10th Brazilian Congress on Computational Intelligence (CBIC'2011), November 8 to 11, 2011, Fortaleza, Ceará Brazil

ALGORITMO GENÉTICO CELULAR PARA OTIMIZAÇÃO DE REDES NEURAIS MULTILAYER PERCEPTRON

Anderson Paulo da Silva 1, Teresa Bernarda Ludermir1

[1]Centro de Informática – Universidade Federal de Pernambuco - UFPE
{aps3, tbl}@cin.ufpe.br

Abstract – This paper presents an optimization method of artificial neural networks (ANNs) using a cellular genetic algorithm (CGA). The main difference of this method compared with a common genetic algorithm (GA) is the use of a cellular automaton (AC) to provide location of GA chromosomes. The use of this method is based on de CGA interesting results found in the literature and in fact has not been found in the literature studied, the application of CGA in ANNs Multilayer Perceptron.

Keywords – Cellular Genetic Algorithm 1, Artificial Neural Networks 2, Genetic Algorithm 3.

1 Introdução

A busca por meios de otimização dos parâmetros das redes neurais artificiais (RNAs) é de fundamental importância para a sua aplicação, uma vez que as RNAs dependem da correta configuração de seus parâmetros para que possam ter desempenho satisfatório. Geralmente o ajuste de tais parâmetros é feito manualmente, através de processos de tentavas sucessivas, acarretando em um trabalho tedioso e propenso a erros[1]. Devido a isso, alguns autores defendem a aplicação de meios automáticos para a obtenção de RNAs com configuração adequada[2]. Entre estes meios, alguns dos mais difundidos são os Algoritmos Evolucionários (AEs) que visam "evoluir" as soluções do problema em questão a partir de uma solução inicial (ou um conjunto de soluções iniciais) [3, 4]. Eles são baseados em processos encontrados na natureza para a obtenção de bons resultados.

Existem várias técnicas pertencentes ao conjunto dos AEs, como os Algoritmos Genéticos (AGs) que são técnicas com inspiração em processos naturais de evolução (principalmente através de conceitos de biologia evolutiva) [5], o Enxame de Partículas (do inglês PSO – *Particle Swarm Optimization*), inspirado no comportamento social de bandos de aves, a Colônia de Formigas (do inglês ACO – *Ant Colony Optimization*) que é inspirado na observação do comportamento das formigas ao saírem de sua colônia para encontrar comida [6] e outras técnicas.

Assim, apresentamos neste trabalho um método de busca por parâmetros de RNAs que utiliza um tipo diferente de AG, conhecido como algoritmo genético celular (AGC). Este algoritmo tem a capacidade de dar localidade aos seus indivíduos, reduzindo as chances de convergência prematura através de uma lenta difusão das soluções. Uma das motivações para este trabalho foi a reduzida quantidade de aplicações de AGCs em conjunto com RNAs *Multilayer Perceptron* encontrada na pesquisa bibliográfica. O trabalho aqui proposto baseia-se na pesquisa apresentada em [1] para a construção dos operadores genéticos que servirão de base para a construção do AGC, estes operadores foram especialmente desenvolvidos para trabalhar com os dados reais das RNAs e foram os únicos operadores encontrados na literatura que utilizam tal tipo de codificação de parâmetros das RNAs *Multilayer Perceptron* (MLP).

O presente trabalho está organizado da seguinte forma: Seção 2 há uma breve explanação sobre redes neurais e algoritmos genéticos; A Seção 3 explica um pouco sobre os algoritmos evolucionários celulares; A Seção 4 aborda o método proposto; Seção 5 os resultados experimentais e Seção 6 as referências bibliográficas.

