



Pós-Graduação em Ciência da Computação

**“GERENCIADOR DE RECURSOS EM
AMBIENTES COMPLEXOS: UMA APLICAÇÃO
AOS JOGOS DE ESTRATÉGIA EM TEMPO
REAL”**

Por

THIAGO DE ANDRADE SOUZA

Dissertação de Mestrado



Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, JULHO/2013



UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

THIAGO DE ANDRADE SOUZA

**“GERENCIADOR DE RECURSOS EM AMBIENTES COMPLEXOS: UMA APLICAÇÃO
AOS JOGOS DE ESTRATÉGIA EM TEMPO REAL”**

*ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA
UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO
PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA
COMPUTAÇÃO.*

ORIENTADOR: GEBER LISBOA RAMALHO
CO-ORIENTADOR: SÉRGIO RICARDO DE MELO QUEIROZ

RECIFE, JULHO/2013

Catálogo na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Souza, Thiago de Andrade

Gerenciador de recursos em ambientes complexos: uma aplicação aos jogos de estratégia em tempo real / Thiago de Andrade Souza. - Recife: O Autor, 2013.

xii, 72 f.: il., fig., quadro

Orientador: Geber Lisboa Ramalho.

Dissertação (mestrado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2013.

Inclui referências.

1. Ciência da Computação. 2. Inteligência Artificial. 3. Jogos Digitais. I. Ramalho, Geber Lisboa (orientador). II. Título.

004

CDD (23. ed.)

MEI2013 – 161

Dissertação de Mestrado apresentada por **Thiago de Andrade Souza** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título “**Resource Management in Complex Environments: Applying to Real Time Strategy Games**” orientada pelo **Prof. Geber Lisboa Ramalho** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Patricia Cabral de Azevedo Restelli Tedesco
Centro de Informática / UFPE

Prof. Charles Andrye Galvão Madeira
Instituto Metr pole Digital / UFRN

Prof. Geber Lisboa Ramalho
Centro de Inform tica / UFPE

Visto e permitida a impress o.
Recife, 30 de agosto de 2013.

Profa. Edna Natividade da Silva Barros
Coordenadora da P s-Gradua  o em Ci ncia da Computa  o do
Centro de Inform tica da Universidade Federal de Pernambuco.

Dedicado à minha família, amigos e professores.

Agradecimentos

De início, desejo agradecer a Deus como criador de todas as coisas. Agradeço também à minha família em especial à minha mãe, Wilma Maria de Andrade, pelos bons princípios e valores passados através da minha criação. Seus ensinamentos me conduziram para que eu tivesse consciência de ética, humanidade e soubesse o devido valor da educação. Agradeço também aos meus amigos que sempre me auxiliam no constante aprendizado da vida, sem esses eu nada seria. Bruno Henrique de Andrade, César Aragão, Felipe Neiva, André Barros, Diogo Modesto, Victor Oliveira e Miguel Gastão, a alegria e o incentivo de vocês sempre me deram forças para eu atingir meus objetivos.

Agradeço também ao meu orientador, professor Geber Lisboa Ramalho, pela sua paciência e auxílio determinante para que este trabalho fosse possível. Agradeço também ao meu co-orientador Sergio Queiroz que foi de grande ajuda na condução deste trabalho.

Aos meus amigos de salas de aula e pesquisa que tanto me ajudaram, especialmente Renê Gadelha e Gilliard Allan, meu muito obrigado, vocês foram muito importantes para meu aprendizado durante esses anos de muito esforço.

Resumo

Gerenciamento de recursos é o processo de aplicar recursos disponíveis no corrente momento e aqueles que estão por se tornarem disponíveis no futuro para atingir objetivos de forma eficiente. Normalmente, quando recursos são escassos, esta tarefa não é fácil, especialmente quando os ambientes são em tempo-real, parcialmente observáveis, dinâmicos e incertos. Apesar de ser muito comum no mundo real assim como nos jogos digitais, particularmente nos jogos de estratégia em tempo real (RTS), existem poucas pesquisas nesse campo. Nesse estudo nós utilizamos RTS para propor uma abordagem baseada em política de investimento para realizar tomadas de decisões relacionadas à gestão de recursos em ambientes complexos. Nós desenvolvemos novas técnicas / conceitos e reusamos outros existentes. Realizamos várias simulações e os resultados se mostraram bastante promissores.

Palavras-chave: Ciência da Computação, Inteligência Artificial, Jogos Digitais, Gerenciamento de Recursos.

Abstract

Resource management is challenged to apply the resources currently available and those that are to become available in the future to achieve goals efficiently. Normally, when resources are scarce, this is not an easy task, especially when the environments are real-time, partially observable, dynamic and uncertain. Despite being very common in the real world as well as in digital games, in particular real time strategy games (RTS), there is little research in this field. In this study we use the RTS to propose an approach based in investment policy to decision-making involved in managing resources in complex environments. We develop new techniques / concepts and reuse some existing ones. We made several simulations and the results were very promising.

Keywords: Computer Science, Artificial Intelligence, Digital Games, Resource Management.

Lista de figuras

Figura 1 Jogo de RTS <i>Starcraft Broodwar</i>	9
Figura 2 Jogo de RTS <i>Plants VS Zombies</i>	10
Figura 3 Recurso de minério do <i>Starcraft Broodwar</i>	12
Figura 4 Recurso de gás do <i>Starcraft Broodwar</i>	12
Figura 5 Unidades do <i>Starcraft Broodwar</i>	13
Figura 6 Edificações do <i>Starcraft Broodwar</i>	13
Figura 7 Janela que exhibe Fog of War no <i>Starcraft Broodwar</i>	13
Figura 8 Agentes integrados	31
Figura 9 Terran Marines	43
Figura 10 Diagrama de estados da postura.....	46
Figura 11 Comparando PICFlex sem e com Investimentos Iniciais	50
Figura 12 Problemas de path finding.....	65

Lista de tabelas

Tabela 1 Crescimento equilibrado por um padrão.....	43
Tabela 2 Distribuição do Player 3	48
Tabela 3 Comparativo das propriedades dos players	49
Tabela 4 Proporções de investimento em Classes	51
Tabela 5 Proporções relativas ao Item Básico.....	52
Tabela 6 Comparativo dos players. T. Sucesso é Taxa de Sucesso. M. Dif. Score é a média da diferença entre o score do player e o score do inimigo. DP Dif Score é o desvio padrão do M. Dif. Score.....	56
Tabela 7 Tempo do Player 6.....	65

Lista de Acrônimos

<i>Acrônimo</i>	<i>Descrição</i>
RTS	<i>Real Time Strategy</i>
PICFlex	Política de Investimento Contextual e Flexível
BTHAI	<i>Starcraft Artificial Intelligence Bot</i>
BWAPI	<i>Broodwar Application Programming Interface</i>
MCPlan	<i>Monte Carlo Planning</i>
PDDL	<i>Planning Domain Definition Language</i>
SLA	<i>Search and Learning Algorithm</i>
MeaPop	<i>Means-End Analysis</i> com planejamento de ordem parcial

Sumário

1	Gerenciamento de recursos.....	1
1.1	Ambientes Complexos.....	2
1.2	Gerenciamento de Recursos em jogos de computador do tipo Estratégia em Tempo Real.....	3
1.3	Objetivos.....	5
1.4	Metodologia adotada	5
2	Problema.....	8
2.1	RTS – Estratégia em Tempo Real	8
2.1.1	Alguns tipos de jogos de Estratégia em Tempo Real	9
2.1.2	Componentes comumente encontrados em um jogo RTS militar	10
2.2	Gerenciamento de Recursos em Jogos de Estratégia em Tempo Real do gênero militar.....	14
2.2.1	Recursos limitados.....	15
2.2.2	Muitas opções de escolha	16
2.2.3	Quantidades a considerar.....	16
2.2.4	Opções de difícil avaliação (multi critério)	17
2.2.5	Dependência entre opções (<i>Quis custodiet ipsos custodes?</i>).....	18
2.2.6	O tempo	18
2.2.7	Contexto.....	19
2.2.8	Combinação entre as avaliações	19

2.3	Recomendações para a solução do problema	20
3	Estado da Arte	22
3.1	Breve panorama de IA em RTS.....	23
3.2	IA em RTS: resvalando gerenciamento de recursos.....	24
3.3	IA com foco no gerenciamento de recursos em RTS	26
4	PICFlex: uma abordagem de gerenciamento de recursos baseada em Política de Investimento Contextual e Flexível.....	32
4.1	Abstração e reformulação.....	33
4.2	Política de investimento.....	34
4.3	Arquitetura do PICFlex	40
4.3.1	Proporções Adaptativas de Classe	40
4.3.2	Item de investimento	41
4.3.3	Item básico de investimento	41
4.3.4	Crescimento equilibrado por um padrão	42
4.3.5	Regras Gatilho	44
4.3.6	Postura	45
4.4	As seis implementações do PICFlex	46
4.4.1	<i>Player 1</i> (player aleatório).....	46
4.4.2	<i>Player 2</i> (player de política de investimento aleatória).....	47
4.4.3	<i>Player 3</i> (player com política de investimento fixa definida por especialistas).....	47
4.4.4	<i>Player 4</i> (player com crescimento equilibrado por especialistas)	48

4.4.5	<i>Player 5</i> (player com política de investimento chaveada pelas posturas)	48
4.4.6	<i>Player 6</i> (player com política de gastos flexível)	48
4.5	<i>Player 6</i> em detalhes.....	49
4.6	Plataforma de simulações e testes	53
5	Resultados.....	55
5.1	Objetivos.....	55
5.2	Protocolo.....	55
5.3	Apresentação	55
5.3.1	Comparativo entre as implementações do PICFlex.....	56
5.3.2	Normalidade dos dados	56
5.3.3	Testes de Hipótese	56
5.4	Análise	59
5.4.1	Análise individual de cada <i>Player</i>	60
5.4.2	Análise Geral	63
6	Conclusões e trabalhos futuros	66
6.1	Conclusões.....	66
6.2	Trabalhos futuros.....	66
	Referências	69

Capítulo 1

1 Gerenciamento de recursos

Gerenciar recursos é o processo de usar recursos da maneira mais eficiente possível. Esses recursos podem ser tangíveis como equipamentos, recursos financeiros, e até mesmo recursos humanos como empregados (Resource Management, 2013). É um aspecto muito comum do mundo real. As pessoas gerenciam recursos a todo o momento, dentro de suas casas, no ambiente de trabalho, inclusive nas relações humanas. O trabalho de gerenciar recursos é intimamente relacionado à atividade de planejar, aspecto este intrínseco ao homem e sua evolução. Para gerenciar qualquer coisa é necessário que se faça um planejamento, por mais básico que este seja. Percebemos que até dormindo se está realizando gerenciamento de um recurso básico: o tempo. O tempo assim como uma quantia em dinheiro, ou pessoas (funcionários de uma empresa, por exemplo) são recursos que, diante de alguma situação que limite a quantidade disponível, precisam ser gerenciados. Um estudante de engenharia, por exemplo, não pode dormir indefinidamente, pois tem que dar conta dos seus compromissos diários como as aulas na universidade, o estágio curricular, e ainda ter tempo de estudar quando chega a sua casa. Ele precisa gerenciar o seu tempo de forma que consiga atingir seus objetivos, realizar o estágio, se formar em engenharia e se tornar um bom profissional.

Um prefeito de um município precisa gerenciar inúmeros tipos de recursos, como recursos financeiros, tempo, pessoas, etc. Por mais que os recursos financeiros dessa prefeitura seja uma cifra enorme, e que sua equipe de funcionários seja grande, ainda assim são limitados. Quando este recebe o orçamento do ano, precisa então encontrar a melhor forma de utilizar os recursos de modo que as atividades básicas sejam atendidas (tais como coleta do lixo, pagamento da folha de funcionários, saúde econômica do município). Outro exemplo de gerenciador de recursos é um pai, ou mãe, arrimo de família que tem que custear o dia a dia da casa se desdobrando para que sua receita (salário) seja suficiente para atender todas as necessidades.

Há gerenciamentos complexos como o exemplo do prefeito ou do pai de família, porém também há também gerenciamentos de recursos mais simples como, por exemplo, quando uma pessoa agenda mentalmente quais programas de televisão assistirá no seu fim de semana. Os programas de TV (nos seus devidos horários) e as próprias TVs são os recursos e a pessoa agenda o momento em que sentará no sofá e ligar o aparelho de TV de forma que consiga assistir os programas que mais lhe agrada, onde o nível da satisfação será o resultado do gerenciamento. Quando a disponibilidade de recursos é maior que a demanda o gerenciamento é muito simples o que leva a crer que na verdade inexistia qualquer gerenciamento.

Diante dos exemplos anteriores, acreditamos que “gerenciar recursos” é um problema bastante relevante e de interesse das pessoas, pois em muitas situações não é uma tarefa fácil de lidar. Uma pessoa estuda muito e se prepara bastante para então assumir um cargo de gestor de uma empresa porque é muito difícil lidar com as inúmeras variáveis intrínsecas aos mercados, com os conflitos de interesse das pessoas e o tempo que raramente é um recurso abundante.

1.1 Ambientes Complexos

Como dito na seção anterior gerenciar recursos vai desde algo simples como gerenciar diante de um guarda-roupa com que roupa uma pessoa irá trabalhar até algo muito mais complexo como gerenciar os recursos de uma prefeitura de uma grande metrópole. Esse último tipo de problema é um bom exemplo de o que chamaremos nesse estudo de “ambiente complexo”. Para referência durante todo este trabalho definimos, seguindo em boa parte a classificação de Russell e Norvig (Russell & Norvig), que um ambiente complexo é um ambiente no qual há os seguintes aspectos:

- Tempo real – o ambiente acontece no tempo real
- Parcialmente observável – nem todas as informações sobre o ambiente estão disponíveis
- Dinâmico – o ambiente é mutável e essas mudanças acontecem sem controle completo
- Incerto – não é possível saber com precisão o próximo estado do ambiente

- Há forte e múltipla interação entre as variáveis – as variáveis presentes no ambiente interferem entre si de diversas formas e não existe o controle completo sobre essas interações

Este último aspecto não é considerado no estudo de Russell e Norvig mas nos parece importante porque é algo que influencia diretamente no contexto e estado do ambiente.

Por se tratar de uma tarefa complexa, gerenciar recursos em ambientes complexos é custoso, comumente demandando tempo, dinheiro e pessoas. Automatizar o processo de gerenciamento de recursos nos parece ser bastante útil visto que facilitaria a vida das pessoas, em particular a vida dos gestores em sua tomada de decisão.

1.2 Gerenciamento de Recursos em jogos de computador do tipo Estratégia em Tempo Real

Baseados nos aspectos listados no conceito de ambiente complexo (definido na seção anterior), nós procuramos ambientes nos quais existissem tais aspectos e fosse possível realizar simulações com fins de aferição e comparação da evolução de soluções automáticas de gerenciamento de recursos. Decidimos então trabalhar com jogos de computador do tipo Estratégia em Tempo Real - RTS (do inglês, *Real Time Strategy*), pois são ambientes complexos para os quais existem diversos simuladores disponíveis à comunidade científica e já utilizados em outros estudos (Churchill & Buro, 2011) e (Weber, Mateas, & Jhala, 2010), o que acreditamos que é determinante para o sucesso da pesquisa. O jogo de computador funcionará como um laboratório de experimentação onde faremos interferências e coletaremos e analisaremos os resultados.

Jogos de computador do tipo Estratégia em Tempo Real, como *Starcraft* da *Blizzard Entertainment* e *Age of Empires* da *Ensemble Studios*, podem ser vistos como verdadeiras simulações militares onde jogadores lutam por recursos dispersos sobre um terreno em 2D (duas dimensões) alimentando sua economia, construindo prédios de defesa, atacando, defendendo, fazendo melhorias tecnológicas, desenvolvendo exércitos e guiando até a batalha em tempo real sempre visando o objetivo final que normalmente é derrotar as equipes inimigas e assegurar que seu império prevaleça (Buro, 2003).

Um jogo de RTS competitivo requer a tomada de decisões multicritério – decisões que levam em conta várias variáveis e opções de escolha – nos níveis estratégicos e táticos

sob restrições de tempo real e num ambiente incerto, dinâmico e parcialmente observável (Weber & Mateas, 2009). No nível estratégico, jogadores tomam decisões de alto nível como quais tecnologias desenvolver e o quanto fortalecer a economia *versus* produzir unidades de combate. No nível tático, jogadores decidem quando e onde atacar os oponentes. Além disso, os jogadores precisam realizar constante reconhecimento do oponente para tomar as decisões de contra ataque ou contingência.

Nos jogos de estratégia em tempo real qualquer criação de exército, construção de prédios ou desenvolvimento de tecnologia requer recursos para que a ação seja realizada. É um problema complexo onde o máximo conhecimento relevante para a tomada de decisão do ambiente é necessário, além de se tratar de ações que podem ser executadas simultaneamente. Dessa forma, para atingir o objetivo principal, que é a vitória da equipe, a coordenação desses agentes e o planejamento da utilização dos recursos é necessária, uma vez que todos os jogadores compartilham o mesmo ambiente e os recursos presentes numa rodada são limitados.

Os jogos de RTS, em sua maioria, contam com vários elementos que simulam itens de um ambiente de guerra ou outro tipo de disputa. As unidades comumente são: Estruturas terrestres, marítimas e aéreas. Além disso, há recursos da natureza os quais geram riquezas que, por sua vez, são necessárias para criar tais unidades. Durante uma sessão de jogo de RTS o jogador faz a si próprio uma série de questionamentos sobre maneiras de investir. Este tem um banco de recursos e durante o jogo é preciso investi-los para ganhar a partida. A decisão de “em quê investir” ou “quanto investir” é difícil de ser analisada e sua correteza é decisiva para o sucesso do jogador. Este inicia coletando recursos para desenvolver forças de ataque e defesa, para desenvolver sua tecnologia e para incrementar seu armamento. O gerenciamento de recursos apropriado é parte vital da estratégia bem sucedida (Buro, 2003).

