CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U nº 198, de 14/10/2016
ASSOCIAÇÃO EDUCACIONAL LUTERANA DO BRASIL

Haroldo Fernando Fritsch

UTILIZAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA CONTROLAR UM AGENTE EM UM JOGO DE ESTRATÉGIA EM TEMPO REAL

Palmas – TO

2016

Haroldo Fenando Fritsch

UTILIZAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA CONTROLAR UM AGENTE EM UM JOGO DE ESTRATÉGIA EM TEMPO REAL

Trabalho de Conclusão de Curso (TCC) elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e M.Sc. Jackson Gomes de Souza.

Palmas – TO

2016

Haroldo Fernando Fritsch

UTILIZAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA CONTROLAR UM AGENTE EM UM JOGO DE ESTRATÉGIA EM TEMPO REAL

Trabalho de Conclusão de Curso (TCC) elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e M.Sc. Jackson Gomes de Souza.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e M.Sc. Jackson Gomes de Souza

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.Sc Parcilene Fernandes de Brito

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. D.r. Edeílson Milhomem da Silva

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2016

Dedico este trabalho aos meus pais, irmãos, minha noiva Sidna, minha filha Sarah e meu filho Vinícius e a toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais que em todos os momentos estiveram presentes e incondicionalmente dispostos a me auxiliarem nas minhas escolhas, apoiando minhas decisões e me auxiliando a alcançar meus objetivos acadêmicos, profissionais e pessoais. Com carinho e respeito me passaram valores que formaram a pessoa que sou hoje. Eu amo vocês e orgulho-me todos os dias de tê-los como pais. Obrigado por tudo!

Agradeço aos meus filhos que mesmo tão pequenos e distantes de mim, foram fonte de incentivo para eu continuar no meu caminho para que um dia, quem sabe, eu possa servir de exemplo para vocês. Sarah e Vinícius, amo vocês!

Eu gostaria de agradecer também ao Victor Martins, Daniel Piccoli e ao Ranyelson por serem os meus companheiros de guerra durante a maior parte do curso em forma de trabalhos em grupo em diversas disciplinas.

Ao meu orientador e professor Jackson Gomes que sempre encontrou disponibilidade para tirar qualquer dúvida sobre qualquer tema, pela paciência para discutirmos sobre este trabalho. Muito obrigado professor!

Em resumo, um muito obrigado a todos os professores pela a amizade de vocês e por acreditarem em mim desde o começo.

E por último, mas não menos importante, à minha noiva Sidna Rodrigues que sempre esteve ao meu lado até mesmo nos momentos de desânimo com este trabalho, me dando carinho e força para enfrentar as dificuldades encontradas e que só foram resolvidas por seu incentivo. Muito obrigado, meu anjo!

RESUMO

Atualmente os jogos de computadores são muito populares e deram origem a uma indústria bilionária. Um dos marcos que permitiu esse avanço foi o emprego da Inteligência Artificial, que forneceu mais realismo e imersão aos jogos. No contexto acadêmico, os jogos eletrônicos se apresentam como uma excelente plataforma de testes e validação de novos métodos e algoritmos de Inteligência Artificial, em especial o gênero de jogos de estratégia em tempo real, que fornece inúmeros desafios como estratégia de coleta e alocação de recursos. Este trabalho apresenta um método de aprendizagem supervisionada através de uma Rede Neural Artificial com utilização do algoritmo *backpropagation* para um NPC sobre o ambiente do jogo *Starcraft*.

Palavras-chave: Inteligência Artificial, Jogos de Estratégia em Tempo Real, Redes Neurais Artificiais, Jogo *Starcraft* e Algoritmo *Backpropagation.*

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D – Bidimensional

3D – Tridimensional

BWAPI - Brood War Application Programming Interface

IA – Inteligência Artificial

RNA – Rede Neural Artificial

RTS – Jogos de Estratégia em Tempo Real

TBS – Jogos de Estratégia em Turnos

ULBRA – Universidade Luterana do Brasil

SUMÁRIO

1. INTRODUÇÃO

Jogos de estratégia em tempo real (jogos RTS, do inglês *Real-Time Strategy games*) são jogos onde um jogador deve se envolver em ações de tempo real, com o objetivo de obter superioridade territorial e militar sobre outros jogadores ou o computador (BURO, 2006). Basicamente, o ponto central de jogos RTS é o domínio de dois problemas chave: a produção de recursos e as batalhas táticas. Na produção de recursos, o jogador deve produzir ou recolher diversas matérias-primas, edifícios, unidades militares e civis, com o objetivo de melhorar sua situação econômica e seu poder militar. Em batalhas táticas, um jogador utiliza suas unidades militares para ganhar território e derrotar as unidades inimigas.

Um típico jogo RTS envolve um período inicial no qual o jogador deve construir sua construir sua economia através da produção de recursos e seguida pelas campanhas militares. Os recursos serão explorados nas campanhas militares tanto no ataque, como na defesa. Assim, a rápida produção de recursos é muitas vezes um fator chave no sucesso global (CHAN et. al, 2007).

O jogo *Starcraft* foi desenvolvido pela *Blizzard Entertainment* é um dos principais jogos para esse gênero de jogo. Lançado em 1998 para o sistema operacional *Windows*, é considerado uma referência para jogos RTS, pois ajudou a definir as principais características desses jogos (WAYWARD, 2015).

O mercado de jogos eletrônicos, segundo Tatai (2003), compõe um mercado bilionário em crescimento. Em 2015, o mercado global de jogos faturou mais de US$ 90 bilhões de dólares, de acordo com um relatório da empresa de pesquisa *Newzoo*[1] (2016). Isso representa um aumento de 9,4% por cento em relação a 2014.

Nesse contexto de crescimento, também cresceu a demanda pelo desenvolvimento de novas tecnologias, o que pode ser facilmente observado pela evolução de vários aspectos dos jogos como, por exemplo, os recursos gráficos e os mecanismos de controle, como sensores de movimento e outros. A evolução de outros aspectos, como as técnicas e os algoritmos de Inteligência Artificial (IA), no entanto, não é facilmente percebida pelos jogadores por se tratarem de mecanismos internos (CASTRO, 2012).

Além disso, segundo Tatai (2003), o emprego da IA nos jogos atuais não é tão bem desenvolvido por diversos motivos, sendo o maior deles a alta complexidade de um jogo RTS. Essa complexidade está associada à natureza de um jogo RTS, onde, dentro de um pequeno

[1] Disponível a partir do endereço https://newzoo.com/solutions/consumer-insights/gamers

intervalo de tempo, centenas de ações são executadas simultaneamente pelos jogadores, o que demanda uma grande capacidade de processamento dos dispositivos que executam os jogos.

A IA, responsável por implementar e controlar os oponentes e aliados do jogador, está ganhando espaço no desenvolvimento de jogos eletrônicos nos últimos anos. Segundo Borges *et. al* (2009) esse destaque deve-se ao fato de que, continuamente, os jogadores buscam não só jogos de qualidade, mas personagens de jogos que atuem de forma inteligente.

Os personagens de jogos, chamados de agentes autônomos (NPC, do inglês *Non-Player Character*), são controlados pelo "computador" através da IA. O NPC pode ser aliado, espectador ou concorrente do jogador. Para a criação de um NPC, várias técnicas foram utilizadas ao longo dos anos, como: Máquina de Estados Finitos, Padrões de Movimento, dentre outras. Técnicas clássicas de IA, como Redes Neurais Artificiais (RNA, do inglês *Artificial Neural Networks*), Algoritmos Genéticos (AG, do inglês *Genetic Algorithms***)** e Lógica *Fuzzy* foram menos utilizadas em jogos comerciais, mas, segundo Neto (2011), no futuro, essas deverão ser mais estudadas pelos desenvolvedores de jogos.

Segundo Champandard (2003), o emprego da IA em um jogo tem o objetivo de facilitar o controle do comportamento de um NPC provendo três aspectos: comportamento primitivo, movimento e tomada de decisão. No primeiro aspecto, a IA proporciona ao NPC a capacidade de capturar itens, usar objetos, gesticular e outros. O segundo aspecto permite ao NPC movimentar-se em uma determinada área do cenário, desviando-se dos obstáculos. E no último, a tomada de decisão, que está acima dos outros dois aspectos, a IA possibilita ao NPC calcular as ações necessárias para cumprir seus objetivos.

Essas facilidades sobre o controle do comportamento de um NPC também podem ser implementadas com a programação tradicional. Contudo, a IA traz soluções mais elaboradas, gerando comportamentos mais qualitativos. Isto é possível através da utilização de funcionalidades como reconhecimento de padrões, predição, árvores de decisão, RNA, algoritmo evolutivo, entre outras.

RNA é uma técnica de IA inspirada em sistemas biológicos e é empregada em aplicações que utilizam tomadas de decisões, processamento de informações e otimização (HAYKIN, 2001). Sua utilização em jogos RTS permite o aprendizado do comportamento de um NPC, no sentido de imitar o comportamento de um jogador, através do treinamento de parâmetros de controle dos personagens durante a etapa de desenvolvimento do jogo.

Os jogos RTS se apresentam como uma excelente plataforma de testes e validação de novos métodos e algoritmos de IA, pois neles se encontram ambientes de jogos sofisticados e complexos. Apesar desses jogos não apresentarem a mesma complexidade do mundo real, os

ambientes apresentados são controlados, o que proporciona uma atmosfera ideal para aplicação desses métodos (TATAI, 2003).

O jogo *Starcraft* se encaixa neste contexto, permitindo, portanto, o desenvolvimento de pesquisas que envolvam IA aplicada a problemas de tomada de decisão.

A biblioteca BWAPI (do inglês, *Brood War Application Programming Interface*) é um framework que permite a criação de módulos de IA a serem utilizados no jogo *Starcraft*. Nele, o desenvolvedor tem acesso a todas as informações que o jogo oferece a qualquer jogador, além de permitir aos módulos de IA executarem quaisquer ações, que um jogador regular poderia realizar durante uma partida.

Com o objetivo de reunir conceitos-chave da Ciência da Computação (IA e RNA) na solução de um problema de tomada de decisão em um ambiente não-trivial (o jogo RTS), o presente trabalho propõe a criação de um agente para disputar batalhas entre unidades no jogo *Starcraft*. A unidade controlada pelo agente terá seu comportamento determinado por técnicas de IA e utilizará a biblioteca BWAPI para se comunicar com o jogo.

2. REFERENCIAL TEÓRICO

Nesta seção são apresentados os conceitos necessários para compreensão do trabalho e uma revisão dos trabalhos relacionados encontrados na literatura. Em primeiro lugar será explanada uma definição geral de jogos eletrônicos e uma definição detalhada do gênero de jogos RTS. Depois serão apresentados conceitos básicos de IA para jogos, aprendizagem com RNA e seus elementos constituintes, como: neurônios artificiais, funções de ativação e algoritmo de aprendizagem. E por último, será apresentada uma síntese de um trabalho sobre aprendizagem por reforço com RNA no desenvolvimento de jogos digitais.

2.1 JOGOS ELETRÔNICOS

Piace (2011) define[2] jogo eletrônico como um jogo que se desenvolve na interação entre um jogador humano e um outro jogador, humano ou não, mediado por um dispositivo eletrônico (computador), constituído por componentes físicos (hardware) e por um conjunto de instruções e procedimentos (software), que, previamente programados, determinam como tais elementos físicos podem ser ativados.

O computador, portanto, mantém o estado do jogo, controla a ordem de funcionamento e modo de interação e também calcula a produção de novo estado de jogo através de regras codificadas em seu programa de operação. Os computadores pessoais, videogames e smartphones são alguns exemplos de dispositivos atuais que possibilitam a execução de jogos eletrônicos.

Segundo Bates (2004), a divisão de gêneros dos jogos eletrônicos pode ser obtida através da classificação dos elementos contidos nos jogos, dos objetivos e também pela maneira como os jogos são jogados. Os próximos parágrafos apresentam uma breve descrição dos seguintes gêneros:

- Jogos de Ação
- Jogos de Aventura
- Jogos Casuais
- Jogos Educacionais
- Jogos de Esporte
- Jogos Online
- Jogos de Estratégia
- Jogos de Quebra-cabeça
- Jogos de RPG
- Jogos de Simuladores

[2] Foram desconsideradas na elaboração desta definição as questões relacionadas aos elementos estéticos e narrativos do jogo eletrônico. Para uma análise mais abrangente desta questão, sugere-se a leitura de Gomes (2008).

O gênero de Jogos de Ação[3] é composto por jogos que desafiam os jogadores a reagirem rapidamente aos elementos do jogo. Essa categoria é dominada por jogos de tiro em primeira pessoa. Nessa categoria, pode-se destacar também os jogos híbridos de ação e aventura, onde a mecânica do jogo além de ser constituída de muita ação, também possui elementos que utilizam enigmas para a resolução de problemas. A Figura 1 ilustra dois jogos desse gênero.

Figura 1. Captura de tela dos jogos Counter-Strike e Battlefield, respectivamente.

Fonte: http://apunkagames.net

A categoria Jogos de Aventura é composta por jogos cujo enredo envolve o jogador em histórias com cenários temáticos. Nesses enredos, o jogador deve solucionar enigmas e quebra-cabeças para que possa avançar nas fases do jogo. A Figura 2 ilustra dois jogos: *Assassin's Creed* –, jogos muito populares atualmente, que misturam traços de jogos de ação e aventura –, e *God of War*[4], que também incorporam elementos de quebra-cabeça e ação.

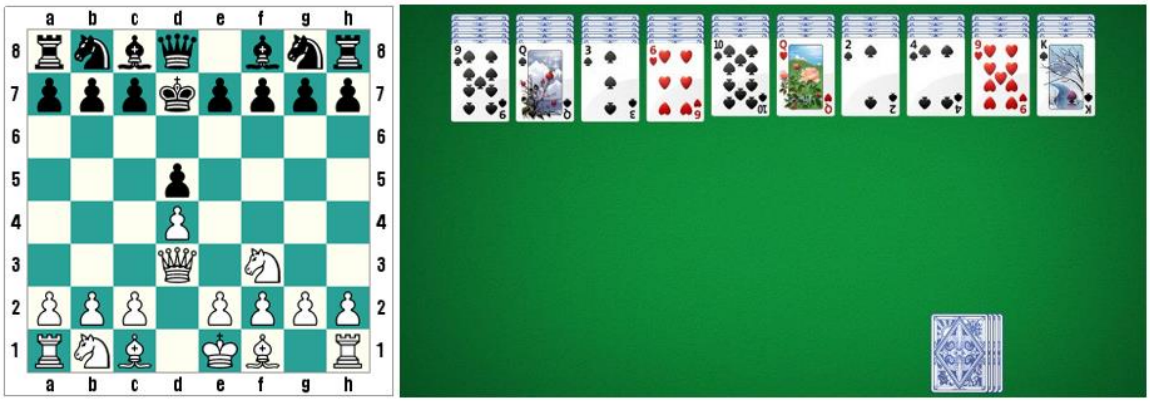
Figura 2. Captura de tela dos jogos Assassin's Creed e God of War, respectivamente.

Fontes: http://pcworld.com e http://2geh.com.

[3] Exemplos do gênero: *Counter-Strike* (produzido pela *Valve Corporation*), *Battlefield* e *Medal of Honor* (produzidos pela EA Digital Illusions CE) e *Call of Duty* (produzido pela *Infinity Ward*).
[4] O jogo *Assassin's Creed* é desenvolvido pela Ubisoft Montreal na engine *Unity* e o jogo *God of War* pela Sony Computer.

O gênero de Jogos Casuais em geral são jogos eletrônicos adaptados de jogos do cotidiano, como xadrez e paciência. Há também jogos desse gênero que não derivam de adaptações, mas de jogos com enredos próprios e baseados em regra simples, onde a principal característica é de um jogo rápido, usado apenas para passar o tempo. A Figura 3 ilustra dois jogos do gênero: o jogo de Xadrez obtido na rede social *Facebook* e o jogo Paciência encontrado na maioria dos sistemas operacionais da *Microsoft Corporation*.

Figura 3. Captura de tela dos jogos Xadrez e Paciência, respectivamente.



Fonte: http://olhardigital.uol.com.br

Os Jogos Educacionais são definidos por uma categoria tem por objetivo fazer com que o jogador aprenda enquanto se diverte. Frequentemente são jogos destinados ao público infantil, onde desenvolvedores e especialistas em educação trabalham lado a lado com o intuito de criar jogos que sirvam de ferramentas de apoio à aprendizagem. A Figura 4 ilustra três jogos educacionais desenvolvidos pela The Learning Company.

Figura 4. Captura de tela dos jogos O Som do Bichos, O Coelho Sabido e Caça-Pistas, respectivamente.

Fonte: http://softmarket.com.br

Jogos de Esporte, como o próprio nome sugere, são jogos baseados em regras e estratégias de esportes como vôlei e futebol, sendo que jogos de futebol são os populares. A

Figura 5 apresenta dois jogos: FIFA e EA Sports UFC. *FIFA*, é um jogo no qual o jogador controla os jogadores durante uma partida simulando um jogo real.

Figura 5. Captura de tela dos jogos EA Sports UFC e Fifa, respectivamente.

