UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ARTHUR LEONARDO DE ALENCAR PAULINO

CENEAT: NEUROEVOLUÇÃO DE TOPOLOGIAS AUMENTANTES COM MELHORIAS CULTURAIS

FORTALEZA

2018

ARTHUR LEONARDO DE ALENCAR PAULINO

CENEAT: NEUROEVOLUÇÃO DE TOPOLOGIAS AUMENTANTES COM MELHORIAS CULTURAIS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Inteligência Artificial

Orientador: Prof. Dr. João Paulo Pordeus Gomes

FORTALEZA

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

P353c Paulino, Arthur Leonardo de Alencar.
ceNEAT: Neuroevolução de Topologias Aumentantes com Melhorias Culturais / Arthur Leonardo de Alencar Paulino. – 2018.
59 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2018.
Orientação: Prof. Dr. João Paulo Pordeus Gomes.

1. Neuroevolução. 2. Topologias Aumentantes. 3. Cultura. I. Título.

CDD 005

ARTHUR LEONARDO DE ALENCAR PAULINO

CENEAT: NEUROEVOLUÇÃO DE TOPOLOGIAS AUMENTANTES COM MELHORIAS CULTURAIS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Inteligência Artificial

Aprovada em: 27 de Novembro de 2018

BANCA EXAMINADORA

Prof. Dr. João Paulo Pordeus Gomes (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. João Fernando Lima Alcântra
Universidade Federal do Ceará (UFC)

Prof. Dr. César Lincoln Cavalcante Mattos
Universidade Federal do Ceará (UFC)

Prof. Dr. Ajalmar Rego Rocha Neto
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Dedico este trabalho à minha esposa Larissa Lorena Gomes Paulino e à família que estamos construindo.

AGRADECIMENTOS

Aos meus pais Francisco de Assis Paulino e Maria Divandete Peixoto de Alencar Paulino e ao meu irmão André Wagner de Alencar Paulino pelo apoio, principalmente nas horas mais difíceis.

À minha esposa Larissa Lorena Gomes Paulino pela inspiração e por todo o trabalho realizado ao meu lado.

Ao Prof. Dr. João Paulo Pordeus Gomes por ter me orientado na elaboração desta dissertação.

Ao Prof. Dr. Yuri Lenon Barbosa Nogueira por trazer à tona detalhes importantes para tornar o seguinte estudo mais interessante.

À Dra. Ananda Freire e ao Me. Paulo Bruno pela atitude prestativa e encorajadora.

Ao Prof. Dr. Carlos Eduardo Fisch de Brito pela inspiração em diversos aspectos.

Aos Me. Luiz Alberto do Carmo Viana e Hugo Carvalho de Paula, aos Ba. Francisco José Lins Magalhães, Pedro Rocha Muniz e Vanilson Nogueira de Azevedo pelas valorosas companhias.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

“Não tente pisar nas pegadas dos sábios. Busque o que eles buscaram.”

(Matsuo Bashō)

RESUMO

O conhecimento transmitido entre gerações por meios não-genéticos pode ser entendido como cultura. A capacidade dos indivíduos de uma espécie de ensinar e aprender exerce um papel fundamental no direcionamento do processo evolutivo. Este trabalho tem por objetivo elencar e avaliar formas de enriquecer a Neuroevolução de Topologias Aumentantes com processos de aprendizado. Os parâmetros envolvidos na análise contemplam os métodos *Backpropagation* e *Extreme Learning Machine*, os indivíduos a serem ensinados, o momento a partir do qual a cultura passa a se manifestar no sistema e a natureza das lições a serem aprendidas. Os resultados mostram que o enriquecimento cultural, bem como algumas das variações nos parâmetros propostos, acelera a convergência do processo evolutivo.

Palavras-chave: Neuroevolução. Topologias Aumentantes. Cultura.

ABSTRACT

Knowledge transmitted between generations by non-genetic means can be understood as culture. The capacity of individuals from a certain species to teach and learn plays a fundamental role in directing the evolutionary process. The goal of this work is to list and evaluate ways of enhancing the Neuroevolution of Augmenting Topologies with learning processes. The parameters involved in the analysis contemplate the methods *Backpropagation* and *Extreme Learning Machine*, the individuals to be taught, the moment from which culture manifests in the system and the nature of the lessons to be learned. The results show that cultural enhancement, as well as some of the variations on the proposed parameters, accelerates the evolutionary process' convergence.

Keywords: Neuroevolution. Augmenting Topologies. Culture.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BPG	*Backpropagation*
ceNEAT	*Culturally Enhanced Neuroevolution of Augmenting Topologies*
ELM	*Extreme Learning Machine*
NEAT	*Neuroevolution of Augmenting Topologies*

LISTA DE SÍMBOLOS

θ	Viés de um perceptron
w	Vetor de pesos para as entradas de um perceptron
x	Uma entrada para a Rede Neural
σ	Função sigmóide
o	Função de saída de um perceptron
α_{BPG}	Taxa de aprendizagem do algoritmo BPG
ε	Função dos perceptrons no algoritmo ELM
Ψ	Solução para o viés e os pesos de um perceptron no algoritmo ELM
α_{ELM}	Taxa de aprendizagem do algoritmo ELM
δ	Distância entre dois cromossomos no algoritmo NEAT
c_1	Fator multiplicativo da quantidade de conexões excessivas no cálculo de δ
c_2	Fator multiplicativo da quantidade de conexões disjuntas no cálculo de δ
c_3	Fator multiplicativo da diferença média entre os pesos das conexões correspondentes no cálculo de δ

SUMÁRIO

1 INTRODUÇÃO

A busca pela inspiração na natureza para a resolução de problemas é uma iniciativa recorrente entre os estudiosos de Inteligência Artificial. Porém, para que esta prática obtenha sucesso, uma pergunta precisa ser feita: quais os mecanismos que regem a inteligência na natureza? Na tentativa de responder a este questionamento, surgem modelos computacionais interessantes.

As Redes Neurais Artificiais, também designadas neste trabalho por Redes Neurais, são inspiradas no funcionamento do cérebro e têm como unidade básica o perceptron, que é uma abstração matemática do neurônio biológico (MCCULLOCH; PITTS, 1988). Devido aos avanços na Ciência e na Engenharia da Computação, as Redes Neurais Artificiais têm ganhado bastante espaço na comunidade científica e atualmente são utilizadas para atacar vários tipos de problemas, alcançando altos níveis de assertividade na literatura (GOODFELLOW *et al.*, 2016).

Já os Algoritmos Genéticos advém da tentativa de simular o processo de evolução natural das espécies no computador como forma de encontrar as melhores soluções para um problema. A chave para esta tarefa está na representação digital do cromossomo, com o qual são realizadas operações documentadas pela Biologia com o objetivo de fazer uma população de indivíduos artificiais evoluir, geração após geração, para encontrar o indivíduo capaz de realizar a tarefa com aptidão satisfatória (BEASLEY *et al.*, 1993; MELANIE, 1999).

Ao combinar essas duas possibilidades, Redes Neurais e Algoritmo Genético, surgem os Algoritmos Neuroevolutivos (LEHMAN; MIIKKULAINEN, 2013). A porta para esta ideia é que os indivíduos sejam definidos como representações de Redes Neurais, as quais são avaliadas pela função de aptidão. Nesse contexto, surgem perguntas relacionadas à forma de representar Redes Neurais com cromossomos, à implementação de cruzamentos para gerar outros indivíduos de forma consistente e sobre as formas de provocar mutações para sanar o aprisionamento em ótimos locais. As redes neurais teriam topologias fixas? Como garantir algum nível de eficiência?

Assim surgiu o algoritmo Neuroevolução de Topologias Aumentantes – em inglês *Neuroevolution of Augmenting Topologies* (NEAT) – com resultados superiores aos da literatura no momento em que foi criado (2002). Pela sua simplicidade e eficiência, serviu de inspiração para diversas variações, tais como rtNEAT (STANLEY *et al.*, 2005), HyperNEAT (STANLEY *et al.*, 2009), cgNEAT (HASTINGS *et al.*, 2009), odNEAT (SILVA *et al.*, 2015) e CoDeepNEAT (MIIKKULAINEN *et al.*, 2017), cada uma com sua especificidade em termos de aplicação. Nenhuma delas, no entanto, leva em consideração a *cultura* da população através das gerações.

Ao se observar certos tipos de animais, é possível notar que alguns padrões comportamentais são aprimorados no decorrer de suas vidas devido ao fato dos mais jovens observarem como os mais velhos executam determinadas tarefas. Por exemplo, existem algumas espécies de pássaros nas quais os mais jovens aprendem novas melodias ouvindo os adultos; filhotes que aprendem a caçar são comuns entre os mamíferos predadores; primatas aprendem regras de convivência observando os mais velhos ou até mesmo por ensinamentos diretos (DAWKINS, 1976). Daí a inspiração para inserir cultura em Algoritmos Neuroevolutivos.

Nessa linha de pesquisa, Nolfi e Parisi (NOLFI *et al.*, 1990) observaram que estavam criando melhores aprendizes do que agentes. Cecconi (CECCONI *et al.*, 1995) mostrou que fazer os mais jovens aprenderem com seus pais pode melhorar a convergência dos Algoritmos Neuroevolutivos. Sasaki e Tokoro (SASAKI; TOKORO, 1997) mostraram que a habilidade de aprender pode ser mais importante do que a habilidade de executar. Paul H. McQuesten (MCQUESTEN, 2002) mostrou os benefícios de ter o indivíduo mais apto da geração como o tutor dos recém-nascidos.

Nesse contexto, o objetivo geral do presente trabalho é avaliar os impactos do enriquecimento cultural no algoritmo NEAT e propor variações na forma de se implementar a transmissão de conhecimento entre as gerações. Os objetivos específicos deste trabalho são *i*) validar cada proposta de variação na inserção de cultura no NEAT para, possivelmente, *ii*) elencar recomendações gerais para a execução do algoritmo proposto no Capítulo 3.

1.1 Estrutura da dissertação

O Capítulo 2 contém o embasamento teórico necessário para a construção da proposta, apresentada no Capítulo 3. O Capítulo 4 expõe os experimentos realizados e seus resultados para que as devidas conclusões sejam tecidas no Capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

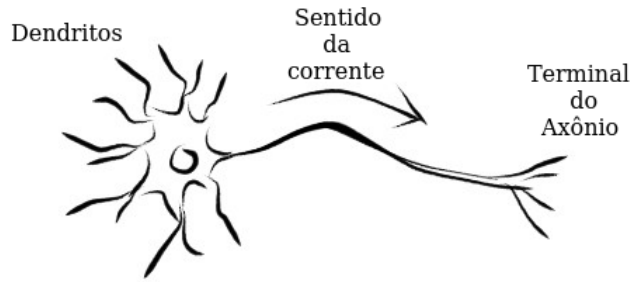
Este capítulo aborda os principais conceitos necessários para o desenvolvimento da dissertação, formalizando o suporte para que os próximos capítulos sejam construídos com uma linguagem de nível mais alto.

A Seção 2.1 apresenta as Redes Neurais e como elas aprendem. A Seção 2.2 introduz o Algoritmo Genético, meta-heurística utilizada para a criação de agentes inteligentes em contextos nos quais é difícil expressar as regras de tomada de decisão explicitamente e a Seção 2.3 apresenta a Neuroevolução de Topologias Aumentantes.

2.1 Redes Neurais

A Rede Neural é inspirada no sistema nervoso dos animais, cuja unidade básica é o neurônio (ROJAS, 1996). Os neurônios são células que, se suficientemente estimuladas, transmitem impulsos elétricos para outras células. A Figura 1 mostra uma representação simplificada de um neurônio.

Figura 1 – Neurônio



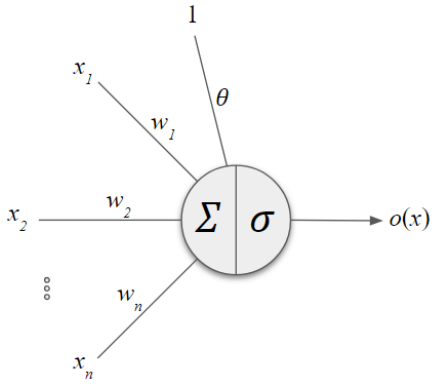
Fonte: Elaborado pelo autor

O neurônio recebe os estímulos através dos dendritos e propaga o sinal pelo terminal do axônio para outro neurônio ou para outro tipo de célula. As capacidades do sistema nervoso advém da forma que os neurônios estão dispostos e conectados, normalmente formando redes bastante complexas como por exemplo o cérebro humano. Para simular uma estrutura como o cérebro no computador, o primeiro passo foi criar uma abstração para o neurônio chamada perceptron (MCCULLOCH; PITTS, 1988).

Em uma descrição enxuta, um perceptron é a composição de um somatório s e uma função de propagação. As parcelas do somatório são as componentes da entrada x ponderadas pelas componentes do vetor w, além de um termo independente θ chamado *viés*, ou seja, $s = \theta + wx$. Uma função de propagação é comumente usada para suavizar a curva da saída e

nesta dissertação será utilizada a função sigmóide $\sigma(y) = 1/(1+e^{y})$. Assim, o valor da saída o do perceptron é calculado com a fórmula $o(x) = \sigma(s(x)) = \sigma(\theta + wx)$. A Figura 2 mostra a representação do perceptron utilizado neste trabalho.

Figura 2 – Perceptron



Fonte: Elaborado pelo autor

2.1.1 Perceptrons em Múltiplas Camadas

Normalmente, para se construir uma Rede Neural artificial propriamente dita, faz-se uso de vários perceptrons dispostos em camadas. Os perceptrons da primeira camada são chamados de sensores. A entrada é processada pelas camadas ocultas, que por sua vez alimentam a camada de saída. Outra configuração comum é que na comunicação entre as camadas, cada perceptron tenha sua saída conectada a todos os perceptrons da camada seguinte. A Figura 3 mostra a arquitetura típica de uma Rede Neural Artificial.

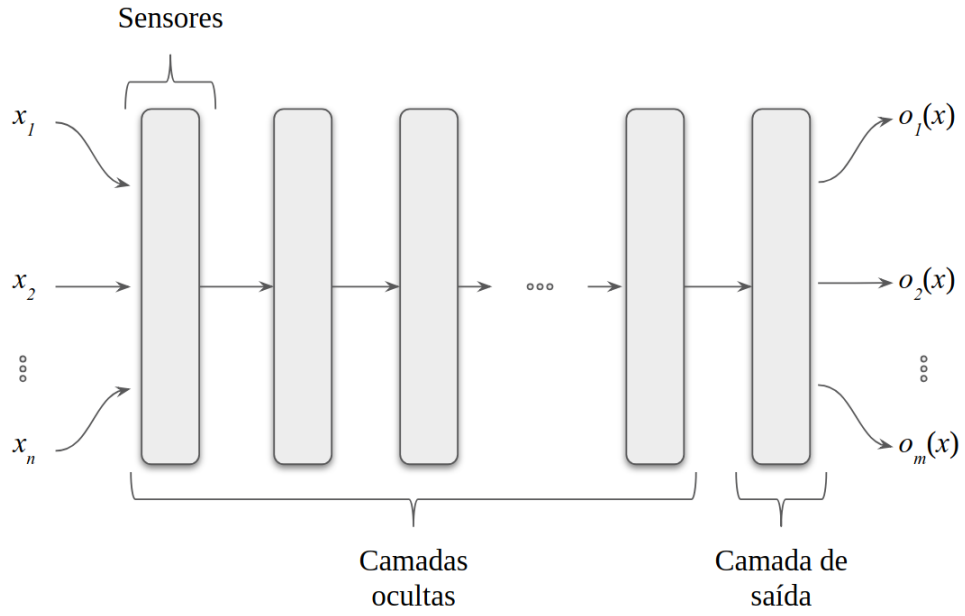
Existem ainda as Redes Neurais Recorrentes, as quais podem conter ciclos nos caminhos das arestas direcionadas que conectam os perceptrons. Tais redes não foram utilizadas neste trabalho por motivos que serão esclarecidos na primeira seção do capítulo 4.

