

# 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

- 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. Esse tipo de técnica é utilizada para o jogador humano não perceber que está jogando contra uma máquina

Os jogos de estratégia em tempo real são jogos onde os jogadores devem construir tropas, coletar recursos, construir edificações, para tentar derrotar o seu adversário. O desenvolvimento de técnicas de IA para esse gênero de jogos possuem algumas dificuldades, pois existe muito pouco tempo para a tomada de decisão, pois existem muitas possibilidades de jogadas dentro do jogo. Com isso técnicas que tentam explorar todas as possibilidades de um jogo se tornam ineficazes. Então é preciso algoritmos eficientes para gerar uma ação de forma rápida.

O algoritmo de AHTN foi proposto para mitigar as limitações computacionais das abordagens de raciocínio em jogos. O algoritmo foi proposto por Santiago Ortanón e combina técnicas de planejamento e busca adversária.

O intuito deste trabalho é explorar eficientemente o espaço de ações de um jogo RTS 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

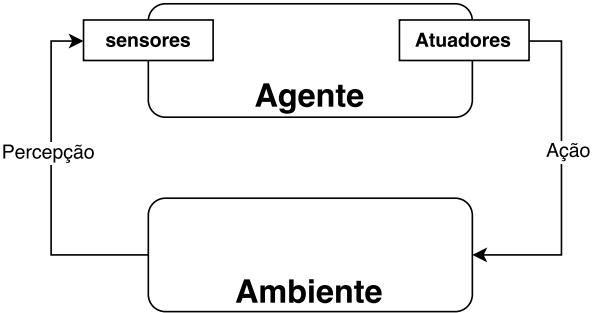
2016-11-30

└ Background

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

2016-11-30

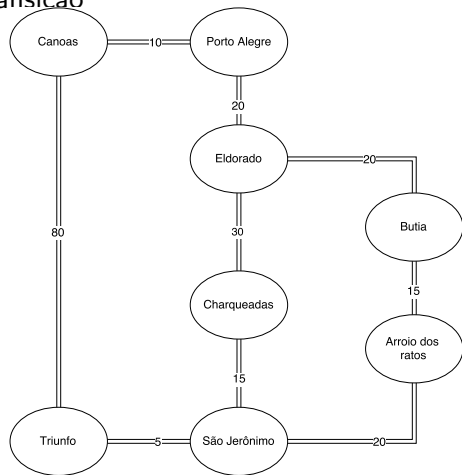
Agentes



agentes são entidades que agem de forma autônoma em um ambiente  
No ambiente é informado os estados que o agente consegue alcançar.  
recebem 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

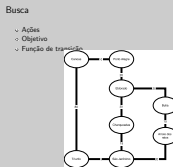
# Busca

- Ações
- Objetivo
- Função de transição



2016-11-30

└ Busca



Técnicas de busca tem como objetivo encontrar uma sequência de ações para que um agente alcance um determinado objetivo.

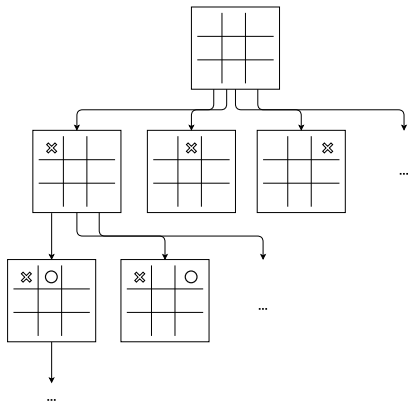
As ações são executadas por agentes no ambiente

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

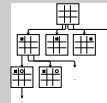


2016-11-30

- └ Busca adversária

Busca adversària:

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



Já a busca adversária é utilizada em ambientes competitivos que possuam mais de um agente

As técnicas utilizam a árvore 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

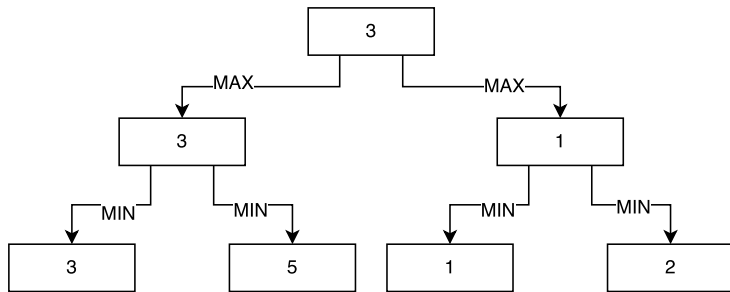
Cada estado do jogo pode ter um valor de utilidade, que é obtido através de uma função de avaliação