2 Redes Neurais Artificiais e Algoritmos Genéticos

Entre as técnicas evolucionárias os AGs são os mais frequentemente aplicados em conjunto com RNAs, formando as redes neurais evolucionárias (RNAEs) definidas por Yao [7]. As RNAEs possuem capacidade de explorar o espaço de soluções em busca do ponto de ótimo global através do AG e em seguida buscar pelo ponto de ótimo local com o uso da RNA, tornando assim, a busca mais efetiva. Contudo, o uso de RNAs e AGs para otimização de parâmetros se diferencia de metodologia para metodologia. Em alguns casos todos os parâmetros das RNAs podem ser procurados por um AG em outros é possível que o AG busque apenas pelos pesos iniciais da rede. De forma geral, o uso de RNAEs tem trazido bons resultados, seja utilizando o AG para uma pesquisa completa dos parâmetros das RNAs [7] ou por apenas parte dos parâmetros. No entanto, mesmo trazendo estes resultados, a aplicação de RNAEs é um processo custoso que possui uma série de peculiaridades. A aplicação do AG, por exemplo, possui questões importantes a serem levantadas como a possibilidade de convergência prematura da população dentro de um ponto que não é o ótimo global no espaço de soluções. Este tipo de erro ocorre, geralmente, devido à perda da diversidade genética causada pelo operador de seleção que constantemente reduz a variedade dos cromossomos, além das perturbações de boas sub-soluções pelas operações de cruzamento e mutação [4]. Em geral a evolução dos indivíduos sem o correto controle é a causa mais comum para este tipo de falha. Quando os indivíduos do AG evoluem sem o devido controle, é possível o desenvolvimento de "super indivíduos" que irão se replicar dentro da população, reduzindo a diversidade genética. Uma vez que a população de indivíduos não tenha diversidade, as chances de encontrar soluções diferentes são reduzidas, uma vez que é de fundamental importância que o AG tenha grande diversidade de indivíduos dentro de sua população para poder

encontrar boas soluções. Dessa forma, torna-se importante encontrar meios de manter a diversidade genética dos AGs durante a pesquisa evolucionária, evitando a possíbilidade da convergência prematura.

3 Algoritmos Evolucionários Celulares

Para contornar os problemas de convergência genética da população, muitos autores propõem modificações nos operadores genéticos dos AGs, como as várias formas de seleção, cruzamento e mutação (como é o caso da seleção por torneio, que tenta tornar o critério de seleção menos elitista que a tradicional seleção por roleta) [5]. Outros autores, por sua vez, têm estruturado a população dos AGs mediante a definição de algum critério de vizinhança entre as soluções manipuladas pelo algoritmo [8]. Uma das formas encontradas para amenizar o problema de convergência prematura causado pelos operadores dos AGs foi proposta por [9], onde os pesquisadores introduziram autômatos celulares (ACs) para realizar a localização e vizinhança na estrutura da população dos AGs.

Um AC é um modelo discreto estudado na teoria da computabilidade, matemática e biologia teórica. Consiste de uma grelha finita e regular de células, cada uma podendo estar em um número finito de estados, que variam de acordo com regras determinísticas. A grelha pode ser em qualquer número finito de dimensões. O tempo também é discreto e o estado de cada célula no tempo é uma função do estado no tempo de um número finito de células na sua vizinhança. Essa vizinhança corresponde a uma determinada seleção de células próximas (podendo eventualmente incluir a própria célula). Todas as células evoluem segundo a mesma regra de atualização, baseada nos valores de suas células vizinhas. Cada vez que as regras são aplicadas à grelha completa, uma nova geração é produzida. Os autômatos celulares foram introduzidos por Von Neumann e Ulam [10] como modelos para estudar processos de crescimento e auto-reprodução. Qualquer sistema com muitos elementos idênticos que interagem local e deterministicamente podem ser modelados usando autômatos celulares.

Assim, a saída encontrada em [9] foi controlar a seleção com base no *grid* do AC para evitar a perda rápida da diversidade durante a pesquisa genética, baseando-se na distribuição paralela das populações, onde a idéia de isolar os indivíduos possibilita grande diferenciação genética [11]. Em alguns casos [12], estes algoritmos, ao utilizar populações descentralizadas, conseguem prover uma melhor amostragem do espaço de soluções e também melhorar o tempo de execução. Esse tipo de combinação entre AGs e ACs é conhecido como algoritmo genético celular (AGC), que, por sua vez, pertence a um subconjunto dos algoritmos evolucionários chamados de algoritmos evolucionários celulares (AEC).

Nos AECs o conceito de vizinhança é muito utilizado, de modo que um indivíduo interage apenas com seus vizinhos mais próximos. Através desta restrição de interação, os AECs conseguem melhorar o comportamento numérico do algoritmo pela estruturação da população, conseguindo em geral, melhores taxas e tempos de execução. A principal idéia do modelo celular é dar à população uma estrutura especial definida como um grafo conectado, no qual cada vértice é um indivíduo que se comunica com os seus vizinhos mais próximos. Ao visualizar a população de AGs desta forma, podemos dizer que em um AG comum, sua população estaria como um grafo completamente conectado (onde os vértices são os indivíduos e as arestas as suas relações), enquanto num AG celular as interações acontecem apenas com os vizinhos mais próximos como um grafo de estrutura, difundindo lentamente as soluções e mantendo a diversidade mesmo entre os vizinhos próximos. A figura 1 mostra estes grafos [13].