A variabilidade de funções que uma Rede Neural pode computar depende da topologia e dos pesos das arestas nas entradas dos perceptrons. A seguir, serão apresentadas formas de ajustar os pesos e vieses de uma Rede Neural para modelar o comportamento de uma função.

2.1.2 Treinamento

Uma das utilidades práticas de uma Rede Neural é simular o comportamento de uma função cuja definição formal é desconhecida, partindo de um conjunto de entradas e as saídas correspondentes. Nesta dissertação serão utilizados dois algoritmos diferentes: *Backpropagation*

Figura 3 – Rede Neural



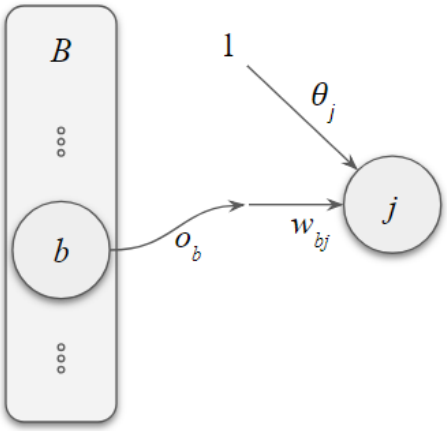
Fonte: Elaborado pelo autor

(BPG) e *Extreme Learning Machine* (ELM).

2.1.2.1 Backpropagation

Trata-se de um método que utiliza o gradiente descendente para diminuir as disparidades entre as saídas da Rede Neural e as da função alvo para as mesmas entradas. Formalmente, sejam x uma entrada para a Rede Neural, $\bar{O}(x)$ a saída para x e $\bar{T}(x)$ a saída alvo que se deseja modelar. $\bar{O}(x)$ e $\bar{T}(x)$ são vetores de dimensões iguais. Seja também $\bar{E}$ a função de erro definida segundo a equação $\bar{E}(x) = \frac{1}{2}\|\bar{O}(x) - \bar{T}(x)\|^2_{F_b}$, onde o subscrito $_{F_b}$ indica a norma Frobenius. O algoritmo BPG minimiza $\bar{E}(x)$ ao modificar os pesos da Rede Neural. Como x estará fixo nos passos a seguir, este será omitido da notação. Sejam, então, $O = \bar{O}(x)$, $T = \bar{T}(x)$ e $E = \bar{E}(x)$.

Figura 4 – Perceptrons b e j



Fonte: Elaborado pelo autor

O algoritmo BPG pode ser dividido em duas fases: *i*) encontrar o gradiente de E em relação aos pesos das arestas e ao viés de cada perceptron e *ii*) atualizar tais valores, movendo-se no sentido contrário ao do gradiente encontrado. Então, como representado na Figura 4, sejam j um perceptron, θ_j o viés de j, B o conjunto de perceptrons que alimentam j, o_b a saída do perceptron b e w_{bj} o peso da aresta que liga b a j. Tem-se que:

$$s_j = \theta_j + \sum_{b \in B} o_b w_{bj}$$

i) Calcular $\partial E/\partial w_{ij}$ e $\partial E/\partial \theta_j$

Aplicando a regra da cadeia em $\partial E/\partial w_{ij}$ tem-se:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_j}\frac{\partial s_j}{\partial w_{ij}}$$

Expandindo $\partial s_j/\partial w_{ij}$:

$$\frac{\partial s_j}{\partial w_{ij}} = \frac{\partial\left(\theta_j + \sum_{b \in B} o_b w_{bj}\right)}{\partial w_{ij}} = \frac{\partial(\theta_j + o_\xi w_{\xi j} + \cdots + o_i w_{ij} + \cdots + o_\zeta w_{\zeta j})}{\partial w_{ij}}$$
$$= \frac{\partial(o_i w_{ij})}{\partial w_{ij}} \overset{*}{=} o_i\left[\frac{d}{dw_{ij}} w_{ij}\right] = o_i$$

* o_i é constante no contexto em que a entrada da rede está fixa. Conclui-se então que

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_j} o_i$$

Denotar-se-á $\partial E/\partial s_j$ por η_j para se guardar as componentes do gradiente fazendo

$$\frac{\partial E}{\partial w_{ij}} = \eta_j o_i$$

Mais adiante será mostrado como calcular os valores de η. Para calcular $\partial E/\partial \theta_j$ o procedimento é semelhante:

$$\frac{\partial E}{\partial \theta_j} = \frac{\partial E}{\partial s_j}\frac{\partial s_j}{\partial \theta_j}$$

Novamente, $\partial E/\partial s_j = \eta_j$. Expandindo $\partial s_j/\partial \theta_j$ tem-se:

$$\frac{\partial s_j}{\partial \theta_j} = \frac{\partial\left(\theta_j + \sum_{b\in B} o_b w_{bj}\right)}{\partial \theta_j} = \frac{d}{d\theta_j}\theta_j = 1$$

Portanto,

$$\frac{\partial E}{\partial \theta_j} = \eta_j$$

Agora serão calculados os valores de η. Sejam j um perceptron da camada de saída, o_i e t_i as componentes i de O e T, respectivamente, e $m = dim(O) = dim(T)$. Ao se expandir a fórmula do erro, obtém-se:

$$E = \frac{1}{2}\left\{\left[(o_1 - t_1)^2 + \cdots + (o_m - t_m)^2\right]^{\frac{1}{2}}\right\}^2 = \frac{1}{2}(o_1 - t_1)^2 + \cdots + \frac{1}{2}(o_m - t_m)^2$$

Ao se derivar E em relação a s_j, todas as parcelas que não dependem de o_j serão anuladas. Assim tem-se

$$\eta_j = \frac{\partial\left[\frac{1}{2}(o_j - t_j)^2\right]}{\partial s_j} = \frac{\partial\left[\frac{1}{2}(\sigma(s_j) - t_j)^2\right]}{\partial s_j}$$

Aplicando a regra da cadeia, tem-se

$$\eta_j = \left[\frac{d}{ds_j}\sigma(s_j)\right][\sigma(s_j) - t_j]$$

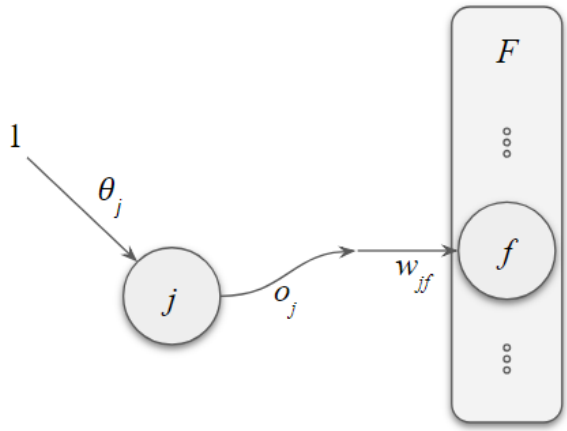
Portanto,

$$\eta_j = \sigma(s_j)[1 - \sigma(s_j)][\sigma(s_j) - t_j] = o_j(1 - o_j)(o_j - t_j)$$

Para os demais perceptrons, os valores de η serão calculados de forma indutiva. Como representado na Figura 5, seja j um perceptron tal que os valores de η dos perceptrons F alimentados por j sejam conhecidos.

Considerando E como uma função dos somatórios s_f, $f \in F$, e aplicando a regra da derivada total, obtém-se:

Figura 5 – Perceptrons j e f



Fonte: Elaborado pelo autor

$$\eta_j = \frac{\partial E}{\partial s_j} = \frac{\partial E(\ldots, s_f, \ldots)}{\partial s_j} = \sum_{f \in F} \frac{\partial E}{\partial s_f} \frac{\partial s_f}{\partial s_j} = \sum_{f \in F} \eta_f \frac{\partial s_f}{\partial s_j}$$

Expandindo $\partial s_f / \partial s_j$:

$$\begin{aligned} \frac{\partial s_f}{\partial s_j} &= \frac{\partial [\theta_f + o_\xi w_{\xi f} + \cdots + o_j w_{jf} + \cdots + o_\zeta w_{\zeta f}]}{\partial s_j} \\ &= \frac{\partial [\sigma(s_j) w_{jf}]}{\partial s_j} = w_{jf} \left[\frac{d}{ds_j} \sigma(s_j) \right] = w_{jf} \sigma(s_j)[1 - \sigma(s_j)] \\ &= w_{jf} o_j (1 - o_j) \end{aligned}$$

Portanto,

$$\eta_j = \sum_{f \in F} \eta_f w_{jf} o_j (1 - o_j) = o_j (1 - o_j) \sum_{f \in F} \eta_f w_{jf}$$

ii) Ajustar os valores w e θ

Uma vez computados $\partial E / \partial w_{ij}$ e $\partial E / \partial \theta_j$ para todos perceptrons i e j, basta modificar w_{ij} e θ_j segundo as fórmulas:

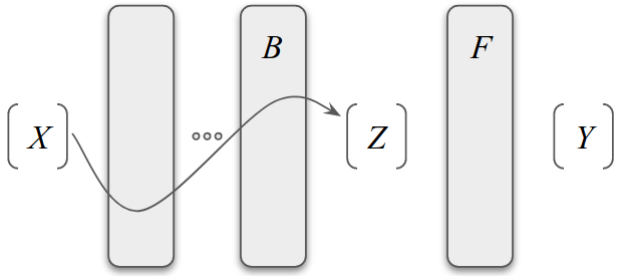
$$w_{ij} = w_{ij} - \alpha_{\text{BPG}} \frac{\partial E}{\partial w_{ij}}$$
$$\theta_j = \theta_j - \alpha_{\text{BPG}} \frac{\partial E}{\partial \theta_j}$$

Onde α_{BPG}, a taxa de aprendizagem do algoritmo BPG, é o tamanho do passo dado no sentido contrário ao gradiente em cada iteração.

2.1.2.2 *Extreme Learning Machine*

O algoritmo ELM foi criado por Guang-Bin Huang (HUANG *et al.*, 2006) e tem por objetivo adaptar Redes Neurais a um determinado conjunto de treinamento ao resolver uma regressão linear para cada perceptron da camada de saída. Assim como representado na Figura 6, sejam X uma matriz de entradas para a rede, Y a matriz de respostas para X, B o conjunto de perceptrons da última camada oculta, Z a matriz das saídas de B quando a rede é alimentada com as entradas de X e F o conjunto de perceptrons da camada de saída. Assuma que a primeira coluna de Z contenha apenas o elemento 1, pois se trata da componente multiplicativa dos vieses dos perceptrons de F. O objetivo do algoritmo é encontrar os pesos das arestas que ligam os perceptrons de B aos de F e os vieses para os perceptrons de F tais que as respostas da rede para X sejam as mais próximas de Y possível.

Figura 6 – Transformação de X para Z



Fonte: Elaborado pelo autor

Para cada perceptron f da camada de saída, sejam y_f o vetor de respostas de Y referentes ao perceptron f, ϕ_f o melhor viés para f e $\omega_f = (\omega_{1f}, \ldots, \omega_{|B|f})$ os melhores pesos para as arestas que conectam os perceptrons de B a f. Devido à função de propagação em f, seja γ_f o vetor calculado com a inversa da função sigmoide aplicada a cada elemento de y_f para que, matricialmente, o problema possa ser formulado segundo o sistema linear:

$$Z \times \overset{\Psi_f}{\begin{bmatrix} \phi_f \\ \omega_{1f} \\ \vdots \\ \omega_{|B|f} \end{bmatrix}} = \gamma_f,$$

Um sistema assim normalmente não tem solução para Ψ_f. Dessa forma, propõe-se encontrar uma solução que minimize a função $\varepsilon(\Psi_f) = \|\gamma_f - Z\Psi_f\|^2_{F_b}$ com o método dos

mínimos quadrados. Expandindo $\varepsilon(\Psi_f)$, obtém-se:

$$\begin{aligned}\varepsilon(\Psi_f) &= (\gamma_f - Z\Psi_f)^T(\gamma_f - Z\Psi_f) \\ &= \gamma_f^T\gamma_f - \gamma_f^T Z\Psi_f - \Psi_f^T Z^T\gamma_f + \Psi_f^T Z^T Z\Psi_f\end{aligned}$$

Para encontrar o mínimo da função ε, iguala-se sua primeira derivada a zero:

$$\frac{\partial\varepsilon}{\partial\Psi_f} = \frac{\partial(\overset{A}{\gamma_f^T\gamma_f})}{\partial\Psi_f} + \frac{\partial(\overset{B}{\Psi_f^T Z^T Z\Psi_f})}{\partial\Psi_f} - \frac{\partial(\overset{C}{\gamma_f^T Z\Psi_f})}{\partial\Psi_f} - \frac{\partial(\overset{D}{\Psi_f^T Z^T\gamma_f})}{\partial\Psi_f} = 0$$

Como $A = 0$, $B = 2Z^T Z\Psi_f$ e $C = D = \gamma_f^T Z$, tem-se:

$$\frac{\partial\varepsilon}{\partial\Psi_f} = 2Z^T Z\Psi_f - 2\gamma_f^T Z = 0$$

Assim, quando Z tem todas as suas colunas linearmente independentes, $Z^T Z$ é invertível e é possível calcular Ψ_f com a fórmula:

$$\Psi_f = (Z^T Z)^{-1}\gamma_f^T Z$$

Para evitar ótimos locais, é possível fazer uma combinação linear do estado atual da rede e da solução dos mínimos quadrados. Assim, no passo atual, sejam θ_f o viés de f, $w_f = (w_{1f}, \ldots, w_{|B|f})$ o vetor dos pesos das suas arestas e α_{ELM} a taxa de aprendizagem do algoritmo ELM, que representa o quão próximo de Ψ_f se deseja chegar. Se não existe Ψ_f, f permanece inalterado. Caso contrário, a atualização de f é feita com as seguintes fórmulas:

$$\begin{aligned}\theta_f &= \theta_f + \alpha_{\text{ELM}}(\phi_f - \theta_f) \\ w_f &= w_f + \alpha_{\text{ELM}}(\omega_f - w_f)\end{aligned}$$

Em seu trabalho, Huang criou o algoritmo ELM para Redes Neurais com apenas uma camada oculta e mostrou que se trata de uma forma eficiente de treinamento. Ainda assim, o ELM será utilizado nesta dissertação para redes com quantidades arbitrárias de camadas ocultas pelo simples fato de ser capaz de minimizar o erro que uma rede comete para um determinado conjunto de treinamento.

2.2 Algoritmos Genéticos

Algoritmos Genéticos são uma classe de métodos de busca para problemas de otimização inspiradas na Teoria da Evolução proposta por Charles Darwin em sua obra *A Origem das Espécies* (DARWIN, 1859; BEASLEY *et al.*, 1993). Embora não haja uma definição formal, alguns conceitos sempre estão presentes nos trabalhos que utilizam estes algoritmos: populações de indivíduos, seleção dos mais aptos, *crossovers* para a reprodução e mutações aleatórias (MELANIE, 1999).

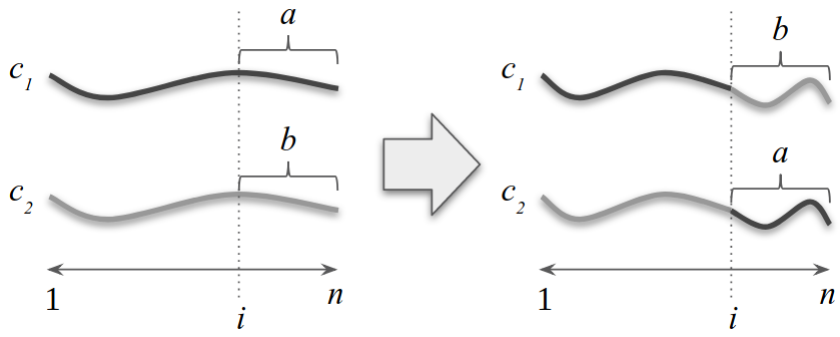
Um cromossomo é a representação de um agente inteligente capaz de agir na tarefa que se deseja automatizar e o cálculo do seu desempenho é feito pela função de aptidão, cuja implementação é responsabilidade de quem deseja resolver o problema.

