Evoluindo os pesos de uma Rede Neural com Algoritmos Genéticos

Aurora Trinidad R. Pozo[1], **Davi Azevedo Q. Santos**[1], **Derik Evangelista R. Silva**[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19081 – 81531-980 – Curitiba – PR – Brasil

{aurora, daqsantos, dersilva}@inf.ufpr.br

Abstract. *This meta-paper describes the style to be used in articles and short papers for SBC conferences. For papers in English, you should add just an abstract while for the papers in Portuguese, we also ask for an abstract in Portuguese ("resumo"). In both cases, abstracts should not have more than 10 lines and must be in the first page of the paper.*

Resumo. *Este meta-artigo descreve o estilo a ser usado na confeco de artigos e resumos de artigos para publicao nos anais das conferncias organizadas pela SBC. solicitada a escrita de resumo e abstract apenas para os artigos escritos em portugus. Artigos em ingls devero apresentar apenas abstract. Nos dois casos, o autor deve tomar cuidado para que o resumo (e o abstract) no ultrapassem 10 linhas cada, sendo que ambos devem estar na primeira pgina do artigo.*

1. Introdução

Contextualização do trabalho:

- Contexto histórico de IA/Aprendizado de máquina
- Motivação
- Objetivos
- Linha do trabalho

2. Metodologia

- Desenvolvimento do trabalho
- Linguagem utilizada
- Tratamento das bases

2.1. Redes Neurais

Segundo [Kasabov 1996] uma rede neural artifical é um modelo computacional biologicamente inspirado o qual consiste de elementos de processamento (neurônios) e conexões entre eles (pesos) que representam a memória do sistema. O primeiro modelo de neurônio artificial foi proposto por McCulloch e Pitts em 1943.

Uma rede neural é definida por quatro parâmetros:

1. Tipo de neurônio: determina as entradas e saídas de um neuronio;
2. Arquitetura conexionista: determina a arquitetura da rede.

3. Algoritmo de aprendizado: determina como o conhecimento é armazenado na rede.
4. Algoritmo de Recall: determina como o conhecimento da rede é recuperado.

Os vários modelos de redes neurais artificais podem ser descritos em termos destes parâmetros. O modelo mais comum é o Multi Layer Perceptron (MLP), que consiste de uma arquitetura totalmente conectada entre as camadas, possui no mínimo três camadas (entrada, escondida e saída), cada neurônio tem um entrada fixa (bias), e a função de ativação mais comum é a sigmóide. O MLP tornou-se bastante utilizado com surgimento do algoritmo Backprogation como algoritmo de aprendizado. Contudo neste trabalho, será utilizado um algoritmo genético para realizar o treinamento da rede neural.

Neste trabalho utilizaremos o MLP para o problema de classificação. Segundo [Kasabov 1996] O problema de classificacão é associar um objeto a um grupo ou classe de objetos já existentes.

2.2. Algoritmos Genéticos

O Algoritmo Genético (AG), geralmente referenciado como *algoritmos genéticos*, foi desenvolvido por John Holland na Universidade de Michigan, em 1975, em um dos livros mais famosos da área, *Adaptation in Natural and Artificial Systems*, publicado pela editora *University of Michigan Press* [Luke 2009b].

Os AGs são algoritmos de busca heurística que tentam reproduzir artificialmente o processo de evolução, da Teoria de Evolução das Espécies, de Darwin. São muito similares a outros algoritmos de busca local, como o Subida de Encosta (*Hill-climbing*), diferindo apenas na quantidade de soluções mantidas a cada iteração e no processo de geração de novas soluções.

De acordo com [Montana and Davis 1989], os AGs necessitam de cinco componentes:

C1 - Uma maneira de codificar uma solução em um indivíduo.
C2 - Uma função de avaliação que retorna um índice de qualidade para cada individuo da população.
C3 - Um procedimento de inicialização da população.
C4 - Operadores que podem ser aplicados nos indivíduos pais no processo de reprodução, alterado a composição genética dos novos indivíduos gerados. Neste componente incluem-se os operadores de mutação, de cruzamento e outros específicos do domínio.
C5 - Uma configuração de parâmetros do algoritmo, os operadores, etc.

Com estes cinco componentes, ainda de acordo com [Montana and Davis 1989], o AG funciona seguindo os seguintes passos:

1. A população é iniciada usando-se o procedimento C3. O resultado é um conjunto de indivíduos, de acordo com C1.
2. Cada indivíduo é avaliado, usando a função definida em C2.
3. A população se reproduz até que um critério de parada seja atingido. A reprodução é realizada seguindo-se os seguintes passos:
   (a) Um ou mais indivíduis são escolhidos para a reprodução. A seleção é estocástica, mas os pais melhores avaliados são favorecidos na escolha. Os parametros em C5 podem influenciar neste processo de seleção.

