Análise da Importância de Parâmetros em um Algoritmo Genético por meio de sua Aplicação no Aprendizado de uma Rede Neural

José Antonio Sánchez Guerrero, Lizet Liñero Suárez, Ricardo Ribeiro Gudwin

Departamento de Engenharia de Computação e Automação Industrial - DCA
Faculdade de Engenharia Elétrica e de Computação - FEEC
Universidade Estadual de Campinas - UNICAMP
{jasg, llinero, gudwin}@dca.fee.unicamp.br

Resumo: Este trabalho visa investigar a sensibilidade e importância dos diversos parâmetros utilizados em um algoritmo genético, utilizando como plataforma de teste sua aplicação ao aprendizado de uma rede neural. Utiliza-se para tanto uma representação modificada do algoritmo genético que é mais rica em graus de liberdade, de modo a aumentar o escopo de possibilidades para a sintonia do GA. Diversas combinações possíveis entre estes parâmetros são testadas e por fim os resultados são comparados ao algoritmo *"Back-Propagation"* tradicional, para verificação de sua eficácia.

1 Introdução

Evoluindo a partir da disciplina "Inteligência Artificial", um conjunto de métodos de computação inspirados em mecanismos da natureza acabou por delinear uma nova área de estudos batizada como "Inteligência Computacional". Estes mecanismos, ou seja, os Algoritmos Genéticos (GAs) [2,3,4], as Redes Neurais (RN) [1,8,11], e a Lógica Fuzzy (FL)[1], apesar de se apresentarem inicialmente como tópicos de estudo isolados, acabaram por se integrar, constituindo o pilar de fundação onde a "Inteligência Computacional" acabou alicerçada.

Com o tempo, a utilização de mecanismos híbridos, combinando possíveis interações entre algoritmos genéticos e outros sistemas evolutivos, redes neurais e lógica fuzzy, passou a ser utilizada de modo a gerar mecanismos computacionais mais robustos e flexíveis. Exemplos de tais combinações podem ser encontrados, por exemplo, em [5,7,9,12,13]. Como áreas de investigação isoladas, tanto a lógica fuzzy como as redes neurais vem atraindo um grande número de pesquisadores, impulsionando seu desenvolvimento a um ritmo acelerado. Os algoritmos genéticos, entretanto, ainda carecem de uma maior base metodológica, que modifique seu status de ferramenta heurística para ferramenta formal. Apesar de uma técnica bem antiga, não existem critérios metodológicos que estabeleçam alguns parâmetros críticos para os GA's, tais como o tamanho da população, os mecanismos de seleção, e/ou os valores de taxas de crossover e mutação. Estes parâmetros acabam sendo levantados de maneira empírica, muitas vezes comprometendo o desempenho do GA. Neste trabalho, pretendemos trazer uma pequena contribuição neste sentido, tentando analisar o efeito na variação de alguns parâmetros do GA e com isso verificar sua importância no desempenho diante de um problema de teste. Para tal, elaborou-se uma plataforma de teste consistindo de um problema bem simples que é proporcionar o aprendizado a uma rede neural, de forma que esta passe a aproximar uma função previamente determinada.

Veremos que apesar de um problema simples, ele se mostra adequado para a investigação que realizamos, permitindo a coleta de subsídios importantes na determinação da importância dos diferentes parâmetros no desempenho de um algoritmo genético.

Como a aprendizagem de redes neurais também pode ser implementada por meio de métodos indutivos tradicionais, apresentamos também, a título de comparação, os resultados do mesmo problema quando solucionado pelo algoritmo *Back-Propagation* [8,11] simples. Apesar de existirem algoritmos indutivos mais eficientes que o *Back-Propagation*, sua implementação se dá aqui somente com propósitos ilustrativos, sem a pretensão de se proporcionar uma comparação exaustiva entre métodos indutivos e evolutivos.