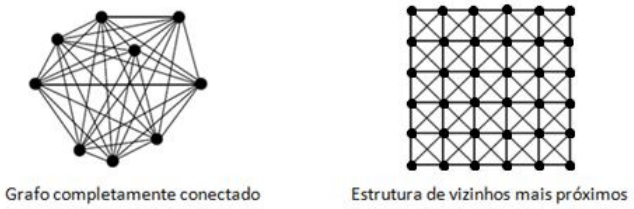

Figura 1 - Grafo Completamente Conectado e Grafo de Estrutura

Cada célula do AC possui um indivíduo do AG e os operadores genéticos (seleção, reprodução e mutação) são aplicados na vizinhança de um determinado cromossomo escolhido no *grid*. Como dito anteriormente, só há cruzamento entre aqueles indivíduos mais próximos, simulando a influência entre os indivíduos numa sociedade. As interações da população dependem de sua topologia de vizinhança. A vizinhança de um ponto particular da grelha (onde há um indivíduo) se define em termos da distância de Manhattan entre o ponto considerado da malha e os outros pontos da população [12]. A distância de Manhattan é definida como sendo o cálculo da distância que seria percorrida para ir de um ponto a (de posição $x1, y1$) até um ponto b (de posição $x2, y2$) seguindo um caminho em grade, ela é formulada como sendo a soma das diferenças de seus componentes correspondentes.

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

De forma geral, os dois tipos de vizinhança mais encontrados na literatura são a vizinhança de Neumann e a vizinhança de Moore, ambas mostradas na figura 2.

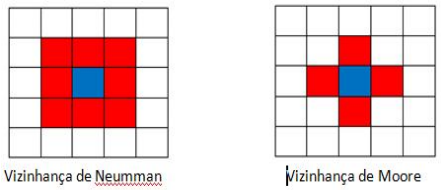

Figura 2 - Tipos de vizinhança

Os AGCs só podem interagir com seus vizinhos no ciclo reprodutor, quando são aplicados os operadores de variação (cruzamento e mutação). Em seguida, o indivíduo é substituído por um de seus descendentes gerados de acordo com algum critério (como o elitismo, onde o descendente substitui o pai caso seja uma solução melhor).

A sobreposição de vizinhanças nos AGCs fornece um mecanismo de migração implícito, uma vez que as melhores soluções são cuidadosamente espalhadas por toda população, mantendo a diversidade genética ao longo do tempo. Esta é a principal vantagem dos AGCs em relação a outros algoritmos evolucionários não estruturados. Esta dispersão lenta das melhores soluções gera um equilíbrio entre a exploração (exploration) e o aproveitamento (exploitation). Assim, a velocidade de dispersão das soluções pode ser controlada pelo tamanho da vizinhança de cada indivíduo. Uma vizinhança grande vai aumentar a velocidade de dispersão das melhores soluções, enquanto uma vizinhança pequena vai reduzir essa velocidade. A melhor aplicação de cada vizinhança também depende de critérios como o tamanho da população utilizada. Um exemplo é que o uso de grandes populações exigem maior vizinhança ou o algoritmo pode levar muito tempo para convergir, enquanto ao usar uma pequena população exige uma vizinhança menor para que o problema de convergência prematura possa ser melhor controlado. A figura 3 mostra um exemplo de sobreposição de vizinhanças e a dispersão das soluções [13].

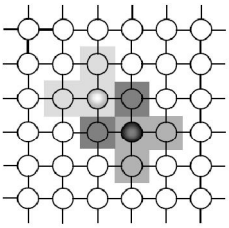

Figura 3 - Sobreposição de vizinhanças e dispersão de soluções

Os AGCs se dividem em dois tipos: Síncronos e Assíncronos. Se o ciclo reprodutor é aplicado a toda população de uma única vez (ao mesmo tempo), chamamos o AGC de síncrono, uma vez que os indivíduos que irão participar da próxima geração são formados todos de uma vez em paralelo. Se o ciclo reprodutor é aplicado sequencialmente seguindo algum determinado critério ao invés de atualizar todos os indivíduos de uma vez, chamamos o AGC de assíncrono [13].

Neste trabalho serão utilizados os dois tipos de AGCs, Síncrono e Assíncrono (com ciclo reprodutor feito através de uma escolha uniforme, onde as células são selecionadas com probabilidade uniforme e com reposição (um indivíduo pode ser escolhido mais de uma vez no ciclo reprodutor).

