

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

APRENDIZADO POR REFORÇO APLICADO
AO COMBATE EM JOGOS ELETRÔNICOS
DE ESTRATÉGIA EM TEMPO REAL

GUTENBERG PESSOA BOTELHO NETO

JOÃO PESSOA - PB

Março-2013

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

APRENDIZADO POR REFORÇO APLICADO AO
COMBATE EM JOGOS ELETRÔNICOS DE
ESTRATÉGIA EM TEMPO REAL

GUTENBERG PESSOA BOTELHO NETO

JOÃO PESSOA - PB

Março-2013

II

GUTENBERG PESSOA BOTELHO NETO

**APRENDIZADO POR REFORÇO APLICADO AO
COMBATE EM JOGOS ELETRÔNICOS DE
ESTRATÉGIA EM TEMPO REAL**

DISSERTAÇÃO APRESENTADA AO CENTRO DE INFORMÁTICA DA
UNIVERSIDADE FEDERAL DA PARAÍBA COMO REQUISITO PARCIAL
PARA A OBTENÇÃO DO TÍTULO DE MESTRE EM INFORMÁTICA
(SISTEMAS DE COMPUTAÇÃO).

ORIENTADOR: Prof. Dr. Claurton de Albuquerque Siebra

JOÃO PESSOA - PB

Março-2013

III

B748a Botelho Neto, Gutenberg Pessoa.
Aprendizado por esforço aplicado ao combate em jogos eletrônicos de estratégia em tempo real / Gutenberg Pessoa Botelho Neto.-- João Pessoa, 2013.
77f. : il.
Orientador: Clauirton de Albuquerque Siebra
Dissertação (Mestrado) - UFPB/CI
1. Informática. 2. Sistemas de computação. 3. Inteligência artificial. 4. RTS. 5. Aprendizagem de máquina. 6. SARSA.

UFPB/BC

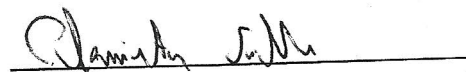
CDU: 004(043)

Ata da Sessão Pública de Defesa de Dissertação de
Mestrado de **Gutenberg Pessoa Botelho Neto**,
candidato ao Título de Mestre em Informática na
Área de Sistemas de Computação, realizada em 28
de Março de 2013.

Ao vigésimo oitavo dia do mês de Março do ano dois mil e treze, às dez horas, no auditório do CCEN- da Universidade Federal da Paraíba, reuniram-se os membros da Banca Examinadora constituída para examinar o candidato ao grau de Mestre em Informática, na área de “Sistemas de Computação”, na linha de pesquisa “Computação Distribuída”, o Sr. **Gutenberg Pessoa Botelho Neto**. A comissão examinadora foi composta pelos professores doutores: CLAUIRTON DE ALBUQUERQUE SIEBRA (PPGI-UFPB), Orientador e Presidente da Banca, NATASHA CORREIA QUEIROZ LINO (PPGI-UFPB), examinadora interna, JOSEANA MACEDO FECHINE (UFCG) como examinadora externa. Dando início aos trabalhos, o professor CLAUIRTON DE ALBUQUERQUE SIEBRA cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado “Aprendizado por Reforço Aplicado a Situações de Combate em Jogos Eletrônicos de Estratégia em Tempo Real”. Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: “Aprovado”. Assim sendo, deve a Universidade Federal da Paraíba expedir o respectivo diploma de Mestre em Informática na forma da lei e, para constar, eu, professor Alisson Vasconcelos de Brito, coordenador deste programa, servindo de secretário, lavrei a presente ata que vai assinada por mim mesmo e pelos membros da Banca Examinadora. João Pessoa, 28 de Março de 2013.

Alisson Vasconcelos de Brito

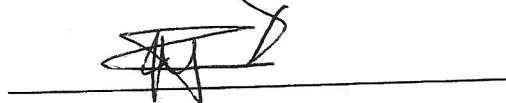
Prof. Dr. Clauirton de Albuquerque Siebra
Orientador (PPGI-UFPB)



Profa. Dra. Natasha Correia Queiroz Lino
Examinadora Interna (PPGI-UFPB)



Profa. Dra. Joseana Macedo Fachine
Examinadora Externa (UFCG)



AGRADECIMENTOS

Aos meus pais, Gutenberg e Hélia, à minha irmã Andréa, e a toda minha família, pelo apoio e incentivo tão importantes nos momentos necessários, sendo um alicerce para todas as horas e ocasiões.

Aos colegas da Universidade Federal da Paraíba com quem convivi nos últimos seis anos, durante os cursos de Graduação em Ciência da Computação e Mestrado em Informática, que comigo compartilharam tantos momentos e foram fundamentais nos êxitos obtidos durante esta caminhada.

Aos professores do Centro de Informática da UFPB, estes me enriqueceram com seus ensinamentos e conselhos, que levarei para sempre como guia nas minhas futuras jornadas acadêmica e profissional.

Aos companheiros de trabalho e de pesquisas acadêmicas, com os quais pude vivenciar as dificuldades e conquistas presentes em ambientes profissionais e desafiadores.

Por fim, a todos os amigos que se fizeram presentes de alguma forma em minha vida e que, à sua maneira, contribuíram para meu crescimento, muito obrigado!

"Em vez de tentar produzir um programa para simular a mente adulta, por que não tentar produzir um que simule a mente de uma criança? Se este fosse então sujeito a uma educação apropriada, obteríamos o cérebro adulto."

Alan Turing

RESUMO

Jogos eletrônicos e, em especial, jogos de estratégia em tempo real (RTS), são cada vez mais vistos como campos viáveis e importantes para pesquisas de inteligência artificial por possuírem características interessantes para a área, como a presença de ambientes complexos, muitas vezes dinâmicos e com múltiplos agentes. Nos jogos RTS comerciais, o comportamento do computador é geralmente definido a partir de técnicas *ad hoc* simples e estáticas, com a necessidade de definição manual de ações e a incapacidade de adaptação às situações encontradas. Esta abordagem, além de demorada e propícia a erros, faz com que o jogo se torne relativamente previsível após algum tempo, permitindo ao jogador eventualmente descobrir a estratégia utilizada pelo computador e desenvolver uma forma ótima de enfrentá-lo. Uma maneira de tentar combater esta previsibilidade consiste na utilização de técnicas de aprendizagem de máquina, mais especificamente do aprendizado por reforço, para permitir ao computador avaliar as situações ocorridas durante as partidas, aprendendo com estas situações e aprimorando seu conhecimento ao longo do tempo, sendo capaz de escolher de maneira autônoma e dinâmica a melhor ação quando necessário. Este trabalho propõe uma modelagem para a utilização de SARSA, uma técnica do aprendizado por reforço, aplicada a situações de combate em jogos RTS, com o objetivo de fazer com que o computador possa se portar de maneira mais adequada nessa área, uma das mais fundamentais para a busca da vitória em um jogo RTS. Nos testes realizados em diversas situações de jogo, o agente aplicando a modelagem proposta, enfrentando o oponente padrão controlado pela IA do jogo, foi sempre capaz de melhorar seus resultados ao longo do tempo, obtendo conhecimento acerca das melhores ações a serem tomadas a cada momento decisório e aproveitando esse conhecimento nas suas partidas futuras.

Palavras-chave: Inteligência artificial, RTS, aprendizagem de máquina, SARSA

ABSTRACT

Electronic games and, in particular, real-time strategy (RTS) games, are increasingly seen as viable and important fields for artificial intelligence research because of commonly held characteristics, like the presence of complex environments, usually dynamic and with multiple agents. In commercial RTS games, the computer behavior is mostly designed with simple *ad hoc*, static techniques that require manual definition of actions and leave the agent unable to adapt to the various situations it may find. This approach, besides being lengthy and error-prone, makes the game relatively predictable after some time, allowing the human player to eventually discover the strategy used by the computer and develop an optimal way of countering it. Using machine learning techniques like reinforcement learning is a way of trying to avoid this predictability, allowing the computer to evaluate the situations that occur during the games, learning with these situations and improving its behavior over time, being able to choose autonomously and dynamically the best action when needed. This work proposes a modeling for the use of SARSA, a reinforcement learning technique, applied to combat situations in RTS games, with the goal of allowing the computer to better perform in this fundamental area for achieving victory in an RTS game. Several tests were made with various game situations and the agent applying the proposed modeling, facing the game's default AI opponent, was able to improve its performance in all of them, developing knowledge about the best actions to choose for the various possible game states and using this knowledge in an efficient way to obtain better results in later games.

Keywords: Artificial intelligence, RTS, machine learning, SARSA

SUMÁRIO

LISTA DE FIGURAS	8
LISTA DE TABELAS	9
LISTA DE ACRÔNIMOS E SIGLAS	10
1. INTRODUÇÃO.....	11
1.1. Motivação	12
1.2. Objetivos.....	14
1.3. Organização do Trabalho	14
2. FUNDAMENTAÇÃO TEÓRICA	16
2.1. Jogos Eletrônicos	16
2.1.1. Jogos de Estratégia em Tempo Real (RTS).....	17
2.2. Ambientes Para a Criação de IA em Jogos RTS.....	22
2.2.1. ORTS	23
2.2.2. Stratagus	24
2.2.3. BWAPI	25
2.3. Aprendizagem de Máquina	27
2.4. Aprendizado por Reforço.....	29
2.4.1. Q-Learning.....	31
2.4.2. SARSA	32
3. TRABALHOS RELACIONADOS	34
3.1. Motivação para Pesquisas na Área	34
3.2. Aprendizagem Evolucionária em Jogos RTS	34
3.3. Raciocínio Baseado em Casos para Diferenciação entre Oponentes.....	35
3.4. <i>Frameworks</i> para Aprendizagem em Jogos RTS	36
3.5. Utilização de Árvores de Jogos em Situações de Combate	39
3.6. SARSA e LSPI Aplicadas a Jogos RTS	40
3.7. Planejamento Aplicado a Situações de Combate.....	41
3.8. Campos Potenciais Aplicados a Situações de Combate	42
3.9. Autonomia Orientada a Objetivos em Situações de Combate	43
3.10. Aprendizado por Reforço em Situação de Combate	43
3.11. Comparativo	44

4.	SOLUÇÃO PROPOSTA.....	47
4.1.	Ambiente.....	47
4.2.	Representação do Estado	48
4.3.	Ações do Agente.....	49
4.4.	Técnica - SARSA.....	51
5.	EXPERIMENTOS.....	54
5.1.	Teste 1 - <i>Protoss Dragoon</i> x <i>Protoss Zealot</i>	54
5.2.	Teste 2 - 4 <i>Protoss Dragoons</i> x 4 <i>Protoss Zealots</i>	56
5.3.	Teste 3 - 3 <i>Zerg Hydralisks</i> x 3 <i>Terran Vultures</i>	58
5.4.	Teste 4 - 1 <i>Protoss Dragoon</i> + 1 <i>Protoss Zealot</i> x 3 <i>Terran Marines</i> + 1 <i>Terran Medic</i>	59
5.5.	Teste 5 - 2 <i>Terran Vultures</i> + 1 <i>Terran Marine</i> x 2 <i>Zerg Hydralisks</i> + 2 <i>Zerg Zerglings</i>	61
5.6.	Resumo	62
6.	CONSIDERAÇÕES FINAIS	64
7.	REFERÊNCIAS BIBLIOGRÁFICAS	66
	ANEXO A	71

LISTA DE FIGURAS

Figura 1: Imagem do jogo Dune II.	20
Figura 2: Imagem de Starcraft: Brood War mostrando unidades e estruturas do jogo. .	21
Figura 3: Imagem do ORTS.	24
Figura 4: Imagem do Wargus.	25
Figura 5: Imagem de Starcraft: Brood War sendo executado com o BWAPI, incluindo artefatos de debug (Em: < http://www.cs.bham.ac.uk/~tpc/StarCraft/ >. Acesso em: fevereiro 2013).	26
Figura 6: <i>Framework</i> básico do RL. Adaptado de (SUTTON, 1999).	30
Figura 7: <i>Framework</i> proposto (ANDERSEN et al., 2009)	38
Figura 8: Exemplo de árvore de jogo para uma situação do Jogo da Velha (LAURSEN; NIELSEN, 2005)	39
Figura 9: Vitórias por Partida no Teste 1	56
Figura 10: Vitórias por Partida no Teste 2	57
Figura 11: Vitórias por Partida no Teste 3	59
Figura 12: Vitórias por Partida no Teste 4	60
Figura 13: Vitórias por Partida no Teste 5	62

LISTA DE TABELAS

Tabela 1: Resultados do Teste 1	55
Tabela 2: Resultados do Teste 2	57
Tabela 3: Resultados do Teste 3	58
Tabela 4: Resultados do Teste 4	60
Tabela 5: Resultados do Teste 5	61
Tabela 6: Resumo de resultados dos testes realizados	63

LISTA DE ACRÔNIMOS E SIGLAS

API - *Application Programming Interface (Interface de Programação de Aplicativos)*

BWAPI – *Brood War Application Programming Interface (Interface de Programação de Aplicativos para Brood War)*

DPS - *Dano Por Segundo*

GDA – *Goal-Driven Autonomy (Autonomia Orientada a Objetivos)*

IA – *Inteligência Artificial*

MAPF – *Multi-Agent Potential Fields (Campos Potenciais Multiagentes)*

MARL – *Multi-Agent Reinforcement Learning (Aprendizado por Reforço Multiagente)*

MDP – *Markov Decision Process (Processo de Decisão de Markov)*

ML – *Machine Learning (Aprendizagem de Máquina)*

NPC – *Non-Playable Character (Personagem Não-Jogável)*

RL – *Reinforcement Learning (Aprendizado por Reforço)*

RPG – *Role-Playing Game (Jogo de Interpretação de Personagens)*

RTS – *Real-Time Strategy (Estratégia em Tempo Real)*

SARSA - *State-Action-Reward-State-Action (Estado-Ação-Recompensa-Estado-Ação)*

1. INTRODUÇÃO

Os jogos eletrônicos tornam-se cada vez mais o foco de muitas pesquisas de inteligência artificial (IA), tanto pela percepção da indústria de que melhorias na IA podem aumentar o interesse dos jogadores, quanto pela visão dos pesquisadores de que os jogos eletrônicos representam um campo importante de estudo que pode trazer melhorias importantes para a área (LAIRD; VAN LENT, 2000; PONSEN, 2004).

Uma das áreas da IA que mais pode trazer avanços para os jogos eletrônicos é a ML (*machine learning*, ou aprendizagem de máquina). Com a ML, é possível fazer com que o computador aprenda no decorrer das partidas disputadas e possa modificar seu comportamento de acordo com as situações encontradas e com o adversário que estiver enfrentando. Desta forma, torna-se mais difícil a leitura da IA presente no jogo, aumentando a sua vida útil e deixando-o mais desafiador e atrativo para os jogadores.

Os jogos de Estratégia em Tempo Real (*Real-Time Strategy Games* - RTS) representam um gênero dos jogos eletrônicos caracterizado pela disputa de duas ou mais facções controladas por humanos ou pelo computador. Nos jogos do gênero, as responsabilidades dos jogadores geralmente incluem: coletar diferentes recursos, adquirir novas unidades e estruturas, explorar e expandir o território, engajar em combate com inimigos, entre outras (AHA et al., 2005; CHURCHILL; BURO, 2011; SAILER et al., 2007).

O fator tempo real é fundamental nestes jogos, pois exige do jogador a capacidade de adaptar-se rapidamente às mudanças no ambiente, tomando decisões rápidas, em múltiplos níveis de abstração, tentando cumprir diversos objetivos simultâneos e analisando diversos dados, muitas vezes sem conhecimento total do estado atual da partida, que podem ter ramificações de curto e longo prazo (WINTERMUTE, 2007). Os diversos aspectos de cada partida e responsabilidades do jogador exigem desse a capacidade de cuidar de diversas áreas de sua facção (e.g., econômica, política, militar) simultaneamente.

O gênero RTS é considerado um dos mais promissores para a pesquisa da IA em jogos eletrônicos pois contém uma variedade de problemas fundamentais da IA: planejamento adversarial em tempo real, tomada de decisão com informação incompleta, modelagem de oponentes, gerenciamento de recursos, colaboração de

grupos, busca de caminhos (*pathfinding*), entre outros (BURO, 2004; BURO; FURTAK, 2003). Jogos RTS oferecem, ainda, ambientes bem definidos para conduzir experimentos com medidas objetivas de desempenho e a possibilidade de criar uma IA específica para cada um dos diversos aspectos do jogo (e.g., combate, coleta de recursos) (BURO, 2004).

O combate é um dos aspectos fundamentais de um jogo RTS e fator de importância na definição do vencedor de cada partida, além de ser uma das mais difíceis tarefas existentes no jogo (SAFADI; ERNST, 2010). No combate, o jogador precisa manobrar suas unidades individuais de maneira coordenada na tentativa de derrotar as unidades inimigas, enquanto o oponente tenta fazer o mesmo com suas próprias unidades. Desta forma, para obter sucesso, é exigido do jogador um bom conhecimento sobre as possibilidades de cada unidade disponível para si e também sobre as possíveis táticas e estratégias de cada situação.

Com o objetivo de melhorar a qualidade da IA dos jogos RTS em situações de combate, o trabalho ora descrito propõe uma modelagem de aplicação de técnicas de ML para permitir que o jogador controlado pelo computador evolua de maneira autônoma o comando de suas unidades para melhorar sua efetividade nos combates, aprimorando-se a cada partida disputada. Com isto, também poderá ajustar suas táticas de combate de acordo com os oponentes e situações que encontrar com maior frequência, trazendo um maior desafio aos jogadores adversários.

1.1. Motivação

Mesmo com o aumento das pesquisas da IA em jogos e, em especial, no gênero RTS, a dificuldade de se criar uma IA completa e desafiadora para os jogadores humanos faz com que muitos jogos utilizem um simples conjunto de regras *ad hoc* para definir o comportamento dos jogadores virtuais, evitando comportamentos mais complexos e imprevisíveis por parte destes. Adicionalmente, é comum permitir que o computador obtenha vantagens não disponíveis ao jogador humano, de forma a oferecer um maior desafio ao mesmo (DIMITRIADIS, 2009; SAILER et al., 2007; SANDBERG, 2011; WALTHER, 2006). Dentre as vantagens mais comuns, estão: obtenção de recursos extras, permitindo que sejam criadas mais unidades e estruturas; dar bônus diversos de

forma a tornar suas ações mais eficientes do que as do humano; dar informações adicionais, como a permissão de poder ver o que está sob o *fog of war* (névoa da guerra - áreas do jogo que permanecem escondidas para um jogador até serem exploradas por ele) (MICIC et al., 2011; PÉREZ, 2011; RORMARK, 2009; WALTHER, 2006).

