Usando *NeuroEvolution of Augmenting Topologies (NEAT)* para jogar Super Mario World

Alexandre de C. Araújo[1], **Artur A. Silva**[2]

[1]Laboratório de Midias Interativas - LabMINT - Universidade Federal do Maranhão (UFMA)
65.085–580 - São Luís - MA - Brasil

[2]Núcleo de Computação Aplicada - NCA - Universidade Federal do Maranhão (UFMA)

alexcararaujo@yahoo.com.br, astobiro@gmail.com

Abstract. *In Artificial Intelligence, the use of genetic algorithms to evolve neural networks is called NeuroEvolution. In this paper, we study NeuroEvolution of Augmenting Topologies, an efficient method in this area that solves a lot of problems faced by similar algorithms with a simple but powerful approach that greatly increases its efficiency. We use Super Mario World as a training environment because of the world complexity it presents. The results found show that it is possible to train a NEAT to play Super Mario World for two levels.*

Resumo. *Na área da Inteligência Artificial, o uso de algoritmos genéticos para evoluir uma rede neural é chamado Neuro-Evolução. Neste artigo, estudamos o NeuroEvolution of Augmenting Topologies, um método eficiente neste ramo que resolve muitos problemas enfrentados por algoritmos semelhantes com uma abordagem simples mas poderosa que aumenta bastante sua eficácia. Usamos o Super Mario World como ambiente de treinamento pela complexidade de mundo que ele apresenta. Os resultados encontrados mostram que é possível treinar um NEAT para jogar Super Mario World em duas fases.*

1. Introdução

Inteligência Artificial (IA) é uma área de grande interesse atualmente, com pesquisas de variadas complexidades buscando sempre inovar. Dentro desta área um assunto muito estudado é o uso de algoritmos genéticos no treinamento de IA, pois eles possibilitam o aprendizado de diversos conhecimentos por um computador. O uso de jogos como ambiente de aprendizado vem também sendo muito discutido pois eles são uma boa forma de modelar mundos em um espaço virtual e por esse motivo, uma boa forma de medir o desempenho e capacidade de algoritmos [Laird and VanLent 2001]. Este trabalho vai então estudar o *NeuroEvolution of Augmenting Topologies* [Stanley and Miikkulainen 2002] como método de evolução. Para o teste do método, será utilizado o jogo *Super Mario World*.

2. Materiais e Métodos

A técnica utilizada para resolver o problema foi o *NeuroEvolution of Augmenting Topologies* (NEAT). O NEAT é um algoritmo genético para gerar redes neurais artificiais. Os experimentos foram feitos em Lua utilizando a ferramenta BizHawk como ambiente de aprendizado, simulação e extração de dados. Estes testes foram realizados em uma ROM de *Super Mario World* para o *Super Nintendo Entertainment System*.

3. *NeuroEvolution of Augmenting Topologies*

Os algoritmos de geração de redes neurais artificiais genéticos normalmente usam uma topologia fixa e fazem evolução dos pesos. O NEAT utiliza uma abordagem diferente, ele tenta resolver alguns problemas que outros *Topology and Weight Evolving Artificial Neural Networks* (TWEANNs) encontram, sendo esses a codificação, *competing conventions*, proteção de inovação, população inicial e inovação topológica.

3.1. Codificação Genética

O NEAT utiliza codificação genética. Essa codificação é projetada de forma a facilitar o alinhamento entre genes no cruzamento de genomas. Os genomas são uma representação linear da conectividade da rede como demostrado na Figura 1 Cada genoma tem um lista de *connection genes*, onde cada conexão representa a ligação entre dois *node genes*. Cada *node gene* tem uma lista de entradas, nos escondidos e saídas que podem ser conectadas. Cada *connection gene* especifica o *in-node*, o *out-node*, o peso da conexão, o estado da conexão (habilitada ou desabilitada) e um número de inovação, que permite achar genes correspondentes.