É importante perceber que a criação de oponentes inteligentes e desafiadores não está limitada a jogos RTS. Simuladores de alto desempenho estão em alta demanda para treinamento militar pessoal e se tornarão parte do núcleo de combate automatizado e/ou sistemas de suporte a decisão em campo de batalha do futuro. Em um estudo de Lippe, Franceschini e Kalphat, está previsto que 20% do exército Americano será robótico em 2015 (von der Lippe, Franceschini, & Kalphat, 1999).

A literatura que estuda os jogos de Estratégia em Tempo Real é vasta. Muitas pesquisas se dedicam a RTS em geral, porém pouca coisa foi feita em Gerenciamento de Recursos em jogos de RTS especificamente.

1.3 Objetivos

O objetivo principal deste estudo é propor um sistema capaz de realizar o gerenciamento de recursos, particularmente em jogos de estratégia em tempo real. Para tanto, pretendemos aprofundar o problema identificando características de investimento de recursos em RTS que maximizam o ganho de pontuação do jogador, assim como investigando as técnicas passíveis de resolver tal problema de gerenciamento.

Ao nosso conhecimento da literatura da área de jogos e Inteligência Artificial, tal estudo não foi ainda realizado, o que indica a contribuição original do trabalho dessa dissertação.

1.4 Metodologia adotada

O início da pesquisa foi marcado por intenso estudo sobre jogos de estratégia em tempo real, o gerenciamento de recursos aplicado nesses jogos, além do uso da Inteligência Artificial nesse domínio. Estudamos também vários aspectos desafiadores dos jogos de RTS como “tomada de decisão sob incerteza” (Hyde), “raciocínio espacial e temporal”(Retro Studios), “aprendizagem de modelagem de oponente” (van den Herik, Donkers, & Spronck, 2005) e “planejamento adversário em tempo real” (Sailer, Buro, & Lanctot, 2007), pois conhecer tais aspectos tão importantes de um jogo de RTS é essencial para tomar decisões de projeto durante o estudo.

Este levantamento tinha como objetivo chegar a um entendimento mais preciso do que é o problema de gerenciamento de recursos. O próximo passo foi estudar as soluções já existentes na literatura, e fazer uma análise delas, e se elas poderiam ser aplicadas no problema.

Pelo fato dos resultados de um jogo de RTS serem dependentes de muitas variáveis o passo seguinte foi estudar a literatura de tomada de decisão sob múltiplos critérios. A literatura de tomada de decisão sob incerteza (Hyde) também foi pesquisada, pois é a realidade de um jogo de RTS onde não há o controle nem conhecimento total do

ambiente. Depois disso a tarefa foi definir os parâmetros a serem avaliados, e quais abordagens seriam propostas para resolver cada um dos problemas encontrados.

Diante da complexidade do problema, optamos por partir de uma abordagem o mais simples possível, explorando o conceito de abstração. Isto foi feito a partir do conceito básico que criamos e denominamos de "política de investimento" que, grosso modo, é um modelo de gerenciamento recursos que obedece a um comportamento padronizado.

A partir daí, fomos tornando o modelo mais sofisticado para poder levar em conta o contexto e para dar mais flexibilidade às decisões. Foi onde introduzimos, respectivamente a possibilidade de troca dinâmica de política de investimento, assim como a ideia de flexibilizar as margens de gastos da política escolhida através das estratégias de execução da política de investimento S .

Foram criadas as combinações possíveis das diferentes abordagens, gerando vários módulos de IA a serem analisados. A escolha destas arquiteturas começou das mais simples para as mais complexas, visando mostrar a variação de comportamento à medida que suas complexidades iriam aumentando.

Cada um dos modelos foram sendo comparados confrontando um mesmo adversário computacional e inteligente de um jogo particular, o jogo *Starcraft Broodwar* com o *bot* BTHAI do pesquisador de inteligência artificial em games e robôs Dr. Johan Hagelbäck¹.

Em seguida foram implementados todos os modelos e definidas as regras das simulações. Com o resultado das simulações então foi feita uma análise individual e comparativa dos agentes no ambiente simulado. Os resultados são promissores e estão detalhados nos capítulos finais deste estudo.

No segundo capítulo apresentaremos os problemas relacionados aos jogos de RTS e gerenciamento de recursos em detalhes. Depois apresentamos o estado da arte da pesquisa de gerenciamento de recursos em jogos de Estratégia em Tempo Real e outros aspectos relacionados a esse campo. No capítulo 4 apresentamos as soluções dos

¹ <http://www.bth.se/tek/jhg.nsf/pages/273c7bbb699ba91cc1256bf80044ef09>

problemas estudados iniciando com a abstração e reformulação do problema e depois aplicando a solução ao mundo dos jogos digitais de RTS. No quinto capítulo apresentamos os resultados obtidos explicando os objetivos, o protocolo utilizado, até a análise final dos dados de forma individual e coletiva. Ao fim deste estudo concluimos nossas observações e citamos alguns trabalhos futuros que podem tomar proveito dos conceitos estudados neste trabalho.

Capítulo 2

2 Problema

Gerenciar recursos é o processo de usar recursos da maneira mais eficiente possível. Esses recursos podem ser tangíveis como equipamentos, recursos financeiros, e até mesmo recursos humanos como empregados (Resource Management, 2013). O trabalho de gerenciar recursos é intimamente relacionado à atividade de planejar, aspecto este intrínseco ao homem e sua evolução. Para gerenciar qualquer coisa é necessário que se faça um planejamento, por mais básico que este seja. E dentre os domínios que requerem gerenciamento de recursos, e que permitem simulações com algum grau de facilidade, estão os jogos de RTS, portanto focamos nessa plataforma.

2.1 RTS – Estratégia em Tempo Real

Jogos RTS (do inglês *Real Time Strategy*) são jogos de estratégia em tempo real com elementos de construção de bases e gerência de recursos. Em sua maioria, conta com vários elementos que simulam itens de um ambiente militar como uma guerra ou outro tipo de disputa. O objetivo de uma partida de um jogo desse gênero é, normalmente, o de destruir os exércitos inimigos, situados num mapa (um campo de batalha virtual). Um exemplo de jogo de estratégia em tempo real pode ser visto na Figura 1. Outros objetivos comuns envolvem garantir a segurança de uma área ou de um conjunto de construções ou desenvolver um exército com características definidas no começo de uma missão. Há, ainda, partidas com tempo pré-determinado. Nessa última modalidade, o vencedor costuma ser aquele que destruir mais unidades inimigas, sem a necessidade de destruir totalmente o inimigo.



Figura 1 Jogo de RTS *Starcraft Broodwar*

Essas unidades comumente são: estruturas terrestres, marítimas e aéreas. Além disso, há recursos do ambiente do jogo, que são objetos de riqueza que por sua vez criam a economia do império, equipe ou time. Esses recursos são essenciais para criar as unidades do jogo sendo pré-requisitos assim como um valor numa moeda monetária corrente de um país é necessário para efetuar uma compra qualquer.

2.1.1 Alguns tipos de jogos de Estratégia em Tempo Real

Por mais que a maioria dos jogos de estratégia em tempo real seja do **gênero militar**, onde a disputa é similar a uma guerra, há também outros tipos de jogos de estratégia em tempo real. Também há alguns outros gêneros que podem ser ditos como sendo de estratégia em tempo real como **tycoon** (ou *business simulation games*) que são jogos focados no gerenciamento de processos econômicos normalmente na forma de negócios². Exemplos de jogos *tycoon* é o *Sid Meier's Railroad Tycoon* da *MicroProse* e o *Air Bucks* da *Impressions Games*. Outro gênero é o **tower defense** no qual o jogador precisa defender uma torre (na tradução pura), onde essa torre pode ser um portal, uma delimitação no terreno ou qualquer coisa que demarque um alvo. O objetivo é que o jogador impeça que o inimigo chegue ao seu alvo. Na Figura 2 pode-se ver um dos jogos da modalidade *tower defense*, o *Plants VS. Zombies* lançado em 2009 pela *PopCap*.

² Definição do WikiPedia: http://en.wikipedia.org/wiki/Business_simulation_game



Figura 2 Jogo de RTS *Plants VS Zombies*

2.1.2 Componentes comumente encontrados em um jogo RTS militar

Neste trabalho vamos nos ater apenas aos jogos de estratégia em tempo real do gênero militar. Esses são simuladores de disputas similares a uma guerra (ou disputa por algo), nos quais os jogadores instruem unidades em tempo real para coletarem os vários tipos de recursos disponíveis, treinam unidades, constroem edificações, escoltam locais inimigos, e geralmente destroem oponentes para vencerem a partida. Na fase de abertura de um jogo de RTS os jogadores normalmente não interagem entre si porque suas posições iniciais se encontram dispersas por todo o mapa e a visibilidade do jogador é limitada a pequenas regiões ao redor de suas unidades, edificações e times amigos. Um dos principais objetivos nessa fase do jogo é estabelecer um fluxo razoável de entrada de recursos pela produção dos trabalhadores (aldeões, coletores, *workers*) que coletam os recursos, para rapidamente construir edificações que são pré-requisitos para outras estruturas ou para poder produzir unidades de combate que permitem criar um exército o mínimo suficiente para defender ou realizar um ataque inicial. Outro objetivo importante nessa fase é enviar escoltas para explorar o terreno e procurar bases inimigas (Churchill & Buro, 2011). Exemplos de jogos de estratégia em tempo real militares são *Starcraft* da *Blizzard Entertainment* visto na Figura 1, *Age of Empires* da *Ensemble Studios* e *Command & Conquer* da *Westwood Studios*.

Apesar de não haver consenso sobre os elementos de um jogo RTS, há alguns elementos e conceitos comuns, que são descritos abaixo.

Unidades Uma unidade (Figura 5) é qualquer objeto presente no jogo com o qual jogadores podem interagir. Neste grupo se destacam as unidades controladas pelos jogadores. Exemplos: soldados, aeronaves, cidadão. As unidades normalmente são combatentes, capazes de atacar e destruir outras unidades ou construções. Normalmente um tipo de unidade só pode ser produzida por um tipo de edificação. Através do desenvolvimento tecnológico, elas podem se especializar em ataque terrestre, de curto ou longo alcance ou ataque aéreo. Outras unidades podem ser capazes de extrair recursos e, comumente, incapazes de atacar de maneira efetiva – generalizado nesse trabalho com o nome de **cidadão**. Cada unidade tem características com valores diferentes, como *range*, *hit points*, *energy*, *speed* e, portanto finalidades diferentes. Cabe ao jogador decidir quais e quantas unidades treinar em função de sua estratégia.

Edificações As edificações (Figura 6) são normalmente usadas para produção e atualização de unidades, armazenamento de recursos, desenvolvimento de tecnologia ou defesa. As edificações também têm como finalidade o armazenamento de recursos ou das próprias unidades, por exemplo, as *Supply Depot* do jogo *Starcraft Broodwar*.

Recursos Recursos (Figura 3 e Figura 4) são pré-requisitos para produção das unidades, construções ou desenvolvimento de tecnologias e precisam estar armazenados antes de serem usados. Esses recursos estão espalhados pelo mapa e precisam ser coletados durante o decorrer de uma partida do jogo. Os recursos são essenciais para o treinamento de uma unidade, construção de uma edificação ou desenvolvimento de uma tecnologia. Funcionam como moeda econômica da seguinte forma: para construir um *Terran Marine* é preciso dispor de 50 pontos em minérios no jogo *Starcraft Broodwar*. Cada unidade, edificação, ou tecnologia da árvore tecnológica do jogo tem seus requisitos em recursos, ou seja, se o jogador não dispuser dos recursos necessários para uma unidade específica não poderá criar uma naquele momento. Cada jogo de RTS tem seu conjunto de recursos, em *Starcraft Broodwar*, por exemplo, o jogador deve coletar minério e gás, em *Command & Conquer*, unicamente minério, enquanto que em *Age Of Empires II*, necessita coletar quatro tipos de recursos: ouro, pedra, madeira e comida.

Tecnologias e atualizações As tecnologias e atualizações são bastante comuns nesse tipo de jogo e determinantes para que o jogador tenha maiores chances no combate com o seu inimigo, pois o desenvolvimento de tais tecnologias torna as suas unidades melhores. Uma tecnologia de armamento desenvolvida pode, por exemplo, aumentar o poder de fogo de todos os soldados do exército. A tecnologia desenvolvida pode melhorar as importantes características de um exército como: velocidade de movimentação, maior poder de ataque, melhor defesa, etc. É possível evoluir até edificações para que essas tenham maior resistência contra ataques inimigos.

Fog of War Representa um evento físico do mundo real que impede as entidades presentes de perceberem o ambiente onde vivem de maneira totalmente transparente. Nos jogos de estratégia esse aspecto pode ser percebido pela região escurecida do mapa na qual não é possível saber o que se passa. No jogo *Starcraft Broodwar* a janela que exibe o *Fog of War* fica no canto inferior esquerdo como pode ser visto na Figura 7.

Microgerência Microgerência é um termo usado para designar a atividade exercida pelo jogador quando sua atenção é direcionada à gerência e manutenção de suas próprias unidades e recursos. Um exemplo de atividade de microgerência é a necessidade de um jogador garantir que nenhum de seus trabalhadores está ocioso, para garantir entrada de recursos constante e manutenção da base.



Figura 3 Recurso de minério do *Starcraft Broodwar*



Figura 4 Recurso de gás do *Starcraft Broodwar*



Figura 5 Unidades do *Starcraft Broodwar*



Figura 6 Edificações do *Starcraft Broodwar*



Figura 7 Janela que exibe Fog of War no *Starcraft Broodwar*

Normalmente num RTS o jogador tem controle total sobre suas unidades e ele é responsável direto pela tática e estratégia a ser seguida. Através dessas unidades é que o jogador vai poder interagir com o mundo. Tais unidades podem ser um prédio, um tanque de guerra, ou um mero aldeão. As ações mais comuns executadas pelo jogador nos jogos RTS são as seguintes:

Coletar recursos O jogador necessita de unidades especiais que fazem a coleta de recursos, e precisa que essa coleta seja feita de forma aperfeiçoada visando otimizar a entrada de recursos na economia do jogador.

Construir edificações Construir edificações na ordem ótima de modo que requisitos como, precisa-se de 1 *barracks* – uma edificação do *Starcraft* – para construir 1 soldado, sejam atendidos.

Treinar unidades e desenvolver tecnologia Da mesma forma que as edificações, as unidades precisam ser treinadas e tecnologias serem desenvolvidas de forma cuidadosa atentando para utilização ótima dos recursos.

Combater os inimigos Os aspectos do jogo explicados anteriormente são realizados para aumentar o poder do exército e fornecer unidades bem preparadas

para o combate contra os inimigos, visto que o objetivo principal desse gênero de jogo é derrotar as equipes inimigas. No combate o jogador faz decisões tanto do nível estratégico, decidindo se vai atacar ou não, quanto no nível tático, onde ele decide que tipo de formação dar ao seu exército ou onde cada unidade deve atacar.

Além de todas essas tarefas, o jogador precisa tomar decisões no nível estratégico, como decidir se é melhor atacar naquele momento, ou se é melhor se defender e coletar mais recursos para treinar mais unidades e fazer um ataque mais forte posteriormente. Em um jogo RTS o jogador precisa também tomar decisões em um nível mais baixo, no nível tático. Por exemplo, no nível estratégico ele toma decisões como atacar ou defender, enquanto no nível tático o jogador vai definir como vai atacar o oponente, qual a formação que as suas unidades vão ter, se os soldados vão em bando, ou isolados, etc.

2.2 Gerenciamento de Recursos em Jogos de Estratégia em Tempo Real do gênero militar

Os jogos de RTS normalmente são vencidos por jogadores que destroem seus oponentes primeiro. Esse objetivo pode ser atingido de diversas formas. Por exemplo, um jogador poderia tentar surpreender (“*rush*”) o oponente investindo recursos no exército atacante logo no início do jogo em detrimento de investir nas edificações que são mais importantes nas fases tardias do jogo. Se o oponente investir em desenvolvimento tecnológico e se despreocupar com sua defesa, o jogador que está atacando terá sucesso facilmente. Mas se o defensor conseguir conter a investida, isto poderá ser um golpe fatal no atacante que “desperdiçará” seu investimento inicial e ficará atrasado em relação ao defensor. Portanto, a escolha das estratégias dos jogadores pode definir o resultado do jogo. Outra forma de vencer a partida seria investir recursos em edificações, montar grandes exércitos armados e desenvolver a tecnologia coletando o máximo de recursos possível para que os investimentos sejam possíveis e numa fase tardia engajar todo o exército contra o inimigo de uma só vez. Entre uma forma (*rush*) e outra (paciente), existem inúmeras possibilidades e adaptações que podem ocorrer durante o jogo em função da estratégia do adversário.

Até as pequenas decisões que o jogador toma durante o jogo, as quais se baseiam na estratégia escolhida são difíceis. Por exemplo: dado que a estratégia é a de construir um grande exército e atacar tardiamente, como investir os recursos em unidades de ataque?

Que tipos de unidade treinar? Quantas unidades treinar? Durante um jogo de RTS o jogador precisa lidar com diversas questões complexas presentes num ambiente dinâmico de tempo real. Por exemplo: um ataque inimigo acabou de começar, então o jogador deve primeiro investir em tecnologias de defesa ou ataque? E se esse ataque inimigo for interrompido, o jogador deve também interromper seus investimentos (que a priori foram criados por conta do ataque)?

Como veremos nas seções seguintes, a ordem na qual os itens presentes nos jogos de RTS, como edificações, unidades e tecnologia, são produzidos é chamada de *build order* (Weber & Mateas, 2009), mas queremos deixar claro que essa pesquisa tem como foco **Gerenciamento de Recursos**, que vai além da ordenação das criações das unidades.

Como será vista na revisão da literatura, não encontramos trabalhos que enumeram e sistematizam os diversos problemas encontrados na gestão de recursos em RTS. Como a análise do problema é um passo essencial na busca da solução, nós fizemos um esforço de produzir uma identificação preliminar destes problemas.

Para tanto, refletimos sobre nossa própria experiência enquanto jogadores de RTS e consultamos nove jogadores e por meio de uma entrevista – descrita resumidamente a seguir – identificamos alguns problemas citados por estes com frequência que, portanto, acreditamos que sejam os mais comuns. A seguir alguns problemas do gerenciamento de recursos em um RTS serão descritos.