Outro problema comum da maioria dos jogos RTS do mercado é o comportamento estático e não-adaptativo dos jogadores controlados pelo computador. Isto faz com que, ao contrário do que se espera de um jogador inteligente, o computador seja um jogador previsível, que utiliza sempre a mesma estratégia. Como resultado, após algum tempo, os jogadores que enfrentam estes oponentes conseguem criar uma estratégia ótima para enfrentá-los, diminuindo o desafio e podendo fazer com que o jogo contra o computador torne-se menos atraente por conta da sua incapacidade de reagir e modificar sua forma de jogar para adaptar-se a seu oponente (DIMITRIADIS, 2009; JONSSON, 2012; KOK, 2008; MICIC et al., 2011; PONSEN, 2004).

Tendo em vista estes fatores, muitas pesquisas recentes da IA em jogos RTS são voltadas a técnicas de aprendizagem de máquina aplicadas a áreas diversas do jogo, com o objetivo de fazer com que o computador seja capaz de jogar partidas de maneira inteligente, com as mesmas informações dos jogadores humanos (i.e., sem vantagens adicionais) e com a possibilidade de adaptar-se ao adversário e testar novas estratégias (AHA et al., 2005; ANDERSEN et al., 2009; BALLA; FERN, 2009; DIMITRIADIS, 2009; JAIDEE et al., 2011; KOK, 2008; PONSEN, 2004; PONSEN; SPRONCK, 2004; RORMARK, 2009).

A motivação deste trabalho é a de analisar a viabilidade da utilização da aprendizagem de máquina voltada especificamente a situações de combate no jogo RTS. O combate é um bom campo para a IA, pois conta com muitas características complexas como a necessidade de interação e cooperação entre unidades aliadas e a presença de unidades inimigas com objetivos conflitantes.

Apesar da importância do combate para a definição do resultado de cada partida, ainda é relativamente escassa a quantidade de trabalhos voltados especificamente para essa área. A maioria dos trabalhos de IA para jogos RTS é voltada para a utilização de técnicas mais genéricas para situações gerais ou para problemas que não obriguem a levar em conta ações inimigas conflitantes, como planejamento de estratégia e coleta de recursos. Esta situação aumenta a importância de trabalhos que buscam explorar a

viabilidade da utilização de técnicas de ML para auxiliar o computador a se portar melhor nas situações de combate.

1.2. Objetivos

O objetivo geral do corrente trabalho é de explorar a viabilidade da aplicação de técnicas da IA, mais especificamente da aprendizagem de máquina, de forma a permitir a otimização ao longo do tempo das ações de combate por parte do computador em um jogo de estratégia em tempo real.

Para alcançar o objetivo geral do trabalho, foram definidos os seguintes objetivos específicos:

- Definir a técnica mais adequada a ser aplicada na implementação do trabalho e os parâmetros que serão utilizados por essa técnica;
- Propor uma modelagem do ambiente e das possíveis tarefas de forma a permitir a aprendizagem do agente;
- Definir o ambiente que será utilizado na implementação e validação do trabalho;
- Implementar a técnica apropriada no ambiente escolhido;
- Testar e validar a proposta a partir de experimentos em cenários específicos de combate de facções opostas.

1.3. Organização do Trabalho

O trabalho está estruturado em seis capítulos. Neste capítulo de introdução é apresentada a motivação inicial do trabalho, dando uma base histórica da utilização da IA em jogos e das pesquisas recentes da área, tratando de conceitos relacionados a jogos eletrônicos e jogos RTS relevantes para o entendimento do trabalho.

No segundo capítulo, são apresentados os conceitos principais do trabalho. É feita inicialmente uma apresentação dos jogos eletrônicos, explicando a viabilidade da utilização desses nas pesquisas de IA. Em seguida, é feito um detalhamento dos jogos

RTS e os desafios que estes apresentam para a IA e ML. O restante do capítulo dedica-se à aprendizagem de máquina, com a apresentação das principais estratégias utilizadas na área, dando ênfase ao aprendizado de reforço, com a explicação de algumas das suas principais técnicas.

No terceiro capítulo, são apresentados resumidamente alguns dos mais importantes trabalhos de IA e ML em jogos RTS, incluindo trabalhos focados no problema do combate nestes jogos.

O quarto capítulo serve para detalhar o trabalho proposto. Nele, é informado o ambiente escolhido para o desenvolvimento e também é explicada a técnica que será utilizada, além da modelagem proposta, por meio da função de recompensa proposta, representação de estado escolhida e ações que estarão disponíveis para os agentes.

No quinto capítulo, são detalhados os experimentos que foram realizados na validação do trabalho, incluindo a apresentação e discussão dos resultados obtidos.

O sexto capítulo trata das considerações finais da proposta apresentada. Nele, é feito um breve resumo do trabalho, de suas motivações e resultados. São destacadas, também, algumas das contribuições do trabalho e sugestões para trabalhos futuros.

Por fim, são apresentadas as referências bibliográficas utilizadas na composição do trabalho.

2. FUNDAMENTAÇÃO TEÒRICA

Nesse capítulo serão apresentados os conceitos necessários à compreensão do trabalho. Inicialmente, serão destacados conceitos mais gerais de jogos eletrônicos e jogos RTS. Em seguida, serão apresentados detalhes da inteligência artificial e da aprendizagem de máquina relevantes ao trabalho. O capítulo trata, ainda, dos ambientes mais utilizados para a implementação e validação de projetos de IA em jogos RTS.

2.1. Jogos Eletrônicos

Os jogos eletrônicos formam uma indústria poderosa, com faturamento de US\$15,9 bilhões no ano de 2010 apenas nos Estados Unidos, um crescimento de 200% em relação ao faturamento do ano 2000 (ESA, 2011). São exemplos de gêneros populares de jogos eletrônicos: Tiro, Ação, Aventura, RPG (*Role-Playing Game*, ou Jogo de Interpretação de Personagens), Estratégia, Simulação, Esportes, etc.

O aumento do interesse em jogos eletrônicos criou uma grande demanda do público por gráficos cada vez mais avançados e fotorrealistas, levando a avanços significativos no desenvolvimento desta área (LAURSEN; NIELSEN, 2005). Todo este foco na parte gráfica causou certo detrimento ao desenvolvimento de outras áreas dos jogos, como a inteligência artificial, que apesar de estar presente em muitos jogos, não obteve avanços tão consideráveis quanto outros aspectos e ainda é comumente implementada com técnicas simples e estáticas, que levam a comportamentos repetitivos e falta de adaptação às diferentes situações por parte dos jogadores controlados pelo computador (SAILER et al., 2007).

Recentemente, o aumento e barateamento de sistemas computacionais mais poderosos permitiu que a pesquisa em IA passasse a ser considerada mais viável para uso em jogos comerciais, com os desenvolvedores percebendo que a IA possui a capacidade de aumentar o valor do produto e o interesse dos consumidores. Os pesquisadores da área também passaram a ver os jogos cada vez mais como bons domínios de pesquisa e intensificaram os esforços nesse campo (LAIRD; VAN LENT, 2000; PONSEN, 2004).

A IA para jogos abrange um conjunto de diferentes problemas, todos unidos pela intenção de fazer com que o NPC (*non-playable character*, ou personagem não-jogável) seja capaz de agir de maneira inteligente, ou seja, de forma que possa observar as ações de seus adversários, analisar essas ações e reagir de uma maneira apropriada (EL RHALIBI et al., 2009; MATEAS, 2003). São características dos jogos eletrônicos que os tornam atrativos para a IA:

- Presença de ambientes complexos, muitas vezes possuindo propriedades do mundo real;
- Ambientes muitas vezes em tempo real e dinâmicos, exigindo decisões inteligentes e rápidas;
- Geralmente multiagentes, exigindo a capacidade de:
 - Pensar em termos de trabalho em equipe dos agentes com objetivos similares;
 - Pensar em termos de competição com relação aos agentes que possuem objetivos conflitantes;
 - Modelagem de NPCs de forma a poder analisar suas ações e planejar de acordo com o conhecimento obtido.

2.1.1. Jogos de Estratégia em Tempo Real (RTS)

Os jogos de RTS representam um gênero dos jogos eletrônicos caracterizado pela disputa de duas ou mais facções controladas por humanos ou pelo computador. Cada jogador controla uma facção e é responsável por tomar decisões nos níveis tático e estratégico que permitirão a expansão e evolução de sua facção na busca por alcançar os meios que poderão acarretar na vitória. Algumas das características da maioria dos jogos do gênero são (CHURCHILL; BURO, 2011; PÉREZ, 2011; SAILER et al., 2007; WINTERMUTE, 2007):

- **Representação do mundo** – o mundo nos jogos RTS é geralmente representado por mapas 2D de tamanhos variados. Muitos jogos utilizam o conceito de *fog of war* (MICIC et al., 2011; RORMARK, 2009; WALTHER, 2006), em que certas

partes do mapa ficam escondidas para cada jogador a menos que o mesmo possua presença na área por meio de uma unidade ou estrutura próxima;

- **Informação incompleta** – o jogador não sabe exatamente onde estão localizadas as estruturas e unidades do inimigo a menos que as encontre a partir da exploração. O jogador também não conhece as ações nem estratégias dos adversários;
- **Grande quantidade de dados** – os jogadores podem chegar a controlar centenas de unidades e estruturas ao mesmo tempo, devendo gerenciar os dados relacionados a cada uma simultaneamente;
- **Construção de unidades e estruturas** – os jogadores devem construir seu exército e sua base com diferentes unidades e estruturas a partir dos recursos coletados, escolhendo entre uma grande variedade de possibilidades, com características específicas e distintas;
- **Coleta de recursos** – os jogadores devem obter no decorrer do jogo, geralmente por meio de unidades trabalhadoras especializadas, recursos adicionais (e.g., ouro, metal, madeira) que servem para enriquecer cada facção e permitir a obtenção e construção de novas unidades e estruturas;
- **Pesquisa de tecnologia** – a pesquisa tecnológica permite que o jogador tenha acesso a estruturas e unidades mais avançadas a partir do gasto de recursos e de tempo;
- **Exploração e expansão de território** – é necessário explorar o território de forma a obter conhecimento sobre a geografia do mapa, incluindo a localização de recursos que devem ser coletados, assim como novos locais para a construção de estruturas que tragam algum benefício ao jogador. O jogador também deve tentar ocupar as porções de território que possam oferecer vantagens na busca do objetivo final de vitória na partida.

Estes jogos também incluem diversos problemas fundamentais da IA, o que os torna atrativos para serem utilizados em pesquisas da área. Entre os problemas normalmente presentes, estão (BURO, 2004; BURO; FURTAK, 2003):

- **Planejamento adversarial em tempo real** – este problema trata da necessidade do agente em um jogo RTS ser capaz de tomar, em tempo real, decisões que lhe darão maior possibilidade de vitória, levando em conta as possíveis ações que o adversário pode tomar para alcançar o mesmo objetivo;
- **Tomada de decisão sob incerteza** – o jogador inicia o jogo com pouca informação disponível, geralmente sem saber nem a localização dos inimigos nem a localização de recursos no mapa. Esta falta de conhecimento completo acerca da situação do jogo exige que sejam criadas hipóteses plausíveis para utilizar na tomada de decisão;
- **Gerenciamento de recursos** – o agente precisa gerenciar de maneira inteligente os recursos disponíveis. Os recursos são utilizados para construir e melhorar as unidades e estruturas e para pesquisar novas tecnologias. Um jogador que foque demais em apenas uma das áreas ficará vulnerável em outra, o que pode ser fundamental no resultado da partida;
- **Combate** – o combate com outras facções, seja para adquirir novos territórios ou para enfraquecer o adversário, é fundamental nestes jogos e obriga o jogador a controlar as suas unidades em um nível tático, buscando a melhor utilização de cada uma de forma a vencer o oponente. Podem ser destacados alguns subproblemas importantes no combate de jogos RTS (LAURSEN; NIELSEN, 2005):
 - **Posicionamento de unidades** – refere-se à necessidade de posicionar e movimentar as unidades disponíveis de forma a obter a maior vantagem possível tendo em vista o terreno onde o combate está acontecendo e as unidades inimigas próximas;
 - **Escolha de alvo** – problema relacionado com a decisão sobre qual unidade adversária deve ser atacada por cada agente aliado de forma a enfraquecer o adversário da maneira mais eficiente (e.g., eliminando o maior número possível de unidades inimigas rapidamente ou derrotando a unidade inimiga mais forte).

O gênero alcançou popularidade na década de 1990, a partir do jogo *Dune II*, de 1992 (Figura 1) e, após isso, por meio de vários jogos de popularidade crescente, como: *Warcraft* (1994) *Command & Conquer* (1995), *Total Annihilation* (1997) e *Starcraft* (1998). Este último, considerado o jogo RTS mais importante e de maior sucesso em todos os tempos, com mais de 11 milhões de vendas (GRAFT, 2009) e responsável pela explosão da cena profissional dos jogos eletrônicos, com jogadores obtendo patrocínios de grandes empresas e partidas sendo transmitidas pela TV em países como a Coreia do Sul (HUHH, 2007; STEWART; CHOI, 2003). *Starcraft* possui uma expansão, *Brood War* (1998) e uma sequência, *Starcraft II* (2010).

Figura 1: Imagem do jogo Dune II.



Starcraft (Figura 2) foi revolucionário pelo seu *design* assimétrico de raças. O jogo possui três raças completamente diferentes de si em termos de mecânicas de jogo, unidades e estruturas disponíveis. As raças são (MICIC et al., 2011):

- **Terran** – representando os humanos, esta é a raça mais equilibrada, possuindo unidades versáteis, relativamente baratas e com capacidade de atacar as unidades inimigas à distância;
- **Protoss** – representam uma raça futurista, com armas e estruturas avançadas. Suas unidades são individualmente as mais fortes do jogo, mas em compensação as mais caras e demoradas para serem produzidas;

- **Zerg** – raça mais primitiva, com as unidades mais baratas e fracas. Em geral, a estratégia dos jogadores que utilizam esta raça é sobrecarregar os oponentes a partir dos números.

O jogo possui dois recursos: minerais e gás. Estes recursos são coletados por unidades trabalhadoras (*workers*) disponíveis para cada raça. Além destes trabalhadores, as unidades restantes de todas as raças se dividem em dois tipos:

- **Unidades de Combate** – unidades voltadas ao ataque ao oponente. Algumas possuem a capacidade de atacar à longa distância (*ranged*) e outras apenas atacam à curta distância (*melee*). Algumas unidades são voadoras e não podem ser atacadas por oponentes *melee*;
- **Unidades de Suporte** – as unidades de suporte servem para ajudar de alguma forma as outras unidades, seja com a recuperação de vida (e.g., *medics*) ou transportando-as para locais previamente inacessíveis.

Figura 2: Imagem de Starcraft: Brood War mostrando unidades e estruturas do jogo.



Em *Starcraft*, assim como nos jogos RTS comerciais em geral, o jogador controlado por computador possui vantagens não disponíveis aos jogadores humanos,

de forma a poder oferecer um maior desafio a estes. Estas vantagens se dão de várias formas possíveis, como a não existência do *fog of war* para o computador, disponibilidade de mais recursos, construção mais rápida de unidades e estruturas, entre outras (DIMITRIADIS, 2009; MICIC et al., 2011; SAILER et al., 2007; SANDBERG, 2011; WALTHER, 2006). Este comportamento faz com que o jogador controlado pelo computador atue, efetivamente, com regras diferentes das aplicadas ao jogador humano, o que pode ser prejudicial para a percepção da qualidade do jogo em si, tendo em vista a má recepção dos jogadores à utilização deste tipo de artifício (NAREYEK, 2004).

Outro problema da IA de *Starcraft* e de outros jogos RTS comerciais é a utilização de técnicas simples e estáticas, muitas vezes com comportamento *ad hoc* definido à mão, para a escolha das ações do computador durante o jogo (LICHOCKI, 2008). Isto permite que o humano que jogue uma quantidade suficiente de partidas possa eventualmente descobrir o comportamento padrão do computador e desenvolver uma estratégia ótima para enfrentá-lo. Com isto, o jogo deixa de ser desafiador, podendo tornar-se chato e desinteressante.

O gênero RTS é considerado um dos mais desafiadores para o desenvolvimento de uma IA (SCOTT, 2002). Por conta das características do gênero e das muitas melhorias que podem ser feitas nas técnicas comumente utilizadas nestes jogos, um esforço tem sido feito nos últimos anos para aumentar a quantidade e o alcance das pesquisas acadêmicas da área de IA para jogos RTS (BURO, 2004; BURO; FURTAK, 2003).

2.2. Ambientes Para a Criação de IA em Jogos RTS

Uma forma de facilitar o desenvolvimento de uma IA para jogos RTS é com a utilização de um ambiente apropriado para tal. O ambiente provê o jogo em si e todas as ferramentas necessárias para a modificação do comportamento do NPC durante uma partida. Isto permite que o desenvolvedor foque apenas na pesquisa e na programação relacionada à IA, e não em outras áreas como o desenvolvimento do jogo propriamente dito. Três ambientes comumente utilizados em pesquisas de IA serão detalhados: ORTS (*Open Real-Time Strategy*) (Em: <<http://skatgame.net/mburo/orts/>>. Acesso em: fevereiro 2013), Stratagus (PONSEN et al., 2005) e BWAPI (*Brood War Application*

Programming Interface) (Em: <<http://code.google.com/p/bwapi/>>. Acesso em: fevereiro 2013).

2.2.1. ORTS

O ORTS (Figura 3) é um projeto de *software* aberto criado por Buro (2002) com a intenção de facilitar a pesquisa na área de IA para jogos RTS, após o autor perceber a falta de ambientes e ferramentas apropriadas para tal. Algumas características do ORTS (BURO; FURTAK, 2003):

- **Simulação pelo lado do servidor** – a simulação dos jogos ocorre em um servidor central, desta forma a segurança é aumentada e a possibilidade de *hacking* existente em sistemas baseados em simulação via cliente é evitada;
- **Especificação flexível** – o ORTS dá a possibilidade de customizar as regras do jogo e também permite modificar as características e ações das unidades. Desta forma, o pesquisador pode implementar a IA sob as condições exatas que desejar;
- **Protocolo aberto de mensagens** – permite que os jogadores utilizem diversos clientes com o simulador, tendo assim a possibilidade de utilizar variadas interfaces de usuário, de utilizar clientes para serem controlados por jogadores humanos, de utilizar clientes para serem controlados por uma IA, clientes híbridos, etc.

O ORTS foi um ambiente muito popular durante anos para o desenvolvimento e validação de IA para jogos RTS, servindo até mesmo como ambiente para competições entre IAs (Em: <<http://skatgame.net/mburo/orts/AIIDE09/>>. Acesso em: fevereiro 2013). Nos últimos anos, porém, o seu uso vem diminuindo e tem sido vista uma transição gradual de pesquisadores e competições para o BWAPI, inclusive por parte do criador do ORTS (CHURCHILL; BURO, 2011).

Figura 3: Imagem do ORTS.