4 Algoritmos Genéticos Celulares para otimização de RNAs

O desenvolvimento do presente trabalho visa buscar os parâmetros de RNAs de forma evolucionária. Estes parâmetros estão codificados para uma busca evolucionária subdividida em camadas conforme os experimentos em [1] e em [14], onde cada camada possui uma determinada informação da RNA a ser "evoluída" por um AG específico (neste caso, por um AGC) que trabalha com a informação pertinente à camada em questão. O uso do AGC visa escapar dos ótimos locais e encontrar redes com melhores configurações iniciais para posterior refinamento através da busca local feita pela própria RNA. Essa pesquisa é feita da seguinte forma:

- Na camada mais baixa é feita a busca evolucionária por pesos iniciais da RNA (População de Pesos Iniciais-PPI);
- Na camada intermediária ocorre a busca evolucionária por arquitetura e funções de ativação (População de Arquitetura e Funções de Ativação – PAF). Nesta busca também estão incluídos o número de camadas escondidas e o número de neurônios por camada;
- Na camada superior é feita a busca evolucionária por regras de aprendizagem, ou seja, os parâmetros dos algoritmos de treinamento (População de Regras de Aprendizagem - PRA).

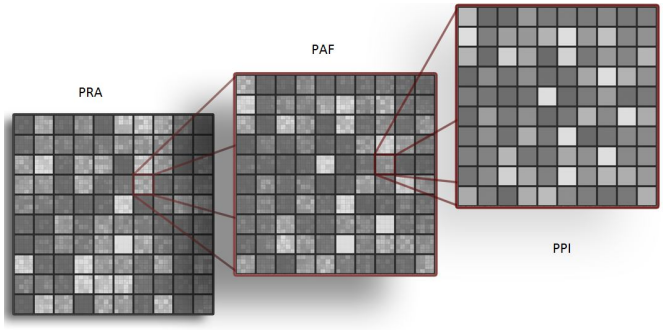

Figura 4 - Camadas da busca evolucionária

Como é possível notar, de acordo com a figura 4, as populações estão divididas em três camadas (cada uma representada por um autômato celular) distintas, mas que estão intimamente ligadas uma as outras. A população de regras de aprendizagem (PRA) possui entre os seus parâmetros uma população de arquiteturas e funções de ativação (PAF), que por sua vez possui uma população de pesos iniciais (PPI) dentro de seu conjunto de dados.

O presente trabalho tem como objetivo buscar por RNAs totalmente conectadas com aprendizado supervisionado que têm uma arquitetura simples (com no máximo 3 camadas ocultas e 12 nodos por camada), convergência rápida e poucas épocas de treinamento. Diferente do trabalho de Ajit Abraham [15] que utilizou até 500 épocas de treinamento, esta pesquisa utiliza até 50 épocas. Resultados anteriores mostraram que os dois algoritmos que obtiveram os melhores resultados foram: o Gradiente Conjugado Escalonado (SCG) e o *Backpropagation* (BP) [14]. Dessa forma, a pesquisa realizada em [14] foi modificada para trabalhar apenas com os dois algoritmos que tiveram os melhores resultados, ao invés dos 4 algoritmos utilizados anteriormente. Também foi realizada uma modificação na função de aptidão dos cromossomos do AGC, que anteriormente possuía um custo computacional considerável e neste trabalho foi substituído apenas pelo MSE (*Mean Square Error*). Essas modificações foram realizadas visando reduzir o custo computacional das RNAEs apresentadas. Cada uma das populações apresentadas na figura 4 trabalha com os valores reais específicos aos seus domínios de problema, ou seja, a PPI possui populações de cromossomos com os dados reais dos pesos das redes neurais, a PAF populações com os dados reais de configuração de arquiteturas e funções de ativação aplicadas e a PRA possui populações com os dados reais dos parâmetros dos algoritmos de aprendizagem aplicados. Uma vez que os dados reais são utilizados, o processo de codificação e decodificação dos cromossomos é dispensado, mas é preciso adaptar os operadores genéticos para funcionarem com tal tipo de codificação. Estes operadores foram desenvolvidos em [1] e foram adaptados para trabalhar com os AGCs em [14], sendo utilizados neste trabalho. A seleção de n indivíduos é realizada usando a estratégia de torneio, com uma taxa de pressão aleatória $p = 50\%$ e uma taxa de elitismo $e = 10\%$. Esta informação foi encontrada empiricamente após sucessivas execuções. O operador de seleção por torneio é usado tanto para seleção de sobreviventes como para seleção de pais para o cruzamento. O cruzamento de PPIs ocorre como descrito em [1, 14]. Diferente da taxa de mutação empregada em [1, 14] que era de 40%, a taxa de mutação empregada é de $m = 10\%$, onde dez por cento dos filhos passarão pelo processo de mutação e estes sofrerão mutação em 10% de sua composição. Este valor foi obtido após várias execuções e contribuiu com bons resultados e a boa manutenção da população. Vale ressaltar que a taxa empregada em [1] visava dar maior diferenciação genética a população (40%) o que mostra uma alta possibilidade de convergência prematura da metodologia. A mesma metodologia aplicada com AGCs reduziu a taxa de mutação a 10%, valor este relativamente alto se comparado a literatura, mas bem abaixo da pesquisa anterior. Esta mutação é feita exatamente como os pesquisadores indicam em [1], através de matrizes esparsas com 10% de seus valores entre $fx = [-0.5, 0.5]$ e os demais valores definidos como zero. Estas matrizes esparsas serão adicionadas aos filhos gerando a mutação.