Fizemos as seguintes perguntas a cada um dos jogadores:

1 - Quais são os maiores desafios relacionados à gestão de recursos que você enxerga?

2 - Como você enxerga a questão do tempo real desse tipo de jogo, ou seja, o tempo real cria algum tipo de dificuldade para você?

3 - Como você analisa o contexto global do jogo para a tomada de decisão de gestão de recursos? Explique resumidamente.

2.2.1 Recursos limitados

Assim como num jogo de RTS, onde os recursos são limitados, em qualquer situação onde se há de investir recursos para obter algo e que esses recursos sejam limitados tem-

se um claro problema. Caso contrário, se os recursos fossem infinitos não haveria por que se preocupar com o quanto se deve gastar nos investimentos, pois, nesta situação hipotética, sempre haveria recursos disponíveis.

Sendo assim, além de escolher o item no qual o investimento será feito, o jogador deverá atentar para o fator recursos, uma vez que, sem esses, nenhum investimento poderá ser feito. Nós entendemos que a análise dos recursos poderia ser feita de duas maneiras. A primeira forma de análise é aquela na qual o jogador poderá avaliar e escolher o item a ser investido e só depois verificar se há recursos disponíveis. Na segunda maneira, antes de fazer qualquer análise em relação aos itens de investimento, o jogador verifica o quanto de recursos há disponível para só então decidir o quanto está disposto a gastar.

2.2.2 Muitas opções de escolha

Um jogador durante uma partida de RTS se depara com uma infinidade de opções de escolhas de investimento de recursos disponíveis ou vindouros. Em cada momento específico do jogo, normalmente sempre estão disponíveis várias opções de escolha. Um tipo de cavaleiro, um tipo de soldado, várias tecnologias que podem ser desenvolvidas, prédios que podem ser construídos, etc. Mais uma vez, numa situação hipotética onde não houvesse tantas escolhas a atividade de tomar a decisão seria mais fácil.

Uma vez que o jogo começa, ações são executadas a todo o instante e em algum momento há de ser considerado investir em algum item (presente na árvore de tecnologia do jogo). O resultado dessa análise pode ser uma decisão de realizar algum investimento ou até mesmo de não realizar qualquer investimento no momento. Além de outros aspectos que serão descritos posteriormente, esse resultado dependerá também do número de itens passíveis de investimento a cada momento. Cada exemplar de jogo de RTS tem um conjunto desses itens, como já foi dito, portanto, outra questão complexa é a avaliação dos investimentos entre os itens passíveis de investimento (presentes no jogo), ou seja, presentes em sua árvore tecnológica.

2.2.3 Quantidades a considerar

Mais do que somente avaliar e escolher o item de investimento, o jogador precisa decidir a quantidade de itens que irá criar a cada vez. Um exemplo é o seguinte: o

jogador em um momento específico do jogo decide criar soldados rasos, mas precisa decidir também quantos itens desse tipo irá construir.

Outro tipo comum de avaliação que é feita durante um jogo de RTS que na situação o jogador sabe que precisa criar itens do tipo A e do tipo B e então precisa avaliar se irá investir em X itens do tipo A ou em Y itens do tipo B. Perceba que a comparação não é entre um item do tipo A e um item do tipo B, o que torna a decisão complexa. Por exemplo: o jogador percebe que precisa de dez soldados rasos e duas catapultas de ataque no breve momento; o jogador deverá então decidir o que é melhor, se é construir primeiro dez soldados ou as duas catapultas.

Vale salientar também que na maioria dos jogos há um número máximo de unidades por cada “império” e por mais que haja repositórios suficientes o jogador será impedido de ultrapassar esse limite. Esse número máximo de unidades existentes num império é importante para o jogador, pois este deverá levar isso em conta no seu planejamento de que opções escolher e em que quantidade irá criar.

2.2.4 Opções de difícil avaliação (multi critério)

Um aspecto bastante complexo é: como comparar as opções de investimento num momento específico? A priori a comparação de itens de uma mesma classe parece ser fácil e direta. Mas mesmo quando essas opções são itens de investimento de uma mesma classe, por exemplo: um soldado raso ou um cavaleiro – ambos são itens de combate que tem propriedades de ataque e defesa equivalentes, mas com valoração diferente, tornando sua comparação uma tarefa difícil, pois as variáveis do contexto são determinantes nessa comparação. Um cavaleiro pode ser melhor de que o soldado raso quando se está jogando contra um tipo de exército A, mas contra um exército do tipo B o cavaleiro é totalmente ineficiente e um soldado raso seria mais interessante.

Quando a comparação se trata de itens de classes distintas a dificuldade é ainda maior. Como comparar se em uma situação específica é melhor criar um *airport* (edificação que constrói itens aéreos) ou desenvolver uma tecnologia de armamento de soldados? O *airport* tem sua utilidade e a tecnologia também. Como saber em qual se deve realizar o investimento primeiro? Essa avaliação parece ser ainda bem mais complexa.

As opções de escolha de tomada de decisão em um jogo de RTS são além de tudo de difícil avaliação, pois são multi critério. Cada item da árvore tecnológica tem inúmeras propriedades associadas onde cada item tem seu conjunto não normalizado de propriedades. Por exemplo, um soldado raso tem poder de ataque com valor 10, defesa 20, resistência ao fogo 0 e *range* de ataque de 6m.

2.2.5 Dependência entre opções (*Quis custodiet ipsos custodes?*)

Fazendo o uso da famosa frase em latim do poeta romano Décimo Júnio Juvenal que em português tem o significado, em grosso modo, de "Quem irá vigiar os próprios vigilantes?" o problema da construção do item *worker* que existe nos jogos RTS que é responsável por construir as unidades de edificação do jogo que por sua vez criam as outras unidades, ou seja, sem *worker* não é possível criar outras unidades. Fazendo analogia com a frase do poeta o problema traduzido à realidade do jogo seria o seguinte: quem criará o *worker*? O “quem” da frase anterior vai muito além do concreto criador que dará vida ao *worker*. Refere-se muito mais à situação e circunstância que dará início a criação dessa unidade. Generalizando esse aspecto, é interessante falar sobre o fato de que há dependência entre as opções, ou seja, para uma opção A ser elencada como investimento possível, uma opção B deve ser antes criada, ou um recurso R estar disponível – o termo recurso nesse caso vai além dos financeiros podendo ser um tipo de item qualquer.

Um exemplo claro de dependência entre as opções que existe na maioria dos jogos RTS são os repositórios de unidades. As unidades não podem existir sem ter uma edificação que sirva de depósito, uma espécie de casa para cada unidade. Cada repositório desse guarda um número específico de unidades. Se não houver casa suficiente para as unidades o jogador não poderá construir nenhuma unidade além das já existentes. Assim como todas as outras unidades, as unidades do tipo repositório têm custos agregados. A avaliação feita relacionada a este problema é o de decidir o momento certo para construção dos repositórios das unidades.

2.2.6 O tempo

O tempo em si é um recurso altamente escasso num jogo de Estratégia em Tempo Real. Trata-se de um fator ortogonal a todas as decisões do jogo de RTS. Como o próprio termo “Estratégia em Tempo Real” já diz o jogo se passa no tempo real e todos os

jogadores aceleram o quanto podem suas ações de forma que consigam montar o maior “império” possível no mesmo espaço de tempo que os inimigos. Por exemplo, se um jogador é lento para tomar decisões e não cria no início do jogo um exército que o defenda esse poderá ser derrotado facilmente por um ataque adversário que o surpreenda.

O tempo é um dos maiores problemas quando a questão é utilizar os recursos que o jogador já detém somados aos recursos que ele irá ter numa janela de tempo qualquer. Por exemplo, não é fácil descobrir em quanto tempo estará disponível o recurso financeiro necessário para que seja possível construir um castelo de defesa, pois o gasto é variável e depende do contexto.

Avaliar e decidir se é melhor utilizar uma quantia específica de recursos no corrente momento em um investimento específico ou adiar o investimento para outro momento que poderá ser mais oportuno e eficiente é uma tarefa bastante complicada. Um exemplo para ilustrar essa dificuldade segue: como decidir se é melhor investir X agora para criar N unidades de infantaria ou esperar T turnos para desenvolver uma tecnologia de armamento que custará $X + Y$? A avaliação e maneira como o tempo é tratado no jogo de RTS é muito difícil e esse fator permeia vários outros problemas.

2.2.7 Contexto

A situação na qual se encontra o império também é um fator determinante para a tomada de decisão. É fácil perceber que quando seu império vive um momento de paz as decisões tomadas são muito diferentes das decisões tomadas numa situação onde seu império está sendo atacado pelo exército inimigo com uma frente de ataque devastadora. Por exemplo, quando o jogador está sendo atacado e não existem soldados para defendê-lo é, a priori, pouco recomendado criar trabalhadores em detrimento de treinar soldados (a não ser que os trabalhadores sejam criados com o intuito de esses construírem os prédios que irão treinar os soldados). A interpretação do contexto do jogo é algo bastante subjetivo e é essencial na avaliação das opções e no delineamento/mudança da estratégia.

2.2.8 Combinação entre as avaliações

Alguns problemas de gerência de recursos foram apresentados nas seções anteriores de forma separada para facilitar o entendimento, porém durante o jogo os problemas

claramente se misturam e ainda existe o contexto que muda a cada instante. Num ambiente de tempo real o fator tempo é ortogonal, influenciando todos ou a maioria dos problemas citados. Quando adicionada a questão do tempo ao problema das múltiplas escolhas torna-se ainda mais complexo, pois um item A pode ser melhor que o item B no tempo t_0 que é o tempo em que foi feita a avaliação, porém pode não ser interessante no tempo t_i que é o tempo no qual o investimento será realizado.

Todas as unidades, edificações e tecnologias tem seu próprio *build time* que é o tempo que é levado para este item ficar pronto. Isso acrescenta outro problema às avaliações que são feitas pelo jogador. Um exemplo claro é o seguinte: o jogador deverá avaliar se é melhor investir em um soldado que tem f de força de ataque e leva t segundos para ficar pronto ou investir em outro soldado que tem $f/2$ de força de ataque e $t/2$ segundos para ficar pronto.

O fator recursos disponíveis também é perpendicular aos outros problemas. Para exemplificar: soma-se ao problema item vs. item o fator recursos financeiros; se o item escolhido na análise feita custar 100U e a quantidade de recurso disponível no momento é de 80U o jogador deverá avaliar e decidir se adiará o investimento, ou se procurará o próximo investimento que caiba na quantidade de recursos disponíveis; para tornar ainda mais real, e complexa, a situação, o jogador deverá avaliar se diante da quantidade escassa de recursos é interessante poupar por um período de tempo t para acumular recursos e assim poder realizar investimentos mais custosos.

2.3 Recomendações para a solução do problema

Jogar RTS requer ao jogador prestar atenção aos inúmeros detalhes do jogo simultaneamente, reagindo rapidamente e apropriadamente em múltiplos níveis de abstração incluindo estratégia, tática e micro gerenciamento. Acreditamos que um bom jogo de RTS requer a combinação de atividades deliberadamente planejadas e respostas em tempo real às mudanças de condições do jogo. A melhor solução de gerenciamento e investimento de recursos será aquela que resultar nas maiores pontuações dos jogos. Outro critério de avaliação será, obviamente, o tempo. O tempo aliado à pontuação do jogo determinará se a vitória ou derrota do jogador foi melhor com o módulo de IA.

Espera-se que a solução proposta por este trabalho atenda a todos esses requisitos:

- Adaptabilidade – a solução deve ser adaptável às constantes mudanças (ambiente dinâmico) inerentes a um jogo de RTS se ajustando ao estado corrente. A adequação dos investimentos deve atender tanto às necessidades instantâneas quanto às necessidades de longo prazo, na situação a qual o módulo investidor poderá antecipar os itens que serão essenciais no futuro e planejá-los.
- Performance – o jogo de RTS é dinâmico onde o estado do mundo virtual muda a todo o momento. Uma solução para o problema do gerenciamento de recursos deve ser rápida o suficiente para que a tomada de decisão dos investimentos não atrase o andamento do jogo e que o tempo levado para tomar uma decisão seja curto o suficiente para o resultado seja aderente ao contexto do início do ciclo de decisão. Cada segundo que é gasto com computação de dados é valioso e pode determinar a defesa em tempo hábil logo após o início de um ataque adversário, ou a derrocada do “império”.

Capítulo 3

3 Estado da Arte

Este capítulo apresenta os trabalhos relacionados a esta pesquisa. Primeiramente são apresentados trabalhos relacionados à Inteligência Artificial em jogos RTS de uma forma geral. Logo em seguida, serão apresentados os trabalhos cujo tema se aproxima do gerenciamento de recursos além de outros trabalhos com temas que não se relacionam diretamente à questão dos recursos de um jogo de RTS, porém servem de base para a análise e avaliação dos resultados. Várias fontes de pesquisa foram consultadas, porém, como veremos nas seções seguintes, há pouca literatura diretamente relacionada a esta pesquisa, o que justifica e mostra a originalidade deste trabalho.

Há muitas pesquisas que tratam de problemas referentes aos jogos de Estratégia em Tempo Real como:

- Análise de terreno – analisar o terreno do mapa do jogo para encontrar obstáculos intransponíveis, determinar o nível de segurança de uma região do mapa, etc.
- Elaboração de um *build-order* – planejar uma ordem de criação de unidades de RTS que irão otimizar os resultados.
- Modelagem de adversário em tempo real – modelar o adversário e identificar aspectos importantes do seu comportamento em tempo real.
- Planejamento estratégico – sem se basear em *scripts* pré-definidos, criar um planejamento em alto nível de forma que um módulo de IA possa jogar sem a interferência de humanos.
- Alocação de recursos – coordenar a alocação de recursos de um jogo de RTS controlando as unidades responsáveis por tais ações

Há diversas outras questões que circundam os jogos de RTS e a lista poderia ser bem maior que essa anterior, porém o que nós percebemos é que não há muitas pesquisas que tratam especificamente do problema de Gerenciamento de Recursos. Na atual

literatura não há uma solução diretamente focada nesse gerenciamento, de forma que há muito espaço para nosso estudo.

3.1 Breve panorama de IA em RTS

O objetivo desta seção é fazer um levantamento das técnicas desenvolvidas nesta área, assim como os problemas associados e as soluções adotadas. É por meio dessa revisão que surgem as bases necessárias para a compreensão de onde este trabalho se situa no universo de jogos RTS e as bases teóricas para seu desenvolvimento.

Nos dias de hoje, a maioria dos jogos comerciais de RTS são desenvolvidos como sistemas baseados em regras com capacidades limitadas de planejamento. Entretanto, algumas técnicas de IA como mapas de influência e análise de terreno para lidar com novos mapas têm sido implementadas em jogos comerciais de RTS (Pottinger, 2000). Também tem havido progresso na problemática de construção de paredes (Teich & Davis, 2006), *pathfinding* (Sturtevant & Buro, 2005) (Demyen & Buro, 2006), e batalha local (Kovarsky & Buro, 2006). A melhoria dos módulos de IA de baixo nível é importante porque sem soluções efetivas nessa área, a pesquisa de raciocínio em alto nível e planejamento não pode seguir adiante. Atualmente, por conta do significativo avanço na IA de baixo nível o elo que falta para um sistema completo de IA de jogos RTS é o módulo de alto nível responsável pelo planejamento global (Kovarsky & Buro, 2006). O trabalho de Renato Cunha (Cunha & Chaimowicz, 2010) trata do problema de melhorar o desempenho do jogador RTS no qual seu objetivo é estabelecer métricas para avaliar o estado local do jogo.

O pesquisador Kevin Dill em 2006 defende a filosofia de que as decisões de IA de um jogo sejam feitas pela percepção do ambiente atual do jogo e não por scripts pré-definidos que são previsíveis. Para isso desenvolve uma arquitetura onde há atores que são as “coisas” do jogo que são controladas pelo homem, objetivos que são as ações do jogador como atacar, defender, etc., egos que são um tipo de postura ou personalidade que definem uma série de valores que determinam o comportamento da IA, o *goal engine* que toma decisões baseados em agendamento ou em eventos, e a IA reativa que supervisiona o *goal engine* e toma decisões mais simples e frequentes do que as dessa última (Dill, 2006). A filosofia de Kevin Dill que apoia as decisões baseadas na percepção do ambiente do jogo é o que guia o nosso estudo, pois entendemos que

scripts pré-definidos empobrecem a experiência do jogador, visto que em algum momento o comportamento da máquina se repetirá e poderá ser percebida pelo humano.

Os pesquisadores Josh MCCoy e Michael Mateas em seu estudo intitulado “*An Integrated Agent for Playing Real-Time Strategy Games*” apresentam um agente de IA de jogos de Estratégia em Tempo Real que integra múltiplos componentes especialistas para jogar de forma completa. Com base em uma análise de como os jogadores humanos qualificados conceitualizam a jogabilidade de um RTS, o espaço do problema é partido em domínios de competência elegidos pelos jogadores humanos especialistas. Os resultados desse trabalho mostram que incorporar conhecimento estratégico de alto nível ao agente permite que este derrote de forma consistente jogadores de IA *scripted*. Além disso, esse trabalho estabeleceu as bases para incorporar táticas e técnicas de microgestão de unidades desenvolvidos ambos por homem e máquina (MCCoy & Mateas, 2008). O estudo desse artigo foi muito importante, pois representou grande ajuda na modelagem da arquitetura do nosso módulo de IA. A divisão das responsabilidades dos agentes introduzida nesse artigo serviu de inspiração para os agentes do nosso sistema.

Comparada à história da pesquisa em IA, a pesquisa em jogos digitais é nova. Um dos primeiros trabalhos a motivar a pesquisa em jogos foi apresentado em (Laird & Jones, 1998), onde os autores, ao desenvolver pilotos artificiais para um projeto da Agência de Projetos de Pesquisa Avançada de Defesa, concluíram que muitos dos problemas por eles encontrados nesse trabalho eram compartilhados pelos desenvolvedores de jogos que buscavam produzir oponentes realísticos. Em um trabalho subsequente, (van Lent, et al., 1999), os autores integraram a arquitetura cognitiva Soar a jogos de tiro em primeira pessoa para a implementação de oponentes inteligentes, ou *bots*, que usavam sua base de conhecimento para abstração de detalhes do jogo.

