Adversarial Hierarchical-Task Network para Jogos em Tempo Real

Matheus de Souza Redecker
Orientador: Prof. Dr. Felipe R. Meneguzzi

Pontifícia Universidade Católica do Rio Grande do Sul

matheus.redecker@acad.pucrs.br

30 de Novembro, 2016

Adversarial Hierarchical-Task Network para Jogos em Tempo Real

Matheus de Souza Redecker
Orientador: Prof. Dr. Felipe R. Meneguzzi

Pontifícia Universidade Católica do Rio Grande do Sul
matheus.redecker@acad.pucrs.br

30 de Novembro, 2016

Motivação

- Inteligência Artificial em Jogos
- Jogos de estratégia em tempo real
- Algoritmo de Adversarial Hierarchical-Task Network

A IA em jogos faz com que os computadores sejam capazes de jogar sem intervenção humana. As técnicas são usadas para ter uma melhor interação com o usuario, e ele não perceber que está jogando contra uma maquina Mas as vezes as técnicas utilizadas nos jogos não são interamente de IA, pois elas podem utilizam conhecimentos proveniente do controle do jogo para tomar suas decisões.
Muitas vezes não há um tempo grande para decidir qual o proximo passo a ser tomado, com isso técnicas que tentam explorar todas as possibilidaes de um jogo se tornam ineficazes. O Xadrez por exemplo tem 10^40 estados possíveis do jogo é preciso algoritmos eficientes para gerar uma ação de forma rapida.
O algoritmo de AHTN foi proposto para mitigar as limitações computacionais de abordagens tradicionais de raciocionio em jogos. O algoritmo foi proposto por Santiago Ortanon e combina técnicas de planejamento e busca adversária.

O intuito deste trabalho é explorar eficientemente o espaço de ações disponíveis no MicroRTS utilizando conhecimento de domínio, a fim de definir qual a próxima ação que deve ser executada.

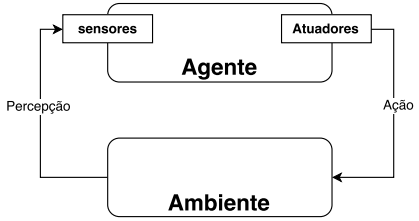
Background

- **Agentes**
- **Busca**
  - **Busca adversária**
  - **Minimax**
- **Planejamento**
  - **Planejamento Hierárquico**
  - **AHTN**

Background

Agentes

- Ambiente
- Percepções
- Sensores
- Atuadores
- Ação

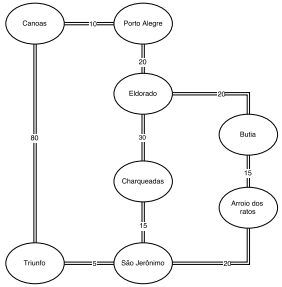


Agentes
Ambiente
Percepções
Sensores
Atuadores
Ação

agentes são entidades que agem de forma continua e autônoma em um ambiente, sendo capazes de receber estímulos do ambiente através de sensores, e assim responder aos estímulos por intermédio de atuadores. Para os agentes os estímulos do ambiente são recebidos como percepções. Os atuadores por sua vez, geram uma ação considerando as percepções

Busca

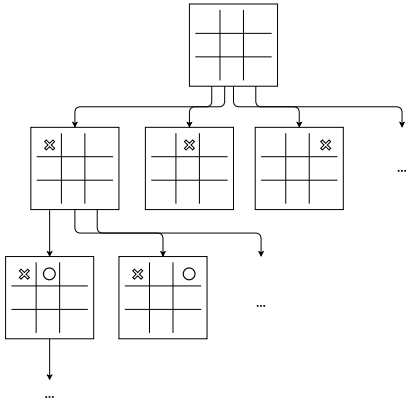
- Ações
- Função de transição
- Objetivo



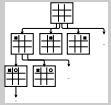
Técnicas de busca tem como objetivo encontrar uma sequência de ações para que um agente alcance um determinado objetivo.
As ações são executadas por agentes no ambiente
A função de transição define que quando um agente está em um estado e executa determinada ação ele vai para outro estado
O objetivo do agente é alcançar alguma coisa

