitando uma análise comparativa mais precisa. Através dos resultados obtidos obteve-se uma situação de trade-off de qualidade de solução e tempo computacional.*

1 Introdução

Os Sistemas Imunológicos Artificiais [1] são abordagens inspiradas nos sistemas imunológicos biológicos. Sua concepção visa reproduzir computacionalmente as principais habilidades, propriedades e princípios dos mecanismos naturais de defesa, seleção e maturação.

O algoritmo de seleção clonal (CLONALG) é inspirado no princípio de seleção clonal e nos mecanismos de seleção e maturação proporcional a afinidade [3]. Este algoritmo é uma abordagem clássica imunológica, cujo princípio básico de funcionamento é classificado como algoritmos evolutivos (AE) e têm se destacado pela eficiente solução de problemas de busca e otimização, principalmente os de caráter combinatório.

Os algoritmos evolutivos propõem um paradigma de solução de problemas através da inspiração na biologia evolutiva, principalmente na teoria da seleção natural. Dada uma população de indivíduos, os de melhor qualidade têm maior probabilidade de sobrevivência e reprodução, podendo continuar no processo de busca pela solução ótima. Esta é uma característica da evolução das espécies que também é observada em alguns algoritmos de sistemas imunológicos artificiais, como o algoritmo CLONALG, que são baseados na teoria da seleção clonal e maturação de afinidade [3].

Neste artigo, apresenta-se um algoritmo CLONALG e um Algoritmo Genético (AG) para resolver o problema do caixeiro viajante. Neste contexto será realizada uma análise comparativa entre os dois métodos, a fim de avaliar o desempenho computacional e a qualidade das soluções obtidas. Para está análise os métodos foram executados no mesmo computador, codificados na mesma linguagem de programação e com o mesmo número de iterações.

Para avaliar os dois métodos ambos foram testados com instâncias simétricas da base de dados TSPLIB (*Traveling Salesman problem Library*) [16]. Trata-se de uma base de dados clássica disponível na literatura.

Este artigo está organizado como a seguir. Na seção 2 apresenta-se o problema do caixeiro viajante. A descrição do algoritmo de seleção clonal (CLONALG) está na seção 3. Na seção 4 apresenta-se a descrição do AG. Os testes e resultados são apresentados na seção 5 e por fim, a seção 6 apresenta a conclusão para este trabalho.

DOI: 10.5540/03.2014.002.01.0106

2 O problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) pode ser descrito da seguinte forma:

Dado um conjunto de *M* cidades, o objetivo é determinar a rota de menor custo (distância) tal que: a rota (tour) inicia e termina na mesma cidade, cada cidade é visitada apenas uma vez, a distância da rota *ij* tem o mesmo valor para a rota *ji*.

Para o PCV existem diversos modelos matemáticos, sendo o modelo de Dantzig-Fulkerson-Johnson um dos mais utilizados na literatura, que é descrito conforme a seguir [9]:

$$\text{Minimizar } z = \sum_{j=1}^{n}\sum_{i=1}^{n} c_{ij}x_{ij} \tag{1}$$

Sujeito a:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in N \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in N \tag{3}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - 1 \qquad \forall S \subset N \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j \in N \tag{5}$$

$$c_{ij} = c_{ji} \qquad \forall i, j \in N \tag{6}$$

em que: *N* é o conjunto de cidades (vértices), *A* é o conjunto de caminhos (arcos), $x_{ij} = 1$, se o arco $i, j \in A$ for escolhido para integrar a solução e $x_{ij} = 0$ caso contrário, *S* é um subgrafo de G=(*N*,*A*), |*S*| é o número de vértices do subgrafo *S* e c_{ij} ou c_{ji} é o custo do caminho.

O PCV é classificado como um problema NP-difícil, desta forma, os métodos exatos não são utilizados para resolver instâncias de grande porte, assim os métodos de otimização combinatorial são comumente empregados em sua resolução [8].