Fonte: http://canaltech.com.br

Na classificação de Jogos Online, pode-se incluir praticamente todos os gêneros de jogos, tendo como diferença o fato de que esses jogos são disputados contra outros jogadores pela Internet. Dentre todos os gêneros jogados de forma online, os que mais se destacam são os jogos de tiro em primeira pessoa, estratégia em tempo real e MMORPG[5]. Essas características compõem os tipos de jogos online mais jogados atualmente. Como por exemplo, a série *Warcraft*, *Starcraft* e *World of Warcraft*[6], tendo este último, mais de 11 milhões de assinantes mensais. A Figura 6 apresenta imagens retiradas dessas séries.

Figura 6. Captura de tela dos jogos *Starcraft* e World of Warcraft, respectivamente.

Fonte: http://us.battle.net

Os Jogos de Estratégia, como dito anteriormente, são jogos que têm como ponto forte o gerenciamento de recursos e tática de ataque. As sagas *Starcraft* e *Warcraft* são exemplos dessa categoria. Esse gênero por ainda ser subdivido em duas categorias: jogos RTS e RTB, e

[5] MMORPG é uma sigla em inglês que significa "Massively Multiplayer Online Role-Playing Game", designando jogos para uma grande quantidade de jogadores de RPGs.
[6] Os jogos dessa série são desenvolvidos pela empresa Blizzard Entertainment

serão apresentados na seção “O gênero de jogos RTS”. A Figura 7 apresenta uma captura de tela do jogo *Warcraft*.

Figura 7. Captura de tela do jogo Warcraft.

Fonte: http://us.battle.net

Em Jogos de Quebra-cabeça, a narrativa principal é proporcionar ao jogador um desafio intelectual, no qual o jogador deverá ficar focado na resolução de problemas. Em geral, os jogos dessa categoria, não são integrados a histórias, como um jogo de aventura, embora existam muitos jogos híbridos, pois alguns jogos de aventura possuem diversos elementos de quebra cabeça durante o decorrer do jogo. Um exemplo dessa categoria é *The Incredible Machine*[7] apresentado na Figura 8.

Figura 8. Captura de tela do jogo The Incredible Machine.

Fonte: http://playdosgamesonline.com

No gênero de Jogos de RPG (do inglês: *Role-Playing*), são representados jogos onde o jogador controla um personagem que realiza o papel de um herói. Em geral esses personagens

[7] O jogo é desenvolvido por Sierra Entertainment.

evoluem, aumentando suas habilidades e obtendo novas armaduras e armas. O combate é um elemento importante nesse gênero, pois permite que os personagens apurem suas técnicas evoluindo e ganhando novas habilidades. Exemplo de jogos RGPs são as séries *Final Fantasy* e *Elder Scroll*[8], apresentados na Figura 9.

Figura 9. Captura de tela dos jogos Final Fantasy e Elder Scroll, respectivamente.

Fonte: http://moviepilot.com e http://blog.us.playstation.com

Os Jogos de Simuladores, o último dos gêneros de jogos apresentado nessa seção, são caracterizados por simularem a operação de máquinas complexas, como aviões ou helicópteros. Alguns jogos de simuladores são utilizados para fins educativos, embora tenham como alvo o público adulto. Um exemplo de simulador apresentado na Figura 10, é jogo *Flight Simulator*[9], cujo objetivo é fazer com que o jogador aprenda a pilotar aviões.

Figura 10. Captura de tela dos jogos Final Fantasy e Elder Scroll, respectivamente.

Fonte: http://microsoft.virtualflight3d.com

Os gêneros de jogos são utilizados para organizar os jogos com base em sua jogabilidade e interação com o jogador. Conforme apresentado nos parágrafos anteriores, um gênero de jogo é caracterizado por um conjunto de desafios de jogabilidade e são

[8] O jogo *Final Fantasy* é desenvolvido por Square Enix e o jogo *Elder Scroll* pela empresa ZeniMax Online Studios.
[9] O jogo é desenvolvido pela empresa Microsoft.

classificados independentemente da sua configuração ou conteúdo, ao contrário de outras obras de ficção, tais como filmes ou livros. A Tabela 1 apresenta um comparativo dessas categorias.

Tabela 1. Os Principais Gêneros de Jogos

Gênero	**Descrição**	**Exemplos**
Jogos de Ação	Categoria dominada por jogos de tiro em primeira pessoa, onde os jogadores são desafiados a reagirem rapidamente aos elementos do jogo. Há também jogos híbridos de ação e aventura, onde a mecânica do jogo é constituída de muita ação e resolução de enigmas.	*Counter-Strike* e *Battlefield,*
Jogos de Aventura	Possuem histórias envolventes, cenários temáticos e geralmente utilizam a resolução de enigmas ou quebra-cabeças como mecanismo para que o jogador possa prosseguir no curso do jogo.	*Assassin's Creed, God of War e The Incredible Machine*
Jogos Casuais	Jogos com regras bastante simplificadas, usado na maioria das vezes para passar o tempo. Jogos de televisão também são incluídos nessa categoria.	Xadrez e Paciência
Jogos Educacionais	Têm por objetivo fazer com que o jogador aprenda enquanto se diverte. Esses são desenvolvidos em conjunto com uma equipe de especialistas em educação para que o jogo sirva como uma ferramenta de apoio à aprendizagem.	O Som do Bichos, O Coelho Sabido e Caça-Pistas
Jogos de Esporte	Compostos por jogos onde as regras e estratégias de esportes são rigorosamente reproduzidas.	*EA Sports UFC* e *FIFA*
Jogos Online	Inclui praticamente por todos os gêneros de jogos, tendo como diferença o fato de que esses jogos são disputados contra outros jogadores pela Internet	*Warcraft, Starcraft* e *World of Warcraft*
Jogos de Estratégia	Jogos que solicitam que o jogador gerencie um conjunto limitado de recursos para que possam alcançar um objetivo predeterminado. Este gerenciamento de recursos frequentemente envolve o uso de estratégias para tomada de decisões por parte dos jogadores.	*Starcraft* e *Warcraft*
Jogos de Quebra-cabeça	Jogos quem têm a finalidade de proporcionar ao jogador um desafio intelectual, nos quais o jogador deverá ficar focado na resolução de problemas.	*The Incredible Machine*
Jogos de RPG	Jogos nos quais o jogador controla um personagem que desempenha o papel de herói, permitindo que esses personagens evoluam, aumentem suas habilidades, armaduras e armas.	*Final Fantasy* e *Elder Scroll*

Jogos de Simuladores	Jogos que simulam a operação de máquinas complexas, como aviões ou helicópteros.	*Flight Simulator*

Alguns jogos podem ser classificados em mais de um gênero, como pode ser observado no jogo *The Incredible Machine* que está incluso tanto no gênero de jogos de aventura, como também no gênero de jogos de quebra-cabeça.

O jogo *Starcraft*, que faz parte deste trabalho, está contido no gênero RTS, conforme apresentado na Tabela 1, por isso, esse gênero será descrito com maiores detalhes na próxima seção.

2.1.1 O gênero de Jogos RTS

O gênero de jogos RTS é representado por jogos que permitem aos jogadores se enfrentarem em um ambiente comumente conhecido por "mapa" (um mapa é definido por uma combinação de terreno e recursos). Uma que vez o jogo começa**,** os jogadores devem, simultaneamente e de forma contínua, adquirir recursos e construir novas unidades, a fim de destruir seus oponentes (GEMINE, et al 2012).

Em jogos RTS, o jogador tem controle total sobre suas unidades e ele é responsável por toda estratégia do jogo. Dependendo das tecnologias que escolhem, os jogadores ganham acesso a diferentes tipos de unidade, cada uma com atributos e habilidades específicas.

Segundo MICIC et al. (2011), os jogadores são constantemente confrontados com uma multidão de decisões a tomar. Eles devem gerenciar sua economia, a produção de recursos, o reconhecimento do mapa e o combate contra seus oponentes, tudo ao mesmo tempo. Os jogadores devem decidir se a renda atual é suficiente ou nova fonte de recursos deve ser reivindicada.

Seguem abaixo algumas das principais características e elementos dos jogos RTS, bem como detalhes sobre cada um deles:

- Representação do mundo.
- Recursos.
- Unidades.
- Névoa de guerra.
- Edifícios.
- Arvore tecnológica.
- Gerenciamento.

Na Representação do mundo – geralmente ilustrada por mapas 2D de tamanhos variados – sobre esse mapa, estão dispostos os recursos, oponentes, unidades de controle e também é o local onde as guerras ocorrem.

Em geral os jogos RTS possuem diversos recursos, o que proporciona aos jogadores uma variedade de estratégias. Esses recursos podem ser aumentados ou diminuídos, dependendo da ação do jogador. Geralmente o ganho de recursos é obtido através de alguma forma extração de recursos, que envolvem o emprego de mineradores em fontes disponíveis no mapa, e o gasto desses recursos ocorre de diversas formas, sendo a mais comum a produção de novas unidades, construção de novos edifícios e pesquisa de novas tecnologias. A Figura 11 apresenta como esses recursos podem ser obtidos.

Figura 11. Depósito de materiais do jogo *Starcraft*.

(a) Deposito de materiais (b) Deposito de materiais ricos em minérios

Fonte: http://us.battle.net/

Os depósitos de minerais são os principais meios para se obter recursos em um jogo. A partir da Figura 11 é possível observar dois tipos de depósitos de materiais no jogo *Starcraft*, cada um com finalidades diferentes. Para colher esses minerais, as unidades precisam viajar para a fonte de mineral, passar algum tempo minerando e só então retornar à base com a carga. Os mineradores devem continuamente repetir este processo afim de fornecer um fluxo constante de minerais, ou seja, garantir a reposição de recursos ao jogador. Quando a fonte de minerais estiver esgotada, uma nova mina deve ser pesquisada.

As unidades existentes nesses jogos são entidades móveis controladas pelo jogador. Elas são produzidas em determinados edifícios do jogo. São caracterizadas por atributos específicos, como, por exemplo: custo de recursos para sua produção, pontos de vida e/ou ataque, tempo requerido para sua produção e a velocidade que a unidade tem durante a movimentação no mapa. Em geral, as unidades podem ser divididas em dois grupos: trabalho e combate. As unidades de trabalhos são responsáveis pela captação de recursos e construção de edifícios. Contudo, elas possuem poucas habilidades de batalha. As unidades de batalha,

em oposição às outras unidades, possuem alta habilidade para batalhas, com especialização em ataque ou defesa, ou ainda no reconhecimento do mapa do jogo.

Os edifícios são unidades controladas pelo jogador. Analogamente às unidades, os edifícios possuem características particulares, como tempo de construção, custo de produção e outras. Eles podem ser divididos em quatro grupos: suporte, defesa, militar e pesquisa. Os de suporte são utilizados para o crescimento militar e econômico do jogo. Os de defesa e militares são empregados, respectivamente, na defesa territorial de inimigos e na produção de unidades de combate. Os edifícios de pesquisa, por fim, são destinados à pesquisa de novas tecnologias.

O conceito designado para os pré-requisitos necessários ao jogador para que este tenha acesso a uma determinada tecnologia é conhecido como árvore tecnológica. Um exemplo de tecnologias obtidas por esse recurso é a obtenção de novas unidades e edifícios ou ainda quaisquer melhorias neles. Outro exemplo comum de melhorias em unidades é o aumento dos pontos de ataque ou de defesa, ou ainda o aumento da velocidade na produção de unidades e a desbloqueio de habilidades especiais.

Outro conceito importante utilizado na maioria dos jogos RTS, é névoa da guerra (no inglês, *fog of war*), que pode ser entendido como a falta de conhecimento durante uma guerra, ou seja, a incerteza de cada lado sobre as capacidades e planos do inimigo (MICIC et al., 2011).

Durante o jogo, esse recurso é apresentado na forma de um nevoeiro que cobre as partes do mapa ainda não explorados, escondendo-os da visão dos jogadores humanos, o que faz com que os jogadores só tenham informações sobre as áreas que foram percorridas. A Figura 12 apresenta esse efeito.

Figura 12. Névoa de Guerra do jogo *Starcraft*.

(a) Grande área sob névoa (b) Grande área descoberta

Fonte: https://www.youtube.com/watch?v=vc4ur_V9zHQ

A Figura 12 apresenta duas imagens de uma mesma área do mapa do jogo com diferente influência da névoa de guerra. Como é possível observar, a névoa de guerra obedece um “raio de visão”, assim, a área visível vai sendo ampliada conforme ocorre o movimento de uma unidade na tela.

A última característica do gênero de jogos RTS apresentada nessa seção é o gerenciamento. Em jogos RTS, as ações dos jogadores estão associadas basicamente a dois tipos de gerenciamento: microgerenciamento (ações relacionadas ao controle de suas unidades) e macrogerenciamento (ações relacionadas ao gasto de recursos). Os jogadores devem desempenhar bem ambas tarefas de gerenciamento, para que eles sejam bem-sucedidos.

2.1.2 *Starcraft*

Starcraft é um jogo RTS de ficção científica lançado pela *Blizzard Entertainment* em março de 1998. Em novembro do mesmo ano, a empresa lançou um pacote de extensão para o jogo, chamado *Starcraft: Brood War*. A série *Starcraft* foi um sucesso comercial, se tornando o jogo mais vendido do ano, vendendo mais de 1,5 milhões de cópias em todo o mundo (SYNNAEVE et. al, 2011).

Durante a última década, a indústria de jogos eletrônicos tem visto o surgimento da *e-sport* (profissionalização de jogos competitivos). Em alguns países, esse mercado em ascensão possui estruturas comparáveis aos jogos esportivos: espectadores, ligas, patrocinadores, fãs e transmissões. A lista de jogos eletrônicos inclui: *Starcraft: Brood War*, *Counter-Strike*, *Quake III*, *Warcraft III*, *Auréola*, *Starcraft II* (SYNMAEVE, 2012).

Starcraft, foi o primeiro jogo a ter jogadores profissionais. Na Coréia do Sul, por exemplo, os melhores jogadores ganham mais do que os jogadores de futebol de topo. Os principais jogadores ganham mais de US$ 400.000,00 (quatrocentos mil dólares americanos) por ano, mas a média profissional é baixa, cerca de US $ 50-60.000 por ano (LIQUIPEDIA, 2016).

Segundo uma consultoria de Nova York, SuperData (2016), os chamados *e-sports* ainda estão longe de movimentar a economia do esporte como o basquete ou o futebol, mas a estimativa é que o mercado global do setor ultrapasse os US$ 748 milhões (R$ 2,9 bilhões) atuais para US$ 1,9 bilhão (R$ 7,5 bilhões) em 2018.

O jogo *Starcraft* possui uma temática espacial, onde o jogador representa um comandante de uma das três raças da galáxia definidas no jogo. A Figura 13 apresenta duas imagens do jogo. Na imagem à esquerda é possível observar uma guerra ocorrendo entre

unidades, e na outra imagem é apresentado o procedimento para se realizar a coleta de recursos no jogo.

Figura 13. Imagens do jogo *Starcraft*.

Fonte: http://us.battle.net/

Em *Starcraft*, como na maioria dos jogos RTS, o jogador tem de gerir um exército através da construção de uma base militar, coletar de recursos (minerais e gás), criar e controlar unidades para destruir todos os edifícios do adversário. Quando os recursos se tornam disponíveis, os jogadores precisam alocá-los para a criação de mais edifícios (que reforçam a economia, isso permite que os jogadores criem unidades ou desbloqueiem as unidades mais fortes), ou ainda pesquisar novas tecnologias, a fim de utilizar as novas habilidades nas unidades ou aprimorar as existentes.

As unidades devem ser distribuídas para que realizem tarefas diferentes, como reconhecimento, defesa e ataque. Durante a execução das tarefas, os jogadores também podem conhecer a geometria do mapa, a fim de decidir onde colocar novos edifícios (concentrar-se em uma única área, ou expandir para áreas diferentes) ou onde definir postos defensivos. Finalmente, quando as unidades ofensivas de dois jogadores se encontram, uma batalha é iniciada, e cada jogador deve rapidamente manobrar suas unidades para lutarem, e isso requer um controle rápido e reativo de cada uma das unidades. A Figura 14 ilustra três unidades do jogo *Starcraft*.

Figura 14. Unidades do jogo *Starcraft*.

(a) Colossus (b) SCV (c) Nydus Worm

Fonte: http://us.battle.net/

A Figura 14 apresenta três unidades do jogo *Starcraft*: o Colossus, uma unidade blindada que pode atacar alvos terrestres; a unidade SCV, usada para construir edifícios, reunir recursos e realizar reparos; a unidade Nydus Worm, uma estrutura que permite transportar outras unidades pelo subterrâneo.

No jogo *Starcraft* o jogador pode escolher entre vários modos de jogo, os quais podem ser divididos em dois tipos: individual e em grupo. As Figuras 15 e 16 ilustram esses modos com suas respectivas opções.

Figura 15. Modo individual de jogo do *Starcraft*.

Fonte: http://us.battle.net/

Com base na Figura 15 é possível observar o modo individual de jogo, no qual o jogador tem duas possibilidades: lutar contra a IA do jogo *Startcraft* ou jogar desafios, esta última tem como objetivo testar as habilidades do jogador.

Figura 16. Modo em grupo de jogo do *Starcraft*.

Fonte: http://us.battle.net/

No modo de jogo em grupo, ilustrado na Figura 16, o jogador tem uma variedade de opções, sendo: 1×1 (batalha individual entre dois jogadores), 2×2 (batalha de quatro jogadores divido em duas equipes), 3×3 (seis jogadores em duas equipes), 4×4 (oito jogadores em duas equipes), todos contra todos (modalidade onde 4 jogadores que se enfrentam, cada um por si), além de jogos personalizados e cooperativos.