A função de avaliação pode utilizar uma heurística para determinar o estado do ambiente

Esse valor serve para determinar o quão bom é o estado, ou ruim

2016-11-30

└ Minimax



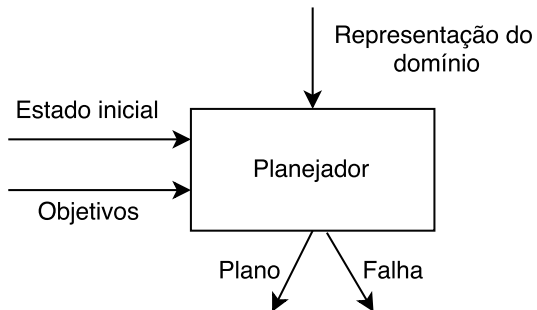
O algoritmo de Minimax é um algoritmo de busca adversária que tem como objetivo determinar qual a melhor jogada para o estado atual. Este método considera dois agentes que estão competindo entre si. O agente Max representa a perspectiva que está tentando aumentar as recompensas das suas ações.

Já o agente Min é com quem o Max está jogando contra, e suas ações estão tentando ser minimizadas.

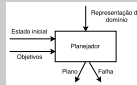
Por exemplo, quando Min tem duas possibilidades de jogadas, uma boa e uma ruim, o algoritmo seleciona a ruim como melhor opção para Max, pois quanto menor a chance de Min ter uma jogada de sucesso, melhor para ele.

2016-11-30

## Planejamento



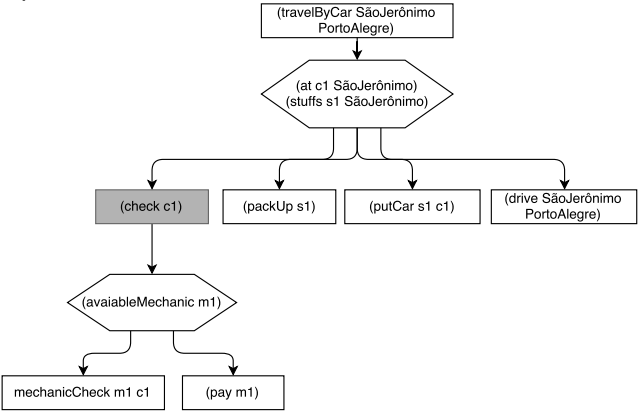
Planejamento é o processo de geração de planos de forma computacional. O planejador utiliza uma representação do domínio, que contém informações das ações disponíveis no ambiente. E a partir de um estado inicial, e um objetivo que o agente deseja alcançar, o planejador encontra a sequência de ações que parte do estado inicial e alcança seus objetivos.





# Planejamento Hierárquico (HTN)

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



2016-11-30

## Planejamento Hierárquico (HTN)



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 nível, 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.

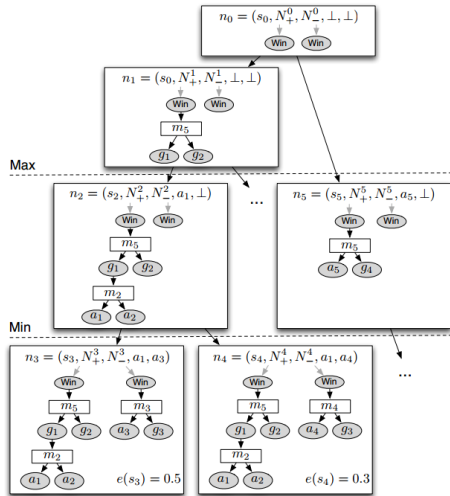
Para isso, é preciso um conhecimento de domínio que contém informações relativas as ações disponíveis e transições que são feitas no ambiente.

Por exemplo, para realizar uma viagem 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!

