PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ADVERSARIAL HIERARCHICAL-TASK NETWORK INTEGRADO COM APRENDIZADO POR REFORÇO PARA JOGOS EM TEMPO REAL

MATHEUS DE SOUZA REDECKER

Trabalho de Conclusão I apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Felipe Rech Meneguzzi

Porto Alegre
2016

ADVERSARIAL HIERARCHICAL-TASK NETWORK INTEGRADO COM APRENDIZADO POR REFORÇO PARA JOGOS EM TEMPO REAL

RESUMO

Jogos de estrategia em tempo real são difíceis ao ponto de vista da IA devido ao grande espaço de estados e a limitação do tempo para tomar uma ação. Uma abordagem recentemente proposta é combinar busca adversaria com técnicas de HTN, o algoritmo é chamado de *Adversarial Hierarchical-Task Network*. Para tentar melhorar o desempenho do algoritmo propomos uma unificação do algoritmo com técnicas de aprendizado por reforço.

Palavras-Chave: planejamento automatizado, HTN, busca adversária, aprendizado por reforço.

ADVERSARIAL HIERARCHICAL-TASK NETWORK INTEGRATED WITH REINFORCEMENT LEARNING FOR REAL-TIME GAMES

ABSTRACT

Real-time strategy games are hard from an AI point of view due to the large state-spaces and the short time to compute each player's action. A recently proposed approach is to combine adversarial search techniques with HTN techniques, in an algorithm called Adversarial Hierarchical-Task Network. To improve the performance, we propose to integrate this algorithm to reinforcement learning techniques.

Keywords: automated planning, HTN, adversarial search, reinforcement learning.

Felipe: Ele não compila direito comigo, algo a ver com a lista de algoritmos

Matheus: ele está com um problema na biblioteca dos algoritmos, estou tentando resolver

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

IA – Inteligência Articial

HTN – Hierarchical Task Network

AHTN – Adversarial Hierarchical Task Network

RTS – Real-time Strategy

SUMÁRIO

1. INTRODUÇÃO

Inteligencia artificial(IA) é uma área em ciência da computação que tem como objetivo fazer com que o computador seja capaz de realizar tarefas que precisam ser pensadas, como é feito pelas pessoas. A IA possui algumas áreas de aplicação, tais como: aprendizado, planejamento, jogos, e mineração de dados entre outras.

As técnicas de IA utilizadas nos jogos são necessárias para conseguir uma melhor interação com o jogador, tornando o jogo mais real e assim prendendo a atenção do jogador [3]. As técnicas utilizadas nos jogos, geralmente, são mais simples do que as que utilizadas no meio acadêmico, pelo fato de que o tempo de resposta dos algoritmos é superior ao tempo que se tem para tomar uma ação ótima dentro do jogo [9]. Nos jogos as reações devem ser quase que imediatas, para isso técnicas que tentam explorar todo o espaço de estados do jogo se tornam inviáveis para jogos mais complexos. Por exemplo, no xadrez a quantidade aproximada de estados possíveis é de 10^{40}, isso mostra que o poder de processamento para gerar, de maneira rápida, uma ação precisa ser alto [3]. Então é difícil conseguir gerar uma ação ótima, em alguns casos são gerados ações sub ótimas para que o tempo de resposta não seja muito alto [9].

A busca é utilizada, dendo da IA, para achar a possível sequencia de ações que resolve um problema, considerando varias possibilidades de sequencia dessas ações. Os algoritmos de busca se diferenciam entre si na forma de escolher qual o próximo estado na busca pelo objetivo. Já a busca adversária é utilizada para a resolução de problemas de busca em modo competitivo. A busca adversária pressupõe que sempre o oponente irá realizar sempre a jogada que mais lhe beneficia, isso nem sempre acontece, seja porque o jogador é iniciante ou comete um erro. O ser humano consegue raciocinar para decidir as ações, mas nem sempre ele é perfeito nas escolhas das ações, ele consegue ser muito bom, mas o modo de pensar não o torna perfeito sempre. Planejamento é uma área da IA que busca a geração de planos de forma automática, parecido com a busca, utilizando técnicas para buscar a geração de um plano que satisfaça um objetivo. A utilização de técnicas de planejamento em jogos é uma tarefa difícil devido a sua grande quantidade de ações possíveis. Esse gênero de jogo possui um fator de ramificação muito grande, e cresce exponencialmente, com isso aplicar algoritmos de planejamento se torna uma tarefa não trivial [9].