2.2.1 Reprodução

Aqui se apresenta uma das peculiaridades da forma canônica dos Algoritmos Genéticos: o fato dos cromossomos serem tratados apenas como sequências de bits sem que sejam feitas inferências a respeito dos seus significados. Por este motivo, Algoritmos Genéticos são classificados como meta-heurísticas. Além disso, é na reprodução que ocorre a exploração do espaço de busca. A população de uma nova geração é criada a partir da atual com o uso de três operadores: seleção, *crossover* e mutação.

- Seleção: escolhe aleatoriamente dois cromossomos para o *crossover*. Um cromossomo pode ser escolhido várias vezes por geração para compor pares e a probabilidade dele ser escolhido é proporcional à sua aptidão.
- *Crossover*: cria dois cromossomos filhos para a geração seguinte a partir de dois cromossomos pais da geração atual. Sejam os vetores de bits $c_1[1..n]$ e $c_2[1..n]$ os cromossomos pais e c'_1 e c'_2 suas cópias, que serão as saídas do *crossover*. Ocorre então um teste binário com probabilidade p_c para 'sim' e caso o teste obtenha resultado positivo, um número i de 1 a n é escolhido aleatoriamente para que seja executada a troca entre os bits em $c'_1[i..n]$ e $c'_2[i..n]$, como mostra a Figura 7. Caso contrário, nenhuma alteração é feita em c'_1 e c'_2. Este é o *crossover* de ponto único, pois foi escolhido apenas um índice (i) para demarcar os limites da troca genética.
- Mutação: inverte cada bit dos cromossomos resultantes do *crossover* com uma probabilidade baixa p_m, por exemplo $0,001$.

Figura 7 – *Crossover*



Fonte: Elaborado pelo autor

O processo de reprodução pode ser entendido como o encadeamento de seleções, *crossovers* e mutações até que seja gerada uma nova população de mesmo tamanho que a atual. Caso a nova população exceda a atual em 1, um indivíduo qualquer pode ser eliminado para manter a coerência ao longo das gerações.

2.2.2 O Algoritmo

Algumas implementações de Algoritmos Genéticos utilizam um corte percentual na população com base na aptidão dos indivíduos antes da fase de reprodução, como é o caso da implementação utilizada nos experimentos desta dissertação. Assim, as ideias mencionadas podem ser organizadas no Algoritmo 1.

2.3 Neuroevolução de Topologias Aumentantes

Neuroevolução é a combinação de Redes Neurais e Algoritmos Genéticos, onde os cromossomos são representações compactas de Redes Neurais, que por sua vez são avaliadas pela função de aptidão (STANLEY, 2004). As particularidades dos Algoritmos Neuroevolutivos estão na codificação da Rede Neural e na implementação dos operadores do Algoritmo Genético.

Kenneth O. Stanley (STANLEY; MIIKKULAINEN, 2002b) propôs um algoritmo denominado Neuroevolução de Topologias Aumentantes (NEAT), que alcançou fortes resultados nos problemas conhecidos na literatura. Em sua publicação original, a eficiência do NEAT é justificada por ele *i*) promover cruzamentos genéticos que não empobrecem a capacidade computacional das proles em relação aos pais, *ii*) proteger inovações genéticas por meio do processo de especiação e *iii*) buscar soluções a partir de topologias simples, incrementando-as gradativamente. A seguir, serão apresentados os detalhes da codificação no NEAT e destes três aspectos mencionados.

Algoritmo 1: Algoritmo Genético

```
Entrada: max_gerações, aptidão_alvo
Saída: solução
início
    população ← nova População();
    solução ← ∅;
    para geração ← 1 até max_gerações faça
        para cada indivíduo ∈ população faça
            indivíduo.aptidão ← avaliar(indivíduo);
            se solução = ∅ ou solução.aptidão < indivíduo.aptidão então
                solução ← indivíduo;
            fim
        fim
        se solução.aptidão ≥ aptidão_alvo então
            break;
        fim
        população ← população.corte().reprodução();
    fim
    retorna solução;
fim
```

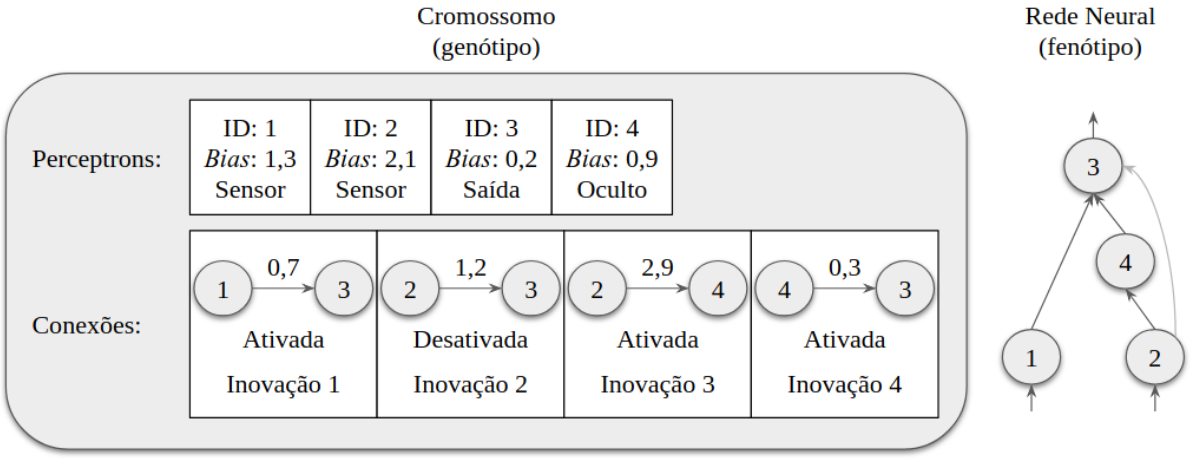
2.3.1 Codificação Genética

O cromossomo no NEAT possui uma lista de perceptrons e uma lista de conexões, como ilustra a Figura 8. Na lista de perceptrons, cada um deles é caracterizado segundo seus identificadores, os valores de seus vieses e seus tipos (sensor, oculto ou saída). Na lista de conexões, cada aresta é definida pelos perceptrons de entrada e saída, pelo seu peso, pelo bit de ativação e pelo identificador da inovação.

A mutação pode alterar tanto os pesos das arestas quanto a estrutura da rede. Na mutação dos pesos, cada aresta pode ser perturbada individualmente. A mutação estrutural, como ilustra a Figura 9, pode ocorrer de duas formas: adicionando perceptrons ou conexões.

Na adição de perceptron, uma aresta é dividida em duas pelo novo perceptron, a conexão antiga é desabilitada, as novas conexões são adicionadas à lista de arestas, a conexão para o novo perceptron recebe peso 1 e a conexão que sai do novo perceptron recebe o peso da conexão desabilitada.

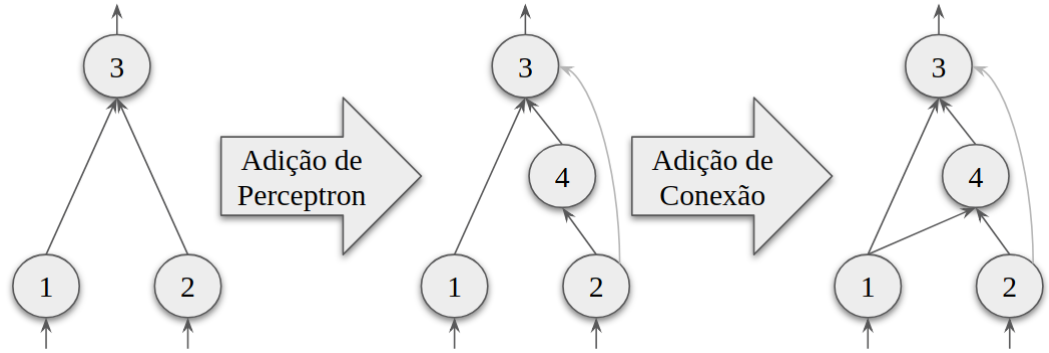
Figura 8 – Cromossomo no NEAT



Fonte: Elaborado pelo autor

Na adição de conexão, uma aresta de peso aleatório passa a ligar dois perceptrons antes desconectados.

Figura 9 – Mutação estrutural no NEAT



Fonte: Elaborado pelo autor

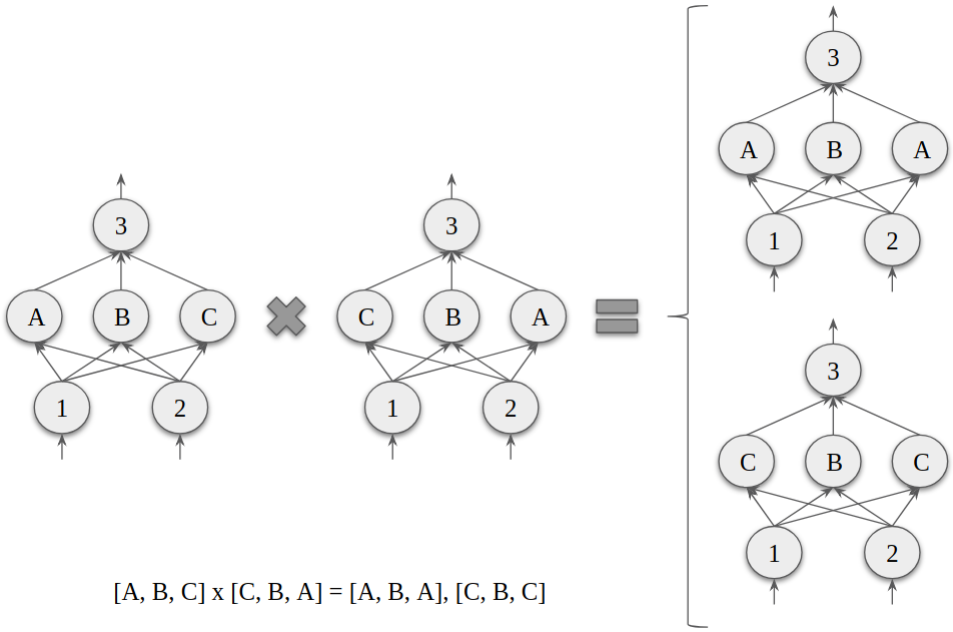
Quando uma mutação estrutural ocorre, a ela é atribuída um identificador de inovação incremental e global. Dentro de uma geração, mantém-se o registro de todas as mutações estruturais para que sejam atribuídos os mesmos identificadores de inovação às mutações estruturais equivalentes. A motivação para esta prática e suas consequências serão explicadas a seguir.

2.3.2 Rastreando Genes com Marcações Históricas

No contexto dos Algoritmos Neuroevolutivos, existe um problema chamado Convenções Concorrentes (SCHAFFER *et al.*, 1992), que ocorre quando há mais de uma forma de codificar subestruturas equivalentes. Em situações assim, é provável que o *crossover* gere filhos deficientes. Por exemplo, a Figura 10 ilustra um cruzamento de dois cromossomos gerando indivíduos mais pobres que os pais, em termos de variabilidade estrutural, pois eles possuem

perceptrons replicados.

Figura 10 – Convenções Concorrentes



[A, B, C] x [C, B, A] = [A, B, A], [C, B, C]

Fonte: Elaborado pelo autor

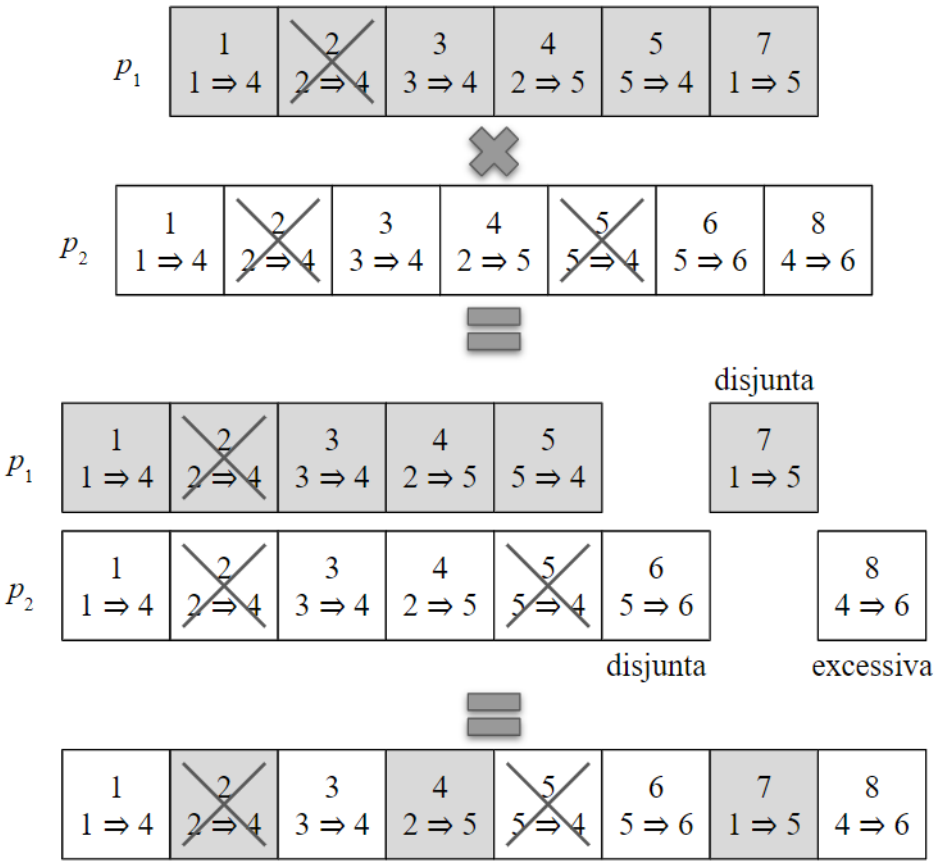
Para resolver este problema, utiliza-se as marcações cronológicas das mutações estruturais como referenciais nos *crossovers* de modo que os identificadores de inovação estejam alinhados, como na Figura 11. Das conexões que possuem marcações correspondentes, o filho herda uma de cada pai aleatoriamente. Das demais conexões, o filho herda do pai com maior aptidão ou dos dois, caso ambos sejam igualmente aptos. Esta forma de executar o *crossover* evita a necessidade de avaliações topológicas complexas e impossibilita que os filhos herdem inovações estruturais redundantes.

2.3.3 Protegendo Inovações com a Especiação

Quando surgem topologias novas, o mais provável é que elas tenham aptidões baixas em relação aos indivíduos da mesma geração. Portanto, para evitar que indivíduos com características promissoras sejam sacrificados prematuramente, a população é particionada em espécies para que cada cromossomo possa competir com seus semelhantes.

O particionamento em espécies é realizado com o auxílio das marcações históricas, novamente evitando análises topológicas complexas. Quando dois cromossomos têm seus identificadores de inovação alinhados, como mostra a Figura 11, as conexões não correspondentes podem ser disjuntas ou excessivas. Para explicar a diferença entre estas qualidades, sejam p_1 e p_2

Figura 11 – *Crossover* no NEAT



Fonte: Elaborado pelo autor

cromossomos e seus indicadores máximos i_1 e i_2, respectivamente. Sem perda de generalidade, assuma que $i_1 \leq i_2$. Todas as conexões não correspondidas e com indicadores menores que, ou iguais a i_1, são consideradas disjuntas. As demais conexões não correspondidas são consideradas excessivas.

A intuição que sustenta o cálculo da distância entre dois cromossomos é que quanto mais conexões não correspondentes entre eles, menos história evolucionária eles compartilham. Sejam então D e E as quantidades de conexões disjuntas e excessivas respectivamente, $\bar{W}$ a diferença média entre os pesos das conexões correspondentes, c_1, c_2 e c_3 parâmetros ajustáveis e N o tamanho do maior dos cromossomos. A distância δ entre dois cromossomos é definida pela seguinte combinação linear:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \bar{W} \tag{2.1}$$

Na geração inicial, todos os indivíduos são agrupados na mesma espécie, para a qual um representante é selecionado aleatoriamente. Nas demais gerações, para enquadrar um cromossomo g em uma espécie, verifica-se a distância δ entre g e os representantes das espécies

da geração anterior. O cromossomo g é inserido na primeira espécie cujo representante está a uma distância inferior a um valor δ_t de g. Caso não haja tal espécie, cria-se uma nova para g.