3 Metodologia

Nesta seção descreve-se a codificação dos algoritmos para a resolução do problema do caixeiro viajante.

3.1 Representação de uma solução para o PCV

O PCV foi codificado conforme ilustrado na figura 1, em que um vetor de tamanho *n* representa a proposta de solução. Cada elemento do vetor representa um vértice do problema, isto é, identifica uma cidade. Assim, o vetor solução apresenta um tour em que as cidades são visitadas na sequencia em que se encontram na codificação.

3	4	2	1	5

Figura 1: Representação de uma proposta de solução.

3.1 Algoritmo de Seleção Clonal aplicado ao PCV

O algoritmo de seleção clonal (CLONALG – *Clonal Selection Algorithm*) se inspira no princípio de seleção clonal que ocorre no sistema imunológico biológico. Este algoritmo foi proposto por [3]. Na sequência apresenta-se o algoritmo CLONALG para resolver o PCV:

DOI: 10.5540/03.2014.002.01.0106

Passo 1: Gere uma população (*P*) com *N* anticorpos (soluções candidatas). A população inicial foi gerada com a heurística do vizinho mais próximo, conforme descrita em [11], [15];

Passo 2: Avalie a afinidade (função objetivo) de cada anticorpo e selecione (processo de seleção) os *n* melhores anticorpos da população *P*, obtendo o conjunto $P_{\{n\}}$;

Passo 3: Reproduza (processo de clonagem) os *n* melhores anticorpos selecionados, gerando uma população (C) com N_c clones. A quantidade de clones de cada anticorpo é diretamente proporcional a sua afinidade;

Passo 4: Submeta a população de clones (*C*) a um processo de hipermutação, onde a taxa de mutação é proporcional à afinidade do anticorpo. Uma população (*C**) de anticorpos maduros/maturados é gerada. Para fazer a maturação nos anticorpos clonados foi utilizado o método de melhoria 2-opt [12];

Passo 5: Avalie a afinidade de cada anticorpo pertencente a (*C**) e re-selecione os *n* melhores anticorpos ($C^*_{\{n\}}$) e os adicione a população *P* descartando os anticorpos de pior qualidade;

Passo 6: Substitua *d* anticorpos de baixa afinidade por novos anticorpos ($P_{\{d\}}$) (diversidade ou metadinâmica). Os anticorpos com baixa afinidade possuem maior probabilidade de serem substituídos. Os novos anticorpos sáo gerados da mesma forma que os;

Passo 7: Repita os passos de 2 a 6 até satisfazer o critério de parada.

O algoritmo CLONALG possui cinco parâmetros principais, descritos a seguir [5]: Tamanho da população de anticorpos (*N*): Especifica o total de anticorpos a serem mantidos pelo sistema; Tamanho da área de seleção (*n*): Refere-se ao número total de anticorpos com maiores afinidades que são selecionados da população para serem clonados; Quantidade de anticorpos a serem substituídos (*d*): Número de anticorpos com baixa afinidade que devem ser substituídos por novos anticorpos; Fator de clonagem (β): Controla o número de clones gerados; Número máximo de gerações: Especifica o número total de iterações que o algoritmo deve realizar.

A quantidade N_c de clones gerada no Passo 3 para cada anticorpo *i* é dada pela equação (7) [4]:

$$N_c^i = round(\frac{\beta N}{i}) \tag{7}$$

onde β é um fator multiplicativo entre [0,1], *N* é a quantidade total de anticorpos da população *P*, e *round*(.) é o operador de arredondamento para o inteiro mais próximo.

