Um Estudo de Mapeamento de Sistemas Multiagentes baseados em FIPA para Arquiteturas GPGPU

Luiz Guilherme Oliveira dos Santos[1]
Orientadora: Flávia Bernardini[1]

[1]Universidade Federal Fluminense(UFF)
Pólo Universitário de Rio das Ostras(PURO)
Departamento de Ciência e Tecnologia(RCT)
Rio das Ostras - RJ, Brasil

24 de Agosto de 2010

PURO

Sumário Principal

PURO

Sumário Principal

PURO

O que é inteligência?

Planejar

Decidir

Raciocinar

Compreender

O que é inteligência?

Resolver

Abstrair

Aprender

Criar

PURO

Discussão Filosófica

Como criar inteligência?

- Mitos
- Engenhocas complexas
- "Penso, Logo Existo" (René Descartes)
- Desenvolvimento da Lógica Matemática
- Turing(Colossus I) e Von Neumann(ENIAC)

Discussão Filosófica

Como criar inteligência?

- Mitos
- Engenhocas complexas
- "Penso, Logo Existo" (René Descartes)
- Desenvolvimento da Lógica Matemática
- Turing(Colossus I) e Von Neumann(ENIAC)

As máquinas podem "pensar"?

- Aprendizado de Hebb [Hebb, 1949]
- Teorias: "Teste de Turing" [Turing, 1950]
- Histórias, Suposições [Asimov, 1950]
- Criação da Área de Inteligência Artificial

PURO

Inteligência Artificial

Entusiasmo Inicial(1952-1969)

- *General Problem Solver* [Newell et al., 1959]
- *Geometry Theorem Prover* [Gelertner, 1959]
- LISP
- Solucionador de problemas de Cálculo, SAINT [Slagle, 1963]
- Solucionador de Problemas de QI, ANALOGY [Evans, 1968]

PURO

Inteligência Artificial

Entusiasmo Inicial(1952-1969)

- *General Problem Solver* [Newell et al., 1959]
- *Geometry Theorem Prover* [Gelertner, 1959]
- LISP
- Solucionador de problemas de Cálculo, SAINT [Slagle, 1963]
- Solucionador de Problemas de QI, ANALOGY [Evans, 1968]

Tempos Difíceis(1969-1980)

- Maioria dos programas não continham de fato um conhecimento
- Teoremas maiores não puderam ser resolvidos
- Sistemas Especialistas como o MYCIN [Shortliffe and Buchanan, 1984]
- PROLOG e PLANNER

PURO

Inteligência Artificial

Resurgimento(1980-Atual)

- Retropropagação
- Redes Bayesianas
- Mineração de Dados
- Teoria dos Agentes

Figura: Carro Autônomo construído por alunos da Universidade de Standford cruza o deserto de Las Vegas em 7 horas

PURO

Agentes Inteligentes

Figura: Robô Mars Rover da NASA [Buchanan, 2005].

- Relacionamento com a Sociedade:

Agentes Inteligentes

Figura: Robô Mars Rover da NASA [Buchanan, 2005].

- Relacionamento com a Sociedade:
  - Comunicação
  - Cooperação
  - Negociação
- Exemplos de Uso:

Agentes Inteligentes

Figura: Robô Mars Rover da NASA [Buchanan, 2005].

- Relacionamento com a Sociedade:
  - Comunicação
  - Cooperação
  - Negociação
- Exemplos de Uso:
  - Carros inteligentes
  - Exploração Espacial e Marinha
  - Entretenimento Digital
  - Simulações

PURO

Sumário Principal

PURO

Agente Inteligente

De acordo com [Russell and Norvig, 1995], as características principais de um agente são:

Agente Inteligente

De acordo com [Russell and Norvig, 1995], as características principais de um agente são:

- Ele age num determinado ambiente
- Comunica-se com outros agentes
- Tem objetivos individuais a atingir ou uma função de satisfação a otimizar
- Tem recursos próprios
- É capaz de perceber seu ambiente (de modo limitado)
- Possui competência e oferece serviços
- Seu comportamento tende a atingir seus objetivos utilizando as competências e os recursos que dispõe
- Possui Funções de percepção e comunicação