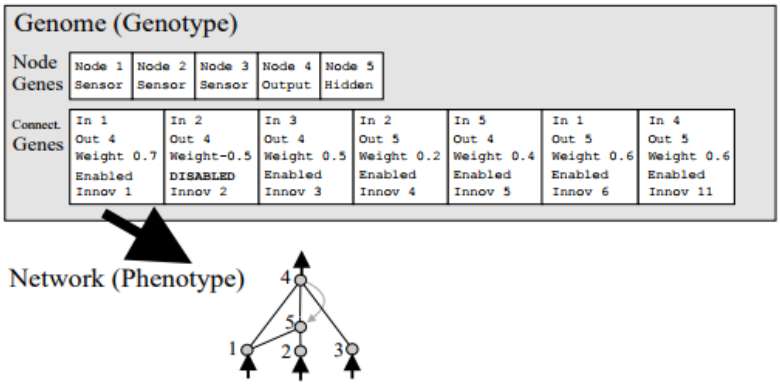

Figura 1. Genoma

A mutação pode alterar tanto o peso de uma conexão, com possibilidade de alteração ou não do peso à cada geração, como a topologia da rede. A mutação da topologia pode ocorrer de 2 maneiras, chamadas de *add connection* e *add node*. Na *add connection* uma nova conexão entre dois *gene nodes* não conectados é criada com peso aleatório. Na *add node* uma conexão já existente é desativada e 2 novas conexões são geradas, uma conexão com peso 1 que tem como entrada a mesma entrada da conexão desativada e um novo *gene node* como saída e uma conexão com o peso da conexão desativada que tem como entrada o novo *gene node* e a saída da conexão desativada. Esse processo é exemplificado na Figura 2.

Essa abordagem evita problemas que outros TWEANNs encontram ao usar outros tipos de codificação. A codificação binaria por exemplo é uma strings de bits usada em uma matriz de conexões pra representar uma rede. Essa codificação gera problemas. Primeiro, esta codificação limita o numero de conexões ao quadrado do numero de nós, segundo, existe a dependência de um operador humano que deve decidir o tamanho máximo da string, que deve ser igual para todos os organismos. O mesmo problema

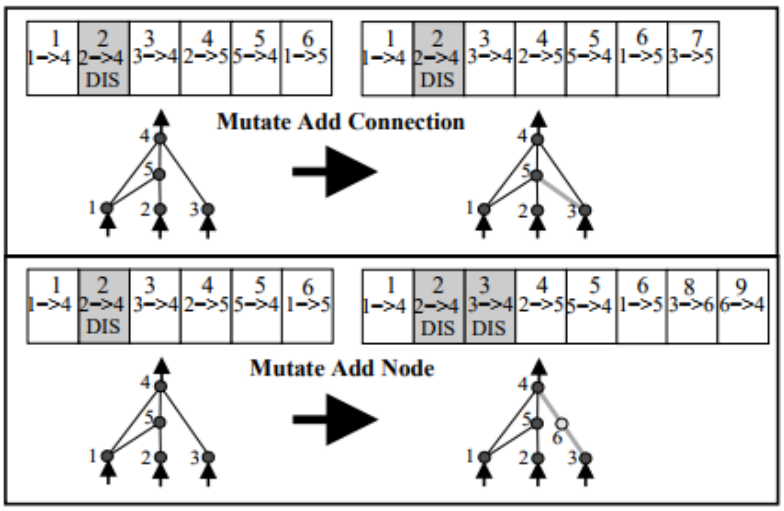

Figura 2. Mutações de topologia

pode ser encontrado na representação com grafos, onde o tamanho limite da rede é limitado pela escolha de um operador humano. Apesar da representação linear do NEAT, as adaptações feitas conseguem prevenir os problemas citados.

3.2. *Competing Conventions*

Um dos maiores empecilhos na Neuro-Evolução é o *Competing Conventions Problem*, também conhecido como problema de permutação. Como demostrado na Figura 3 o cruzamento de 2 soluções para um problema não necessariamente geram um bom descendente se eles não tiverem a mesma codificação. Outra forma de *competing conventions* são as soluções com diferentes topologias e com tamanhos de genoma diferentes.