Na busca de mitigar as limitações de eficiência computacional de abordagens tradicionais de raciocínio em jogos, Santiago Ontañón e Michael Buro propuseram o algoritmo chamado *Adversarial Hierarchical Task Network (AHTN)* [7]. Neste algoritmo são combinadas técnicas de HTN com o algoritmo de busca adversaria *minimax search*.

Propomos a utilização do algoritmo AHTN combinado com uma técnica de aprendizado por reforço em um jogo de estrategia em tempo real combinando uma técnica de

aprendizado por reforço. Com este trabalho pretendemos mostrar que o algoritmo de AHTN apresenta melhores resultados quando aplicado junto com técnicas de aprendizado por reforço.

2. AGENTES

3. BUSCA

3.1 Busca adversaria

A busca adversaria é utilizada para ambientes competitivos, como nos jogos. Como em um jogo o jogador, preferencialmente, não informa suas jogadas previamente, o ambiente se torna imprevisível, e com isso os objetivos dos jogadores entram em conflito, ambos estão em busca da vitoria. Como solução para esse problema é preciso gerar uma solução de contingencia para tentar antecipar as jogadas do adversário [9].

Para explicar como resolver esse problema, primeiro é preciso considerar um jogo com dois jogadores, um é chamado de MAX e o outro de MIN. O jogador MAX começa o jogo e o jogo e então é alternado uma jogada de MIN e uma de MAX até o final do jogo. Ao final do jogo, quem vence obtém uma recompensa positiva e quem perde uma negativa. Um jogo pode ser formalizado como [9]:

- S_0 - O estado inicial, que especifica como o jogo se configura no inicio.
- Jogadores(s) - Define qual jogador tem o movimento no estado.
- Ações(s) - Conjunto das ações possíveis em um estado.
- Resultado(s, a) - Um modelo de transição, que define o resultado da ação a aplicada ao estado s.
- Terminal(s) - Verifica se o estado é um estado onde o jogo terminou.
- Utilidade(s,p) - Define qual é o valor numérico para o jogo quando atingir um estado s terminal por um jogador p.

O estado inicial, as ações e os resultados definem a arvore das jogadas para o jogo. A arvore representa em cada nodo um estado do jogo e cada ligação com os níveis de baixo são os estados resultantes após a execução de cada ação possíveis para o estado. A alternância entre as jogadas de MAX e MIN até chegar as folhas da arvore que correspondem aos estados terminais. Como o ponto de vista é do MAX, o valor de cada nodo folha representa o valor de utilidade para o MAX, e os maiores valores representam bons resultados para o MAX e ruins para o MIN. Com isso o caminho resultante indica que aquela ação será a melhor ação para o estado atual.

Matheus: colocar uma figura de uma game tree?

Este tipo de busca leva em consideração que o jogador adversário sempre realizará a jogada que mais lhe beneficiará. Um algoritmo que utiliza deste recurso é o algoritmo *minimax search*.

4. PLANEJAMENTO

Felipe: Sempre que citar livro, dizer capítulo e páginas

Planejamento automatizado é uma área da inteligencia artificial que estuda o processo de geração de planos de forma computacional. Este tipo de planejamento está preocupado com a forma geral da composição dos planos [1]. O planejamento na computação se diferencia das outras áreas pelo fato de que todo o plano é gerado automaticamente.

