Configuração automática de arquitetura de rede neural artificial por algoritmo genético

Edson Simões dos Santos
Instituto Federal Fluminense
Rua Coronel Walter Kramer, 357 - Parque Santo Antônio - Campos dos Goytacazes, RJ - CEP 28080-565
edson.santos@iff.edu.br

Italo Matias
Universidade Cândido Mendes
Rua Anita Peçanha, 100 – Pq. São Caetano – Campos dos Goytacazes, RJ – CEP.: 28030-335
italo@ucam-campos.br

Márcio Oliveira Pontes
Instituto Federal Fluminense
Rua Coronel Walter Kramer, 357 - Parque Santo Antônio - Campos dos Goytacazes, RJ - CEP 28080-565
molpontes@gmail.com

Gustavo Lemos Schwartz
Instituto Federal Fluminense
Rua Coronel Walter Kramer, 357 - Parque Santo Antônio - Campos dos Goytacazes, RJ - CEP 28080-565
gustavolschwartz@gmail.com

RESUMO

Os algoritmos de otimização, baseados em métodos heurísticos, têm sido largamente utilizado na solução de diversos problemas, principalmente quando uma solução analítica não é viável em relação ao custo e tempo de obtenção de um resultado satisfatório. Também se tornou crescente o emprego da Inteligência Computacional (IC) em demanda da complexidade destes problemas e pela possibilidade de serem tratados computacionalmente. O objetivo deste estudo é projetar um sistema, através de um Algoritmo Genético (AG), capaz de autoconfigurar uma arquitetura de Rede Neural Artificial (RNA) utilizada numa aplicação específica. O algoritmo evolutivo utilizado (AE) manipula a quantidade de camadas ocultas, número de neurônios das camadas e funções de ativação para obter uma melhoria na arquitetura da rede utilizada no problema proposto sem que um conhecimento especialista do sistema a ser modelado pela RNA seja exigido. Estratégia de crossover, mutação e elitismo são utilizadas na evolução da população do AG. As operações de otimização neste trabalho ocorrem num espaço de busca limitado pelo autor num teste prévio. O método de configuração automática da RNA por AG obteve uma

configuração de rede em conjunto com o parâmetro função de ativação, e retornou o menor erro após o treinamento para um determinado conjunto de soluções gerado.

Palavra-chave: Algoritmo Genético; Configuração Automática; Inteligência Computacional; Redes Neurais Artificiais.

ABSTRACT

Optimization algorithms, based on heuristic methods, have been widely used in the solution of several problems, especially when an analytical solution is not feasible in relation to the cost and time of obtaining a satisfactory result. The use of Computational Intelligence (CI) has also become increasingly important due to the complexity of these problems and the possibility of being treated computationally. The objective of this study is to design a system, through a Genetic Algorithm (GA), capable of autoconfiguring an Artificial Neural Network (RNA) architecture used in a specific application. The evolutionary algorithm used (EA) manipulates the number of hidden layers, number of neurons of the layers and activation functions to obtain an improvement in the architecture of the network used in the proposed problem without being required a specialized knowledge of the system to be modeled by RNA. Strategy of crossover, mutation and elitism are used in the evolution of the GA population. The optimization operations in this work occur in a search space limited by the author in a previous test. The RNA automatic configuration method by GA obtained a network configuration in conjunction with the activation function parameter, and returned the smallest error after training for a given set of solutions.

Keywords: Artificial Neural Networks; Automatic Configuration; Computational Intelligence; Genetic Algorithm.

1. INTRODUÇÃO

O crescimento dos recursos computacionais incentivou o tratamento de diversos problemas computacionalmente e, também, permitiu analisá-los num estado de maior complexidade. Estes problemas exigem soluções complexas que, geralmente, são difíceis para programadores humanos conceber, recorrendo assim à implementação de teorias evolutivas para alcançar soluções de qualidade num tempo viável. Pode-se citar a RNA como um excelente exemplo de filosofia evolutiva da computação (MITCHELL, 1999). Embora as RNA's apresentem bom desempenho na resolução destes problemas, ainda existe uma carência no quesito metodologia ou regras exatas para obter a melhor configuração da rede. Em sua maioria, a configuração de uma RNA se caracteriza mais como arte do que ciência, pois demanda uma considerável experiência de cada projetista.

Cabe lembrar que, em alguns casos, os métodos exatos tendem a ser inadequados para a resolução de problemas de elevada complexidade, pois geralmente possuem alto custo para sua obtenção com a possibilidade de não fornecer uma solução viável dentro de um tempo satisfatório. Contudo métodos heurísticos, apesar de não garantirem a melhor solução, costumam fornecer boas soluções num tempo razoável.