Em todos os modos, exceto no desafio, o jogador deve escolher entre três raças: os *Terrans*, humanos prisioneiros lutando pela sobrevivência; os *Protoss*, alienígenas desenvolvidos psiquicamente; e os *Zergs*, uma espécie super-humanos desenvolvidos biologicamente que têm por objetivo aniquilar todas as outras raças. A Figura 17 apresenta os tipos de raças do jogo.

Figura 17. Tipos de raças do jogo *Startcraft*

Fonte: http://us.blizzard.com

Um dos aspectos importantes do jogo é que os poderes dessas raças são equilibrados:

- Os *Terrans* fornecem unidades que são versáteis e flexíveis, dando uma opção equilibrada para lutar contra *Protoss* e *Zergs*.

- As unidades *Protoss* têm processos de fabricação demorados e dispendiosos, mas eles são fortes e resistentes. Jogadores que escolhem essas condições estão focados na estratégia de qualidade sobre a quantidade.
- As unidades *Zergs* possuem unidades com baixo custo para fabricação. Elas podem ser produzidas rapidamente, encorajando os jogadores a dominar os adversários com números absolutos.

A escolha do jogo *Starcraft* deu-se principalmente pelo fato de ser um dos jogos de RTS mais completos e com maior número de restrições e pré-condições entre seus recursos, o que valida ainda mais os resultados obtidos no decorrer do trabalho.

Outro aspecto que influenciou na escolha do jogo é a possiblidade, através do modo de jogo individual de luta contra IA e em conjunto com a biblioteca BWAPI, introduzir algoritmos de IA diretamente dentro do jogo, o que permite gerenciar e controlar o ambiente da mesma forma que um jogador.

A escolha da biblioteca BWAPI ocorre naturalmente por sua integração com o *Starcraft*, pois essa biblioteca foi desenvolvida especificamente para esse objetivo.

A próxima seção apresenta uma visão geral de como IA é empregada em jogos eletrônicos e também qual é sua importância para a evolução dos jogos eletrônicos.

2.2 IA EM JOGOS

Segundo Russel et. al. (2009), a IA é uma importante disciplina da Ciência da Computação que, através de aplicação técnicas e dispositivos computacionais, busca simular a inteligência humana.

Segundo Millington (2006), existe uma diferença entre o estudo das técnicas de IA por pesquisadores acadêmicos e as técnicas aplicadas no desenvolvimento de jogos digitais. Esta distinção sugere que a IA acadêmica representa uma área de pesquisa mais abrangente, por exemplo. As técnicas para o desenvolvimento de sistemas inteligentes não são de interesse, não podem ser diretamente aplicadas ou não correspondem exatamente às técnicas abordadas pela IA para Jogos na indústria de jogos.

O foco do desenvolvimento de IA para jogos na indústria é voltado à diversão dos jogadores. Sua importância é equivalente aos resultados, e não ao esforço necessário para se chegar a esse resultado, ou seja, o problema não é como o sistema pensa, mas sim como ele age. Jogos eletrônicos são negócios, e seus os consumidores os compram em busca de diversão, não interessando o desenvolvimento da inteligência de um personagem, desde que

ela torne o jogo divertido e desafiador, além de, claro, tomar decisões coerentes com o contexto do jogo (TOZOUR, 2002).

2.2.1 Breve Histórico da IA em Jogos eletrônicos

Inicialmente as técnicas de IA foram utilizadas nas pesquisas em busca em espaço de estados[10] em jogos como xadrez e damas. A vantagem do uso desses tipos de jogos no estudo da IA reside no fato desses jogos possuírem regras bem definidas e serem fáceis de se jogar.

No princípio do desenvolvimento de jogos eletrônicos, a aplicação de técnicas de IA, segundo Schwab (2004), era mais usualmente conhecida por "programação de jogabilidade", pois não havia nada de inteligente sobre os comportamentos exibidos pelos NPCs. A Tabela 1 contém alguns exemplos de como a IA foi utilizada em jogos com o passar do tempo.

Tabela 2. Linha de tempo da IA em jogos (SCHWAB, 2004).

Ano	Descrição	IA utilizada
1962	Primeiro jogo de computador, *Spacewar*, para 2 jogadores.	Nenhuma
1972	Lançamento do jogo *Pong*, para 2 jogadores.	Nenhuma
1974	Jogadores tinham que atirar em alvos móveis em *Pursuit* e *Qwak*.	Padrões de movimento
1975	*Gun Fight* lançado, personagens com movimentos aleatórios.	Padrões de movimento
1978	*Space Invaders* contém inimigos com movimentos padronizados, mas também atiram contra o jogador.	Padrões de movimento
1980	O jogo *Pac-man* conta com movimentos padronizados dos inimigos, porém cada fantasma (inimigo) tem uma "personalidade" sobre o modo em que caça o jogador	Padrões de movimento
1990	O jogo de estratégia em tempo real, *Herzog Wei*, é lançado. Junto, os jogadores puderam noticiar uma péssima busca de caminho.	Máquina de estados
1993	*Doom* é lançado como primeiro jogo de tiro em primeira pessoa.	Máquina de estados
1996	*BattleCruiser*: *3000AD* é publicado como o primeiro jogo a utilizar redes neurais em um jogo comercial.	RNA
1998	*Half-Life* é lançado e analisado como a melhor IA em jogos até a época, porém, o jogo utiliza IA baseada em scripts.	Máquina de estados / Script
2001	O jogo *Black* & *White* é alvo da mídia a respeito de como as criaturas do jogo aprendem com as decisões feitas pelo jogador. Utiliza redes neurais, *reinforcement* e *observational learning*.	Diversos

Fonte: (SCHWAB, 2004)

A falta de memória e a limitação existente na velocidade de processamento obrigava os programadores de jogos eletrônicos a implementar padrões de movimentos ou movimentos repetitivos e/ou aleatórios para os personagens controlados pelo computador.

[10] O uso de uma máquina de estados finitos em jogos eletrônicos tem o objetivo de definir para cada estado, um conjunto de ações a serem tomadas. Estas ações podem ser implementadas diretamente na linguagem do motor do jogo, ou na forma de uma linguagem de script.

O aumento da capacidade computacional permitiu que a indústria de jogos aplicasse mais realismo aos jogos. E com o advento dos gráficos em 3D, a IA ganhou ainda mais espaço. Os ambientes de jogos cada vez mais complexos obrigou os desenvolvedores a utilizarem algoritmos mais elaborados para evitar que os NPCs atravessassem paredes, ficassem presos em cantos do cenário ou agissem de maneira indiferente a estímulos visuais e sonoros produzidos pelo jogador (SCHWAB 2004).

A próxima seção descreve os conceitos principais relacionados a NPC, bem como, a forma com que esse tipo de agente é empregada no trabalho.

2.2.1 Controle de NPCs

Segundo Synnaeve (2012), os NPCs, também chamados de "mobs", representam os jogadores que não são concebidos para serem jogados por seres humanos, porém o seu comportamento é controlado por alguma forma de IA. Eles existem nos jogos *multiplayer* (vários jogadores), e *singleplayer* (um só jogador), sendo que é no estilo *singleplayer* que ganham mais destaque, pois representam o adversário (oponente) do jogador (BORGES, et. al. 2009).

Os NPCs podem classificados em dois tipos: ajudantes ou oponentes. Os ajudantes são os agentes que ajudam o jogador (ou ajudam o personagem do jogador). Em oposição, os oponentes são aqueles que atrapalham o jogador (NETO 2011).

Por exemplo, em um jogo de futebol, no modo *singleplayer*, o jogador controla os seus jogadores, mas em determinada situações, não controla o seu goleiro. Seu goleiro decide sozinho (seu comportamento é controlado por uma IA) para qual lado irá pular, para defender a bola. Portanto, o goleiro é um tipo de NPC ajudante. Em contrapartida, todos os jogadores do time adversário são NPCs adversários (oponentes).

Para que um jogo obtenha um tom mais realístico em relação ao comportamento dos personagens, os NPCs devem adquirir um certo nível de “inteligência”. No início do desenvolvimento dos jogos essa “inteligência” era implementada através de regras por meio da estrutura *if-then-else.* No caso de jogos em que o NPC recebe um lote de condições e ainda precisa avaliar os casos especiais, esse procedimento além de tornar mais complexo a mudança de algum comportamento do personagem, gerava uma precariedade em termos de comportamentos mais complexos (BORGES, et. al. 2009).

Para essa dificuldade ser superada, o NPC necessita ser autônomo e inteligente, isso significa que ele deve perceber o ambiente que o cerca e conseguir agir de forma autônoma a fim de atingir seus objetivos, além disso ele deve aprender através de suas ações.

Segundo Borges, et. al. (2009), o ambiente é um fator importante nesse tipo de NPC. A aplicação de técnicas de IA como RNA são bastante importantes para o uso desses agentes, pois eles têm um conhecimento limitado no início do jogo, e a partir do desenrolar do jogo e das operações (ações) feitas pelo jogador, vão adquirindo conhecimento. Assim, pode-se considerar o agente autônomo como um agente que busca suas decisões baseado na experiência, ou seja, o que ele já passou e o ambiente que ele percorreu o influenciou.

Segundo Jennings (1996), para que um NPC possa agir de forma autônoma, ele deve ser capaz de perceber o ambiente, interpretar mensagens, ter raciocínio com base em crenças, planejamento a curto, médio e longo prazo, além de outras habilidades como, tomada de decisão e transmissão de mensagens. Ele categoriza os agentes em função da capacidade de resolução de problemas:

- **Reativos** – são agentes que reagem às mensagens de outros agentes e também às alterações do ambiente. Contudo, eles não são capazes de raciocinar sobre suas intenções.
- **Proativos**: em oposição aos reativos, os proativos têm a habilidade de raciocínio sobre suas intenções, pois conseguem criar e executar planos de ações.
- **Sociais**: agentes proativos são considerados sociais quando estes são capazes de utilizar modelos de outros agentes, sobre os quais raciocina, toma decisões e cria planos.
- **Híbridos**: possuem capacidade tanto de agentes reativos como dos agentes proativos. Esses agentes têm a capacidade de reação rápida e adequada a situações não previstas, além de buscar alternativas para seu comportamento quando a situação do ambiente é diferente dos seus objetivos.

Neste trabalho o NPC empregado, foi baseado em um agente reativo, pois, conforme a apresentação do ambiente ou da proximidade do opoente, o agente deverá percebê-los e agir conforme as ações definidas pela saída da RNA. Contudo, o NPC não realizará estratégias de combate, somente tomadas de decisão baseadas na saída da rede.

Um NPC pode agir de forma simples, sem inteligência, realizando tarefas previsíveis e repetitivas. Entretanto, o interessante é o desafio dos jogadores contra NPCs inteligentes (GALDINO, 2007).

A próxima seção descreve os principais conceitos de Redes Neurais Artificiais, como: modelo matemático de um neurônio, funções de ativação e algoritmo de aprendizagem da rede.

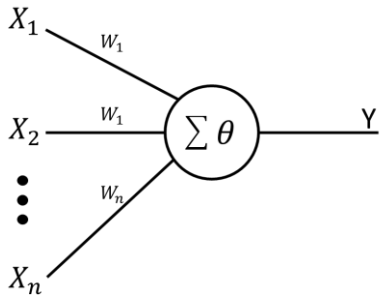
2.3 REDES NEURAIS

Esta seção apresenta os principais conceitos de RNAs como modelos computacionais inspirados no sistema nervoso central e sua capacidade de realizar o aprendizado de máquina, bem como o reconhecimento de padrões. Além disso, será também apresentado o neurônio artificial, funções de ativação dos neurônios e um algoritmo de aprendizagem.

2.3.1 Modelo Matemático do Neurônio

Segundo Valença (2007) o modelo matemático de um neurônio, criado por McCulloch e Pitts em 1943, é bastante simples (em relação ao neurônio biológico) e procura representar a estrutura e funcionalidades de um neurônio biológico através de um conjunto de entradas, uma regra de propagação e uma ou mais saídas. Na Figura 18, pode-se observar a representação neural proposta.

Figura 18. Modelo matemático do neurônio proposto por McCulloch e Pitts



Fonte: Haykin (2001)

A Figura 18 apresenta um modelo matemático de um neurônio (representado por X), que demonstra a transformação dos estímulos (entrada) em uma informação (saída) onde:

- Os sinais de entrada são representados pelo vetor $X = \{x_1, x_2, \dots, x_n\}$.
- Os pesos sinápticos são representados pelo vetor $\mathrm{W} = \{w_1, w_2, \dots, w_n\}$.
- A saída é representada pelo neurônio Y.

A combinação de diversos neurônios artificiais forma uma rede neural artificial. As entradas da RNA simulam uma área de captação de estímulos, que por sua vez, podem ser conectados em muitos outros neurônios, resultando assim, em uma série de saídas, onde cada neurônio representa uma saída (KARLSSON, 2005).

A próxima seção apresenta o conceito de uma RNA, além da inspiração para o seu desenvolvimento, objetivos e aplicações.

2.3.2 Redes Neurais Artificiais

Segundo Haykin (2001), de uma forma geral, uma RNA pode ser entendida como um processador, maciçamente e paralelamente distribuído, constituído de unidades elementares de processamento, onde tem a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso.

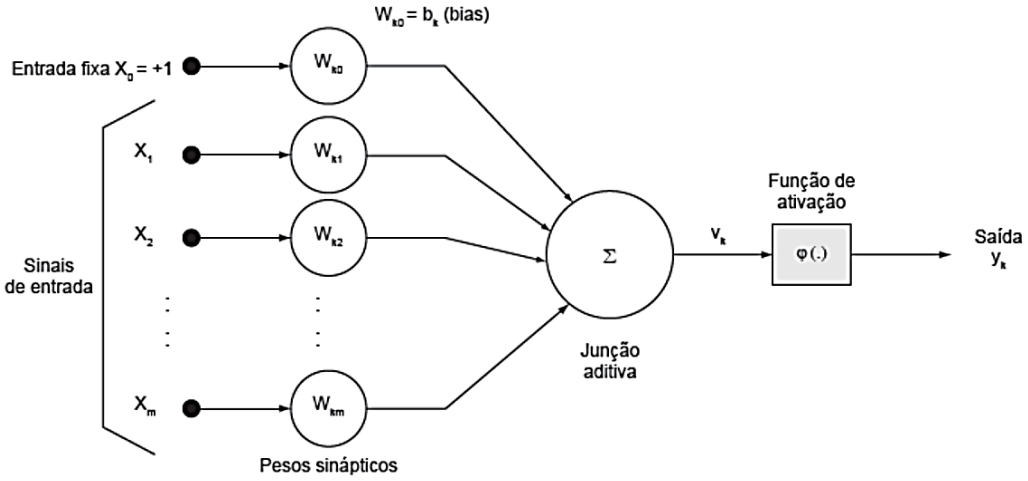
As RNAs foram inspiradas no funcionamento do cérebro humano e sua motivação tem sido pela capacidade do cérebro em processar informações de forma completamente diferente dos computadores digitais convencionais. Segundo Haykin (2001), o cérebro é um "computador" capaz de processar informações altamente complexas, não-lineares e em paralelo, e tem a capacidade de organizar seus elementos estruturais, conhecidos por neurônios, de forma a realizar vários processamentos, como: reconhecimento de padrões, percepção e controle motor, muito mais rapidamente que os computadores atuais.

Uma RNA tem por objetivo imitar de forma simplificada e especifica o funcionamento do cérebro humano, onde o conhecimento é adquirido a partir do seu ambiente através de um processo de aprendizagem. Para atingir esse objetivo, a RNA utiliza pequenas unidades de processamento, identificadas como neurônios artificiais (essas unidades fazem alusão ao processo de aprendizagem do cérebro humano), onde emulam, por meio de modelos matemáticos, o processamento de informações dos neurônios biológicos através da aprendizagem e aproximação. As conexões existentes desses neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido (HAYKIN, 2001).

Segundo Barreto (2002), atualmente as aplicações das RNA ocorrem em várias áreas, como por exemplo no reconhecimento de padrões, na distribuição de energia elétrica, no mercado de capitais, em aplicações navais, em sistemas especialistas e outros.

A Figura 19 ilustra a estrutura de um neurônio artificial que simula o neurônio biológico e suas características básicas, como adaptação e a representação de conhecimentos baseada em conexões.

Figura 19. Estrutura de um neurônio artificial



Fonte: Haykin (2001)

Com base na Figura 19 é possível observar as ligações entre os neurônios artificiais através das setas direcionais. A função dessas conexões é tornar o sinal de saída de um neurônio em um sinal de entrada de outro, ou ainda, orientar o sinal de saída de um neurônio. No modelo da Figura 3, "os sinais de entradas", representados pelas variáveis $\{x_0,\ x_1,\ x_2\ e\ x_n\}$, são associados a "pesos sinápticos", representados pelas variáveis $\{w_0,\ w_1,\ w_2\ e\ w_n\}$, e então combinadas usando uma "junção aditiva" para produzir um estado de ativação no neurônio de saída.

Um neurônio de uma RNA pode ser descrito em termos matemáticos pelas equações a seguir:

$$u = \sum_{j=1}^{m} w_j \times x_j \tag{1}$$

$$y = \varphi\ (u + b) \tag{2}$$

Onde:

- $x_1, x_2, \dots, x_m$ são os sinais de entrada;
- $w_{k1}, w_{k2}, \dots, w_{km}$ são os pesos sinápticos do neurônio k;
- u_k é a saída do combinador linear devida aos sinais de entrada;
- b_k é a polarização, ou bias;
- $\varphi()$ é a função de ativação; e
- y_k é o sinal de saída do neurônio.