Busca adversária

- Árvore das jogadas (*game tree*)
- Estado terminal
- Utilidade

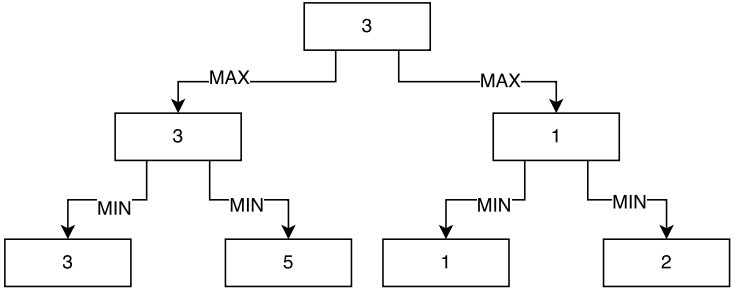


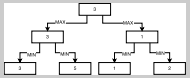
Busca adversária
- Árvore das jogadas (*game tree*)
- Estado terminal
- Utilidade



Já a busca adversária é utiliza em ambientes competitivas que possuam mais de um agente
As técnicas utilizam a arvore das jogadas para percorrer o espaço de estados dos jogos
Por exemplo, no jogo da velha, cada possibilidade de jogada deve ser percorrida para que o algoritmo possa determinar qual a melhor jogada
O estado terminal indica quando o jogo chegou ao fim
Cada estado do jogo pode ter um valor de utilidade, que é obtido através de uma função de avaliação
A função de avaliação pode utilizar uma heuristica para determinar o estado do ambiente
Esse valor serve para determinar o quão bom é o estado, ou ruim

Minimax

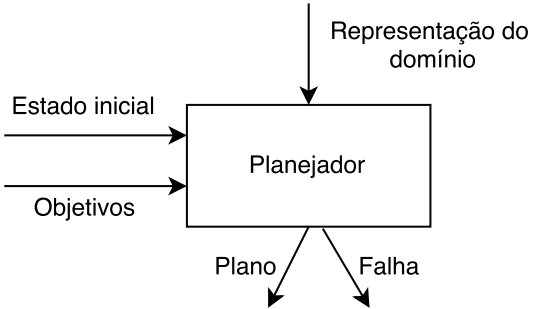




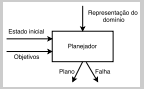
O algoritmo de Minimax é um algoritmo de busca adversária que tem como objetivo determinar qual a melhor jogada para o estado atual.
Este método considera dois agentes que estão competindo entre si
O agente Max representa a perspectiva que está tentando aumentar as recompensas das suas ações.
Já o agente Min é com quem o Max está jogando contra, e suas ações estão tentando ser minimizadas.

Por exemplo, quando Min tem duas possibilidades de jogadas, uma boa e uma ruim, o algoritmo seleciona a ruim como melhor opção para Max, pois quanto menor a chance de Min ter uma jogada de sucesso, melhor para ele.

Planejamento



Planejamento

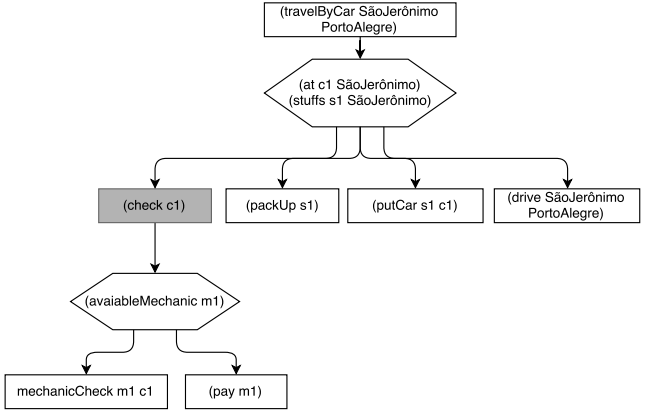


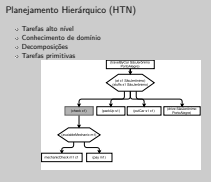
Planejamento estuda o processo de geração de planos de forma computacional.

O planejador utiliza uma representação do domínio, que contém informações das ações disponíveis no ambiente. E apartir de um estado inicial, e um objetivo que o agente deseja alcançar, o planejador encontra a sequencia de ações que parte do estado inicial e alcança seus objetivos.