Após o procedimento descrito acima, para cada espécie e é calculada sua aptidão média $\bar{f}_e$ e é realizado um corte, eliminando uma fração r dos seus cromossomos de acordo com suas aptidões. Então, os cromossomos restantes são pareados aleatoriamente para que ocorram *crossovers* até que seja atingido o limite máximo de indivíduos para e, que é proporcional a $\bar{f}_e$. A população de e é completamente substituída e seu novo representante é novamente escolhido de forma aleatória.

Existe ainda o conceito de *estagnação*, que ocorre quando a aptidão do melhor indivíduo de uma espécie não melhora de uma geração para a seguinte. A mitigação para este problema é descartar toda a espécie após K gerações em estagnação. Caso toda a população chegue a ser eliminada por conta desta medida, uma nova é instanciada e o processo continua.

2.3.4 Partindo de Estruturas Mínimas

Iniciar uma população de Redes Neurais densas pode ser desnecessariamente custoso, pois o algoritmo precisaria de muito tempo para otimizar estruturas complexas (STANLEY; MIIKKULAINEN, 2002a). Ao invés, o NEAT instancia populações de cromossomos que representam Redes Neurais sem camadas ocultas e com poucas – ou sem – arestas.

Os incrementos nas topologias ocorrem com as mutações estruturais e os piores cromossomos de cada espécie são filtrados pela função de aptidão. Ou seja, as complexificações nas topologias que permanecem no longo prazo são justificadas por melhorias nas aptidões dos indivíduos, possibilitando ao NEAT percorrer o espaço de busca explorando primeiro soluções mais simples e fáceis de otimizar.

2.3.5 O Algoritmo

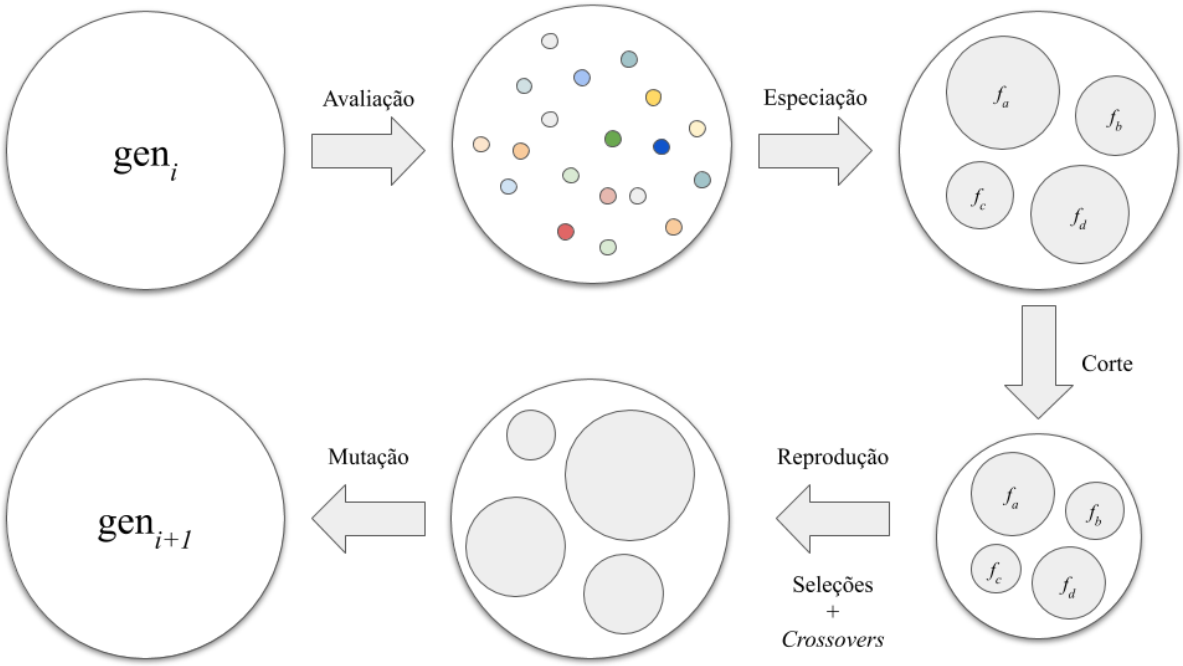
Compilando os aspectos já mencionados, o NEAT pode ser compreendido como um processo iterativo no qual cada ciclo contempla as seguintes fases:

1. Avaliação: processo no qual cada indivíduo é submetido à prova de aptidão;
2. Especiação: agrupamento dos agentes com base na função de distância δ (Equação 2.1);
3. Corte: eliminação de um percentual dos indivíduos (os menos aptos) de cada espécie;
4. Reprodução: criação de novos agentes a partir do cruzamento (Figura 11) entre indivíduos aleatórios do mesmo grupo até se alcançar o limite máximo permitido para cada espécie;

5. Mutação: alterações nos indivíduos por meio de perturbações nos pesos das arestas ou de mutações estruturais (Figura 9).

O algoritmo pode ser interrompido caso algum agente alcance a aptidão desejada na fase de avaliação. Visualmente, o processo para se produzir a geração gen_{i+1} a partir da geração gen_i pode ser representado como na Figura 12.

Figura 12 – Ciclo do NEAT



Fonte: Elaborado pelo autor

2.4 Conclusão

Este capítulo apresentou os conceitos necessários para a elaboração da proposta desta dissertação, iniciando com uma introdução às Redes Neurais e aos algoritmos genéticos, para então combiná-los no algoritmo NEAT. Os algoritmos BPG e ELM servirão para a transmissão de conhecimento entre os indivíduos envolvidos no processo evolutivo, procedimento que será detalhado no capítulo seguinte.

3 *CENEAT* — NEUROEVOLUÇÃO DE TOPOLOGIAS AUMENTANTES COM MELHORIAS CULTURAIS

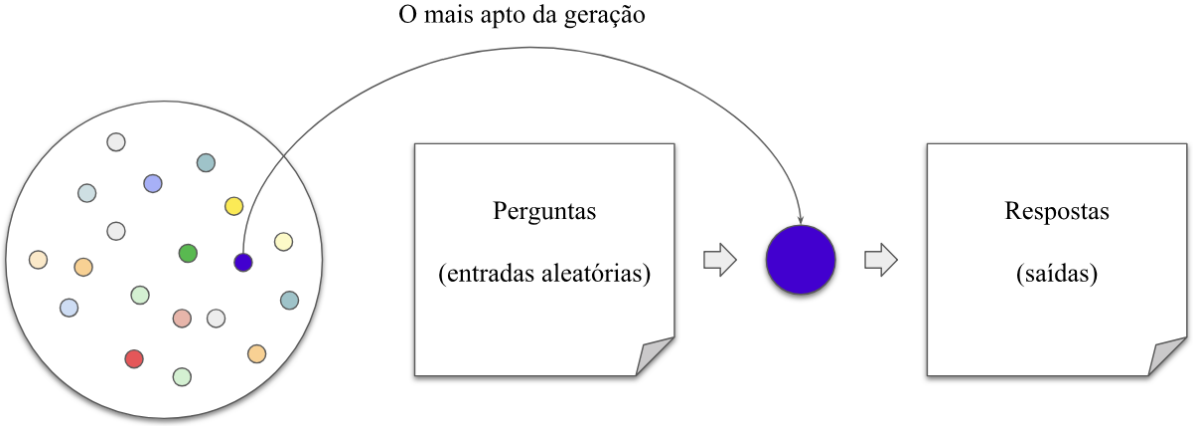
Este capítulo apresenta a proposta da dissertação em quatro seções. A Seção 3.1 explica em mais detalhes como o conhecimento adquirido em uma geração é passado para os indivíduos da próxima. A Seção 3.2 traz reflexões sobre a persistência genética do aprendizado. Na Seção 3.3 são apresentadas algumas variações para a inserção de cultura no processo evolutivo e a Seção 3.4 encerra o capítulo com algumas ponderações sobre o custo computacional.

3.1 A Cultura no NEAT

O conhecimento transmitido entre gerações por meios não-genéticos pode ser entendido como cultura (SPECTOR; LUKE, 1996). A ideia de se inserir cultura em um Algoritmo Neuroevolutivo advém do fato de que os indivíduos são Redes Neurais, podendo ter seus comportamentos modificados por meio de alterações nos pesos de suas arestas. Desta forma, é possível fazer com que certos indivíduos absorvam padrões de comportamento de alguma fonte de conhecimento antes de serem submetidos à avaliação de aptidão. Resta então definir formalmente o que seria esta fonte de conhecimento e como exatamente os indivíduos podem aprender com ela.

Em sua tese de doutorado, Paul Herbert McQuesten (MCQUESTEN, 2002) propõe soluções para as questões acima. Em seu trabalho, a base de conhecimento é denominada de *syllabus* e é construída a partir de 20 a 40 lições (perguntas e respostas): as perguntas são entradas aleatórias para a Rede Neural do indivíduo mais apto da geração atual e as respostas são as saídas desta rede para tais entradas, como mostra a Figura 13.

Figura 13 – Construção do *syllabus*

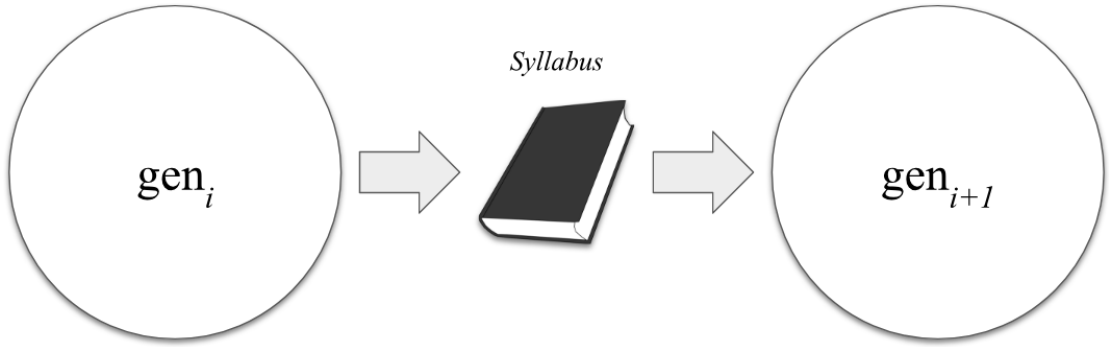


Fonte: Elaborado pelo autor

Após a fase de mutação, quando a população está pronta para compor a geração

seguinte, os novos indivíduos são submetidos ao processo de aprendizagem, que consiste em executar o algoritmo BPG para cada lição do *syllabus*. Por este motivo, diz-se que o conhecimento é transferido de uma geração para a seguinte, como representado na Figura 14.

Figura 14 – Transferência de conhecimento



Fonte: Elaborado pelo autor

A primeira proposta desta dissertação é incluir os passos descritos acima no algoritmo NEAT, utilizando um *syllabus* de tamanho 30. É importante mencionar que o treinamento faz com que os agentes fiquem diferentes uns dos outros, ao ponto de surgirem quase tantas espécies distintas quanto indivíduos na população, causando problemas para o funcionamento do NEAT. Portanto, para que o enriquecimento cultural seja compatível com a especiação, é necessário computar as distâncias δ entre os agentes, segundo a Equação 2.1, levando em consideração apenas suas dessemelhanças evolutivas e ignorando os pesos das suas arestas ($c_3 = 0$).

Os indivíduos utilizados para formar as respostas do *syllabus* são chamados de tutores e os indivíduos escolhidos para o treinamento são chamados de aprendizes. Este processo no qual os aprendizes absorvem o comportamento do tutor por meio de lições é chamado de tutoria. As ideias apresentadas nesta seção estão condensadas no Algoritmo 2.

Embora o algoritmo NEAT também seja capaz de evoluir Redes Neurais Recorrentes, este tipo de rede não será utilizado neste trabalho devido aos problemas referentes ao seu treinamento (PASCANU *et al.*, 2013).

3.2 O Efeito Baldwin e a Mecânica Lamarckiana

A cultura desenvolvida por uma população é capaz de direcionar o seu processo evolutivo, mesmo que os comportamentos adquiridos após o nascimento não sejam transmitidos para a geração seguinte por meios genéticos (BALDWIN, 1896). Este fenômeno é conhecido como *Efeito Baldwin* e a justificativa atribuída à sua manifestação é que a capacidade de

Algoritmo 2: *Culturally Enhanced Neuroevolution of Augmenting Topologies* (ceNEAT)

```
Entrada: max_gerações, aptidão_alvo
Saída: solução
início
    população ← nova População();
    solução ← ∅;
    para geração ← 1 até max_gerações faça
        para cada indivíduo ∈ população faça
            indivíduo.aptidão ← avaliar(indivíduo);
            se solução = ∅ ou solução.aptidão < indivíduo.aptidão então
                solução ← indivíduo;
            fim
        fim
        se solução.aptidão ≥ aptidão_alvo então
            break;
        fim
        syllabus ← {∅};
        para contador_lição ← 1 até 30 faça
            pergunta ← aleatório();
            resposta ← solução.reagir(pergunta);
            syllabus.adicionar( (pergunta, resposta) );
        fim
        população ← população.especiação();
        população ← população.corte().reprodução();
        para cada individuo ∈ população.filhos faça
            para cada (pergunta, resposta) ∈ syllabus faça
                backpropagation(individuo, pergunta, resposta);
            fim
        fim
    fim
    retorna solução;
fim
```

aprendizagem dos indivíduos também é uma característica genética. Logo, em contextos nos quais a cultura exerce um papel importante para o sucesso dos indivíduos, o padrão genético dos

bons aprendizes tem maiores chances de ser perpetuado. Inspirados por essa ideia, Ronald C. Keesing (KEESING; STORK, 1991), Nolfi e Parisi (NOLFI *et al.*, 1990) mostraram que o Efeito Baldwin pode melhorar o desempenho dos Algoritmos Neuroevolutivos.

Anterior ao trabalho de Darwin (DARWIN, 1859), Jean-Baptiste Lamarck publicou o livro Filosofia Zoológica (LAMARCK, 1809), no qual ele defendia que as mudanças adquiridas ao longo da vida dos animais podiam ser transmitidas via reprodução. Atualmente, esta hipótese não é aceita como regra geral para a continuidade dos traços hereditários observáveis na natureza. No entanto, trata-se de uma possibilidade aberta para as simulações computacionais. No contexto dos Algoritmos Neuroevolutivos que envolvem aprendizagem, a Mecânica Lamarckiana pode ser implementada atualizando os pesos das arestas nos cromossomos após as seções de treinamento das respectivas redes.

Embora P. McQuesten não tenha explicitado se os cromossomos eram atualizados após as execuções do algoritmo BPG nos experimentos da sua tese, a Mecânica Lamarckiana foi adotada nesta dissertação. O motivo desta decisão advém dos trabalhos de Frederic Gruau (GRUAU *et al.*, 1993), Darrell Whitley (WHITLEY *et al.*, 1994) e Bryant A. Julstrom (JULSTROM, 1999), os quais, além de reforçarem as consequências positivas do Efeito Baldwin para os Algoritmos Neuroevolutivos, mostram também que a Mecânica Lamarckiana é capaz de torná-los ainda mais eficientes.

Um risco associado a esta prática é restringir demais a exploração do espaço de busca, pois além dos indivíduos serem treinados para se assemelharem a um mesmo tutor em cada geração, esta similaridade é transmitida para suas proles. Porém, tal preocupação é mitigada pelo fato do NEAT implementar mutações estruturais – visto que a tutoria afeta apenas os pesos das arestas –, o que indica que o algoritmo NEAT e a inserção de cultura por meio da Mecânica Lamarckiana combinam bem. Na seção seguinte, são apresentadas algumas variações que também têm o objetivo de minimizar a semelhança excessiva aos tutores.

3.3 Variantes para o Enriquecimento Cultural

Neste trabalho também são propostas algumas variações na introdução de cultura no NEAT como extensão do trabalho de P. McQuesten e cujo objetivo é explorar ainda mais possibilidades de se implementar o enriquecimento cultural.