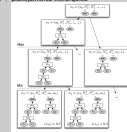
# Adversarial Hierarchical-Task Network

- 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 hierárquico. 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 o 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**

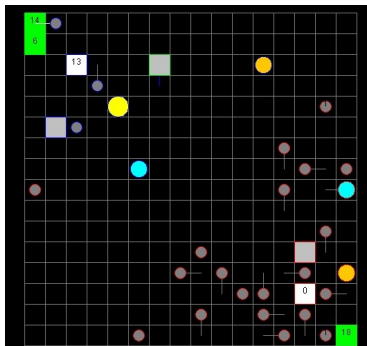
2016-11-30

## Recursos Utilizados

- Recursos Utilizados
  - ↳ **MicroRTS**
  - ↳ **JSHOP2**

# MicroRTS

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



O MicroRTS é um jogo de estratégia em tempo real, criado por Santiago Ortanon em java

É um framework que permite a implementação de técnicas de IA

Construções: base - edificação principal, 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

- Planejador
- Domínio
- Problema

```
(defdomain move
  (
    (:operator (!move ?from ?to)
      ((at ?from))
      ((at ?from))
      ((at ?to))
    )
    (:method (travel ?from ?to)
      ((at ?from) (path ?from ?to))
      ((!move ?from ?to))
      ((not (path ?from ?to)) (path ?from ?somewhere))
      ((!move ?from ?somewhere) (travel ?somewhere ?to))
    )
  )
)
```

2016-11-30

JSHOP2

- ↳ Planejador
- ↳ Domínio
- ↳ Problema

```
(defdomain move
  (:operator (!move ?from ?to)
    ((at ?from))
    ((at ?to))
  )
  (:method (travel ?from ?to)
    ((at ?from) (path ?from ?to))
    ((!move ?from ?to))
    ((not (path ?from ?to)) (path ?from ?somewhere))
    ((!move ?from ?somewhere) (travel ?somewhere ?to))
  )
)
```

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**

2016-11-30

AHTN para Jogos em Tempo Real

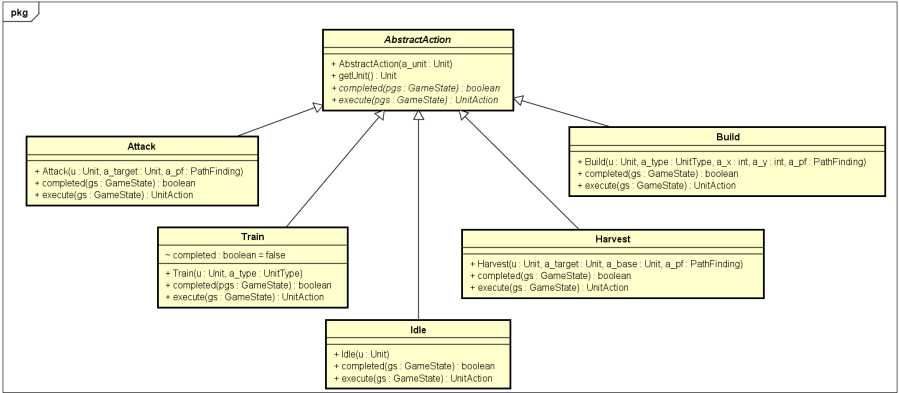
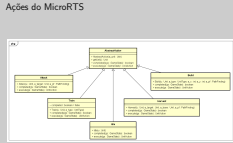
## Implementação

Implementação

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

2016-11-30

## 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

- Operadores
- Métodos
- Unidade de ataque

Foram criados dois domínios

A grande diferença dos domínios 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 necessário

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

### Heurísticas

- Unidades adversárias
- Pesos
- Estado terminal

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

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

- 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 heurísticas

A heurística 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 construída

Mas assim, as unidades do inimigo não são levadas em consideração, Um inimigo pode ter várias unidades contra uma do jogador, isso faz com que ele esteja perdendo