3.2 IA em RTS: resvalando gerenciamento de recursos

Nesta seção vamos abordar os estudos que tratam de IA nos jogos de RTS mas que tangem de alguma forma o gerenciamento de recursos.

Uma abordagem interessante do problema de **gerenciamento de recursos** num jogo de RTS é feita por Harmon (Harmon, 2002) que se baseia no conceito econômico de utilidade marginal. O autor faz uma interessante analogia com a economia onde as

unidades do jogo são itens de consumo e o jogador é o próprio consumidor. Cria um modelo de avaliação dos itens de investimento que toma como entrada, basicamente, o custo monetário do item e seu tempo de produção. Essa ligação estabelecida pelo autor torna possível tratar aspectos de uma partida de RTS como conceitos da economia, que é uma ciência antiga e homologada.

O modelo de utilidade, do domínio da economia, é utilizado para criar um sistema que se adapta à realidade dinâmica de um jogo de RTS. É baseado no conceito de utilidade que se refere à quantidade de satisfação que um consumidor percebe em um produto, ou quão desejado é aquele produto. Uma função heurística estima se um dado estado está no caminho do objetivo, verificando o quão ajustado ao objetivo é este estado. Quando se modela um agente jogador, precisa-se de frameworks que provêm soluções eficientes aos problemas imprevisíveis do universo dos jogos e o conceito de modelo de utilidade, explicado nesse estudo, nos pareceu muito interessante, dessa forma, decidimos utilizar esse modelo para auxiliar a tomada de algumas decisões do nosso módulo de IA.

Em 2005 o pesquisador, bastante engajado na área de RTS, Michael Buro junto com Michael Chung e Jonathan Schaeffer (Chung, Buro, & Schaeffer, 2005) apresentaram um framework – MCPlan – para planejamento Monte Carlo. O estudo visou diminuir o uso de *scripts* nos jogos de RTS por estes serem custosos e previsíveis. Para essa finalidade os autores adaptaram o planejador Monte Carlo, que é usado com sucesso em jogos baseados em turnos como *poker* (Billings, Pena, Schaeffer, & Szafron, 1999) e gamão (Tesauro, 1995), à realidade de um jogo RTS que com informação imperfeita, movimentos simultâneos e em um ambiente estocástico. A ideia é aplicar a técnica Monte Carlo ao **planejamento estratégico de alto nível** em um jogo de RTS e para isto os autores abstraem os conceitos dos jogos de estratégia em tempo real. Os autores explicam que, no planejamento com o MCPlan, a abstração é a chave para obter busca em profundidade necessária para os jogos de RTS. Na prática, o MCPlan define um plano de alto nível aleatoriamente, simula e executa o plano e depois avalia o estado ao final da sequência e repete todos os passos – com restrições a cada ciclo – e ao final do ciclo escolhe o plano que obteve melhores resultados estatísticos. Os planos podem ser de ataque, de exploração, investir recursos em defesa, etc. cabendo ao utilizador do MCPlan a liberdade de escolher o nível de abstração e implementá-los.

O artigo mostra resultados promissores quando é aplicado a um cenário simples como o “capture a bandeira”, porém, os autores reconhecem que, para cenários mais complexos de jogos de RTS, a técnica requer melhores métodos de abstração. O trabalho utiliza uma abordagem interessante de planejamento estratégico, porém ainda requer bastante intervenção de especialistas em RTS para elaborarem planos e avaliarem os resultados de cada ciclo de execução de plano.

Num jogo de RTS há uma unidade conhecida normalmente por trabalhador (do inglês, *worker*), também conhecida como cidadão, aldeão, peão etc., que é a unidade responsável por construir todas as outras unidades de um jogo de RTS e coletar recursos além de construir edificações. Esta unidade, presente na grande parte dos jogos de RTS, é normalmente a principal relacionada com o gerenciamento de recursos desse tipo de jogo. O estudo de Shoemaker (Shoemaker, 2006) trata do problema da IA do cidadão que deve ter o mínimo de interferência do jogador. Para isso o artigo define o que chama de cidadão inteligente (*smart citizen*) que é um cidadão diferente do comum nos jogos de RTS. O **cidadão inteligente** toma decisões que diminuem a interferência do jogador e deixa-o livre para tomar decisões mais complexas – ou seja, parte da gerenciamento de recursos é tratada pela própria IA do jogo. Dentre algumas coisas que esse cidadão faz há as seguintes: quando esse cidadão constrói uma estrutura coletora de recursos de ouro, por exemplo, ele prontamente procura a mina de ouro mais próxima e já inicia a coleta desse recurso sem esperar pela ordem explícita do jogador; ou, assim que o cidadão inteligente está desocupado e procura maneiras de ajudar seu time, por exemplo, auxiliando na construção de alguma estrutura próxima. Esse esquema apresenta alguns benefícios e é relativamente simples de implementar novas ações e gatilhos estendendo a FSM (acrônimo para Máquina de Estados Finita, do inglês *Finite State Machine*) básica. Apesar de nos parecer interessante deixar parte da gerenciamento de recursos por conta da IA nativa do jogo, entendemos que alguns inconvenientes podem surgir do fato de ter menos controle sobre as ações do *smart citizen*. Além do que, esse aspecto de controle parcial do cidadão pode degradar o nível dos jogadores mais experientes, visto que alguns comportamentos serão automatizados.

3.3 IA com foco no gerenciamento de recursos em RTS

Nesta seção o foco é abordar as pesquisas que têm foco em aspectos relacionados a gerenciamento de recursos propriamente dito. Entretanto, uma grande parte das

pesquisas que estudamos e que tangem a gestão de recursos está muito relacionada a um conceito chamado de *build-order* que é algo um pouco mais específico do que gerenciamento de recursos no sentido amplo.

Uma grande parte dos módulos de IA dos jogos digitais atuais são desenvolvidos como um complexo sistema de *scripts* baseado em regras. Especificamente os *game designers* tentam prever com todos os possíveis cenários o que os personagens de IA podem encontrar e assim escrevem o *script* de ações para todas as situações. Esses sistemas são de difícil manutenção e expansão. Além disso, é impossível prever e planejar efetivamente todos os futuros potenciais estados do jogo. A maioria dos jogos não tem a capacidade de lidar com situações não familiares (Kovarsky & Buro, 2006). Nesses casos o comportamento da IA é normalmente ineficiente e/ou bastante previsível, o que enfraquece a qualidade da experiência do jogador humano. Acredita-se que adicionar capacidades de planejamento e gerenciamento em tempo real aos jogos de computador resulta em um comportamento de IA melhorado. Um bom exemplo é o estudo de Orkin (Orkin, 2005) que por meio do jogo FEAR mostra efeitos positivos que o planejamento automático pode ter na performance dos personagens de IA. O estudo não trata propriamente de gerenciamento de recursos, porém é útil para mostrar como em FEAR, diferente de outros jogos de ação, a IA executa planejamento em tempo real e adapta o comportamento dos personagens à situação corrente.

Build Order

Os pesquisadores Churchill e Buro (Churchill & Buro, 2011) definem *build-order* como sendo a ordem na qual unidades e estruturas são produzidas num jogo de RTS. Esse conceito é estreitamente relacionado ao conceito de gerência de recursos. A difícil tarefa de elaborar uma *build-order* depende, além de outras coisas, dos recursos existentes e dos que estão por serem captados.

Entendemos que *build-order* é na verdade uma simplificação do gerenciamento de recursos, pois tem, a priori, a ideia de algo estático, quando um jogo de RTS é completamente dinâmico. Entendemos que a ordem na qual as entidades são criadas num jogo de RTS deve ser contextualizada e a criação de uma lista ordenada – que em suma é o *build-order* – descarta o contexto do jogo. O nosso estudo em gerenciamento de recursos engloba a tarefa de elaborar a *build-order* o que torna nosso trabalho ainda

mais desafiador. Talvez seja interessante para um trabalho futuro a comparação das duas abordagens: uma seria o *build-order* puro e a outra seria a nossa abordagem. O fato é que há bastante espaço para pesquisas nesta área.

No estudo intitulado de “*A first look at build-order optimization in real-time strategy games*” de Alex Kovarsky e Michael Buro (Kovarsky & Buro, 2006) é focado na otimização de *build-order* em jogos de RTS. A pesquisa tem como objetivo otimizar a coleta de recursos e a criação de unidades e edificações nos estados iniciais dos jogos de RTS. São considerados dois tipos de problemas de otimização: minimizar o tempo para atingir um certo objetivo, por exemplo, construir dois tanques e cinco soldados, ou maximizar a quantidade de recursos em um tempo específico, por exemplo, maximizar a coleta de ouro em dez minutos. O problema de otimização de *build order* apresenta uma série de desafios que ainda não foram previamente pesquisados nos estudos de planejamento. Um dos problemas é a criação de novos itens que podem atuar no mundo. Outro é a execução concorrente de ações em jogos de RTS que pode levar a problemas como determinar se um conjunto de ações é executável em concorrência, e gerar esses conjuntos de ações concorrentes de maneira eficiente. No restante do artigo, o problema é formulado no domínio da linguagem PDDL (do inglês *Planning Domain Definition Language*, (Ghallab, Aeronautiques, Isi, & Wilkins, 1998)). O estudo modela bem a especificação do problema de *build-order* com PPDL, porém não há testes nem coleta de dados para análise e produção de resultados de seu modelo. Os autores propõem que estudos futuros validem questões “como lidar com criação e destruição de objetos em PPDL” e argumenta que apenas lançaram novas ideias sobre maneiras de lidar com esses problemas. Agregamos ao nosso sistema alguns conceitos do modelo de otimização de *build-order* desse estudo, particularmente no módulo que decide qual objeto criar quando há requisitos de outros objetos ou recursos.

Já os autores Branquinho e Lopes focaram num aspecto diferente, porém muito importante para o sucesso do gerenciamento de recursos, o processo de coleta de recursos (Branquinho & Lopes, 2010). Primeiramente um planejador de ordem parcial é proposto para determinar quais ações devem ser executadas, entretanto este método não especifica o tempo das ações. Então um processo de busca em tempo real e aprendizado com o uso de uma heurística admissível para agendamento de ações é aplicado, determinando o tempo exato do plano. O agendamento é baseado no algoritmo de busca e aprendizado A* (SLA*). Desconsiderando o fato de o SLA* necessitar de mais tempo

para agendar alguns planejamentos, ficou claro que uma característica muito importante para o planejamento em tempo real em um ambiente dinâmico é a do aprendizado. Esse estudo combina duas técnicas, MeaPop (MEA com planejamento de ordem parcial) e o algoritmo de busca e aprendizagem A* (SLA*). Os resultados do estudo melhoram os intervalos de tempo gerados pelo MEA, porém requerem muito mais tempo para realizar a computação, o que impede de ser utilizado em busca de tempo real. Os autores estão pesquisando novas maneiras de melhorar o tempo de execução do SLA*.

Assim como no xadrez, nos jogos de RTS os jogadores iniciantes precisam estudar e praticar a execução de jogadas ordenadas (uma analogia ao *build-order* do universo dos jogos RTS) e adaptá-las aos adversários específicos. Para evitar essa tarefa complexa e ambiciosa, o estudo de Churchill e Buro “*Build order optimization in starcraft*” foca na otimização do ordenamento dos investimentos usando como base o jogo *Starcraft* da empresa *Blizzard Entertainment*, assumindo que a ordem já é dada (Churchill & Buro, 2011). Agir rápido é muito importante num jogo de RTS devido ao fato de que os jogadores atuam de forma assíncrona. Então o objetivo do jogo é achar a sequência de ações que atingem o objetivo enquanto minimiza-se o intervalo temporal do plano. Esse processo é chamado de *build-order optimization*. A nossa experiência mostrou que a otimização de um plano de criação de unidades é determinante para o sucesso do jogador. Um plano fraco no início do jogo comumente leva a uma rápida derrota.

No estudo “*Online planning for resource production in real-time strategy games*” os autores Chan, Fern, Ray, Wilson e Ventura empregam *means-ends analysis* (MEA) para gerar planos de *build-order*, seguidos por uma fase de heurística de reagendamento que tenta minimizar o intervalo como um todo. Mesmo produzindo planos satisfatórios eles não são ótimos devido à complexidade da natureza do problema de reagendamento (Chan, Fern, Ray, Wilson, & Ventura, 2007a). Em alguns casos eles são capazes de vencerem os intervalos gerados por jogadores humanos, mas não mencionam o nível de experiência desses jogadores. Essa técnica é estendida em outro estudo “*Extending online planning for resource production in real-time strategy games with search*” dos mesmos autores, incorporando o algoritmo de busca *best-first* em uma tentativa de reduzir o intervalo de tempo posterior resolvendo subobjetivos intermediários (Chan, Fern, Ray, Wilson, & Ventura, 2007b). O estudo admite que faltam a seu algoritmo de busca várias otimizações, e seus resultados mostram que não é só mais lento que o do trabalho anterior mas que este não pode produzir soluções significativamente melhores.

O estudo de Josh MCCoy e Michael Mateas trata do problema de *build order* de forma *ad-hoc*, de forma que a construção de unidades de exército, o desenvolvimento tecnológico e a construção de edificações em são feitas por diferentes “estratégias” (MCCoy & Mateas, 2008). Nesse estudo, os autores utilizaram a plataforma de RTS *Wargus* que é usada em diversos estudos da área. Em cada momento do jogo somente uma estratégia está habilitada iniciando com uma estratégia de abertura que eles chamam de *InitialStrategy* responsável pelo crescimento inicial da economia e da produção militar. A estratégia que segue após esta é a *TierStrategy* que é a responsável somente pelo crescimento militar que constrói um soldado para cada trabalhador criado. De forma *ad-hoc* essa estratégia é responsável por construir dois *barracks* e um *blacksmith* que são edificações militares. Após a construção sequencial dessas unidades, dois soldados são construídos para cada trabalhador criado.

A Figura 8 exibe os blocos azuis que são os gestores e suas relações. As caixas tem os nomes abreviados onde TM é gestor tático, RM é gestor de reconhecimento, SM é o gestor de estratégia de alto nível, IM é o gestor de alocação de recursos, PM é o gestor de produção de unidades. Na parte inferior da figura são entidades de jogo encontrados em *Wargus*. Dos pesquisados esse é o estudo que mais se aproxima do nosso e tiramos conceitos interessantes como a proporcionalidade das construções das unidades de um jogo de RTS e utilizamos no nosso módulo de IA de gerenciamento de recursos. Fazendo analogia com este último estudo, pode-se dizer que nosso estudo terá os módulos PM (responsável pela produção de unidades), IM (responsável por gerenciar *workers* na construção das unidades) e uma parte do SM (responsável pela estratégia de alto nível, porém só relacionada a gerenciamento de recursos). Um dos problemas que enxergamos é a quantidade de ações *ad-hoc* empregadas no agente modelado por esse estudo, o que dificulta a validação deste modelo em outros ambientes.

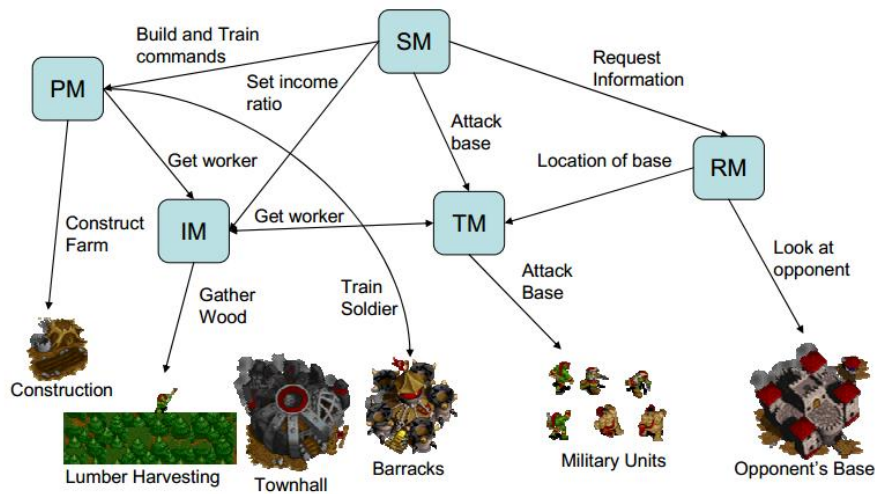


Figura 8 Agentes integrados

Aproveitamos muitas ideias e conceitos das pesquisas listadas neste capítulo. Uma das contribuições está em (Harmon, 2002) que utiliza o conceito de Utilidade nos jogos de RTS, outra em (MCCoy & Mateas, 2008) que integra vários agentes com a finalidade de jogar um RTS sem a interferência humana e acaba por utilizar proporcionalidade de unidades – que é um conceito que o nosso sistema utiliza – além da arquitetura do seu sistema que serve de inspiração para o nosso. Os conceitos aprendidos de *build-order* presentes nos estudos (Kovarsky & Buro, 2006), (Churchill & Buro, 2011) também foram de grande importância e os agregamos ao nosso sistema.

Capítulo 4

4 PICFlex: uma abordagem de gerenciamento de recursos baseada em Política de Investimento Contextual e Flexível

O problema era de tal maneira complexo que tivemos que assumir um conjunto de aplicações de abstrações e reformulações para assim conseguirmos fazer simplificações. O nosso problema trata questões de tomada de decisão multicritério em tempo real. O número de estados possíveis num ambiente finito, dinâmico e parcialmente visível é potencialmente muito grande. A combinação de todas as variáveis em todos os estados que estas podem assumir resulta em um conjunto de possíveis estados enorme o que torna esse problema intratável pelas soluções baseadas puramente em busca exaustiva (Cunha & Chaimowicz, 2010). Criar regras para tratar cada uma das possibilidades e as nuances de contexto não parece ser razoável, pois além da dinamicidade do ambiente há também o fato de este ser parcialmente observável, o que traz o fator incerteza. Além disso, o ambiente poderá ter restrições de tempo – que poderiam inclusive ser restrições de tempo real – as quais possivelmente seriam desrespeitadas devido à quantidade de possibilidades que seriam analisadas. Então, para superar os problemas citados anteriormente, e para simplificar a tomada de decisão do módulo investidor, um nível mais alto de abstração se mostrou essencial.