A busca por arquiteturas é feita para encontrar configurações com até três camadas escondidas e até 12 nodos por camada escondida. A seleção dos pais é feita pelo algoritmo 1, que começa definindo um valor de aleatoriedade para determinar quantos indivíduos serão escolhidos aleatoriamente, visando tornar a seleção menos elitista. Em seguida é realizada a seleção dos melhores indivíduos da população (paf_n) de acordo com a sua aptidão e taxa e de elitismo. Estes indivíduos são retirados da população atual (paf_k) visando preservá-los na população final escolhida. Assim, enquanto uma determinada quantidade de elementos não for escolhida, será feito o sorteio de dois elementos da população paf_k. Diferente do algoritmo utilizado em [1], este algoritmo guarda as posições dos elementos no vetor paf_k, isto é feito visando modificar este elemento na população caso existam mais de um elemento com as mesmas características. Fazendo, assim, o uso de mais um recurso que evite a convergência prematura na população. Os passos seguintes escolhem aleatoriamente entre os indivíduos com menor ou maior aptidão. Quando o número de indivíduos aleatórios é alcançado, a escolha é feita apenas do indivíduo com melhor aptidão entre os dois escolhidos. O passo seguinte faz a verificação da igualdade entre a PAF escolhida e as PAFs do vetor paf_n, ou seja, é verificado se existem arquiteturas iguais (em quantidade de camadas e nodos por camada) dentro do vetor paf_n. Se forem diferentes, o elemento é adicionado a paf_n, senão ele é mutado e colocado novamente na sua mesma posição na população paf_k, visando evitar a convergência prematura e acelerar o tempo de escolha dos indivíduos.

O operador de cruzamento da PAF pode produzir indivíduos similares aos seus pais, ocasionado pela simplicidade do operador. Por isso, após o cruzamento é preciso aplicar o operador de mutação para manter a diversidade da população. O algoritmo de mutação das PAFs está descrito em detalhes em [14]. Ele começa com a definição de filhos que sofrerão mutação. Em seguida é gerado um número aleatório no intervalo [0,1]. Este valor será utilizado como uma probabilidade que indicará se uma arquitetura irá sofrer um incremento ou decremento na quantidade de camadas ou no número de neurônios escondidos por camada. Na etapa de mutação, são avaliados todos os casos possíveis de quantidade de camadas (1, 2 ou 3). De acordo com uma determinada probabilidade o indivíduo pode ter sua quantidade de camadas aumentada ou sofrer uma redução de uma ou duas camadas. Caso o indivíduo tenha uma ou duas camadas ocultas, há maiores chances de este passar a ter 3 camadas (visando reduzir o número de nodos por camadas). A probabilidade de redução de camadas só é maior para indivíduos com 3 camadas ocultas (tendo maiores chances de ser 2 camadas). De forma geral, esta etapa prioriza gerar indivíduos com mais camadas e menos nodos por camada. Caso a probabilidade não atinja um valor esperado para adicionar ou remover camadas dentro das arquiteturas, ocorrerá uma mudança nas dimensões das camadas, ou seja, a arquitetura será mantida, mas a quantidade de nodos em cada camada será modificada através de valores gerados entre $arqval = [-2,5]$ que serão adicionados as dimensões atuais da arquitetura (tendo no mínimo 1 e no máximo 12 nodos por camada). As PAF são selecionadas de acordo com sua melhor aptidão que leva em consideração a aptidão das PPIs (MSE).