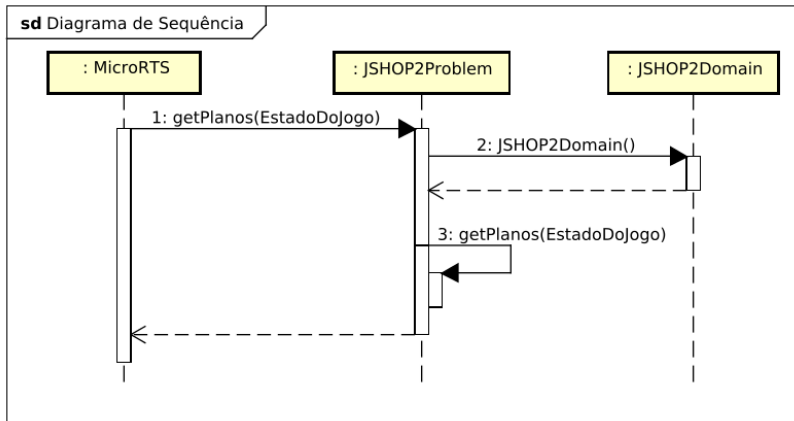
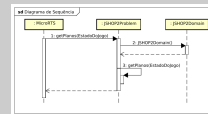
Para contornar esse problema, foi criada a heurística 2, nela o valor obtido da heurística 1 para o jogador é diminuído do valor obtido para as unidades inimigas.

A heurística 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

- 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

2016-11-30

## Pseudo código do AHTN implementado

```

1: function AHTNMax(estado, planoMax, planoMin, prof)
2:   if terminal(estado)  $\vee$  prof  $\leq 0$  then
3:     return (planoMax, planoMin, avaliacao(estado))
4:   end if
5:   nextAction(planoMax)
6:   (Pmax', Pmin', ev') =  $\perp, \perp, -\infty$ 
7:   for all plano  $\in$  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

```

```

1: function AHTNMax(estado, planoMax, planoMin, prof)
2:   if terminal(estado)  $\vee$  prof  $\leq 0$  then
3:     return (planoMax, planoMin, avaliacao(estado))
4:   end if
5:   nextAction(planoMax)
6:   (Pmax', Pmin', ev') =  $\perp, \perp, -\infty$ 
7:   for all plano  $\in$  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

```

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

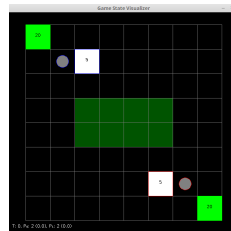
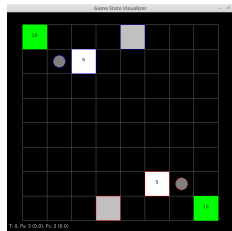
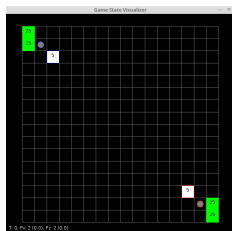
2016-11-30

## 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
- Lados do jogo



## Mapas



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

└ Tamanho das técnicas

Adversário	Tamanho	Porcentagem de derrotas
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%

<b>Adversário</b>	<b>Tamanho</b>	Porcentagem de derrotas
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%

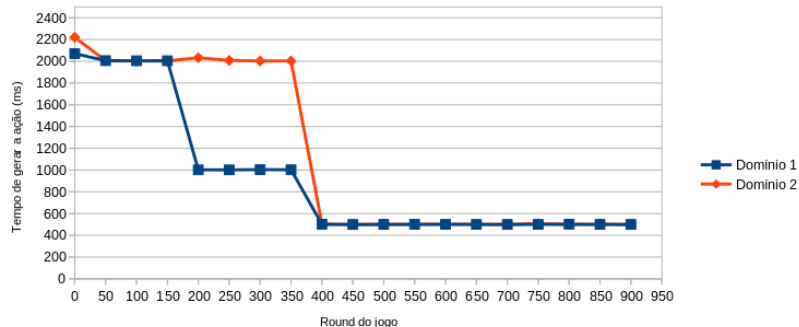
A maior técnica presente no MicroRTS é a MonteCarlo, o algoritmo consegue ganhar 30% dela.

A técnica de WorkerRush não perdeu em nenhum teste, e ela apresentou um tamanho mediano em relação as demais técnicas

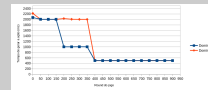
Uma das técnicas mais simples é a Random, e ela em alguns casos consegue vencer

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



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
  - Machine Learning
  - Tempo de geração das ações
- Influência do domínio
  - Planos de contingência
  - Troca do domínio
- Trabalhos futuros
  - Machine Learning
  - Limitador de tempo
  - Domínios

O objetivo inicial era integrar 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 mínimo 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

2016-11-30