A taxa de mutação (α) de cada clone é definida pela equação (8) [4]:

$$\alpha = \exp(-\rho D^*) \tag{8}$$

em que ρ é um parâmetro de controle de amortecimento da função exponencial e *D** é o valor normalizado da afinidade *D*, que pode ser calculado conforme apresentado na equação (9).

$$D^* = \frac{D_{\min}}{D} \tag{9}$$

Desta forma, cada clone sofre um processo de mutação dado por [6]:

$$m = round(\alpha * N(0,1)) \tag{10}$$

onde *m* é a quantidade de mutações que cada clone do anticorpo sofrerá, *round*(.) é o operador de arredondamento para o inteiro mais próximo, α é a taxa de mutação e *N*(0,1) é uma variável randômica gaussiana de média zero e desvio padrão $\sigma = 1$.

4 Algoritmo Genético aplicado ao PCV

A versão mais popular do AG foi proposta em [10]. O AG representa a principal classe dos algoritmos evolutivos. Na sequência apresenta-se um AG para o PCV:

Passo 1: Gere uma população (*P*) com *N* indivíduos (possíveis soluções). A população inicial foi gerada com a heurística do vizinho mais próximo, conforme descrita em [11], [15];

DOI: 10.5540/03.2014.002.01.0106

Passo 2: Avalie o fitness (função objetivo) de cada individuo da população (*P*);
Passo 3: Selecione a partir da população (*P*) pares de indivíduos com melhor valor de fitness. Neste caso foi utilizada a seleção por torneio, sendo selecionados 50% dos indivíduos;
Passo 4: Implemente o operador de recombinação genética. Nesta etapa os pares de indivíduos (pais) selecionados são combinados, através do operador *order crossover* proposto em [2], gerando dois descendentes cada.
Passo 5: Implemente o operador de mutação genética. A mutação consiste em criar variabilidade na população através de pequenas perturbações (mudanças na codificação genética), com o intuito de introduzir diversidade nos descendentes gerados. Para a mutação foi utilizado o método de melhoria 2-opt [12]. Os descendentes mutados substituem os pais na nova população;
Passo 6: Repita os passos de 2 a 5 até que o critério de parada seja satisfeito.

Os principais parâmetros do AG são [2]: *N*: é a quantidade de indivíduos da população; *pr*: taxa de recombinação (crossover); *pm*: taxa de mutação; Número máximo de gerações: Especifica o número total de iterações que o algoritmo deve realizar.

4 Resultados

Neste tópico apresentam-se os resultados obtidos nos testes realizados para os algoritmos. Todos os testes foram realizados utilizando um PC Intel Core 2 Duo 1.9 GHz, 2 GB de Memória RAM, e sistema operacional Windows 7 Ultimate 32 bits. Os dois algoritmos foram codificados e implementados em MATLAB ® [13]. Visando garantir a veracidade das soluções, cada teste foi executado 20 vezes e foi calculada a média e desvio percentual das soluções obtidas em relação a melhor solução conhecida.

Nos testes realizados os dois algoritmos foram executados com a mesma quantidade de iterações. Para fazer as comparações e testes, foram utilizados 15 problemas obtidos na base de dados TSPLIB (*Traveling Salesman problem Library*) [16]. Os problemas possuem entre 26 e 176 cidades.

A Tabela 1 apresenta os parâmetros do AG. Na Tabela 2 apresentam-se os parâmetros do algoritmo CLONALG. Todos os parâmetros, de ambos os métodos, foram obtidos de forma empírica, através de testes.

Tabela 1 – Parâmetros do AG.

Parâmetros	Valor
Tamanho da população (*N*)	120
Taxa de Mutação (*pm*)	0,15
Taxa de Recombinação (*pr*)	0,8
Número máximo de gerações	3000

Tabela 2 – Parâmetros do CLONALG.

Parâmetros	Valor
Tamanho da população (*N*)	120
Tamanho da área de seleção (*n*)	30
Quantidade de anticorpos a serem substituídos (*d*)	5
Fator de Clonagem (β)	0,2
Taxa de controle da função exponencial (ρ)	2
Número máximo de gerações	3000

Na Tabela 3 apresentam-se os resultados dos testes computacionais realizados, em que: MTC é o melhor tour conhecido na literatura, MT é a distância do melhor tour encontrado por cada algoritmo, MDT é a distância média das 20 execuções, DESV é o desvio Percentual da solução média para a melhor solução conhecida (DESV = MDT / MTC) e TP é o tempo médio de processamento das 20 execuções em segundos.