(b) Os operadores de C4 são aplicados aos pais para geração dos filhos. Os parametros em C5 ajudam a determinar quais operadores serão usados.
(c) Os filhos são então avaliados e inseridos de volta na população. Em algumas versões de AG, toda a população é substituida. Em outras, apenas um subconjunto é substituido.

A cada iteração, os indivíduos melhor avaliados tem mais chances de serem escolhidos para reprodução, fazendo com que, em teoria, sejam gerados melhores indivíduos a cada geração. Ao término do algoritmo, senão a ótima, a tendência é ter-se uma solução muito próxima a ela.

2.3. Bases de dados

As bases de dados utilizadas foram retiradas de [Frank and Asuncion 2010]. Para o experimento foram utilizadas as seguintes bases:

1. *Breast Cancer Wisconsin (Original) Data Set*, doravante denominada *Câncer*;
2. *Pima Indians Diabetes Data Set*;
3. *Glass Identification Data Set*;
4. Statlog (Heart) Data Set, doravante denominada *Heart*;
5. *Iris Data Set*.

3. Implementação

A linguagem Java foi utilizada para implementar os algoritmos. A rede neural foi codificada como um vetor de números em ponto flutuante, também conhecido como *real-coded* [Liu et al. 2004]. A figura 1 ilustra esse processo, que foi baseado em [Montana and Davis 1989]. Desta forma, é possível determinar rapidamente quais são os pesos de entrada e saída de cada neurônio e assim facilitar a implementação dos operadores genéticos. O número de camadas da rede é fixo e consiste em uma camada de entrada, uma camada oculta e uma camada de saída. Na tabela 1, temos o número de neurônios de cada camada para cada base de dados a ser testada.

Base	Atributos (Entradas)	Classes (Saídas)	Escondidos
Cancer	9	2	5
Pima Indians Diabetes	8	2	10
Glass Identification	9	7	10
Heart	13	2	5
Iris	4	3	10

Tabela 1. Número de neurônios de cada camada

O algoritmo genético foi implementado conforme [Luke 2009a]. O *fitness* de cada indivíduo é dado pelo Erro Quadrado Médio (EQM), ou seja, um indivíduo mais apto é aquele que possui o menor valor. O EQM é calculado de acordo com [Liu et al. 2004].

Os operadores genéticos utilizados foram:

1. Operadores de Mutação, segundo [Montana and Davis 1989] e [Liu et al. 2004]:
   (a) Biased Mutate Weights: O valor de um gene pode ser substituído por um outro qualquer da distribuição de probabilidade inicial (distribuição normal com média 0 e desvio padrão 5).

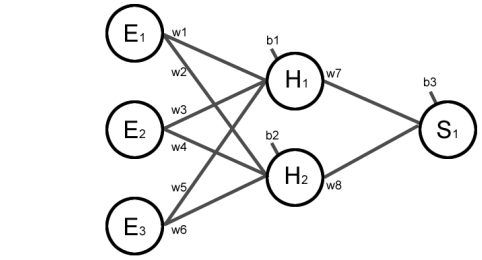

Figura 1. Codificacão da rede

(b) Unbiased Mutate Weights: Um valor da distribuição de probabilidade inicial é acrescido a um gene.
(c) Mutate Nodes: Este operador seleciona n neurônios das camadas oculta e de saída e adiciona um valor da distribuição de probabilidade inicial a cada peso de entrada dos neurônios. Nos nossos o valor de n foi 2.
(d) Mutate Weakest Nodes: Este operador seleciona o neurônio mais fraco (aquele que menos contribui para o saída da rede) e aplica uma mutação *unbiased* ou *biased* em cada peso do neurônio mais fraco.
(e) SinglePointRandom: Cada gene do cromossomo é substituído por um valor aleatório entre [-50, 50].
(f) NonUniform: implementado conforme [Michalewicz 1994]. Na nossa implementacão o parâmetro b é 4.

2. Operadores de Crossover, segundo [Montana and Davis 1989] e [Liu et al. 2004]:
   (a) Crossover Weights: O valor gene de um indivíduo filho é escolhido pela seleção aleatória do mesmo gene de um dos pais.
   (b) Crossover Nodes: Para cada neurônio de um dos pais escolhidos aleatoriamente os pesos associados são passados diretamente para o filho.
   (c) Crossover Features: Para cada neurônio da rede do primeiro pai, o algoritmo reorganiza o segundo pai de forma que os neurônios que desempenham o mesmo papel (isto é, possum a mesma saída para uma determinada entrada) fiquem na posição, assim formando um pai intermediário. Depois disso é aplicado o operador Crossover-Nodes entre o primeiro pai e o pai intermediário.
   (d) Line Recombination: A implementação foi um hibrido entre [Liu et al. 2004] e [Luke 2009a] onde o parâmetro é dado por uma distribuição normal com média 0 e desvio padrão 5.