Estudos na área de Inteligência Artificial (IA) mostraram que os exemplos naturais de aprendizagem e adaptação são tesouros de procedimentos e estruturas implementa-

dos por métodos heurísticos através da construção de algoritmos de busca robustos. Estes são procedimentos de busca probabilísticos projetados para trabalhar em grandes espaços, destaque entre eles para o Algoritmo Genético (GOLBERG, HOLLAND, 1988).

No presente trabalho, para contornar o problema enfrentado na configuração da Rede Neural (tipo ***feedfoward***), um AG foi utilizado como mecanismo de configuração e otimização, determinando a quantidade de camadas ocultas, suas funções de ativação e, também, o número de neurônios nas camadas ocultas. O método heurístico AG foi escolhido por apresentar fácil implementação computacional, obter soluções viáveis com maior simplicidade, exigindo somente uma função de custo.

2. REVISÃO BIBLIOGRÁFICA

As informações da revisão bibliográfica se encontram em ordem com a proximidade do tema deste trabalho, e serão apresentadas do tema geral para o tema específico, ou seja, para o mais próximo do tema deste estudo.

Recentemente, pesquisas sobre rede neural tem recebido cada vez mais atenção por pesquisadores da área de controle na identificação, análise e projeto de sistemas de controle devido ao potencial de aprendizagem e reconstrução das relações não lineares. O tamanho da rede, muitas vezes, é medido pelo número de unidades de neurônios em camadas ocultas, que reflete na capacidade da RNA para aproximar uma função arbitrária, surgindo assim a questão: qual é o tamanho necessário da rede neural para resolver um problema específico? Atualmente, pesquisas têm sido realizadas para solucionar esta questão.

Yao e Liu (1997) utilizam um algoritmo de evolução, denominado EPNet (Evolutionary Programming Network), para a evolução de Redes Neurais Artificiais. O algoritmo evolutivo utilizado na EPNet baseia-se na programação evolutiva de Fogel's (PE) e dá ênfase na mutação, evoluindo a arquitetura e pesos das conexões. Para evitar o problema da permutação, o crossover não é adotado como estratégia, e sim a evolução dos indivíduos. A RNA utilizada é do tipo feedforward e a função de ativação é a função sigmoide.

Gutiérrez et al. (2005) utilizam um esquema de codificação construtivo indireto com base na abordagem celular autônoma (Cellular automata), proposto para encontrar automaticamente uma arquitetura de RNA apropriado para um determinado problema. Um Algoritmo Genético é utilizado para geração de uma população e sua manutenção, atividades com crescimento celular e poda ocorrem durante a otimização. A função de ativação utilizada na RNA é uma função sigmoidal.

Rivero et al. (2008) utilizam um AG para configuração automática da RNA. Características como número de camadas ocultas, pesos e bias são alteradas para promover a evolução. Técnicas de crossover e mutação também são adotadas. Na solução do problema, uma codificação do tipo gráfica (Graph Codification) é utilizada.

Mekki e Chtourou (2012) sugerem trabalhar com uma arquitetura de RNA fixa quadrada de tamanho significativo e suficiente para descrever a região de trabalho, utilizando como função de ativação uma função de base radial. Também propõe uma rede auto-organizável, variando sua estrutura dinamicamente, adicionando ou removendo Funções Gaussianas de base radial, de modo a assegurar a precisão e aproximação desejada e ao mesmo tempo manter a complexidade de rede apropriada. O tamanho das redes é afetado pela distância entre os pontos necessários para descrever a função aproximada dentro de

uma dada precisão.

No trabalho de Schuma e Birdwell (2013) o algoritmo de treinamento utilizado para essas redes é evolutivo. Uma população de RNA é gerada, mantida e uma função de aptidão para a aplicação específica é aplicada a cada rede na população. Redes com maior aptidão são preferencialmente selecionadas para a reprodução. A cada seleção, duas redes com aptidão relativamente alta são selecionadas e as operações de cruzamento e mutação são escolhidas com alguma probabilidade. As operações de cruzamento e de mutação não afetam apenas os pesos das sinapses, mas o número de neurônios e o número de sinapses.

A proposta do presente trabalho é aplicar um mecanismo de configuração automática da RNA, adicionando como opção no algoritmo utilizado, quatro funções de ativação, além da opção de quantidade de camadas ocultas e números de neurônios nestas camadas. Assim aumentam-se os parâmetros de configurações da rede diferenciando das publicações anteriores.

A RNA deste trabalho foi construída com objetivo de ser utilizada como parte de um controlador adaptativo num modelo de pêndulo invertido conforme o trabalho de Santos (2013). Esta rede possui 3 entradas (massa do sistema, tempo de acomodação e sobressinal), e 3 saídas cujos valores são os polos do controlador por realimentação de estados. O conjunto de dados utilizados no treinamento supervisionado da rede foi extraído do modelo não linear de um pêndulo invertido.