2 Algoritmos Genéticos

Os Algoritmos Genéticos (GAs), correspondem a uma família de métodos adaptáveis que podem ser utilizados para solucionar problemas de busca e otimização. Apesar de não existir um algoritmo único ou padrão, todos os algoritmos genéticos comungam de uma mesma base funcional, caracterizada da seguinte maneira:

- codificação uma possível solução ao problema de otimização por meio de uma sequência de parâmetros, também chamada de cromossomo ou indivíduo
- avaliação não somente de um único indivíduo (ou uma única solução), mas de toda uma população de indivíduos simultaneamente

- população inicial (geralmente aleatória) é utilizada para gerar populações subsequentes por meio de operações entre seus indivíduos e mecanismos de seleção que a cada iteração selecionam os indivíduos mais aptos, que passam a integrar as novas populações.

2.1Princípios Básicos do GA utilizado

O algoritmo genético clássico, introduzido por Holland possui uma série de limitações, tais como a necessidade de se utilizar uma codificação binária, o uso do crossover simples (que torna a codificação uma atividade crítica) e a mutação simples aplicada a indivíduos aleatórios. Diversas metodologias foram desenvolvidas de modo a superar as limitações originais do algoritmo genético clássico, tais como o uso do crossover uniforme e metodologias de seleção alternativas ao clássico *Roulette Wheel* [4], tais como a seleção por diversidade, seleção bi-classista, e outras. O algoritmo genético utilizado neste experimento é na verdade uma abstração de todas estas metodologias, constituindo-se na verdade de um meta-algoritmo razoavelmente sofisticado, permitindo que uma vasta gama de operadores de transformação e seleção sejam utilizados. O algoritmo final acaba sendo função de diversos parâmetros que são arbitrados pelo projetista do GA. Se por um lado esta atitude traz flexibilidade, também aumenta a responsabilidade na determinação dos paâmetros mais adequados a uma determinada aplicação em particular. O funcionamento básico do algoritmo modificado utilizado se encontra na figura 1 abaixo.

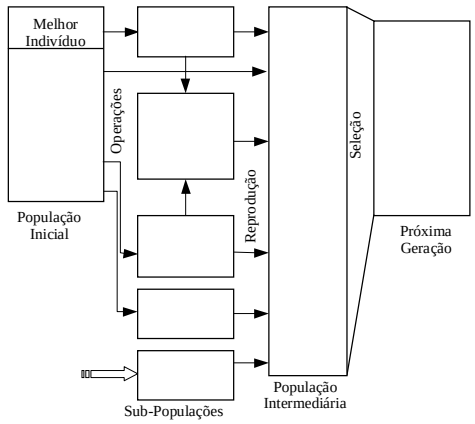

Figura 1 Princípio de Funcionamento de um GA Modificado.

A partir de uma população inicial, que é avaliada segundo uma função de fitness, de onde se destaca um melhor indivíduo, são geradas diversas sub-populações. Cada sub-população emprega uma única metodologia, que determina basicamente como indivíduos são escolhidos a partir da população inicial, e quais as operações que são empregadas entre eles. Exemplos típicos de operações são o crossover e mutação, com todas suas variações. Algumas populações podem escolher indivíduos não na população inicial, mas entre outras sub-populações. Pode-se ainda gerar sub-populações totalmente aleatoriamente. Todas as sub-populações são então agregadas em uma população intermediária, e seus indivíduos re-avaliados em relação à função de fitness. Por fim, utiliza-se um critério de seleção para compor uma nova geração, que será a população inicial de um novo ciclo como este.

A vantagem deste mecanismo baseado em sub-populações é que diferentes métodos evolutivos podem ser utilizados em paralelo, e dosados de modo a compor o mecanismo evolutivo final da população. Assim, pode-se por exemplo compor sub-populações baseadas somente no melhor indivíduo da população original, bem como outras baseadas em outros critérios como diversidade e espalhamento no espaço de soluções. Pode-se utilizar os mesmos mecanismos, com diferentes taxas em paralelo. Por exemplo, pode-se compor uma sub-população com 50% de crossover uniforme e outra com apenas 1%. Pode-se desta maneira, executar uma suite de operadores que pode ser mais ou menos sofisticada, atendendo desta maneira à necessidade das aplicações, que podem ser mais simples e/ou mais complexas. O tamanho de cada sub-população também é arbitrário, o que flexibiliza ainda mais sua aplicação.