PURO

Estrutura de um agente

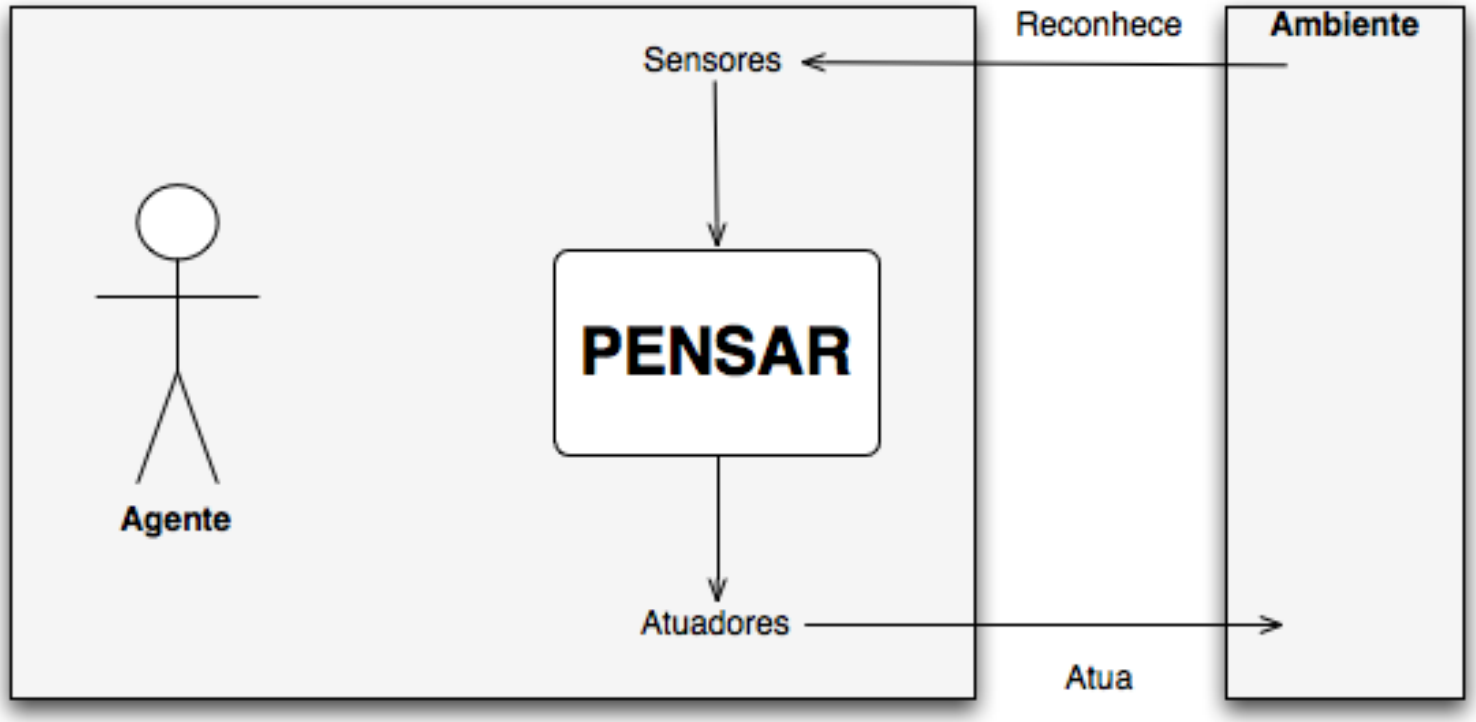
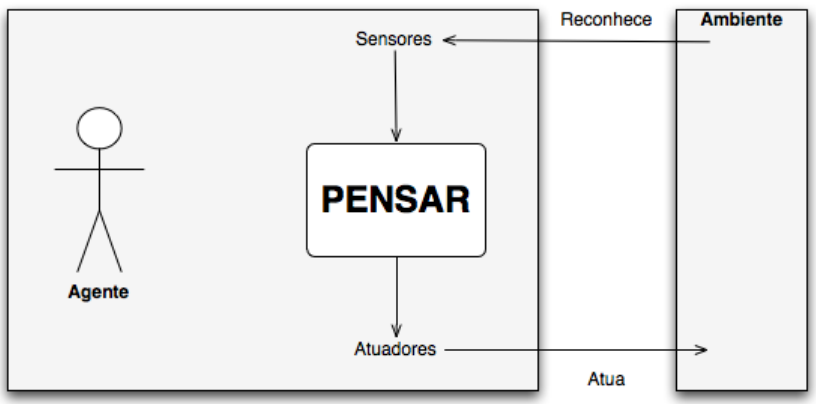