Planejamento Hierárquico (HTN)

- Tarefas alto nível
- Conhecimento de domínio
- Decomposições
- Tarefas primitivas

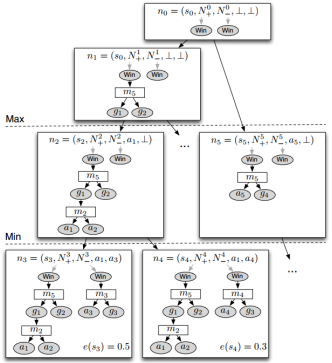




Problemas de grande complexidade acabam demorando muito com planejamento clássico.
Como alternativa para isso, foi proposto o planejamento hierarquico, ou HTN.
Ele se diferencia do anterior pelo fato de que as tarefas são tratadas em mais alto nivel, ou seja, as tarefas em alto nível, chamadas de tarefas não primitivas, são decompostas em tarefas menores, chamadas de primitivas, e assim por diante até que restem tarefas primitivas, que possam ser realizadas no ambiente.
Para isso, é preciso um conhecimento de domínio que contém informações relativas as ações disponíveis e transições que são feitas no ambiente.
Por exemplo, para realizar uma viajem de carro, não é só pegar o carro e dirigir, é preciso ir ao mecânico, fazer as malas, colocar as malas no carro, e então viajar.
As tarefas primitivas são as tarefas que fazem parte do plano!

Adversarial Hierarchical-Task Network

- Como funciona o algoritmo?
- Minimax + planejamento hierárquico



Adversarial Hierarchical-Task Network
Como funciona o algoritmo?
Minimax + planejamento hierárquico

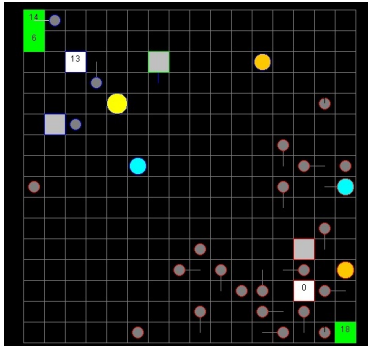
O algoritmo de AHTN é uma combinação do algoritmo de Minimax com o conhecimento de domínio do planejamento hierarquico
O algoritmo visa encontrar os melhores planos para Min e para Max
Para isso, os planos de cada um das perspectivas vão sendo decompostos.
O algoritmo pode ir até uma determinada profundidade, ou decompor todo os espaço de estados
A grande diferença do algoritmo é que a troca de perspectivas é feita apenas quando há uma tarefa primitiva a ser realizada no ambiente, caso o contrario o algoritmo decompõe as tarefas não primitivas e segue na mesma perspectiva

Recursos Utilizados

- **MicroRTS**
- **JSHOP2**

MicroRTS

- O que é?
- Construções
- Unidades
- Técnicas
- Camada de abstração



MicroRTS
- O que é?
- Construções
- Unidades
- Técnicas
- Camada de abstração

O MicroRTS é um jogo de estrategia em tempo real, criado por Santiago Ortanon em java
É um framework que permite a implementação de técnicas de IA
Construções: base - edificação prinicipal, e quartel- treinar unidades de ataque
Unidades: 3 unidades de ataque, 1 ataca de longe, 1 forte, 1 rapida e fraca
Técnicas: comportamentos pre determinados (script), MonteCarlo, Minimax, Portifolio Search
Camada de abstração- fornece as ações do jogos em classes para serem executadas