3 O Problema Utilizado

Uma rede neural artificial é um sistema de processamento de informações inspirado nas redes neurais biológicas, tendo sido desenvolvido como uma generalização do modelo matemático para neurônios biológicos, sendo utilizadas, dentre outras coisas, para solucionar problemas de reconhecimento de padrões, aproximação e interpolação de funções. Tais problemas, quando solucionados via redes neurais acabam se transformando em um problema de aprendizagem, onde o que se deseja é obter os parâmetros de uma rede neural que solucionam o problema. O processo de obtenção destes parâmetros é chamado do aprendizado de uma rede neural.

Existem diversas arquiteturas que podem ser utilizadas para a resolução de um problema em particular. Para este trabalho, foi utilizada a rede neural *Multi-Layer Perceptron* [11], que são redes do tipo *Feed-Forward* com uma ou mais camadas intermediárias de neurônios entre os neurônios de entrada e de saída.

O problema que se propôs foi a aproximação da seguinte função:

$$f(x, y, z) = e^{-z} \cdot (sen(x + y))^2$$

no intervalo:

$$-10\pi \le x \le 10\pi$$
$$-10\pi \le y \le 10\pi$$
$$-1 \le z \le 1$$

A topologia de rede escolhida apresenta uma camada de entrada com três neurônio, uma camada de saída com um neurônio e duas camadas intermediárias, cada uma com cinco neurônios, conforme a figura 2.

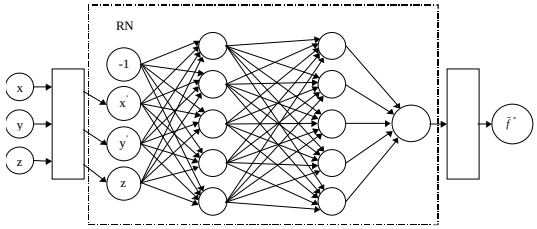

Figura 2 - Estrutura da Rede Neural utilizada.

3.1 Aprendizagem da RN utilizando o algoritmo Back-Propagation

Inicialmente, utilizou-se o algoritmo Back-Propagation, de modo a criar uma referência para o desempenho do algoritmo genético a ser implementado posteriormente. Utilizaram-se 1000 amostras de dados distribuídas dentro do intervalo de interesse, associadas ao valor correspondente que deveriam gerar pela função a ser aproximada e o algoritmo *Back-Propagation* para atualizar os pesos da rede. O algoritmo tem como condição de parada 1000 iterações. Este valor foi escolhido depois de alguns ensaios onde observou-se que depois disso, a rede mantém um valor praticamente fixo para os erros obtidos. Aplicou-se o método de treinamento por lotes, obtendo-se os erros quadrático médio, erro médio e erro máximo.

Foram feitos alguns ensaios de treinamento, variando-se alguns parâmetros do algoritmo *Back-Propagation*. No primeiro ensaio, alterou-se os valores do ganho (0.01, 0.1 e 0.8), sem a utilização de um valor de momento. O melhor comportamento deu-se para um ganho de 0.1. No segundo ensaio, tomou-se um ganho de η=0.1 e um valor de momento α=0.0001. Os resultados obtidos por este ensaio são comparados com o melhor comportamento do ensaio anterior, sem valor de momento, observando-se que com a introdução do momento, a rede converge mais rapidamente.

No terceiro ensaio, adotou-se um ganho η=0.1 e um valor de momento variável, que vai diminuindo com as iterações, através da equação: $y = k\ell^{-ct}$ onde, k é uma constante (0.1, 0.5), c = 0.9, e t é o número da iteração. O melhor desempenho foi obtido para um k=0.5.