Tudo que foi dito até agora serve para perceber que a tarefa de gerenciar os recursos no ambiente de um RTS pode ser muito complexa. Visando encontrar uma solução, adotamos uma abordagem comum na Inteligência Artificial nestes casos, que é a de reformular o problema simplificando-o e abstraindo os detalhes, para encontrar uma primeira solução que tenha em mente as principais características do problema. Só depois iríamos refinar a solução incluindo nuances. Este trabalho de modelagem nos levou a um conjunto de conceitos básicos como o de política de investimento, como descrevemos neste capítulo.

4.1 Abstração e reformulação

Em (Holte & Choueiry, 2003) analisando a resolução humana eficiente e criativa de problemas ficou claro que o sucesso frequentemente deveu-se a olhar o problema de diferentes pontos de vista. Um solucionador naturalmente inicia com o ponto de vista sugerido pela declaração do problema, porém se esse não levar ao sucesso, um bom solucionador de problemas mudará o ponto de vista do problema para uma forma mais perspicaz. George Polya diz em seu livro *How to solve it* (Polya, 1945) que muitas técnicas são do tipo: troque um problema pela sua generalização, introduza algum elemento auxiliar e adote uma boa notação. Em muitos casos a solução, ou não solução, do problema modificado dá diretamente a resposta ao problema original, enquanto que em outros casos a solução do problema modificado dá a compreensão necessária para a solução do problema original ou um guia de como ele deve ser resolvido.

Herbert Simon argumentou que existe uma íntima conexão entre reformulação e resolução de problemas:

"All mathematics exhibits in its conclusions only what is already implicit in its premises [...] Hence all mathematical derivation can be viewed simply as change in representation, making evident what was previously true but obscure. This view can be extended to problem solving—solving a problem simply means representing it so as to make the solution transparent" (Simon, 1981)

O trabalho de Giunchiglia e Walsh bastante citado no campo da abstração (Giunchiglia & Walsh, 1992), define abstração como sendo um processo de mapear uma representação inicial de um problema. Esse mapeamento é construído escondendo os detalhes, para simplificar a exploração do espaço preservando algumas "propriedades" desejáveis para o mapeamento reverso da solução abstrata. Segundo Zucker,

"An abstraction is a change of representation, in the same formalism, that hides details and preserves desirable properties" (Zucker, 2003).

A definição de Zucker é mais restritiva do que a proposta por Giunchiglia e Walsh (Giunchiglia & Walsh, 1992), uma vez que requer que o formalismo permaneça inalterado na transformação. A utilidade dessa restrição é melhor em distinguir a essência de uma abstração da noção geral de mudança de representação. No entanto,

frequentemente uma abstração e uma mudança de formalismo são combinadas em uma mudança de representação. De fato, abstração torna possível alterar a quantidade de informação representada, permitindo uma reformulação menos expressiva ou um formalismo mais simples ser possível.

Neste contexto, o termo reformulação é restrito a caracterizar precisamente uma mudança somente do formalismo de representação. Como diz Zucker,

“A reformulation is a change of representation, from one to another formalism, preserving the quantity of information involved”(Zucker, 2003).

Posto que a abstração e reformulação sejam ferramentas úteis na resolução de problemas, decidimos aplicar essas técnicas ao problema de gerenciamento de recursos em um jogo de RTS. Procuramos abstrair o problema ao nível do problema de gerenciamento de recursos em um ambiente genérico com as principais características de um jogo RTS: ambiente finito, dinâmico, parcialmente visível e de tempo real.

4.2 Política de investimento

O conceito central do nosso modelo é *política de investimento*, em torno do qual se articulam outros conceitos como o de classe de investimento, contexto de investimento, flexibilidade da política, demanda de investimento, etc.

Damos o nome de **política de investimento**, o padrão de gastos dos recursos segundo o tipo da demanda. Formalmente, uma política de investimento P é dada por:

$$P = \{[c_1, t_1], [c_2, t_2], \dots, [c_n, t_n]\}, \quad \text{onde } \sum_{i=1}^n t_i = 100\%$$

onde c_n é a n -ésima classe de investimento e t_n é a meta de referência para o investimento na n -ésima classe de investimento. Em outras palavras, os valores percentuais representam metas de gastos para cada classe de investimento.

Classes de investimento

Nomeamos **classes de investimento** mais comuns nos jogos de RTS: *Army*, a classe que se refere aos exércitos, *Building*, a classe correspondente às edificações, *Upgrade*, que representa os investimentos em melhoramento da artilharia, defesa e *Tech* para melhorar a tecnologia. Em particular, no jogo *Starcraft* as classes *Upgrade* e *Tech* são tratadas de forma diferente, mas, a diferença entre as duas classes pode não ser tão clara em outros jogos. Para facilitar a solução, este trabalho trata ambas como sendo a mesma classe.

Para entender os dois conceitos, tomemos, por exemplo, $P = \{[Army, 40\%], [Building, 25\%], [Building, 35\%]\}$. Isto quer dizer que o jogador deve procurar, como meta, investir 40% dos recursos construindo unidades de exército, 25%, construindo prédios e 35% melhorando as tecnologias.

Escolha da política de investimento

O jogador humano experiente percebe as nuances do jogo com o decorrer da partida. O contexto geral muda: ora está defendendo seu “império” de um ataque adversário, ora está atacando o inimigo, ou até mesmo pode estar em momentânea paz. Em função do contexto jogador adota estratégias diferentes que se desdobram em comportamentos diferentes e políticas de investimento distintas. Por exemplo, as metas de investimentos em momentos de paz são normalmente diferentes daquelas quando o jogador está sob ataque ou realizando ataques contra os inimigos.

Logo, não é recomendável ter uma política única. Como o ambiente é dinâmico, o contexto muda e uma política de investimento adequada em um momento poder não sê-la mais em outro momento. É preciso escolher a melhor política segundo o contexto. em outras palavras, ter a metas para cada classe de investimento que sejam condizentes com a realidade atual. Isto nos levou a formular o conceito de **função de escolha de política de investimento** (ou simplesmente escolha de política), F , que formalmente é definida como

$$F(P, C) \rightarrow P'$$

onde P é a política de investimento atual, C é o contexto atual do jogo e P' é a nova política escolhida por F , sendo P' supostamente a mais adaptada ao novo contexto do jogo.

Criamos duas implementações da função F para utilizarmos nas nossas implementações do PICFlex. A primeira delas é a **simple** (Fs) que independente do contexto sempre retorna a mesma política, ou seja, a política de investimento nunca é alterada. Ela pode ser definida da seguinte forma:

$$Fs(P, C) \rightarrow P$$

A segunda implementação de F é mais sofisticada e analisa o contexto para escolher uma nova política de investimento mais adequada ao novo contexto, dentro de um conjunto de políticas pré-existentes e que foram identificadas como adequadas por especialistas em contextos específicos. Chamamos essa função de **adequate** (Fa), pois cria uma nova política de investimento que por sua vez adapta o PICFlex à nova realidade do jogo. Fa está definida abaixo

$$Fa(P, C, G) \rightarrow P'$$

onde P é a política de investimento atual, C é o contexto em questão, e G é um conjunto de políticas pré-existentes. Na prática, F permite um chaveamento de políticas.

É preciso salientar que é possível criar inúmeras versões da função F , mais sofisticadas que Fa , não precisando se apoiar em políticas pré-existentes. Porém, para o escopo deste trabalho e tendo em vista nossa filosofia de não complicar sem necessidade aparente, adotamos a função Fa como versão alternativa da versão ainda mais simples, a Fs .

Contexto

O contexto de um jogo é composto por variáveis que descrevem o estado corrente do jogo. Para o PICFlex, definimos somente três contextos que são: como "tempo de paz", "sob ataque", ou "realizando ataque". Sabemos que o contexto é algo bem mais complexo, porém para simplificar nossa implementação definimos somente os três contextos anteriores.

Estratégia de execução da política de investimento

A política de investimento irá estabelecer metas de gastos para cada classe de investimento, mas ela não diz como estas metas devem ser perseguidas. Entende-se que,

no longo prazo, a adoção da política proporcionará um "equilíbrio" nos gastos por classe segundo as metas propostas, mas o processo de perseguição destas metas pode não ser linear. Ora, como a meta pode funcionar como um teto de gastos, pode haver complicações na execução da política. Por exemplo, suponhamos que a meta de gastos com prédios seja de 25% dos recursos e que os gastos feitos nesta classe até o momento estejam em 23%. Se ficar evidente, no contexto do jogo, que seria preciso criar um novo prédio que elevaria os gastos percentuais com esta classe para 27%, o jogador deveria fazer ou não tal investimento? Se ele seguir estritamente a política, não deve fazê-lo, mas ele poderia construir o prédio, porque entende sua urgência e utilidade, e compensar o "estouro" da meta nos investimentos futuros, trazendo o percentual de gastos com prédios para o patamar de 25% como especificado na política.

Para explicitar a forma com que se deve ou pode-se perseguir as metas, introduzimos o conceito de estratégia de execução da política de investimento S , que formalmente é definida

$$S(P, D, H) \rightarrow \text{booleano}$$

onde P é a política de investimento atual, D é a demanda atual e H é o histórico de gastos realizados. O histórico dos gastos é uma lista que contém todos os gastos que já foram realizados pelo PICFlex.

A função S recebe uma demanda que pode ser a criação de um item de exército, ou o desenvolvimento de uma tecnologia qualquer, e deverá autorizá-la ou não em função do histórico de gastos.

Criamos diversas variações de S para utilizarmos nas nossas implementações do PICFlex. A função **generous** (St , generoso) sempre permite que a demanda seja realizada retornando o booleano verdadeiro para quaisquer que sejam a política de investimento, demanda e histórico de gastos. Essa estratégia acaba por anular os efeitos da política de investimento escolhida uma vez que dada uma demanda, esta será sempre autorizada independente do histórico de gastos ou política de investimento. Formalmente, St é definida por

$$St(P, D, H) \rightarrow \text{verdadeiro}$$

A segunda implementação de S é chamada de ***accept once*** (So , aceita uma vez) só permite que uma única demanda extrapole o teto da meta da classe de investimento, da segunda demanda (que extrapole o teto da meta) em diante serão negadas até que o equilíbrio de gastos seja reestabelecido.

$$So(P, D_1, H_1) \rightarrow \text{verdadeiro}$$

$$So(P, D_2, H_2) \rightarrow \text{falso}$$

...

$$So(P, D_n, H_n) \rightarrow \text{falso}$$

onde D_1, \dots, D_n extrapolam o teto da meta D_2, \dots, D_n são consecutivas a D_1 .

A terceira implementação de S que modelamos é chamada de ***rigid*** (Sr , rígida) e não permite que uma demanda, que extrapole o teto da meta de uma classe de investimento, seja atendida e é definida da seguinte forma:

$$Sr(P, D_1, H_1) \rightarrow \text{falso}$$

se o custo de criar D_1 extrapola o teto da meta de sua classe. Na prática, essa estratégia não permite que os gastos ultrapassem o teto das metas.

A quarta implementação de S que criamos é chamada de ***observed growth*** (Sg , crescimento observado) e é um pouco diferente que a anterior de modo que esta permite que as demandas que extrapolam o teto da meta de uma classe x sejam atendidas até que uma demanda de uma classe y seja rejeitada por extrapolar o seu teto. Definimos assim:

$$Sg(P, D, H) \rightarrow \text{booleano}$$

$$Sg(P, D_1, H_1) \rightarrow \text{verdadeiro}$$

$$Sg(P, D_2, H_2) \rightarrow \text{verdadeiro}$$

$$Sg(P, D_3, H_3) \rightarrow \text{falso}$$

$$Sg(P, D_4, H_4) \rightarrow \text{falso}$$

onde D_1 , D_2 , D_3 e D_4 são consecutivas e extrapolam o teto da meta e D_1 , D_2 e D_4 são da classe x porém D_3 é da classe y .

Demanda

Uma demanda para o PICFlex é uma necessidade por algum item de investimento e é definida como sendo

$$D = \{Objeto, Tempo de criação, Custo, Classe\}$$

onde o *Objeto* é algo que precisa ser criado, *Tempo de criação* é o tempo que o objeto leva para ser criado, *Custo* é o preço que se paga para criar o objeto e *Classe* é a classe de investimento a qual o objeto pertence.

Quando a demanda surge o investidor deve analisá-la e decidir se irá atender ou não e por motivos de simplificação, as dividimos em dois tipos: **perenes** e **circunstanciais**. Uma demanda perene é a demanda que acontece de forma constante, sem que nenhum evento extraordinário aconteça. Essa demanda é agendada e quando se momento chega o PICFlex irá analisar e decidir sobre se irá atende-la ou não. Para ilustrar um exemplo de uma demanda perene imaginem se caso esse nosso modelo fosse aplicado a uma prefeitura de um município, então uma demanda perene seria a coleta de lixo, a ronda policial nos bairros ou até mesmo a fiscalização da iluminação pública. O outro tipo de demanda, a circunstancial, acontece quando situações do ambiente criam alguma necessidade não agendada. Utilizando novamente o exemplo da prefeitura, um exemplo de uma demanda circunstancial seria a construção de uma nova escola pública quando a população estudante crescesse.

Podemos também dividir as demandas circunstanciais em demandas **internas** ou **externas**. As demandas internas são as que surgem por circunstâncias internas ao gestor de recursos como o exemplo da escola descrito no parágrafo anterior – nesse caso o Investidor é o prefeito. Já as demandas externas são originadas por alguma entidade externa. Para exemplificar utilizando a prefeitura, uma demanda externa seria a contratação de mais professores que fora determinada pela criação de uma lei federal – nesse caso o agente externo é o governo federal.

A maneira que o PICFlex trata a demanda leva em conta o tipo da demanda e a sua urgência. O PICFlex por alguma situação circunstancial e urgente pode achar que uma

demanda externa tem mais prioridade que uma interna, porém a situação mais comum é a de o PICFlex tratar as demandas internas com maior prioridade que as externas e tratar as demandas circunstanciais com maior prioridade sobre as perenes.

4.3 Arquitetura do PICFlex

Nesta seção descrevemos o módulo que fará as análises necessárias e por fim tomará as decisões. Para simulação do comportamento do PICFlex, utilizamos a plataforma de simulação do jogo *Starcraft Broodwar* disponibilizado pela empresa *Blizzard Entertainment* que torna possíveis os testes do módulo inteligente. Procuramos modelar o agente de maneira mais genérica possível de forma que possa ser traduzido para outros ambientes.

Como já foi dito anteriormente, o processo de tomada de decisão em um ambiente complexo envolve inúmeros aspectos que tornam a tarefa bastante complicada. Portanto, realizamos diversas observações em partidas de jogos de *Starcraft Broodwar* de jogadores experientes para identificar as técnicas que estes utilizavam e traduzi-las para a realidade do nosso PICFlex. O intuito dessas observações era identificar os aspectos do jogador humano experiente que levavam este a vitória. Entendemos que se o nosso PICFlex tivesse o comportamento inteligente similar ao do jogador humano sua taxa de sucesso poderia ser melhorada. Então, verificamos a frequente presença de alguns aspectos comportamentais, para os quais atribuímos os seguintes rótulos: **proporções adaptativas de classe, crescimento equilibrado por um padrão, regras de gatilho e posturas**. Estes rótulos serão descritos em detalhes a seguir.

4.3.1 Proporções Adaptativas de Classe

Identificamos que os jogadores obedecem a uma proporção de investimento de recursos entre as classes de investimento. Basicamente, uma proporção de investimento em uma classe específica é o quanto é recomendado investir em cada classe de investimento. Uma possível distribuição é a da Tabela 4. Essas proporções não são rígidas, de forma que se houver necessidade elas poderão ser modificadas e adaptadas de acordo com a demanda.

4.3.2 Item de investimento

Nós chamamos todas as entidades, que podem ser criadas nos jogos RTS, de **itens de investimento**. São eles os possíveis investimentos que o PICFlex poderá efetuar. Quando os investimentos são feitos e os itens são adquiridos, passa-se a ser chamado de “Objeto”. Portanto, podemos dizer que o Item de Investimento é uma classe e o Objeto é a instância dessa classe. Cada Objeto tem vida e comportamento próprios além de características ou propriedades as quais cada uma dessas características terá um valor atribuído. São esses itens que, através de suas características próprias, mudarão o estado atual do ambiente. Os itens de investimento serão agrupados em Classes de Investimento.

As propriedades essenciais e comuns a todos os itens de investimento são:

- Custo de criação – o valor em recursos monetários do item. Por exemplo: num ambiente de uma montadora de automóveis, o custo da criação de um carro é medido na unidade monetária do país.
- Tempo de criação – tempo que leva para ser criado. Por exemplo: um carro para ser construído (uma vez que há todos os recursos necessários) leva um mês.
- Objetos requeridos – o item tem uma lista dos objetos que devem existir previamente para ser possível a sua criação. Por exemplo: não é possível investir na criação de um carro se não houver o projeto deste carro.
- Construtor/criador – tipo do item que utiliza os recursos para criar o item em questão. Por exemplo: o funcionário montador utiliza as máquinas, o ferro e o aço para criar o carro; nesse caso o Construtor é o funcionário montador.
- Papel – o papel que o item desenvolve no ambiente. Por exemplo: o papel do carro é transportar pessoas ou objetos.

4.3.3 Item básico de investimento

Criamos o conceito de item básico de investimento para esse trabalho que seria o item de exército mais representativo. No jogo *Starcraft*, por exemplo, para a raça *Terran*, o *Marine* foi eleito como o mais representativo, pois além de ser o primeiro item de exército que pode ser criado, é pouco custoso e tem boas características de ataque. Esse conceito foi criado, pois é algo que será utilizado pelo PICFlex para tomar sua decisão

de crescimento do exército. Esse item básico será utilizado como referência para o crescimento equilibrado por um padrão como veremos a seguir.