Uma entrada necessária para qualquer algoritmo de planejamento é a representação do problema a ser resolvido. A representação de um problema é onde estão descritos os estados e as transições dos estados. Um estado do sistema é representados por um conjunto de átomos que resultam em verdadeiro ou falso dependendo da interpretação do ambiente. Para determinar as transições dos estados são utilizadas ações que são representadas por operadores de planejamento que alteram os valores dos átomos presentes em determinado estado. Um operador de planejamento é definido como *op* = (nome(*op*), precondições(*op*), efeitos(*op*)), onde cada elemento é definido como [1]:

- nome(*op*) - É o nome do operador de planejamento e *op* é o conjunto de todas as variáveis que irão aparecer qualquer parte do operador de planejamento.
- precondições(*op*) - *op* é o conjunto de átomos ou átomos negativos que representa a precondição do operador de planejamento.
- efeitos(*op*) - *op* é o conjunto de átomos ou átomos negativos que representa o efeito do operador de planejamento

O nome deve ser único pelo proposito de o nome poder se referir ao operador por completo, ou seja, após definido apenas com o nome pode-se inferir as pré e pós condições, assim escrevendo o nome(*op*) para se referir a todo o operador de planejamento *op*.

Um problema de planejamento é descrito como $P = (\Sigma, s_0, g)$. Onde Σ é a representação do problema, s_0 é o estado inicial, estado onde o problema começa, e g é o objetivo, estado onde o problema deve acabar. Um planejador é responsável por pegar essas informações e gerar um plano que comece pelo estado inicial e chegue ao objetivo através de um conjunto de ações descritas na representação do problema. A figura 4.1 representa esse processo.

4.1 HTN

Dentro da área de planejamento existe o planejamento hierárquico, chamado de *Hierarchical Task Network* (HTN). HTN é igual ao planejamento clássico, quando se refere

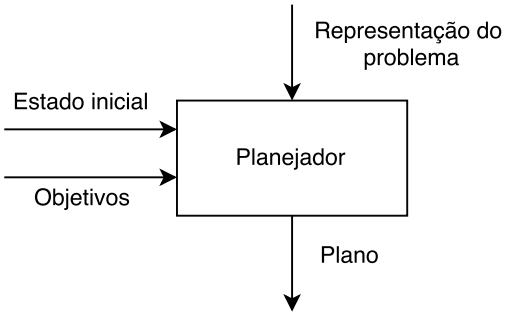


Figura 4.1 – Problema de planejamento

a representação do problema, entretanto o diferencial vem do que é planejado e como é planejado [1], mas acaba sendo mais expressivo que o planejamento clássico, porque no planejamento clássico o conjunto de solução dos problemas é uma linguagem regular, ou seja, pode ser expressada por uma expressão regular, já em HTN dependendo da técnica utilizada o conjunto de solução dos problemas pode chegar a uma linguagem livre de contexto, que contém as linguagens regulares como um subconjunto, pois existem linguagens livres de contexto que não são linguagens regulares. No pior caso o resultado gerado do planejamento clássico será exponencialmente maior do que o gerado pelo planejamento HTN [1].

Em planejamento HTN as ações são tratadas em mais alto nível [9]. e o objetivo é achar um conjunto de tarefas que resolve determinado problema. Para isso, além da representação do problema, um conjunto de métodos é adicionado como entrada, onde este conjunto serve para decompor as tarefas em tarefas menores. O planejamento é feito decompondo tarefas não primitivas recursivamente até chegar em tarefas primitivas, que podem ser realizadas com um operador de planejamento [1]. Um método HTN é definido como m = (nome(m), tarefa(m), subtarefas(m), limitação(m)), onde cada elemento é definido como [1]:

- nome(m) - Nome do método, que deve ser único, e m é o conjunto de variaveis que será utilizado em m.
- tarefa(m) - É uma tarefa não primitiva.
- (subtarefa(m),limitação(m)) - É uma ligação de tarefa de ligação.

Uma tarefa de ligação é definida como um par w = (U,C), onde U é um conjunto de tarefas de ligação e C um conjunto de limitações. Cada limitação deve ser satisfeita em todo o plano e cada tarefa de ligação é decomposta em sub tarefas até todas as tarefas se tornarem primitivas.