Os resultados dos ensaios podem ser vistos na figura 3 a seguir:

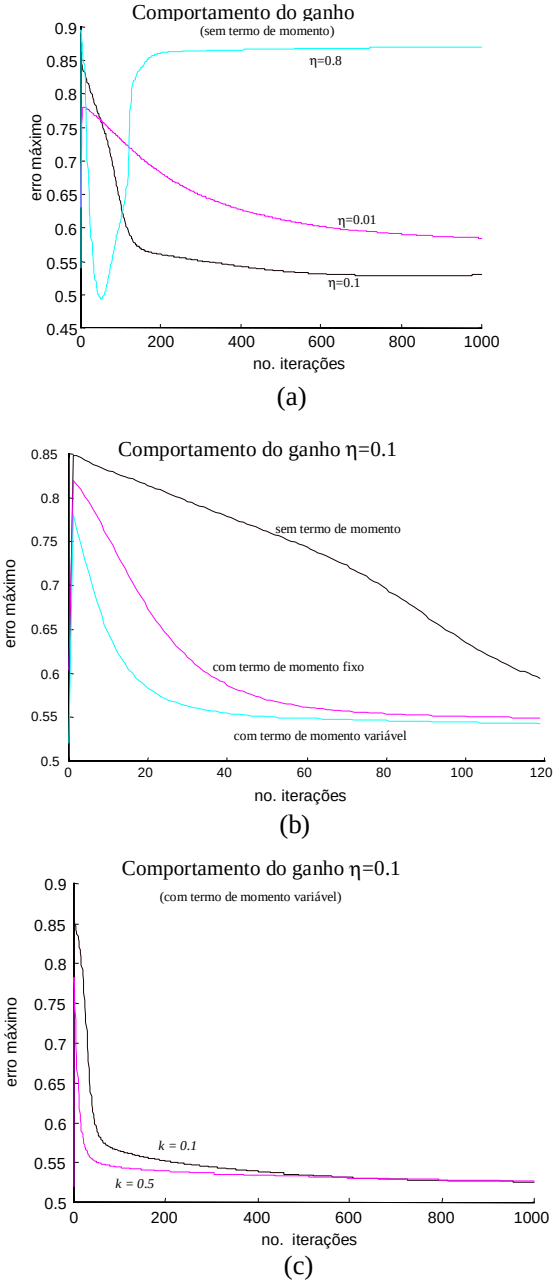

Figura 3 - Erros na Aprendizagem via Back-Propagation

3.2 Aprendizagem da RN utilizando um GA

Utilizando o GA, adotou-se como condição de parada 40 gerações, ou seja, 40 iterações do algoritmo genético. Este valor foi escolhido depois de alguns ensaios. Adotou-se a codificação binária devido a

particularidades do problema, onde cada peso da rede neural podia estar situado no intervalo real [-1,1].

A função de *fitness* adotada foi escolhida de modo a representar uma composição entre o erro quadrático médio e o erro máximo, dado pela seguinte equação:

$$ff = \alpha \cdot Eqm + (1-\alpha) \cdot E\,max$$

O objetivo do GA é minimizar a função ff, desta forma encontrando os parâmetros da rede neural que melhor aproxima a função escolhida.

Para aplicar o GA no treinamento da rede neural, efetuaram-se diversos experimentos, de modo a mapear a sensibilidade do GA aos parâmetros adotados.

Para definir o tamanho da população com melhor desempenho, escolheu-se diferentes amostras de tamanho da população, colhendo-se os resultados obtidos. Como se pode observar nos gráficos da figura 4, a importância do tamanho da população é um fator crítico para o desempenho do GA. Com populações pequenas, da ordem de 500 a 1000 indivíduos, o desempenho da rede é fraco, e mesmo utilizando-se o GA, não se foge dos mínimos locais. A medida que aumenta-se o tamanho da população, vai-se encontrando mínimos locais cada vez mais próximos do mínimo global, até que para um dado valor de população (3000 indivíduos) o mínimo global é atingido.

Utilizou-se o algoritmo genético modificado da figura 1, utilizando-se diferentes operadores genéticos para a geração das sub-populações. Uma primeira sub-população foi formada apenas por reprodução, ou seja, uma das sub-populações é uma cópia da população original. Para uma segunda sub-população, adotou-se o crossover uniforme onde os pares de indivíduos a realizar o crossover eram escolhidos na população inicial utilizando-se o *RouletteWheel* [4]. Utilizando-se o operador de mutação, diversos experimentos foram realizados, gerando diferentes conjuntos de sub-populações. Foram realizados experimentos utilizando-se 2, 4 e 6 sub-populações geradas por mutação. Para os experimentos com 2 sub-populações de origem por mutação, mutou-se 10% dos genes (escolhidos aleatoriamente) do melhor indivíduo da população. Na primeira das sub-populações, esta mutação alterou em ±1% o valor dos genes mutados, a partir de seus valores originais. Na segunda sub-população, alterou-se em ±30% o valor dos genes mutados.