3.3.1 Algoritmo de Aprendizagem

Para que os aprendizes tomem decisões um pouco mais parecidas com as do tutor, os pesos de suas arestas são modificados de modo que estes obtenham uma melhor pontuação nas lições do *syllabus*. Ou seja, a distância entre as respostas dos aprendizes e as do tutor precisa ser minimizada. Na ideia original isto é feito com o algoritmo BPG.

Uma outra forma de alcançar este objetivo é com o algoritmo ELM. As principais diferenças com relação ao BPG são *i*) o resultado do treinamento não depende da ordem das lições do *syllabus*, *ii*) apenas as arestas da camada de saída são modificadas e *iii*) as taxas de aprendizagem representam grandezas diferentes para cada algoritmo.

Dadas as diferenças apontadas, utilizar o algoritmo de aprendizagem ELM para treinar os aprendizes se torna uma possibilidade interessante a ser explorada, capaz levar a resultados diversificados. Para evitar que os indivíduos submetidos ao treinamento fiquem muito semelhantes entre si, será utilizada a taxa de aprendizagem do algoritmo ELM, mencionada no Capítulo 2. Esta variante será indicada pelo sufixo “ELM”.

3.3.2 Momento do Aparecimento de Cultura

A primeira preocupação que surge quando vários indivíduos são treinados para se comportarem de forma semelhante é com relação à redução no espaço de busca, ou seja, o aumento da similaridade entre as soluções encontradas. É possível que a busca por soluções mais diversas ocorra no início da evolução, pois em estados mais avançados o algoritmo tentaria refinar os melhores indivíduos já encontrados. Outra possibilidade é que as lições do *syllabus*, no início da evolução, sejam compostas por respostas de tutores pouco aptos a executarem a tarefa e, portanto, utilizá-las para moldar o comportamento dos outros indivíduos poderia atrasar a convergência do algoritmo.

Por estes motivos, uma das variantes propostas é iniciar o treinamento dos aprendizes a partir da metade do processo evolutivo. Desta forma, a restrição do espaço de busca não ocorre na fase inicial da evolução e os tutores só servirão de exemplo para os aprendizes em gerações mais maduras, nas quais espera-se construir *syllaba* mais confiáveis. Esta variante será indicada pelo sufixo “LATE”.

3.3.3 Seleção dos Aprendizes

Pelo funcionamento dos Algoritmos Neuroevolutivos, a aptidão de um indivíduo é determinada pela sua carga genética, a qual depende das características dos pais e dos processos aleatórios que ocorrem no *crossover* – à exceção da primeira geração. Sendo assim, executar o treinamento logo após o nascimento dos indivíduos afeta uma das fontes de variabilidade genética do algoritmo.

Uma forma alternativa de inserir cultura no NEAT é treinar alguns indivíduos antes da fase de reprodução, definindo os sobreviventes do corte da população antes da seleção artificial como os aprendizes, os quais serão os pais dos indivíduos da geração seguinte. Como consequência, o comportamento absorvido na tutoria é passado adiante devido à Mecânica Lamarckiana e a variabilidade genética alcançada na criação das proles possivelmente será menos afetada. Além disso, ao se treinar apenas os pais, existe também um ganho de eficiência em relação ao treinamento dos filhos, pois a parcela de sobreviventes é pequena se comparada ao tamanho da população. Esta variante será indicada pelo sufixo “PARENTS”.

3.3.4 Natureza das Lições

Para esta variante, os indivíduos precisam de memória para que possam lembrar de alguns instantes aleatórios de suas avaliações. A proposta é que o *syllabus* seja construído a partir da vivência do tutor e contenha lições que dizem respeito à tarefa propriamente dita ao invés de perguntas aleatórias. Com relação ao custo computacional, há uma troca de processamento por memória, pois as perguntas e as respostas foram geradas durante a avaliação de aptidão. Esta variante será indicada pelo sufixo “EXP”.

Louis e Johnson (LOUIS; JOHNSON, 1997) avaliaram esta abordagem e evidenciaram que a melhoria na convergência pode não ser tão significativa caso as lições sejam referentes a um subconjunto muito específico do espaço do problema. Eles foram mencionados no trabalho de P. McQuesten, o qual, no entanto, não contempla testes comparativos. Portanto, esta dissertação põe as duas ideias à prova, com a ressalva de que as lições sejam construídas com amostras aleatórias das memórias dos tutores no caso em que as entradas não são aleatórias.

Nesse caso, não há razão clara para que os resultados alcançados com lições provenientes da memória do tutor sejam piores que os alcançados com lições aleatórias, pois, em ambos os casos, as restrições no espaço de busca provenientes dos treinamentos são semelhantes.

A vantagem do primeiro está na usabilidade das lições absorvidas pelos aprendizes, pois a experiência do tutor contém informações objetivas sobre como alcançar maiores aptidões.

Ainda em sua tese, P. McQuesten cita o trabalho de Federico Cecconi (CECCONI *et al.*, 1995), no qual propõe-se construir o *syllabus* com base na experiência dos pais, e argumenta que a utilização de lições aleatórias é uma abordagem mais flexível e que apresenta melhores resultados. Portanto, a ideia de F. Cecconi não será testada nesta dissertação.

3.4 Conclusão

Este capítulo apresentou em detalhes a proposta desta dissertação, primeiramente mostrando como utilizar a proposta de P. McQuesten com o objetivo de aprimorar o algoritmo NEAT. Em seguida, foram exploradas algumas variações na forma de implementar o enriquecimento cultural.

É importante notar que as variantes das Subseções 3.3.2, 3.3.3 e 3.3.4 diminuem o custo computacional associado ao enriquecimento cultural. Então, mesmo que estas alcancem a mesma aptidão que o NEAT com cultura para um mesmo algoritmo de aprendizagem, ainda assim se trata de um resultado positivo. A primeira pergunta é se o enriquecimento cultural realmente propicia um incremento na aptidão ao final do processo evolutivo. E caso haja mesmo uma melhoria na qualidade das soluções, outra pergunta que surge é se esta melhoria não seria simplesmente devido ao aumento no custo computacional em cada geração. Ou seja, o enriquecimento cultural não seria apenas um acréscimo de processamento de modo que o NEAT sem cultura alcançaria soluções tão boas quanto no mesmo intervalo de tempo?

No capítulo seguinte, serão exploradas as hipóteses sobre a melhora na qualidade das soluções e no desempenho do algoritmo NEAT. Será também avaliada a influência das variantes apresentadas na convergência do algoritmo.

4 EXPERIMENTOS E RESULTADOS

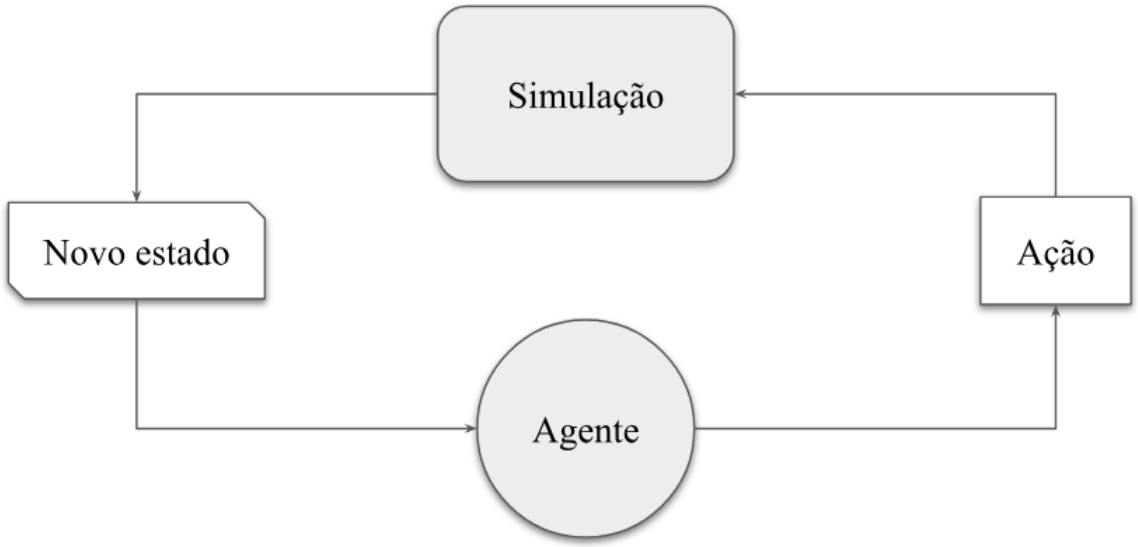
Este capítulo apresenta a validação experimental das propostas desta dissertação. A Seção 4.1 deste capítulo traz uma descrição mais detalhada dos problemas utilizados e da modelagem das Redes Neurais. A Seção 4.2 introduz a lógica geral da sequência de experimentos e a Sessão 4.3 mostra os resultados obtidos em cada um.

4.1 Os problemas e a modelagem

Problemas de decisões sequenciais (BARTO, 1990; MORIARTY, 1997) estão entre os mais difíceis na área de aprendizagem de máquina, pois muitas ações devem ser feitas antes de se obter o desempenho final do agente sem que se conheça o desdobramento final de cada uma isoladamente. Por este motivo, além do fato de serem fundamentais para o campo da automação e da robótica, esses problemas têm sido utilizados para avaliar o desempenho de Algoritmos Neuroevolutivos, como na publicação do próprio NEAT (STANLEY; MIIKKULAINEN, 2002a).

O passo de um problema de decisões sequenciais normalmente é executado por um motor de simulação, que, dada uma decisão, é capaz de computar o estado seguinte para o problema. É neste momento que entra a Inteligência Artificial: no desenvolvimento de um agente capaz de tomar decisões que reflitam em um bom desempenho final, segundo a métrica escolhida. Visualmente, a Figura 15 é uma representação clássica da interação entre um agente autônomo e um problema desta natureza.

Figura 15 – Agente atuando em um problema de decisões sequenciais



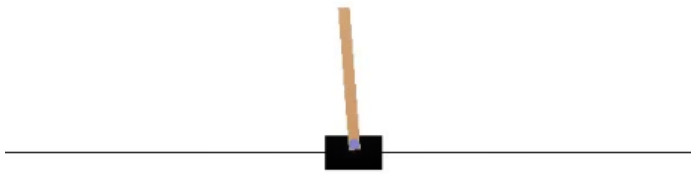
Fonte: Elaborado pelo autor

Para validar as contribuições propostas neste trabalho, dois problemas de decisões sequenciais foram utilizadas para a realização dos experimentos: *Cart-Pole* e *Mountain-Car*. São problemas nos quais a definição do estado seguinte depende somente do estado atual e da escolha feita pelo agente inteligente. Ou seja, a descrição do estado atual é sempre completa e os indivíduos não precisam desenvolver memória para lembrar de informações dos estados anteriores. Portanto, não foram utilizadas Redes Neurais Recorrentes, já que possibilitar a um Algoritmo Neuroevolutivo que sejam criados perceptrons recorrentes aumenta o tamanho do espaço de busca, medida que é necessária quando a informação faltosa em cada estado precisa ser inferida pela própria rede (GOMEZ; MIIKKULAINEN, 1999).

As Redes Neurais foram modeladas com tantos perceptrons de saída quantas possibilidades de ação e tantos perceptrons de entrada quantos foram necessários para descrever cada estado dos problemas. Em cada quadro, a ação escolhida era a referente ao perceptron cuja saída tinha a maior magnitude.

4.1.1 Cart-Pole

Figura 16 – *Cart-Pole*



Fonte: OpenAI

O objetivo é equilibrar verticalmente uma vareta, de comprimento 1 m e massa 0,1 kg, presa a um carrinho, de massa 1,0 kg, que pode se mover para a esquerda ou para a direita, sem atrito, como ilustra a figura 16. A vareta está conectada ao carrinho por uma das pontas e tem liberdade para girar nos sentidos horário e anti-horário segundo a aceleração gravitacional de 9,8 m/s^2. Para cumprir o objetivo, o agente inteligente pode interferir no sistema aplicando uma força horizontal no carrinho de 10,0 N ou -10,0 N. A Rede Neural tem acesso às leituras de

quatro sensores: a posição e a velocidade do carrinho, além do ângulo e da velocidade angular da vareta. O ambiente de avaliação dos indivíduos utilizado é uma adaptação do experimento definido por Andrew G. Barto (BARTO *et al.*, 1983) e a implementação do seu passo está descrito no Apêndice A.

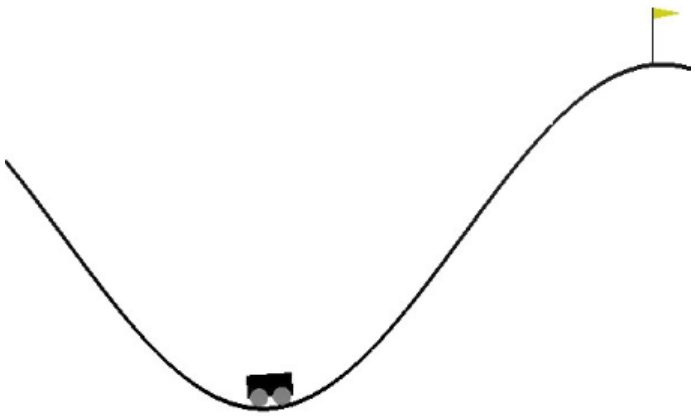
A simulação termina quando:

- O carrinho se distancia a mais de 2,4 m do centro (falha) ou
- A vareta se distancia a mais de 12^{o} da vertical (falha) ou
- A avaliação dura 500 quadros sem infração às condições anteriores (sucesso)

A cada quadro, a aptidão do agente é adicionada de $1/(1+d)$ pontos, onde d é a distância em metros até o centro do ambiente.

4.1.2 Mountain-Car

Figura 17 – *Mountain-Car*



Fonte: OpenAI

Neste problema, o carro precisa subir a montanha mas não tem aceleração suficiente para seguir em uma única direção, como ilustra a figura 17. Portanto, a solução é pendular até conquistar velocidade suficiente para alcançar a bandeira. A posição horizontal mínima é -1,2 m, a máxima é 0,6 m e o objetivo está na posição 0,5 m. O módulo máximo para a velocidade horizontal é 0,07 m/s e a aceleração gravitacional é simulada por um movimento harmônico simples regido por uma cossenoide. O agente pode interferir no sistema acelerando para a esquerda, para a direita ou não acelerando. A rede tem acesso às leituras de dois sensores: a posição horizontal e a velocidade do carro. O ambiente utilizado é uma adaptação do problema proposto por Andrew W. Moore (MOORE, 1990) e a implementação do seu passo está no

Apêndice B.

A simulação termina quando:

- A avaliação dura 500 quadros (falha) ou
- O carro sobe a montanha e alcança a bandeira (sucesso)

Ao final da avaliação, a aptidão dada ao agente é igual a $(x_{max} + 0{,}5) \times 10^4 / t$, onde x_{max} é a posição máxima alcançada pelo carro e t é a duração da simulação (em número de quadros).

4.2 Metodologia

Para cada problema, os experimentos foram compostos por três fases:

1. Validação do ceNEAT

   As soluções do algoritmo ceNEAT, cuja configuração se encontra na Tabela 1, foram comparadas às geradas pelo algoritmo NEAT. Uma vez confirmado o incremento na qualidade das soluções até uma determinada geração, é preciso verificar se os tempos necessários para estas estratégias alcançarem soluções de mesma qualidade são condizentes, ou seja, se o NEAT sem cultura demora mais para alcançar a mesma aptidão que o ceNEAT quando não há limite para a quantidade de gerações.

Tabela 1 – Configuração do ceNEAT

Aprendizes	Surgimento de cultura	Natureza das lições
Filhos	Inicial	Aleatória

Fonte: Elaborado pelo autor

2. Testes com BPG

   Ainda utilizando o mesmo algoritmo de aprendizagem, o objetivo desta fase é avaliar a influência dos seguintes aspectos na qualidade das soluções: *i*) o momento no qual cultura é inserida no processo evolutivo; *ii*) quais indivíduos serão submetidos ao processo de aprendizagem e *iii*) a origem das lições que compõem o *syllabus*. As soluções geradas serão comparadas com o ceNEAT.