Os parâmetros utilizados para o Algoritmo Genético foram os seguintes:

1. Tamanho da População: 50 e 100 indivíduos;
2. Número de Gerações: 500;
3. Tamanho do torneio: 2 e 4;
4. Número de indivíduos da elite: 10;
5. Probabilidade de Crossover: 100%;
6. Probabilidade de mutação: 10%.

Base	Treinamento	Validação	Teste	Total
Cancer	350	175	174	699
Pima Indians Diabetes	252	258	258	768
Glass Identification	114	50	50	214
Heart	130	70	70	270
Iris	70	40	40	150

Tabela 2. Separação dos conjuntos de dados

Para a análise dos resultados foi utilizado a validação cruazada [Haykin 1998]. Na validação cruzada o conjunto de dados é particionado aleatoriamente em um conjunto de teste e um conjunto de treinamento, que ainda é divido em um subconjunto de treinamento e um subconjunto de validação. O conjunto de dados foi separado de acordo com a tabela 2.

4. Resultados obtidos

Foram realizadas 10 execuções para cada combinação possível de:

1. Parâmetros do algoritmo genético (explicitados na seção 3);
2. Operadores de mutação;
3. Operadores de cruzamento;

Para os operadores de mutação e cruzamento, considerou-se também um operador extra, que seleciona aleatoriamente qualquer um dos operadores descritos.

4.1. Influência dos parâmetros do AG

Um tamanho de torneio maior, mostrou, de uma maneira geral, ter um impacto positivo na média dos valores de fitness. Entretanto, como podemos ver nas figuras 2 e 3, a média piora em alguns casos e, nos casos em que há melhora, ela é muito pequena, como podemos ver nas figuras 4 e 5.

Já um maior tamanho da população apresenta uma influência bem mais significativa que o tamanho do torneito em praticamente todas as execuções. Podemos notar essa diferença analisando as figuras 6 e 7. Entretanto, ainda assim, a relação de consumo de recursos computacionais, associado ao maior tempo para execução e o ganho real de uma população maior pode não ser vantajoso.

4.2. Influência dos Operadores Genéticos

Diferentemente dos parâmetros em 4.1, a escolha certa dos operadores genéticos tem um impacto muito grande. Podemos observar facilmente este impacto analisando a figura 8. Podemos notar uma grande amplitude entre a pior combinação de operadores, que termina com a média de fitness um pouco maior que 0.15 e a melhor combinação, que termina com um valor de fitness menor que 0.05.

Também é interessante notar que em algumas casos, como o mostrado nas figuras 6 e 9, muitas combinações de operadores mantem os valores de fitness muito próximos entre si e, como visto na figura 7, há uma tendência a, em algum ponto, estabilizar, passando muitas gerações sem uma melhora sensível no valor. Estas características encontradas podem ser, em partes, remediadas aplicando-se, tal qual proposto