Figura: Estrutura Básica de um Agente

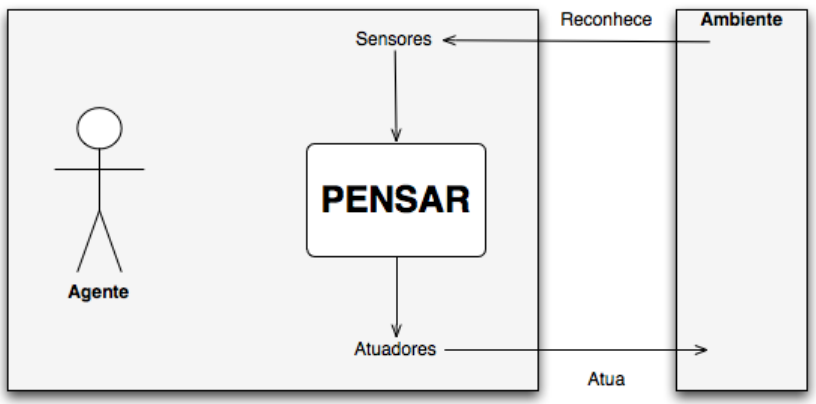
PURO

Estrutura de um agente



- Sequência de Percepção, $f(x_1, x_2, \cdots, x_n)$

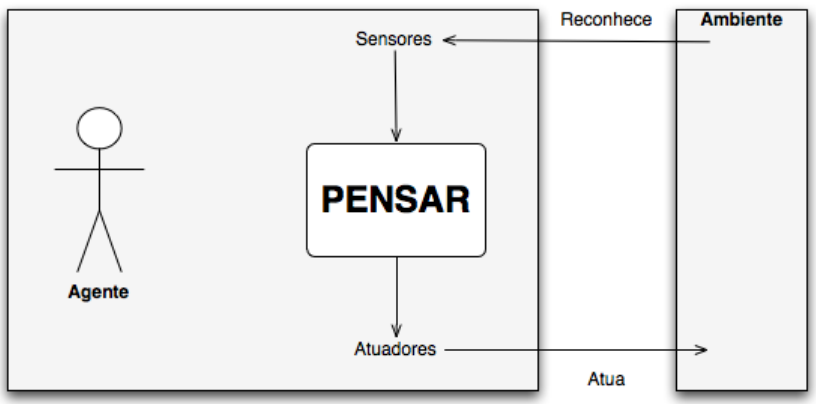
Estrutura de um agente



- Sequência de Percepção, $f(x_1, x_2, \cdots, x_n)$
- Autonomia de Decisão

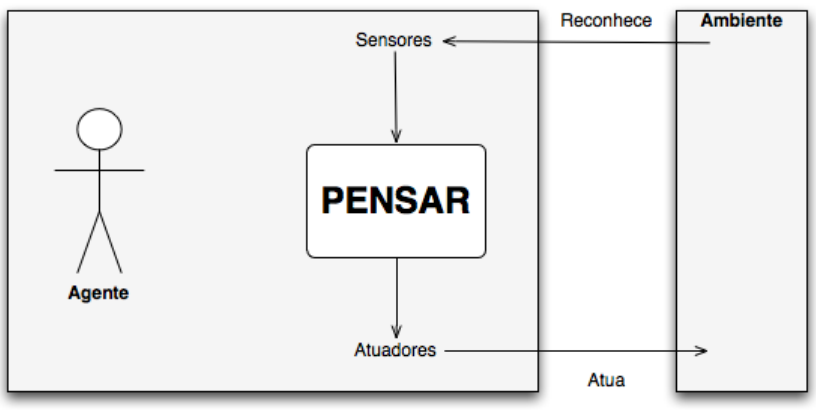
PURO

Estrutura de um agente



- Sequência de Percepção, $f(x_1, x_2, \cdots, x_n)$
- Autonomia de Decisão
- Agenda Própria