As equações 1 e 2 podem ser reescritas na forma vetorial, usando produto escalar:

$$y = \varphi(\mathbf{W} \cdot \mathbf{X} + b) \tag{3}$$

Onde o produto escalar

$$\mathbf{W} \cdot \mathbf{X} = \sum_{j=1}^{m} w_j \times x_j. \tag{4}$$

O modelo matemático de um neurônio inclui ainda uma polarização externa (bias), denotada por b. O bias tem o efeito de aumentar ou diminuir o argumento da função de ativação, caso seja positivo ou negativo, respectivamente.

A função de ativação ($\varphi(.)$) limita a faixa de amplitude permitida do sinal de saída a algum valor finito. Geralmente, a faixa de amplitude normalizada da saída de um neurônio é restrita a intervalos unitários.

Na próxima seção serão apresentados com maiores detalhes os conceitos das funções de ativação, bem como as funções mais comumente utilizadas.

2.3.3 Funções de Ativação

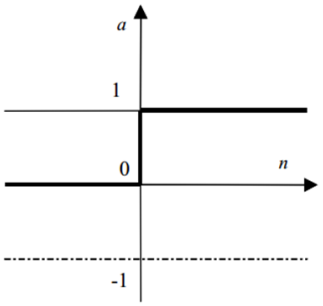
As funções de ativação processam um conjunto de entradas recebidas e os transforma em estado de ativação. Os estados de ativação que os neurônios podem assumir, são: valores binários {0 e 1}, valores bipolares {-1 e 1} ou valores reais. Segundo Haykin (2001), as funções de ativação são escolhidas em função da necessidade do problema. Esta seção descreve algumas das funções de ativação mais utilizadas com algumas tabelas de exemplos.

2.3.3.1 Função de Limiar

A primeira rede com neurônios artificiais utilizava uma função de ativação de limiar. Neste modelo, as unidades são binárias, ou seja, suas saídas assumem valores de 0 e 1. Esta unidade responde com uma saída igual a 1, se o valor da soma ponderada for superior a um determinado valor. Caso contrário, a saída da unidade assume o valor zero. A Figura 20 ilustra essa função.

Figura 20. Representação da função limiar

(a) Gráfico da função



(b) Sistema da função

$$a = \begin{cases} 1, & se\ n \geq b \\ 0, & se\ n < b \end{cases}$$

Fonte: Souza (2007).

Tabela 3. Exemplos da função de ativação de limiar

Valor entrada (a)	Sistema	Saídas
4	$a = \begin{cases} 1, & se\ n \geq b \\ 0, & se\ n < b \end{cases}$ Onde, b = 0.	1
3		1
-2		0
-5		0

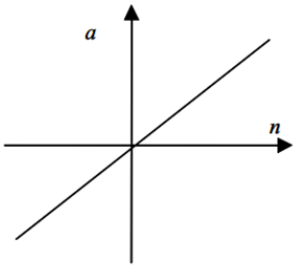
A função de ativação de limiar é também conhecida como função degrau, que, por sua vez, possui uma variação chamada “função de ativação degrau bipolar”, onde, as diferenças entre essas funções residem nas saídas possíveis, no caso 1 e -1 para a degrau bipolar.

2.3.3.2 Função Linear

A função linear, segundo Coppin (2010), é uma das funções de ativação mais frequentes encontradas em RNAs. De forma simplificada, ela é obtida pela soma ponderada das entradas da rede. Normalmente utilizada na primeira camada da rede, trabalha em conjunto com outras funções como a sigmoidal, pois apresenta uma saída com valores reais. Esta função pode ser observada na Figura 21.

Figura 21. Representação da função linear

(a) Gráfico da função (b) Sistema da função



$$a = n$$

Fonte: Souza (2007).

Tabela 4. Exemplos da função de ativação de linear

Valor entrada (a)	Sistema	Saídas
4	$a = n$	4
3		3
-2		-2
-5		-5

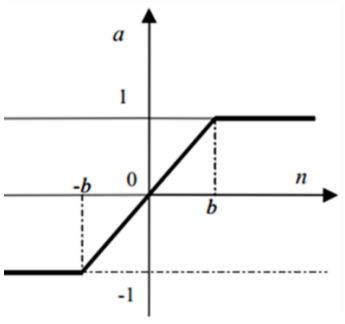
A função linear representada pela Figura 21 não limita a saída da rede e é usada apenas para armazenar entrada e saída de dados. Os neurônios que possuem esta função atuam como aproximadores lineares.

2.3.3.3 Função Linear por Partes

De forma similar a função de ativação linear, a linear por partes permite delimitar a saída dos dados, conforme pode ser observado na Figura 22.

Figura 22. Representação da função linear por partes

(a) Gráfico da função (b) Sistema da função



$$a = \begin{cases} -1, & se\ n \leq b \\ n, & se\ -b < n < b \\ 1, & se\ n \geq b \end{cases}$$

Fonte: Souza (2007).

Tabela 5. Exemplos da função de ativação de linear por partes

Valor entrada (a)	Sistema	Saídas
2,0	$a = \begin{cases} -1, & se\ n \leq b \\ n, & se\ -b < n < b \\ 1, & se\ n \geq b \end{cases}$ Onde, b = 1.	1,0
1,5		1,0
-0,4		-0,4
-1,3		-1,0

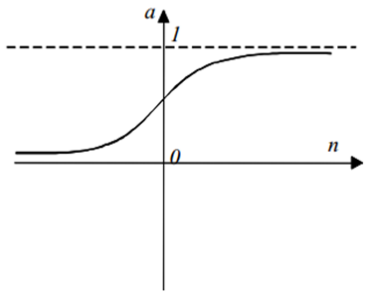
A função de ativação linear por partes é também conhecida por "função de ativação tipo rampa". Aqui essas duas funções retornam valores que variam linearmente entre dois pontos de saturação simetricamente dispostos em torno da origem; fora dessa faixa, produz valores constantes.

2.3.3.4 Função Sigmoidal

Também conhecida como "função logística", a função de ativação sigmoidal é uma versão contínua da função rampa. A função sigmoide é uma colocação que possui um domínio finito e produz respostas não-lineares dentro de uma faixa predefinida. É dada pela equação abaixo, onde determina a suavidade (inclinação) da curva. A Figura 23 ilustra graficamente a função sigmoide.

Figura 23. Representação da função linear

(a) Gráfico da função



(b) Equação

$$f(x)=\frac{1}{1+e^{-\alpha x}}$$

Fonte: Souza (2007).

Tabela 6. Exemplos da função de ativação de sigmoide

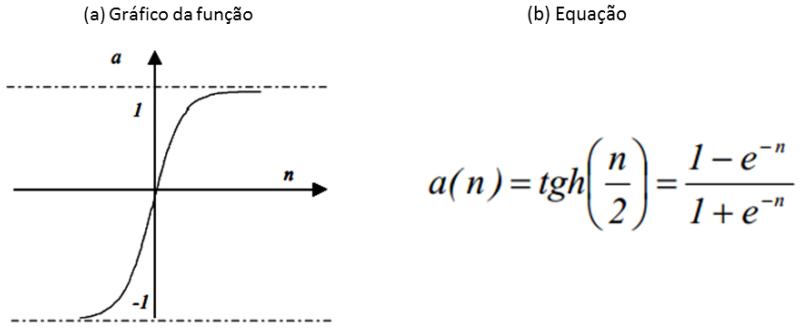
Valor entrada (x)	Equação	Saídas
2	$f(x)=\frac{1}{1+e^{-\alpha x}}$ Onde, $a = 1$.	0,89
0,5		0,62
0		0,50
-1,5		0,38

Para a função sigmoide ilustrada na Figura 23, o parâmetro *a* ilustrado em *(b) equação* é a variável que define a inclinação (ganho) da função sigmoide. Nesse tipo de função, a saída do neurônio assumirá valores reais entre 0 e 1, como apresentado na Tabela 6.

2.3.3.5 Função Tangente Hiperbólica

A função tangente hiperbólica é uma função derivada da função sigmoide, onde os pontos extremos de saídas da função estão 1 e -1. A Figura 24 ilustra graficamente a função tangente hiperbólica.

Figura 24. Representação da função tangente hiperbólica



Fonte: Souza (2007).

Tabela 7. Exemplos da função de ativação de tangente hiperbólica

Valor entrada (n)	Equação	Saídas
2,0	$a(n) = tgh\left(\frac{n}{2}\right) = \frac{1-e^{-n}}{1+e^{-n}}$	0,76
1,5		0,66
0,5		0,25
0,0		0,00
-1,5		-0,64

Segundo Souza (2007), uma arquitetura de rede neural multicamadas *feedforward*, apresentada na seção 2.3.5, consiste de uma camada de entrada, que é assumida ter neurônios com função de ativação linear (funções baseadas em saídas binárias), e uma camada de saída ou mais camadas intermediárias de neurônios, que não são nem entrada e nem de saída. Os neurônios das camadas intermediárias são assumidos serem não lineares. A não linearidade normalmente inclui: função sigmoide, função tangente hiperbólica, função de base radial, entre outras.

A próxima seção apresenta a estrutura de uma rede *Perceptron*. Esta seção é primordial para o entendimento da seção 2.3.5, que aborda sobre redes neurais de multicamadas, *Multilayer Perceptron*.

2.3.4 Rede Perpeptron

Segundo Valença (2007), a Rede Perceptron é o modelo simplificado de uma rede neural constituído de um neurônio não linear, tendo as seguintes características básicas: topologia constituída de apenas uma camada com um ou mais neurônios; valores de entrada e saída em binários; função de ativação degrau; algoritmo de aprendizado supervisionado e regra de propagação definida como:

$$NET_J = \sum x_{ij} \,.\, w_{ij} + b_j \tag{5}$$

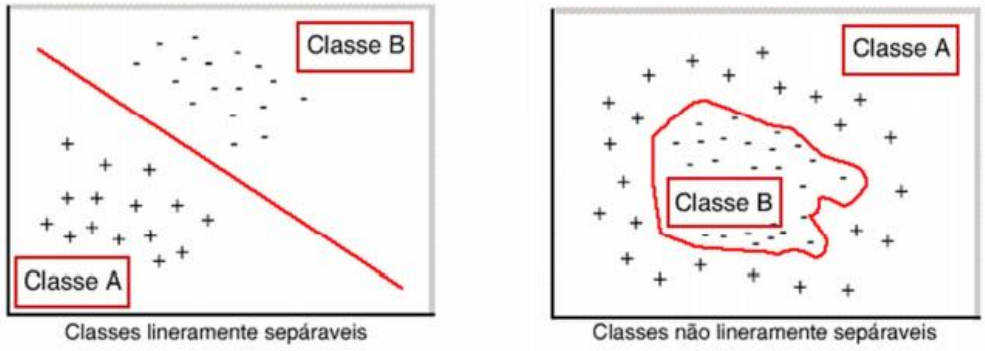
A capacidade de aprender do Neurônio Perceptron é adquirida através dos exemplos apresentados à rede Perceptron. Os pesos sinápticos são reajustados através de uma regra de aprendizagem.

Segundo Valença (2007), a regra de aprendizagem de Hebb diz que:

> Se um neurônio recebe um estímulo de outro neurônio e ambos são altamente ativos, o peso entre estes neurônios deve ser fortalecido, caso contrário enfraquecido.

Contudo, uma rede Perceptron tem como limitação o fato de só resolver problemas de classificação de conjuntos linearmente separáveis. A Figura 25, ilustra a classificação de classes.

Figura 25. Classes linearmente separáveis e não linearmente separáveis



Fonte: Souza (2007)

A rede Perceptron de camada simples pode ser descrita matematicamente pela equação: 6 (VALENÇA, 2001).

$$\Delta w_{ij} = \eta x_i (d_j - y_j) \tag{6}$$

Onde:

- **n** é a taxa de aprendizagem
- **xi** o valor apresentado a rede
- **dj** o valor desejado na saída e
- **yi** o valor calculado na saída.

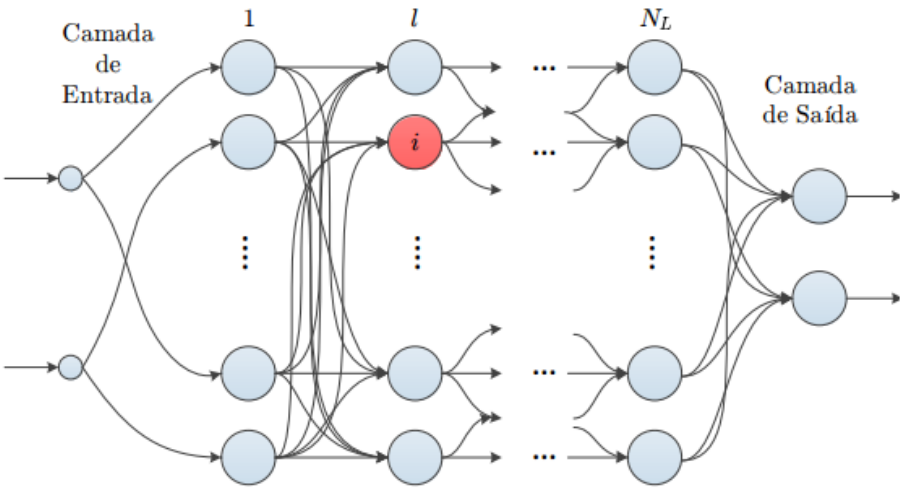
Inicialmente os valores dos pesos são atribuídos aleatoriamente. Em seguida, a resposta da rede é calculada e comparada ao o valor desejado para o reajuste ou não dos pesos sinápticos. O treinamento da rede Perceptron é do tipo supervisionado, uma vez que o valor desejado na saída é apresentado à rede, juntamente com as variáveis de entrada durante o treinamento.

Para a resolução de problemas não linearmente separáveis, pode-se utilizar a rede Perceptron de várias camadas, que será apresentada na próxima seção.

2.3.5 Rede Multilayer Perceptron

Segundo Haykin (2001), as redes *Multi-Layer Perceptron* são formadas por múltiplas camadas de neurônios *Perceptron* (do inglês, *multilayer perceptron*, MLP). A Figura 26 exibe a arquitetura geral de uma rede MLP.

Figura 26. Arquitetura de uma rede MLP.



Fonte: Souza (2007)

Em resumo, uma rede MLP é formada por diversas unidades de processamento, como sugere o modelo apresentado na Figura 26. Segundo Boccato (2013), ela pode ser dividida em três camadas principais:

- **entradas**: a camada de entrada recebe os sinais externos e os transmite para a primeira camada intermediária.
- **intermediárias:** formada por uma ou mais camadas intermediárias, tem o objetivo de lançar os sinais de entrada com base em um mapeamento não-linear sobre os dados, em espaços de dimensão maior que a do espaço original dos dados.
- **saída:** nessa camada os neurônios sofrem as ativações da última camada intermediária, onde através de combinações lineares destes sinais é realizada uma transformação não-linear dos dados, para produzir as saídas da RNA.

Com base no i-ésimo neurônio da l-ésima camada intermediária, como destacado na Figura 26, sendo i = 1,..., Nl e l = 1,..., NL, onde N1 representa o número de unidades presentes na camada l e NL o número total de camadas intermediárias da MLP. É apresentada pela equação 7.

$$y_i^l(n) = \varphi^l \left(\sum_{k=1}^{N^{l-1}} w_{ik}^l y_k^{l-1} + w_{i0}^l \right), \tag{7}$$

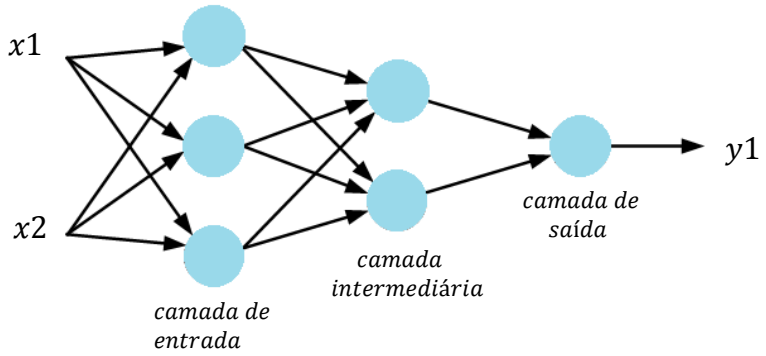
Onde, $y_i^l(n)$ representa o peso sináptico da conexão que liga o k-ésimo neurônio da camada l −1 ao i-ésimo neurônio da camada l.

O processo de treinamento de uma rede MLP consiste no ajuste de todos os pesos sinápticos das camadas intermediárias e de saída, tendo como objetivo encontrar o conjunto de pesos sinápticos que melhor realize o mapeamento possível de entrada-saída, isto é, a partir de um conjunto de entradas, fornecer valores para as saídas os mais próximos possíveis dos valores desejados (Haykin 2001).

2.3.6 Algoritmo de Treinamento Backpropagation

O algoritmo Backpropagation (BPN), ou propagação de erro, é um método comum aplicado em RNAs para aquisição do conhecimento sobre conjunto de dados. Foi descrito incialmente por Arthur E. Bryson e Yuchi Ho em 1969. A Figura 27 exibe o processo de funcionamento do algoritmo.