2.2.2. Stratagus

O Stratagus é um *engine* de código aberto para a criação de jogos RTS que também pode ser utilizado para pesquisas de IA. Algumas das vantagens do Stratagus:

- **Configurável** – o *engine* permite a criação de diversos jogos RTS diferentes, incluindo jogos focados em tarefas específicas de IA. Também permite modificar facilmente as características de unidades e estruturas a partir da edição de arquivos de texto específicos;
- **Integração com Lua** – apesar de ser baseado na linguagem C, o Stratagus permite a integração dos jogos com a linguagem de *scripting* Lua. Desta forma, os desenvolvedores podem utilizar a linguagem Lua para modificar a IA dos jogos em vez de trabalhar diretamente no código da *engine*;
- **Fast forwarding** – o Stratagus possui uma opção de *fast forwarding* baseado no desligamento parcial dos gráficos, que permite simular com maior rapidez as partidas. Com isto, torna mais rápida a execução de experimentos;

- **Estatísticas** – várias estatísticas úteis são gravadas automaticamente pelo Stratagus durante e após as partidas (e.g., tempo de jogo, número de unidades perdidas) que auxiliam na avaliação das partidas disputadas.

Um jogo baseado em Stratagus particularmente popular para as pesquisas de IA é o Wargus (Figura 4), um clone do jogo *Warcraft 2*, um dos mais populares jogos RTS de todos os tempos (Em: <<http://wargus.sourceforge.net/index.shtml>>. Acesso em: fevereiro 2013). Apesar de não ser completamente fiel ao jogo original por conta do *engine* diferente utilizado, o Wargus tornou-se uma opção muito popular por permitir a validação das IAs dentro de uma simulação de um jogo comercial. A popularidade só veio a diminuir com o lançamento do BWAPI, que passou a permitir a implementação da IA diretamente no jogo original, em vez de necessitar de um clone para tal.

Figura 4: Imagem do Wargus.



2.2.3. BWAPI

O BWAPI (Figura 5) é um *framework* livre que permite a criação de módulos de IA, com a linguagem de programação C++, a serem utilizados no jogo *Starcraft: Brood War*.

O *framework* oferece ao desenvolvedor todas as informações que o jogo oferece a qualquer jogador (e.g., unidades e estruturas disponíveis, locais visíveis do mapa) e permite que seja executada programaticamente qualquer ação que um jogador regular poderia executar durante uma partida. Durante uma partida com o BWAPI, a interface do usuário é desabilitada de forma a impedir que o humano tome controle da facção em determinados momentos para obter mais chances de vitória.

Figura 5: Imagem de Starcraft: Brood War sendo executado com o BWAPI, incluindo artefatos de debug (Em: <<http://www.cs.bham.ac.uk/~tpc/StarCraft/>>. Acesso em: fevereiro 2013).



Apesar de não ser tão versátil quanto outros ambientes, o BWAPI tem a vantagem de permitir ao desenvolvedor implementar e executar a IA diretamente no jogo mais popular de RTS existente, garantindo a possibilidade de testar suas técnicas diretamente no ambiente de um jogo comercial real, diferentemente do Wargus que apenas simula tal ambiente, além de já oferecer um jogo completo no qual a IA pode ser aplicada, diferentemente do ORTS que oferece apenas os aspectos mais básicos do jogo e deixa para o desenvolvedor o trabalho restante (WINTERMUTE, 2007). Por conta

destes fatores, esta é, atualmente, a principal ferramenta para desenvolvimento e validação de IA para jogos RTS.

Além das publicações recentes de trabalhos implementados no BWAPI, a popularidade do ambiente pode ser comprovada pelas competições de IA que o utilizam como base (Em: <<http://code.google.com/p/bwapi/wiki/Competitions>>. Acesso em: fevereiro 2013).

2.3. Aprendizagem de Máquina

A aprendizagem de máquina é uma área da IA que busca permitir que os computadores sejam capazes de modificar seu comportamento a partir de conhecimento adquirido por meio de dados diversos. Técnicas de ML são utilizadas com sucesso em campos como (MITCHELL, 2006): reconhecimento de voz, visão computacional, biovigilância, controle de robôs, controle autônomo de helicóptero (ABBEEL et al., 2010), entre outros.

Pelo fato de que a IA em jogos eletrônicos costuma ser representada de forma estática e imutável, tornando-se previsível para o jogador após algum tempo, acredita-se que a aprendizagem de máquina pode ser uma das grandes inovações a serem aplicadas a jogos eletrônicos (DIMITRIADIS, 2009; JONSSON, 2012; KOK, 2008; MICIC et al., 2011; PONSEN, 2004), possuindo a capacidade de deixar os jogos mais robustos e obrigando os jogadores a evoluir suas estratégias e modificar continuamente seu comportamento ao enfrentar o computador.

Técnicas de ML podem ser *online* ou *offline* (PONSEN, 2004; SUTTON; WHITEHEAD, 1993). Nas aplicações *offline*, todo o treinamento do sistema é feito *a priori* e o conhecimento do sistema em relação ao ambiente não é modificado durante a sua execução prática. Nas aplicações *online*, o sistema refina continuamente o seu conhecimento de mundo a partir dos dados coletados durante a operação, adaptando sua função de aprendizado de acordo com os novos dados.

Aplicações *online* possuem várias vantagens em relação a aplicações *offline* (SUTTON; WHITEHEAD, 1993). Além de serem mais robustas, por permitirem a correção de erros do conjunto de treinamento original a partir dos novos conhecimentos

obtidos, aplicações *online* também facilitam a geração de dados de treinamento para alimentar o sistema, que podem ser obtidos apenas pela execução normal, porém, em muitos casos é difícil a obtenção anterior de dados de treinamento confiáveis e em volume suficiente. Por conta destas vantagens, o fato de uma aplicação ser *online* pode ser fundamental em determinados casos para a obtenção de resultados positivos, incluindo treinamento para jogos como o Gamão (TESAURO, 1992).

Existem três abordagens principais para o aprendizado:

- **Aprendizado supervisionado** – neste tipo de aprendizado, há um conjunto de dados de entrada e também um conjunto de saídas corretas para estes dados. Desta forma, o sistema já possui um banco de dados de início e pode se basear neste banco para fazer induções, formulando hipóteses que levarão ao aprendizado, na forma de um classificador (para saídas discretas) ou de uma função de regressão (para saídas contínuas);
 - **Exemplos de algoritmos:** Regressão Linear, *k-Nearest Neighbors* (k-Vizinhos Mais Próximos).
- **Aprendizado não-supervisionado** – neste tipo de aprendizado há apenas um conjunto de entradas sem que o agente possua ou seja capaz de inferir qualquer informação sobre as saídas corretas. A ideia é que seja possível estruturar e encontrar grupos de dados similares dentro do conjunto de entrada;
 - **Exemplos de algoritmos:** *k-Means Clustering* (k-Meios), DBSCAN.
- **Aprendizado por reforço (RL – *Reinforcement Learning*)** – neste aprendizado não há conhecimento das saídas corretas para cada entrada, mas existe uma função de recompensa que retorna um valor para cada entrada passada para o algoritmo de forma que o agente possa descobrir por conta própria as saídas que julgar melhores.
 - **Exemplos de algoritmos:** Q-Learning, SARSA.

A abordagem do aprendizado por reforço é considerada a mais apropriada para problemas estocásticos ou com espaços de busca muito grandes (SUTTON, 1999), o que é normalmente o caso dos problemas relacionados a jogos computacionais e voltados à melhoria de desempenho do agente computacional no decorrer das partidas.

O combate em um jogo RTS apresenta estas características e, por conta disto, o aprendizado por reforço será a abordagem seguida pelo trabalho, sendo detalhado a seguir.

2.4. Aprendizado por Reforço

O aprendizado por reforço (SUTTON, 1999) geralmente utiliza uma formalização de um Processo de Decisão de Markov (MDP – *Markov Decision Process*), em que o ambiente atual é representado por um estado s_t , com t representando um momento discreto de tempo no qual há interação do agente com o ambiente, $t = 0, 1, 2, \dots$. O estado deve idealmente conter todas as informações necessárias para que um agente possa se situar no mundo e escolher da melhor forma seus passos futuros.

Para cada estado, o agente seleciona uma ação a_t de uma lista de ações A . Cada ação mudará o ambiente estocasticamente para um novo estado s_{t+1} e resultará numa recompensa numérica r_t que indicará a qualidade daquela ação específica tomada naquele estado.

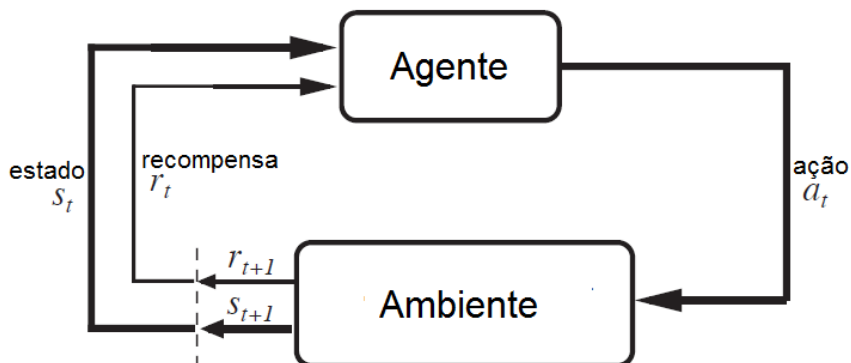
Mais formalmente, um MDP é uma tupla (S, A, P, R) , em que (DIMITRIADIS, 2009):

- $S = \{s_1, s_2, \dots, s_n\}$ representa o espaço finito de estados do ambiente, sendo n o número de possíveis estados;
- $A = \{a_1, a_2, \dots, a_m\}$ representa o espaço finito de ações que podem ser tomadas pelo agente, sendo m o número de possíveis ações;
- $P(s, a, s')$ é o modelo de transição de Markov, informando a possibilidade de chegar ao estado s' após executar a ação a no estado s ;
- $R(s, a)$ é a função de recompensa que indica a recompensa recebida pela execução da ação a no estado s .

A partir da recompensa recebida por cada entrada e no decorrer de seguidas execuções, o computador vai melhorando o seu conhecimento do mundo no qual o problema está inserido, devendo ser capaz de formular uma política (*policy*) definindo

quais ações ele acredita serem as melhores para os possíveis estados do ambiente. Geralmente, o objetivo do agente é escolher a ação que maximize o *retorno descontado esperado*: $E\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots\}$ (SUTTON, 1999), em que γ representa o fator de desconto, $0 \leq \gamma \leq 1$, indicando a importância dada às ações futuras. Na Figura 6, é ilustrado o *framework* básico do aprendizado por reforço.

Figura 6: *Framework* básico do RL. Adaptado de (SUTTON, 1999).



Como dito anteriormente, a representação do estado deve servir para que o agente seja capaz de entender da melhor maneira possível a situação atual do ambiente e seu posicionamento nele, sendo assim fundamental para a aprendizagem. Em geral, nos problemas de interesse para a IA, é impossível representar o estado com exatidão, pois o espaço de busca cresce exponencialmente, tornando proibitivo o aprendizado a partir das técnicas que utilizam a formalização de Markov, que tendem a possuir provas teóricas mais fortes de convergência para resultados ótimos (BUSONI et al., 2011). Desta forma, uma das tarefas importantes para a utilização destas técnicas costuma ser a definição de uma abstração que seja capaz de representar o estado de forma simplificada, diminuindo o espaço de busca, porém mantendo uma dose adequada de fidelidade ao ambiente real para não prejudicar o aprendizado.

No RL, como o agente deseja maximizar a recompensa recebida, é importante que ele dê preferência a ações tomadas previamente com bons resultados. Apesar disso, também é necessário que o agente investigue outras ações de forma a explorar ao máximo o espaço de busca e poder encontrar novas possibilidades que sejam melhores do que as previamente conhecidas. Este é um dilema fundamental do RL e é preciso que o agente seja capaz de realizar ambas as tarefas para obter sucesso (SUTTON, 1999).

Uma técnica comumente utilizada para ajudar neste balanceamento é a ϵ -greedy (WATKINS, 1989), na qual o valor de ϵ , $0 \leq \epsilon \leq 1$, representa a possibilidade de escolher uma ação aleatória no estado atual. Desta forma, a utilização de ϵ -greedy com $\epsilon = 0$ resultará em uma técnica totalmente gananciosa, escolhendo sempre a melhor ação disponível e sem buscar alternativas, enquanto que a utilização de $\epsilon = 1$ resultará em uma técnica totalmente aleatória, sem aproveitar o conhecimento obtido anteriormente.

Os algoritmos de RL podem ser divididos entre os baseados em modelo (*model-based*) e livres de modelo (*model-free*). Os algoritmos baseados em modelo, como Dyna-Q (SUTTON, 1990) e R-Max (BRAFMAN; TENNENHOLTZ, 2002), tentam inicialmente obter um modelo confiável do ambiente do problema, identificando as relações existentes entre as ações tomadas e suas recompensas e, com o modelo construído, utilizam o mesmo para avaliar ações futuras e dar uma resposta para o agente. Os algoritmos livres de modelo, como Q-Learning (WATKINS, 1989) e SARSA (RUMMERY; NIRANJAN, 1994), pulam a etapa da modelagem e tentam fazer uma associação direta das melhores ações para cada estado.

Não há comprovação de superioridade de uma estratégia em relação à outra (KAELBLING et al., 1996). Acredita-se, porém, que em problemas dinâmicos e mais complexos, com diversos fatores influenciando nos resultados das ações, pode ser muito difícil e demorado criar um modelo preditivo eficaz (MATARIC, 1997). Por conta disto, técnicas livres de modelo são mais comumente utilizadas nestes casos. Em seguida, serão detalhadas as duas principais técnicas de RL livres de modelo.

2.4.1. Q-Learning

Q-Learning é uma técnica livre de modelo e é tido como o mais popular algoritmo de RL (BIANCHI, 2004). Entre os fatores fundamentais para a sua popularidade estão a facilidade de implementação e a sua aplicação com sucesso em muitos domínios. O Q-Learning comprovadamente converge para uma política ótima (i.e., que garante sempre a escolha da melhor ação para cada estado) sob condições apropriadas (SUTTON, 1999).

A técnica é nomeada a partir dos valores de pares estado-ação $Q(s, a)$ que são utilizados para representar o retorno esperado descontado da ação a no estado s . Este valor é atualizado de acordo com a equação (1) mostrada a seguir:

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)) \quad (1)$$

em que α , $0 \leq \alpha \leq 1$, representa a taxa de aprendizado do sistema, r representa a recompensa recebida e s' representa o novo estado do ambiente.

O algoritmo do Q-Learning é da forma:

1. Inicia $Q(s, a) = 0$ para todos os pares de estado-ação (s, a) ;
2. Observa o estado s_0 ;
3. Para cada $t = 0, 1, 2, \dots$:
 - 3.1. Seleciona e executa uma ação a_t ;
 - 3.2. Recebe recompensa r ;
 - 3.3. Observa o estado s_{t+1} ;
 - 3.4. Atualiza $Q(s_t, a_t)$.

2.4.2. SARSA

SARSA (*State-Action-Reward-State-Action*, ou *Estado-Ação-Recompensa-Estado-Ação*) é uma variação do Q-Learning que também utiliza os valores $Q(s, a)$ de pares de estado-ação para representar a utilidade das ações e mantém as mesmas regras de convergência para a política ótima.

No Q-Learning, o valor $\max_{a'} Q(s', a')$ é utilizado para atualizar $Q(s, a)$. Porém, há dúvidas se este valor provê a melhor estimativa do retorno esperado para o estado s , já que o valor máximo pode ser superestimado, principalmente nas execuções iniciais do sistema (RUMMERY; NIRANJAN, 1994). O SARSA busca resolver este problema ao oferecer uma atualização alternativa de $Q(s, a)$, mostrada na equação (2):

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma Q(s', a') - Q(s, a)) \quad (2)$$

em que a' é a ação a ser tomada na próxima iteração do sistema. Desta forma, é utilizado diretamente o valor $Q(s', a')$ referente à próxima ação a ser aplicada em vez de ser utilizado de forma gananciosa o valor máximo entre todos os pares de estado-ação existentes.

O algoritmo do SARSA é da forma:

1. Inicia $Q(s, a) = 0$ para todos os pares de estado-ação (s, a) ;
2. Observa o estado s_0 ;
3. Seleciona uma ação a_0 ;
4. Para cada $t = 0, 1, 2, \dots$:
 - 4.1 Executa a ação a_t ;
 - 4.2 Recebe recompensa r ;
 - 4.3 Observa o estado s_{t+1} ;
 - 4.4 Seleciona uma ação a_{t+1} ;
 - 4.5 Atualiza $Q(s_t, a_t)$.

Não há consenso quanto à melhor técnica entre Q-Learning e SARSA. Ambas são técnicas muito conhecidas da aprendizagem de máquina que já foram utilizadas com sucesso em diversos problemas de RL e são consideradas escolhas válidas para trabalhos na área.

3. TRABALHOS RELACIONADOS

Neste capítulo será apresentada uma revisão geral sobre alguns dos principais trabalhos da literatura relacionados à utilização de técnicas de IA e, mais especificamente, ML em jogos de estratégia em tempo real.

3.1. Motivação para Pesquisas na Área

Dois artigos, por Buro e Furtak (2003) e por Buro (2004), já mencionavam e detalhavam a importância da pesquisa de IA em jogos RTS. Os artigos citam o aumento de pesquisa de IA em tempo real tanto nos campos de jogos eletrônicos comerciais como nos campos de simulações militares e posicionam os jogos RTS como uma junção entre os dois campos, podendo ser considerados simulações simplificadas.

Os artigos falam da baixa qualidade das IAs presentes nos jogos RTS comerciais, mencionando algumas das possíveis razões para tal (e.g., ambientes muito grandes, com informação incompleta e ação rápida, enquanto que pesquisas de sucesso para IA em jogos costumam utilizar jogos lentos, baseados em turnos e com informação completa).

Os autores tratam também dos muitos problemas fundamentais de pesquisa em IA que estão presentes nos jogos RTS, mencionados na Seção 2.1.1, e afirmam que pesquisas de sucesso em jogos RTS acarretarão em um aumento da compreensão destes problemas fundamentais, não apenas trazendo considerável impacto para a área, como também possuindo aplicações em domínios externos.

Também é feita uma explicação do ORTS (Seção 2.2.1), mencionando a motivação para o seu desenvolvimento e os objetivos que tentaram ser alcançados com a criação do ambiente, voltado a aumentar a viabilidade das pesquisas de IA em jogos RTS.

3.2. Aprendizagem Evolucionária em Jogos RTS

Marc Ponsen e Pieter Spronck propuseram a utilização de aprendizagem evolucionária *offline* aplicada a IA em jogos RTS (PONSEN, 2004; PONSEN; SPRONCK, 2004). Inicialmente, é dada uma visão geral da situação da IA em jogos comerciais, incluindo o fato de que mais de 80% destes jogos utilizam técnicas determinísticas como *scripts* e máquinas de estados finitos, que necessitam de uma definição manual dos aspectos comportamentais do agente, obrigando o desenvolvedor a pensar em todas as situações e estados de jogo possíveis e como responder a eles.