Um problema de planejamento HTN é descrito como P = (s_0, w, O, M), onde s_0 é o estado inicial, w a tarefa de ligação inicial, O é um conjunto de operadores de planejamento, e M é um conjunto de métodos.

Na busca pelo plano, o planejamento HTN começa planejando por um caminho, quando um caminho de resolução leva a um fim de linha é realizado um retrocesso(*backtracking*) até um caminho que tenha uma possibilidade diferente de caminho do que foi tomado anteriormente.

4.2 AHTN

Adversarial hierarchical-task network (AHTN) é um algoritmo proposto para tentar solucionar o problema do grande fator de ramificação dos jogos em tempo real [7]. Nele são combinados técnicas de HTN com o algoritmo *minimax search*. O algoritmo 4.1 é a representação da técnica de AHTN. Cada nodo da arvore das jogadas é definido por uma tupla (s, N_+, N_-, t_+, t_-), onde s é o estado corrente do ambiente, N_+ e N_- são a representação de planos HTN para os jogadores max e min, respectivamente, t_+ e t_- representam ponteiros para qual parte do plano HTN está sendo executado, sendo t_+ uma tarefa de N_+ e t_- uma tarefa de N_-. *nextAction(N,t)* é uma função que, dado um HTN N e um ponteiro t, encontra a tarefa primitiva que deve ser executada em N. Se N ainda não estiver completamente decomposto, ou seja, ainda existem tarefas não primitivas, então *nextAction(N,t)* = $\perp$. $decompositions_+(s, N_+, N_-, t_+, t_-)$ denota o conjunto das decomposições validas que adicionem apenas um novo método em N_+. A partir de uma função de avaliação, que pode ser aplicada sobre um estado, é retornado a recompensa de max em um estado terminal ou uma aproximação se o estado for não terminal. A partir destas definições, o algoritmo para AHTNMin é análogo. O algoritmo retorna o melhor plano encontrado para os dois jogadores, e também o resultado da função de avaliação no nodo terminal alcançado após a execução dos planos. A grande diferença entre o algoritmo de AHTN e o algoritmo do *minimax search*, é que as chamadas recursivas nem sempre se alternam entre max e min. O algoritmo troca de nodos max para min apenas quando os planos estão totalmente decompostos a ponto de gerar uma ação.

Algorithm 4.1 AHTNMax(s, N_+, N_-, t_+, t_-, d)

1: **if** terminal(s) $\vee$ d $\leq$ 0 **then**
2: **return** (N_+, N_-, e(s))
3: **end if**
4: **if** nextAction(N_+, t_+) $\neq \perp$ **then**
5: t = nextAction(N_+, t_+)
6: **return** AHTNMin(γ(s,t), N_+, N_-, t, t_-, d-1)
7: **end if**
8: $N_+^* = \perp$, $N_-^* = \perp$, $v^* = -\infty$
9: $\aleph$ = *decompositions*$_+(s, N_+, N_-, t_+, t_-)$
10: **for all** $N \in \aleph$ **do**
11: $(N_+', N_-', v') = AHTNMax(s, N, N_-, t_+, t_-, d)$
12: **if** $v' > v^*$ **then**
13: $N_+^* = N_+'$, $N_-^* = N_-'$, $v^* = v'$
14: **end if**
15: **end for**
16: **return** (N_+^*, N_-^*, v^*)

5. APRENDIZADO

Para os humanos o aprendizado ocorre durante toda a vida. O aprendizado é o ato de adquirir novos conhecimentos, ou modificar conhecimentos já existentes ou ainda adquirir uma experiencia por repetição do ato de forma incorreta. Aprendizado pode variar de adquirir conhecimento de tarefas simples, como decorando um numero de telefone, até tarefas mais complicadas, como a formulação de novas teorias [9].

5.1 Aprendizado de Máquina

A área na computação que estudo esse aprendizado de forma computacional é o aprendizado de maquina, melhor conhecida como *machine learning*. A definição de aprendizado de maquina proposta por Tom Mitchell [4] é a seguinte:

> Definição: Um programa de computador é dito que aprende de uma experiencia E com relação a alguma classe de tarefas T, e medida de performance P, se essa performance sobre as tarefas em T, medida por P, melhora com a experiencia E.