Em um segundo experimento, utilizou-se 4 sub-populações originadas por mutação do melhor indivíduo. As duas primeiras foram equivalentes às do experimento anterior, e as novas realizaram a mutação de 50% dos genes do melhor indivíduo, também alterando-se ±1% e ±30% do valor original.

Por fim, em um terceiro experimento, utilizou-se 6 sub-populações geradas por mutação do melhor indivíduo. Além das 4 anteriores realizou-se dois ensaios da mutação de 100% dos genes do melhor indivíduo, no primeiro caso alterando-se seu valor de ±1% e no segundo caso ±30%.

O comportamento do GA para esses conjuntos de sub-populações pode ser visualizado na figura 5.

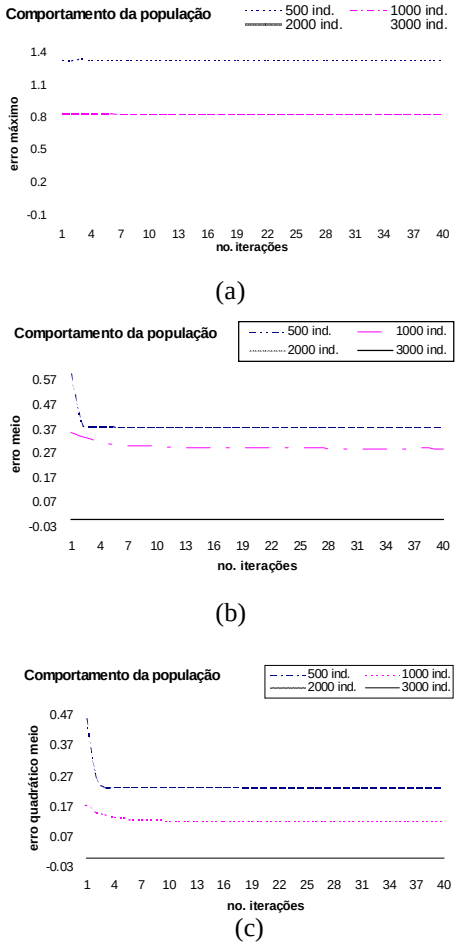

Figura 4 - Comportamento do Erro em função do tamanho da população .

Como se pode observar a partir dos gráficos da figura 5, os melhores resultados se obtiveram com a aplicação das 6 sub-populações geradas a partir da mutação do melhor indivíduo. A mutação deste mesmo indivíduo, de diferentes maneiras, faz com que, em diferentes níveis de granularidade, melhores resultados sejam descobertos, acelerando a convergência a mínimos locais, o que não é possível em Gas tradicionais.

Para escolher o método de seleção que leva a uma nova população, parâmetro importante de um GA, testou-se 4 diferentes estratégias de seleção, sumarizadas na figura 6.

Seleção elitista: são escolhidos os n melhores indivíduos da população intermediária.

Seleção bi-classista 10-90: são escolhidos 10% dos melhores indivíduos e 90% dos piores indivíduos

Seleção bi-classista 30-70: são escolhidos 30% dos melhores indivíduos e 70% dos piores indivíduos

Seleção bi-classista 50-50: são escolhidos 50% dos melhores indivíduos e 50% dos piores indivíduos

A lógica por trás da seleção bi-classista é a de tentar preservar a diversidade da população como um patrimônio genético. Ou seja, a idéia de que eventualmente a partir do material genético de indivíduos ruins, pode haver a possibilidade de combinações que possam gerar indivíduos melhores. Nesse caso, evitaria-se que a população como um todo convergisse para um mínimo local.

Na figura 6, observa-se que esta lógica tem um fundo de realidade, pois os melhores resultados foram obtidos com a seleção bi-classista 50-50.