4.3.4 Crescimento equilibrado por um padrão

Verificamos que os exércitos dos jogadores cresciam de forma equilibrada por um padrão de cada jogador. Cada jogador humano tinha um padrão de crescimento do seu exército e criavam unidades de acordo com esse padrão. Esse padrão determina que para cada X itens do tipo A são necessários Y itens do tipo B. Então sempre que esse a quantidade dos itens sair do padrão o PICFlex deverá agendar o investimento das unidades necessárias para o equilíbrio ser reestabelecido.

Para o PICFlex escolher qual item deve ser criado durante o jogo utilizamos a nossa política de investimento no subnível dos itens de exército. Sendo a política de investimento:

$$P = \{[c_1, t_1], [c_2, t_2], \dots, [c_n, t_n]\}, \quad \text{onde } \sum_{i=1}^n t_i = 100\%$$

aplicamos essa definição utilizando o conceito de classe aplicada aos diversos tipos de itens que podem ser criados num jogo de RTS. Nesse caso cn é a n -ésimo item possível de investimento – soldado, cavaleiro, arqueiro, são exemplos de cn – e tn é a meta de referência que deve ser perseguida pela política – o número relativo de itens do tipo c que deve existir no exército.

A função F de escolha da política de investimento utilizada para a sub política de investimento do nível da classe *army* (do inglês, exército) utilizada foi a Fs *simple* definida por:

$$Fs(P, C) \rightarrow P$$

ou seja, a sub política é fixa durante todo jogo. E a estratégia de perseguição das metas que utilizamos nessa sub política é a Sr *rigid* de forma que qualquer demanda cn que ultrapasse a meta de referência tn seja negada.

$$Sr(P, D_i, H_i) \rightarrow falso$$

onde D_i é uma demanda por cn que ultrapassa a meta de referência tn .

Para simplificar a criação deste padrão para o nosso PICFlex, elegemos um **item básico de investimento** do *Starcraft Broodwar* que foi o *Terran Marine* Figura 9. Esse item foi escolhido por ser um dos menos custosos e não ter pré-requisito (de unidades de exército) algum. Sendo assim o equilíbrio do crescimento do exército terá como base de referência o número de *Terran Marine* existentes no exército.



Figura 9 Terran Marines

Por exemplo: digamos que item básico de investimento seja o Terran Marine e que o padrão de crescimento seja determinado pelos valores da Tabela 1.

Crescimento equilibrado por um padrão	
<i>c</i>	<i>t</i>
<i>Firebat</i>	14,2%
<i>Goliath</i>	50,0%
<i>Medic</i>	16,6%

Tabela 1 Crescimento equilibrado por um padrão

Essa tabela significa que para cada Marine existente existe 14,2% de um *Firebat*, 50,0% de um *Goliath* e 16,6% de um *Medic*. Portanto para calcular a distribuição do exército temos:

1 *Terran Marine* = 14,2% e X *Terran Marine* = 100%, fazendo uma regra de três simples temos que X = 7 (consideramos somente números inteiros nesse resultado). Ou seja, para cada 7 *Terran Marine* o investidor criará 1 *Firebat* para reestabelecer o padrão de crescimento do exército.

4.3.5 Regras Gatilho

Observando os jogos, verificamos que diante de situações particulares os jogadores tomavam sempre o mesmo padrão de decisão. Por exemplo, quando o exército era pequeno e havia poucos itens de repositório de unidades (que servem de “casa” para as unidades de exército no *Starcraft Broodwar*), ou repositórios insuficientes, o jogador assim que podia ordenava a criação de mais repositórios. Regras gatilho são reativas e disparam eventos de agendamento de investimentos de acordo com situações específicas e acabam por criar as demandas circunstanciais.

Para exemplificar outra aplicação das regras de gatilho, suponhamos que o PICFlex verifica a necessidade de um item A, porém ele analisa a árvore tecnológica e percebe que A tem como pré-requisito a existência do item B; este é um cenário típico de uma regra de gatilho que poderia disparar um evento que alerta sobre a necessidade de um item B.

Itens pré-requisitos pela árvore tecnológica

Nos jogos de RTS, todas as unidades que podem ser criadas têm um conjunto de pré-requisitos que podem ser desde recursos ou outras unidades que devem ser obtidos antes de iniciar suas construções. A árvore que contém todas as unidades do jogo com seus respectivos pré-requisitos é chamada de árvore tecnológica (do inglês, *tech tree*) (Churchill & Buro, 2011). Sendo assim, os chamados Investimentos de itens pré-requisitos são os itens de investimento sem os quais não é possível construir outro item que se queira. Vale salientar que esse item pode ser desde uma unidade de exército (no *Starcraft* não há unidades de exército pré-requisitos para outras), quanto uma edificação como, uma *Factory* que é pré-requisito para *Armory*³, ou um item de tecnologia como *Vehicle Weapons*⁴, requer *Science Facility*⁵.

³ <http://starcraft.wikia.com/wiki/Armory>

⁴ http://starcraft.wikia.com/wiki/Vehicle_Weapons

⁵ http://starcraft.wikia.com/wiki/Science_facility

4.3.6 Postura

Criamos o conceito de **postura** que nada mais é do que um conjunto de valores atribuídos às propriedades de política de investimento. Em termos técnicos podemos dizer que **uma postura é uma instância de uma política de investimento**.

Para simplificar, criamos três posturas básicas e essenciais que o PICFlex pode assumir durante o jogo: **inicial**, **defensiva** e **agressiva**. Na prática a postura é como o PICFlex se comporta num dado momento jogo – podendo assumir outra política a qualquer momento. A postura da política de investimento é escolhida pela função de F que definimos anteriormente.

É a política de investimento que será determinante para o PICFlex assumir um comportamento mais agressivo e investir em um grande exército com a finalidade de tomar de assalto o oponente ou ficar mais defensivo fortalecendo suas edificações e unidades de defesa. Para esclarecer o que foi dito antes no conceito de postura, no que diz respeito à implementação, a postura chaveia um conjunto de valores de variáveis que definem um comportamento específico do PICFlex e do próprio exército, ou seja, especificam valores dos atributos das políticas de investimento. São as posturas:

Inicial Nessa postura o PICFlex obedece às regras *ad-hoc*. É a primeira postura assumida pelo PICFlex, pois no início do jogo há alguns investimentos que devem ser feitos com urgência para evitar a derrocada prematura do império por uma investida do oponente.

Defensiva Postura na qual o PICFlex tem como objetivo principal investir no crescimento e fortalecimento do exército construindo edificações para a produção das unidades de exército e realizando *upgrades/techs* que fortalecerão tais unidades e edificações. Essa postura é normalmente assumida em momentos de paz.

Agressiva Nessa postura o objetivo é aumentar o poderio do exército aumentando em número e poder de fogo. Um momento de guerra intensa contra o inimigo leva o PICFlex a assumir essa postura.

O diagrama de estados referente às posturas pode ser visto na Figura 10. As políticas de investimento conseguem simular o comportamento do jogador. As propriedades da política de investimento determinam as proporções de investimento nas classes de

investimento. Além disso, são as variáveis da política de investimento que determinam o intervalo de tempo no qual as rotinas responsáveis pelo crescimento do exército serão executadas (quanto mais curto o intervalo mais rápido o exército cresce, porém mais recursos são consumidos).

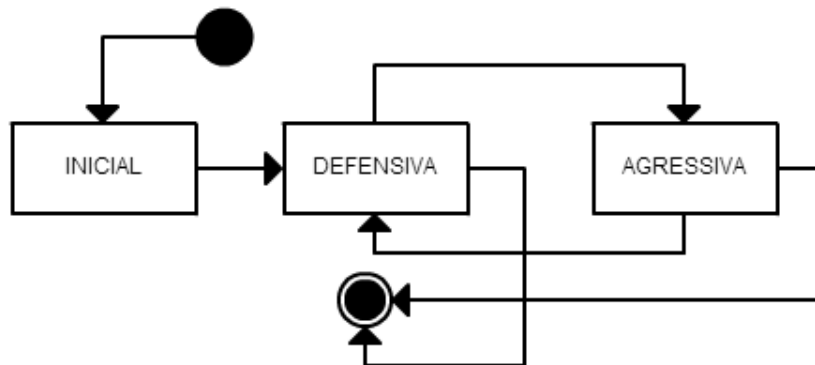


Figura 10 Diagrama de estados da postura

4.4 As seis implementações do PICFlex

Para avaliarmos as técnicas estudadas e desenvolvidas por nós, criamos seis implementações incrementais do PICFlex. As implementações foram chamadas de “*Player 1*” até “*Player 6*” e serão descritas a seguir em detalhes.

4.4.1 *Player 1* (player aleatório)

A primeira das implementações do PICFlex é completamente aleatória e seu comportamento, com relação ao gerenciamento de recursos é impossível de ser previsto. Não existe qualquer política de investimento em seu comportamento. O *player 1* não tem política de investimento nem há crescimento equilibrado por um padrão. O player aleatório também não tem a rotina que programa o crescimento do exército investindo no item básico a cada período de tempo.

Assim como todas as implementações que serão descritas, o *Player 1* inicia seu jogo com uma estratégia de abertura que é suficiente para não ser derrotado na partida logo no início do jogo – e eventualmente ganhar a partida somente através da estratégia de abertura, ou pelo acaso da sua aleatoriedade. E desta estratégia de abertura em diante, este *player* passa a tomar suas decisões de gerenciamento de recursos de forma aleatória e sem qualquer controle.

4.4.2 *Player 2* (player de política de investimento aleatória)

O segundo player foi criado com o intuito de avaliar nosso conceito de política de investimento. O *Player 2* é uma alteração do *Player 1*, descrito na seção anterior; as diferenças seguem abaixo:

Crescimento do exército baseado no item básico de investimento

Há uma rotina que a cada X rodadas de planejamento e execução dispara uma rotina que cria uma unidade de exército que seria o item eleito como sendo o mais básico do jogo (elegemos o *Marine*).

Política de investimento

A política de investimento é criada no início do jogo com seus valores atribuídos de forma randômica. A função de F utilizada é a Fs *simple* que não altera a política de investimento independentemente do contexto.

$$Fs(P, C) \rightarrow P$$

A estratégia de perseguição das metas S utilizada é a St *generous* que sempre permite o investimento independentemente da demanda ou histórico de gastos.

$$St(P, D, H) \rightarrow verdadeiro$$

Padrão de equilíbrio de crescimento

O padrão de equilíbrio de crescimento do exército desse *player* existe, porém é aleatório.

4.4.3 *Player 3* (player com política de investimento fixa definida por especialistas)

Seguindo com as alterações, o *Player 3* é uma alteração do anterior. Para avaliar o conceito de política de investimento, porém desta vez com os valores dos atributos da política sendo escolhidos por especialistas. Por exemplo, a distribuição que trouxe melhores resultados para esse *player* foi a da Tabela 2.

Melhor distribuição para o <i>Player 3</i>	
<i>c</i>	<i>t</i>
<i>Army</i>	45%
<i>Building</i>	33%
<i>Upgrade</i>	20%
<i>Tech</i>	12%

Tabela 2 Distribuição do Player 3

Este *player* utiliza a mesma função *Fs* que o *player 2* de forma que a política é fixa, porém a estratégia de perseguição das metas utilizada foi a *St rigid* que não permite nenhum gasto acima do teto da meta de classe.

$$Sr(P, D, H) \rightarrow falso$$

4.4.4 *Player 4* (player com crescimento equilibrado por especialistas)

O quarto player que criamos é idêntico ao *Player 3*, porém com a alteração de que o padrão de crescimento equilibrado do exército foi definido por especialistas e esse padrão pode ser visto na Tabela 5.

4.4.5 *Player 5* (player com política de investimento chaveada pelas posturas)

A única diferença entre o *Player 5* e o anterior é o fato de que para o *Player 4* (player com crescimento equilibrado por especialistas) a política de investimento é fixa, portanto seu comportamento é igual para qualquer situação do jogo. Entretanto, para o *Player 5* a política é alterada durante a partida (através da mudança de **postura**) e dessa forma o *player* assume comportamentos mais adequados aos diferentes contextos do jogo. A estratégia de perseguição das metas *S* utilizada continuou sendo a *Sr rigid*, porém a postura é escolhida pela função de escolha da política de investimento *Fa adequate* que analisa o contexto do jogo e retorna uma política de investimento mais aderente ao estado corrente.

$$Fa(P, C) \rightarrow P'$$

4.4.6 *Player 6* (player com política de gastos flexível)

O *player* desta seção é, mais uma vez, igual ao *player* anterior, porém com a diferença de que os gastos com as classes de investimento são coordenados pela estratégia de perseguição das metas *Sg observed growth*.

Essa estratégia permite que demandas ultrapassem o teto das metas de classes sejam atendidas até que alguma outra demanda de outra classe seja negada. Isso provê uma maior flexibilidade dos gastos fornecendo mais recursos a quem demanda mais.

$$Sg(P, D, H) \rightarrow \text{booleano}$$

<i>Player</i>	<i>Player 1</i>	<i>Player 2</i>	<i>Player 3</i>	<i>Player 4</i>	<i>Player 5</i>	<i>Player 6</i>
Crescimento baseado no item básico de investimento		X	X	X	X	X
Política de Investimento		X	X	X	X	X
Padrão de equilíbrio de crescimento		X	X	X	X	X
Política de investimento fixa definida por especialistas			X			
Crescimento equilibrado por especialistas				X	X	X
Política de investimento chaveada pelas posturas					X	X
Política de investimento de gastos flexível						X

Tabela 3 Comparativo das propriedades dos players

Na Tabela 3 é possível visualizar as propriedades de cada player de forma comparativa.

4.5 *Player 6* em detalhes

Esta seção tem como objetivo descrever como o PICFlex, na sua forma mais completa, se comporta – o *Player 6* é a implementação do PICFlex que agrega todos os conceitos citados nas seções anteriores.

Nesse trabalho tratamos o jogo de RTS no qual no início da partida o “império” vem com poucas unidades e recursos disponíveis na economia – muitas vezes só o suficiente para construir algumas poucas unidades do tipo *worker* (que capturam recursos e constroem edificações), como pode ser visto na **Error! Reference source not found.**. Então, logo que o jogo começa a postura atribuída é a chamada **inicial**. A postura inicial determina regras de gatilho para construção de unidades essenciais no início do jogo como, por exemplo, a construção de um Barracks, um *Academy*, vinte *Marines* e seis *Medics*. Esses investimentos são agendados, pois o oponente utilizado para avaliação da solução ataca o império coordenado pelo nosso *player* logo no início da partida. Caso esses investimentos não sejam feitos de forma *ad-hoc*, não haverá tempo para o PICFlex identificar essa necessidade e então perderá a partida na maioria das vezes. Como pode ser visto na **Error! Reference source not found.**, o PICFlex quando roda com

investimentos iniciais agendados de forma *ad-hoc* atinge melhores resultados se comparado com a mesma implementação do PICFlex, porém sem os investimentos iniciais. A Figura 11 exibe um histograma no qual o eixo horizontal a pontuação do PICFlex enquanto que o eixo vertical a frequência de ocorrência das pontuações. Através dessa análise, é fácil perceber que uma vitória só é possível com uma estratégia de abertura.

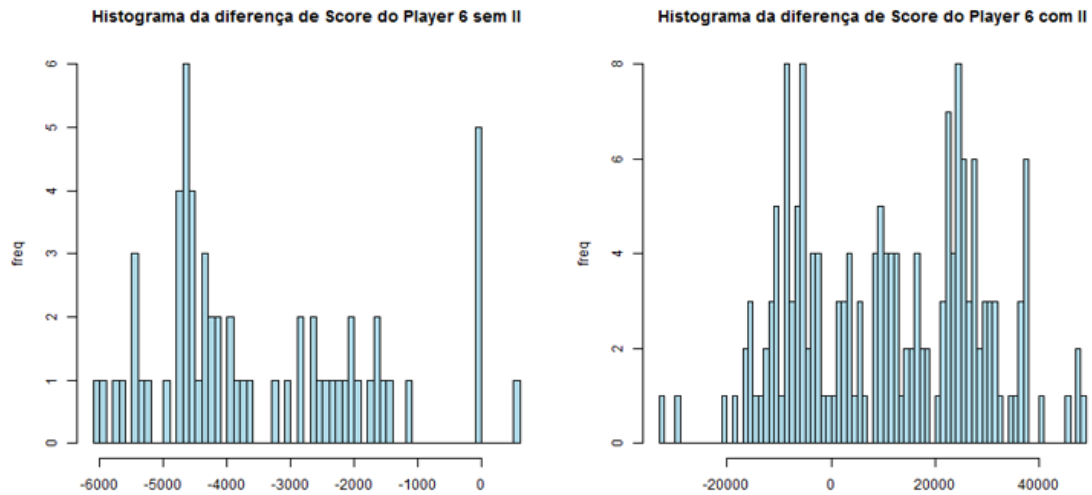


Figura 11 Comparando PICFlex sem e com Investimentos Iniciais

Quando o império do jogador atinge um nível de maturidade específico, definido por um conjunto de requisitos – que pode ser generalizado para qualquer jogo de RTS – o PICFlex assume a postura **defensiva**. Para o caso das simulações realizadas nesse trabalho, o conjunto de requisitos que, quando atingidos pelo estado do jogo, dispara o evento de mudança de postura é composto pela verificação da existência de dois tipos de edificações: *Barracks* e *Academy*; itens básicos para um exército se defender no início do jogo. Baseados nas observações dos jogadores humanos experientes, nós verificamos que quando são criadas essas duas edificações os jogadores assumem outro comportamento mais agressivo e passam a investir fortemente no crescimento do exército de ataque.

A postura defensiva tem como propósito do fortalecimento do império como um todo induzindo o PICFlex optar por criar várias edificações, treinar unidades de exército e desenvolver sua árvore de tecnologia. Uma boa distribuição das proporções de investimento para a postura defensiva é exibida na Tabela 4. Os investimentos em edificações e exército são contemplados com proporções um pouco maiores, pois é a

grande maioria dos investimentos possíveis nos jogos de RTS e, portanto demandam mais recursos.