Estrutura de um agente



- Sequência de Percepção, $f(x_1, x_2, \cdots, x_n)$
- Autonomia de Decisão
- Agenda Própria
- Reativos $\times$ Cognitivos
- Pode ser modelado como uma máquina de estados

PURO

Estrutura de um Agente

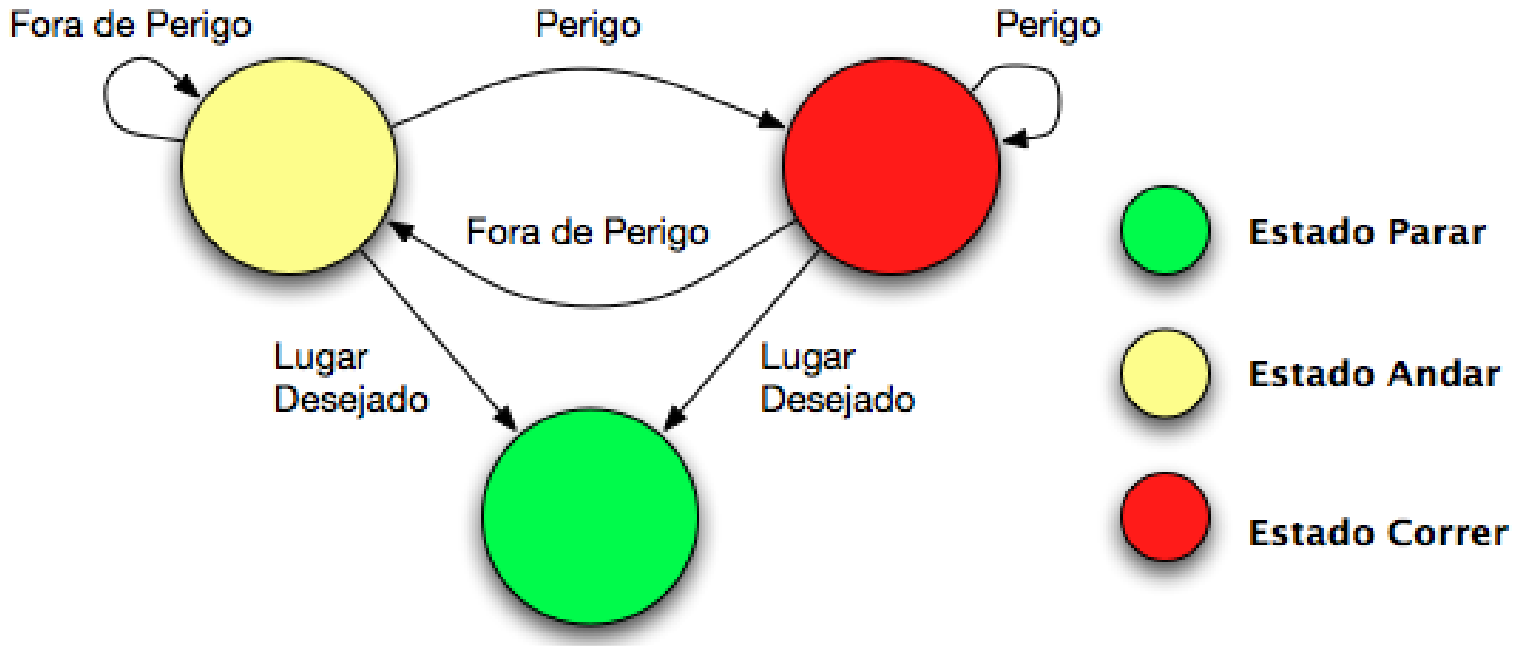