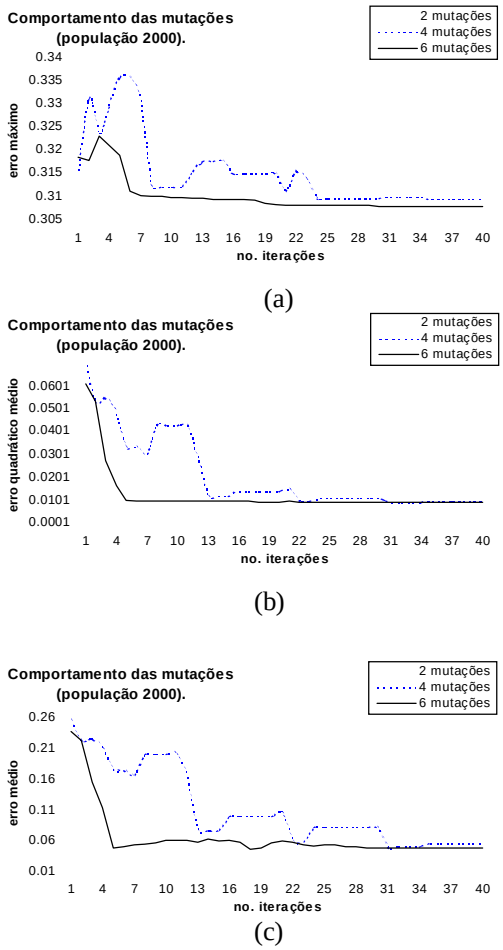

Figura 5 Comportamento dos Erros em virtude de diferentes sub-populações geradas por mutação

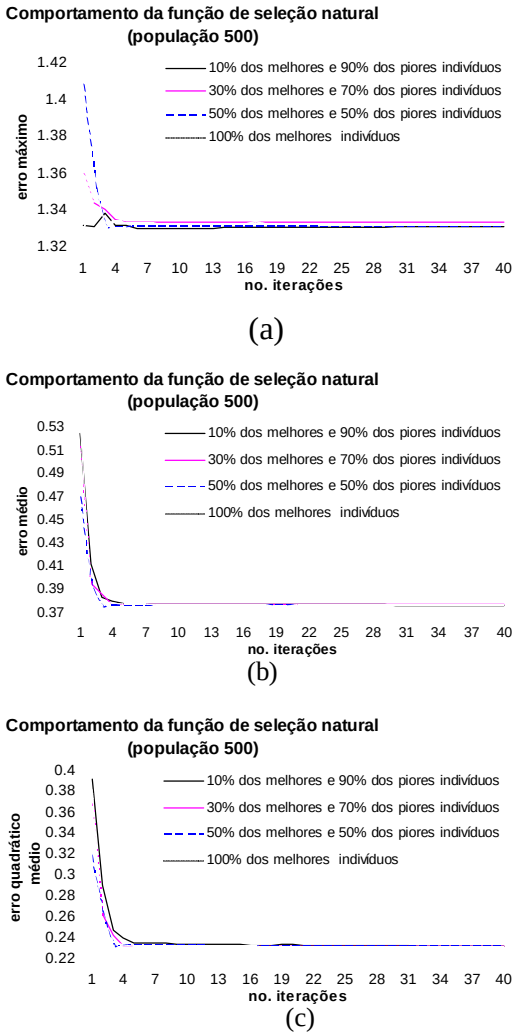

Figura 6 Comportamento do Erro em função das estratégias de seleção adotadas

3.3 Comparação dos métodos de aprendizagem utilizados para a Rede Neural

Para uma análise da efetividade da aplicação do GA ao problema do aprendizado de uma rede neural, fazemos uma comparação dos resultados obtidos com aqueles obtidos a partir do *Back-Propagation*. Vemos que para valores de população abaixo de 2000 indivíduos, o algoritmo *Back-Propagation* continua sendo mais efetivo para o aprendizado da rede. Somente com uma população de 3000 indivíduos o algoritmo genético pôde superar o desempenho do *Back-Propagation.* Em termos de recursos computacionais, o GA exige muito mais que o *Back-Propagation,* tanto levando-se em consideração o consumo de memória como o tempo de processamento. Entretanto, para aplicações onde se tenha tempo disponível, o GA acaba sendo mais efetivo, pois efetivamente consegue escapar dos mínimos locais,