Postura defensiva	
<i>c</i>	<i>t</i>
<i>Army</i>	30%
<i>Building</i>	35%
<i>Upgrade</i>	20%
<i>Tech</i>	15%

Tabela 4 Proporções de investimento em Classes

Crescimento equilibrado por um padrão

Como já foi explicado antes, na seção 4.3.4 Crescimento equilibrado por um padrão é o comportamento que rege o crescimento do exército com relação aos tipos de unidades que serão criadas e a quantidade de cada tipo. Mas para o exército crescer é necessário que exista uma regra de gatilho que dispara a criação do Item básico de investimento a cada X rodadas de planejamento.

```
rotina plano_de_crescimento
    fc recebe contador_de_frame
    se fc menos ultimo_contador_de_frame é maior que X então
        ultimo_contador_de_frame recebe fc
        chama rotina investir_em_item_basico
    fim do se
fim da rotina
```

Várias simulações foram executadas em busca da melhor configuração de proporções entre os itens de investimento e o item básico de investimento. Esses valores são o que determinam quais tipos de unidades serão criadas em questões de quantidade e podem ser vistos na Tabela 5.

Proporções relativas ao item básico	
<i>c</i>	<i>t</i>
<i>Firebat</i>	14,2%
<i>Goliath</i>	50,0%
<i>Medic</i>	16,6%
<i>Wraith</i>	25,0%
<i>Vulture</i>	50,0%
<i>Siege Tank</i>	33,3%
<i>Battlecruiser</i>	25,0%

Tabela 5 Proporções relativas ao Item Básico

Para exemplificar como o exército cresce com base no item básico (*Terran Marine*, nesse caso) temos que: a cada 7 *Marine* existentes, ou agendados para treinamento, 1 *Firebat* é agendado para ser criado – pois $100\% / 14,2\% = 7,02$; a cada 2 *Marine* 1 *Goliath* é agendado para ser criado – uma vez que $100\% / 50\% = 2,0$ – e assim por diante. Dessa forma o exército continua a crescer até que se atinja o número máximo de unidades por império, pois a maioria dos jogos de RTS tem esse limite, ou acabem os recursos. Enquanto esse crescimento acontece, a postura continua sendo defensiva até que o evento de mudança de postura seja disparado.

O jogo se segue e itens de investimento são alocados e os investimentos são realizados. O império cresce e à medida que ele cresce alguns critérios são analisados no ambiente corrente para a mudança de postura. A regra que dispara a mudança da postura corrente que é a defensiva para postura **agressiva** – o único estado possível partindo da postura defensiva – é descrita da seguinte forma:

```

rotina deve_engajar
    prop_att recebe tamanho_esquadrao_ataque vezes 100 dividido por contagem_total_unidades
    se prop_att é maior que 50 então
        postura recebe 'agressiva'
    fim do se
fim da rotina

```

Adotamos um critério simples para determinar o momento de atacar o oponente, pois essa análise é bastante complexa e não é o foco deste estudo. Então para esse fim a função **deve_engajar** foi criada. Essa analisa se o exército deve atacar ou continuar com seu comportamento corrente. Basicamente verifica se a proporção do número de unidades do esquadrão de ataque com relação ao número de todas as unidades do império é maior que 50%. Dessa forma, o PICFlex percebe o momento certo de atacar o

inimigo e, além disso, assume a postura agressiva, que é a postura mais indicada para momentos de ataque ao inimigo.

O império que já passou pela postura inicial e defensiva e pôde investir em tecnologia, em edificações, algum exército e agora poderá então investir intensamente em aumentar o exército de ataque que já será criado de maneira fortalecida – pelas tecnologias desenvolvidas anteriormente – além de crescer de forma mais acelerada pelas edificações construídas nos tempos das posturas anteriores.

Quando o PICFlex assume a postura agressiva esse terá mais recursos para investir em exército – em termos proporcionais, pois como já foi dito, a postura “chaveia” os atributos da política de investimento. Por exemplo, quando a postura assumida pelo PICFlex é a **agressiva**, a classe de investimento *army* terá mais quota de investimento que todas as outras classes (*buiding*, *tech* e *upgrade*), o que permitirá maiores quantias de recursos a serem gastas nessa classe.

4.6 Plataforma de simulações e testes

Para implementar a solução e executar os testes em computadores digitais precisávamos de uma plataforma que tornasse possível a criação de módulos de IA e os testes para a análise dos resultados. A BWAPI (*BroodWar* API) é um framework em C++ de código fonte livre para a criação de módulos de inteligência artificial para *Starcraft Broodwar*. Usando BWAPI, os programadores podem obter informações sobre os jogadores e as unidades individuais de *Starcraft*, bem como emitir uma grande variedade de comandos de unidades, abrindo as portas para IAs com algoritmos personalizados. Essa API vem sendo utilizada para estudar os aspectos de IA dos jogos de RTS (Buro & Churchill, 2012).

Para que pudéssemos testar o nosso módulo inteligente, nós precisávamos construir um *bot* para que este jogasse diversas partidas de *Starcraft* sem a interferência humana e coletar os resultados ao termino das partidas. No próprio site da BWAPI existem diversos *bots* disponíveis ⁶. Decidimos utilizar o *bot* de um pesquisador da área de

⁶ Uma lista de *bots* que utilizam a BWAPI pode ser encontrado em <https://code.google.com/p/bwapi/wiki/Projects>

Inteligência Artificial em jogos de RTS, o BTHAI⁷, do Ph.D. Johan Hagelbäck (Hagelbäck, 2012).

O ambiente de desenvolvimento foi um PC rodando o sistema operacional Microsoft Windows 7 e a IDE de desenvolvimento precisava ser o Microsoft Visual Studio 2008, pois é uma restrição da BWAPI na versão que utilizamos. Instruções de instalação do *Starcraft BroodWar* bem como a injeção do módulo personalizado de IA dentro do jogo estão disponíveis no site do BWAPI.

A versão do BWAPI utilizada foi a 3.7.2⁸ e por mais que esse framework venha sendo constantemente atualizado pelos mantenedores do seu código fonte, ainda contém diversos *bugs* assim como ausência de acesso a algumas informações importantes do jogo como a pontuação detalhada dos jogadores, o que dificultou a implementação da simulação.

⁷ Informações sobre o BTHAI podem ser encontradas em <http://code.google.com/p/bthai>

⁸ A API em C++ *StarCraft BroodWar* Interface (BWAPI) pode ser encontrada em <http://code.google.com/p/bwapi>

Capítulo 5

5 Resultados

O objetivo deste capítulo é apresentar os resultados e será feita uma análise sobre eles, tanto individual quanto coletiva.

5.1 Objetivos

Fizemos experimentos sobre as nossas implementações do PICFlex para saber se as técnicas utilizadas melhoram ou não o desempenho da tarefa de gerenciar recursos.

5.2 Protocolo

Na seção 4.4 nós descrevemos seis implementações do PICFlex que foram criadas e são esses que executarão o gerenciamento de recursos dentro do jogo. Então, pusemos os nossos players um a um para combater a IA nativa do jogo *Starcraft Broodwar* e coletamos os dados de 150 partidas para cada *player*. Os dados coletados para análise foram:

1. Pontuação da partida do *player*
2. Pontuação da partida da IA oponente nativa

No *Starcraft Broodwar* não existem níveis de dificuldade (como fácil, normal e difícil) acessíveis para o jogador selecionar como em outros jogos. Além disso, todos os nossos *players* foram implementados utilizando a raça *Terran*.

5.3 Apresentação

Nesta seção os dados coletados dos seis *players* serão expostos, porém só serão analisados na seção 5.4.

5.3.1 Comparativo entre as implementações do PICFlex

Abaixo segue a Tabela 6 que faz a comparação entre as implementações do PICFlex. Nessa tabela há as informações das amostras que foram coletadas para cada *player* assim como o número de vitórias e derrotas. Taxa de sucesso (**T. Sucesso**) que é o percentual de vitórias em comparação com o número de amostras. Média da diferença de pontuação (**M. Dif. Score**) onde a diferença de pontuação é a pontuação do *player* subtraída da pontuação do oponente. E, por fim, o desvio padrão calculado para essa diferença de pontuação (**DP. Dif. Score**).

Agente	Vitórias	Derrotas	T Sucesso	M Dif Score	DP Dif Score
Player 1	23	127	15,33%	-1851,92	14536,95
Player 2	4	146	2,67%	-1863,62	6468,54
Player 3	31	119	20,67%	1839,42	12918,82
Player 4	38	112	25,33%	2389,47	21559,54
Player 5	40	110	26,67%	7062,02	14640,15
Player 6	63	87	42,00%	10801,01	17343,73

Tabela 6 Comparativo dos players. T. Sucesso é Taxa de Sucesso. M. Dif. Score é a média da diferença entre o score do player e o score do inimigo. DP Dif Score é o desvio padrão do M. Dif. Score

5.3.2 Normalidade dos dados

Para averiguar a normalidade dos dados executamos o *Shapiro-Wilk Normality Test*, sobre a diferença da pontuação dos *Players* e dos seus oponentes, e os resultados foram:

- *Player 1*: p-value = 2,219e-06
- *Player 2*: p-value < 2,2e-16
- *Player 3*: p-value = 4,181e-12
- *Player 4*: p-value = 0.0002642
- *Player 5*: p-value = 5,937e-08
- *Player 6*: p-value = 0.001167

O que sugere que os dados não têm distribuição normal. Dessa forma, testes de hipótese que assumem a normalidade dos dados, como o *t-Test*, não são apropriados.

5.3.3 Testes de Hipótese

Sendo todos os nossos dados de distribuição não normal, então utilizamos para os testes de hipótese das próximas seções o *Wilcoxon signed-rank test*, o não assumi que os dados comparados têm distribuição normal.

5.3.3.1 Player 2

Sendo **P1** a pontuação do *Player 1* e **P2** a pontuação do *Player 2*, estabelecemos a seguinte hipótese:

- Hipótese Alternativa **Ha1: $P2 > P1$**
- Hipótese Nula **Hn1: $P2 = P1$**

Executamos o *wilcox-Test* para um $\alpha = 0.05$ e os resultados foram um o p-value = 0,0004065 o que significa que a hipótese nula **Hn1** pode ser rejeitada.

5.3.3.2 Player 3

Sendo **P3** a pontuação do *Player 3*, estabelecemos a seguinte hipótese:

- Hipótese Alternativa **Ha2: $P3 > P2$**
- Hipótese Nula **Hn2: $P3 = P2$**

Executamos o *wilcox-Test* para a hipótese **Ha2** um $\alpha = 0.05$ e os resultados foram um o p-value = 0,9698 o que significa que a hipótese nula **Hn2 não** pode ser rejeitada.

- Hipótese Alternativa **Ha3: $P3 > P1$**
- Hipótese Nula **Hn3: $P3 = P1$**

Executamos o *wilcox-Test* para a hipótese **Ha3** um $\alpha = 0.05$ e os resultados foram um o p-value = 0,001898 o que significa que a hipótese nula **Hn3** pode ser rejeitada.

5.3.3.3 Player 4

Sendo **P4** a pontuação do *Player 4*, estabelecemos a seguinte hipótese:

- Hipótese Alternativa **Ha4: $P4 > P3$**
- Hipótese Nula **Hn4: $P4 = P3$**

Executamos o *wilcox-Test* para a hipótese **Ha4** um $\alpha = 0.05$ e os resultados foram um o p-value = 0,4412 o que significa que a hipótese nula **Hn4 não** pode ser rejeitada.

- Hipótese Alternativa **Ha5: $P4 > P2$**
- Hipótese Nula **Hn5: $P4 = P2$**

Executamos o *wilcox-Test* para a hipótese **Ha5** um $\alpha = 0.05$ e os resultados foram um o p-value = 0,6311 o que significa que a hipótese nula **Hn5** não pode ser rejeitada.

- Hipótese Alternativa **Ha6: $P4 > P1$**
- Hipótese Nula **Hn6: $P4 = P1$**

Executamos o *wilcox-Test* para a hipótese **Ha6** um $\alpha = 0.05$ e os resultados foram um o p-value = 0,01789 o que significa que a hipótese nula **Hn6** pode ser rejeitada.

5.3.3.4 Player 5

Sendo **P5** a pontuação do *Player 5*, estabelecemos a seguinte hipótese:

- Hipótese Alternativa **Ha7: $P5 > P4$**
- Hipótese Nula **Hn7: $P5 = P4$**

Executamos o *wilcox-Test* para a hipótese **Ha7** um $\alpha = 0.05$ e os resultados foram um o p-value = 0,003252 o que significa que a hipótese nula **Hn7** pode ser rejeitada.

- Hipótese Alternativa **Ha8: $P5 > P3$**
- Hipótese Nula **Hn8: $P5 = P3$**

Executamos o *wilcox-Test* para a hipótese **Ha8** um $\alpha = 0.05$ e os resultados foram um o p-value = 5,53e-05 o que significa que a hipótese nula **Hn8** pode ser rejeitada.

- Hipótese Alternativa **Ha9: $P5 > P2$**
- Hipótese Nula **Hn9: $P5 = P2$**

Executamos o *wilcox-Test* para a hipótese **Ha9** um $\alpha = 0.05$ e os resultados foram um o p-value = 3,81e-08 o que significa que a hipótese nula **Hn9** pode ser rejeitada.

- Hipótese Alternativa **Ha10: $P5 > P1$**
- Hipótese Nula **Hn10: $P5 = P1$**

Executamos o *wilcox-Test* para a hipótese **Ha10** um $\alpha = 0.05$ e os resultados foram um o p-value = 6,331e-10 o que significa que a hipótese nula **Hn10** pode ser rejeitada.

5.3.3.5 Player 6

Sendo **P6** a pontuação do *Player 6*, estabelecemos a seguinte hipótese:

- Hipótese Alternativa **Ha11: P6 > P5**
- Hipótese Nula **Hn11: P6 = P5**

Executamos o *wilcox-Test* para a hipótese **Ha11** um $\alpha = 0.05$ e os resultados foram um o p-value = 0,01637 o que significa que a hipótese nula **Hn11** pode ser rejeitada.

- Hipótese Alternativa **Ha12: P6 > P4**
- Hipótese Nula **Hn12: P6 = P4**

Executamos o *wilcox-Test* para a hipótese **Ha12** um $\alpha = 0.05$ e os resultados foram um o p-value = 7,888e-05 o que significa que a hipótese nula **Hn12** pode ser rejeitada.

- Hipótese Alternativa **Ha13: P6 > P3**
- Hipótese Nula **Hn13: P6 = P3**

Executamos o *wilcox-Test* para a hipótese **Ha13** um $\alpha = 0.05$ e os resultados foram um o p-value = 3,737e-07 o que significa que a hipótese nula **Hn13** pode ser rejeitada.

- Hipótese Alternativa **Ha14: P6 > P2**
- Hipótese Nula **Hn14: P6 = P2**

Executamos o *wilcox-Test* para a hipótese **Ha14** um $\alpha = 0.05$ e os resultados foram um o p-value = 1,415e-12 o que significa que a hipótese nula **Hn14** pode ser rejeitada.

- Hipótese Alternativa **Ha15: P6 > P1**
- Hipótese Nula **Hn15: P6 = P1**

Executamos o *wilcox-Test* para a hipótese **Ha15** um $\alpha = 0.05$ e os resultados foram um o p-value = 1,152e-11 o que significa que a hipótese nula **Hn15** pode ser rejeitada.

5.4 Análise

O objetivo desta seção é fazer uma análise dos resultados obtidos, tentando identificar os pontos fortes e fracos de cada abordagem. A análise é feita em duas etapas. Primeiramente é feita uma análise individual de cada *player*, e de seu comportamento na

simulação. Posteriormente é feita uma análise geral, mostrando em quais situações os agentes levaram vantagens e desvantagens.

5.4.1 Análise individual de cada *Player*

Nesta seção a análise dos *players* será feita de forma individual e incremental.

5.4.1.1 *Player 1*

O *Player 1* é totalmente aleatório, como foi dito na seção 4.4.1, e não tem política de investimento. Este *player* joga de forma imprevisível, porém com uma estratégia de abertura definida. O que percebemos é que esse *player* joga de forma muito inocente e é facilmente derrotado pelo adversário, pois não faz qualquer raciocínio sobre o contexto do jogo nem faz qualquer planejamento de ataque e defesa.

O primeiro dos *players* tem uma taxa de sucesso de 15,33%, ou seja, venceu 23 partidas das 150 disputadas. Percebemos que a estratégia de abertura ou o acaso da sua aleatoriedade foi o que determinou que esse *player* conseguisse vencer nessa taxa, pois este não tem sequer uma rotina de crescimento de exército ou qualquer outro planejamento que aferisse dados do contexto para agir em contrapartida. Depois da estratégia de abertura o *player* realizava investimentos completamente aleatórios o que em 15,33% das vezes lhe levou a vitória.

5.4.1.2 *Player 2*

No segundo *player* inserimos o conceito de política de investimento, porém de forma aleatória. Uma rotina randômica define os valores dos atributos da política de investimento, como quota de investimento em classes, e a partir daí esses valores não serão mais adaptados independentemente do que aconteça durante a partida. Este *player* também tem o conceito de crescimento equilibrado por um padrão (que nesse caso foi o item básico de investimento), porém a atribuição desse padrão também foi feita de forma aleatória, ou seja, o padrão era definido no início do jogo de forma randômica. Uma vez definido o padrão, este nunca mais era alterado, permitindo que o *Player 2* tivesse o crescimento do seu exército de forma previsível (de acordo com o padrão definido).

O segundo *player* teve uma taxa pior que a do primeiro *player*. O *Player 2* obteve taxa de sucesso de somente 2,67% vencendo 4 das 150 partidas que este disputou. Observamos que a mudança da política de investimento foi a causa dessa taxa de vitórias pior que a do *Player 1*, pois a política de investimento, nesse *player* existe e é escolhida de forma aleatória, o que acaba travando investimentos que eram demandados pelo contexto. Por exemplo, o exército demandava um *Barracks* que é uma unidade que cria soldados, porém caso a política de investimento em *buildings* (edificações) impedisse o exército não poderia crescer e então o time acabava sendo derrotado. A política escolhida pode impedir um investimento caso a quota de investimento em uma determinada classe já tenha sido atingida.

Para a nossa surpresa, através do teste da hipótese **Ha1**, pode-se rejeitar a hipótese nula **Hn1** o que significa que $P2 > P1$ indicando que *Player 2* é superior ao *Player 1*. Talvez por conta do fato de que quando o *Player 2* perdia as partidas, a pontuação era próxima da pontuação do inimigo; já o *Player 1* quando perdia tinha pontuação bem inferior ao inimigo e quando vencida era por uma diferença pequena.