Figura: Exemplo de uma máquina de estados para um agente inteligente

PURO

Taxonomia dos Agentes

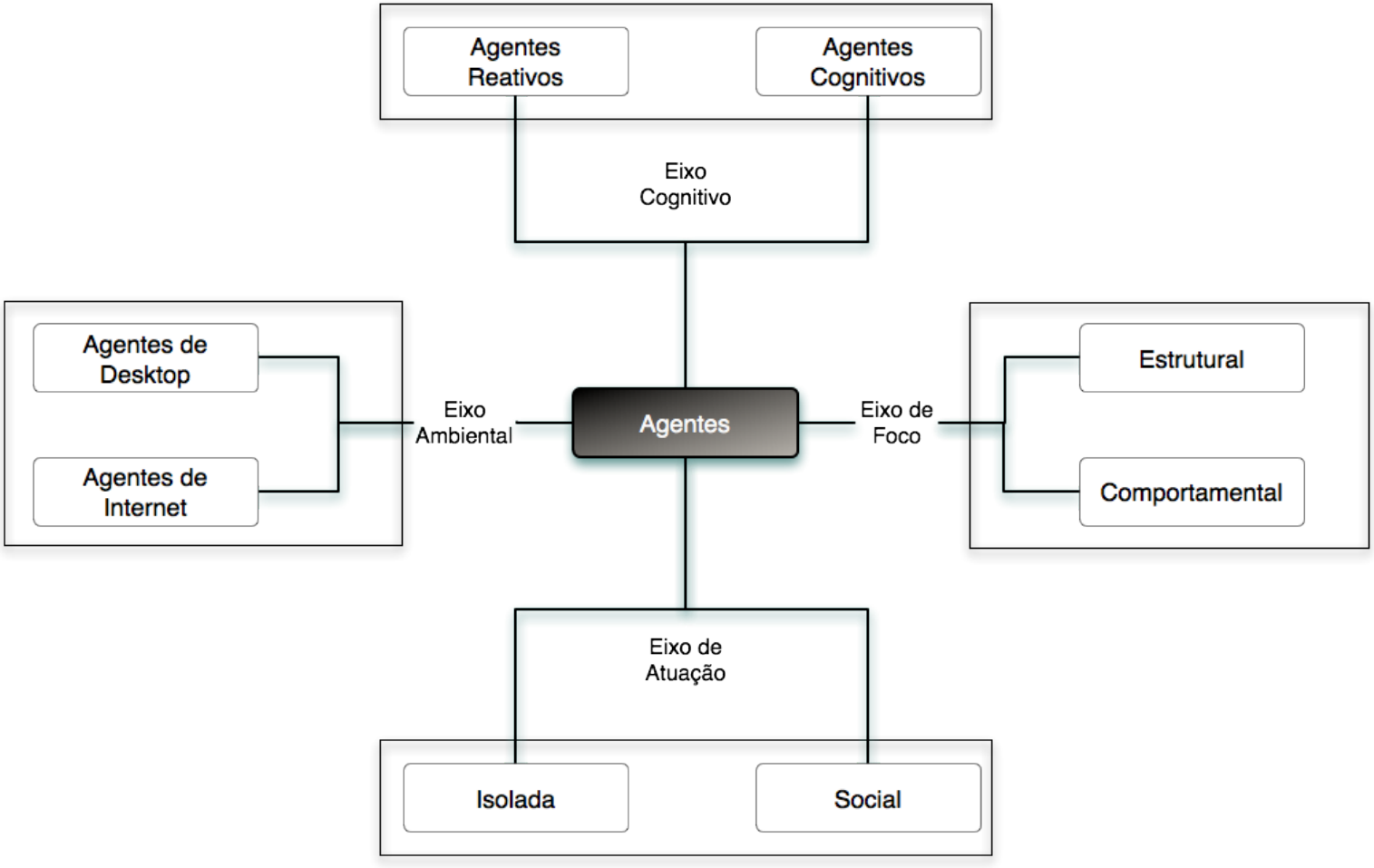


Figura: Taxonomia dos Agentes [Garcia and Sichman, 2003]

IA Distribuída

- Até o momento falamos como o agente resolve problemas de forma isolada.