DOI: 10.5540/03.2014.002.01.0106

Tabela 3 – Resultados.

TSPLIB	MTC (melhor Tour Conhecido)	Algoritmo Genético				Algoritmo CLONALG			
		MT (Melhor Tour)	MDT (Tour Médio)	DESV (Desvio Percentual)	TP (Tempo Médio)	MT	MDT	DESV	TP
brazil158	25395	25408	25623,8	0,901	163,45	25395	25549,7	0,690	202,48
pr76	108159	108192	108982	0,761	82,14	108159	108262	0,095	175,26
ch150	6528	6553	6592,6	0,990	176,45	6534	6579,95	0,796	198,60
bayg29	1610	1610	1624,8	0,919	36,12	1610	1620,4	0,646	88,64
pr124	59030	59030	59081	0,086	153,45	59030	59077	0,080	142,58
pr107	44303	44303	44303	0,000	89,53	44303	44303	0,000	91,52
pr136	96772	98736	100840	4,204	169,54	97990	98990	2,292	211,20
pr144	58537	58537	58776,1	0,408	195,32	58741	59055,1	0,885	245,23
eil101	629	632	638,2	1,463	112,23	629	635	0,954	234,15
lin105	14379	14379	14389,5	0,073	142,56	14379	14386,3	0,051	305,45
eil176	538	583	549	2,045	74,56	543	546,8	1,636	158,95
eil151	426	438	431,05	1,185	42,15	441	434,3	1,948	97,12
berlin52	7542	7542	7542	0,000	42,83	7542	7542	0,000	153,06
st70	675	675	682,8	1,156	69,42	675	678,25	0,481	181,21
bier127	118282	118493	119957,4	1,416	192,42	118282	119216	0,621	303,18
Mínimo				0,000	36,12			0,000	88,64
Média				1,040	116,14			0,745	185,90
Máximo				4,204	195,32			2,292	305,45

Analisando os resultados obtidos pelo AG, a média da melhor solução obtida se difere em 1,040% do ótimo. Avaliando todas as soluções obtidas, observa-se que no máximo uma solução se difere em 4,204% do ótimo.

Nos resultados obtidos pelo algoritmo CLONALG, observa-se que, em média, a melhor solução obtida se difere em 0,745% do ótimo. Avaliando todas as soluções obtidas, observa-se que no máximo uma solução se difere em 2,292% do ótimo.

Comparando os resultados obtidos pelos dois algoritmos, o CLONALG foi capaz de obter soluções de melhor qualidade para a maioria das instâncias, como pode ser observado pelos valores em destaque na Tabela 3. Em geral, nas execuções o algoritmo CLONALG encontrou a solução ótima mais vezes que o AG, analisando as melhores soluções encontradas pelos dois algoritmos, o CLONALG superou o AG em 7 instâncias.

Fazendo uma comparação em relação ao tempo de processamento computacional observa-se que o AG tem melhor desempenho que o algoritmo CLONALG. O tempo médio de execução para todas as soluções, o AG foi executado em 116,14 s, e o algoritmo CLONALG foi executado em 185,90 s. O tempo de processamento mais elevado do CLONALG se justifica, pelo fato de que são gerados muitos clones de anticorpos, assim o processo de mutação para todos os clones é muito caro em termos computacionais.

O algoritmo CLONALG encontra soluções de melhor qualidade em contraste com seu maior tempo de execução em comparação ao AG. Isto é um exemplo clássico de *trade-off*, ou seja, quando compara-se dois algoritmos e uma apresenta soluções de melhor qualidade, mais tempo de processamento inferior ao outro algoritmo [14]. Assim, caso a prioridade seja uma solução de melhor qualidade, o algoritmo CLONALG é a melhor escolha entre os dois algoritmos, abrindo mão de um menor tempo de resposta. No entanto, caso a prioridade seja uma resposta rápida, pode ser possível abrir mão de uma melhor qualidade de solução, escolhendo o AG.