3. APLICAÇÃO DA INTELIGÊNCIA COMPUTACIONAL

Na formalização do problema e busca da solução, o conhecimento especialista é de suma importância para obtenção de sucesso na resolução de problemas específicos e complexos. A solução de problemas complexos exige uma combinação de entendimento teórico e empírico sobre um problema específico, apoiado por um conjunto de regras heurísticas (conhecimento do maior número de estados possíveis assumidos no processo de solução) (LUGER, 2004).

Entretanto, hoje, alguns pesquisadores acreditam que grandes conjuntos de "regras complexas"são demasiadamente custosos para os cientistas. Em vez disso, acredita-se que o melhor caminho para a inteligência artificial é através de um paradigma em que os seres humanos só devem escrever conjuntos de regras simples e fornecer um meio para que o sistema se adapte. Comportamentos complexos como a inteligência vão emergir da aplicação paralela e interação destas regras (MITCHELL, 1999). Assim, os sistemas especialistas devem ser construídos através da extração do conhecimento de um especialista humano utilizando Inteligência Artificial (IA) para solução de problemas similares (LUGER, 2004).

No que se refere às RNA's propostas, elas são apresentadas com abordagem evolutiva para a otimização da configuração estrutural das RNA's e similares, como EP (evolutionary programming) encontrado na literatura de Yao E Liu (1997). Porém, pelo fato de explorar uma região de busca contendo todas as arquiteturas possíveis, tornam-se extremamente custosas computacionalmente, embora sejam capazes de produzir bons resultados. Indivíduos diferentes, mas com mesma aptidão também podem ter dificuldades numa abordagem evolutiva.

4. DESENVOLVIMENTO

De acordo com Fernandes (2008), os AG's são métodos adaptativos que podem ser usados para resolver problemas de busca e otimização. Estes métodos baseiam-se na teoria evolutiva postulado em Darwin (1859), na qual as populações evoluem na natureza de acordo com os princípios de seleção natural e sobrevivência dos mais aptos. Imitando este processo, os AG's são capazes de fornecer soluções para os problemas do mundo real.

A metodologia proposta por Michalewicz (1999) foi utilizada neste trabalho para implementação computacional do AG. Pequenas adaptações para o crossover foram feitas no algoritmo utilizado e mostrado em seu pseudocódigo. Destaca-se que um benefício em trabalhar com AG é que este não depende da obtenção de gradiente para a otimização.

4.1. Projeto 1

O AG binário utilizado neste estudo possui uma população pré-determinada. Cada indivíduo possui três conjuntos de genes que são responsáveis por três características distintas: a característica "a" representa a quantidade de camadas ocultas com a função de ativação tansig; "b" representa a quantidade de camadas ocultas com a função de ativação radbas e "c", a quantidade de camadas ocultas com a função de ativação purelin. Como sugerido por Mekki e Chtourou (2012) numa estrutura variável RNA, o número de funções pode ser aumentado ou diminuído de acordo com estratégias de design da arquitetura de rede para que esta não seja subdimensionada ou superdimensionada para um conjunto de dados específicos. As funções de ativação são mostradas na Figura 1.

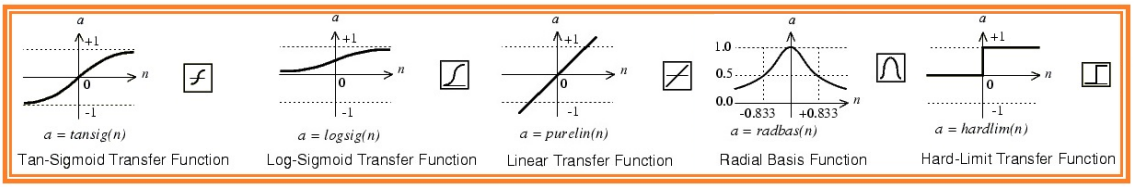

FIGURA 1. Funções de ativação. Fonte: Matlab R2013a. Disponível em: <http://www.mathworks.com/help>. Acesso em: abril de 2019

O número de neurônios (**n**) em cada camada oculta está relacionado com a seguinte equação: **n= 2(característica)+5**, sendo o número mínimo de neurônios em uma camada igual a 7 quando se tem uma camada oculta referente a uma função de ativação, e o número máximo igual a 19 para 7 camadas ocultas para uma função de ativação. Se **a**, **b** e **c** igual a 7, teremos 21 camadas ocultas, cada uma com 19 neurônios. O número mínimo de neurônios foi arbitrado pelo autor para que houvesse a capacidade de reconhecimento de não linearidades com um mínimo de performance exigido. Esta prática pode ser observada nos trabalhos de Liu (1999), Mekki et. al. (2006), Mekki E Chtourou (2012) e Lian et. al. (2008). A Figura 2 apresenta um indivíduo na base binária e suas características.

Neste AG as gerações são criadas a partir do cruzamento correspondente a 25% dos melhores indivíduos da população (fitness mais baixo, erro médio quadrático retornado após o treinamento da RNA). A estratégia utilizada é mostrada na Figura 3.