IA Distribuída

- Até o momento falamos como o agente resolve problemas de forma isolada.
- Apesar de existirem problemas de forma isolada, eles representam uma pequena parcela
- Uma inteligência distribuída precisa de:

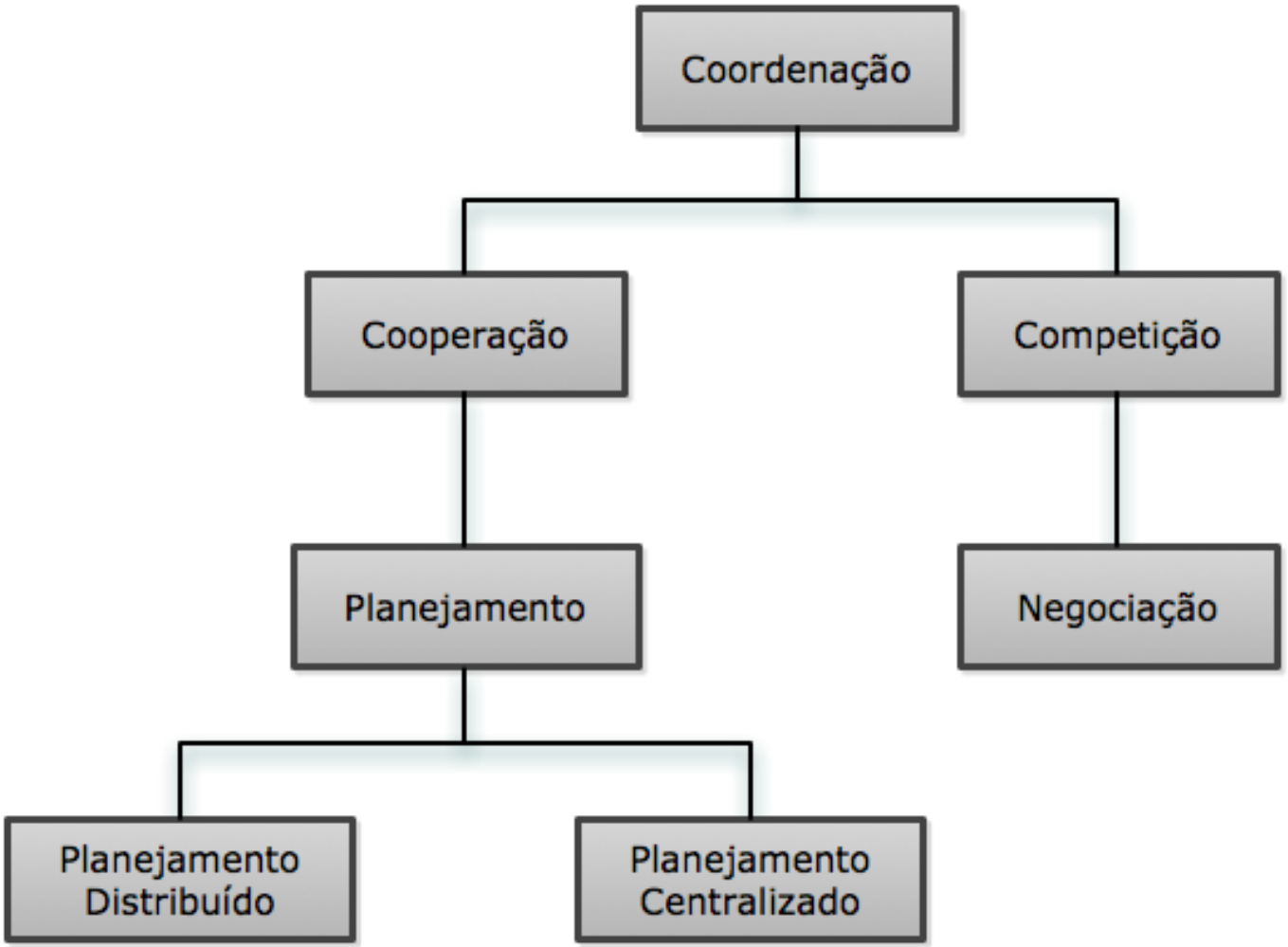
IA Distribuída

- Até o momento falamos como o agente resolve problemas de forma isolada.
- Apesar de existirem problemas de forma isolada, eles representam uma pequena parcela
- Uma inteligência distribuída precisa de:
  - Infraestrutura
  - Interação entre os agentes
  - Planejamento Distribuído
  - Protocolos de Comunicação
  - Protolocos de Negociação