5 Conclusão

Neste artigo foram utilizados dois algoritmos para resolver um problema clássico de otimização combinatória, o PCV. O algoritmo CLONALG é uma técnica de sistemas imunológicos artificiais com características evolutivas. O AG compõe a classe de algoritmos evolutivos mais

DOI: 10.5540/03.2014.002.01.0106

difundidos na literatura. Para avaliar o desempenho dos algoritmos foram feitos vários testes com instâncias disponíveis na literatura, das quais são conhecidas as melhores soluções, possibilitando uma avaliação e comparação mais precisa dos algoritmos. Os resultados demonstraram uma boa performance de ambos os métodos, sendo que, em média, o AG obteve soluções que desviam 1,040% do ótimo e o algoritmo CLONALG obteve soluções que desviam 0,745% do ótimo. O algoritmo CLONALG encontrou uma maior quantidade de soluções ótimas, superando o AG em 7 instâncias. Na análise de tempo de processamento, constatou-se que o AG é superior ao CLONALG, desta forma, ocorre o *trade-off* entre qualidade de solução e tempo de processamento. Assim conclui-se que o algoritmo CLONALG foi capaz de obter soluções de melhor qualidade que o AG, no entanto, com um esforço computacional maior.

Agradecimentos

Os autores agradecem a CAPES e ao CNPq pelo apoio financeiro de pesquisa.

Referências

[1] D. Dasgupta. "Artficial Immune Systems and Their Applications". Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[2] L. Davis. Handbook of genetic algorithm, Van Nostrand Reinhold Company. 1991.

[3] L. N. de Castro and J. F. Von Zuben. The clonal selection algorithm with engineering applications. In: Workshop Proceedings of Gecco, Workshop on Artificial Immune Systems and Their Applications, 2000, Las Vegas. p. 36-39, (2000).

[4] L. N. de Castro. "Engenharia Imunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais". Tese de Doutorado, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, Brasil, 2001.

[5] L. N. de Castro and J. I. Timmis. Artificial immune systems: A new computational intelligence approach, Springer-Verlag, Heidelberg. 2002.

[6] F. O. de França; J. F. VON ZUBEN and L. N. de CASTRO. An Artificial Immune Network for Multimodal Function Optimization on Dynamic Environments." In: Proc. GECCO, Washington, DC, USA, pp. 289–296, (2005).

[7] S. Forrest; A. Perelson; L. Allen and R. Cherukuri. Self-Nonself Discrimination in a computer, Proc. do IEEE Symposium on Research in Security and Privacy, pp. 202-212 (1994).

[8] M. R. Garey and D. S. Johnson. Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman & Co., New York (1979).

[9] M. C. Goldbarg; H. P. L. Luna. Otimização combinatória e programação linear: modelos e algoritmos. 2. ed. Rio de Janeiro: Campus, 2005.

[10] J. H. Holland. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence, MIT Press (1992).

[11] G. Laporte; M. Gendreau; J.Y. Potvin e F. Semet. Classical and modern heuristics for the vehicle routing problem, International Transactions in Operational Research, v.7, n4/5, p.285-300, (2000).

[12] S. Lin and B. W. Kernighan. An Effective Heuristic Algorithm for the Traveling Salesman Problem,

Operations Research, v.21, p.498-516, (1973).

[13] Matlab (2011). 7.8 Version, Mathworks Company.

[14] Z. Michalewicz and D. B. Fogel. How to solve it: Modern heuristics, Springer (2000).

[15] A. G. Novaes. Logística e Gerenciamento da Cadeia de Distribuição. Rio de Janeiro: Campus, (2001).

[16] TSPLIB (2013). Traveling Salesman problem Library, disponível em: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