Figura 27. Processo de funcionamento do BPN



Fonte: http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Baseado na ilustração da Figura 27, e conforme Tosing (2000), e considerando uma rede supervisionada e com alimentação adiante (da esquerda para a direita), o processo de funcionamento do BPN é dividido em duas etapas: na primeira fase um conjunto de dados são apresentados para a camada de entrada da rede. Esses dados então são propagados através dos nós da rede, camada por camada, até que cheguem à última camada para que produzam um valor de saída. Na segunda fase, a saída obtida é comparada à saída desejada, e se a saída obtida não estiver dentro do valor esperado, o erro é calculado e propagado a partir da camada de saída até a camada de entrada.

Durante o processo de propagação, o valor recalculado na última camada pela BNP é transmitido para a camada de entrada, camada por camada. E nesse processo todos os pesos sinápticos são atualizados. Após a BPN chegar à camada de entrada, a primeira fase é executada novamente, e assim sucessivamente, até que a saída da rede seja satisfatória. Depois que a rede estiver treinada e o erro estiver em um bom nível de acurácia, ela poderá ser utilizada como uma ferramenta para classificação de novos dados.

A próxima seção apresenta um trabalho relacionado que emprega RNA com algoritmo de aprendizagem *backpropagation* em jogos eletrônicos.

2.4 TRABALHOS RELACIONADOS

Segundo Malone (1981), a pesquisa científica em jogos eletrônicos, quando comparada a outras áreas da computação, como a IA, por exemplo, é bastante recente. Embora existam registros de pesquisas envolvendo jogos, após o crescimento da indústria dos jogos e, por consequência, com a ampliação da demanda de profissionais mais especializados na área, é que houve um aumento considerável de interesse acadêmico.

Os jogos eletrônicos em RTS apresentam ambientes de jogos sofisticados e com certo nível de complexidade, fornecendo uma excelente base para testes e validação de novos métodos e algoritmos de IA (TATAI, 2003).

O trabalho revisado de Assis (2014) utiliza a aprendizagem por reforço com RNA em um ambiente de jogo 3D (um tabuleiro 10x10, totalizando 100 posições) desenvolvido na plataforma *Unity 3D*, e com a utilização de scripts para controlar o NPC. Nesse modelo, o agente interage com o ambiente e aprende a desviar de objetos e a vencer o oponente por tentativa e erro. O agente é recompensado por executar ações eficientes ou punido caso contrário. E dentro dos testes realizados, os resultados foram satisfatórios.

De forma mais detalhada, o trabalho de Assis (2014) apresenta um método de aprendizagem por reforço com RNA aplicado a jogos de ação em terceira pessoa. O modelo proposto permite que um personagem NPC explore o ambiente interativamente e execute uma ação. De acordo com sua eficiência, um reforço positivo ou negativo é dado ao agente. O ambiente possui itens e inimigos. O NPC é constituído por uma RNA *feedforward* que utiliza aprendizagem por reforço para buscar suas melhores ações.

As ações possíveis para o NPC poder explorar ou atacar um inimigo estão descritas na Tabela 8.

Tabela 8. Ações executadas pelo NPC em Assis (2014).

Ação	**Função**	**Equivalente binário**
1	Ataque ao norte	000
2	Ataque ao sul	001
3	Ataque ao leste	010
4	Ataque ao oeste	011
5	Explorar ao norte	100
6	Explorar ao sul	101
7	Explorar ao leste	110
8	Explorar ao oeste	111

A Tabela 9 mostra os valores de retornos que o NPC pode receber do ambiente.

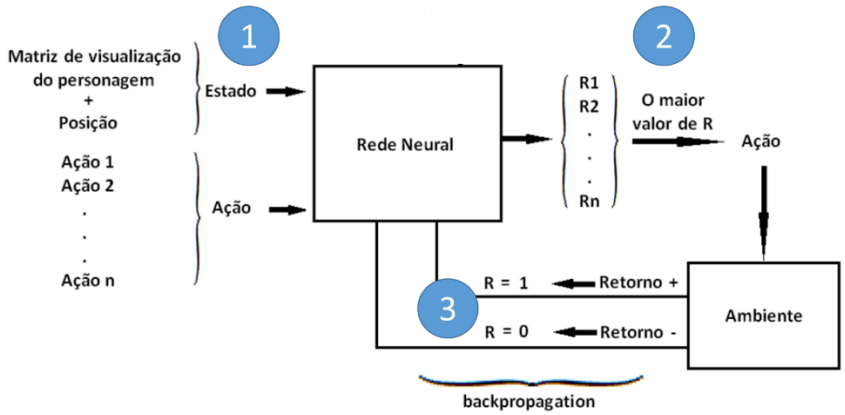
Tabela 9. Retorno do ambiente em Assis (2014)

Ação	**Retorno**
Dano	-5
Derrota um inimigo	1
Recolhe um item	5
Movimento simples	0

A partir da Tabela 9 é possível perceber que recolher um item é a ação que traz maior recompensa para o agente, pois, neste trabalho, essa é a sua principal colocação.

A ação do NPC é determinada em função do estado atual – posição, borda e visualização – e da ação que produz a maior saída da RNA. A Figura 28 apresenta esse processo.

Figura 28. Sistema de aprendizagem por reforço com RNA



Fonte: Assis (2014)

Baseado na Figura 28, o aprendizado ocorre da seguinte forma:

- **Primeiro**: o NPC percebe todo o ambiente, desde sua posição atual até a visualização da vizinhança, ou seja, se há oponentes na proximidade.
- **Segundo**: realiza várias estimativas para suas pretensas ações, conforme descritos na tabela 8 (explorar ou atacar), e verifica qual o retorno para cada uma dessas estimativas. O NPC escolhe, então, o retorno de maior valor.
- **Terceiro**: para cada ação de retorno existe um estado final, assim a rede produz um sinal de 0 e 1, que representa uma estimativa do sinal de reforço, sendo: 1 para estimativas positivas e 0 para negativas.

Assim, a ação escolhida é aquela que produz a maior saída da rede para uma dada entrada. Em outras palavras, escolhe-se o estado + ação que produz estados positivos.

Depois de executar a ação sugerida, o algoritmo de aprendizagem ajusta, em tempo real, os pesos da rede neural como segue: com um retorno positivo do ambiente, o elemento de aprendizagem considera que a saída desejada para a entrada atual (estado + ação que produziu maior saída) será 1 (ação bem sucedida); caso contrário, com um retorno negativo do ambiente, a saída desejada para a entrada atual será 0.

No conjunto de testes realizados, alguns parâmetros foram ajustados: o primeiro com algumas alterações no modelo de aprendizagem e o segundo referente às ações do personagem.

O primeiro grupo de testes se mostrou inadequado para o modelo proposto. No segundo grupo, o NPC, após um determinado tempo, optou apenas por atacar. O terceiro mostrou um NPC capaz de saber qual a melhor ação a tomar em um determinado estado; porém, necessitou de uma quantidade maior de interações para que isso ocorresse.

Segundo Assis (2014), os testes realizados mostraram a necessidade de uma ação além dos reforços provenientes do ambiente para que o NPC possa explorar o ambiente de forma mais eficiente.

A pesquisa de Assis (2014), serviu, em alguns aspectos, como guia para o desenvolvimento deste trabalho, como: utilização de RNA em jogos eletrônicos e utilização de tabelas para representar as ações do NPC. Contudo, em oposição às principais características da pesquisa de Assis (2004), o presente trabalho utiliza uma RNA com *backpropagation* para aprendizagem e um ambiente de jogo comercial.

A próxima seção apresenta a metodologia empregada no desenvolvimento deste trabalho.

3. MATERIAIS E MÉTODOS

Esta seção apresenta a metodologia utilizada para a realização deste trabalho, desde sua finalidade até os materiais e etapas necessárias para sua conclusão.

3.1 Desenho de Estudo

De acordo os objetivos deste trabalho, que é utilizar uma RNA para controlar o comportamento de um jogador no jogo *Starcraft*, esta pesquisa foi classificada como uma pesquisa aplicada, devido à utilização de fontes de conhecimento, tais como os conceitos apresentados no capítulo 1 (referencial teórico) para a resolução de problemas não triviais.

Para atingir o objetivo, foi realizado um levantamento bibliográfico em artigos científicos, teses e livros de IA. Foi realizada a aquisição do jogo comercial, e também foram obtidas as bibliotecas livres, BWAPI[11] e Encog[12]. As etapas empregadas para a execução do presente trabalho podem ser descritas em: criação do mapa do jogo; implementação do agente; definição da arquitetura da RNA; implementação de um módulo de treinamento da RNA e realização de testes.

Na implementação do agente do jogo alguns aspectos importantes foram considerados: definição da representação do estado do jogo; ações do agente e arquitetura RNA. Esses definições são descritas nas seções 3.3, 3.4 e 3.7 respectivamente.

3.2 Materiais

A comunicação do agente como o jogo Starcraft foi obtida através da biblioteca livre e de código aberto escrita em C++ BWAPI. Ela é utilizada para interagir com o jogo *Starcraft*, no qual estudantes, pesquisadores e amadores podem criar agentes de Inteligência Artificial (AI) que jogam o jogo.

A biblioteca revela somente as partes visíveis do estado do jogo para os módulos IA. As informações sobre as unidades que voltaram para o nevoeiro da guerra é negada à IA. Isso não permite que os programadores escrevam IA enganadoras, mas oportunamente IA mais competitivas, que devem planejar e operar em condições de informação parcial. A BWAPI também nega a entrada do usuário por padrão, garantindo que o usuário não possa assumir o controle das unidades de jogo enquanto a IA está sendo reproduzida.

A ferramenta Encog utilizada na implementação do módulo de treinamento, é uma estrutura de aprendizagem de máquina que suporta uma variedade de algoritmos, bem como

[11] Obtida em https://github.com/bwapi/bwapi
[12] Obtida em http://www.heatonresearch.com/encog

classes de suporte para normalizar e processar dados. A biblioteca possui vários algoritmos de aprendizagem de máquina, como: Máquinas de Vetores de Suporte, Redes Neurais Artificiais, Redes Bayesianas, Modelos de Markov, entre outros.

A biblioteca está disponível para as linguagens de programação Java, C# e C++. Também possui uma interface gráfica disponível em Java para criação de arquitetura de redes mais intuitivas e sem precisar usar programação.

O Encog possui suporte para tratamento de dados, funções de ativação e algoritmos para treinamento da rede. Também dá suporte ao critério de parada do treinamento, porém a biblioteca não possui um ponto de parada na convergência da taxa de erro, ou seja a parada é realizada através do número de iterações ou do valor da taxa de erro.

O emprego dessas bibliotecas deu-se principalmente pela vasta documentação existente na internet e também pelo suporte Java que elas oferecem.

3.3 Representação do Estado do Jogo

O estado do jogo deve possuir as informações necessárias para que o agente seja capaz de entender da melhor forma possível a situação atual do jogo a fim de escolher com maior excelência seus passos futuros. Entretanto, não foi encontrada durante a pesquisa, uma representação padrão de estados para a situação de combate em jogos RTS. Desta forma, para a etapa atual do trabalho foi necessário elaborar uma representação das possíveis situações dos componentes do jogo e das unidades do mapa que fosse capaz de prover conhecimento suficiente sobre todos os elementos da partida.

A representação do estado do jogo foi dividida em dois aspectos: agente e oponente. Esses aspectos compreendem o estado do jogo a cada momento de tempo em que a biblioteca BWAPI realiza uma leitura do jogo *Starcraft*. A lista dessas representações são:

1) **NPC:**
   - Ação anterior do NPC (explorar, atacar ou fugir)
   - Ação atual do NPC (explorar, atacar ou fugir)
   - Aproveitamento do NPC (positiva ou negativa)
   - Desempenho da ação (positiva ou negativa)
2) **Oponentes:**
   - Presença do oponente dentro do nevoeiro de guerra (sim ou não)
   - Distância do oponente (números inteiros)
   - Atacando (sim ou não)
   - Aproveitamento do oponente (positiva ou negativa)

A partir dessa representação do estado do jogo, supõe-se que o agente durante uma situação de batalha seja capaz de saber se está sendo atacado, se corre o risco de morrer em breve, se o oponente está em seu campo de visão e não está atacando-o, se é necessário desviar de obstáculo presente etc.

Os estados do jogo dessa representação fornecem os dados para a entrada de dados da RNA e a saída da rede fornece a próxima ação do agente. A seção seguinte descreve todas as ações previstas para o agente.

3.4 Ações do agente

O número de possíveis ações que podem ser executadas pelo agente em uma partida do jogo *Starcraft* pode ser demasiado. Segundo AHA et. al., (2005), a quantidade de ações possíveis de serem tomadas em um dado momento é estima em 1.500, em comparação com um jogo de xadrez que possui cerca de 30 ações, o jogo *Starcraft* oferece um índice de 98% superior de ações.

Apesar de alguns fatores considerados na estimativa não se aplicarem ao trabalho, como criação de base militares e mineração, o número serve para mostrar que pode ser inviável considerar todas as ações possíveis para um agente na tomada de decisão.

Sendo assim, foi definido que o agente poderá executar três ações: explorar, atacar ou fugir. Cada uma dessas ações tem associada a elas uma postura de alto nível, fazendo com que as implementações de baixo nível fiquem fora do aprendizado da RNA. Segue a lista com ações conjuntamente com suas posturas:

- **Explorar:** caso não exista um oponente dentro do campo de visão do agente, este deverá manter uma postura exploratória em relação ao mapa, percorrendo-o a fim de encontrar oponentes.
- **Atacar:** caso um oponente esteja dentro do seu campo visão, o agente deverá manter uma postura ofensiva, buscando causar o máximo de dano ao oponente.
- **Fugir:** nesta postura o agente deve ser capaz de recuar de uma batalha, caso sua sobrevivência esteja em risco real. Para isto, o agente deverá abdicar momentaneamente da batalha e fugir para um local seguro.

Esses comportamentos do agente apesar de representar uma pequena fração de todas as ações possíveis dentro de um jogo, busca estabelecer uma variedade suficiente de ações para derrotar o oponente do jogo. A próxima seção descreve o desenho da arquitetura da RNA

3.5 Implementação do agente

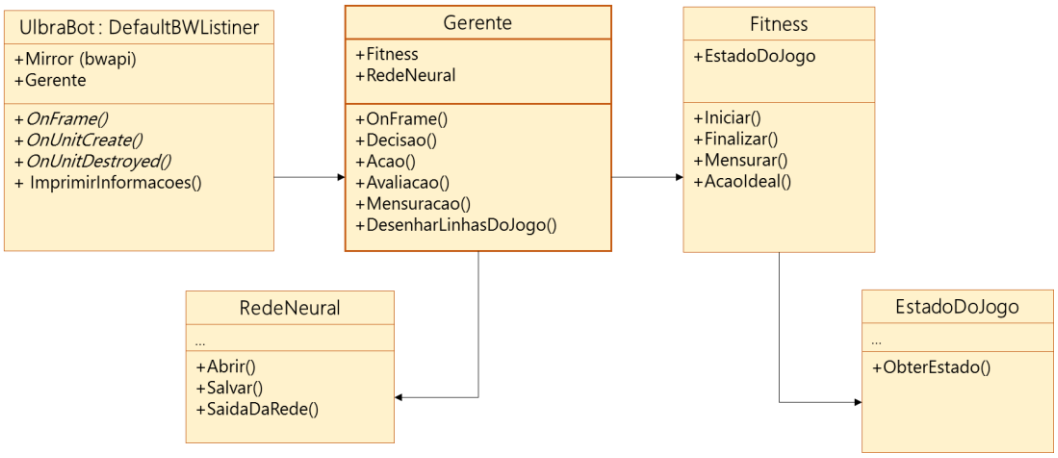
Segundo Micic (2011), uma maneira comum de implementar um agente inteligente capaz de jogar um jogo RTS completo é dividir o agente em gerentes. Assim, cada gerente tem a tarefa de gerenciar um aspecto importante do jogo, como, por exemplo, criar prédios e unidades, reunir recursos, explorar, atacar etc.

A ordem nas quais as estruturas são construídas durante um jogo é uma parte importante da estratégia global e muito relevante para o resultado final do jogo. Empregar um gerente exclusivamente dedicado a este propósito é uma tática interessante, porque jogos RTS são muito dinâmicos e novas informações podem influenciar a ordem de construção de forma significativa (MCCOY e et al., 2008).

Considerando que o propósito do trabalho é utilizar um ambiente de batalha entre duas unidades sem a construção de prédios e coleta de recursos, a unidade do agente de IA acumulará as funções de gerente e soldado, pois, além de coletar informações do jogo, ele ainda realizará o combate com o oponente.

Assim, o gerente de unidade deverá lidar com as decisões e ações de sua própria unidade. O diagrama de classes do gerente é descrita na Figura 31.