Essa definição mostra que o sistema aprimora seu conjunto de tarefas T com uma performance P através de experiencias E. Ou seja, um sistema baseado em aprendizado de maquina deve, através de experiencias, ter um ganho nas informações para solucionar os seus problemas. Para começar a resolver um problema utilizando aprendizado de maquina é preciso escolher qual experiencia será aprendida pelo sistema [4]. Para isso existem algumas técnicas que tratam aprendizado de maquina com objetivos diferentes [9]. Alguma das técnicas são:

- Aprendizado supervisionado: Aprender através de algum conjunto de exemplos a realizar a classificação de algum problema. Cada problema é mapeado para uma saída.
- Aprendizado não supervisionado: Aprender através das observações, algum pedrão ou regularidade, para classificar em grupos os problemas.
- Aprendizado por reforço: Aprender através das execuções, bem ou mal sucedidas. Aprende quais ações são melhores de ser executadas.

O aprendizado por reforço também é conhecido como *reinforcement learning*. O objetivo deste aprendizado é usar as recompensas obtidas nas observações para aprender uma politica do ambiente [9]. Para que isso aconteça, cada estado contém uma utilidade, que mostra o quão desejável é este estado, pode ser um desejo bom ou um ruim. Com essa informação é possível determinar, dado as ações disponíveis no estado, qual ação o sistema deve escolher para seguir a execução.

5.2 Aprendizado por Reforço

6. JOGOS

6.1 Jogos Real-time Strategy

6.2 MicroRTS

Jogos eletrônicos são muito populares, principalmente pela grande quantidade de gêneros, existem jogos de ação, aventura, esportes, estrategia, entre outros. Dentro dos jogos de estratégia há uma subseção que se chama jogos de estratégia em tempo real, neles os jogadores estão se enfrentando no mesmo momento, como o nome já diz.

Um exemplo deste gênero é o Starcraft[1]. Uma simplificação do Starcraft foi feita por Santiago Ontañón [6], chamada de MicroRTS. O MicroRTS foi desenvolvido para fins acadêmicos, com o intuito de aplicar e desenvolver técnicas de IA e para servir como prova de conceito para as técnicas criadas. O objetivo do jogo é destruir a base adversaria. Existem trabalhadores que podem coletar recursos e construir outros prédios. Os recursos são coletados dos minerais. Com os recursos é possível construir bases de ataque, onde são realizado o treinamento de unidades de ataque. Para conseguir realizar o objetivo de destruir a base adversaria é preciso ter unidades de ataque. O jogo oferece três destas unidades, são elas:

- Heavy - Possui um alto poder de ataque, mas sua velocidade é lenta.
- Light - Possui um baixo poder de ataque, mas sua velocidade é rápida.
- Ranged - Possui um ataque de longa distancia.

A Figura 6.1[2] mostra uma tela do jogo que representa o que foi explicado e ainda é possível observar que o fator de ramificação pode ser muito alto dependendo do cenário do jogo.

No ambiente há algumas estrategias implementadas, cada estrategia possui variações dos algoritmos. Algumas das estrategias são:

- Minimax Alpha-Beta Search Strategies - O que muda entre as técnicas é o jeito com que é feito a expansão do grafo.
- Monte Carlo Search Strategies - Executa jogadas aleatórias para planejar e após utiliza uma heurística para determinar em qual caminho seguir.