Toda IA precisa do método getaction para retornar a ação

JSHOP2

- Planejador
- Domínio
- Problema

JSHOP2

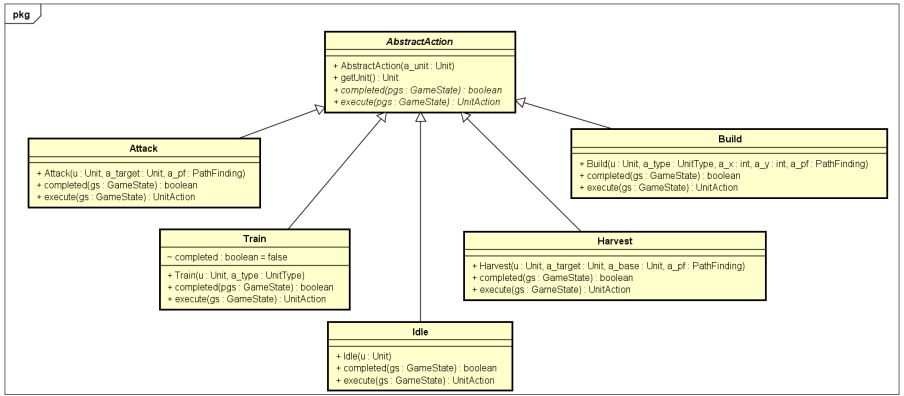
O JSHOP2 é um sistema de planejamento independente de domínio baseado em HTN.
O JSHOP2 foi desenvolvido em Java por Dana Nau e sua equipe de pesquisa.
O JSHOP2 preciso de 2 arquivos de entrada para gerar o plano
O domínio contém as descrições dos operadores (tarefas primitivas), e dos métodos (tarefas não primitivas)
O problema contém a especificação do estado inicial do ambiente, junto com uma ou mais tarefas que desejam ser executada
O JSHOP2 gera as tarefas primitivas do plano
O JSHOP2 foi escolhido por ser em java

Implementação

- **Ações do MicroRTS**
- **Modelagem do domínio**
- **Heurísticas**
- **Geração dos Planos**
- **Algoritmo AHTN**

Implementação

Ações do MicroRTS



Ações do MicroRTS

A camada de abstração fornece essas classes

Mas ainda assim, é preciso dizer qual unidade realiza cada método

Para isso, para cada ação disponível na camada, foi feita um método em Java relativo

Por exemplo, somente o worker pode extrair recurso, para isso é preciso saber qual recurso vai ser extraído e em qual base o recurso será armazenado

Modelagem do Domínio

- Operadores
- Métodos
- Unidade de ataque

Foram criados dois domínios
A grande diferença dos domínio é que no primeiro domínio, apenas uma unidade de ataque é criada
Já no segundo domínio, podem haver mais de uma unidade
Nos dois domínios, apenas uma base, um trabalhador e um quartel são criados
O trabalhador fica responsável por extrair recursos e construir as edificações quando necessario
O quartel cria as unidades de ataque
Os operadores do domínio, são as operações que podem ser executados no MicroRTS
Já os métodos implementam as regras do jogo
Como por exemplo, apenas um quartel pode ser criado, então o método deve garantir que apenas quando não houver quartel que o método é chamado.

Heurísticas

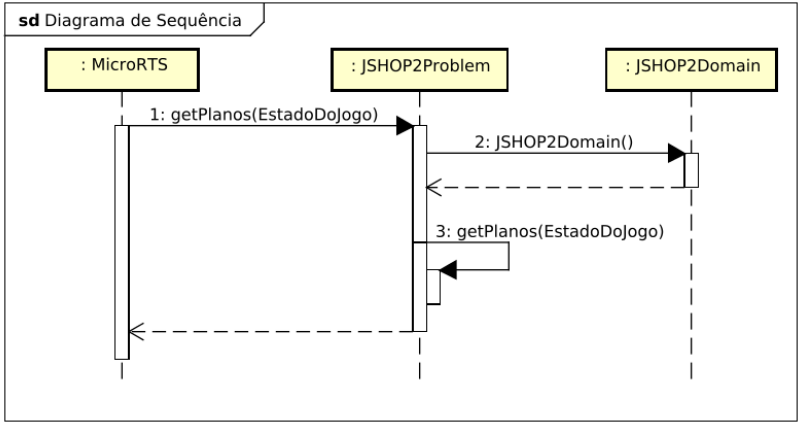
- Unidades adversárias
- Pesos
- Estado terminal