A proposta visa utilizar algoritmos evolucionários, um grupo de algoritmos de otimização baseados em princípios da evolução biológica, para evoluir a utilização de *scripts* dinâmicos, uma técnica de aprendizagem baseada no aprendizado por reforço que forma um novo *script* para controlar o comportamento de um jogador a partir de uma série de regras pré-definidas, em que a chance de uma regra ser escolhida para formar o comportamento do jogador é baseada no valor de seu peso e os pesos das regras são modificados de acordo com os resultados de sua utilização em partidas.

Inicialmente, foi aplicada a técnica básica de *scripts* dinâmicos em diversas situações de combate no ambiente Wargus, obtendo resultados bons apenas nas situações mais simples. Após isto, os autores utilizaram os algoritmos evolucionários para descobrir novas estratégias, obtendo bons resultados em todas as situações testadas. Por fim, algumas das soluções obtidas pelos algoritmos evolucionários foram incluídas no conjunto de regras dos *scripts* dinâmicos, permitindo a formação de comportamentos mais sólidos para o jogador e também obtendo bons resultados nas situações testadas.

3.3. Raciocínio Baseado em Casos para Diferenciação entre Oponentes

O artigo de Aha, Molineaux e Ponsen (2005), parte de um trabalho anterior, que empregava algoritmos genéticos e *scripts* dinâmicos para gerar táticas contra adversários fixos (PONSEN; SPRONCK, 2004) e propõe a utilização de raciocínio

baseado em casos para evoluir o trabalho de forma a utilizar diferentes táticas para diferentes oponentes.

O trabalho também utiliza o Wargus e consiste em gravar os casos ocorridos durante a partida juntamente com variáveis representando a situação do jogo para descrever cada caso, avaliando os casos de acordo com a pontuação dos jogadores em diversos momentos da partida, lembrando uma abordagem de aprendizado por reforço.

A implementação utilizou o *middleware* TIELT (AHA; MOLINEAUX, 2004) para a integração da IA com o ambiente que, apesar de facilitar esta etapa, tem o ponto negativo de adicionar um grau de não-determinismo ao trabalho por conta de sua natureza *multithreaded* e de atrasos de comunicação, fazendo com que partidas disputadas entre adversários iguais e com mesmo estado inicial e sistema decisório possam ter resultados completamente diferentes, o que os autores justificam ao dizer que o mesmo ocorreria com partidas entre humanos, também não-determinísticos.

Os resultados preliminares foram positivos, com o agente derrotando seus oponentes em 80% dos jogos e conseguindo aprender mesmo disputando partidas contra adversários distintos, diferenciando um do outro, mas conseguindo utilizar o conhecimento previamente adquirido contra qualquer oponente. Apesar disto, os resultados são relativamente incertos por não demonstrarem o desempenho exato do agente quando disputando vários jogos seguidos contra oponentes alternados (KOK, 2008).

3.4. Frameworks para Aprendizagem em Jogos RTS

Eric Kok (2008) propôs a implementação de agentes com aprendizagem em jogos RTS a partir de uma plataforma para a criação de agentes inteligentes em situações gerais. O ambiente utilizado para a implementação foi o *Bos Wars*, um jogo RTS de código aberto.

O trabalho utilizou a plataforma 2pl da Universidade de Utrech, baseada na arquitetura BDI para agentes racionais, para a criação de agentes inteligentes. A ideia é ter um conjunto de regras definidas em *scripts* estáticos e fazer com que o agente

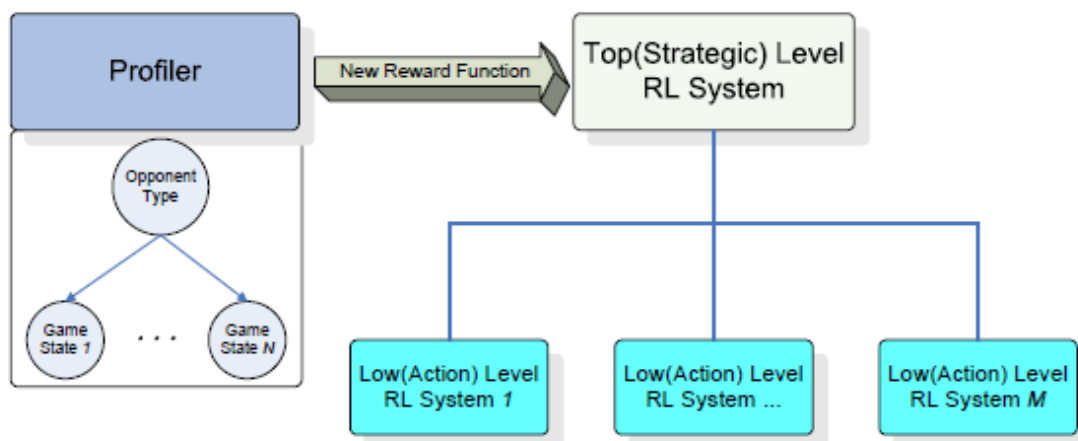
seleccione durante a partida as melhores regras a serem utilizadas a cada momento, de forma a obter a maior possibilidade de vitória.

O aprendizado foi implementado como um *framework* genérico dentro da plataforma 2pl, integrando os conceitos do aprendizado, como a recompensa recebida, às ações de um agente 2pl, como a crença do agente com relação à sua situação atual. O autor decidiu utilizar apenas recompensas ao final do jogo para uma vitória ou derrota. Desta forma, espera-se que o agente aprenda estratégias completas dentro de uma partida e não ações isoladas em situações específicas.

Foram aplicadas e testadas diferentes técnicas de aprendizado no *framework*, como a Monte Carlo e a de *scripts* dinâmicos, além de combinações entre elas. Foram realizados testes em situações diversas, inicialmente com implementações simples dos algoritmos para aprender uma estratégia geral, depois com conceitos mais sofisticados como melhores técnicas de escolha de ação e redução do espaço de busca, obtendo bons resultados na maioria dos casos contra diferentes *scripts* de IA estáticos. Por fim, são realizados testes contra inimigos com *scripts* dinâmicos na tentativa de encontrar estratégias vencedoras contra esses oponentes mais sofisticados. Após obter resultados piores nestes testes, o autor sugere possíveis alterações futuras que podem trazer melhorias de desempenho para o agente.

Outro trabalho que propõe um *framework* para a aprendizagem em jogos RTS foi realizado em 2009 (ANDERSEN et al., 2009). Nele, é proposto um *framework* com múltiplas camadas, possuindo uma camada *Profiler* para identificar o perfil do oponente, uma camada estratégica de alto nível que recebe recompensas baseadas no perfil encontrado pelo *Profiler* e diversas camadas de ações de baixo nível (Figura 7).

Figura 7: *Framework* proposto (ANDERSEN et al., 2009)



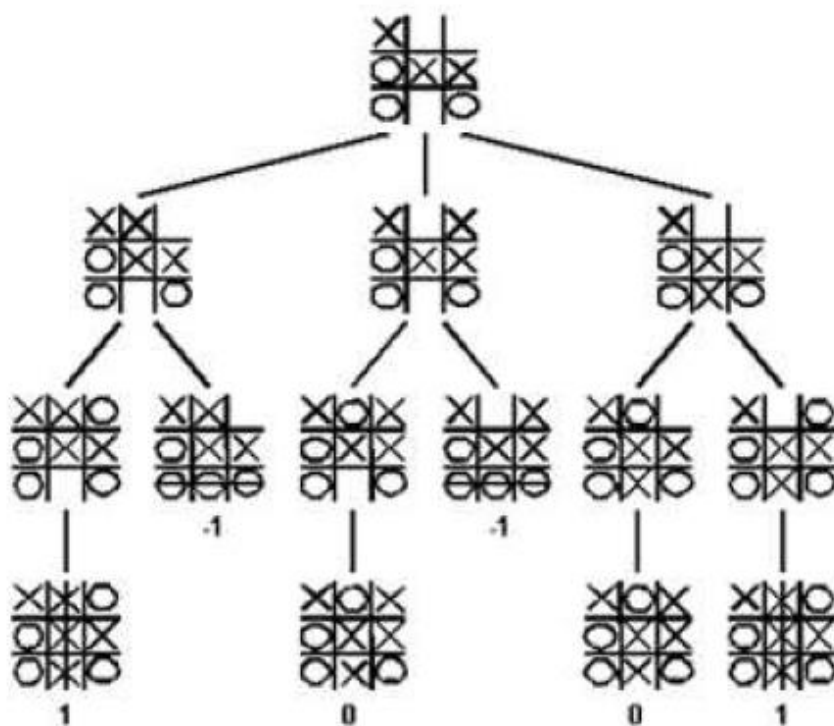
Os autores desenvolveram um jogo próprio chamado *Tank General* para validar o trabalho, com as características padrão de jogos RTS (e.g., múltiplos tipos distintos de unidades, recursos para obtenção de novas unidades, presença de *fog of war*). Foram definidos três possíveis perfis de oponentes: Agressivo, Defensivo e Focado em Recursos, com a utilização de um classificador bayesiano ingênuo para a classificação do oponente. Foram definidas quatro possíveis estratégias de alto nível: Explorar, Defender, Atacar Base Adversária e Atacar Recursos, com ações de baixo nível específicas para cada estratégia.

Foram testadas versões utilizando os algoritmos SARSA e Q-Learning contra oponentes estáticos com um dos três possíveis perfis definidos. Na maioria dos casos o agente utilizando o *framework* para aprendizado se saiu melhor do que o inimigo estático e o Q-Learning obteve resultados levemente melhores que o SARSA nestas situações. Após os resultados iniciais, o autor decidiu deixar de lado o SARSA e focar apenas no Q-Learning, em testes com agentes utilizando o *Profiler* e agentes que não utilizavam o *Profiler* contra os inimigos estáticos, que provaram que o agente com o classificador se sai melhor por possuir conhecimento sobre o adversário. Por fim, são comparados os desempenhos do agente utilizando o *framework* proposto com os de um agente utilizando uma técnica de aprendizado com única camada contra os oponentes estáticos, com o agente utilizando o *framework* de múltiplas camadas obtendo melhores resultados nos experimentos.

3.5. Utilização de Árvores de Jogos em Situações de Combate

Laursen e Nielsen (2005) investigaram a utilização de árvores de jogos para resolução de situações de combate em jogos RTS, no ambiente Wargus. A proposta visa calcular as possíveis situações futuras do jogo a partir do estado atual, considerando tanto as escolhas futuras do oponente quanto as escolhas futuras do próprio jogador, e escolher as ações que levem à melhor situação. Os autores citam a utilização com sucesso desta técnica em jogos por turnos, como o Xadrez e o Jogo da Velha (Figura 8).

Figura 8: Exemplo de árvore de jogo para uma situação do Jogo da Velha (LAURSEN; NIELSEN, 2005)



Como as árvores de jogos usuais baseiam-se na premissa de que cada jogador executa uma ação por vez e as ações levam a mesma quantidade de tempo, é preciso modificar as árvores de forma a poderem ser utilizadas em situações de combate em jogos RTS, nos quais as ações são simultâneas e podem ter durações distintas. Os autores propõem a utilização de árvores com marcas de tempo (*timestamped*), que armazenam o número de ciclos do jogo associado a cada estado da árvore, de forma que os filhos de determinado nó da árvore sempre terão uma marca de tempo maior que seu

pai e de acordo com o tempo necessário para completar a ação a qual se refere. Desta forma, se o estado atual possuir marca de tempo 100 e determinada ação levar 15 ciclos para completar, o filho relacionado a essa ação terá a marca 115.

Os autores também tratam de outros problemas para a utilização de árvores, como a inviabilidade de observar todos os possíveis estados do jogo na árvore, por conta do tamanho do espaço de busca, para o qual é sugerida a utilização de heurísticas para podar a árvore, e também a dificuldade de avaliar o valor de cada estado do ambiente, que eles buscam resolver por meio de avaliação automática com aprendizagem de máquina. Para a poda da árvore, os autores elaboraram diversas regras de comportamento e propõem algumas sequências de regras específicas com objetivos distintos (e.g., sequência focada em atacar os inimigos mais próximos e com menos vida), removendo da árvore as ações que não se encaixem na sequência atual.

Foram realizados vários experimentos no Wargus em situações diversas de combate e também com diferentes sequências de regras e estratégias de avaliação de estado atual. Os resultados demonstraram a dificuldade de criar boas estratégias para situações de combate, com os autores afirmando que a forma ótima de se portar em cada situação depende não apenas das suas próprias unidades, mas também da situação do oponente. Além disso, não foram encontradas sequências de regras ou estratégias de avaliação que fossem superiores em todas as situações, já que dependendo das características de cada experimento, determinadas estratégias se mostravam superiores. Apesar dos problemas, a técnica com árvore de jogos foi capaz de superar a IA padrão do Wargus, demonstrando sua capacidade de obter comportamentos consistentes e adequados em situações de combate.

3.6. SARSA e LSPI Aplicadas a Jogos RTS

Dimitriadis (2009) explorou a utilização de técnicas de aprendizado por reforço em jogos RTS, utilizando o jogo Glest como ambiente.

A representação de estado neste trabalho foi feita por meio de quatro variáveis discretas: número de unidades mortas pelo jogador, número de unidades produzidas pelo jogador, recursos colhidos pelo jogador e um contador de ciclos do jogo. A função

de recompensa foi baseada na fórmula para pontuação de cada jogador utilizada pelo próprio Glest, na forma de uma soma ponderada da diferença entre as variáveis de estado do agente contra as do inimigo (com exceção do contador de ciclos, que não é considerado), com peso maior para as unidades mortas, peso intermediário para as unidades produzidas e peso menor para os recursos colhidos.

Foram realizados experimentos com duas técnicas: SARSA (*online*) e LSPI (*offline*), em dois cenários do Glest: *Duel* (nível de dificuldade médio) e *Tough Duel* (nível de dificuldade alto). Tanto o SARSA quanto o LSPI conseguiram bons resultados no cenário mais simples, mas ambos foram incapazes de descobrir uma estratégia vencedora no cenário mais difícil, atribuído pelo autor às vantagens dadas ao jogador controlado pelo computador neste cenário. O autor também sugere que a falta de convergência para uma política estável de qualidade por parte das técnicas pode ter ocorrido pelo fato de que a representação de estado escolhida foi demasiadamente simplificada, ocasionando a perda de boa parte da informação referente ao estado do jogo.

3.7. Planejamento Aplicado a Situações de Combate

Uma alternativa para o controle pelo computador de unidades de um jogo RTS em situações de combate proposta por Balla e Fern (2009) é a utilização de UCT, uma técnica de planejamento utilizada previamente com sucesso no jogo de tabuleiro Go.

A proposta baseia-se no conceito de grupos de unidades. A cada momento decisório, são computados os grupos de unidades existentes e um planejamento é efetuado de forma *online* para definir ações abstratas para os grupos, que são executadas até a chegada do próximo momento decisório. Foram definidas duas ações abstratas: $Join(G)$, em que G é um conjunto de grupos, é uma ação que faz com que os grupos de G se movam até a localização centroide deles, de forma a se juntarem e formarem um grupo maior e $Attack(f, e)$, em que f é um grupo aliado e e um grupo inimigo, que ordena o grupo f a se mover em direção ao grupo e com a intenção de atacá-lo. Os autores não definiram comportamentos individuais para as unidades de cada grupo, de forma que as ações específicas de ataque são controladas pela IA padrão do jogo.

Os experimentos foram realizados em doze cenários diferentes no Wargus, com números diversos de unidades tanto inimigas quanto aliadas e posicionamento e agrupamento distintos destas unidades. Cada cenário foi simplificado de forma a conter apenas um tipo de unidade e também garantir a existência de ao menos uma estratégia vencedora. Foram testados dois tipos de UCT, um que tenta vencer o jogo o mais rapidamente possível e outro que tenta maximizar os pontos de vida das unidades aliadas durante os combates. O desempenho das duas técnicas de UCT foi comparado com o desempenho de outras estratégias de IA nos mesmos cenários, desde uma aleatória até uma controlada por um jogador humano, contra a IA padrão do jogo. Os resultados dos experimentos foram positivos e as técnicas de UCT foram capazes de encontrar estratégias vencedoras em todos os cenários, ultrapassando inclusive o jogador humano nas métricas de avaliação.

3.8. Campos Potenciais Aplicados a Situações de Combate

Outra alternativa proposta para situações de combate em jogos RTS foi a da utilização de campos potenciais multiagentes (MAPF – *Multi-Agent Potential Fields*), por Sandberg (2011), com a utilização de algoritmos genéticos para a seleção dos parâmetros dos campos que garantam a melhor qualidade da IA resultante.

A técnica de campos potenciais simula a existência de campos magnéticos, com campos de força positivos (forças de atração) e negativos (forças de repulsão) posicionados em diversos locais do ambiente. A ideia é que o agente presente no ambiente observe o estado atual e tente se mover para o local que resultar no maior valor positivo para ele.

A proposta foi validada no ambiente BWAPI. O autor aplicou a técnica para a facção *Terran*, utilizando oito campos potenciais diferentes (e.g., campo de atração para o centroide da equipe para manter o grupo coeso), com diferentes fórmulas de carga para cada campo. As fórmulas utilizam diversos parâmetros cujos valores são definidos pelos algoritmos genéticos.

Os experimentos foram efetuados em diversos cenários específicos de combate, primeiramente com execuções utilizando algoritmos genéticos para encontrar os

parâmetros ideais e depois em execuções com os parâmetros encontrados para avaliar a qualidade da IA resultante. O agente proposto foi capaz de derrotar a IA padrão do jogo nos cenários avaliados, tanto em situações em que havia grupos de apenas um tipo de unidade quanto em grupos com diversos tipos de unidades diferentes.

3.9. Autonomia Orientada a Objetivos em Situações de Combate

Jaidee, Muñoz-Avila e Aha (2011) sugeriram uma alternativa para a aprendizagem por reforço a partir da utilização de uma técnica de aprendizado baseada em casos para agentes com autonomia orientada a objetivos (GDA – *Goal-Driven Autonomy*) para situações de combate em jogos RTS, aplicada no ambiente Wargus.

A ideia de um agente GDA é executar determinada ação no ambiente com uma expectativa do resultado e, após isto, verificar as discrepâncias entre o estado resultante de fato com o que se esperava. O algoritmo de aprendizado no GDA deve ser capaz de aprender automaticamente os melhores objetivos durante o jogo a partir de expectativas e discrepâncias observadas por meio da análise de casos.