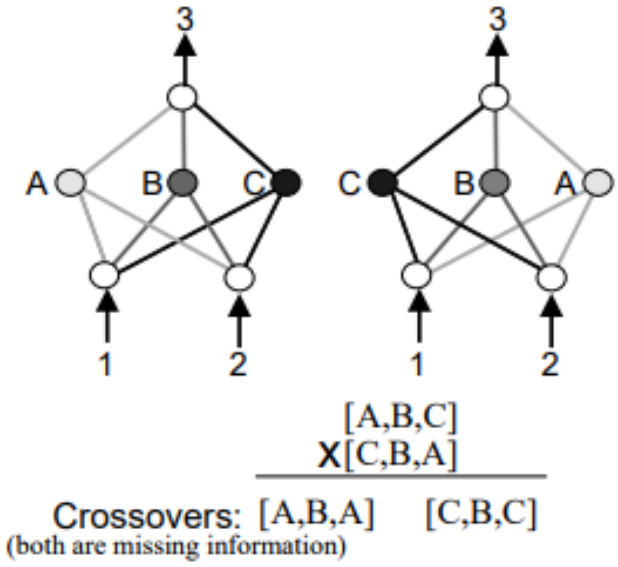

Figura 3. Problema de permutação

o NEAT resolve este problema fazendo *tracking* da origem dos genes através de um numero de inovação. A ideia é que 2 genes com a mesma origem histórica devem

representar a mesma estrutura, mas com pesos diferentes já que ambos são derivados de um mesmo gene. Toda vez que um novo gene aparece, um valor global de inovação é incrementado e atribuído a esse novo gene. A possibilidade de que uma inovação estrutural receba mais de um numero de inovação é evitada ao manter uma lista de todas as inovações na geração atual, assim quando uma mesma inovação aparecer mais de uma vez, ela sera marcada com o mesmo valor.

Esse *tracking* da origem dos genes permite que o NEAT, no cruzamento, consiga alinhar genes com mesma origem em ambos os genomas. Esses gene são chamados de *matching genes*. Aqueles que não são alinhados são chamados de *excess* ou *disjoint* dependendo de sua posição nos genomas cruzantes. Eles representam estruturas presentes em um dois pais e não no outro. No cruzamento, *matching genes* são escolhidos aleatoriamente entre os pais para serem passados ao descendente enquanto os *excess* e os *disjoint* são todos passados do parente mais apto. Este processo é demostrado na Figura 4.

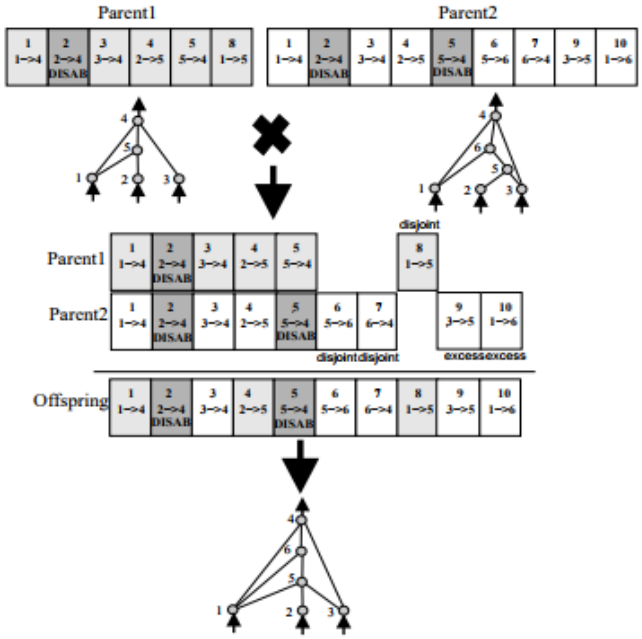

Figura 4. *crossover* no NEAT

3.3. Proteção de Inovação

Ao sofrer mutação, uma estrutura geralmente perde desempenho momentaneamente e pode ser descartada de forma precoce. O método que o NEAT aplica nesta situação é de especiar os genomas, para que inovações topológicas compitam em um nicho de indivíduos parecidos, dando assim a oportunidade para as novas estruturas se desenvolverem antes de competirem com o a população total. Para fazer uma medição da compatibilidade entre genomas e poder classificar sua especie, o número de inovação é usado.