5.4.1.3 *Player 3*

Jogadores especialistas em jogos de RTS atribuíram os valores da política de investimento do *Player 3*, o que fez com que o este tivesse uma taxa de sucesso bem maior que a do seu anterior. Este *player* venceu 31 das 150 partidas que disputou e ficou com uma taxa de 20,67%. O *Player 3* também teve mais sucesso que o *Player 1* o que indica que a política de investimento ou o crescimento equilibrado por um padrão (ainda randômico neste *player*) foram determinantes para o melhoramento dos resultados.

Os testes de hipótese não mostraram a superioridade do *Player 3* sobre o *Player 2* uma vez que a hipótese **Hn2** não pode ser rejeitada, porém conseguiu indicar superioridade do *Player 3* sobre o *Player 1* pois a hipótese nula **Hn3** pode ser rejeitada.

5.4.1.4 *Player 4*

Jogadores especialistas em jogos de RTS atribuíram valores do padrão que define o crescimento equilibrado do *Player 4*, e este *player* foi ainda melhor que todos os seus antecessores. Obteve uma taxa de vitória de 25,33% vencendo 38 das 150 partidas disputadas. Entendemos a partir dos dados que o padrão que define o crescimento –

quando bem escolhido – fez com que esse *player* tivesse melhores resultados que os anteriores.

Os testes de hipótese não indicaram superioridade do *Player 4* sobre o *Player 3* e sobre o *Player 2* pois ambas hipóteses nulas **Hn4** e **Hn5** não puderam ser rejeitadas. Porém, indicou superioridade do *Player 4* sobre o *Player 1* uma vez que a hipótese nula **Hn6** pode ser rejeitada.

5.4.1.5 *Player 5*

O *Player 5* obteve uma taxa de sucesso muito próxima do seu antecessor, ficou com 26,67% vencendo 40 partidas das 150 que disputou. Acreditamos que somente o chaveamento das posturas de acordo com o contexto não foi suficiente para aumentar substancialmente os resultados desse *player* em relação ao *Player 4*.

Mesmo o número de vitórias desde *Player* sendo pouco maior que o seu anterior, a hipótese nula **Hn7** pode ser rejeitada o que confirma uma superioridade do *Player 5* sobre o *Player 4*. Sobre os outros a superioridade também é confirmada pela rejeição das hipóteses **Hn8**, **Hn9** e **Hn10**.

5.4.1.6 *Player 6*

O sexto *player* é o mais completo de todas as implementações e obteve os melhores resultados se comparados aos outros vencendo 63 partidas das 150 que disputou, ficando com a taxa de 42,00% de sucesso.

A política de investimento desse *player*, além de ser chaveada de acordo com o contexto, é flexível. Acreditamos que os ajustes automáticos que são feitos na política de investimento nesse *player* foram determinantes para esse sucesso, pois, por exemplo, quando havia demanda por algum investimento que a política não aceitava, ao invés da política negar a demanda, ela mesma se ajustava às necessidades. Os detalhes de como esses ajustes funcionam estão descritos na seção 4.3.1 Proporções Adaptativas de Classe. Definitivamente o *Player 6* é o mais eficaz dos *players* pela rejeição das hipóteses **Hn11**, **Hn12**, **Hn13**, **Hn14** e **Hn15**.

5.4.2 Análise Geral

Como imaginamos, os conceitos que inserimos aos nossos *players*, como o crescimento equilibrado por um padrão e a política de investimento melhoraram os resultados obtidos com exceção do *Player 2* que obteve a pior taxa de sucesso. Como já esclarecemos anteriormente, acreditamos que o fracasso desse *player* se deveu ao fato de que a política era escolhida de forma totalmente aleatória e não era ajustável, e isso acabava impedindo que investimentos importantes fossem feitos, pois a política mal escolhida (randomicamente) não os permitia. O *Player 3* já obteve melhores resultados que os anteriores pelo fato da política ser montada por especialistas. O *Player 4* e o *Player 5* tiveram taxas de sucesso muito próximas, nos levando a crer que somente o chaveamento das posturas não foi suficiente para incrementar de forma expressiva os resultados. A melhor taxa de sucesso foi obtida pelo *Player 6* que tem em si só todos os conceitos citados nesse trabalho. Acreditamos que o sucesso mais expressivo desse último *player* se deve ao fato de que sua política, além de ser chaveada (de acordo com o contexto do jogo), é ajustável, dando assim mais liberdade aos investimentos.

5.4.2.1 Problemas e limitações dos experimentos

Entendemos que nossos agentes não obtiveram taxas de vitórias ainda melhores porque há muitos aspectos deficientes no *BOT* que utilizamos. Como nosso foco era somente no gerenciamento de recursos, todos os outros aspectos, particularmente as questões de ordem tática, não foram abordadas, pois alongariam muito a pesquisa com aspectos fora do seu escopo.

De fato, em muitos casos o nosso *player* se deparava com problemas de *path finding* que impedia que o exército avançasse de encontro ao inimigo para um ataque, ou que se posicionasse melhor no momento da defesa. O módulo que na BWAPI utiliza para identificar caminhos e obstáculos no terreno do jogo têm inúmeras deficiências e isso contribui bastante para as derrotas dos nossos agentes. A **Error! Reference source not found.** é um *screenshot* de uma das partidas disputadas pelos nossos agentes e ilustra bem o que várias vezes acontecia durante as partidas, uma parte do exército dos nossos agentes impedindo a passagem de outra parte como pode ser visto. As unidades de *worker* na parte de cima da figura estão amontoadas impedindo a passagem do exército de ataque na parte de baixo da figura. Enquanto as unidades do exército dos nossos agentes tentavam transpor os obstáculos – o que quase nunca conseguiam fazer – o

exército inimigo estava preparando um ataque e então pegava o império de surpresa. Percebe-se através do mini mapa da Figura 12 que o exército do nosso *player* (em verde) está em número bem maior que o inimigo (amarelo), porém por conta desse tipo de impasse várias partidas foram perdidas.

Outro grande problema do *BOT* que utilizamos era a unidade responsável por reconhecer o terreno desvendando áreas das quais fosse possível extrair recursos, ou descobrir a localização do inimigo. Essa unidade, que é coordenada pelo *BOT* e não pelo nosso módulo de IA, muitas vezes não conseguia atingir algumas áreas do terreno, pois, mais uma vez, o módulo BWTA (*Broodwar Terrain analyzer* for BWAPI) não informava os espaços que esse *player* poderia atingir. Então muitas vezes essa unidade ficava andando em círculos pelo terreno do jogo sem desvendar novas áreas do mapa de suma importância.

Há também outros problemas, como na coordenação tática de ataque do exército, que é de responsabilidade do *BOT*. Identificamos que algumas vezes quando uma parte do exército, ou alguma edificação do nosso *player* estava sendo atacadas, as outras unidades de exército levavam muito tempo para auxiliar a defesa. Consideramos os problemas encontrados no BOT uma limitação do experimento, pois estes podem ter interferido nas nossas conclusões e aferições dos dados.



Figura 12 Problemas de path finding

Coletamos o tempo de processamento do PICFlex e o tempo total para cada partida dos nosso seis *Players*. Abaixo está a Tabela 7 com um resumo dos valores para o *Player* 6. O tempo foi medido em contagem de *ticks* do processador.

	Total	PICFlex
Tempo das 150 partidas	2081413	824232.1
Tempo médio	13876.09	5494.88
Desvio padrão do tempo	4387.324	1865.789

Tabela 7 Tempo do Player 6

Podemos verificar que o tempo do PICFlex representou 39,59%, o que consideramos um tempo relativamente longo e pode ter influenciado negativamente os resultados. O que pode acontecer é que enquanto o PICFlex está consumindo tempo com a análise do que deve fazer, o oponente pode já ter tomado sua decisão e já ter efetuado ações, se colocando à frente em número de ações por período de tempo.

Capítulo 6

6 Conclusões e trabalhos futuros

6.1 Conclusões

Neste trabalho foi apresentado um módulo de inteligência artificial com foco em resolver problemas de gerenciamento de recursos em ambientes complexos. Por motivos de simulação e de aderência ao nosso escopo escolhemos os jogos de estratégia em tempo real para testar nosso módulo. Desenvolvemos então o PICFlex que é uma solução de gestão de recursos baseada em política de investimento contextualizada e flexível.

Percebemos que o nosso conceito de política de investimento trouxe resultados positivos principalmente quando foram utilizadas na política de investimento melhores funções de escolha da política e estratégia de execução. Realizamos as simulações e coletamos os dados e após a análise percebemos que o PICFlex na sua forma mais completa gerou resultados positivos e possui potencial para implementação em jogos reais, além de ser possível estender sua implementação para realizar melhoramentos.

Identificamos alguns problemas no BOT que utilizamos como plataforma de testes, portanto não pudemos melhorar ainda mais nossos resultados, pois barramos em questões táticas que não eram o foco deste trabalho. Além das limitações do BOT que utilizamos no estudo, a própria API disponível para os testes (BWAPI) tem diversas limitações e *bugs* por ainda serem resolvidos, o que dificultou ainda mais nossas simulações e coleta de dados e criou uma limitação no nosso experimento.

6.2 Trabalhos futuros

As pesquisas com temas que abordam gerenciamento de recursos de forma ampla em ambientes complexos, como de um jogo de RTS são poucas e há uma vasta gama de problemas ainda por serem solucionados. Isto abre inúmeras oportunidades de trabalhos nesse campo.

Durante este trabalho escrevemos sobre *build-order* que é em suma a ordem com a qual entidades são criadas nos jogos de RTS. Mesmo se elas são em pequena quantidade, já existem pesquisas que tratam dessa problemática. Há também pesquisas que tem seu foco na otimização da geração de um *build-order*. E entendemos que nossa proposta de gerenciamento de recursos vai além da criação de uma ordem com a qual coisas são criadas. Então seria interessante um trabalho que comparasse a solução que propõe puramente um *build-order* com o PICFlex.

Criamos diversas implementações para nossas funções F e S e obtivemos algum sucesso, porém acreditamos que implementações mais sofisticadas delas poderiam resultar em resultados ainda melhores. Um trabalho interessante seria focar na modelagem de novas F e S e comparar os resultados com os nossos visando maiores pontuações do jogador.

O contexto de um jogo de RTS é bastante complexo e a definição do contexto neste trabalho foi bastante simplificada, portanto, uma ótima oportunidade de melhoria seria uma melhor definição do contexto, mais detalhado e com a avaliação das variáveis de estado, aliada a melhoria da função F (que recebe como parâmetro o contexto) e posterior avaliação frente aos nossos resultados.

Diante dos problemas encontrados no BOT que utilizamos como plataforma de testes, outra gama de melhorias podem ser feitas neste trabalho, pois acreditamos que uma vez solucionadas as questões táticas, várias oportunidades de melhorar a estratégia serão possíveis.

Após analisarmos o tempo que o PICFlex consome para realizar seu processamento de análise de contexto, planejamento, e execução das ações verificamos que este é custoso e representa quase 40% do tempo total da IA. Acreditamos que esse tempo extra influenciou negativamente os nossos resultados. Então, um estudo interessante seria focar em melhorar a performance do PICFlex e comparar com os resultados deste trabalho.

O nosso trabalho foi feito utilizando a raça *Terran* portanto seria interessante utilizar o PICFlex com as outras raças do próprio jogo Starcraft que são *Zergs* e *Protoss*. Uma comparação posterior dos resultados entre as três raças seria uma boa contribuição.

Outro trabalho que acreditamos ser muito interessante é a adaptação do PICFlex para outros ambientes complexos que não sejam os jogos de RTS, ou até mesmo que não seja ambientes de jogos digitais. Encontrar um outro ambiente complexo em que seja possível realizar simulações e coleta de dados e validar o PICFlex seria muito interessante.

Por fim, acreditamos que devido ao fato de ser um problema de tomada de decisão sob múltiplos critérios, uma abordagem ao problema que aplique técnicas de aprendizagem ou técnicas evolucionárias para parametrizar as políticas de investimento pode trazer resultados promissores.

Referências

- BILLINGS, D., PENA, L., SCHAEFFER, J., & SZAFRON, D. (1999). Using probabilistic knowledge and simulation to play poker. *Proceedings of the National Conference on Artificial Intelligence* (pp. 697-703). JOHN WILEY & SONS LTD.
- BRANQUINHO, A. A., & LOPES, C. (2010). Planning for resource production in real-time strategy games based on partial order planning, search and learning. *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference*, (pp. 4205-4211).
- BURO, M. (2003). Real-time strategy games: a new AI research challenge. *Proceedings of the 18th international joint conference on Artificial intelligence*, (pp. 1534-1535).
- BURO, M., & CHURCHILL, D. (2012). Real-Time Strategy Game Competitions. *AI Magazine*.
- CHAN, H., FERN, A., RAY, S., WILSON, N., & VENTURA, C. (2007a). Online planning for resource production in real-time strategy games. *Proceedings of the International Conference on Automated Planning and Scheduling*. Rhode Island.
- CHAN, H., FERN, A., RAY, S., WILSON, N., & VENTURA, C. (2007b). Extending online planning for resource production in real-time strategy games with search. *Workshop on Planning in Games, ICAPS*.
- CHUNG, M., BURO, M., & SCHAEFFER, J. (2005). *Monte Carlo Planning in RTS Games*. University of Alberta.
- CHURCHILL, D., & BURO, M. (2011). Build order optimization in starcraft. *Proceedings of AIIDE*, 14-19.
- CUNHA, R., & CHAIMOWICZ, L. (2010). *Um sistema de apoio ao jogador para jogos de Estratégia em Tempo Real*.
- DEMYEN, D. J., & BURO, M. (2006). Efficient triangulation-based pathfinding. *AAAI Conference*. Boston.

- DILL, K. (2006). Prioritizing Actions in a Goal-Based RTS AI. *AI Game Programming Wisdom, 3*, 331-340.
- GHALLAB, M., AERONAUTIQUES, C., ISI, C. K., & WILKINS, D. (1998). PDDL- the planning domain definition language.
- GIUNCHIGLIA, F., & WALSH, T. (1992). A theory of abstraction. *Artificial Intelligence*, 323-389.
- HAGELBÄCK, J. (2012). Potential-Field Based navigation in Starcraft. *Proceedings of 2012 IEEE Conference on Computational Intelligence and Games (CIG)*.
- HARMON, V. (2002). An economic goal approach to Goal-directed reasoning in an RTS. *Ai Game Programming Wisdom*, 402.
- HOLTE, R. C., & CHOUEIRY, B. Y. (2003). Abstraction and reformulation in artificial intelligence. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 1197-1204.
- HYDE, D. (s.d.). Using Bayesian Networks to Reason About Uncertainty. In: *AI Game Programming Wisdom 4*.
- KOVARSKY, A., & BURO, M. (2006). A first look at build-order optimization in real-time strategy games. *Proceedings of the GameOn Conference*, (pp. 18-22).
- KOVARSKY, A., & BURO, M. (2006). Heuristic search applied to abstract combat games. *Advances in Artificial Intelligence*, 55-77.
- LAIRD, J. E., & JONES, R. M. (1998). Building advanced autonomous AI systems for large scale real time simulations. *Freeman, M., editor, Proceedings of the 1998 Computer Game Developers' Conference*, (pp. 365–378).
- MCCOY, J., & MATEAS, M. (2008). An Integrated Agent for Playing Real-Time Strategy Games. *Association for the Advancement of Artificial*.
- ORKIN, J. (2005). Agent architecture considerations for real-time planning in games. *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment*.
- POLYA, G. (1945). *How to solve it: A new aspect of mathematical method*.

- POTTINGER, D. C. (2000). Terrain analysis in realtime strategy games. *Proceedings of Computer Game Developers Conference*.
- RESOURCE MANAGEMENT. (11 de 07 de 2013). Fonte: Business Dictionary: <http://www.businessdictionary.com/definition/resource-management.html>
- RETRO STUDIOS. (s.d.). Using a Spatial Database for Runtime Spatial Analysis. In: P. Tozour, *AI Game Programming Wisdom 2*.
- RUSSELL, S. J., & NORVIG, P. (s.d.). *Artificial intelligence: a modern approach* (Vol. 74). Prentice hall Englewood Cliffs.
- SAILER, F., BURO, M., & LANCTOT, M. (2007). Adversarial planning through strategy simulation. *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, (pp. 80-87).
- SHOEMAKER, S. (2006). RTS Citizen Unit AI. *AI Game Programming Wisdom, 3*.
- SIMON, H. A. (1981). The sciences of the artificial. MIT press.
- STURTEVANT, N., & BURO, M. (2005). Partial pathfinding using map abstraction and refinement. *Proceedings of the National Conference on Artificial Intelligence* (p. 1392). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- TEICH, T., & DAVIS, I. L. (2006). AI Wall Building in Empire Earth® II.
- TESAURO, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, 58-68.
- van den HERIK, H. J., DONKERS, H., & SPRONCK, P. H. (2005). Opponent Modelling and Commercial Games. *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG'05)*, (pp. 15-25).
- van LENT, M., LAIRD, J., BUCKMAN, J., HARTFORD, J., HOUCARD, S., & STEINKRAUS, K. (1999). Intelligent agents in computer games. *Proceedings of The Sixteenth National Conference on Artificial Intelligence*, (pp. 929–930).

- von der LIPPE, S., FRANCESCHINI, R., & KALPHAT, M. (1999). A robotic army: The future is CGF. *Proceedings of the 8th Conference on Computer Generated Forces and Behavioral Representation*.
- WEBER, B. G., & MATEAS, M. (2009). Case-based reasoning for build order in real-time strategy games. *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference, AAAI Press*, (pp. 106-111).
- WEBER, B. G., MATEAS, M., & JHALA, A. (2010). Applying goal-driven autonomy to StarCraft. *Proceedings of the Sixth Conference on Artificial Intelligence and Interactive Digital Entertainment*.
- ZUCKER, J.-D. (2003). A grounded theory of abstraction in artificial intelligence. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 1293-1309.