Os autores definiram cinco possíveis objetivos a serem perseguidos pelo agente e foram realizados testes em dois mapas, um médio e um grande, contra dois outros agentes, um utilizando Q-Learning e outro utilizando uma versão alternativa de aprendizado GDA. Foram obtidos resultados positivos contra o agente com Q-Learning, porém os resultados contra o agente com a versão alternativa do GDA foram mais equilibrados. Apesar disto, os autores citam que neste caso o agente com a técnica proposta ganhou as últimas nove partidas disputadas no mapa médio e as últimas sete partidas disputadas no mapa grande, o que sugere que o seu desempenho melhora com o tempo e que poderia obter resultados mais positivos com maior número de jogos de avaliação.

3.10. Aprendizado por Reforço em Situação de Combate

Aleksandar Micic, Davío Arnarsson e Vignir Jónsson (2011) analisaram a aplicação do aprendizado por reforço em uma situação de combate no jogo Starcraft: Brood War.

No trabalho, os autores fizeram validação em duas situações: *Dragoon x Zealot* e *4 Dragoons x 4 Zealots*. O *Dragoon* é uma unidade *ranged*, que ataca à distância, enquanto o *Zealot* é uma unidade que ataca apenas de perto. A ideia dos autores era fazer com que o computador, utilizando a técnica proposta, fosse capaz de aprender a controlar o *Dragoon* de forma a conseguir derrotar consistentemente o inimigo, mesmo sendo uma unidade mais fraca. Isto é possível utilizando a estratégia de *kiting*, ou seja, mantendo a distância para o oponente de forma a poder atacá-lo sem ser atacado.

O agente era modelado como uma máquina de estados finitos para representar seus comportamentos de baixo nível e a recompensa do aprendizado por reforço, baseada na vida restante das unidades, era distribuída ao fim de cada *frame* da partida. Os autores, em vez de desenvolver uma técnica genérica, fizeram modificações na técnica e nas ações disponíveis aos agentes para cada um dos cenários de forma a melhorar a efetividade do algoritmo. Ao fim do aprendizado, os resultados no primeiro teste chegaram próximos a 60% de vitória para o computador utilizando o aprendizado por reforço. No segundo teste, mais complexo e com múltiplos agentes, a taxa de vitórias ficou em 32%.

3.11. Análise Comparativa

Este capítulo apresentou alguns dos principais trabalhos relacionados à pesquisa de IA em jogos RTS. No Quadro 1 é apresentada uma comparação resumida entre os trabalhos apresentados.

Quadro 1: Comparativo entre os trabalhos apresentados

Trabalho	Inclui aprendizagem	Validado em combate	Ambiente	Técnica(s)
(BURO; FURTAK, 2003; BURO, 2004)	Não	Não	Nenhum	Nenhuma

(PONSEN; SPRONCK, 2004)	Sim, <i>offline</i>	Sim	Wargus	Algoritmos evolucionários aplicados a <i>scripts</i> dinâmicos
(AHA et al., 2005)	Sim, <i>offline</i>	Não	Wargus	Raciocínio baseado em casos
(KOK, 2008)	Sim, <i>online</i>	Não	<i>Bos Wars</i>	Técnicas de aprendizagem (e.g., Monte Carlo, <i>scripts</i> dinâmicos) aplicadas a agentes racionais
(ANDERSEN et al., 2009)	Sim, <i>online</i>	Não	<i>Tank General</i>	Técnicas de aprendizagem (e.g., Q-Learning, SARSA) aplicadas a um <i>framework</i> próprio
(LAURSEN; NIELSEN, 2005)	Sim, na avaliação de estados de jogo	Sim	Wargus	Árvores de jogos com marcas de tempo
(DIMITRIADIS, 2009)	Sim, <i>online</i> e <i>offline</i>	Não	Glest	LSPI e SARSA
(BALLA; FERN, 2009)	Não	Sim	Wargus	UCT
(SANDBERG, 2011)	Sim, apenas para ajustar parâmetros dos campos	Sim	BWAPI	Campos potenciais multiagentes
(JAIDEE et al., 2011)	Sim, <i>online</i>	Sim	Wargus	Aprendizado baseado em casos para agentes com autonomia orientada a objetivos
(MICIC et al., 2011)	Sim, <i>online</i>	Sim	BWAPI	SARSA, com variação para cada um dos cenários

A partir do quadro, é possível perceber a importância que tem sido dada à utilização de técnicas diversas de aprendizagem nos jogos RTS. Apesar desta importância e do crescimento recente dos trabalhos especificamente voltados a situações de combate, ainda é visível a pouca utilização de algumas técnicas clássicas de aprendizagem na definição de táticas de combate para situações gerais nos jogos do gênero.

É importante notar que, mesmo nos trabalhos que mencionam a utilização de SARSA, a técnica não foi aplicada extensivamente. Em um dos casos, SARSA foi utilizada apenas como um teste inicial antes de ser abandonada (ANDERSEN et al.,

2009) e, em outro, foi modelada com uma representação demasiadamente abstrata e não obteve bons resultados em uma situação geral de jogo (DIMITRIADIS, 2009).

No único trabalho avaliado que utiliza SARSA especificamente em situações de combate (MICIC et al., 2011), os autores escolheram por modificar a técnica de acordo com cada teste proposto, prejudicando a viabilidade da aplicação da técnica em situações gerais para um jogo comercial. Além disso, foram testados apenas dois cenários diferentes, ambos controlando o mesmo tipo de unidade, com resultados negativos no cenário mais complexo.

4. SOLUÇÃO PROPOSTA

O trabalho apresentado visa propor uma modelagem para a aplicação de uma técnica de aprendizagem de máquina *online* para controlar unidades de um jogo RTS durante o combate com outra facção (NETO; SIEBRA, 2012). Com isto, espera-se que o computador seja capaz de aprimorar sua estratégia ao longo do tempo, tornando-se cada vez mais difícil de ser derrotado. Outro resultado da utilização da ML deve ser a capacidade de adequar-se aos oponentes após disputar uma quantidade suficiente de partidas, evoluindo sua estratégia de forma a aumentar a sua imprevisibilidade e tentar combater o oponente de maneira mais eficiente.

Existem cinco aspectos fundamentais da proposta: o ambiente utilizado para implementação e validação, a representação do estado, as possíveis ações dos agentes e a técnica de aprendizagem de máquina escolhida, juntamente com seus parâmetros, que serão detalhados a seguir, além dos experimentos realizados para validação do trabalho, que serão detalhados no capítulo seguinte.

4.1. Ambiente

O ambiente escolhido para ser utilizado no trabalho foi o BWAPI. Como explicado na Seção 2.2.2, este ambiente possui diversas vantagens com relação às alternativas existentes, principalmente por permitir a inserção da IA diretamente no jogo *Starcraft: Brood War*, o mais popular RTS existente, além de ser um ambiente que vem sendo cada vez mais utilizado para a pesquisa acadêmica de IA em jogos. O fato de poder validar a IA em um jogo comercial também oferece a vantagem de poder testar a qualidade da técnica desenvolvida contra a IA padrão do jogo, que já está implementada.

Por padrão no BWAPI, cada IA deve ser desenvolvida como um módulo a ser compilado em um arquivo DLL que é injetado no jogo *Starcraft: Brood War* original e substitui as ações que seriam do jogador humano durante as partidas. A IA deve ser desenvolvida na linguagem C++, no ambiente Microsoft Visual C++ 2008 Express Edition para Windows (Em: <http://www.microsoft.com/en->

us/download/details.aspx?id=3092. Acesso em: fevereiro 2013>, a partir do *template* oferecido pelo BWAPI (Em: <<http://code.google.com/p/bwapi/downloads/list>>. Acesso em: fevereiro 2013).

A API (*Application Programming Interface*, ou Interface de Programação de Aplicativos) oferece acesso a todas as informações disponíveis para um jogador comum durante a partida e também permite que sejam executadas todas as ações pertinentes para cada unidade do jogo. Desta forma, acredita-se que o BWAPI ofereça as condições mais adequadas para o desenvolvimento do trabalho proposto.

4.2. Representação do Estado

A representação do estado é fundamental para o sucesso do trabalho, pois deve permitir que o agente compreenda a situação atual da partida e selecione as ações apropriadas, mas não pode ser exageradamente detalhada de forma a tornar-se grande demais e prejudicar a aprendizagem (Seção 2.4)

Não existe representação padrão de estados para a situação de combate em um jogo RTS. Desta forma, para a etapa atual do trabalho foi pensada uma representação, após avaliação do problema e das possíveis situações das unidades no ambiente, que deve ser capaz de prover ao agente conhecimento suficiente sobre uma batalha sem prejudicar seu processo de tomada de decisão.

A representação conta com seis variáveis possuindo três possíveis valores cada uma, totalizando 729 (3^6) possíveis estados nos quais o agente pode estar situado. As variáveis propostas para representar o estado, juntamente com seus possíveis valores, são as seguintes:

- **Vida:** representa a vida atual do agente. Possíveis valores: *baixa* (menos de 1/3 da vida restante), *média* (entre 1/3 e 2/3 da vida restante), *alta* (mais de 2/3 da vida restante);
- **Inimigos que podem ser atacados:** representa a quantidade de inimigos que podem ser atacados pelo agente a partir da sua posição atual, sem exigir movimentação extra. Possíveis valores: 0, 1, 2+;

- **Inimigos que podem atacar:** representa a quantidade de inimigos próximos que podem atacar o agente em sua posição atual. Possíveis valores: *0, 1, 2+*;
- **Inimigos atacando:** representa a quantidade de inimigos que estão atualmente atacando o agente. Possíveis valores: *0, 1, 2+*;
- **Aliados próximos:** representa a quantidade de aliados próximos ao agente. Os aliados próximos são aqueles que estão no raio de vizinhança do agente, definido pelo jogo de acordo com o alcance da arma da unidade. Possíveis valores: *0, 1, 2+*;
- **Vida média de inimigos próximos:** representa a média de vida dos inimigos que estão próximos ao agente. Possíveis valores: *baixa* (menos de 1/3 de vida média restante), *média* (entre 1/3 e 2/3 de vida média restante), *alta* (mais de 2/3 de vida média restante).

Com esta representação, acredita-se que o agente será capaz de obter um sentido geral da sua situação na batalha, tendo a capacidade de saber se está cercado por inimigos, se os inimigos estão em boa ou má situação, se possui aliados próximos que poderão ajudar, se está em risco de morrer em breve, etc.

4.3. Ações do Agente

A quantidade de possíveis ações a serem tomadas pelo agente também influencia no tamanho do espaço de busca. Desta forma, assim como na representação de estado, é preciso fazer uma abstração para diminuir a quantidade de ações existentes para que o espaço de busca não cresça demasiadamente.

Em jogos RTS, por exemplo, a quantidade de ações possíveis de serem tomadas em um dado momento foi estimada em 1500 (AHA et al., 2005), comparado a cerca de 30 para um jogo complexo como o xadrez. Apesar de alguns dos fatores levados em conta na estimativa não se aplicarem ao trabalho corrente, como ações para estruturas e unidades trabalhadoras, o número serve para dar uma ideia da inviabilidade de

considerar todas as possíveis ações de cada agente em um determinado momento do jogo na hora da tomada de decisão.

Para o trabalho atual, a proposta é a de ter cada ação definindo um comportamento geral, uma postura de alto nível para o agente, deixando a implementação de baixo nível de fora do aprendizado. As seguintes posturas foram definidas para os agentes, juntamente com as ações que as compõem:

- **Ataque:** o agente deve tomar uma postura ofensiva, buscando causar o máximo de dano possível ao oponente. Para isto, foram definidas as seguintes prioridades, em ordem de importância:
 1. Atacar inimigos próximos com menos vida;
 2. Atacar inimigos que estejam atacando o agente;
 3. Atacar inimigos que já estejam sendo atacados por aliados;
 4. Atacar inimigos mais próximos;
 5. Em último caso, na falta de inimigos próximos na posição atual, tenta deslocar-se de forma a estar em posição de ataque no próximo momento de decisão.
- **Suporte:** nesta postura o agente está mais preocupado em auxiliar seus aliados, seja compondo a ofensiva ao lado dos mesmos ou buscando melhorar a situação tática do jogo. Para compor esta postura, são definidas as seguintes prioridades:
 1. Busca aliados próximos com menos vida,
 - i. Caso o agente possua alguma ação de suporte (e.g., cura), executa a ação no aliado;
 - ii. Caso contrário, ataca inimigos próximos que tenham o aliado como alvo atual;
 2. Tenta mover-se em modo patrulha (i.e., pronto para atacar inimigos que se aproximem) para a posição centroide dos aliados mais próximos, de forma a aumentar a coesão do grupo;
 3. Na falta de qualquer aliado próximo, tenta deslocar-se de forma a estar em posição de ataque no próximo momento de decisão.

- **Recuo:** o agente assume uma postura de fuga para garantir sua sobrevivência. Para isto, abdica momentaneamente das ações ofensivas e tenta escapar para um local seguro. Mais detalhadamente:
 1. Busca a posição que possa ser alcançada dentro do intervalo de tempo atual que o deixe com o menor número de inimigos que possam atacá-lo, valorizando as posições mais distantes dos inimigos;

Estas abstrações, apesar de representarem apenas uma fração de todas as possíveis ações que um agente poderia tomar durante o jogo, buscam fornecer uma variedade suficiente de comportamentos para cada agente que, agindo em combinação, terão capacidade de derrotar os oponentes em combate.

4.4. Técnica - SARSA

A técnica escolhida para a implementação do trabalho foi a SARSA. Como explicado na Seção 2.4.2, esta é uma técnica baseada no Q-Learning que mantém as garantias de convergência para uma política ótima sob condições apropriadas. Além de ser uma técnica madura já aplicada com sucesso em diversos problemas, um fator importante para a sua escolha foi o fato desta técnica ainda não ter sido extensivamente explorada em situações de combate em jogos RTS (Seção 3.11).

Apesar de o ambiente de um jogo RTS ser multiagente em sua natureza, a proposta não envolve a utilização de técnicas de aprendizado por reforço multiagente (*Multi-Agent Reinforcement Learning* - MARL). Isto ocorre pois, apesar do aumento de pesquisas recentes, as técnicas de MARL em geral ainda possuem problemas sérios para sua utilização prática, como o aumento exponencial do espaço de busca de acordo com o número de agentes no sistema e a falta de garantia de convergência para políticas ótimas de comportamento.

O fato de não ser utilizada uma técnica específica de MARL não significa que fatores importantes de ambientes com múltiplos agentes, como a cooperação entre os agentes, devam ser descartados no trabalho. De fato, é fundamental que exista algum tipo de sentido de cooperação dos agentes que estão no mesmo time, pois se eles agirem

de maneira completamente independente é possível que executem ações com objetivos conflitantes entre si e prejudiciais ao time. Uma forma de valorizar o trabalho em equipe é a partir da função de recompensa. Para o trabalho, foi definida uma função de recompensa que se propõe a cumprir estes objetivos. A função segue a forma mostrada na equação (3):

$$(\sum_{i \in I} h_{i,t} - \sum_{i \in I} h_{i,t+1}) + (100 * n_{\Delta t}) - [(v_t - v_{t+1}) + (100 * m_{\Delta t})] \quad (3)$$

em que I representa o conjunto de inimigos próximos do agente no tempo t , $h_{i,t+1}$ representa a vida da unidade i no tempo $t+1$, $h_{i,t}$ representa a vida da unidade i no tempo t , $n_{\Delta t}$ representa a quantidade de inimigos mortos entre os tempos t e $t+1$, v_{t+1} representa a vida que o agente possui no tempo $t+1$, v_t representa a vida que o agente possuía no tempo t , $m_{\Delta t}$ vale 1 se o agente tiver morrido entre os tempos t e $t+1$ e, caso contrário, vale 0. Com esta função, valoriza-se a vida perdida por todos os inimigos próximos ao agente, contabilizando o dano que seja causado por aliados e dando bônus por inimigos mortos. Assim, se o agente tomar uma ação que possibilite que seus aliados próximos sejam mais efetivos e obtenham melhor desempenho, será recompensado por isto. A penalização se dá pela vida perdida pelo agente e também há uma perda maior caso o agente tenha morrido após tomar a ação.

O trabalho utiliza uma tabela Q única para todas as unidades do mesmo tipo (e.g., *Terran Marines*, *Protoss Zealots*). Isto é útil por permitir que cada agente utilize o conhecimento adquirido pelos outros agentes semelhantes a ele, acelerando o processo de aprendizado. Outra vantagem é que, como normalmente cada jogador utiliza em um ataque diversas unidades de cada tipo (e.g., um jogador *Zerg* pode atacar com um grupo de unidades *melee* como os *Zerglings* junto a um grupo de unidades *ranged* como os *Hydralisks*), serão aliviadas as necessidades de memória do sistema na maioria dos casos, já que não será preciso armazenar tabelas Q diferentes para cada unidade individual do jogo.

No BWAPI é possível controlar o jogo *frame* a *frame*. Um *frame* apenas é insuficiente para avaliar o resultado de uma ação, então o trabalho também propõe que a passagem de tempo t se dê após uma quantidade determinada de *frames*. A quantidade específica de *frames* tem que ser grande o bastante para dar tempo de avaliar uma ação

do agente, mas não pode ser grande demais a ponto de fazer com que o agente mantenha a mesma ação durante muito tempo e não reaja aos acontecimentos do jogo. O intervalo de tempo utilizado no trabalho foi de 20 *frames*. Desta forma, se o tempo t ocorrer no *frame* 80 do jogo, o tempo $t+1$ ocorrerá no *frame* 100.

A escolha de ações para os agentes se dá por meio do ε -greedy, com valor ε de 0,1. Para a técnica SARSA, o fator de desconto foi definido como 0,9 e a taxa de aprendizado como 0,1. Com estes valores, buscou-se um equilíbrio entre a necessidade de exploração do ambiente e de aproveitamento das ações já executadas anteriormente, com a devida importância dada às possíveis ações futuras do agente de acordo com sua escolha atual.

A aplicação da técnica foi pensada de maneira a utilizar conceitos conhecidos do RL, já aplicados com sucesso a diversos ambientes. Também houve a preocupação de escolher uma modelagem simples, tendo em vista o fato de que o tempo de processamento destinado à IA para combate não é exclusivo, mas dividido com o tempo cedido a todos os outros aspectos da simulação do jogo e, portanto, exige execução rápida da computação necessária.

5. EXPERIMENTOS

Os experimentos para validação do trabalho foram realizados no jogo *Starcraft: Brood War*, por meio de partidas disputadas entre a IA desenvolvida com as técnicas de ML contra a IA padrão do jogo. Estas partidas aconteceram em mapas diversos, específicos para situações de combate (i.e., que não possuem base ou recursos para serem coletados pelos jogadores) e com configurações distintas de formação de equipes tanto da IA proposta quanto do adversário.

O objetivo principal dos testes foi mostrar que a técnica proposta é capaz de evoluir o comportamento do computador, inicialmente sem nenhum conhecimento específico do jogo, na situação de combate de forma a permitir que, ao longo do tempo, ele se torne capaz de derrotar seu oponente consistentemente e em várias situações distintas. Quando possível, também buscou-se realizar testes similares aos aplicados em outros trabalhos relacionados, de forma a prover uma base de comparação para os resultados obtidos.