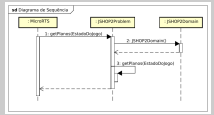
$$h1 = (1 * worker) + (5 * quartel) + (10 * base) + (2 * unidadesDeAtaque) \quad (1)$$

$$h2 = h1(jogador) - h1(inimigo) \quad (2)$$

Foram criados duas heuristicas
A heuristica 1 é feita para levando em conta as unidades que o jogador possui.
Os pesos são relativos a quanto custa de recursos para cada unidade ser construida
Mas assim, as unidades do inimigo não são levadas em consideração, Um inimigo pode ter varias unidades contra uma do jogador, isso faz com que ele esteja perdendo
Para contornar esse problema, foi criado a heuristica 2, nela o valor obtido da heuristica 1 para o jogador é diminuido do valor obtido para as unidades inimigas.
A heuristica indica um estado terminal para um dos jogadores em 2 determinadas situações do jogo:
Só existe uma base e não há recursos para fazer worker
Só existe worker e não há recursos para fazer base

Geração dos Planos





O JSHOP2 gera duas classes java, uma relativa ao domínio e uma relativo ao problema

Essas classes traduzem as informações do domínio e do problema para estruturas java

Como o domínio é estatico e não é alterada em nenhuma execução

Já o problema muda a cada nova configuração do ambiente

Para conseguir gerar os planos dentro do MicroRTS foi preciso criar uma classe java que representasse o Estado do Jogo

Nela está contido as informações provenientes do jogo, e então é possível converter as informações para as estruturas utilizadas pela classe java do JSHOP2

Assim o MicroRTS chama a geração dos planos, a classe do problema instancia um domínio, e então com as informações do jogo os planos são gerados

Algoritmo de AHTN

- Diferenças
- Tempo das ações

O JSHOP2 gera apenas as tarefas primitivas que devem ser executados para alcançar o plano
Por essa razão, o algoritmo passou por uma adaptação
A cada chamada há uma troca de perspectiva
Pelo fato de que não há a informação com o JSHOP2 dos métodos que estão sendo usados para a decomposição das tarefas
As tarefas dentro do MicroRTS levam alguns ciclos para serem realizadas
Por essa razão foi estipulado um valor de 50 rounds de jogo para que o algoritmo gere uma jogada novamente

Pseudo código do AHTN implementado

```
1: function AHTNMax(estado, planoMax, planoMin, prof)
2:     if terminal(estado) ∨ prof ≤ 0 then
3:         return (planoMax, planoMin, avaliacao(estado))
4:     end if
5:     nextAction(planoMax)
6:     (Pmax′, Pmin′, ev′) =⊥, ⊥, −∞
7:     for all plano ∈ getPlanos(estado) do
8:         (Pmax, Pmin, ev) = AHTNMin((estado, planoMax, planoMin, prof − 1))
9:         if ev′ > ev then
10:            (Pmax′, Pmin′, ev′) = (Pmax, Pmin, ev)
11:        end if
12:    end for
13:    return (Pmax′, Pmin′, ev′)
14: end function
```


Pseudo código do AHTN implementado

O algoritmo começa testando se a profundidade chegou ao fim ou se o estado é terminal Caso isso não acontece, uma ação é executada do plano, e então para cada ação possível nesse novo estado do jogo para Min é gerado os planos, assim o algoritmo seleciona o plano com a maior função de avaliação.

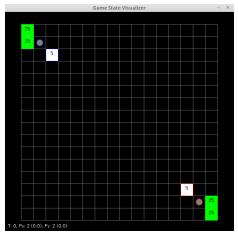
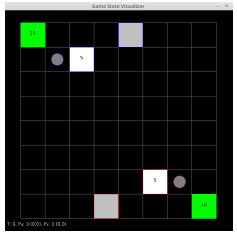
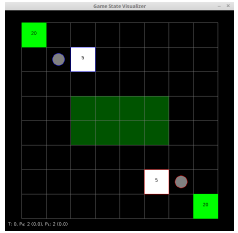
O algoritmo AHTNMin é analogo, mas ao inves da maior função de avaliação é buscado a menor

Resultados

- **Mapas**
- **Tamanho das técnicas**
- **Tempo de geração das ações**

Mapas

- Mapa 1
  - 16x16, base e trabalhador
  - Domínio HTN 2
- Mapa 2
  - 8x8, base, trabalhador e quartel
- Mapa 3
  - 8x8, base, trabalhador e obstáculos
  - Domínio HTN 2
- Lado do jogo

Mapas
- Mapa 1
  - 16x16, base e trabalhador
  - Domínio HTN 2
- Mapa 2
  - 8x8, base, trabalhador e quartel
- Mapa 3
  - 8x8, base, trabalhador e obstáculos
  - Domínio HTN 2
- Lado do jogo