O número de *disjoints* e *excess* entre genomas é um medidor natural de sua compatibilidade. Quanto maior o número, menos compatíveis 2 genomas são. A partir dessa preposição, podemos então medir a compatibilidade δ entre 2 genomas como uma combinação linear do número de *disjoints* D e *excess* E e a média da diferença de peso

dos *matching genes* $\bar{\mathrm{W}}$, incluindo os genes desabilitados.

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \bar{W}$$

Os coeficientes c_1, c_2, c_3 permitem ajustar a importância dos 3 fatores enquanto o fator *N*, o numero de genes no maior genoma, normaliza para o tamanho do genoma. A distancia δ permite especiar usando um limiar de compatibilidade. Uma lista de espécies pode então ser mantida. Cada espécie é representada por um genoma aleatório da espécie da geração anterior. Um genoma *g* da geração atual é colocado na primeira espécie em que *g* e o representante da espécie são compatíveis. Se *g* não for compatível com nenhuma espécie, uma nova é criada tendo *g* como representante.

O mecanismo de reprodução utilizado pelo NEAT é *explicit fitness sharing* [Goldberg et al. 1987], onde organismos da mesma especie compartilham o *fitness* de seu nicho. Assim, nenhuma espécie consegue dominar a população mesmo que muitos de seus tenham boa performance. O *fitness* ajustado *f* para o organismo *i* é calculado a partir de sua distancia δ para cada outro organismo na população:

$$f = \frac{f_i}{\sum_{j=1}^{n} fc(\delta(i,j))}$$

A função de compartilhamento *fc* é 0 se a distancia δ(i,j) for maior que o limiar δ, senão é 1. Assim $\sum_{j=1}^{n} fc(\delta(i,j))$ é reduzido ao número de organismos na mesma espécie que *i*. A cada espécie é atribuído um número de descendentes proporcional à soma dos *fitness* ajustados *f*. As espécies então eliminam o membro com menor performance da população e toda a população é substituída pelos descendentes dos membros restantes de cada espécie.

3.4. Minimização da dimensionalidade

A maioria das TWEANNs tem população inicial com topologias aleatórias para introduzir diversidade desde o começo. Em contraponto, o NEAT inclina a busca para uma dimensionalidade miníma espacial ao iniciar todos os membros com uma população uniforme de redes sem camadas escondidas, onde as entradas são diretamente ligadas à saída. Novas estruturas vão sendo adicionadas a partir das mutações, mas somente aquelas julgadas úteis através do *fitness* se mantém. Ao partir da forma mais simples possível, Sem camadas escondidas, e somente adicionando aquilo que é necessário, o NEAT busca soluções num espaço mínimo, melhorando a performance do método.

4. Testes Realizados

Para modelar a estrutura da rede proposta pelo NEAT, precisamos definir o terreno e inimigos do jogo como as entradas e os 8 botões básicos como saídas (Cima, Baixo, Esquerda, Direita, X, Y, A e B). Quando o NEAT gera a ligação da entrada com um botão este é pressionado enquanto houverem entradas ligadas a ele. O *fitness* é definido como a distancia percorrida, quanto mais longe maior o *fitness*. É dado também um tempo de *timeout*, quando nenhuma ação é feita por um determinado tempo, a execução daquele genoma é parada. Os parâmetros utilizados foram de 300 para a população inicial, 2 para o peso de *disjoint*, e 1 como limiar de δ. chance de mutação *add connection* foi de 25% e de mutação *add node* foi de 50%.

Foram realizados dois testes distintos para a validação dos resultados. No primeiro teste, a rede era treinada na fase *Yoshi's Island 2* quando chegava em uma geração que conseguia completar a fase, essa rede treinada era então colocada para tentar a fase *Yoshi's Island 1*, se não conseguisse, era treinada para passar dela. Por fim a rede treinada nas duas fases era colocada para jogar a fase *Yoshi's Island 1* para poder confirmar que o conhecimento persistia mesmo ela treinando em outra fase diferente. Chamaremos esse teste de *TesteFD*.