Na Figura 3, o cruzamento ocorre da seguinte forma: divide-se o cromossomo

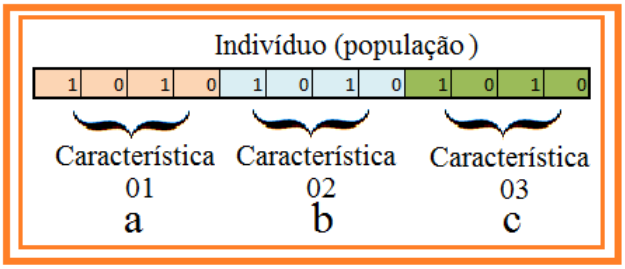

FIGURA 2. Cromossomo representado na base binária. Fonte: Próprio Autor.

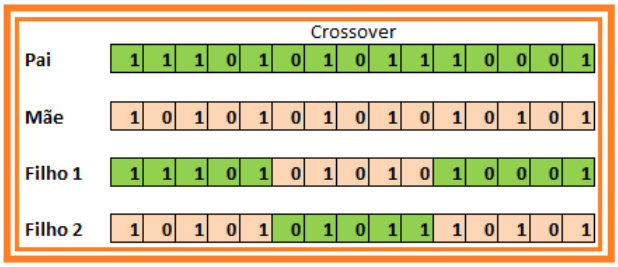

FIGURA 3. Geração de descendentes por crossover. Fonte: Próprio Autor.

em três partes, nas quais um descendente recebe duas partes do pai e uma da mãe, e o segundo recebe duas partes da mãe e uma do pai.

A mutação é aplicada em 5% dos indivíduos da população global através do método de roleta (valor arbitrado), no qual um gene é modificado de uma posição aleatória do cromossomo.

O número de gerações também é pré-determinado. O AG utilizado neste projeto tem o seguinte pseudocódigo:

Algoritmo 1:

```
início
    Gera população (aleatoriamente num intervalo preestabelecido)
    Atribui fitness individual e calcula o fitness global
    Aplica o Crossover
    Aplica a Mutação
    Atualiza fitness individual e global
    Seleciona melhor indivíduo
    Se a condição de nº de gerações não for atendida, retorna-se ao passo 3
fim
```

Configurações: $0 \leq \mathbf{a} \leq 7$, $0 \leq \mathbf{b} \leq 7$, $0 \leq \mathbf{c} \leq 7$. Estas variáveis possuem valores inteiros. Os autores optaram por limitar o tamanho da RNA em 21 camadas ocultas devido ao custo computacional requerido. A população inicial foi de 64, um limite de gerações foi definido igual a 5. Não foi utilizada a estratégia de elitismo neste AG.

4.2. Projeto 2

Um segundo AG foi utilizado para configuração da RNA. Este pode ser acessado pelo comando do Matlab ga(@função_de_avaliação, números_variáveis, A,b), onde **A** e **b** são matrizes das inequações ($\mathbf{A} \leq \mathbf{b}$) e estas matrizes determinam o range dos parâmetros. A função de avaliação do AG possui nove parâmetros: a, b, c, A, B, C, α, β, γ.

Os parâmetros **a**, **b** e **c** são responsáveis pela quantidade de camadas ocultas (camadas ocultas divididas em três blocos). Os parâmetros **A**, **B** e **C** são parâmetros responsáveis pelas funções de ativação que serão utilizadas nas camadas ocultas (**a**, **b** e **c**), que podem ser: **tangente sigmoide, base radial, linear e função logarítmica sigmoide**. O número de neurônios de cada camada é definido pelos parâmetros α, β e γ.

Este algoritmo é iniciado com uma população de 20 indivíduos e uma nova população é gerada com 80% dos indivíduos provenientes de crossover. A mutação aleatória é aplicada num valor de 1 de desvio padrão, utilizando uma curva Gaussiana na população inicial e este valor decresce linearmente para 0 ao final das gerações. Dois melhores indivíduos são passados para nova população utilizando a estratégia de elitismo, ocorrendo uma migração de 20% da população antiga para a nova população. O número máximo de gerações foi limitado em 100. A função de custo é a mesma utilizada no projeto 1 e retorna a performance obtida no treinamento da RNA. A configuração do AG pode ser visualizada inserindo o comando options = gaoptimset(@ga) no workspace.

Configuração: a seguir é apresentada a inequação $\mathbf{A} \leq \mathbf{b}$.

Tabela 1. A

a	b	c	A	B	C	α	β	γ
1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1
-1	0	0	0	0	0	0	0	0
0	-1	0	0	0	0	0	0	0
0	0	-1	0	0	0	0	0	0
0	0	0	-1	0	0	0	0	0
0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	0	0
0	0	0	0	0	0	-1	0	0
0	0	0	0	0	0	0	-1	0
0	0	0	0	0	0	0	0	-1

Fonte: Próprio Autor.