Figura 29. Diagrama de Classes do Gerente



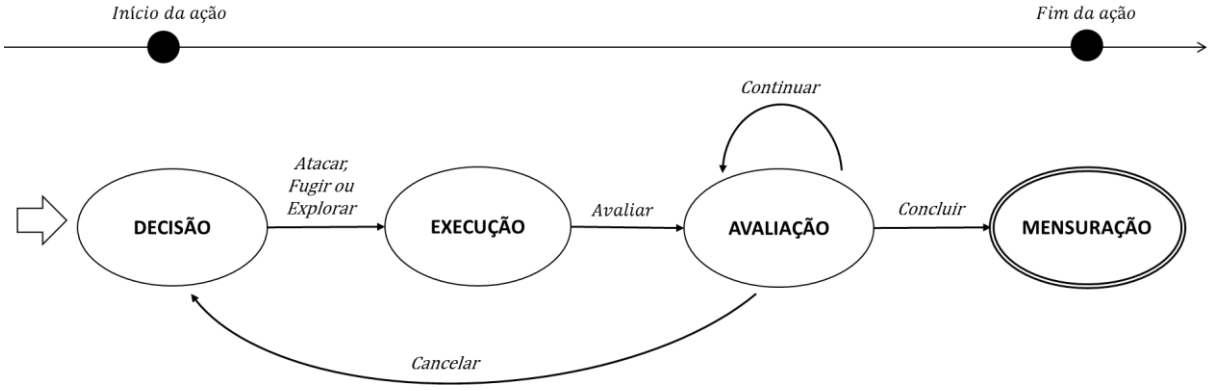
A classe UlbraBot, derivada da classe padrão DefaultBWListiner da API BWMirror, possui o método *OnFrame()* que é executado a cada frame do jogo *Starcraft*, e os métodos *OnUnitCreate()* e o *OnUnitDestroyed()*, que são chamados imediatamente quando uma unidade é criada ou destruída.

A implementação da classe Gerente tem basicamente três objetivos: decidir qual ação o agente deverá executar; controlar a execução dessa ação no jogo (considerando que uma

ação leva vários frames para ser executada) e, por fim, mesurar a ação logo após sua conclusão.

O controle da execução da ação e sua mensuração é baseada em uma Máquina de Estado Final (FSM). Esta máquina contém quatro estados: Decisão, Execução, Avaliação e Mensuração. A Figura 32 apresenta essa estrutura.

Figura 30. Máquina de estados finitos das ações do agente



No estado inicial de Decisão é realizado um processo de decisão da ação. Ele pode ser executado de duas formas: aleatório e a partir da saída da RNA. O processo de escolha aleatória é utilizado para criar o arquivo de treinamento da RNA. E o processo de escolha a partir da saída da RNA recebe uma saída da rede com base em uma entrada gerada a partir da representação atual do estado do jogo. Após escolher uma ação para o agente, a ação irá para o estado de Execução.

Durante o estado de Execução, o gerente envia o comando da ação para o jogo e passa para o estado de Avaliação.

No estado de Avaliação, o gerente avalia se a ação foi concluída ou se deve ser cancelada. Uma ação pode ser cancelada por dois motivos: quando a posição do oponente não pode ser determinada, ou quando uma ação está a muito tempo em execução. Caso ação seja executada com sucesso ela transitará para o estado de Mensuração. No estado de Mensuração é realizada uma medição do desempenho da ação do agente.

A Tabela 10 descreve como os valores da representação do estado do jogo são utilizados para realizar a medição do desempenho da ação do agente.

Tabela 10. Mensuração do desempenho das ações do NPC

Presença do oponente	Ataques em sequência	Fugas em sequência	Aproveitamento da ação do NPC	Aproveitamento da ação do oponente	Desempenho do NPC	Saída Ideal
0	**0**	**0**	**0**	**0**	**0,5**	**Explorar**
1	**0**	**0 0,5 1**	**0**	**0**	**1**	**Atacar**
1	**0,5**	**0**	**0**	**0**	**1**	**Atacar**
1	**0 0,5 1**	**0**	**1**	**0**	**1**	**Atacar**
1	**0**	**0 0,5 1**	**1**	**0**	**1**	**Atacar**
1	**0**	**0 0,5**	**1**	**1**	**0,5**	**Atacar**
1	**0**	**0**	**1**	**1**	**0,5**	**Atacar**
1	**0**	**0 0,5**	**0**	**0**	**0,5**	**Fugir**
1	**1**	**0**	**0**	**0**	**0,5**	**Fugir**
1	**0**	**0 0,5 1**	**0**	**1**	**0**	**Fugir**
1	**0 0,5 1**	**0**	**0**	**1**	**0**	**Fugir**
1	**0 0,5**	**0**	**1**	**1**	**0,5**	**Fugir**
1	**0**	**1**	**1**	**1**	**0,5**	**Fugir**

Os valores utilizados na tabela 10 são obtidos através do estado do jogo no início da ação e ao final da ação. A descrição detalhada dessas colunas é apresentada a seguir:

- **Presença do oponente** - caso exista um oponente dentro da área livre da nuvem de guerra, o valor desse atributo será 1, caso contrário será 0.
- **Ataques em sequência** – caso o agente tenha realizado anteriormente um ou nenhum ataque o valor desse atributo será 0, caso tenha realizado dois ataques em

sequência o valor será 0,5, ou se caso tenha realizado mais de dois ataques seguidos o valor será 1.

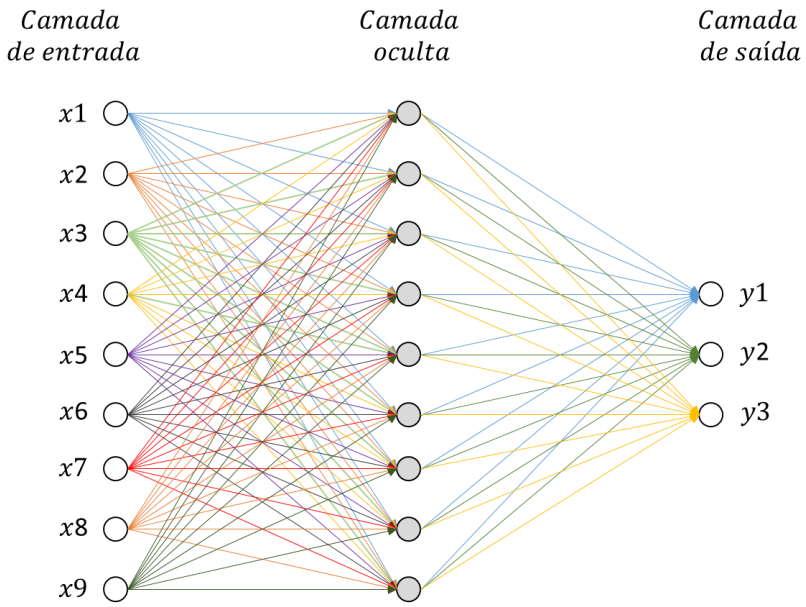
- **Fugas em sequência** – caso o agente tenha realizado anteriormente uma ou nenhuma fuga o valor desse atributo será 0, caso tenha realizado duas fugas em sequência o valor será 0,5, ou se caso tenha realizado mais de duas fugas seguidas o valor será 1.
- **Aproveitamento da ação do agente** – entre o estado inicial e o estado da ação final o NPC não tenha perdido pontos de vida o valor desse atributo será 1, caso contrário o valor será 0.
- **Aproveitamento da ação do oponente** - entre o estado inicial e o estado final da ação o oponente não tenha perdido pontos de vida o valor desse atributo será 1, caso contrário o valor será 0.
- **Desempenho do agente** – caso o aproveitamento da ação do oponente seja superior ao do agente o valor desse atributo será 0, caso os dois tenham o mesmo aproveitamento o valor será 0,5, ou caso o agente tenha um aproveito superior o valor será 1.

O empregado dessa tabela foi devido à dificuldade em encontrar na pesquisa bibliográfica uma função *fitness* para avaliar as ações do agente.

3.6 Arquitetura da RNA

Conforme as definições do trabalho, a arquitetura de rede utilizada será a *feedforward*, por ser uma arquitetura de rede supervisionada, e o algoritmo de aprendizagem *backpropagation*, conforme indicado na introdução deste trabalho. A definição da quantidade de neurônios na camada de entrada são equivalentes a quantidade de variáveis identificadas na seção 3.3 da representação do estado do jogo. E os neurônios da última camada foram definidos com base no número de ações do agente no jogo. A Figura 31 ilustra a arquitetura escolhida.

Figura 31. Arquitetura da RNA



A Figura 31 ilustra a quantidade de neurônios utilizados na arquitetura da RNA, sendo: 9 (nove) camadas de entrada, 9 (nove) camadas ocultas e 3 (três) camadas de saídas. As duas últimas camadas possuem a função Sigmoide como função de ativação e foi empregado o Bias na primeira e segunda camada.

Na camada de entrada os atributos foram definidos com base na representação do estado do jogo, onde:

- x1: presença do inimigo
- x2: distancia para o inimigo
- x3: ação anterior do agente
- x4: ação atual do agente
- x5: sequência de ataques
- x6: sequência de fugas
- x7: aproveitamento do agente
- x8: aproveitamento do oponente
- x9: desempenho do agente

Na camada de saída os atributos foram definidos com base nas ações do agente, onde:

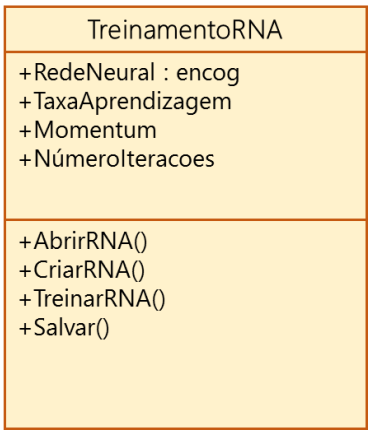
- y1: atacar
- y2: fugir
- y3: explorar

Para o processo de treinamento da RNA foi empregado o algoritmo de aprendizagem *Backpropagation*, e as taxas definidas foram:

- Aprendizagem: 0,5;
- Momentum: 0,7
- Critério de parada: 300 iterações
- Taxa de erro < $4{,}070^{-14}$

Com o aumento da taxa de aprendizado desconsiderando o valor definido para o momentum a rede se tornou instável. Contudo, se não for definido um momentum ou se o valor for baixo, o tempo para estabilizar a taxa de erro se torna muito longo. Na seção 4.4 é apresentado um gráfico que demonstra o momento em que a taxa de erro da rede estabiliza.

Figura 32. Diagrama de Classe do módulo de treinamento



O processo de treinamento da RNA foi realizado em um módulo secundário ao módulo de IA. A Figura 32 ilustra a classe e os métodos utilizados nessa implementação. A biblioteca Encog, apresentada na seção 3.2, foi utilizada para realizar o treinamento e gravação do arquivo da rede em disco.

Os dados utilizados no treinamento da RNA foram obtidos pelo módulo de IA no modo de execução: criação de arquivos para treinamento. Esse modo de execução armazena em disco, um arquivo com todas as ações do jogo, na qual o agente obteve vitória sobre o oponente. O arquivo gerado possui os dados de entrada e saída ideal que serão utilizados no treinamento da RNA.

A próxima seção apresentado os resultados e discussões realizados nesse trabalho.

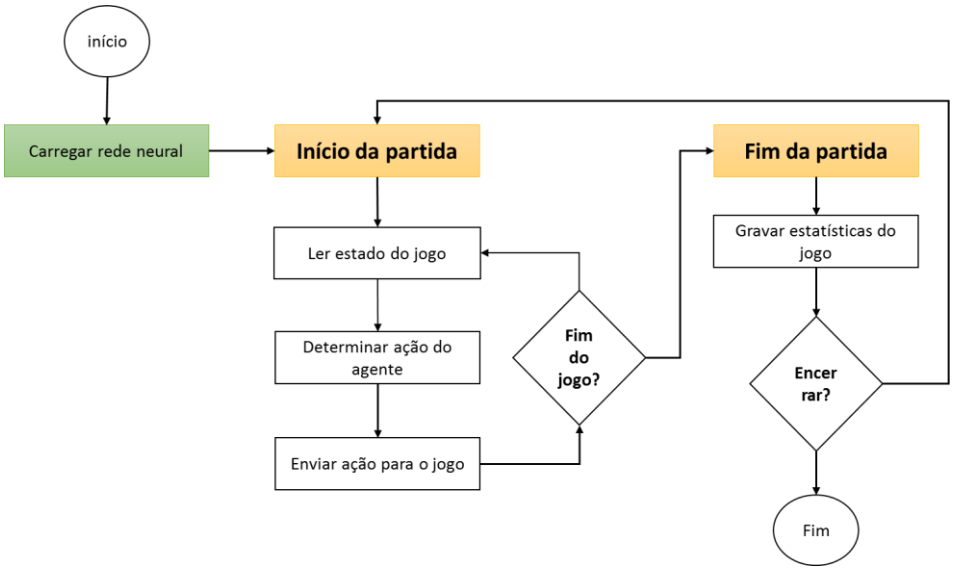
4. RESULTADOS E DISCUSSÕES

A presente seção mostra uma visão geral do módulo de IA utilizado para execução da RNA em batalhas do jogo *Starcraft*, bem como o mapa de jogo utilizado e suas restrições, além da criação e treinamento da rede neural e pôr fim a análise dos resultados obtidos.

4.1 O Módulo de IA

Para atingir o objetivo de utilizar uma RNA para controlar o comportamento de um agente no jogo *Starcraft*, foi necessário desenvolver um módulo de IA. O módulo desenvolvido controla uma unidade do jogo através da biblioteca BWAPI e a manipulação da RNA é obtida através da biblioteca Encog. A Figura 33 ilustra o funcionamento do módulo.

Figura 33. Fluxo de processos do módulo IA



O diagrama apresentado na Figura 33 ilustra o funcionamento do módulo de IA. Inicialmente, o módulo carrega os pesos da RNA através da biblioteca Encog para a memória, e inicia a partida no jogo Starcraft e logo em seguida entra em uma estrutura de repetição, onde três processos são executados repetidamente até que o fim da partida seja alcançado. Quando a partida chega ao final, o módulo grava um arquivo com as estatísticas obtidas. Os processos utilizados na estrutura de repetição estão detalhados na Figura 34.

Figura 34. Trecho de código do módulo IA

```
m_mensurarAcao.Iniciar(AcaoDaUnidade.Nenhuma);
double[] vetorDeEntrada = m_mensurarAcao.getEstadoDoJogo();

int indexAcao = m_neuralNet.classify(new BasicMLData(vetorDeEntrada));
novaAcao = AcaoDaUnidade.values()[indexAcao];

m_mensurarAcao.setAcaoRealizada(novaAcao);
m_acaoAtual = novaAcao;
```

A Figura 34 que apresenta um trecho do código do módulo de IA ilustra os três processos apresentados no fluxo de processos ilustrado na Figura 33.

O primeiro conjunto de linhas tem o objetivo de capturar através da biblioteca BWAPI o estado do jogo, que é armazenado na variável ***vetorDeEntrada.***

O segundo conjunto de linhas é utilizado para determinar a ação do agente com base no estado atual do jogo que está armazenado na variável ***vetorDeEntrada***. O resultado dessa ação é armazenado na variável ***novação,*** o valor armazenado - a ação do agente - foi obtido pela saída da rede neural através da biblioteca Encog.

O último conjunto de linhas é utilizado para enviar a ação escolhida para a unidade do jogo *Starcraft*, controlada pelo agente.

Figura 35. Imagem do Módulo de IA em funcionamento.

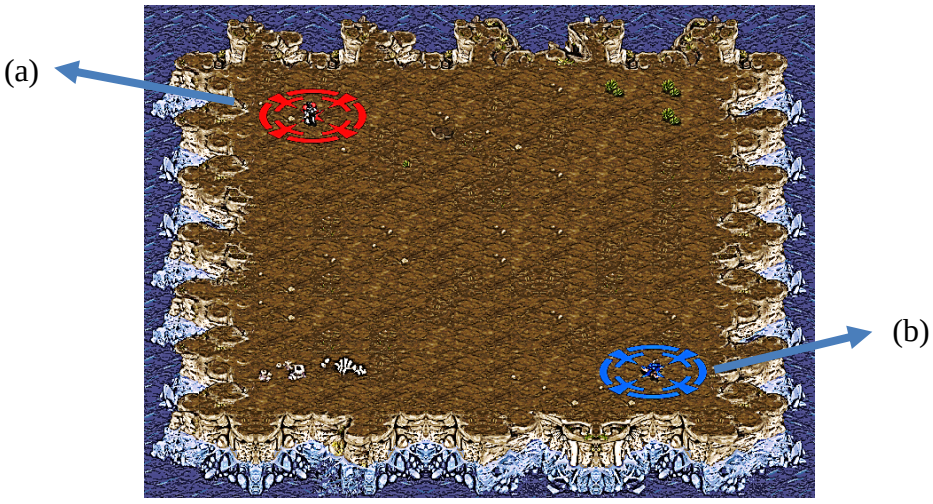
A Figura 35 apresenta uma captura de tela do jogo *Starcraft*, com o módulo de IA em modo de treinamento. No canto superior esquerdo é apresentado algumas estatísticas do agente, ao centro é identificado o agente em posição de ataque e o inimigo. O círculo branco identifica a área de ataque do agente.

As próximas seções apresentam as definições do mapa utilizado nas batalhas e também como a RNA foi criada, treinada e integrada ao módulo de IA.

4.2 Mapa do Jogo

Um mapa é um ambiente no qual os jogadores disputam uma partida. Ele é formado por uma área limitada que pode ser representada de diversas formas. No jogo *Starcraft*, um mapa pode ter cinco tamanhos diferentes, que vão desde 2.048 até 8.192 pontos. A Figura 36 ilustra uma pequena área de um mapa criado pela ferramenta *Starcraft Campaign Editor* fornecida juntamente com o jogo.

Figura 36. Figura do mapa do jogo Starcraft



O mapa apresentado na Figura 36 foi desenvolvido devido à dificuldade de encontrar na internet mapas com tamanho e recursos reduzidos, adequados ao propósito do trabalho.