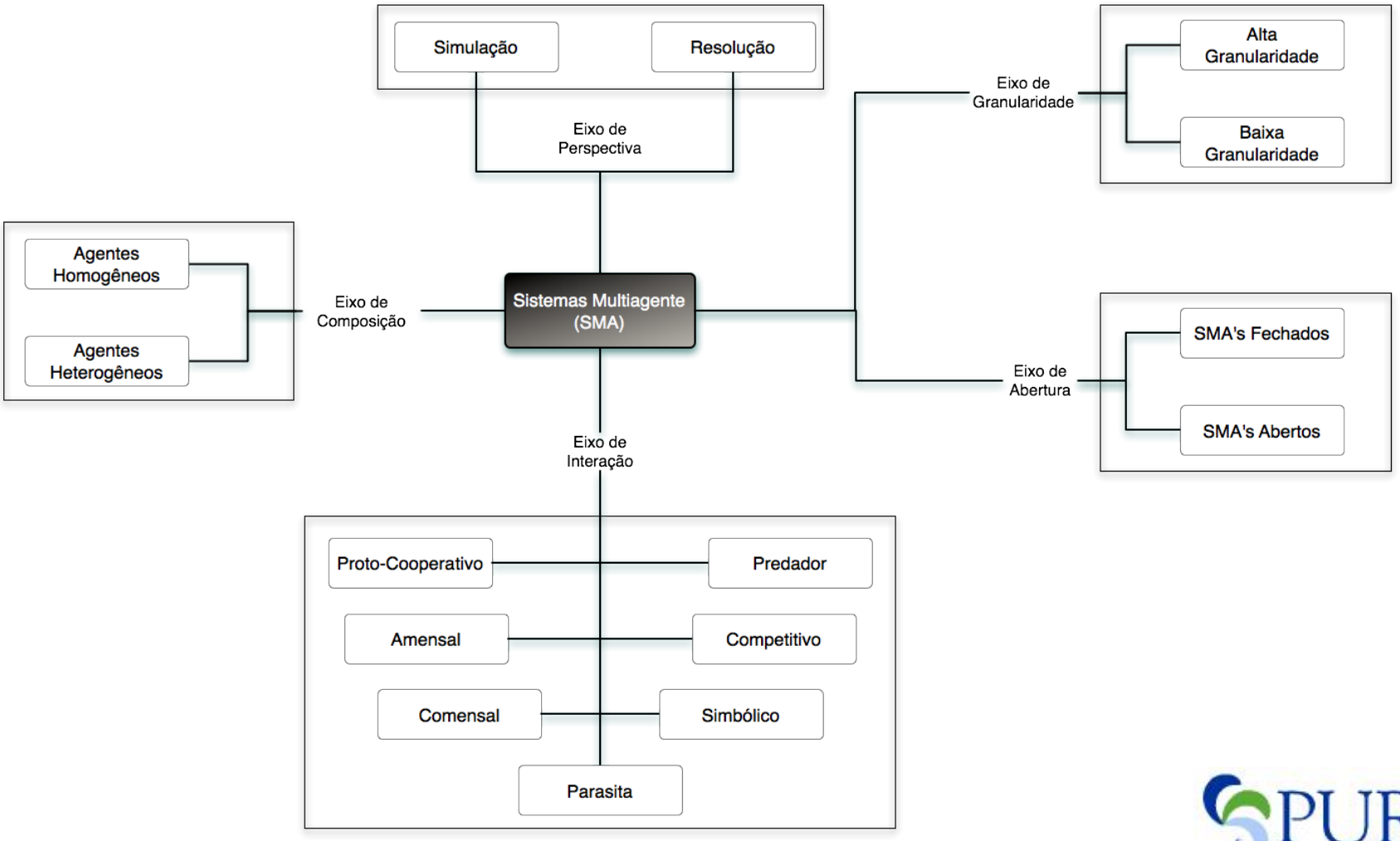
Definição

"Um Sistema Multiagente é um sistema que consiste de um número de agentes que ***interagem uns com os outros****. Além disto, para interagir de forma eficaz, os agentes deste sistema devem ser capazes de* ***cooperar, se coordenar e negociar entre si****." [Wolldridge, 2002]*