$\leq$

Tabela 2. b

10
10
10
4
4
4
30
30
30
0
0
0
-1
-1
-1
-1
-1
-1

Fonte: Própio Autor.

A função de avaliação deste AG possui nove parâmetros (**a, b, c, A, B, C,** α, β e γ), o intervalo destes são definidos pelas matrizes de inequação, $0 \leq \mathbf{a} \leq 10$, $0 \leq \mathbf{b} \leq 10$, $0 \leq \mathbf{c} \leq 10$, $1 \leq \mathbf{A} \leq 4$, $1 \leq \mathbf{B} \leq 4$, $1 \leq \mathbf{C} \leq 4$, $1 \leq \alpha \leq 30$, $1 \leq \beta \leq 30$, $1 \leq \gamma \leq 30$. Todos os parâmetros são inteiros.

5. CONFIGURAÇÃO DA RNA

Uma feedforward Perceptron de Múltiplas Camadas (MLP- Multilayer Perceptron), com duas ou mais camadas ocultas pode ser considerada como um aproximador universal, pois consegue realizar um mapeamento com relação entrada/saída, uma característica existencial. Isto porque, cada neurônio, na primeira camada, cria uma saliência, e a próxima camada combina estas saliências em regiões disjuntas do espaço. Entretanto, um problema ainda sem solução, é a determinação do número ótimo de camadas escondidas e do número de neurônios em cada camada para um dado problema, faltando um mecanismo sistemático de se chegar à rede neural mais indicada em cada aplicação (GEMAN et al., 1992 apud RAVIV e INTRATOR, 1999). A Figura 4 mostra o modelo de neurônio biológico e o modelo de neurônio artificial proposto por McCulloch e Pitts.

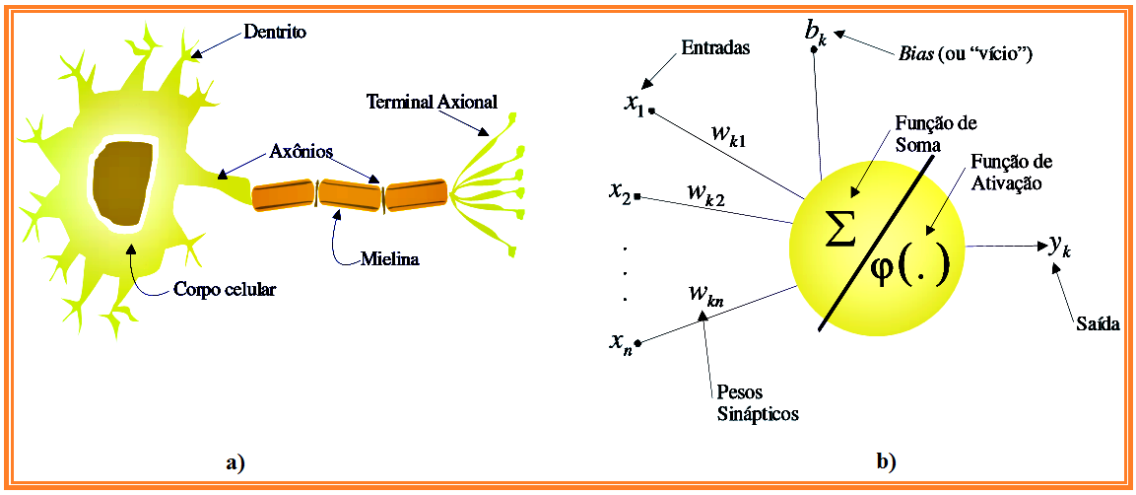

FIGURA 4. : a) Modelo de neurônio biológico b) Modelo de neurônio artificial. Fonte: Mota, (2007, p.25.)

Além dos desafios associados ao processo de otimização de parâmetros e determinação da estrutura da RNA na etapa de aprendizado supervisionado, outro desafio está associado à capacidade de generalização da rede. Assim, deve-se controlar a flexibilidade do mapeamento resultante sem que este seja mais complexo do que o necessário para melhorar o desempenho no treinamento (GEMAN et al., 1992 apud RAVIV e INTRATOR, 1999). A Figura 5 apresenta um modelo de RNA multicamadas.

No treinamento supervisionado, a RNA aprende com valores de entrada pré-classificados (padrões) para uma certa classe (saída padrão). O treino é dado, neste caso, a partir de uma referência externa, comparando a um alvo real, gerando um erro. Este erro é usado pela rede para o mapeamento dos valores de entrada para o aprendizado (ajustes dos valores de pesos e bias) de uma determinada classe. Aprender através de correção de erro é o mecanismo de aprendizagem mais utilizado (TAWIL,1999).