A pesquisa evolucionária por regras de aprendizagem acontece com a busca dos parâmetros dos algoritmos *Back-Propagation(*BP) e Gradiente Conjugado Escalonado (SCG). De acordo com o algoritmo de aprendizagem adotado, uma PRA terá parâmetros deste algoritmo para funcionar. A seleção do indivíduo a partir da PRA também usa a estratégia de torneio. As taxas de mutação e elitismo são as mesmas apresentadas anteriormente. Como os indivíduos da PRA têm informações relacionadas a parâmetros do algoritmo de aprendizagem em questão, o seu cruzamento ocorre com a geração de novos valores dentro de um intervalo com base nos valores dos pais. De acordo com uma probabilidade os filhos realizam a mutação adicionando ou subtraindo valores as suas respectivas taxas. O valor da mutação nestas taxas é de 10% para mais ou para menos, dependendo também de uma probabilidade. O funcionamento geral do AGC assíncrono é apresentado no algoritmo 2 e o AGC Síncrono é descrito em seguida. Os AGCs iniciam sua busca gerando todos os indivíduos e calculando suas aptidões. Em seguida, os operadores genéticos são utilizados. Tais operadores tentam manter a diversidade genética utilizando a seleção por torneio, a mutação e o autômato celular, enquanto fazem a evolução das soluções. Como já citado, a manutenção da diversidade é um ponto considerado importante no uso de AGs, visto que tal diversidade evita que a busca caia em pontos de mínimos locais no espaço de pesquisa. A manutenção da diversidade com o uso do autômato celular é feita através da localidade, uma vez que os indivíduos só conseguem cruzar localmente, as chances de escapar dos pontos de mínimo local aumentam, diminuindo a possibilidade de convergência prematura da população.

Entrada: $\boldsymbol{paf}_n, n, p, e$ // pop. atual, tamanho da nova pop., aleatoriedade e taxa de elitismo
Saída: $\boldsymbol{paf_n}$ // nova população com *n* indivíduos

```
1  inicio
2      aleatórios ← ((p/100) * k)
3      paf_n ← melhores (paf_k, e)
4      paf_k ← paf_k - paf_n
5      selecionados ← tamanho(paf_n)
6      enquanto selecionados ≤ n faça
7          [a, posA, b, posB] ← sorteio(paf_k)
8          se aleatórios > 0 e rand(1) > probs faça
9              [ [escolhido, posEscolhido] ← min (a_aptidão, posA, b_aptidão, posB)
10             aleatórios ← aleatórios − 1
11         senão
12             escolhido ← max(a_aptidão, b_aptidão)
13         se verificarIgualdade(paf_n, escolhido) = falso então
14             paf_k ← paf_k − escolhido
15             paf_n ← escolhido
16             selecionados ← selecionados + 1
17         senão
18             escolhido ← mutaUmaPAF(escolhido)
19             paf_k[posEscolhido] ← escolhido
20 fim
```

Algoritmo 1 – Seleção de PAFs

Entrada:Parâmetros a serem otimizados // Vide [14]

Saída: *Um conjunto de redes neurais ótimas ou quase − ótimas*// novos descendentes

inicio
  Gere aleatoriamente as populações de: Pesos iniciais (PPI), Arquiteturas e funções (PAF) e Regras de aprendizagem (PRA)
  para cada *algoritmo de aprendizagem* **faça**
    Calcule a aptidão para todos os indivíduos
      para cada *geração* ∈ *BERA* **faça**
        para cada *PAF* ∈ *PRA* ***e*** *geração* ∈ *BEAFA* **faça**
          para cada *PPI* ∈ *PAF* ***e*** *geração* ∈ *BEP* **faça**
            Crie um autômato celular At de acordo com o tamanho da população P;
            Insira em At a população de acordo com o seu fitness;
            Inicie o cruzamento através do AGC;
            Seja Q um número de filhos previamente determinado
            para *numFilhos* ← 1: Q **faça**
- Selecione aleatoriamente um indivíduo de At e selecione seus vizinhos por torneio;
- Comece o processo de cruzamento entre os vencedores do torneio para produzir descendentes
- Dependendo de uma probabilidade m aplique a mutação nos descendentes;
- Avalie os descendentes
- Se algum filho gerado for melhor que o pai, substitua o pai pelo filho na mesma posição no grid do autômato At