[1] http://us.battle.net/sc2/pt/
[2] https://github.com/santiontanon/microrts

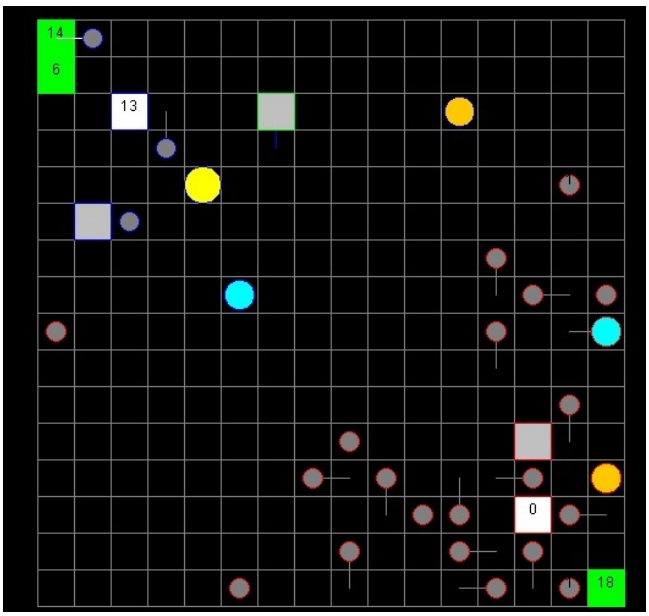


Figura 6.1 – Um exemplo da tela do MicroRTS

A plataforma já foi utilizada para aplicar técnica de IA. Por esse motivo a utilização desta plataforma se torna viável. A comparação entre as estrategias já existentes com a que estou propondo pode mostrar que a abordagem resulta em um melhor desempenho.

7. OBJETIVOS

Atualmente, há uma dificuldade na utilização de técnicas de IA para jogos em tempo real. Isso ocorre pelo fato que a IA precisa ser capaz de resolver tarefas, do mundo real, de maneira rápida e satisfatória. Geralmente, o espaço de estados dos jogos é enorme, isso faz com que não se tenha tempo de explorar todas as possibilidades de solução [3].

O objetivo geral deste trabalho é implementar o algoritmo de planejamento AHTN [7] que combina técnicas de HTN com o algoritmo de *minimax serach*. Após implementado, integrar aprendizado por reforço através da ferramenta WEKA[1] com o objetivo de melhorar o desempenho do algoritmo. Ao final do trabalho, o objetivo é conseguir comparar resultados obtidos com os resultados [5, 2, 8].

7.1 Objetivos Específicos

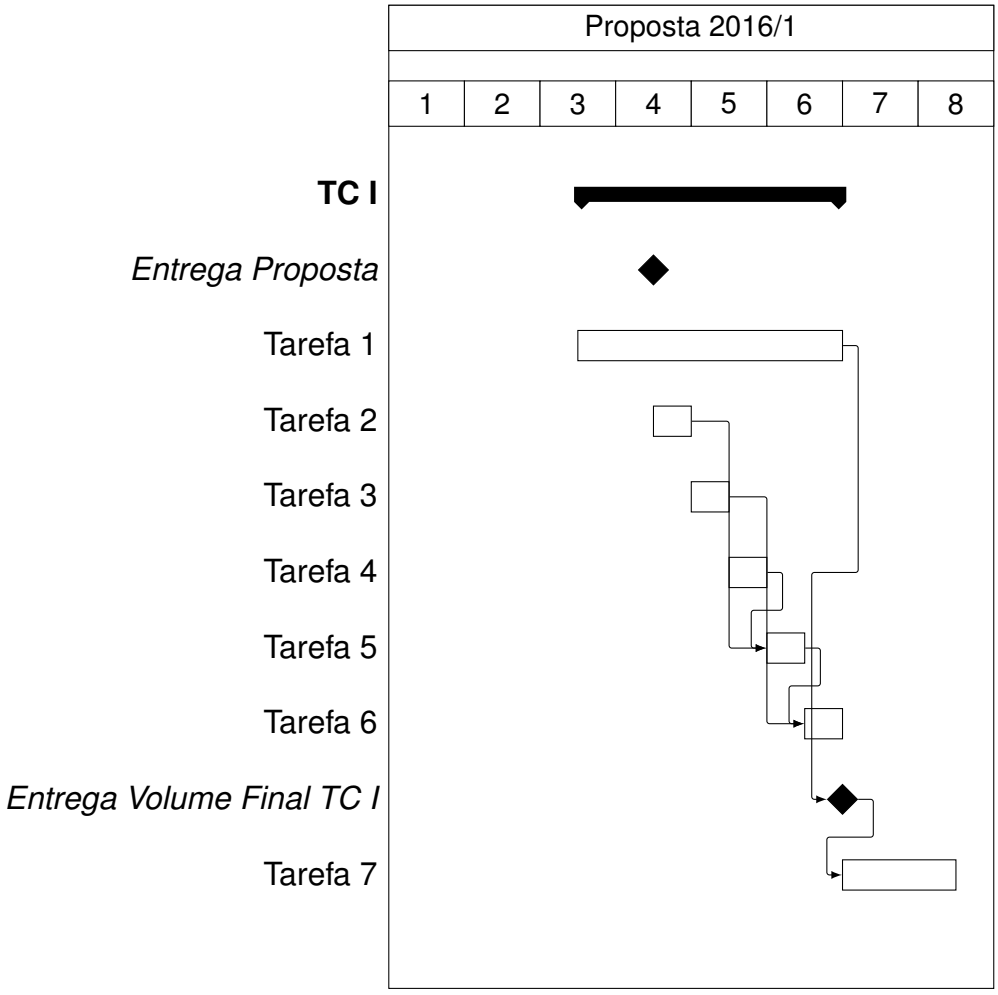
- Entender as técnicas de HTN.
- Entender o algoritmo de AHTN.
- Entender algoritmos de aprendizado por reforço.
- Estudar como integrar o algoritmo de AHTN com uma técnica de aprendizado por reforço.
- Avaliar a aplicabilidade dos algoritmos junto ao jogo escolhido.
- Definir a estratégia de implementação.
- Comparar resultados com outras abordagens.