A RNA utilizada nos projetos 1 e 2 é do tipo ***feedforward***. Em seu treinamento,

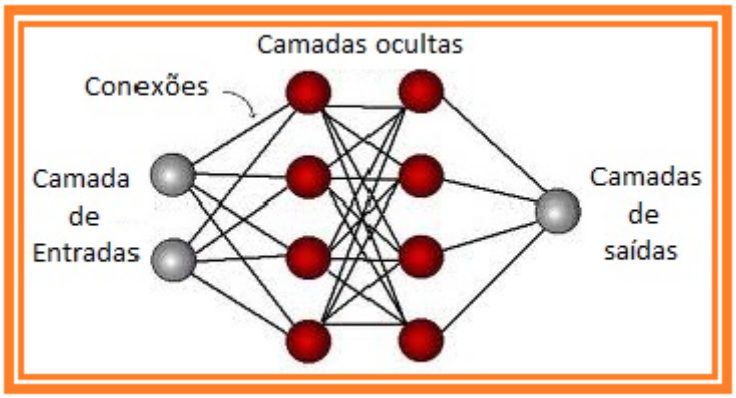

FIGURA 5. Topologia de uma RNA multicamadas. Fonte: Redes Neurais Artificiais Disponível em: http:www.icmc.usp.br/pessoas/andre/research/neural/

utiliza-se o algoritmo de aprendizagem *Levemberg-Marquartd*, na qual os valores de pesos e bias iniciais são atribuídos aleatoriamente. O número máximo de iterações foi estipulado em 3000, o gradiente mínimo de 10^{-20}, a falha na validação e checagem aceitável igual a 50 e o desempenho da rede é calculado através do erro médio quadrático (mse) utilizado como fitness dos AG's. Estes critérios foram definidos pelo autor de maneira intuitiva e levando-se em consideração a capacidade de processamento da máquina utilizada, para que o treinamento do pior caso (maior número de camadas ocultas e neurônios) não ultrapassasse o tempo de cinco minutos. Um conjunto de dados foi extraído do modelo pêndulo invertido e passou por um tratamento para ser utilizada no treinamento supervisionado das RNA's nos dois projetos.

6. RESULTADOS E DISCUSSÃO

O AG utilizado no Projeto 1 retornou à seguinte configuração de arquitetura da RNA: 7 camadas ocultas, 19 neurônios em cada camada oculta, todas com a função de ativação tangente sigmoide.

O erro médio quadrático (fitness) encontrado após o treino foi igual a 0,0697. A Tabela 3 mostra a evolução do erro, até encontrar o melhor indivíduo **n**.

Tabela 3. Evolução dos melhores indivíduos.

Indivíduo	n-10	n-9	n-8	n-7	n-6	n-5	n-4	n-3	n-2	n-1	n
Erro	22,71	2,41	1,66	1,51	1,33	1,30	1,09	1,09	0,99	0,96	0,07

Fonte: Próprio Autor.

A Figura 6 mostra a evolução dos melhores indivíduos através do gráfico.

O AG utilizado no Projeto 2 retornou a seguinte configuração de arquitetura da RNA: 5 camadas ocultas com 5 neurônios em cada camada, com a função de ativação tangente sigmoide.

A configuração do melhor indivíduo encontrado neste projeto é mostrada na

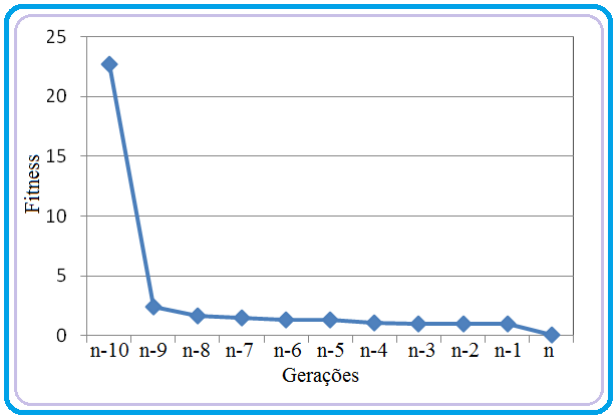

FIGURA 6. Gráfico da evolução dos melhores indivíduos. Fonte: Próprio Autor.

Tabela 4.

Tabela 4. Configuração do melhor indivíduo.

Camadas Ocultas			Função de Ativação			N° de Neurônios		
Camada 1	Camada 2	Camada 3	Camada 1	Camada 2	Camada 3	Camada 1	Camada 2	Camada 3
5	0	0	1	2	3	5	3	4

Fonte: Próprio Autor.

O indivíduo da Tabela 4 possui um total de 5 camadas ocultas (5+0+0), é composto pela função de ativação 1 (tangente sigmoide) e o primeiro conjunto de camadas possui 5 neurônios. Como nas camadas 2 e 3 é atribuído valor igual a zero, as outras características tornam-se sem efeito (células em vermelho). O erro médio quadrático encontrado após o treinamento foi igual a 0,8704.