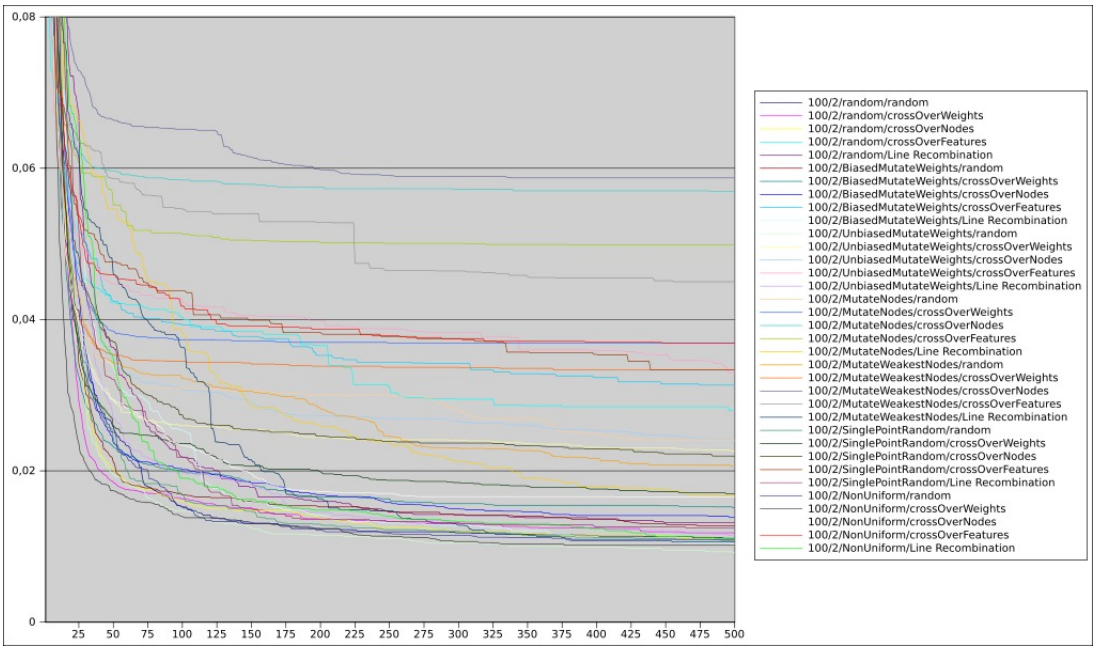

Figura 2. Fitness - Câncer; População 100, torneio 2;

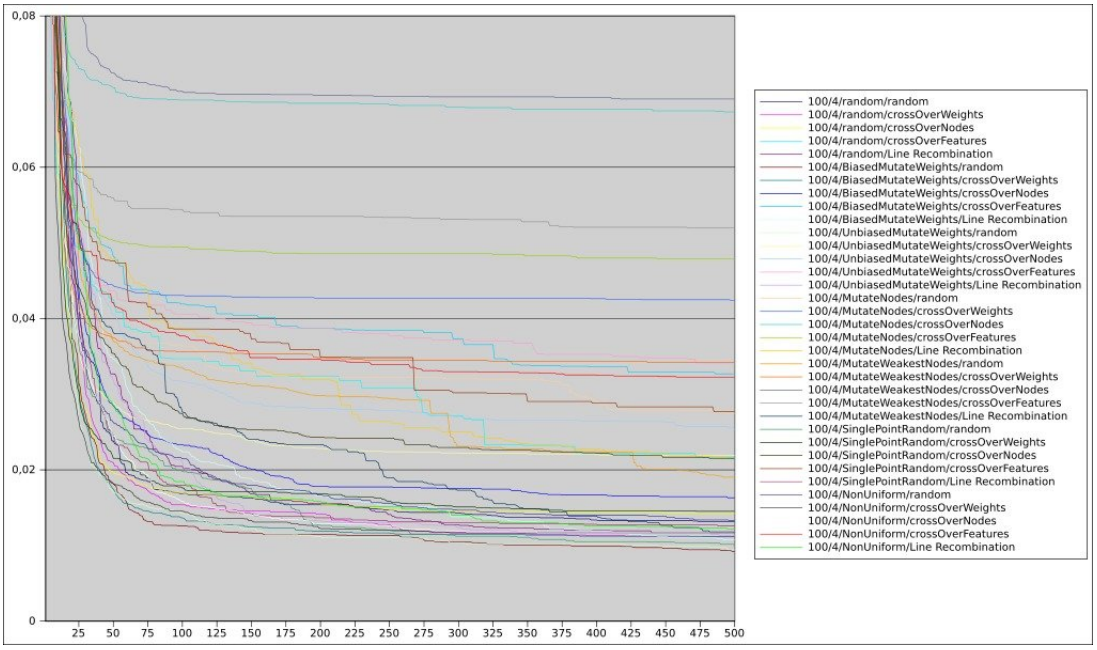

Figura 3. Fitness - Câncer; População 100, torneio 4;