Ele é formado por uma área quadrada total de 2.048 pontos, e possui uma pequena região de 600 pontos que fica isolada do restando do mapa. O mapa não possui obstáculos e contêm somente duas unidades: (a) Terran Ghost e (b) Zerg Zerling.

Um mapa além de conter regiões para criação de bases militares, extração de recursos e batalhas entre forças, permite durante sua confecção, a definição de alguns elementos da batalha, como, por exemplo: a quantidade de jogadores que disputará a partida (limitado a

oito jogadores), a quantidade de unidades que cada jogador irá iniciar a partida, os pontos de vida para cada unidade etc.

No mapa criado foi reduzido ao mínimo o número de recursos e unidades. Assim, no início das partidas cada unidade iniciará com 50 pontos de vida e com somente uma unidade de comando para cada jogador. Quando ocorrer a morte de uma unidade, esta voltará ao ponto de início conforme ilustrado na Figura 36 e as duas unidades reiniciarão os pontos de vida. Com isso, espera-se criar um ambiente de batalha dinâmico e equilibrado entre as unidades.

4.3 O arquivos de treinamento da RNA

O arquivo de treinamento da RNA utilizado nesse trabalho foi gerado no módulo de IA durante a execução do jogo *Starcraft*. A descrição detalhada desse processo pode ser observada na seção 4.5.

Os dados sobre a criação do arquivo são:

- **Número de exemplos do arquivo (ações do NPC)**: 35.400
- **Tempo de execução do módulo IA**: 1h e 24m
- **Número de partidas disputadas**: 5.000

As ações utilizadas no processo de treinamento fazem parte da coleção de ações executadas pelo agente durante as partidas no jogo *Starcraft*, nas quais o agente obteve vitória sobre o oponente. Esse filtro representa uma estratégia de não fornecer para o treinamento da RNA as ações malsucedidas do agente.

A estrutura do arquivo de treinamento pode ser observada na Figura 37.

Figura 37. Exemplo do arquivo de treinamento

```
1  1,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0; 0,0; 0,0; 0,0; 1,0; 0,0; 1,0; 1,0; 0,0; 0,0
2  1,0; 0,1; 1,0; 0,0; 0,0; 1,0; 0,0; 0,0; 0,5; 0,0; 1,0; 0,0; 1,0; 1,0; 0,0; 0,0
3  1,0; 0,6; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0; 1,0; 0,0; 1,0; 1,0; 0,5; 0,0; 1,0; 0,0
4  1,0; 0,6; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0; 0,5; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0
5  1,0; 1,0; 1,0; 0,0; 0,0; 1,0; 0,0; 0,0; 0,0; 0,0; 1,0; 0,0; 1,0; 1,0; 0,0; 0,0
6  1,0; 1,0; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,5; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0
7  1,0; 0,4; 1,0; 0,0; 0,0; 1,0; 0,0; 0,0; 0,0; 0,0; 1,0; 0,0; 1,0; 1,0; 0,0; 0,0
8  1,0; 0,4; 1,0; 0,0; 0,0; 1,0; 0,0; 0,0; 0,5; 0,0; 1,0; 0,0; 1,0; 1,0; 0,0; 0,0
9  1,0; 0,9; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0; 1,0; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0
10 1,0; 1,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0; 0,0; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0
11 1,0; 0,7; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0; 0,5; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0
12 1,0; 0,7; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0
13 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 1,0
14 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 1,0
15 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 1,0
16 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 1,0
17 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 0,0; 1,0
18 1,0; 0,0; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0; 0,0; 0,0; 0,0; 0,0; 1,0; 0,0; 0,0; 1,0
```

Conforme apresenta a Figura 37, os valores são separados por um ponto e vírgula e utiliza como separador decimal a virgula, contudo essa definição é indiferente para o

treinamento da rede, pois o módulo de treinamento descrito na seção 3.8 permite ler esse arquivo e fornecer para a rede um *dataset* no padrão próprio da biblioteca Encog.

4.4 Normalização dos dados para RNA

Antes de utilizar os dados na rede, estes precisam ser normalizados para que sejam eliminadas informações nulas ou valores muito altos. Na RNA do trabalho foram utilizados valores entre 0 e 1 para representar os estados do jogo. Esses valores podem ser observados na Tabela 12.

Tabela 11. Normalização dos dados de entrada da RNA

Atributo da rede	**Normalização**
Presença do oponente	0 – não está presente no duelo; 1 – está presente no duelo
Ataques em sequência	0 – para nenhum ou um ataque; 0.5 – para dois ataques; 1 – três ou mais ataques
Fugas em sequência	0 – para nenhuma ou um fuga; 0.5 – para duas fugas; 1 –três ou mais fugas
Aproveitamento da ação do NPC	0 – para sem aproveitamento; 1 – para com aproveitamento
Aproveitamento da ação do oponente	0 – para sem aproveitamento; 1 – para com aproveitamento
Desempenho do NPC	0 – para desempenho negativo; 0.5 – para empate de desempenho com o oponente 1 – para desempenho positivo

A formulação das entradas da RNA atinge um delicado equilíbrio entre tentar minimizar o número de pares do estado de jogo fornecidos para RNA aprender e manter as informação relevantes para a rede.

4.5 Criação e treinamento da RNA

A criação da rede neural obedeceu as definições apresentadas na seção 3.6. A arquitetura e os parâmetros utilizados, bem como o algoritmo de aprendizagem podem ser observados logo abaixo:

- **Camadas de entradas:** 9 neurônios sem função de ativação, e com o bias definido pela biblioteca Encog.
- **Camadas ocultas:** 9 neurônios ativados pela função Sigmoide e sem o bias.

- **Camadas de saídas:** 3 neurônios ativados pela função Sigmoide e com o bias definido pela biblioteca Encog.
- **Tipo de conexões:** *feed-forward.*
- **Inicialização:** inicializada randomicamente pela biblioteca Encog.
- **Função de Aprendizado:** *backpropagation.*
- **Taxa de Aprendizagem:** 0,5;
- **Momentum:** 0,7
- **Critério de parada:** 300 iterações
- **Taxa de erro** < $4{,}070^{-14}$
- **Arquivo de Padrões de Treinamento**: 1 arquivo (descrito na seção 4.3).

Considerando que o número de informações que podem ser obtidas pela representação do estado do jogo e utilizados como atributos de entrada da rede é relativamente alto, e também que o número de neurônios de entrada tem relação com o tempo de duração do treinamento, buscou-se um equilíbrio entre a quantidade de neurônio e as representações relevantes. Assim, os neurônios da camada de entrada e da camada oculta são resultado de estratégia para obter uma rede adequada para execução dos testes.

O número de neurônios da camada de saída representa, cada um, uma ação do agente no jogo, ou seja: atacar, fugir e explorar.

A Figura 38 apresenta o resultado do treinamento da rede realizado no módulo de treinamento com o uso da biblioteca Encog.

Figura 38. Resultado do treinamento da RNA

```
## Relatório do treinamento ##

+ Arquivos
  Nome do arquivo de log        : C:\jogos\Starcraft\ulbrabot\treinar1.log
  Nome do arquivo de rede neural: C:\jogos\Starcraft\ulbrabot\redeNeural-001.ann

+ Treinamento
  Número de registro...: 35.400
  Taxa de aprendizagem.: 0.5
  Momentum ............: 0.7
  Número de iteracoes..: 300
  Taxa de erro.........: 4.070446548765E-14
  Tempo de treinameto..: 09:51.774
```

Através da Figura 38 pode-se observar que o tempo gasto para o treinamento da rede para um arquivo com mais de 30 mil exemplos foi de menos de 10 minutos. A escolha para o número de iterações é explicado na próxima seção.

4.6 Estabilidade da taxa de erro

Uma RNA não consegue fornecer uma saída ideal a todo o momento, mas é importante diminuir esse erro. Essa diferença reside entre a saída da rede com a saída ideal.

A saída ideal utilizada no treinamento da rede é obtida através de um método implementado no módulo de IA. Esse método funciona como uma função de avaliação, onde é realizada uma mensuração do aproveitamento do NPC entre o estado do jogo de quando a ação iniciou e o estado do jogo quando ação foi concluída.

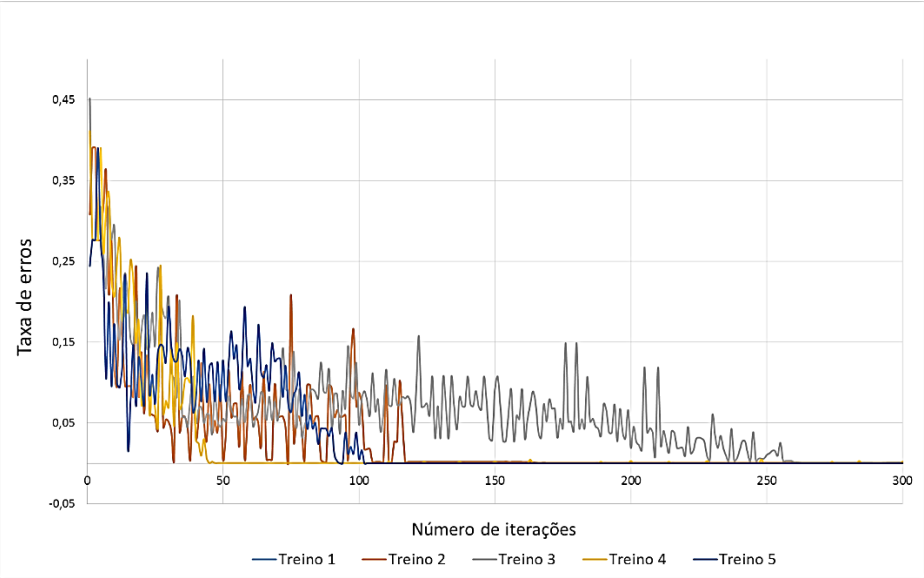
Por exemplo, caso o agente tenha realizado uma ação de ataque e, entre o início e o fim da ação o agente tenha perdido menos pontos de vida que seu oponente, considera-se que a ação realizada teve um desempenho positivo, nesse caso, para esse estado do jogo do início da ação, a ação atacar é a saída ideal. Caso contrário, se o desempenho do agente fosse negativo, a ação ideal seria fugir.

Outro exemplo é para o estado do jogo em que não existe oponente no raio de visão do agente, nesse caso, independente do desempenho do agente a ação ideal será a de explorar.

A Tabela 10 apresenta com maiores detalhes os valores utilizados para gerar a saída ideal.

Para determinar uma taxa de erro aceitável ou encontrar o momento em que a taxa de erro estabiliza durante o processo de treinamento foi utilizado bateria de 5 treinos com mesmo arquivo, porém com os pesos inicias definidos de forma randômica, o resultado dessa bateria é apresentado na Figura 39.

Figura 39. Estabilidade da taxa de treinamento



No gráfico da Figura 39 é possível perceber que a taxa de erro nos 5 treinos realizados começa a estabilizar próximo a 100 iterações e normaliza em 260. Essa bateria de teste permitiu definir um valor fixo de 300 iterações para o treinamento da RNA utilizada nos experimentos.

É importante considerar que esse processo foi utilizado, porque a biblioteca Encog não oferece suporte ao critério de parada do treinamento quando a rede estabiliza a taxa de erro, porém a biblioteca oferece duas forma de ponto de parada: quando a taxa de erro atingir um valor mínimo especifico, ou através de um número fixo de iterações.

4.7 Dados e informações sobre o testes

Para realizar as baterias de testes é importante considerar alguns aspectos sobre as raças escolhidas, como: ataque a longa distância e velocidade da movimentação no mapa. Essa seção aborda em detalhes esses comportamentos.

Figura 40. Unidades utilizadas nos combates

A Figura 40 apresenta as raças que foram utilizadas nas batalhas e também as raças controladas pelo módulo de IA do presente trabalho e as raças controladas pela IA padrão do jogo *Starcraft.* Assim, a primeira batalha foi entre duas unidades da raça Protoss, Dragoon e Zealot, e a segunda batalha entre duas raças distintas, o Ghost da raça Terran e Zergling dos Zergs.

Porém, antes de identificar o comportamento de cada raça durante uma batalha será apresentada algumas definições de âmbito geral das unidades.

Ao analisar as informações obtidas pela representação do jogo foi observado que, quando o oponente não puder ser localizado dentro do mapa as ações de atacar ou fugir do NPC geravam ações com tempo de execução relativamente longas e repetitivas. Assim, foi definido que, quando a localização do oponente não puder ser determinada, a ação de explorar deve ser executada até que o oponente possa ser localizado.

Outra informação importante diz respeito a escolha das raças utilizadas nas batalhas, que ocorreu durante a criação do mapa. Ficou definido que as batalhas que ocorrem no mapa serão entre as unidades de *Terran Ghost* versus *Zerg Zerling*, a justificativa da escolha dessas raças paira na intenção de encontrar um equilíbrio de forças para o duelo. Pois algumas unidades possuem características relativamente fortes para duelos em detrimentos de outras unidades.

O *Terran Ghost* tem um ataque de longo alcance e um tempo médio para recarregar sua arma, o *Zerg Zerling* por sua vez tem um ataque de curto alcance, porém possui uma mobilidade maior que seu oponente e um tempo baixo para recarregar suar arma. Outro ponto positivo que todas as unidades da raça *Zerg* possui é a cura natural para danos sofridos.

Assim, o *Terran Ghost* leva vantagem quando inicia um ataque distante do oponente, porém o *Zerg Zerling* tentará se aproximar rapidamente do *Terran Ghost* e iniciará seu ataque. Em um ataque cara a cara o *Terran Ghost* e *Zerg Zerling*, o *Terran* sairá perdedor na maioria dos casos.

Logo, o desafio do *Terran Ghost* é iniciar um ataque a distância e fugir do oponente quando este se aproximar. Contudo, o *Terran Ghost* não deve demorar para retomar o ataque, pois o *Zerg Zerling* pode se curar naturalmente.

Para a realização dos experimentos foram criadas 2 baterias com 5 de testes cada uma. O primeiro teste tem o objetivo de comparar a eficiência da RNA com outros trabalhos correlatos. O segundo teste tem o objetivo de validar se o processo de mensuração da ação gera informações relevantes para rede.

4.8 Teste 1 – Protoss Dragoon x Protoss Zealot

Apesar do arquivo de treinamento ter sido criado com uma base de exemplo de batalhas das unidade *Terran Ghost* e *Zerg Zergling*, para efeito de comparação com outros dois testes correlatos, foi realizado para o mesmo mapa criado no projeto, uma batalha entre as unidades *Dragoon* e *Zealot* ambos da raça *Protoss*. A unidade *Dragoon* tem um comportamento similar à Ghost, como: atacar a distância, tempo médio para recarga da arma e velocidade de movimentação média. As unidades *Zergling* e *Zealot* também possuem uma equivalência de suas características, como por exemplo a superioridade de força e velocidade sobre seus oponentes, contudo, essas unidade só atacam quando estão bem próximas de seus oponentes.

Este teste é similar aos testes descritos por Micic et. al. (2011) e BOTELHO NETO (2013), onde o NPC controla uma unidade *Dragoon* da raça *Protoss* e o oponente, no caso a IA padrão do jogo, controla o *Zealot* também da raça *Protoss.*

Tabela 12. Resultados das vitórias do teste 1

Bateria	Vitórias após 20 jogos (%)	Vitórias após 50 jogos (%)	Vitórias após 100 jogos (%)
1	100	98	95
2	100	98	91
3	98	95	92
4	100	96	94
5	95	95	92
Média	99	96	93

Na Tabela 12 é apresentado o valor percentual de vitórias obtidas pelo NPC após 20, 50 e 100 partidas disputadas dentro de uma bateria de 5 testes. A partir dos resultados é possível perceber que o NPC possui uma estratégia superior ao do oponente de batalha, obtendo bons resultados em todas as baterias, apresentando uma média de 96% de vitórias sobre o oponente. Em comparação com os resultados de outros trabalhos, o agente conseguiu superar os resultados obtidos por Micic et al. (2011) que, no melhor caso, alcançou 72% de vitória na mesma situação, e também por BOTELHO NETO (2013), que alcançou uma média 90% de vitórias.

Porém, é importante considerar que a RNA utilizada pelo agente não foi treinada com uma base de exemplos de batalhas das unidades utilizadas no teste. Uma forma de medir esse efeito é verificar a relação entre as ações apresentadas pela rede durante o processo de tomada de decisão da ação do NPC e a ação medida pelo processo de mensuração da ação. Esses valores são apresentados na Tabela 13.