atingindo o mínimo global. Entretanto, apesar da tradição que leva neste sentido, o GA também está sujeito ao problema dos mínimos locais, somente conseguindo escapar destes quando um tamanho de população adequado é utilizado. Os diferentes esforços que se fizeram no sentido de incrementar o desempenho somente alterando-se outros parâmetros tais como a taxa de mutação, taxa de crossover, e gerando outros tipos de sub-populações, sem alterar o tamanho da população, mostraram-se infrutíferos. Ou seja, o tamanho da população pode ser considerado como um dos fatores mais críticos de um GA, demandando um especial cuidado quando de sua aplicação em outros problemas de engenharia.

Conclusões

Por meio dos experimentos aqui conduzidos, pudemos avaliar a sensibilidade de um algoritmo genético, diante de variações em seus parâmetros e compará-lo a um método de aprendizagem clássico em redes neurais que é o *Back-Propagation.*

Apesar de modestas, as conclusões do presente estudo são muito importantes, pois nos dão uma orientação na condução de projetos baseados em algoritmos genéticos a serem desenvolvidos no futuro. Pudemos verificar a extrema importância do parâmetro "tamanho da população" na efetividade do GA em escapar aos mínimos locais. Sendo assim, quando do desenvolvimento de soluções baseadas em algoritmos genéticos a problemas que envolvam o aparecimento de mínimos locais (à semelhança do aprendizado de redes neurais), devemos tomar um cuidado especial quanto à sintonia do tamanho da população, antes de sentenciar que os algoritmos genéticos não estão dando resultados efetivos. Ao invés de investir em novos operadores e modificações nos valores das taxas de mutação e crossover, devemos investir no aumento do tamanho da população de modo a resultar um incremento em desempenho.

Referências

[1.] B. Kosko, “Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence”, Prentice-Hall International, 1992.

[2.] D. Beasley, D.R. Bull, R.R. Martin. “An Overview of Genetic Algorithms: Part 1, Fundamentals” – University Computing 15(2), 1993 58-69.

[3.] D. Beasley, D.R. Bull, R.R. Martin. “An Overview of Genetic Algorithms: Part 2, Research Topics” – University Computing 15(4), 1993 170-181.

[4.] D.E. Goldberg. “Genetic Algorithms in search, optimization and machine learning.”, Addison-Wesley, 1989.

[5.] H. Saito, N. Kobayashi. “Evolutionary Computation Approaches to Halftoning Algorithm” – Proccedings of the First IEEE Conference on Evolutionary Computation, Vol II (1994) 787-791.

[6.] J. Arabas, Z. Michalewics, J. Mulawka. “GAVaPS- a Genetic Algorithm with Varying Population Size” - Proccedings of the First IEEE Conference on Evolutionary Computation, Vol I (1994) 73-78.

[7.] J. Renders, S. P. Flasse. “Hybric Methods using Genetic Algoritms for Global Optimization” – IEEE Tran-sactions on Systems, Man and Cybernetics-Part B: Cybernetics, Vol. 26, No. 2, April 1996, 243-258.

[8.] L. Fausett. “Fundamentals of Neural Networks, architectures, algorithms and applications.”, Prentice Hall Interna-tional, 1994.

[9.] L. Tsinas, B. Dachwald. “A Combined Neural and Genetic Learning Algoritm” - Proccedings of the First IEEE Conference on Evolutionary Computation, Vol II (1994) 770-774.

[10.] R.R. Gudwin. “Contribuições ao estudo matemático de sistemas inteligentes”, Tese de Doutorado, UNICAMP, Campinas 1996.

[11.] R. P. Lippmann. “An Introduction to Computing with Neural Nets”, IEEE ASSP Magazine, April, 1987.

[12.] S. Bengio, Y. Bengio, J. Cloutier. “Use of Genetic Programming for the Search of a New Learning Rule for Neural Networks” - Proccedings of the First IEEE Conference on Evolutionary Computation, Vol I (1994) 324-327.

[13.] S. G. Romaniuk. “Applying Crossover Operators to Automatic Neural Network Construction” – Proccedings of the First IEEE Conference on Evolutionary Computation, Vol II (1994) 750-752a.