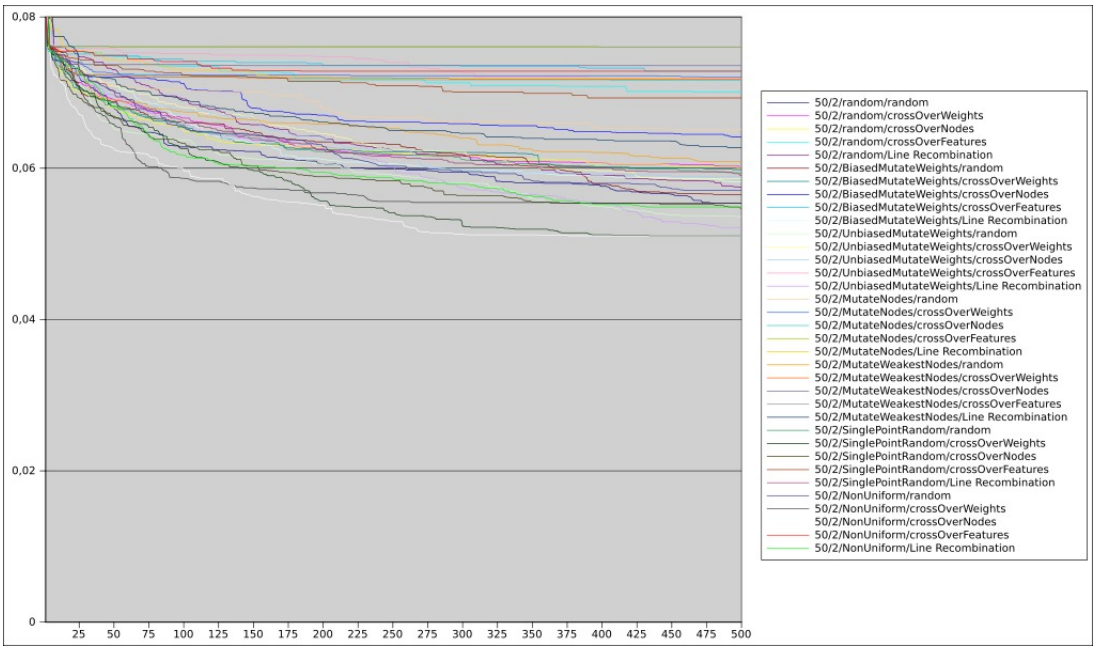

Figura 4. Fitness - Glass; População 50, torneio 2;

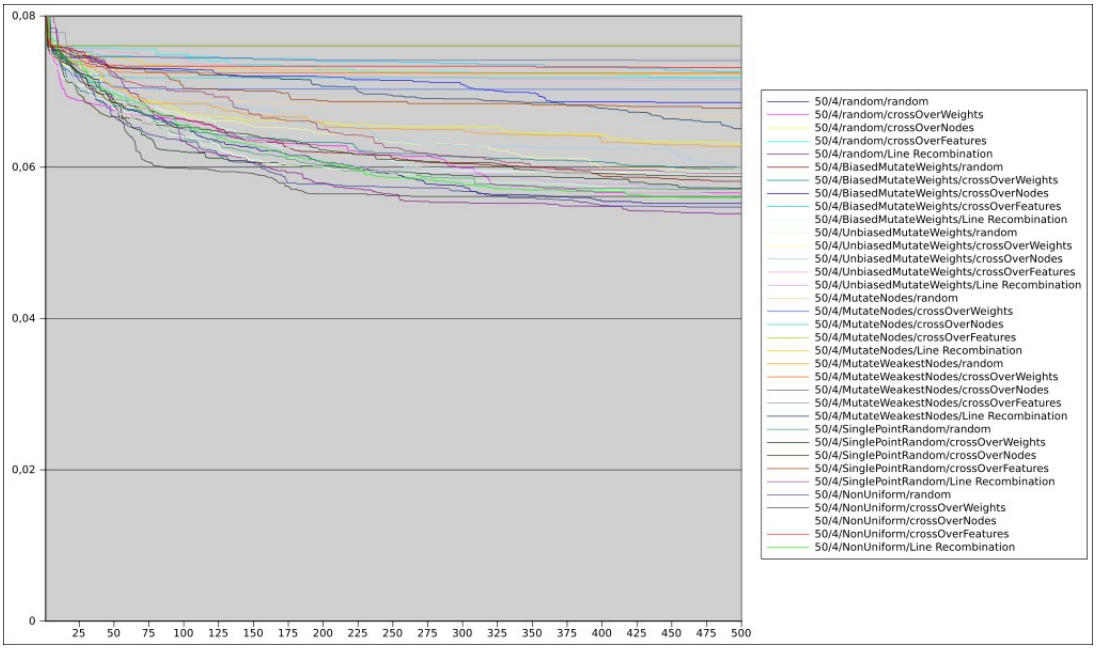

Figura 5. Fitness - Glass; População 50, torneio 4;