[1] http://www.cs.waikato.ac.nz/ml/weka/

8. ATIVIDADES

Para está etapa do projeto do Trabalho de Conclusão, foi proposto o plano das atividades apresentado na diagrama de Gantt com a respectiva legenda.

- Tarefa 1 - Escrita do TC I e revisão bibliográfica.
- Tarefa 2 - Compreender o algoritmo de AHTN.
- Tarefa 3 - Compreender a usabilidade da ferramenta WEKA.
- Tarefa 4 - Compreender a arquitetura e código fonte do jogo MicroRTS.
- Tarefa 5 - Projetar a implementação do algoritmo AHTN na plataforma.
- Tarefa 6 - Projetar a união do algoritmo AHTN com o de aprendizado por reforço.
- Tarefa 7 - Planejar o TCII.



REFERÊNCIAS BIBLIOGRÁFICAS

[1] Ghallab, M.; Nau, D.; Traverso, P. “Automated planning: theory & practice”. Elsevier, 2004.

[2] Hogg, C.; Kuter, U.; Munoz-Avila, H. “Learning methods to generate good plans: Integrating htn learning and reinforcement learning.” In: AAAI, 2010.

[3] Millington, I.; Funge, J. “Artificial intelligence for games”. CRC Press, 2009.

[4] Mitchell, T. M. “Machine Learning”. New York, NY, USA: McGraw-Hill, Inc., 1997, 1 ed..

[5] Ontanon, S. “Experiments with game tree search in real-time strategy games”, *arXiv preprint arXiv:1208.1940*, 2012.

[6] Ontanón, S. “The combinatorial multi-armed bandit problem and its application to real-time strategy games”. In: Ninth Artificial Intelligence and Interactive Digital Entertainment Conference, 2013.

[7] Ontañón, S.; Buro, M. “Adversarial hierarchical-task network planning for complex real-time games”. In: Proceedings of the 24th International Conference on Artificial Intelligence, 2015, pp. 1652–1658.

[8] Ontanón, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; Preuss, M. “A survey of real-time strategy game ai research and competition in starcraft”, *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 5–4, 2013, pp. 293–311.

[9] Rusell, S.; Norvig, P. “Artificial intelligent: A modern approach”, 2003.