3. Testes com ELM

   Com o algoritmo de aprendizagem ELM, primeiramente faz-se uma breve busca pela taxa de aprendizagem que será utilizada. Em seguida, a qualidade das soluções é comparada com as aptidões dos indivíduos produzidos pelo NEAT e pelo ceNEAT, para então serem

avaliados os tempos que tais estratégias levam para alcançar soluções de mesma qualidade. Espera-se que os tempos sejam condizentes com as qualidades encontradas, ou seja, quanto maior a aptidão alcançada em uma determinada geração, menor o tempo para se alcançar uma aptidão fixa em um contexto no qual não há limite superior para a quantidade de gerações. Por fim, serão avaliados os parâmetros enumerados no item 2.

O objetivo dos experimentos não é encontrar o melhor conjunto de parâmetros para cada problema, mas sim validar a influência das variantes na qualidade das soluções. Em todos os experimentos o *syllabus* foi construído com 30 lições, como proposto por P. McQuesten, e para todos os experimentos que envolviam o algoritmo BPG, a taxa de aprendizagem utilizada foi 0.05. A população foi construída com 100 indivíduos e, em cada geração, cada um dos indivíduos foi avaliado três vezes, sendo sua aptidão definida como a média das aptidões alcançadas em cada avaliação. Os gráficos gerados são compostos pelas aptidões dos melhores indivíduos de cada geração e cada estratégia foi testada 500 vezes.

A implementação foi feita em Python 2.7.14, utilizando como base as bibliotecas abertas NEAT-Python 0.92 e Gym 0.9.4. A versão final do código fonte está no Github (*github.com/arthurpaulino/neat_gym*) e os parâmetros fornecidos ao algoritmo NEAT estão evidenciados na Tabela 7 do Apêndice C. Os experimentos foram realizados no sistema operacional Arch Linux x86_64, Kernel 4.15.14-1-ARCH, que tinha a sua disposição um processador Intel i5-7200U (4) a 3,1 GHz e memória RAM suficiente (7,6 GiB). Vale ressaltar que a biblioteca NEAT-Python 0.92 também implementa a remoção de conexões como possibilidade de mutação estrutural, o que intensifica a busca pelas topologias mais simples.

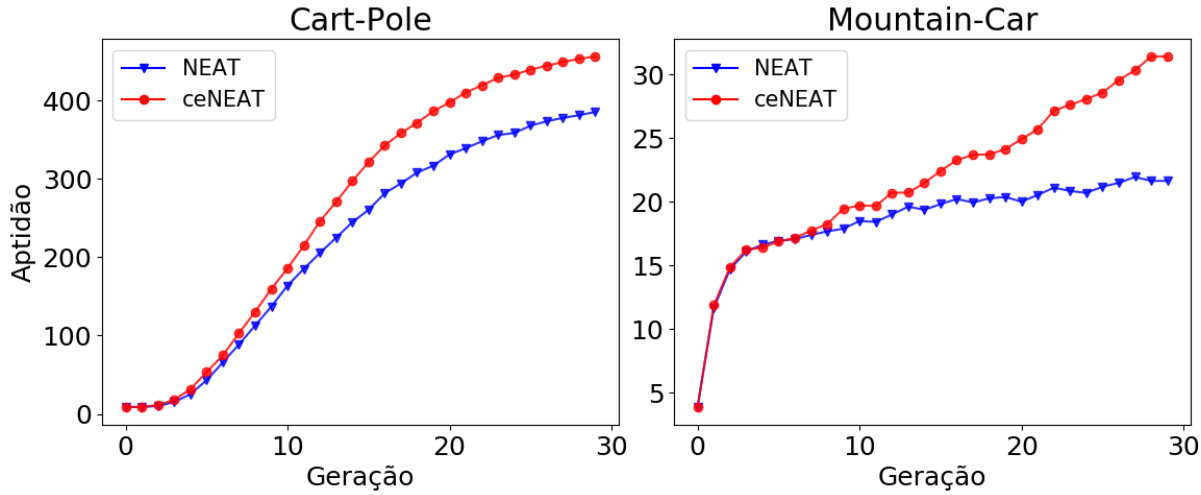
4.3 Resultados

Como explicado na Seção 4.2, os experimentos ocorreram em três fases: *i*) Validação do ceNEAT; *ii*) Testes com BPG e *iii*) Testes com ELM, a seguir.

4.3.1 Validação do ceNEAT

Nesta parte, foram realizados experimentos comparando o NEAT comum com o ceNEAT. Esta fase tem por objetivo verificar a efetividade das estratégias de melhorias culturais no NEAT. As topologias de algumas soluções e suas aptidões para os problemas *Cart-Pole* e *Mountain-Car* estão no Apêndice D.

Figura 18 – NEAT vs. ceNEAT



Fonte: Elaborado pelo autor

Na Figura 18 é possível observar um incremento nas aptidões dos melhores indivíduos em relação ao NEAT, confirmando a eficácia do enriquecimento cultural no algoritmo NEAT para os problemas *Cart-Pole* e *Mountain-Car*. As aptidões alcançadas pelo ceNEAT para tais problemas foram 456,23 e 31,40, respectivamente. Estes valores foram utilizados como alvos para os teste de eficiência, nos quais ambas as estratégias deveriam alcançar a mesma aptidão em cada problema porém sem limite superior para o número de gerações. A Tabela 2 mostra as médias dos tempos necessários para que cada estratégia alcançasse a aptidão alvo.

Tabela 2 – Eficiência do ceNEAT

Estratégia	*Cart-Pole* (s)	*Mountain-Car* (s)
NEAT	11,65	327,59
ceNEAT	5,23	72,28

Fonte: Elaborado pelo autor

O algoritmo ceNEAT não só foi capaz de gerar soluções melhores ao longo das gerações como também alcançou soluções de mesma qualidade em menos tempo.

4.3.2 Testes com BPG

Aqui se inicia a sequência de experimentos cujo objetivo é explorar variações na mecânica do enriquecimento cultural no algoritmo NEAT aplicado aos problemas *Cart-Pole* e *Mountain-Car*. A Tabela 3 é uma compilação das estratégias utilizadas e seus respectivos conjuntos de parâmetros.

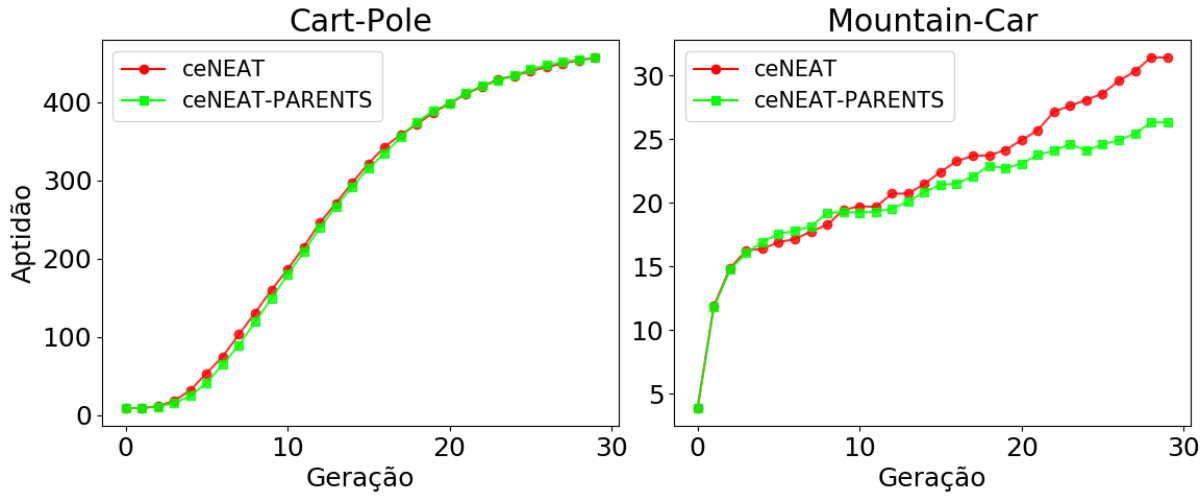
A Figura 19 mostra que a variante ceNEAT-PARENTS alcançou soluções tão boas

Tabela 3 – Configurações dos experimentos com BPG

Estratégia	Aprendizes	Surgimento de cultura	Natureza das lições
ceNEAT	Filhos	Inicial	Aleatória
ceNEAT-PARENTS	Pais	Inicial	Aleatória
ceNEAT-LATE	Filhos	Tardio	Aleatória
ceNEAT-EXP	Filhos	Inicial	Experiência

Fonte: Elaborado pelo autor

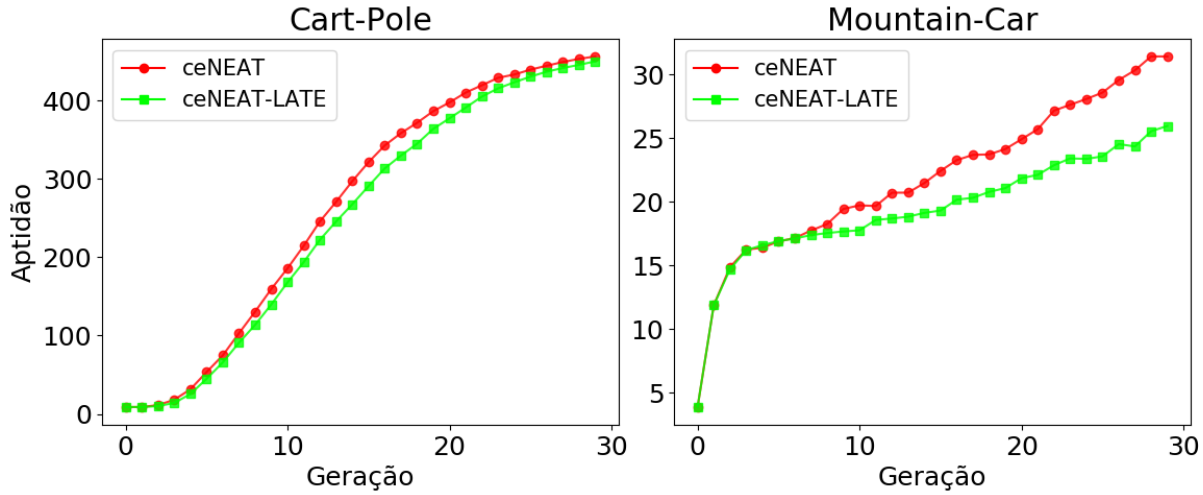
Figura 19 – BPG: Pais como aprendizes



Fonte: Elaborado pelo autor

quanto o ceNEAT para o problema *Cart-Pole*. Isto é um resultado positivo, pois a quantidade de pais em uma geração é bem inferior à quantidade de filhos, resultando em um custo computacional menor. No entanto, o mesmo não aconteceu para o problema *Mountain-Car*.

Figura 20 – BPG: Surgimento tardio de cultura

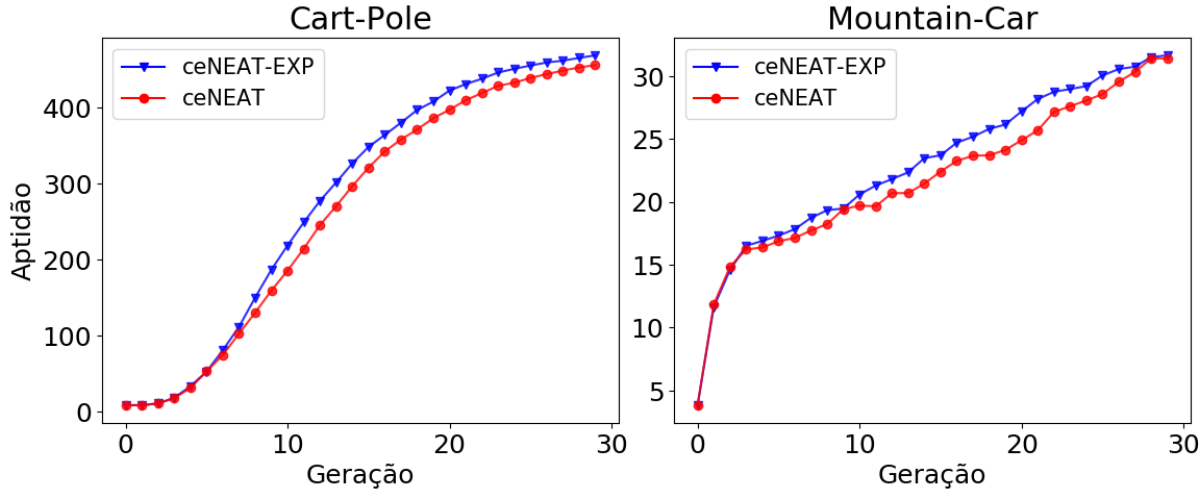


Fonte: Elaborado pelo autor

Na Figura 20 percebe-se que iniciar o processo de aprendizagem tardiamente piorou a qualidade das soluções obtidas, indicando que o enriquecimento cultural por meio do algoritmo

BPG não prejudicou a exploração inicial do espaço amostral para os problemas avaliados.

Figura 21 – BPG: Lições com base na experiência



Fonte: Elaborado pelo autor

Na Figura 21 é possível observar uma melhoria em relação ao ceNEAT quando as lições que compõem o *syllabus* são extraídas da experiência do tutor, indicando que é mais eficaz ensinar sobre situações com maiores probabilidades de serem enfrentadas pelos aprendizes.

4.3.3 Testes com ELM

Nesta parte será explorado outro algoritmo de aprendizagem para a avaliação dos parâmetros dispostos na Tabela 3. Mas antes, é necessário encontrar uma taxa de aprendizagem para o ELM e avaliar a efetividade deste método para o enriquecimento cultural do NEAT. A Tabela 4 apresenta as taxas de aprendizagem testadas.

Tabela 4 – Estratégias de exploração de taxas de aprendizagem para o algoritmo de aprendizagem ELM

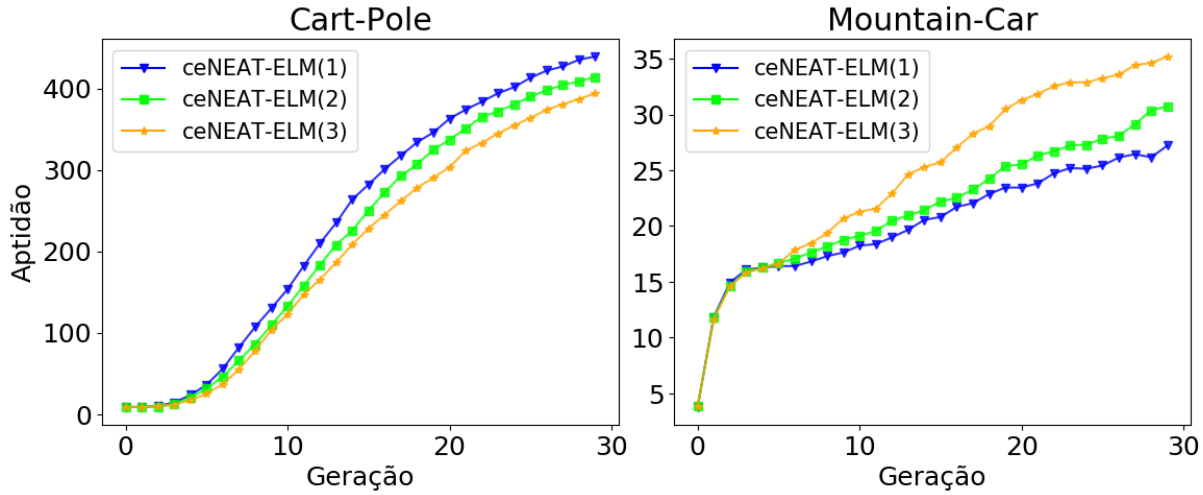
Estratégia	Taxa de aprendizagem
ceNEAT-ELM(1)	0,1
ceNEAT-ELM(2)	0,2
ceNEAT-ELM(3)	0,3

Fonte: Elaborado pelo autor

A Figura 22 mostra que as melhores dentre as taxas testadas para os problemas *Cart-Pole* e *Mountain-Car* foram 0,1 e 0,3, respectivamente. Estas taxas foram utilizadas nos outros experimentos que envolvem o método ELM.