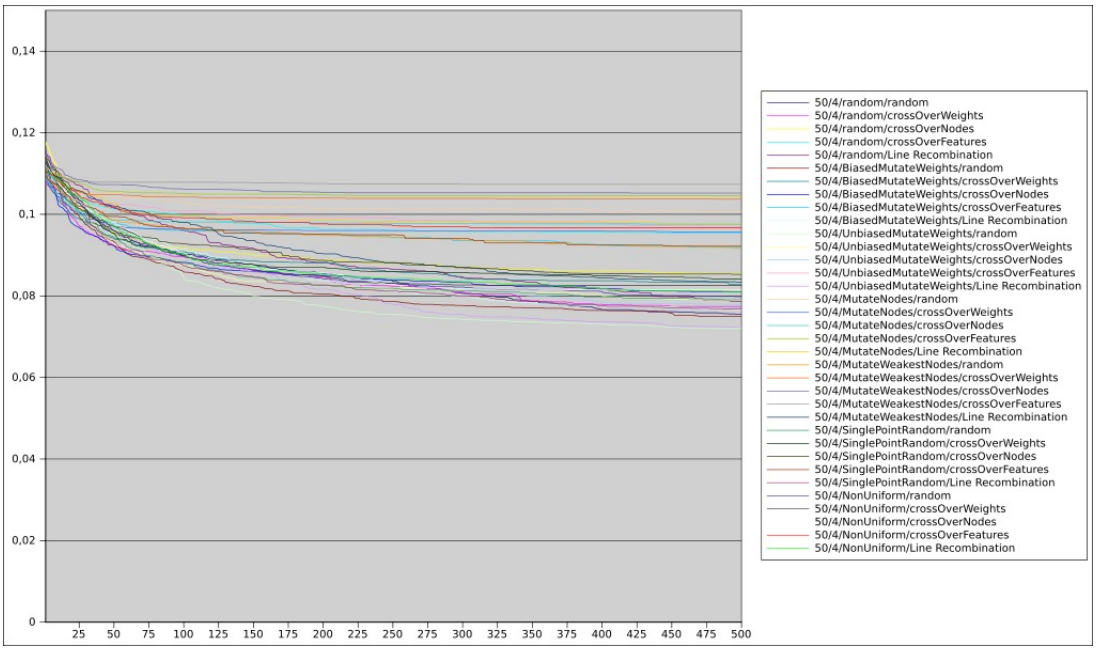

Figura 6. Fitness - Diabetes; População 50, torneio 4;

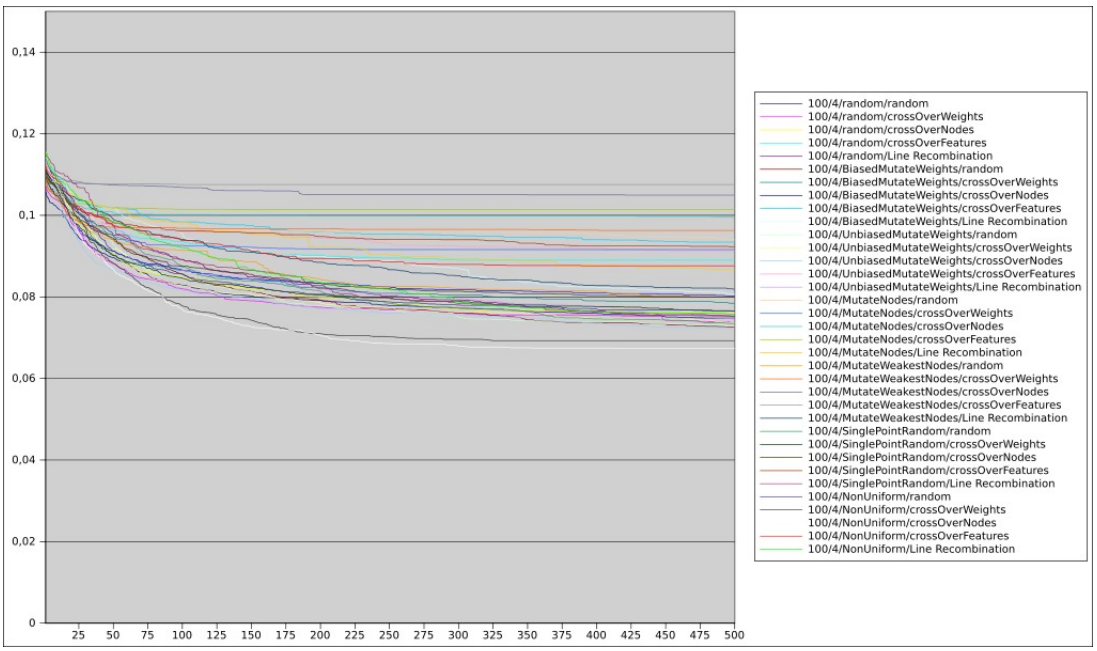

Figura 7. Fitness - Diabetes; População 100, torneio 4;