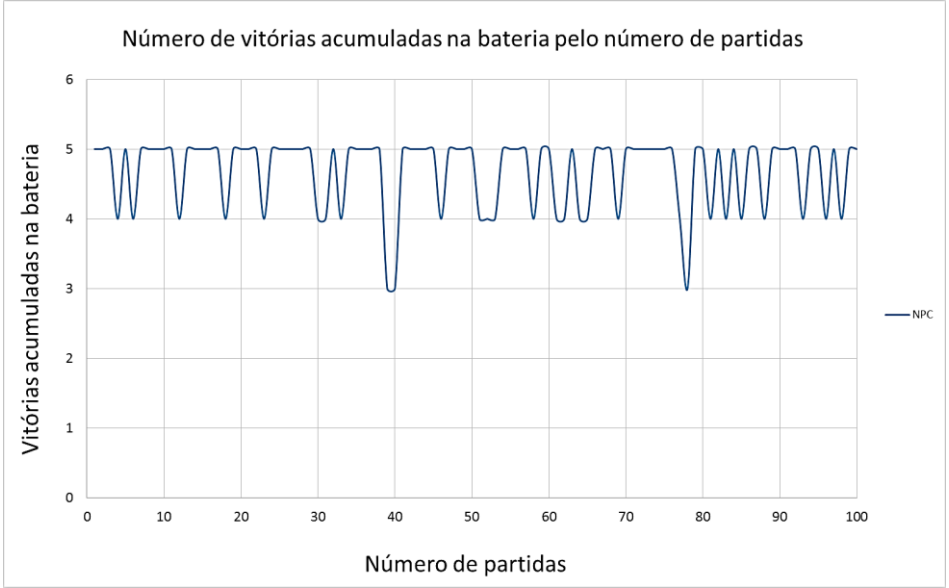
Tabela 13. Resultados das ações do NPC para o teste 1

Bateria	Ações iguais ao fitness	Ações diferentes do fitness	Correspondência entre as ações (%)
1	4.271	1.794	42
2	4.148	1.766	43
3	4.279	1.822	43
4	4.132	1.725	41
5	4.915	1.731	35

Na Tabela 13 é possível perceber uma distância entre as ações escolhidas pela rede e as ação mensuradas pela função *fitness*. Umas das hipóteses para essa diferença é que, mesmo que as unidades *Terran Ghost* e *Protoss Dragoon* possuam uma equivalência entre seus habilidades e características, ainda assim existem diferenças consideráveis, por exemplo, a unidade *Terran Ghost* pode iniciar um ataque a 220 pontos de distância do oponente, a unidade *Dragoon* porém, inicia uma ataque somente com menos de 170 pontos de distância. Esse fato pode gerar novas situações para as quais não foram realizados treinamentos.

A distribuição das vitórias que ocorrem dentro da bateria de teste pode ser observada na Figura 41.

Figura 41. Número de vitorias acumuladas no teste 1

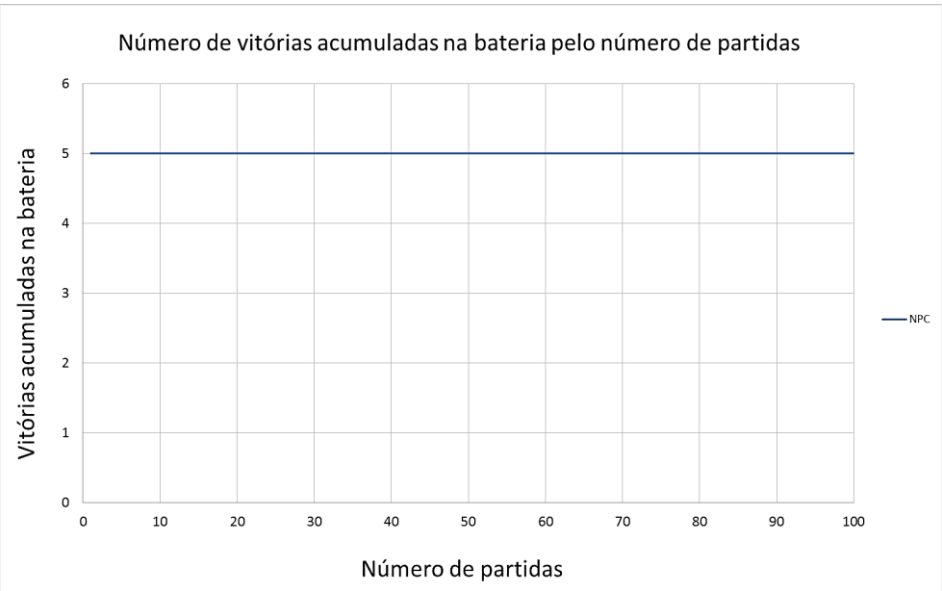


No gráfico da Figura 41 é possível perceber que a distribuição de vitórias durante a bateria de teste foi homogênea. O que demonstra a capacidade da rede em apresentar excelentes predições de ações para o agente diante da representação do estado do jogo durante as partidas.

4.9 Teste 2 – Terran Ghost x Zerg Zergling

Esse teste apresenta o contexto exato de unidades de combate e de exemplos para o qual o módulo de IA foi modelado. Assim, é possível verificar a eficiência da RNA na predição de ações, bem como dos métodos de mensuração da ação. Porém, não foi estabelecer uma comparação com outros trabalhos, por não terem realizado combate com essas unidades. A Figura 42 apresenta o resultado das vitórias do agente durante a rodada da bateria de teste.

Figura 42. Número de vitorias acumuladas no teste 2



No gráfico da Figura 42 é possível perceber que a distribuição de vitórias durante a bateria de teste foi retilínea, ou seja, o agente obteve a vitória em todas as partidas realizada nessa bateria. Esse resultado demonstra a superioridade de combate do agente sobre seu oponente controlado pela IA padrão do jogo *Starcraft*.

A relação entre as tomadas de decisões realizadas pela saídas da rede e as saídas ideais apontadas pela função *fitness* para essa bateria pode ser observada na Tabela 14.

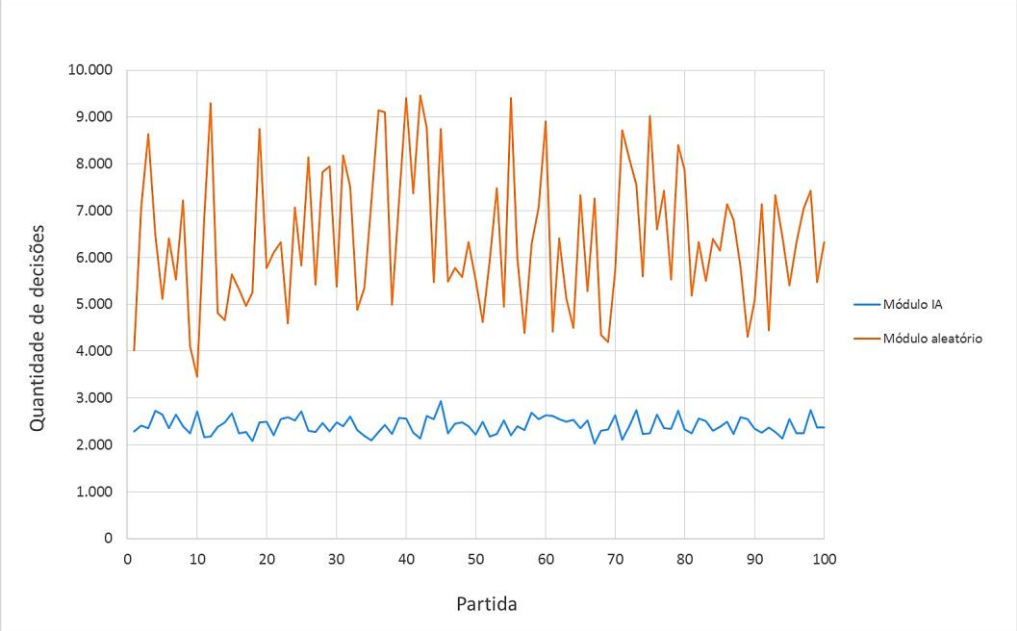
Tabela 14. Resultados das ações do NPC para o teste 2

Bateria	**Ações iguais ao fitness**	**Ações diferentes do fitness**	**Correspondência entre as ações (%)**
1	14.061	408	3
2	14.641	419	3
3	13.845	406	3
4	14.210	384	3
5	14.356	421	3

Na Tabela 14 é possível perceber que a distância entre as ações escolhidas pela rede e as ação mensuradas pela função de avaliação da ação é consideravelmente pequena. A partir desse resultado é razoável considerar que a rede foi capaz de aprender com os resultados da função de avaliação implementada.

A relação entre a quantidade de tomadas de decisões realizadas na execução do módulo IA, no modo de geração de exemplos para treinamento, pode ser observada na Figura 43.

Figura 43. Quantidade de decisões realizadas no processo de criação do arquivo de treinamento



No gráfico da Figura 43 é possível observar uma grande diferença na quantidade média de tomadas de decisões realizadas entre o módulo de IA e do processo aleatório de tomadas de decisões. Os detalhes sobre essa implementação pode ser encontrado na seção “Implementação do agente”.

A diferença observada no gráfico da quantidade de decisões em números absolutos, pode ser explicada pela forma aleatória com que módulo de IA responde quando uma requisição de decisão é feita. A motivação para apresentar essa diferença consiste em verificar se o agente processa as requisição de decisões de mais ordeira e planejada, o que é justificado no gráfico.

5. CONSIDERAÇÕES FINAIS

Este trabalho apresentou os principais conceitos e características da IA, com ênfase em jogos eletrônicos e também uma de suas técnicas, as RNAs, além de apresentar a descrição dos jogos RTS que são o domínio de aplicação, como também a ferramenta utilizada no processo de criação, treinamento e validação das RNAs, o Encog. O estudo de todos estes conceitos é indispensável para a compreensão do domínio e o desenvolvimento deste trabalho.

A implementação do módulo de inteligência foi desenvolvido em *Java* e utilizou a biblioteca BWAPI para realizar a interação com o jogo comercial *Starcraft: Brood War*, e controlar o comportamento de uma unidade. Esse ambiente foi escolhido de forma a servir como uma representação mais adequada para a demonstração da viabilidade da técnica proposta em situações de combates em jogos RTS.

Este trabalho abordou além do emprego de técnicas de aprendizagem de máquina, outros aspectos que são necessários à implementação com sucesso da proposta, como, por exemplo, um módulo de inteligência artificial para o jogo *Starcraft*, a representação do estado do jogo e as ações possíveis para um agente, além de uma arquitetura de RNA para o contexto de jogos RTS.

Foram realizados dois testes, o primeiro entre duas unidades da raça Protoss, o Dragoon e o Zealot, e a segunda entre duas raças distintas, o Ghost da raça Terran e o Zergling dos Zergs. Em todos os casos, verificou-se a ocorrência do aprendizado por parte do agente, que foi capaz de apresentar um comportamento superior nas situações apresentadas. Com isto, mostrou-se que a partir de uma modelagem adequada do problema, é possível aplicar técnicas clássicas do aprendizado supervisionado ao problema do combate em um RTS comercial com resultados positivos.

O fato do problema de tomada de decisão em um ambiente não-trivial - combate em jogos RTS - possuir características consideravelmente complexas para a inteligência artificial, como: inúmeros estados de jogo possíveis, ambiente com múltiplos agentes trabalhando em cooperação e competindo com outras equipes, coleta de recursos para evolução da estrutura militar, informação incompleta gerada pela nuvem de guerra, serve para aumentar a relevância do trabalho e reiterar a necessidade de ampliar os esforços na utilização deste tipo de técnica para a resolução de problemas do domínio de jogos RTS.

Deste forma, para trabalhos futuros, uma proposta é permitir que o agente trabalhe em equipe e passe a considerar as situações mais complexas, com as situações reais que os jogadores do jogo *Starcraft* encontram no cotidiano, e não apenas o combate entre duas unidades.

Para isso, o desafio maior será o de modelar o ambiente do jogo de forma a permitir que o agente trabalhe em equipe e possa se realizar a coleta de recursos, desenvolvimento de uma base militar, aprimoramento das unidades do jogo, sem que o espaço de busca torne-se proibitivamente grande de forma a impedir o aprendizado.

6. REFERÊNCIAS

AHA, D. W.; MOLINEUAX, M. **"Integrating Learning in Interactive Gaming Simulators".** Challenges in Game AI: Papers of the AAAI'04 Workshop, 2005.

ASSIS, Edival F. S. **Aprendizagem por Reforço com Rede Neural no Desenvolvimento de Jogos Digitais.** PUC-MG, Belo Horizonte-MG. 2012.

BARRETO, Jorge M. **Introdução às Redes Neurais Artificiais.** Tese de mestrado, UFSC, Florianópolis-SC. 2002.

BOCCATO, L. **Novas Propostas e Aplicações de Redes Neurais com Estados de Eco**. UNICAMP, Campinas-SP. 2013.

BORGES, D. M., Barreira, R. G., Souza, J. G. S. **Comportamento de personagens em jogos de computador**. Palmas: Centro Universitário Luterano de Palmas, 2009.
BURO, Michael; Kovarsky, A. **A first look at build-order optimization in real-time strategy games**. In Proceedings of the 2006 GameOn Conference, pages 18–22, 2006.

CASTRO, Israel H. L. **Um Framework para Pesquisa de Inteligência Artificial em Jogos de Estratégia por Turnos.** Belo Hozironte, UFMG, 2012.

CHAN, Hei.; Fern, A.; Ray, S.; Wilson, N. e Ventura, C. **Online Planning for Resource Production in Real-Time Strategy Games,** Oregon State University, Oregon: Corvallis, 2007.

COPPIN, Bem. **Inteligência Artificial**. Rio de Janeiro: LTC, 2010

GALDINO, Carlos Henrique Silva. **Inteligência artificial aplicada no desenvolvimento de jogos de computador**. Out. 2007. Disponível em: <http://www.programadoresdejogos.com/trab_academicos/carlos_galdino.pdf >. Acesso em 01 mai 2016.

GAMASUTRA, The Art & Business of Making Games, **Attempting to Define the RTS Genre**, 2015, Disponível em: <http://gamasutra.com/blogs/BrandonCasteel/20151002/255199/Attempting_to_Define_the_RTS_Genre.php>. Acesso em 01 abr. 2016

GAMEREPLAYERS, The History of Real Time Strategy, 2008, Disponível em: <http://sietch.net/Downloads/The_History_of_Real_Time_Strategy.pdf>. Acesso em 28 mar. de 2016

GEMINE, Quentin; Safadi, F., Fonteneau, R. e Ernst, D. **Imitative Learning for Real-Time Strategy Games**, in Proc. IEEE CIG, 2012

HAYKIN, Simon. Redes Neurais: princípios e prática; trad. Paulo Martins Engel. 2ª ed. Porto Alegre: Bookman, 2001.

JENNINGS, N.R. **Coordination tecniques for distributed artificial intelligence.** In: O'HARE, G.M.P.; JENNINGS, N.R. (Eds.). Foundations of distributed artificial intelligence. New York: John Wiley & Sons, 1996.

KARLSSON, Börje F. F. **Um Middleware de Inteligência Artificial para Jogos Digitais**. Dissertação de Mestrado, PUC-RJ. Rio de Janeiro-RJ. 2005.

MALONE, Thomas. **What makes computer games fun?**. In *Proceedings of the joint conference on Easier and more productive use of computers (CHI'81)*, Ann Arbor, USA. 1981

MICIC, A.; Arnarsson, D.; Jónsson, V. **Developing Game AI for the Real-Time Strategy Game *Starcraft***. Technical Report, Reykjavik University, 2011.

MILLINGTON, Y. **Artificial Intelligence for Games**, 2006.

MOURA, Jose. C. **Uma estratégia eficiente de coleta multiagente para jogos RTS.** Recife, UFPE, 2006.

NETO**,** Natal V. de S. **Aplicação de Técnicas de Inteligência Artificial na Implementação de Jogos Eletrônicos,** Uberlândia: Unitri, 2011.

NEWZOO, The Experts on all Things. **Consumer Insights**, 2016. Disponível em: <https://newzoo.com/solutions/consumer-insights/gamers/>. Acesso em 05 abr. de 2016.

RUSSELL, Stuart e Norvig, P. **Inteligência Artificial**. Tradução: Vanderberg D. De Souza. Rio de Janeiro: Elservier, 2004.

SCHWAB, B**. AI Game Engine Programming. Hingham**: Charles River Media. 2004.

SOUZA, Luciana de F. Rodrigues. **Redes Neurais Artificiais na Predição de Respostas e Estimação de Derivadas Aerodinâmicas de Aeronaves**. USP. São Paulo-SP. 2007

SYNNAEVE, B. e Bessiere, P. **A Bayesian Model for Plan Recognition in RTS Games Applied to *Starcraft***, Paris-FR, LPPA, College de France, 2011.

SYNNAEVE, G. **Bayesian programming and learning for multi-player video games**, Ph.D. dissertation, Université de Grenoble, 2012.

BOTELHO NETO, Gutenberg P., **Aprendizado por Reforço Aplicado ao Combate em Jogos Eletrônicos de Estratégia em Tempo Real.** João Pessoa-PB, UFPB, 2013.

TATAI, Victor Kazuo. **Técnicas de Sistemas Inteligentes Aplicadas ao Desenvolvimento de Jogos de Computador**. Campinas: Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, 2003.

TECH, Extreme. ***Starcraft* 2 AI hacks its way to victory,** Disponível em: <http://www.extremetech.com/gaming/102413-*Starcraft*-ii-playing-artificial-intelligence-shows-promise>. Acessado em 13 abr. de 2016.

TONSIG, Sérgio Luiz. **Redes Neurais Artificiais Multicamadas e o Algoritmo de Backpropagation**. Campinas, 2000. Disponível em: <http://209.123.181.8/~archives/tutoriais/1243.zip>. Acesso em 21 abr 2016.

TOZOUR, P. **The Evolution of Game AI from AI Game Programming Wisdom**. Hingham: Charles River Media. 2002.

VALENÇA, M. J. S. **Fundamentos das Redes Neurais: Exemplos em Java**. 1. ed. Recife: Livro Rápido, 2007.

WAYWARD, Strategist. **What is real-time strategy game? an exploration**. Set 2015. Disponível em: < http://waywardstrategist.com/2015/09/25/what-is-an-rts-game/>. Acesso em 10 abr 2016