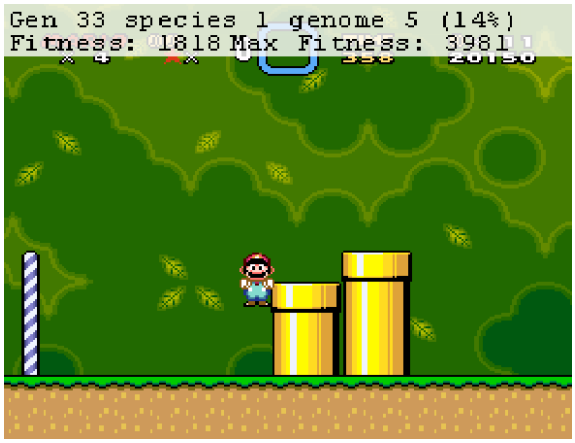

Figura 5. *Teste DF* na fase *Yoshi's Island 2*

O segundo teste era da mesma forma trocando somente a fase inicial para *Yoshi's Island 1*. Chamaremos esse teste de *Teste DF*. Os testes foram realizados dessa forma para poder visualizar o comportamento da rede, a velocidade de aprendizado, quando treinada inicialmente em uma fase mais difícil (*Yoshi's Island 1*) e então colocada para jogar uma mais fácil (*Yoshi's Island 2*), e treinada inicialmente em uma fase mais fácil e então colocada para jogar em uma mais difícil.

5. Resultados e Discussões

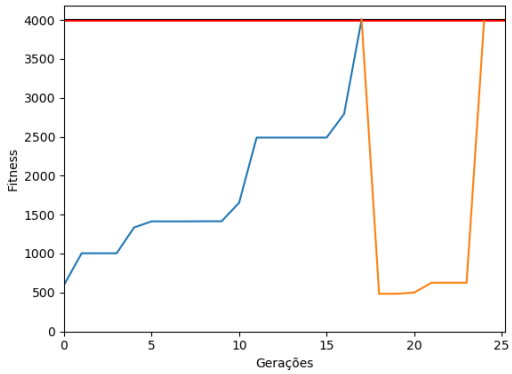

Figura 6. Curva de aptidão para o *Teste FD*

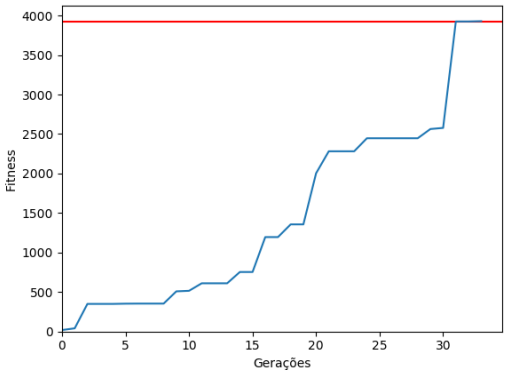

Figura 7. Curva de aptidão para o *Teste DF*

O *Teste FD* conseguiu passar da fase *Yoshi's Island 2* em 16 gerações com um fitness de 4006, e com essa rede, não conseguiu passar da fase *Yoshi's Island 1*. Foi

colocado então para treinar, e após 8 gerações conseguiu passar com um fitness de 3990. Já o *Teste DF* conseguiu passar da fase *Yoshi's Island 1* em 33 gerações com um fitness de 3925 e conseguiu passar da fase *Yoshi's Island 1*.

6. Conclusões

Este trabalho detalha a técnica *NeuroEvolution of Augmenting Topologies* e descreve como a utilizar para jogar *Super Mario World* em diversas fases. Os resultados apresentados mostram que é possível usar o NEAT em *Super Mario World* em duas fases, mantendo o conhecimento aprendido no treinamento destas duas fases.

Como trabalhos futuros, reproduzir o trabalho em um maior número de fases para verificar se o NEAT consegue jogar *Super Mario World* em sua totalidade. E também comparar estes resultados com outras abordagens para assim medir a eficacia da técnica utilizada.

Referências

[Goldberg et al. 1987] Goldberg, D. E., Richardson, J., et al. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum.

[Laird and VanLent 2001] Laird, J. and VanLent, M. (2001). Human-level ai's killer application: Interactive computer games. *AI magazine*, 22(2):15.

[Stanley and Miikkulainen 2002] Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127.

[TASvideos] TASvideos. Bizhawk. `http://tasvideos.org/BizHawk.html`. Accessed: 24/07/2017.