            Selecione entre todos os indivíduos, inclusive os novos, sobreviventes para próxima rodada
          Repita os passos da BEP
        Repita os passos da BEP
  Selecionar redes quase-ótimas obtidas com cada algoritmo de aprendizagem
fim

Algoritmo 2 – CGA Assíncrono

O AGC Síncrono é bastante parecido com o Assíncrono, substituindo apenas a etapa de busca evolucionária de cada população pelos seguintes passos:

Crie um autômato celular At de acordo com o tamanho da população P;
Insira em At a população de acordo com o seu fitness;
Crie um autômato celular At_{aux} e faça-o igual a At
Inicie o cruzamento através do AGC;
para x ← 1: *linhasDeAt* **faça**
  para y ← 1: *colunasDeAt* **faça**
- Selecione o indivíduo (x,y) de At e selecione seus vizinhos para torneio;
- Comece o processo de cruzamento entre os vencedores do torneio para produzir descendentes
- Dependendo de uma probabilidade m aplique a mutação nos descendentes;
- Avalie os descendentes
- Se algum filho gerado for melhor que o pai, substitua o pai pelo filho na mesma posição no grid do autômato At_{aux}

Faça $At = At_{aux}$

5 Resultados experimentais e conclusões

A tabela 1 representa os resultados obtidos com quatro bases do repositório UCI [16] em relação ao método NNGA-DCOD proposto em [1], que utiliza um AG comum para a obtenção dos parâmetros inicias de RNAs e teve resultados satisfatórios quando comparado a busca manual.

Métodos	ALG.	VALORES DO MSE (EM %) PARA CADA BASE							
		Credito Alemão		Ionosfera		Veículos		cavalos	
		Média	Desvio	Média	Desvio	Média	Desvio	Média	Desvio
NNGA-DCOD	BP	24,3870	0,1425389	24,3079	0,4563	24,9331	0,0278	24.1395	0,1244015
	SCG	20,3479	2,3656681	16,6267	3,3796	21,8682	1,391	16.1976	2,034981
AGC-1 ASSINCRONO	BP	22,4144	0,46740	23,6326	0,9057	24,9385	0,0248	23,10312	0,002641
	SCG	16,8964	0,22202	7,6622	0,9364	9,8864	0,3673	13,6008	0,001005
AGC-2 (SINCRONO)	BP	22,5116	0,3833819	23,80388	0,37960	24,9298	0,0164	23.0633	0,640212
	SCG	16,91576	0,0866647	8,25933	0,60568	9,778	0,3546	15.7953	0,417595

Tabela – 1 Resultados experimentais

Os experimentos foram realizados visando comparar os resultados do método celular em relação ao NNGA-DCOD. As bases utilizadas foram Crédito Alemão com 24 atributos (atb), 1000 exemplos (exp) e 2 classes (cla); Ionosfera com 34 atb, 351 exp e 2 cla; Veículos com 18 atb, 846 exp e 4 cla e Cavalos com 58 atb, 364 exp e 3 cla. A divisão do conjunto de dados seguiu o método utilizado em [17], onde a cada iteração os dados foram aleatoriamente divididos em metades. Uma das metades foi utilizada como entrada para os algoritmos com 70% de seus dados para treinamento e 30% para o conjunto de validação. A outra metade da base foi utilizada como conjunto de teste.

Como é possível perceber, tanto o AGC Síncrono quanto o Assíncrono tiveram vantagem na maioria das bases, tendo ambos os AGCs médias maiores que o NNGA-DCOD apenas na base veículos com o algoritmo BP. Apenas a base veículos teve resultados muito próximos entre as três técnicas com o algoritmo BP, mas com o SCG os resultados foram favoráveis aos dois métodos celulares. Vale ressaltar que as quatro bases são consideradas difíceis e tanto o AGC Síncrono quanto o AGC Assíncrono obtiveram desempenho considerável.

Para determinar se os resultados obtidos possuem realmente significância estatística, foi utilizado um teste F com o auxílio do software BioStat 5.0. O teste F é um teste de hipóteses que é aproximadamente distribuído com cerca de cinco a dez graus de liberdade e rejeita a hipótese nula de que os dois algoritmos têm a mesma taxa de erro com uma significância de 0,05 se F > 4,74. Esta metodologia foi utilizada devido ao fato do método usual (teste T de Student) gerar um aumento de erros do tipo 1, onde os resultados são erroneamente considerados diferentes com mais freqüência do que o esperado, dado o nível de confiança utilizado[17]. O teste foi aplicado sobre a distribuição de 10 resultados de cada base no conjunto de erros de testes das RNAs.