Em todos os testes, foram disputadas 5 baterias de 100 partidas. Ou seja, o aprendizado do agente ocorre no decorrer de 100 partidas e, ao final, as tabelas de conhecimento são reiniciadas e o processo recomeça. Com isto, espera-se obter resultados apropriados com relação ao conhecimento obtido pelo agente, diminuindo a prevalência de possíveis desvios estatísticos.

5.1. Teste 1 - *Protoss Dragoon x Protoss Zealot*

Este teste é similar ao descrito por Micic et al. (2011), onde o agente modelado controla uma unidade *Dragoon* da raça *Protoss*, e o inimigo controla a unidade *Zealot*, da mesma raça. O *Zealot* é uma unidade mais forte e com maior capacidade de ataque do que o *Dragoon*, mas tem a desvantagem de poder atacar apenas ao estar próximo ao oponente. Como explicado no trabalho, este teste exige que o agente aprenda a executar uma estratégia de *kiting*, ou seja, já que é impossível o *Dragoon* vencer em uma disputa cara a cara, é necessário posicioná-lo de forma a distanciar-se do oponente e atacar à distância, buscando sempre abrir espaço quando o *Zealot* adversário se aproximar.

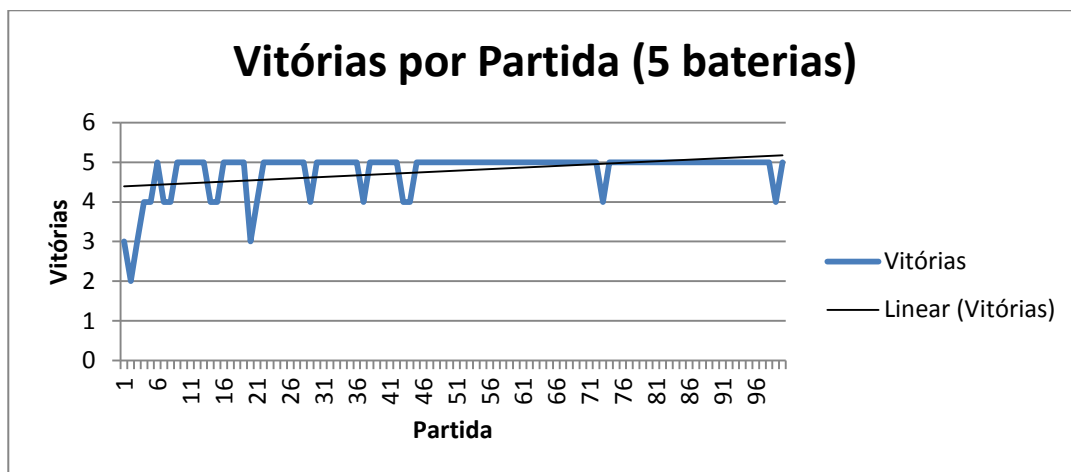
Tabela 1: Resultados do Teste 1

Bateria	Vitórias após 20 jogos (%)	Vitórias após 50 jogos (%)	Vitórias após 100 jogos (%)
1	90	94	96
2	90	94	96
3	85	94	97
4	85	92	96
5	75	86	93

Na Tabela 1 é apresentada a porcentagem de vitórias obtidas pelo agente após 20, 50 e 100 jogos disputados, para as baterias de teste realizadas. É possível perceber que, nesta situação relativamente simples, o agente é capaz de aprender muito rapidamente sobre a melhor estratégia para derrotar o oponente, obtendo bons resultados mesmo após apenas 20 jogos disputados. Em todas as baterias, o agente provou-se capaz de, mesmo após obter bons resultados, continuar melhorando seu desempenho, e foi verificado um aumento na porcentagem de vitórias com o aumento do número de partidas disputadas, configurando um aprendizado contínuo do agente nesta situação específica de jogo. Em todos os casos, o agente conseguiu superar os resultados obtidos por Micic et al. (2011) que, no melhor caso, alcançou 72% de vitórias na mesma situação.

A Figura 9 mostra a quantidade total de vitórias obtidas pelo agente em cada partida de índice x para todas as baterias. Assim, se em todas as cinco baterias disputadas o agente perder a primeira partida, o valor y da abscissa na posição 1 do gráfico será zero. Se nas cinco baterias o agente vencer três vezes a primeira partida, o valor y será três, e assim por diante para todas as cem partidas disputadas em cada bateria.

Figura 9: Vitórias por Partida no Teste 1



No gráfico é possível perceber que, em pouco tempo, o agente passa a vencer constantemente todas ou quase todas as partidas de mesmo índice nas cinco baterias disputadas e a tendência linear demonstra a melhoria de desempenho ao longo das partidas disputadas pelo agente, comprovando a eficiente aplicação do conhecimento obtido.

5.2. Teste 2 - 4 Protoss Dragoons x 4 Protoss Zealots

Este teste, também realizado em (MICIC et al., 2011), é similar ao anterior, mas agora é necessária a cooperação entre os agentes da mesma equipe, de forma a não apenas buscar as ações que os ajudem a afastar-se dos inimigos para atacá-los com segurança, mas também as que auxiliem seus companheiros de equipe a alcançarem os objetivos. Neste teste, espera-se mostrar que a técnica permite que o aprendizado ocorra não apenas em situações simples, com um único agente, mas também em situações mais complexas, como as que normalmente existirão nas partidas reais.

Na Tabela 2 são apresentados os resultados obtidos nesta situação. Pelos números apresentados, é possível perceber que, ao aumentar a complexidade da situação na qual o agente é colocado, também aumenta a dificuldade para o agente encontrar estratégias adequadas para seu comportamento, o que pode fazer com que o aprendizado demore mais a ocorrer e, por consequência, o agente demore mais a obter resultados positivos. Apesar disto, ainda houve aprendizado relativamente rápido, com o agente obtendo mais de 50% de vitórias após 20 jogos em quatro baterias. Em todos os casos

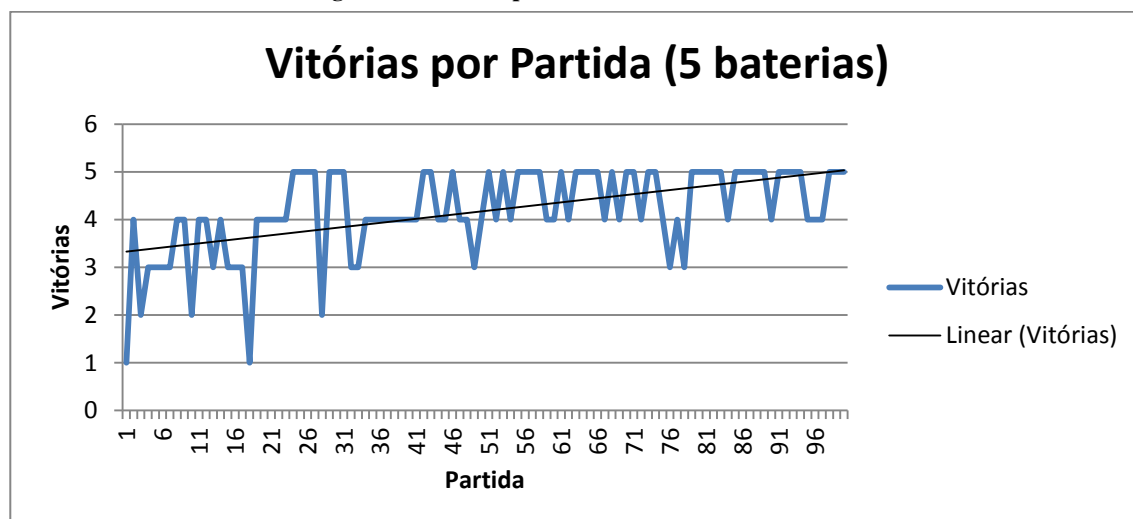
houve melhoria de desempenho com o aumento do número de jogos e, após os 100 jogos, o agente sempre obteve retrospecto positivo de vitórias. Assim como na situação anterior, o agente conseguiu superar em todos os casos os resultados obtidos por Micic et al. (2011) que, mesmo após modificar sua modelagem para adequar-se a esta situação, alcançou apenas 32% de vitórias.

Tabela 2: Resultados do Teste 2

Bateria	Vitórias após 20 jogos (%)	Vitórias após 50 jogos (%)	Vitórias após 100 jogos (%)
1	35	62	76
2	90	92	93
3	65	78	88
4	60	68	77
5	60	74	84

A Figura 10 apresenta o gráfico de vitórias obtidas pelo agente em cada partida disputada e mostra que neste caso também há tendência de aumento de vitórias do agente com o aumento do número de partidas disputadas.

Figura 10: Vitórias por Partida no Teste 2



5.3. Teste 3 - 3 Zerg Hydralisks x 3 Terran Vultures

Este teste busca representar uma situação equilibrada de combate. Os *Hydralisks* da raça *Zerg* são unidades muito similares aos *Vultures* da raça *Terran*. Ambas as unidades são capazes de atacar à distância, sendo o alcance do *Vulture* levemente maior que o do *Hydralisk*. As duas unidades possuem a mesma quantidade de pontos de vida e possuem DPS (Dano por Segundo) similar, sendo que cada ataque do *Hydralisk* causa metade do dano de um ataque do *Vulture*, mas o *Hydralisk* executa o dobro de ataques que o *Vulture* na mesma quantidade de tempo. Para vencer a partida, é necessário coordenar inteligentemente a movimentação e escolha de alvo dos agentes, já que devido ao equilíbrio entre as equipes, qualquer erro pode levar à derrota nesta situação.

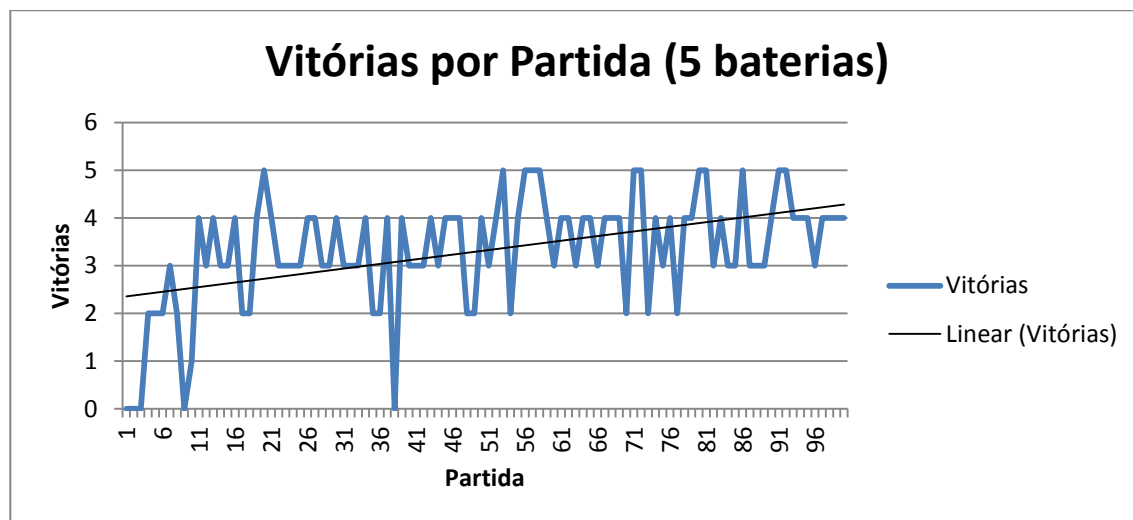
Nesta situação, como pode ser visto na Tabela 3, mesmo com equipes totalmente equilibradas, o agente foi capaz de obter aprendizado suficiente para, ao fim dos 100 jogos, vencer a maioria das partidas contra seu oponente em todas as baterias. Assim como nos casos anteriores, todas as baterias apresentaram melhoria de desempenho do agente com o aumento do número de partidas, demonstrando o aprendizado contínuo obtido pelo mesmo. Ao fim dos 100 jogos, o agente obteve média positiva de 66,4% de vitórias.

Tabela 3: Resultados do Teste 3

Bateria	Vitórias após 20 jogos (%)	Vitórias após 50 jogos (%)	Vitórias após 100 jogos (%)
1	40	46	54
2	50	68	71
3	65	70	75
4	45	58	76
5	30	40	56

Na Figura 11 mais uma vez vê-se claramente a tendência positiva de resultados do agente, com seu desempenho tornando-se superior nas partidas posteriores de cada bateria.

Figura 11: Vitórias por Partida no Teste 3



5.4. Teste 4 - 1 *Protoss Dragoon* + 1 *Protoss Zealot* x 3 *Terran Marines* + 1 *Terran Medic*

Esta situação coloca duas unidades fortes da raça *Protoss*, o *Dragoon* e o *Zealot*, contra três membros da unidade fundamental de combate da raça *Terran*, os *Marines*, juntamente com um *Medic*. O *Dragoon* e o *Zealot* são, em geral, capazes de derrotar facilmente três *Marines*. Mas, nesta situação, a presença do *Medic* exige uma mudança de estratégia. Por conta da capacidade que o *Medic* possui de curar seus aliados, é necessário que o agente controlando o *Dragoon* e o *Zealot* aprenda que, para ter alguma chance de vitória, é preciso deixar de lado o ataque às unidades *Marine* ofensivas no início da partida e focar todo o ataque no *Medic*, mesmo deixando suas unidades temporariamente expostas. Apenas após eliminar o *Medic*, deve-se atacar as outras unidades.

A Tabela 4 mostra que o agente também foi capaz de melhorar seu desempenho no decorrer das partidas nesta situação. Apesar de em duas das baterias o agente ter finalizado os 100 jogos com retrospecto levemente negativo, a média de vitórias ao final das baterias foi positiva, de 56,2%. Pelo fato da situação ser um pouco mais complexa,

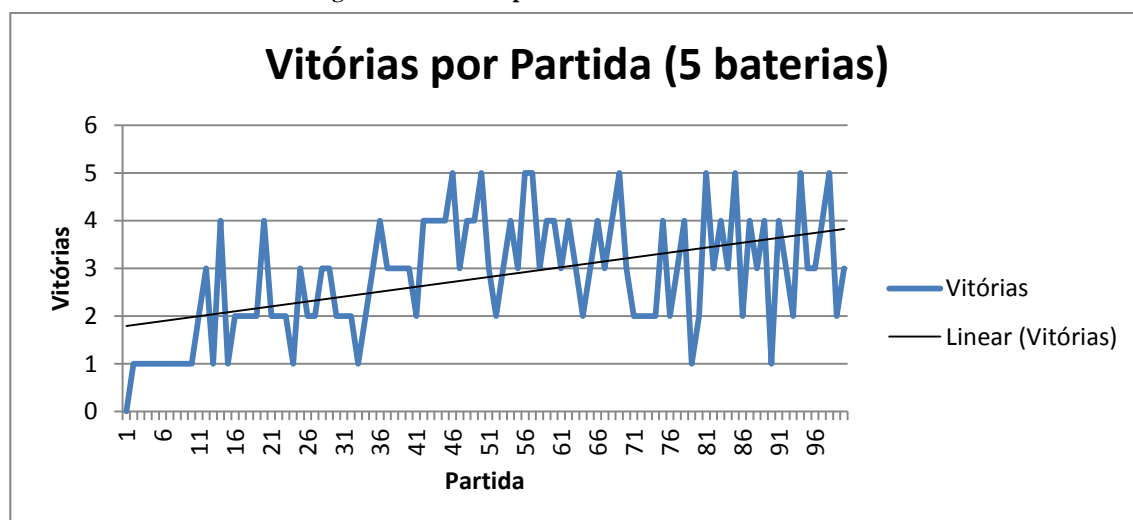
verificou-se em alguns casos a necessidade de um maior número de partidas para que o agente fosse capaz de desenvolver uma estratégia mais eficiente. Mesmo assim, em todas as baterias verificou-se que o agente foi eventualmente capaz de aprender a estratégia correta para a situação (atacar imediatamente a unidade de cura) e manteve-se o fato de conseguir obter sempre desempenhos superiores em todas as baterias com o aumento do número de partidas disputadas.

Tabela 4: Resultados do Teste 4

Bateria	Vitórias após 20 jogos (%)	Vitórias após 50 jogos (%)	Vitórias após 100 jogos (%)
1	20	28	46
2	50	52	58
3	35	58	60
4	40	62	69
5	15	38	48

Assim como nos casos anteriores, o gráfico de vitórias por partida da Figura 12 é positivo, mostrando o crescimento no número de vitórias do agente com o decorrer das partidas, com linha de tendência crescente.

Figura 12: Vitórias por Partida no Teste 4



5.5. Teste 5 - 2 *Terran Vultures* + 1 *Terran Marine* x 2 *Zerg Hydralisks* + 2 *Zerg Zerglings*

Este teste visa mostrar que a modelagem também permite o aprendizado em situações com agentes de diferentes tipos. Como explicado no Teste 3, os *Vultures* são basicamente equivalentes aos *Hydralisks*. O *Marine* é uma unidade mais forte do que um *Zergling* mas, contra dois *Zerglings*, é considerado inferior. Assim, o agente controlando as unidades *Terran* está em desvantagem inicial e, para vencer o combate, deve manejar suas unidades eficientemente de forma cooperativa para superar o maior poder ofensivo do inimigo.

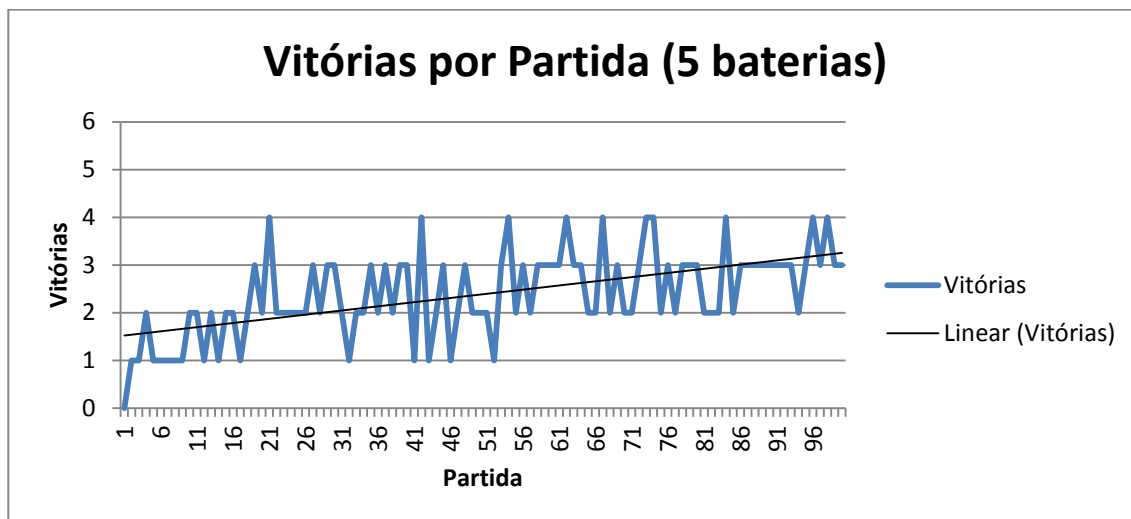
Os resultados apresentados na Tabela 5 mostram que esta situação é de fato a mais complicada para o agente. Apesar disto, ainda foi possível perceber a evolução do agente visto que, assim como nos casos anteriores, em todas as baterias o agente melhorou seu desempenho com o aumento do número de partidas disputadas. Mesmo controlando uma equipe teoricamente mais fraca, o agente conseguiu aproximar-se de 50% de vitórias após 100 partidas em todas as baterias, chegando inclusive a ultrapassar este valor em uma delas. Ao final, o agente conseguiu uma média de 47,8% de vitórias após 100 jogos, com visível evolução de comportamento nas partidas finais disputadas, o que leva a crer que o desempenho seria ainda melhor se mais partidas fossem utilizadas para o aprendizado.