A avaliação dos resultados foi feita em três mapas e com 10 jogos
O mapa 1 e 3 apresentam melhor desempenho no domínio 2, pelo fato de que varias tropas são criadas e técnicas que não ganhavam passam a ganhar no outro domínio
O mapa 2 perde rapido pois as técnicas são muito violentas
O lado do jogo influencia muito nos resultados, pois no lado vermelho no mapa 1 é posicionado em melhor local para criação das tropas
Nos demais mapas, as técnicas no MicroRTS não se comportam da mesma maneira que no outro lado
Elas demoram um pouco para atacar e com isso dão tempo do AHTN criar tropas

Tamanho das técnicas

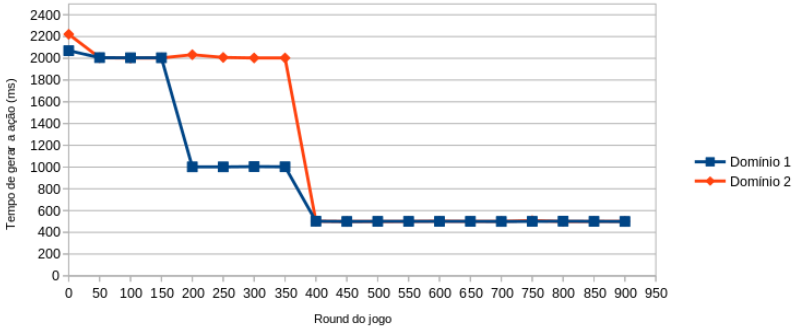
Adversário	**Tamanho**	Porcentagem de vitórias
Domínio 1	19,9 kB	-
Domínio 2	20,1 kB	-
RandomBiasedIA	4,6 kB	80%
WorkerRush	13,6 kB	0%
MonteCarlo	18,1 kB	30%

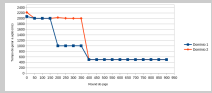
Tamanho das técnicas

Adversário	Tamanho	Porcentagem de vitórias
Domínio 1	19,9 kB	-
Domínio 2	20,1 kB	-
RandomBiasedIA	4,6 kB	80%
WorkerRush	13,6 kB	0%
MonteCarlo	18,1 kB	30%

Foi utilizado o algoritmo de compressão do zip para medir o tamanho de código das técnicas
A maior técnica presente no MicroRTS é a MonteCarlo, o algoritmo ocnsegue ganhar 30% dela.
A técnica de WorkerRush não perdeu em nenhum teste, e ela apersente um tamanho mediano em relação as demais técnicas
Uma das técnicas mais simples é a Random, e ela em alguns casos consegue vencer
Os domínios ficaram pesados pela integração do JSHOP2 na geração dos planos, por essa razão eles acabaram ficando com o maior peso entre as técnicas

Tempo de geração das ações





isso ocorre porque há menos ações que podem ser realizadas no jogo, o que implica em menos níveis na árvore de busca que o algoritmo tem que percorrer.

No final do jogo, o jogador já tem tropas de ataque e apenas manda elas atacar.

Conclusão

- Problemas
- Influência do domínio
- Trabalhos futuros

Conclusão

O objetivo inicial era integar uma técnica de Machine Learning ao algoritmo para tentar melhorar os resultados. Devido ao tempo perdido na integração do JSHOP2 com o MicroRTS e na criação do domínio, isso não foi possível

O principal problema do algoritmo é no tempo de geração das ações, pois todas as outras técnicas do microRTS levam 1 milissegundo para gerar uma ação, enquanto o algoritmo leva no minimo meio segundo, isso prejudica o desempenho em jogos RTS

O domínio tem influencia nos resultados, pois como ele não possui um plano de contingencia, por exemplo, nas técnicas que criam trabalhadores para atacar, o domínio segue criando seu quartel para criar suas unidades de ataque. Uma vantagem dessa abordagem é poder construir um domínio e acoplar facilmente ao algoritmo

Como trabalhos futuros- Acoplar uma técnica de Machine Learning - Implementar um mecanismo de tempo de procura por uma ação - Criar domínios mais abrangentes

Obrigado!

Obrigado!