BP	C. Alemão	Ionosfera	Veículos	Cavalos
NNGA-DCOD	> 4,74	> 4,74	< 4,74	> 4,74
AGC-1 SINC				

Tabela – 2 Teste F do algoritmo BP entre NNGA-DCOD com o AGC Síncrono

SCG	C. Alemão	Ionosfera	Veículos	Cavalos
NNGA-DCOD	> 4,74	> 4,74	> 4,74	> 4,74
AGC-1 SINC				

Tabela – 3 Teste F do algoritmo SCG entre NNGA-DCOD com o AGC Síncrono

BP	C. Alemão	Ionosfera	Veículos	Cavalos
NNGA-DCOD	> 4,74	< 4,74	< 4,74	>4,74
AGC-2 ASSINC				

Tabela – 4 Teste F do algoritmo BP entre NNGA-DCOD com o AGC Assíncrono

SCG	C. Alemão	Ionosfera	Veículos	Cavalos
NNGA-DCOD	> 4,74	> 4,74	> 4,74	> 4,74
AGC-2 ASSINC				

Tabela – 5 Teste F do algoritmo SCG entre NNGA-DCOD com o AGC Assíncrono

De acordo com o teste F os resultados do AGC Síncrono tiverem relevância estatística nas 4 bases testadas em relação ao NNGA-DCOD, mostrando que a ferramenta obteve o êxito proposto e conseguiu atingir os resultados desejados. O AGC assíncrono obteve relevância estatística nas quatro bases, com exceção da base veículos e Ionosfera com uso do algoritmo BP.

6 Referências

[1] L. M. Almeida and T. B. Ludermir, (2007), "Automatically Searching near optimal artificial neural networks". ESANN, 549 – 554

[2] J.A. Bullinaria, (2005), "Evolving neural networks: is it really worth the effort?", Proceedings of the 13th European Symposium on Artificial Neural, 267 – 272

[3] A. Abraham, (2004), "Meta learning evolutionary artificial neural networks". **Neurocomputing, no. 56,** 1-38

[4] H. Rajabalipour, H. Haron and M. I. Jambak, (2009), "The improved genetic algorithm for assignment problems", ICSPS, 187-191

[5] R. Linden, (2006), "Algoritmos Genéticos – Uma importante ferramenta da inteligência computacional", Rio de Janeiro, Brasport 1a edição.

[6] R. Umarani and V. Selvi, (2010), "Comparative Analysis of ant colony and particle swarm optimization techniques". International Journal of Computer Applications. 1-6

[7] X. Yao, (1999), "Evolving artificial Neural Networks", Proceedings of the IEEE, vol.87, no. 9, 1423-1447

[8] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro and E. Alba, (2007) "A study of strategies for neighborhood replace and archive feedback in a multiobjective cellular genetic algorithm", Proceeding of the evolutionay multi-criterion optimization (EMO), vol. 4403, 126-140

[9] Y. J. Cao and H. Q. Wu, (1998) "A cellular automata based genetic algorithm and its application in machine design optimization" ICC, 1593-1598

[10] S. Wolfram, (2002), "A new kind of science", Wolfram Media, ISBN 1-57955-08-8

[11] S. Wright, (1943), "Isolation by distance", **Genetics,** 28, 114-138

[12] E. Alba, and J. M. Troya, (2002), "Improving flexibility and efficiency by adding parallelism to genetic algorithm", Statistics and Computing, vol. 9, no. 2, 91-114

[13] E. Alba and B. Dorronsoro, (2008), "Cellular Genetic Algorithms", Operations research computer science interfaces series, Vol. 42. ISBN – 978-0-387-77609-5

[14] A. P. Silva, T. B. Ludermr and L. M. Almeida, (2011), "Método AGCRN-CR para busca automática de redes neurais artificiais", ENIA – Encontro Nacional de Inteligência Artificial, Sessão Poste, Artigo 7

[15] A. Abraham, (2004), "Meta learning evolutionary artificial neural networks", Neurocomputing, vol. 56, 1-38

[16] D. J. Newman, S. Hettich, C.L. Blake, and C.J. Merz, (1998) "UCI repository of machine learning databases" [Online]. Available: http://archive.ics.uci.edu/ml/

[17] E. Cantú-Paz and C. Kamath, (2005), "An empirical comparison of combination of evolutionary algorithms and neural networks for classification problems". IEEE Transactions on Systems, Man, and Cybernetcs, Part B, vol. 35, no. 5, 915-927