Tabela 5: Resultados do Teste 5

Bateria	Vitórias após 20 jogos (%)	Vitórias após 50 jogos (%)	Vitórias após 100 jogos (%)
1	30	40	48
2	35	42	48
3	30	38	46
4	25	40	52
5	25	36	45

Mesmo neste teste mais complexo, com mais dificuldades para o agente, o gráfico da Figura 13 mostra a repetição do ocorrido nos testes anteriores, com linha de tendência positiva indicando melhoria de desempenho no decorrer das partidas disputadas, comprovando a capacidade de aprendizado também para esta situação específica.

Figura 13: Vitórias por Partida no Teste 5



5.6. Resumo

Na Tabela 6 é apresentado um resumo dos resultados obtidos em todos os testes efetuados. Nela, é mostrada a média de vitórias obtidas em cada um dos testes realizados, após 20, 50 e 100 partidas disputadas.

Pelos resultados, é possível observar que, em todos os testes, o agente foi capaz de melhorar seu desempenho ao longo do tempo, obtendo desempenho sempre melhor com o aumento do número de partidas disputadas, o que demonstra a ocorrência do aprendizado, conforme esperado.

Tabela 6: Resumo de resultados dos testes realizados

Teste	Média de vitórias após 20 jogos (%)	Média de vitórias após 50 jogos (%)	Média de vitórias após 100 jogos (%)
1	85	92	95,6
2	62	74,8	83,6
3	46	56,4	66,4
4	32	47,6	56,2
5	29	39,2	47,8

Na validação, buscou-se realizar testes representando diversas situações diferentes de jogo, com o agente controlando unidades das mais distintas de todas as raças disponíveis no jogo *Starcraft: Brood War*, de forma a comprovar a viabilidade da aplicação de técnicas de aprendizagem a situações reais de combate em jogos de Estratégia em Tempo Real. Pelo quadro percebe-se que, com os testes realizados, foi possível mostrar que a modelagem para técnica proposta é capaz de eventualmente gerar agentes mais eficientes do que os utilizados por padrão no jogo (i.e., capazes de derrotá-los consistentemente) nas situações avaliadas, e também de melhorar o comportamento de tais agentes a partir das partidas disputadas, tornando a IA mais atrativa para os jogadores do que uma IA estática.

6. CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma proposta de modelagem para a utilização de uma técnica da aprendizagem de máquina aplicada a situações de combate em jogos RTS. A solução desenvolvida utiliza SARSA, uma técnica clássica do aprendizado por reforço, de forma a permitir que o computador seja capaz de adequar as suas táticas de combate de acordo com as situações que se apresentem e dos adversários que enfrente, aprimorando a sua habilidade a cada partida disputada. Assim, torna-se possível criar uma IA mais adaptável e atrativa de ser enfrentada pela perspectiva do jogador humano. O trabalho foi desenvolvido no ambiente BWAPI e integrado diretamente no jogo comercial *Starcraft: Brood War*. O ambiente foi escolhido de forma a servir como representação mais adequada para a demonstração da viabilidade da técnica proposta em situações reais.

O trabalho abordou, além da utilização da técnica de aprendizagem de máquina em si, os outros fatores necessários à implementação com sucesso da proposta, como o ambiente a ser utilizado, a representação do estado e as ações disponíveis para cada agente, explicando cada um destes pontos e apresentando possíveis soluções para esses.

A proposta apresentou os conceitos fundamentais de inteligência artificial, aprendizagem de máquina e por reforço. Foi mostrada também uma visão geral da situação da IA em jogos eletrônicos e, mais especificamente, em jogos RTS, e das pesquisas que estão sendo realizadas na área, tratando ainda de algumas das principais questões e dificuldades relacionadas à utilização de técnicas de IA nestes jogos. Para isto, foi realizada uma extensa pesquisa bibliográfica para a obtenção de conhecimento com relação ao estado da arte da IA em jogos.

Foram realizados testes em cinco diferentes cenários de combate. Em todos os casos, verificou-se a ocorrência do aprendizado por parte do agente, que foi sempre capaz de melhorar seu comportamento ao longo do tempo, de forma a tornar-se mais eficiente nas situações apresentadas. Com isto, mostrou-se que é possível aplicar técnicas clássicas do aprendizado por reforço ao problema do combate em um jogo RTS comercial com resultados positivos a partir da modelagem adequada do problema.

Espera-se que o trabalho ajude a mostrar que há viabilidade na utilização de técnicas clássicas da IA e ML em jogos eletrônicos comerciais, que possam aumentar a

qualidade do jogo e a percepção de valor dos jogadores. O fato do problema escolhido – combate em jogos RTS – possuir muitas das características mais complexas que os jogos eletrônicos podem apresentar para a IA (e.g., ambiente estocástico e com inúmeros possíveis estados, ambiente com múltiplos agentes trabalhando tanto em cooperação quanto em competição, informação incompleta para o agente), serve para aumentar a importância do trabalho e demonstra a necessidade de ampliar os esforços na utilização deste tipo de técnica a problemas similares.

É provável que, para que seja utilizada de fato em jogos comerciais, a aprendizagem de máquina deva poder ser aplicada a partidas completas. Deste forma, para trabalhos futuros, uma proposta é a de expandir o aprendizado para tratar de todas as situações em um jogo RTS, e não apenas ao combate especificamente. Assim, um agente poderia disputar uma partida inteira contra seu oponente e aprender as melhores ações a tomar com relação à coleta de recursos, pesquisa tecnológica, combate, estratégia geral de jogo, entre outros. O maior desafio neste caso será o de modelar o ambiente do jogo de forma a permitir que o agente possa se situar em todas estas situações, sem que o espaço de busca torne-se proibitivamente grande de forma a impedir o aprendizado.

7. REFERÊNCIAS BIBLIOGRÁFICAS

2009 ORTS RTS Game AI Competition. Disponível em: <<http://skatgame.net/mburo/orts/AIIDE09/>>. Acesso em: Fevereiro 2013.

ABBEEL, P.; COATES, A.; NG, A.Y. “**Autonomous Helicopter Aerobatics through Apprenticeship Learning**”. The International Journal of Robotics Research, 2010. DOI: 10.1177/0278364910371999.

AHA, D. W.; MOLINEAUX, M. “**Integrating Learning in Interactive Gaming Simulators**”. Challenges in Game AI: Papers of the AAAI’04 Workshop (Technical Report WS-04-04), 2004.

AHA, D. W.; MOLINEAUX, M.; PONSEN, M. “**Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game**”. Proceedings of the Sixth International Conference on Case-Based Reasoning, pp. 5-20, 2005.

ANDERSEN, K. T.; ZENG, Y.; CHRISTENSEN, D. D.; TRAN, D. “**Experiments with Online Reinforcement Learning in Real-Time Strategy Games**”. Applied Artificial Intelligence, Vol. 23, pp. 855-871, 2009.

BALLA, R.-K.; FERN, A. “**UCT for Tactical Assault Planning in Real-Time Strategy Games**”. Proceedings of the International Joint Conference on Artificial Intelligence, pp. 40-45, 2009.

BIANCHI, R. A. da C. “**Uso de Heurísticas para a Aceleração do Aprendizado por Reforço**”. Tese de Doutorado, Escola Politécnica da Universidade de São Paulo, Brasil, 2004.

BRAFMAN, R. I.; TENNENHOLTZ, M. “**R-Max – A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning**”. Journal of Machine Learning Research, Vol. 3, pp. 213-231, 2002.

BURO, M. “**Call for AI Research in RTS Games**”. AI in Games Workshop, AAAI, 2004.

BURO, M. “**ORTS: A Hack-Free RTS Game Environment**”. Proceedings of the International Computers and Games Conference, pp. 280-291, 2002.

BURO, M.; FURTAK, T. “**RTS Games as Test-Bed for Real-Time Research**”. Invited Paper at the Workshop on Game AI, JCIS, pp. 481-484, 2003.

BUSONI, L.; ERNST, D.; DE SCHUTTER, B.; BABUSKA, R. “**Approximate Reinforcement Learning: An Overview**”. IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011.

BWAPI - An API for interacting with Starcraft: Broodwar (1.16.1). Disponível em: <<http://code.google.com/p/bwapi/>>. Acesso em: Fevereiro 2013.

BWAPI - Competitions held for BWAPI. Disponível em: <<http://code.google.com/p/bwapi/wiki/Competitions>>. Acesso em: Fevereiro 2013.

BWAPI – Downloads. Disponível em: <<http://code.google.com/p/bwapi/downloads/list>>. Acesso em: Fevereiro 2013.

CHURCHILL, D.; BURO, M. “**Build Order Optimization in StarCraft**”. AI and Interactive Digital Entertainment Conference (AIIDE), 2011.

DIMITRIADIS, V. K. “**Reinforcement Learning in Real Time Strategy Games Case Study on the Free Software Game Glest**”. Technical University of Crete, Greece, 2009.

EL RHALIBI, A.; WONG, K. W.; PRICE, M. “**Artificial Intelligence for Computer Games**”. International Journal of Computer Games Technology, 2009.

ENTERTAINMENT SOFTWARE ASSOCIATION (ESA). “**2011 Essential Facts About the Computer and Video Game Industry**”. 2011.

GRAFT, K. “**Blizzard Confirms One "Frontline Release" for '09**”. Disponível em: <<http://www.edge-online.com/news/blizzard-confirms-one-frontline-release-09>>. Acesso em: Fevereiro 2013.

HUHH, J. “**Culture and Business of PC Bangs in Korea**”. Games and Culture 3(1), pp.26-37, 2007. DOI: 10.1177/1555412007309525.

JAIDEE, U.; MUÑOZ-AVILA, H.; AHA, D. W. “**Case-Based Learning in Goal-Driven Autonomy Agents for Real-Time Strategy Combat Tasks**”. Case-Based Reasoning in Computer Games: Papers from the ICCBR Workshop, 2011.

JONSSON, B. “**Representing Uncertainty in RTS Games**”. M.Sc. Project Report. Reykjavik University, Iceland, 2012.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. “**Reinforcement Learning: A Survey**”. Journal of Artificial Intelligence Research, Vol. 4, Issue 1, pp. 237-285, 1996.

KOK, ERIC. “**Adaptive Reinforcement Learning Agents in RTS Games**”. M.Sc. Thesis. University Utrecht, Netherlands, 2008.

- LAIRD, J. E.; VAN LENT, M. **“Human-Level AI’s Killer Application: Computer Game AI”**. Proceedings of AAAI 2000 Fall Symposium on Simulating Human Agents, pp. 80-87, 2000.
- LAURSEN, R.; NIELSEN, D. **“Investigating Small Scale Combat Situations in Real Time Strategy Computer Games”**. M.Sc. Thesis. University of Aarhus, Denmark, 2005.
- LICHOCKI, P. **“Evolving Players for a Real-Time Strategy Game Using Gene Expression Programming”**. M.Sc. Thesis. Poznan University of Technology, Poland, 2008.
- MATARIC, M. J. **“Reinforcement Learning in the Multi-Robot Domain”**. Autonomous Robots, Vol. 4, pp. 73-83, 1997.
- MATEAS, M. **“Expressive AI: Games and Artificial Intelligence”**. Level Up: Digital Games Research Conference, 2003.
- MICIC, A.; ARNARSSON, D.; JÓNSSON, V. **“Developing Game AI for the Real-Time Strategy Game Starcraft”**. Technical Report, Reykjavik University, 2011.
- MITCHELL, T; **“The Discipline of Machine Learning”**. Machine Learning Department technical report CMU-ML-06-108, Carnegie Mellon University, 2006.
- NAREYEK, A. **“AI in Computer Games”**. Queue – Game Development Magazine, Vol. 1, Issue 10, pp. 58-65, 2004.
- NETO, G.; SIEBRA, C. A. **“Uma Proposta de Modelagem para uso do Aprendizado por Reforço na Definição de Táticas de Combate em Jogos de Estratégia em Tempo Real”**. XI Simpósio Brasileiro de Jogos e Entretenimento Digital, 2012.
- ORTS Homepage. Disponível em: <<http://skatgame.net/mburo/orts/>>. Acesso em: Fevereiro 2013.
- PÉREZ, A. U. **“Multi-Reactive Planning for Real-Time Strategy Games”**. M.Sc. Report. Universitat Autònoma de Barcelona, Spain, 2011.
- PONSEN, M. **“Improving Adaptive Game AI with Evolutionary Learning”**. M.Sc. Thesis. Delft University of Technology, Netherlands, 2004.
- PONSEN, M.; SPRONCK, P. **“Improving Adaptive Game AI with Evolutionary Learning”**. Computer Games: Artificial Intelligence, Design and Education, pp. 389-396, 2004.
- PONSEN, M.; URBAN-LEE, S.; MUÑOZ-AVILA, H.; AHA, D.; MOLINEAUX, M. **“Stratagus: An Open-Source Game Engine for Research in Real-Time Strategy Games”**. IJCAI Workshop on Reasoning Representation and Learning in Computer Games, 2005.

RORMARK, R. **“Thanatos – A Learning Game AI”**. M.Sc. Thesis. University of Oslo, Norway, 2009.

RUMMERY, G. A.; NIRANJAN, M. **“On-Line Q-Learning Using Connectionist Systems”**. Technical Report no. 166, Engineering Department, University of Cambridge, 1994.

SAFADI, F.; ERNST, D. **“Organization in AI Design for Real-Time Strategy Games”**. M.Sc. Thesis. University of Liège, Belgium, 2010.

SAILER, F.; BURO, M.; LANCTOT, M. **“Adversarial Planning Through Strategy Simulation”**. IEEE Symposium on Computational Intelligence and Games, pp. 80-87, 2007.

SANDBERG, T. W. **“Evolutionary Multi-Agent Potential Field Based AI Approach for SSC Scenarios in RTS Games”**. M.Sc. Thesis. IT University, Denmark, 2011.

SCOTT, B. **“Architecting an RTS AI”**. AI Game Programming Wisdom, Charles River Media, pp. 397-401, EUA, 2002.

Starcraft AI Club. Disponível em: <<http://www.cs.bham.ac.uk/~tpc/StarCraft/>>. Acesso em: Fevereiro 2013.

STEWART, K.; CHOI, H. P. **“PC-Bang (Room) Culture; A Study of Korean College Students’ Private and Public Use of Computers and the Internet”**. Trends in Communication 11(1), pp. 61-77, 2003.

SUTTON, R. S. **“Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming”**. Proceedings of the Seventh International Conference on Machine Learning, pp. 216-224, 1990.

SUTTON, R.S. **“Reinforcement Learning”**. The MIT Encyclopedia of the Cognitive Sciences, MIT Press, 1999.

SUTTON, R.S.; WHITEHEAD, S.D. **“Online Learning with Random Representations”**. Proceedings of the International Conference on Machine Learning, pp. 314-321, 1993.

TESAURO, G. **“Practical Issues in Temporal Difference Learning”**. Machine Learning 8, pp. 257-277, 1992.

Visual Studio 2008 Express. Disponível em: <<http://www.microsoft.com/en-us/download/details.aspx?id=3092>>. Acesso em: Fevereiro 2013.

WALTHER, A. **“AI for Real-Time Strategy Games”**. M.Sc. Thesis. IT-University of Copenhagen, Denmark, 2006.

Wargus. Disponível em: <<http://wargus.sourceforge.net/index.shtml>>. Acesso em: Fevereiro 2013.

WATKINS, C. J. C. H. “**Learning from Delayed Rewards**”. Ph.D Thesis, University of Cambridge, England, 1989.

WINTERMUTE, S.; XU, J.; LAIRD, J. E. “**SORTS: A Human-Level Approach to Real-Time Strategy AI**”. Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE), 2007.

ANEXO A - *Short Paper* - "Uma Proposta de Modelagem para uso do Aprendizado por Reforço na Definição de Táticas de Combate em Jogos de Estratégia em Tempo Real"

Um *short paper* foi produzido durante o desenvolvimento do trabalho descrevendo a sua proposta geral e os resultados que eram esperados após a realização dos testes. O artigo foi submetido e aceito no SBGames 2012 (Simpósio Brasileiro de Games e Entretenimento Digital). O *short paper* é apresentado a seguir, em conjunto com a sua notificação de aceitação.

Uma Proposta de Modelagem para uso do Aprendizado por Reforço na Definição de Táticas de Combate em Jogos de Estratégia em Tempo Real

Gutenberg Pessoa Botelho Neto Clauriton de Albuquerque Siebra

Universidade Federal da Paraíba, Programa de Pós-Graduação em Informática (PPGI/UFPB)
João Pessoa – PB – Brasil

Abstract

Jogos de estratégia em tempo real estão sendo cada vez mais utilizados em pesquisas de inteligência artificial por contarem com características importantes para a área, como a presença de ambientes complexos, muitas vezes dinâmicos e com múltiplos agentes. Este trabalho propõe a utilização de SARSA, uma técnica de aprendizagem de máquina, aplicada a situações de combate em jogos do gênero. O domínio do combate é fundamental na busca da vitória nestes jogos e, com a utilização da aprendizagem de máquina, busca-se permitir ao computador avaliar de maneira autônoma e dinâmica as situações que encontrar nas partidas, aprendendo com elas e mudando seu comportamento ao longo do tempo. Assim, são evitados problemas comuns existentes na inteligência artificial de jogos comerciais, que além de serem geralmente estáticas, tornando-se previsíveis para o jogador após algum tempo, são desenvolvidas com comportamentos definidos manualmente, o que é demorado e propício a erros.

Keywords: inteligência artificial, aprendizado por reforço, RTS, SARSA

Authors' contact:

gutenberg@fuze.cc
clauriton@di.ufpb.br

1. Introdução

Jogos de Estratégia em Tempo Real (Real-Time Strategy Games – RTS) representam um gênero dos jogos eletrônicos caracterizado pela disputa de duas ou mais facções controladas por humanos ou pelo computador. As responsabilidades dos jogadores nestes jogos incluem, geralmente: coletar recursos, adquirir unidades e estruturas, explorar e expandir o território, engajar em combate com inimigos, entre outras [Aha et al. 2005; Churchill and Buro 2011; Sailer et al. 2007].