Diferente do algoritmo proposto por Yao e Liu (1997), que representa a evolução da RNA por meio de pesos e da arquitetura, no presente trabalho, apenas a arquitetura da rede em conjunto com as funções de ativação serão repassadas para as próximas gerações. Novos pesos e bias são atribuídos para cada indivíduo gerado durante o treinamento e ao final fornece o valor de fitness (mse). Outro diferencial neste trabalho foi a otimização conseguida pelo algoritmo no parâmetro função de ativação que dentre quatro disponíveis, uma foi selecionada pelo AG.

Como no trabalho de Gutierrez (2005), pode-se observar que ao utilizar o algoritmo proposto (evolutivo), o número de gerações realizadas sobre a população é menor quando comparado à utilização de uma codificação direta, que faz uma busca de todas as combinações e arranjos possíveis. Esta dificuldade é encontrada no trabalho de Martins et al. (2015) no qual foi necessário testar todas as $2{,}49x10^{11}$ configurações de RNA para validar uma técnica de aprendizagem, além de ter como preocupação a configuração (arquitetura) da RNA para cada aplicação. No Projeto 1, são gerados no máximo 224 indivíduos (algoritmo evolutivo), sendo o número de arranjos igual a 343 indivíduos para técnica de codificação direta. No Projeto 2, são gerados no máximo 220 indivíduos (algo-

ritmo evolutivo), sendo o número de arranjos igual a 2299968000 indivíduos para técnica de codificação direta. No número de arranjos, contam-se até os indivíduos nulos, ou seja, se a primeira camada tiver o valor zero atribuído (primeiro bloco de três de camadas ocultas), não resultará em novos indivíduos (indivíduos válidos) ao mudar a função de ativação e o número de neurônios na camada oculta, pois esta camada não existe.

Como em Mekki e Chtourou (2012), a arquitetura automática de rede proposta mostrou-se útil na aproximação das não-linearidades desconhecidas do sistema dinâmico estudado.

Assim como em Rivero et. al (2008), o método de autoconfiguração deste trabalho destina-se a encontrar um conjunto de parâmetros úteis para qualquer problema e, portanto, não há necessidade de conhecimento especialista para este fim. Também é possível diferenciar as funções de ativação que não são relevantes para a resolução do problema em estudo, uma vez que estas não estão presentes na RNA configurada (não foram relevantes as funções de ativação: **base radial, linear e função logarítmica sigmoide)**.

Observa-se que mesmo sendo massas de dados diferentes, o número de camadas ocultas encontradas no presente trabalho (Projeto 1: 7 camadas ocultas; Projeto 2: 5 camadas ocultas), é semelhante à encontrada no trabalho de Cantú-Paz e Kamath (2005), problema Iris com 4 entradas, 5 camadas ocultas, 3 saídas e 80 epochs. Contudo, esta semelhança não dá a possibilidade de fazer comparações pontuais, pois o número de camadas ocultas e neurônios são fatores necessários para a descrição do comportamento da massa de dados e suas não-linearidades, e o aumento ou diminuição destes parâmetros não implica diretamente na melhora ou piora da performance da arquitetura de rede encontrada, ou seja, uma arquitetura com maior número de camadas ocultas ou neurônios não representa necessariamente o melhor resultado.

Não foi possível verificar se a ordem das funções de ativação seria relevante para o caso em estudo, uma vez que uma única função foi selecionada pelo algoritmo de otimização.

7. CONCLUSÃO

O AG mostrou-se satisfatório na configuração da RNA pois conseguiu atender aos critérios impostos no trabalho de Santos (2013), o qual destinou-se à aplicação da RNA para sintonia de um controlador de um sistema não-linear por ganho agendado, tendo retornado uma diferença nos valores de sobressinal entre o desejado e o obtido menor ou igual a 10% e a diferença do tempo de acomodação desejado e obtido menor ou igual a 0,25s. Mesmo não tendo a garantia desta ser a melhor configuração (por se tratar de um método heurístico), o método permitiu encontrar uma arquitetura que atendeu aos objetivos com o menor erro após o treinamento das arquiteturas analisadas. Desta forma, pode-se obter uma arquitetura sem que fosse fundamental o conhecimento de um especialista em relação às funções de ativação, à quantidade de camadas ocultas e ao número de neurônios destas camadas para configurar a RNA. A possibilidade de não precisar de um conhecimento especialista para determinar a função de ativação apropriada para a aplicação se destaca em relação aos trabalhos das literaturas apresentadas, em que se faz necessário a escolha desta, o que pode ser de difícil determinação para o especialista.