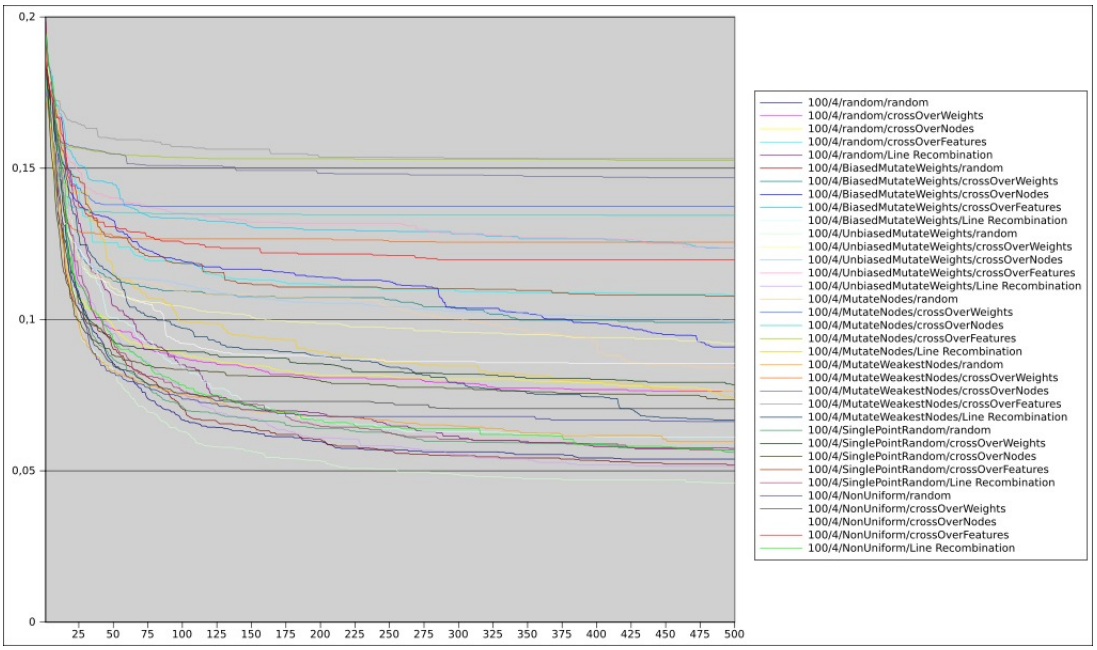

Figura 8. Fitness - Heart; População 100, torneio 4;

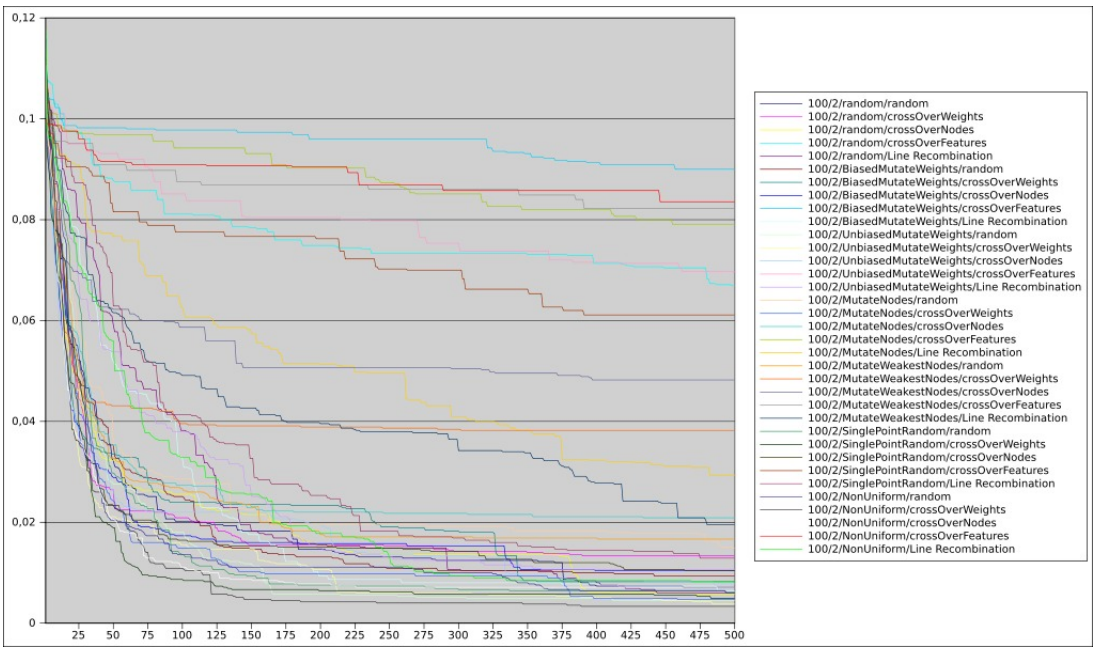

Figura 9. Fitness - Iris; População 100, torneio 2;

por [Montana and Davis 1989], uma espécie de torneio com os operadores, dando preferência para aqueles que estão produzindo um maior impacto na rede.

4.3. Taxas de Acerto

Ao término das execuções, conseguimos analisar qual a taxa de acerto média da rede neural com seus pesos evoluidos através de um algoritmo genético.