Este gênero é considerado um dos mais promissores para pesquisas de inteligência artificial (IA) em jogos, pois engloba uma variedade de problemas fundamentais da IA: planejamento adversarial em tempo real, tomada de decisão com informação incompleta, colaboração de grupos, entre outros [Buro 2004; Buro and Furtak 2003].

Mesmo com o aumento das pesquisas na área, a dificuldade de desenvolver uma IA completa e desafiadora faz com que muitos jogos comerciais não utilizem uma IA “verdadeira”, permitindo que o computador trapaceie para oferecer um maior desafio ao oponente [Dimitriadis 2009; Sailer 2007; Sandberg 2011; Walther 2006]. Além disso, a IA é geralmente estática e não-adaptativa. Assim, o computador torna-se um jogador previsível, que utiliza sempre a mesma estratégia, e os jogadores que o enfrentam são capazes de criar uma estratégia ótima para derrotá-lo [Dimitriadis 2009; Kok 2008; Ponsen 2004].

Por conta destes fatores, muitas das pesquisas recentes da IA em jogos RTS são voltadas a técnicas de aprendizagem de máquina (Machine Learning - ML) aplicadas a áreas diversas do jogo, buscando permitir ao computador jogar partidas de maneira inteligente, com as mesmas informações dos jogadores humanos e com a possibilidade de adaptar-se ao adversário e explorar novas estratégias.

Este trabalho propõe uma técnica para a utilização de aprendizagem de máquina voltada a situações de combate em jogos RTS. O combate é um campo interessante de pesquisa, pois conta com características complexas como a necessidade de interação e cooperação entre unidades aliadas e a presença de unidades inimigas com objetivos conflitantes, além de não ter sido tão explorado como outras situações da IA de jogos RTS (e.g., planejamento de estratégia de alto nível, coleta de recursos).

A proposta utiliza SARSA, uma técnica de uma subárea da aprendizagem de máquina chamada de aprendizado por reforço (Reinforcement Learning – RL), e será implementada no ambiente BWAPI, que permite o desenvolvimento e validação da IA diretamente no jogo comercial Starcraft: Brood War.

2. Aprendizagem de Máquina

A aprendizagem de máquina é uma área da IA que busca fazer com que os computadores possam modificar seu comportamento através do conhecimento adquirido a partir de dados diversos.

O aprendizado pode ocorrer de forma offline, onde todo o treinamento é realizado a priori e o

conhecimento do sistema não é modificado durante a execução, ou de forma online, onde o sistema refina continuamente o seu conhecimento do mundo a partir dos dados coletados durante sua operação, adaptando seu comportamento de acordo com os novos dados. As aplicações online possuem diversas vantagens em relação às offline [Sutton and Whitehead 1993], sendo mais robustas por permitirem a correção de erros do conjunto de treinamento original a partir dos novos conhecimentos obtidos e também facilitando a geração de dados de treinamento confiáveis e em volume adequado.

O aprendizado por reforço é uma subárea da aprendizagem de máquina, caracterizada por sistemas que recebem apenas um conjunto de entrada e buscam obter informação sobre os possíveis valores de cada entrada através de uma função de recompensa e geralmente formalizada por um Processo de Decisão de Markov (Markov Decision Process – MDP) [Sutton 1999]. Nesta formalização, o ambiente é representado por um estado s_t , sendo t um momento discreto de tempo no qual há interação com o ambiente, onde o agente deve escolher uma ação a_t que mudará o ambiente estocasticamente para um novo estado s_{t+1} , resultando numa recompensa numérica r_t indicando se aquela ação específica tomada naquele estado foi boa ou não. A partir da recompensa recebida por cada entrada e após seguidas execuções, o computador passa a melhorar seu conhecimento de mundo e deve ser capaz de formular uma política definindo quais ações ele considera as melhores para os possíveis estados do ambiente. O framework básico do aprendizado por reforço é ilustrado na Figura 1.

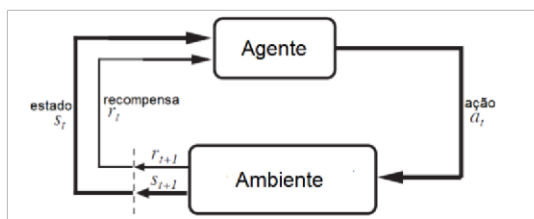


Figura 1: Framework básico do RL. Adaptado de [Sutton 1999].

Um ponto fundamental para a utilização com sucesso de técnicas de RL baseadas em MDP é a representação do estado. Como nesta formalização o espaço de busca cresce exponencialmente, é normalmente impossível representar com exatidão os estados do ambiente do problema. A representação do estado é uma abstração que deve simplificar o estado, diminuindo o espaço de busca, mas também mantendo uma dose adequada de fidelidade ao ambiente real para não prejudicar o aprendizado. Outra questão importante do RL é o dilema aproveitamento-exploração, referente à necessidade de balancear a preferência que deve ser dada a ações tomadas previamente com bons resultados pelo agente com a investigação de outras possíveis ações que possam permitir ao agente alcançar resultados melhores do que os atualmente conhecidos.

SARSA [Rummery and Niranjan 1994] é uma técnica do aprendizado por reforço, bastante popular por ser de fácil implementação e por garantir convergência para uma política ótima (i.e., que garanta a escolha da melhor ação para cada estado) sob condições apropriadas. A técnica já foi utilizada com sucesso em diversos problemas de RL e, portanto, é considerada uma escolha atraente para a investigação de problemas da área.

Nesta técnica, cada par de estado-ação (i.e., entrada e saída) é representado por um valor $Q(s,a)$, indicando a qualidade da ação a no estado s , levando em conta o estado atual e também as situações futuras. Com isto, garante-se que o sistema não terá uma visão imediatista, buscando apenas a maior recompensa atual, mas também avaliará possíveis melhorias que a ação possa trazer a longo prazo.

3. Ambiente

Uma forma de facilitar o desenvolvimento de uma IA para jogos RTS é a utilização de um ambiente apropriado para isto. O ambiente provê o jogo em si e todas as ferramentas necessárias para a modificação do comportamento do jogador durante uma partida. Isto permite que o pesquisador foque apenas na programação relacionada à IA, e não em outras áreas como o desenvolvimento do jogo propriamente dito. Para este trabalho, o ambiente escolhido foi o BWAPI (Brood War Application Programming Interface).

O BWAPI é um framework que permite a criação de módulos de IA a serem utilizados no jogo Starcraft: Brood War. Nele, o desenvolvedor tem acesso a todas as informações que o jogo oferece a qualquer jogador e pode ordenar que a IA execute qualquer ação que um jogador regular poderia executar durante uma partida.

A grande vantagem deste ambiente é permitir o desenvolvimento e validação da IA diretamente em um jogo RTS comercial de alta popularidade, ajudando a demonstrar a viabilidade prática dos trabalhos desenvolvidos no mesmo. Por conta disto, tornou-se a principal ferramenta utilizada em pesquisas de IA para jogos RTS.

4. Trabalhos Relacionados

A importância da pesquisa da IA em jogos RTS já era mencionada e detalhada em dois artigos de 2003 e 2004 [Buro 2004; Buro and Furtak 2003], citando o fato dos jogos do gênero envolverem muitos problemas fundamentais da IA.

O problema do combate em jogos RTS já foi tratado com técnicas variadas, como a utilização de árvores de jogos [Laursen and Nielsen 2005], que apesar de obter resultados satisfatórios teve dificuldades por conta da complexidade do problema;

com planejamento [Balla and Fern 2009], que obteve bons desempenhos em diversos cenários de testes; com campos potenciais [Sandberg 2011], que também utilizou o BWAPI e conseguiu derrotar a IA padrão do jogo; com autonomia orientada a objetivos [Jaidee et al. 2011], também com resultados interessantes em dois cenários diferentes e com melhoria de desempenho por conta de aprendizado ao longo do tempo.

5. Proposta

O trabalho proposto visa à aplicação de uma técnica online de ML para controlar unidades de um jogo RTS durante o combate com outra facção. Assim, espera-se que o computador seja capaz de aprimorar sua estratégia no decorrer das partidas, tornando-se cada vez mais difícil de ser derrotado. Outro resultado da utilização da ML deve ser a capacidade de adequar-se aos oponentes após disputar uma quantidade suficiente de partidas, evoluindo sua estratégia de forma a diminuir a sua previsibilidade e tentar combater o oponente de maneira mais eficiente.

Além do ambiente escolhido para o desenvolvimento, existem outros aspectos fundamentais da proposta que serão detalhados a seguir. São eles: representação do estado, possíveis ações dos agentes, detalhes da utilização da SARSA, além de uma ideia inicial de como será realizada a validação futura do trabalho.

5.1 Representação do Estado

Como dito na Seção 2, a representação do estado é a forma que o agente tem de perceber o mundo ao seu redor e é fundamental para o aprendizado, pois cada conhecimento obtido pelo agente é associado a um dos estados do espaço de busca. Assim, a representação deve permitir que o agente compreenda a situação atual da partida e deve oferecer informação suficiente para que as melhores ações sejam escolhidas.

Como não há abstração padrão de estados para a situação de combate em jogos RTS, uma das contribuições deste trabalho é a definição de uma representação para o agente nesta situação. A representação conta com seis variáveis possuindo três valores cada, totalizando 729 possíveis estados em que o agente pode estar situado. As variáveis usadas na representação, assim como seus possíveis valores, são:

- **Vida** atual do agente. Valores: baixa (menos de 1/3 restante), média (entre 1/3 e 2/3 restante), alta (mais de 2/3 restante);
- Quantidade de **inimigos que podem ser atacados** pelo agente a partir da sua posição atual, sem exigir movimentação extra. Valores: 0, 1, 2+;
- Quantidade de **inimigos que podem atacar** o agente em sua posição atual. Valores: 0, 1, 2+;

- Quantidade de **inimigos atacando** atualmente o agente. Valores: 0, 1, 2+;
- Quantidade de **aliados próximos** ao agente, que estão no raio de vizinhança do mesmo, definido pelo jogo de acordo com o alcance da arma da unidade. Valores: 0, 1, 2+;
- **Vida média** dos inimigos que estão próximos ao agente. Valores: baixa (menos de 1/3 restante), média (entre 1/3 e 2/3 restante), alta (mais de 2/3 restante).

Com esta representação, acredita-se que o agente será capaz de obter um sentido geral da sua situação na batalha, sem trazer prejuízos ao seu processo de decisão de ação a ser tomada.

5.2 Ações

Como as ações também influenciam no tamanho do espaço de busca, é necessário fazer uma abstração das possíveis ações assim como na representação de estado.

Para o trabalho atual, a proposta é a de ter ações que definam uma postura de alto nível para o agente, deixando a implementação de baixo nível de fora do aprendizado. Como sugestão inicial tem-se as seguintes posturas que podem ser tomadas:

- **Ataque:** postura ofensiva, buscando causar o máximo de dano possível ao oponente;
- **Defesa:** nesta postura o agente está mais preocupado em sobreviver, mas sem fugir da batalha corrente;
- **Recuo:** o agente assume uma postura de fuga para garantir sua sobrevivência, abdicando momentaneamente de ações ofensivas.

5.3 Técnica

O sistema foi pensado de maneira a utilizar conceitos conhecidos do RL já aplicados com sucesso a diversos ambientes. Também houve a preocupação de propor uma utilização simples, tendo em vista o fato de que o tempo de processamento destinado a IA para combate não será exclusivo, mas dividido com o tempo cedido aos outros aspectos da simulação do jogo e, portanto, exigirá execução rápida da computação necessária.

O trabalho utilizará uma tabela de valores Q única para todas as unidades de um mesmo tipo. Isto é útil por permitir que cada agente utilize o conhecimento adquirido pelos outros agentes semelhantes a ele, acelerando o processo de aprendizado. Outra vantagem é que, como normalmente cada jogador utiliza em um ataque diversas unidades de cada tipo, serão aliviadas as necessidades de memória do sistema na maioria dos casos.

Apesar de SARSA não ser uma técnica multi-agentes, é fundamental que o sistema tenha um sentido

de cooperação de agentes do mesmo time, de forma a permitir que estes trabalhem de maneira conjunta e sem ações conflitantes. Uma forma de obter isto é a partir da função de recompensa utilizada na técnica. A proposição de uma fórmula nestes termos segue como contribuição adicional do trabalho:

$$\left(\sum_{i \in I} h_{i,t} - \sum_{i \in I} h_{i,t+1} \right) + (100 * n_{dt}) - [(v_{t+1} - v_t) + (100 * m_{dt})]$$

Figura 2: Função de recompensa

onde I representa o conjunto de inimigos próximos do agente no tempo t , $h_{i,t+1}$ e $h_{i,t}$ representam a vida da unidade i nos tempos $t+1$ e t , n_{dt} representa a quantidade de inimigos mortos entre os tempos t e $t+1$, v_{t+1} e v_t representam a vida do agente nos tempos $t+1$ e t e m_{dt} vale 1 se o agente tiver morrido entre os tempos t e $t+1$ e vale 0, caso contrário. Não há indicação do agente causador de dano aos inimigos. Assim, valoriza-se a ação conjunta ofensiva dos aliados próximos, mas penaliza-se o agente pelo seu dano individual. Com esta abordagem híbrida, espera-se dar ao agente feedback suficiente sobre a sua ação tanto em relação a seu desempenho individual quanto ao de sua equipe.

5.4 Validação

A validação se dará com partidas disputadas entre a IA proposta e a IA padrão do jogo em mapas diversos, específicos para combate (i.e., sem base ou recursos a coletar), com configurações distintas de equipes. O objetivo será mostrar a capacidade de evoluir o comportamento do computador ao longo do tempo com a técnica de ML, de modo que possa eventualmente aprender o suficiente para derrotar consistentemente seu adversário nas situações avaliadas.

6. Conclusão e Trabalhos Futuros

Atualmente, o sistema proposto já se encontra em desenvolvimento, com o algoritmo da técnica SARSA implementado e integrado ao ambiente BWAPI, assim como a obtenção do estado atual por parte de cada agente e de acordo com a representação sugerida. O foco no futuro próximo é na definição e execução dos testes a serem utilizados para validação, de forma a poder confirmar a viabilidade da técnica descrita.

Com a aplicação de sucesso da proposta, espera-se comprovar a possibilidade de utilização de técnicas clássicas da IA e ML em jogos comerciais, que possam aumentar a qualidade do jogo e a percepção de valor dos jogadores. O fato do problema escolhido – combate em jogos RTS – possuir muitas das características mais complexas que os jogos eletrônicos podem apresentar para a IA serve para aumentar a importância da investigação. Outra contribuição esperada é a de mostrar o resultado da utilização de SARSA, já aplicada com sucesso em diversas situações, ao problema do combate em jogos RTS.

Referências Bibliográficas

- AHA, D. W., MOLINEAUX, M. AND PONSEN, M., 2005. Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. In: Proceedings of the Sixth International Conference on Case-Based Reasoning, 5-20.
- BALLA, R-K. AND FERN, A., 2009. UCT for Tactical Assault Planning in Real-Time Strategy Games. In: Proceedings of the International Joint Conference on Artificial Intelligence, 40-45.
- BURO, M., 2004. Call for AI Research in RTS Games. AI in Games Workshop, AAAI
- BURO, M. AND FURTAK, T., 2003. RTS Games as Test-Bed for Real-Time Research. Invited Paper at the Workshop on Game AI, JCIS, 481-484.
- CHURCHILL, D. AND BURO, M., 2011. Build Order Optimization in Starcraft. AI and Interactive Digital Entertainment Conference (AIIDE).
- DIMITRIADIS, V. K., 2009. Reinforcement Learning in Real Time Strategy Games Case Study on the Free Software Game Glest. Technical University of Crete.
- JAIDEE, U., MUÑOZ-AVILA, H. AND AHA, D. W., 2011. Case-Based Learning in Goal-Driven Autonomy Agents for Real-Time Strategy Combat Tasks. Case-Based Reasoning in Computer Games: Papers from the ICCBR Workshop.
- KOK, E., 2008. Adaptive Reinforcement Learning Agents in RTS Games. M.Sc. Thesis, University Utrecht.
- LAURSEN, R. AND NIELSEN, D., 2005. Investigating Small Scale Combat Situations in Real Time Strategy Computer Games. M.Sc. Thesis, University of Aarhus.
- PONSEN, M., 2004. Improving Adaptive Game AI with Evolutionary Learning. M.Sc. Thesis, Delft University of Technology.
- RUMMERY, G. A. AND NIRANJAN, M., 1994. On-Line Q-Learning Using Connectionist Systems. Technical Report no. 166, Engineering Department, University of Cambridge.
- SAILER, F., BURO, M. AND LANCTOT, M., 2007. Adversarial Planning Through Strategy Simulation. IEEE Symposium on Computational Intelligence and Games, 80-87.
- SANDBERG, T. W., 2011. Evolutionary Multi-Agent Potential Field Based AI Approach for SSC Scenarios in RTS Games. M.Sc. Thesis, IT University.
- SUTTON, R. S., 1999. Reinforcement Learning. The MIT Encyclopedia of Cognitive Sciences, MIT Press.
- SUTTON, R. S. AND WHITEHEAD, S. D., 1993. Online Learning with Random Representations. In: Proceedings of the International Conference on Machine Learning, 314-321.
- WALTHER, A., 2006. AI for Real-Time Strategy Games. M.Sc. Thesis, IT-University of Copenhagen.

Your SBGames 2012 - Trilha de Computação paper 106237

o <ricardo.nakamura@poli.usp.br>

Para: gutenberg@lavid.ufpb.br

Cc: Clairton Siebra <cas@cin.ufpe.br>

Dear Mr. Gutenberg Neto:

Congratulations! Your paper #106237 "Uma Proposta de Modelagem para uso do Aprendizado por Reforço na Definição de Táticas de Combate em Jogos de Estratégia em Tempo Real" for SBGames 2012 - Trilha de Computação has been accepted.

The reviews are below or can be found at

<https://submissoes.sbc.org.br/PaperShow.cgi?m=106237>

Please note the following:

1) The deadline for uploading your final camera-ready paper is September 20th, 2012 23:55pm. Use the same entry in JEMS for uploading your final manuscript. Please take into account the comments from the reviewers when preparing your final submission. Only pdf file format is accepted

2) Please do not forget to add the authors and contact information in your camera-ready paper since this information was missing due to blind review procedures

3) You will need to fill out the copyright form available at

http://www.sbc.org.br/index.php?option=com_jdownloads&Itemid=195&task=viewcategory&catid=74 by the same deadline. You will receive further instructions on how to proceed related to this form.

4) At least one author of each accepted paper must be registered for the conference for publication of the paper in the proceedings. For registration go to <http://www.sbgames2012.com.br/>

We hope to see you at SBGames 2012, in Brasilia!

Regards,
Conference Chairs