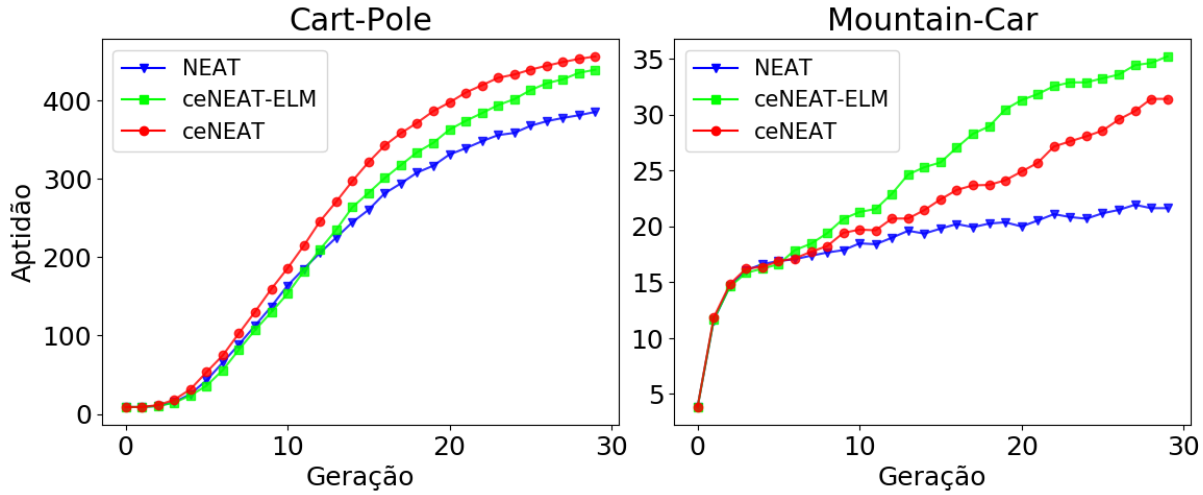
Para avaliar a eficácia do enriquecimento cultural feito com o algoritmo ELM, a

Figura 22 – ELM: Exploração de taxas de aprendizagem



Fonte: Elaborado pelo autor

Figura 23 – ELM: Validação de eficácia



Fonte: Elaborado pelo autor

Figura 23 mostra esta estratégia frente ao NEAT comum e ao ceNEAT. Observa-se que embora a variante ceNEAT-ELM tenha sido pior que o ceNEAT, suas soluções foram melhores que as alcançadas pelo NEAT comum para o problema *Cart-Pole*. Já para o problema *Mountain-Car*, a variante ceNEAT-ELM foi melhor que o NEAT comum e o ceNEAT. Agora resta verificar se os tempos são condizentes com estes resultados.

Tabela 5 – Eficiência do algoritmo de aprendizagem ELM

Estratégia	*Cart-Pole* (s)	*Mountain-Car* (s)
NEAT	11,65	327,59
ceNEAT	5,23	72,28
ceNEAT-ELM	6,45	48,94

Fonte: Elaborado pelo autor

A Tabela 5 mostra os tempos que cada estratégia precisou para alcançar as aptidões 456,23 e 31,40 nos problemas *Cart-Pole* e *Mountain-Car*, respectivamente. Refletindo os resultados da Figura 23, a variante ceNEAT-ELM obteve resultados intermediários no problema *Cart-Pole* e alcançou o menor dos tempos no problema *Mountain-Car*. Isto indica que, assim como para o algoritmo BPG, o enriquecimento cultural realizado com o algoritmo ELM não causa aumentos indesejáveis no tempo de execução do algoritmo NEAT.

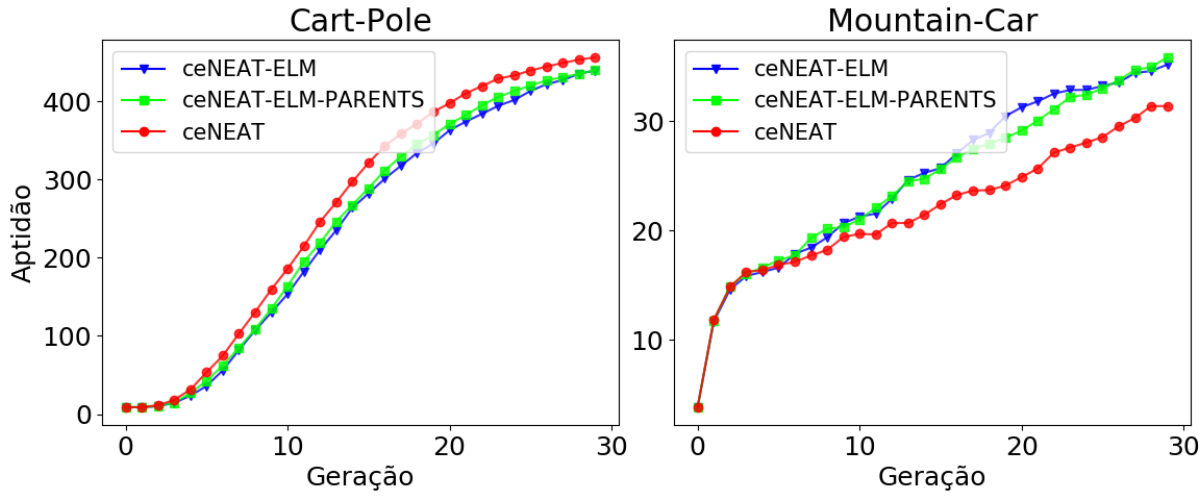
Seguem então os testes referentes às demais variantes. Semelhante à Tabela 3, a Tabela 6 apresenta as estratégias que foram testadas utilizando o método ELM. Os resultados da estratégia ceNEAT também serão expostos para que se perceba a variabilidade das qualidades das soluções.

Tabela 6 – Configurações dos experimentos com ELM

Estratégia	Aprendizes	Surgimento de cultura	Natureza das lições
ceNEAT	Filhos	Inicial	Aleatória
ceNEAT-ELM	Filhos	Inicial	Aleatória
ceNEAT-ELM-PARENTS	Pais	Inicial	Aleatória
ceNEAT-ELM-LATE	Filhos	Tardio	Aleatória
ceNEAT-ELM-EXP	Filhos	Inicial	Experiência

Fonte: Elaborado pelo autor

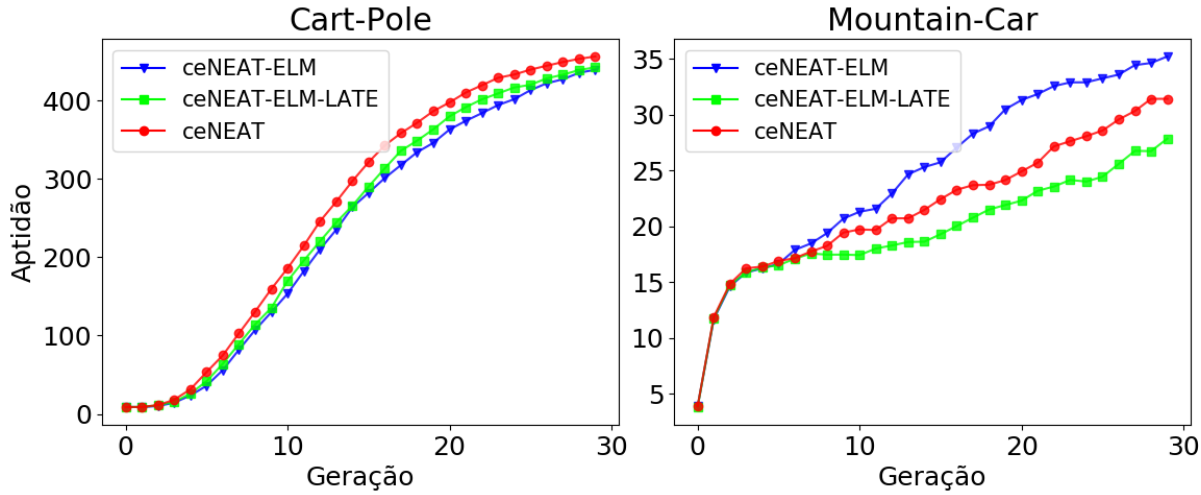
Figura 24 – ELM: Pais como aprendizes



Fonte: Elaborado pelo autor

Pela Figura 24, novamente é possível observar que submeter ou os pais ou os filhos à tutoria acarretam em soluções com qualidades semelhantes, mas ainda inferiores ao ceNEAT para o problema *Cart-Pole*. Este é um indicativo de que a Mecânica Lamarckiana cumpre um papel importante para melhorar a eficiência do enriquecimento cultural.

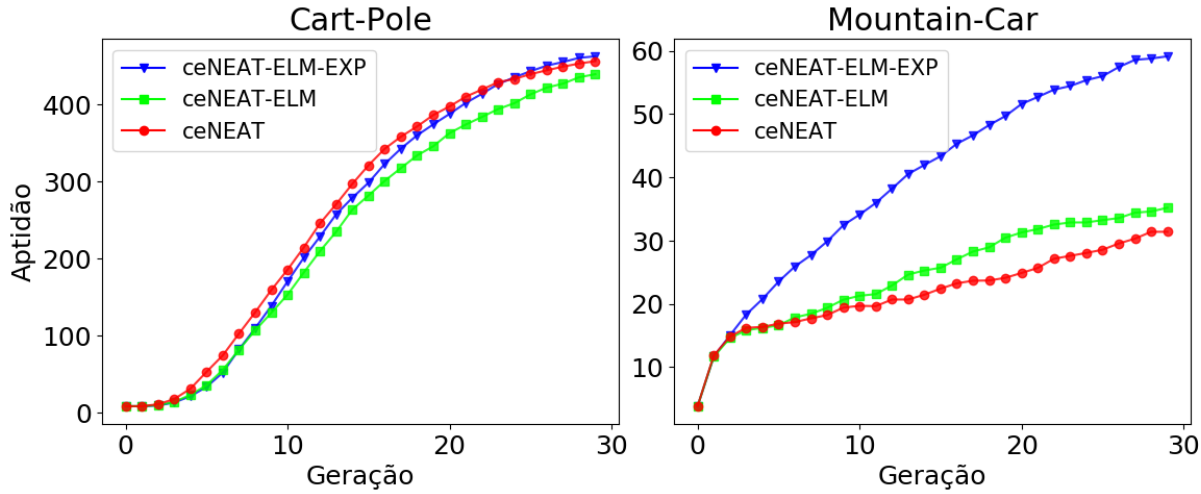
Figura 25 – ELM: Surgimento tardio de cultura



Fonte: Elaborado pelo autor

Na Figura 25 não é possível distinguir um padrão de melhoria na qualidade das soluções quando cultura é introduzida tardiamente no processo evolutivo para o problema *Cart-Pole*, pois em algumas gerações à frente da 15, a estratégia ceNEAT-ELM-LATE pareceu melhorar em relação à ceNEAT-ELM, porém essa diferença se tornou inexpressiva na geração 30. Mas como introduzir cultura tardiamente no sistema reduz o custo computacional, este acaba sendo um resultado positivo para a estratégia ceNEAT-ELM-LATE. Com relação ao problema *Mountain-Car*, esta variante simplesmente piorou a qualidade das soluções.

Figura 26 – ELM: Lições com base na experiência



Fonte: Elaborado pelo autor

A Figura 26 mostra que utilizar lições baseadas na experiência do tutor foi melhor que utilizar lições aleatórias. Para o problema *Cart-Pole*, os resultados foram até melhores que o ceNEAT, embora o algoritmo ELM não tenha funcionado tão bem quanto o BPG no geral.

Para o problema *Mountain-Car*, esta variante proporcionou um salto na qualidade das soluções, dando à curva do gráfico um formato distinto das demais. Esta diferença se deve ao fato da variante ceNEAT-ELM-EXP ter encontrado uma maior proporção de soluções realmente capazes de alcançar a bandeira no problema *Mountain-Car*.

4.4 Conclusão

Este capítulo expôs os experimentos que foram realizados, a lógica que os conecta e os resultados obtidos. Sua meta foi testar a eficácia do NEAT enriquecido com cultura bem como a validade das modificações propostas: o algoritmo de aprendizagem, o momento de inserção de cultura no processo evolutivo, os indivíduos a serem submetidos ao treinamento e a origem das lições do *syllabus*.

5 CONCLUSÃO

Esta dissertação se iniciou com uma motivação para o estudo realizado, seguida do arcabouço teórico necessário e da apresentação da proposta propriamente dita. Logo após, foram expostos os resultados dos experimentos utilizados para a validação da proposta. Para finalizar, este breve capítulo liga os resultados dos experimentos às hipóteses motivadores deste trabalho.

Após a validação da inserção de cultura feita com ambos algoritmos de aprendizagem frente ao NEAT comum, o primeiro ponto a se observar foi que o NEAT foi capaz de alcançar melhores resultados com o enriquecimento cultural. Os algoritmos de aprendizagem surtiram efeitos distintos dentre as tarefas utilizadas: para a tarefa *Cart-Pole*, em geral, o enriquecimento cultural feito com o algoritmo BPG alcançou melhores resultados do que com o algoritmo ELM, ao contrário do que foi observado no problema *Mountain-Car*.

Das variações avaliadas, a que apresentou resultados menos notórios foi iniciar o enriquecimento cultural tardiamente, mostrando a importância da cultura para o NEAT nas tarefas testadas desde o início do processo evolutivo. Portanto, no geral, a restrição no espaço de busca bem como a construção de *syllaba* por tutores imaturos trouxeram mais benefícios que malefícios, ao contrário do que se esperava. É possível que as mutações estruturais e o fato do tutor mudar frequentemente amenizem os efeitos das hipóteses que motivaram esta variante.

Com relação aos indivíduos submetidos ao treinamento, os experimentos mostraram bons resultados quando apenas os pais aprenderam com as lições dos tutores. Com exceção dos experimentos feitos com BPG na tarefa *Mountain-Car*, as qualidades das soluções alcançadas foram semelhantes quando se treinava os pais ou os filhos. Isto é um bom resultado porque cada geração tem bem menos pais do que filhos, ocasionando um menor custo computacional oriundo do treinamento.

A última variação diz respeito à natureza das lições ensinadas aos aprendizes: aleatórias ou baseadas na experiência dos tutores. Os experimentos mostraram bons resultados para a segunda opção, principalmente para o algoritmo de aprendizado ELM na tarefa *Mountain-Car*. Uma possível causa para isso é que em ambos os casos os aprendizes treinaram com a mesma quantidade de lições e convergem em direção aos tutores com a mesma velocidade. Logo, a restrição no espaço de busca nos dois casos é semelhante. Porém, os que treinaram com lições extraídas de situações reais aprendem a lidar melhor com as dificuldades das provas de aptidão.

Conclui-se então que o enriquecimento cultural, bem como a maioria das variações propostas para a sua implementação, se mostraram salutares ao algoritmo NEAT, alcançando-se

assim os objetivos gerais do presente trabalho. Além disso, com relação aos objetivos específicos, a recomendação para a execução do ceNEAT é que se utilize lições com base na experiência do tutor, pois esta variante alcançou bons resultados em ambos os problemas e para ambos métodos de aprendizagem. Não há uma recomendação específica para o algoritmo de aprendizagem ideal, pois este depende do problema em questão. A variante na qual se submete apenas os pais ao treinamento diminui o tempo empreendido na tutoria, mas pode ocasionar queda na qualidade das soluções, assim como para a variante na qual a cultura é introduzida no sistema tardiamente.

5.1 Trabalhos Futuros

Cultura foi tratada nesta dissertação como conhecimento transmitido de uma geração para a seguinte. No entanto, este processo foi feito de forma rígida, ou seja, havia uma fase em cada geração na qual alguns indivíduos estavam destinados a passar por um treinamento. Mas como a cultura se manifestaria em um sistema mais livre?