A utilização do AG para configuração da rede neural do tipo ***feedforward*** é aplicada em algumas literaturas recentes, porém com limitações na configuração da ar-

quitetura em relação ao número de camadas ocultas e neurônios. No presente trabalho amplia-se o uso desta ferramenta (AG para definição de arquitetura de RNA), em que toda a configuração da RNA é obtida através do algoritmo, inclusive as funções de ativação, que hoje não são escolhidas automaticamente.

Embora tenha sido considerado satisfatório o resultado obtido pelo AG na configuração da RNA, e tendo retornado a configuração de rede com o menor erro dentre as configurações presentes na otimização, o comando utilizado para a aplicação do algoritmo de aprendizagem train(net,x,y) aloca inicialmente um valor aleatório de pesos e bias, podendo retornar um valor significativamente diferente entre dois treinos consecutivos com o mesmo tipo de RNA e com as mesmas configurações, dificultando a avaliação de aptidão de indivíduos iguais ou similares.

8. REFERÊNCIAS BIBLIOGRÁFICAS

Cantú-Paz E.; Kamath C. (2005). An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems, IEEE Transactions on systems, Man and Cybernetics – Part B: Cybernetics, 915-927.

Fernandes, A. M. (2008) Inteligência Artificial: noções gerais. 3. ed. Florianópolis: Visual Books.

Golberg D.E., Holland J.H. (1988). Genetic algorithms and machine learning. Mach Learn, v. 3, 95–99.

Gutiérrez, G.; et al. (18-Oct-2005). Non-Direct encoding method based on cellular automata to design neural network architectures. Computing and Informatics, v. 24, 1001–1023.

Lian J.; Lee Y.; Sudhoff S. D.; Zak S. H. (March 2008). Self-Organizing radial basis function netwoek for real-time approximation of continuous-time dynamical systems, IEEE Transactions on neural network, v. 19, n. 3.

Liu G. P.; Kadirkamanathan V.; Billings S. A. (Feb, 1999). Variable neural networks for adaptive control of nonlinear systems, IEEE Trans. Trans. Syst. Man Cybern. B. Cybern., vol. 398, n. 1, 34-43.

Luger, G. F. (2004). Inteligência Artificial: estruturar e estratégias para solução de problemas complexos. 4 ed. Porto Alegre: Bookmann.

Martins, E. R.; et al. Configuração de redes neurais artificiais para prognose da produção de povoamentos clonais de eucalipto. Revista Brasileira de Ciências Agrárias, v. 10, n. 4, p. 532–537, 2015.

Matworks. Multilayer neural network architecture. Disponível em: <https://www.mathworks.com/help/deeplearning/ug/multilayer-neural-network-architecture.html>. Acesso em: 24 de abril de 2019.

Mekki H.; Chtourou M.; Derbel N. (2006). Variable structure neural networks for adaptive control of nonlinear systems using the stochastic approximation, Simulation Modeling Practice and Theory 14, 1000-1009.

Mekki H.; Chtourou M. (2012). Variable structure neural networks for real time

approximation of continuous time dynamical systems using evolutionary artificial potential fields. Wseas Transactions on Systems, n. 2, 75–84.

Michalewicz, W. (1999). Genetic Algorithms + Data Structures = Evolution Programs. 3.ed. New York: Springer.

Mitchell, M, Taylor, C. E. (1999). Evolutionary computation: An overview. Annual Review of Ecology and Systematics, v.30, 593-616.

Mota, J. F. (2007). Um estudo de caso para a determinação do preço de venda de imóveis urbanos via redes neurais artificiais e métodos estatísticos multivariados. Dissertação (Métodos Numéricos em Engenharia). Universidade Federal do Paraná, Curitiba-PR.

Raviv, Y. Intrator, N. (1999). Variance reduction via noise and bias constraints. In: Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems. London, Springer-Verlag, 163-175.

Rivero, D; et al. (2008). Artificial neural network development by means of genetic programming with graph codification. World Academy of Science, Engineering and Technology, v. 21, 880–885.

Santos, E. S. (2013). Projeto e construção de um controlador adaptativo por realimentação de estados de um pêndulo invertido utilizando inteligência computacional. Dissertação (Mestrado em Pesquisa Operacional e Inteligência Computacional) UCAM, Campos dos Goytacazes-RJ.

Schuma, C. D.; Birdwell, J. D. (2013). Variable structure dynamic artificial neural networks. Biologically Inspired Cognitive Architectures. n. 6, 126–130.

Tawil M. (1999). Künstliche neuronale netze - methode und anwendung. IMW Institutsmittelung, n. 24, 69-72.

Universidade de São Paulo. Redes Neurais Artificiais. Disponível em: <http://www.icmc.usp.br/pessoas/andre/research/neural/>. Acesso em: 22 de abril de 2019.

Yao, X; Liu, Y. (1997). A new evolutionary system for evolving artificial neural networks. IEEE Transactions on Neural Networks, v. 8, n.3, 694-713.