Na tabela 3, podemos visualizar as melhores taxas de acerto, tanto para o conjunto de testes, quanto para o conjunto de validação, assim como qual combinação de parâmetros e operadores maximizou o acerto. É interessante notar que apenas dois (dos cinco possíveis) operadores de cruzamento aparecem (Line Recombination e Aleatório). Já os operadores de mutação são mais heterogêneos. Podemos concluir, então, que o cruzamento tem um efeito maior na saída da rede que a mutação. Entretanto, se considerarmos que a mutação ocorre apenas em 10% das vezes, ao passo que o cruzamento ocorre sempre, este comportamento é esperado.

Também podemos perceber que, em alguns casos, a rede se comportou melhor com uma população e torneio menores, embora na maior parte das execuções uma grande população e um maior torneio tenham superado as outras combinações.

Ao compararmos com a média de acerto para estas bases, na tabela 4, à excessão da base Câncer, o desempenho da rede foi bem abaixo do esperado, principalmente nas bases Glass e Iris. Analisando as figuras 5 e 9, percebemos que o fitness para estas bases converge rapidamente para um mínimo local e estaguina, não conseguindo sair deste mínimo. Por esse motivo, os valores encontrados para estas bases fica tão distante do esperado.

	Parâmetros				Taxas	
Base	Pop	Tor	Mutação	Cruzamento	Teste	Validação
Câncer	100	2	Single Point Random	Aleatório	0.964	**0.954022**
Câncer	50	2	Aleatório	Line Recombination	**0.961142**	0.959770
Diabetes	100	4	Single Point Random	Line Recombination	0.695348	**0.690310**
Diabetes	100	2	Unbiased Mutate Weights	Line Recombination	**0.663953**	0.692635
Glass	100	4	NonUniform	Aleatório	0.414	**0.426**
Glass	100	4	NonUniform	Aleatório	**0.414**	0.426
Heart	50	2	Unbiased Mutate Weights	Line Recombination	0.795714	**0.774285**
Heart	100	4	Single Point Random	Line Recombination	**0.784285**	0.788571
Iris	100	4	NonUniform	Aleatório	0.414	**0.426**
Iris	100	4	NonUniform	Aleatório	**0.414**	0.426

Tabela 3. Melhores taxas de acerto

Base	Acerto esperado
Câncer	0.96
Diabetes	0.77
Glass	0.63
Heart	0.82
Iris	0.96

Tabela 4. Taxas de acerto esperado

Base	Teste	Validação
Câncer	0.914048	0.913756
Diabetes	0.619648	0.620949
Glass	0.1881	0.189014
Heart	0.694551	0.695867
Iris	0.1881	0.189014

Tabela 5. Média das taxas de acerto

5. Considerações Finais

Baseado na análise dos resultados obtidos, podemos notar que, apesar do algoritmo conseguir uma rápida queda nos valores de fitness em poucas gerações, o que aumenta as taxas de acerto da rede, a evolução dos pesos da rede sofre com o problema recorrente aos AGs, de otimização fina. Ao chegar em um certo valor de fitness, por mais que se alterem os códigos genéticos dos filhos, é muito difícil continuar a melhorar a rede, causando uma estagnação dos valores. É provavel que, mesmo que aumentemos o número de gerações, a melhora da rede seja ínfima.

Para evitar isto, se faz necessário a criação de novos (e melhores) operadores genéticos, que consigam impactar a rede de forma que os pesos continuem a evoluir. Além disso, desenvolver um mecanismo de evolução dos operadores, conforme descrito por [Montana and Davis 1989], de forma que os operadores sejam escolhidos de acordo com o impacto que este teve anteriormente na rede, pode melhorar ainda mais a evolução dos pesos. Isto pode ser feito em trabalhos futuros.

Referências

Frank, A. and Asuncion, A. (2010). UCI machine learning repository.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition.

Kasabov, N. K. (1996). *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. MIT Press, Cambridge, MA, USA, 1st edition.

Liu, Z., Liu, A., Wang, C., and Niu, Z. (2004). Evolving neural network using real coded genetic algorithm (ga) for multispectral image classification. *Future Gener. Comput. Syst.*, 20(7):1119–1129.

Luke, S. (2009a). *Essentials of Metaheuristics*. Lulu, 15th edition. Disponível em: <http://cs.gmu.edu/∼sean/book/metaheuristics/>.

Luke, S. (2009b). Populational methods. In Luke, S., editor, *Essentials of Metaheuristics*, pages 29–56. Lulu.

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., New York, NY, USA.

Montana, D. J. and Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1*, IJCAI'89, pages 762–767, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.