NOGUEIRA *et al.*, 2016, propôs uma codificação para Algoritmos Neuroevolutivos com o intuito de estudar os padrões emergentes quando não há função objetivo definida explicitamente e quando os indivíduos são capazes de interagir entre si por compartilharem o mesmo ambiente de simulação. Seu trabalho é uma porta para a investigação da pergunta acima.

Dando aos agentes a capacidade de aprender com os que estão em seu campo de visão e de escolherem se iriam mesmo fazê-lo, seria possível verificar se a cultura surgiria segundo algum padrão interpretável? Será se em gerações avançadas os indivíduos se tornariam mais criteriosos com relação a quem seriam seus tutores? O que caracterizaria um bom tutor? Já que os indivíduos têm diferentes topologias de rede e portanto diferentes capacidades de aprendizagem, surgiriam grupos de bons aprendizes? Todas estas perguntas motivam um aprofundamento no tema.

REFERÊNCIAS

BALDWIN, J. M. A new factor in evolution. **The american naturalist**, Edwards and Docker, v. 30, n. 354, p. 441–451, 1896.

BARTO, A. G. **Neural Networks for Control**. Cambridge, MA, USA: MIT Press, 1990. 5–58 p. ISBN 0-262-13261-3.

BARTO, A. G.; SUTTON, R. S.; ANDERSON, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. **IEEE Transactions on Systems, Man, and Cybernetics**, SMC-13, n. 5, p. 834–846, Sept 1983. ISSN 0018-9472.

BEASLEY, D.; BULL, D. R.; MARTIN, R. R. An overview of genetic algorithms: Part 1, fundamentals. **University computing**, v. 15, n. 2, p. 56–69, 1993.

CECCONI, F.; MENCZER, F.; BELEW, R. K. Maturation and the evolution of imitative learning in artificial organisms. **Adaptive Behavior**, SAGE Publications Sage CA: Thousand Oaks, CA, v. 4, n. 1, p. 29–50, 1995.

DARWIN, C. R. **On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life**. United Kingdom of Great Britain and Ireland.: John Murray, 1859.

DAWKINS, R. **The Selfish Gene**. New York: Oxford University Press, 1976. ISBN 019857519.

GOMEZ, F. J.; MIIKKULAINEN, R. Solving non-markovian control tasks with neuroevolution. In: **IJCAI**. [S.l.: s.n.], 1999. v. 99, p. 1356–1361.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016.

GRUAU, F.; WHITLEY, D.; PYEATT, L. Adding learning to the cellular development of neural networks: Evolution and the baldwin effect. **Evolutionary Computation**, v. 1, p. 213–233, 1993.

HASTINGS, E. J.; GUHA, R. K.; STANLEY, K. O. Automatic content generation in the galactic arms race video game. **IEEE Transactions on Computational Intelligence and AI in Games**, IEEE, v. 1, n. 4, p. 245–263, 2009.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: theory and applications. **Neurocomputing**, Elsevier, v. 70, n. 1-3, p. 489–501, 2006.

JULSTROM, B. A. Comparing darwinian, baldwinian, and lamarckian search in a genetic algorithm for the 4-cycle problem. In: **Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference**. [S.l.: s.n.], 1999. p. 134–138.

KEESING, R.; STORK, D. G. Evolution and learning in neural networks: the number and distribution of learning trials affect the rate of evolution. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 1991. p. 804–810.

LAMARCK, J.-B. P. A. de Monet de. **Philosophie Zoologique**. França: Museum d'Histoire Naturelle (Jardin des Plantes), 1809.

LEHMAN, J.; MIIKKULAINEN, R. Neuroevolution. **Scholarpedia**, v. 8, n. 6, p. 30977, 2013.

LOUIS, S. J.; JOHNSON, J. Solving similar problems using genetic algorithms and case-based memory. In: **In Proceedings of the Seventh International Conference on Genetic Algorithms**. [S.l.]: Morgan Kauffman, 1997. p. 283–290.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. 1943. **Bulletin of mathematical biology**, v. 52 1-2, p. 99–115; discussion 73–97, 1988.

MCQUESTEN, P. H. **Cultural Enhancement of Neuroevolution**. Tese (Doutorado) — The University of Texas at Austin, Austin, TX 78712, 2002.

MELANIE, M. **An Introduction to Genetic Algorithms**. London, England - Cambridge, Massachusetts: The MIT Press, 1999. 7 - 10, 128 - 130 p.

MIIKKULAINEN, R.; LIANG, J. Z.; MEYERSON, E.; RAWAL, A.; FINK, D.; FRANCON, O.; RAJU, B.; SHAHRZAD, H.; NAVRUZYAN, A.; DUFFY, N.; HODJAT, B. Evolving deep neural networks. **CoRR**, abs/1703.00548, 2017.

MOORE, A. W. **Efficient memory-based learning for robot control**. Tese (Doutorado) — University of Cambridge, Trinity Hall, 1990.

MORIARTY, D. E. **Symbiotic Evolution Of Neural Networks In Sequential Decision Tasks**. 117 p. Tese (Doutorado) — Department of Computer Sciences, The University of Texas at Austin, 1997. Technical Report UT-AI97-257.

NOGUEIRA, Y. L. B.; BRITO, C. E. F. de; VIDAL, C. A.; NETO, J. B. C. emergent vision system of autonomous virtual characters. **Neurocomputing**, v. 173, p. 1851 – 1867, 2016. ISSN 0925-2312.

NOLFI, S.; ELMAN, J. L.; PARISI, D. Learning and evolution in neural networks. **Adaptive Behavior**, v. 3, p. 5–28, 1990.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. **On the difficulty of training Recurrent Neural Networks**. 2013.

PETERSEN, K. B.; PEDERSEN, M. S. **The Matrix Cookbook**. [S.l.]: Technical University of Denmark, 2012. 8, 11 p. Version 20121115.

ROJAS, R. **Neural Networks - A Systematic Introduction**. Berlin, New-York: Springer-Verlag, 1996. 3 - 27, 151 - 184 p.

SASAKI, T.; TOKORO, M. Adaptation toward changing environments: Why darwinian in nature? In: **In P. Husbands and I. Harvey (Eds.), Fourth European Conference on Artificial Life (ECAL97**. [S.l.]: MIT Press, 1997. p. 145–153.

SCHAFFER, J. D.; WHITLEY, D.; ESHELMAN, L. J. Combinations of genetic algorithms and neural networks: a survey of the state of the art. In: **[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks**. [S.l.: s.n.], 1992. p. 1–37.

SILVA, F.; URBANO, P.; CORREIA, L.; CHRISTENSEN, A. L. odneat: An algorithm for decentralised online evolution of robotic controllers. **Evolutionary Computation**, MIT Press, v. 23, n. 3, p. 421–449, 2015.

SPECTOR, L.; LUKE, S. **Cultural Transmission of Information in Genetic Programming**. 1996.

STANLEY, K. O. **Efficient Evolution of Neural Networks Through Complexification**. Tese (Doutorado) — Department of Computer Sciences, The University of Texas at Austin, 2004.

STANLEY, K. O.; BRYANT, B. D.; MIIKKULAINEN, R. Real-time neuroevolution in the nero video game. **IEEE transactions on evolutionary computation**, IEEE, v. 9, n. 6, p. 653–668, 2005.

STANLEY, K. O.; D'AMBROSIO, D. B.; GAUCI, J. A hypercube-based encoding for evolving large-scale neural networks. **Artificial life**, MIT Press, v. 15, n. 2, p. 185–212, 2009.

STANLEY, K. O.; MIIKKULAINEN, R. Efficient evolution of neural network topologies. In: **Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)**. [S.l.: s.n.], 2002. v. 2, p. 1757–1762. ISBN 0-7803-7282-4.

STANLEY, K. O.; MIIKKULAINEN, R. Evolving neural networks through augmenting topologies. **Evolutionary Computation**, v. 10, p. 99 – 127, 2002.

WHITLEY, D.; GORDON, V. S.; MATHIAS, K. **Lamarckian Evolution, The Baldwin Effect and Function Optimization**. [S.l.]: Springer-Verlag, 1994. 6–15 p.

APÊNDICE A – PASSO DO PROBLEMA CART-POLE

```
GRAVITY = 9.8
MASS_CART = 1.0
MASS_POLE = 0.1
TOTAL_MASS = (MASS_POLE + MASS_CART)
LENGTH = 1.0
POLE_MASS_LENGTH = (MASS_POLE * LENGTH / 2)
FORCE_MAG = 10.0
TAU = 0.02

def step(self, action):
    x, x_dot, theta, theta_dot = self.state
    force = FORCE_MAG if action==1 else -FORCE_MAG
    costheta = math.cos(theta)
    sintheta = math.sin(theta)
    temp = (force + POLE_MASS_LENGTH * theta_dot *
       theta_dot * sintheta) / TOTAL_MASS
    thetaacc = (GRAVITY * sintheta - costheta * temp) / (
       LENGTH * (2.0/3.0 - MASS_POLE * costheta * costheta
       / TOTAL_MASS))
    xacc  = temp - POLE_MASS_LENGTH * thetaacc * costheta /
        TOTAL_MASS
    x  = x + TAU * x_dot
    x_dot = x_dot + TAU * xacc
    theta = theta + TAU * theta_dot
    theta_dot = theta_dot + TAU * thetaacc

    self.state = (x,x_dot,theta,theta_dot)
    return np.array(self.state)
```

APÊNDICE B – PASSO DO PROBLEMA MOUNTAIN-CAR

```
MIN_POSITION = -1.2
MAX_POSITION = 0.6
MAX_SPEED = 0.07

def step(self, action):
    position, velocity = self.state
    velocity += (action-1) * 0.001 + math.cos(3 * position)
        * (-0.0025)
    velocity = np.clip(velocity, -MAX_SPEED, MAX_SPEED)
    position += velocity
    position = np.clip(position, MIN_POSITION,MAX_POSITION)
    if (position==MIN_POSITION and velocity<0):
        velocity = 0

    self.state = (position, velocity)
    return np.array(self.state)
```

APÊNDICE C – PARÂMETROS DOS EXPERIMENTOS

Tabela 7 – Parâmetros Fornecidos ao NEAT nas tarefas *Cart-Pole* e *Mountain-Car*

Parâmetro	*Cart-Pole*	*Mountain-Car*	Observação
fitness_criterion	max	max	
fitness_threshold	500.0	130.0	456.23 e 31.4 nos testes cronometrados
pop_size	100	100	
reset_on_extinction	True	True	
num_inputs	4	2	Descrições dos estados
num_hidden	0	0	
num_outputs	2	3	Possibilidades de ação
initial_connection	partial 0.01	partial 0.01	Inicia com redes rarefeitas
feed_forward	True	True	
compatibility_disjoint_coefficient	1.0	1.0	$c_1 \leftarrow c_2$ (Equação 2.1)
compatibility_weight_coefficient	1.0	1.0	$c_3 \leftarrow 0.0$ se há cultura (Equação 2.1)
conn_add_prob	0.2	0.2	
conn_delete_prob	0.2	0.2	Particular ao NEAT-Python 0.92
activation_default	sigmoid	sigmoid	
activation_options	sigmoid	sigmoid	
activation_mutate_rate	0.0	0.0	
aggregation_default	sum	sum	
aggregation_options	sum	sum	
aggregation_mutate_rate	0.0	0.0	
bias_init_mean	0.0	0.0	
bias_init_stdev	1.0	1.0	
bias_replace_rate	0.1	0.1	
bias_mutate_rate	0.7	0.7	
bias_mutate_power	0.5	0.5	
bias_max_value	30.0	30.0	
bias_min_value	-30.0	-30.0	
response_init_mean	1.0	1.0	
response_init_stdev	0.0	0.0	
response_replace_rate	0.0	0.0	
response_mutate_rate	0.0	0.0	
response_mutate_power	0.0	0.0	
response_max_value	30.0	30.0	
response_min_value	-30.0	-30.0	
weight_max_value	30.0	30.0	
weight_min_value	-30.0	-30.0	
weight_init_mean	0.0	0.0	
weight_init_stdev	1.0	1.0	
weight_mutate_rate	0.8	0.8	
weight_replace_rate	0.1	0.1	
weight_mutate_power	0.5	0.5	
enabled_default	True	True	
enabled_mutate_rate	0.01	0.01	
compatibility_threshold	3.0	3.0	
species_fitness_func	max	max	
max_stagnation	20	20	
elitism	5	5	
survival_threshold	0.2	0.2	

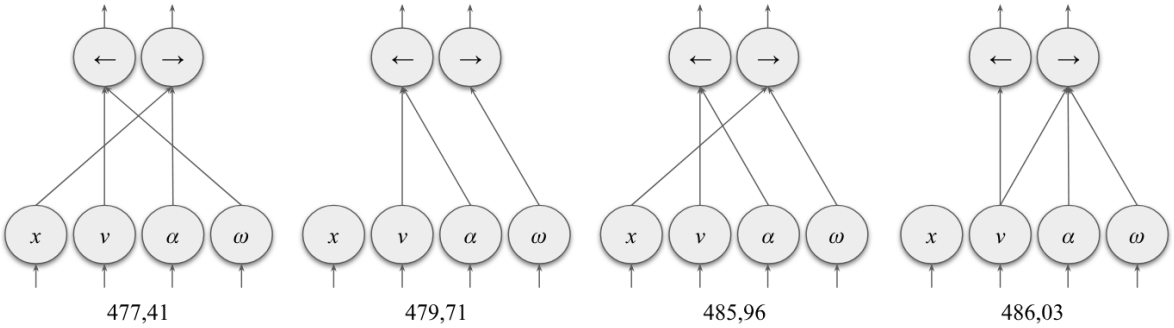
Fonte: Elaborado pelo autor

APÊNDICE D – TOPOLOGIAS DE ALGUMAS SOLUÇÕES

Símbolo	Significado
x	Posição horizontal
v	Velocidade horizontal
α	Ângulo da vareta
ω	Velocidade angular da vareta
$\leftarrow$	Movimentar para a esquerda
$\rightarrow$	Movimentar para a direita
$\leftrightarrow$	Não movimentar

Cart-Pole

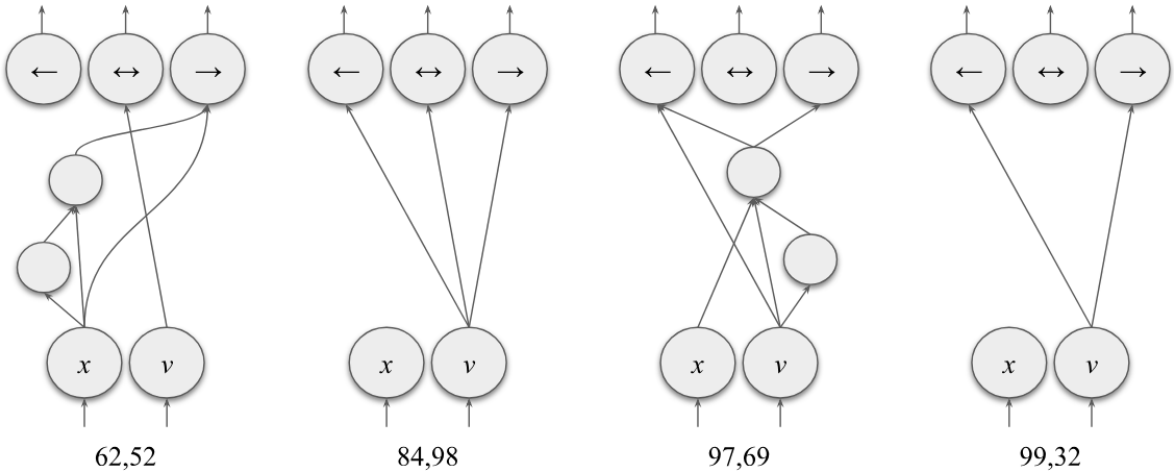
Figura 27 – Topologias (sem pesos) para o problema *Cart-Pole* e suas respectivas aptidões



Fonte: Elaborado pelo autor

Mountain-Car

Figura 28 – Topologias (sem pesos) para o problema *Mountain-Car* e suas respectivas aptidões



Fonte: Elaborado pelo autor