Taxonomia de Comportamento [Wolldridge, 2002]



Taxonomia Dos Sistemas Multiagentes [Garcia and Sichman, 2003]



Programação Orientada a Agentes(AOP)

- Paradigma de Computação centrado na teoria de agentes

Programação Orientada a Agentes(AOP)

- Paradigma de Computação centrado na teoria de agentes
- Arquitetura *peer-to-peer*

Programação Orientada a Agentes(AOP)

- Paradigma de Computação centrado na teoria de agentes
- Arquitetura *peer-to-peer*
- Necessita da criação de padrões

Programação Orientada a Agentes(AOP)

- Paradigma de Computação centrado na teoria de agentes
- Arquitetura *peer-to-peer*
- Necessita da criação de padrões
- Podem ser facilmente modelados para Linguagens Orientado a Objetos

Similaridades

Diferenças

PURO

Programação Orientada a Agentes(AOP)

- Paradigma de Computação centrado na teoria de agentes
- Arquitetura *peer-to-peer*
- Necessita da criação de padrões
- Podem ser facilmente modelados para Linguagens Orientado a Objetos

Similaridades

- Encapsulamento
- Herança
- Comunicação

Diferenças

PURO

Programação Orientada a Agentes(AOP)

- Paradigma de Computação centrado na teoria de agentes
- Arquitetura *peer-to-peer*
- Necessita da criação de padrões
- Podem ser facilmente modelados para Linguagens Orientado a Objetos

Similaridades

- Encapsulamento
- Herança
- Comunicação

Diferenças

- Autonomia dos Agentes
- Comportamento Dinâmico
- Linha de execução própria do agente

PURO

FIPA

- Do inglês *Foundation For Intelligent Physical Agents*, criada em 1996

FIPA

- Do inglês *Foundation For Intelligent Physical Agents*, criada em 1996
- Objetivo: Desenvolver padrões para desenvolvimento de aplicações relacionadas a teoria de agentes
- Define questões importantes como:

FIPA

- Do inglês *Foundation For Intelligent Physical Agents*, criada em 1996
- Objetivo: Desenvolver padrões para desenvolvimento de aplicações relacionadas a teoria de agentes
- Define questões importantes como:
  - Controle dos Agentes
  - Estrutura de mensagens
  - Serviço de transporte de mensagens
  - Ações Comunicativas
  - Protocolos de interação
  - Entre outros.

Framework JADE

- Desenvolvido pela Telecom Itália
- Sob licença LGPL(*Library GNU Public License*)
- Implementado em JAVA

Framework JADE

- Desenvolvido pela Telecom Itália
- Sob licença LGPL(*Library GNU Public License*)
- Implementado em JAVA
- Segue os padrões FIPA

Framework JADE

- Desenvolvido pela Telecom Itália
- Sob licença LGPL(*Library GNU Public License*)
- Implementado em JAVA
- Segue os padrões FIPA
- Para administrar os agentes são criados *Containers* onde os agentes se registram
- Existe um container espacial chamado de *Main Container*, e dois agentes especiais, o DF e o AMS responsáveis pelo Diretório facilitador de Serviço de mensagens respectivamente.

Framework JADE

- Desenvolvido pela Telecom Itália
- Sob licença LGPL(*Library GNU Public License*)
- Implementado em JAVA
- Segue os padrões FIPA
- Para administrar os agentes são criados *Containers* onde os agentes se registram
- Existe um container espacial chamado de *Main Container*, e dois agentes especiais, o DF e o AMS responsáveis pelo Diretório facilitador de Serviço de mensagens respectivamente.
- Uso de *Behaviours*

Sumário Principal

PURO

Histórico da GPU(*Graphics Processing Unit*)

- Primeiros controladores gráficos aparecem nos anos 80, com o Atari 8 bits

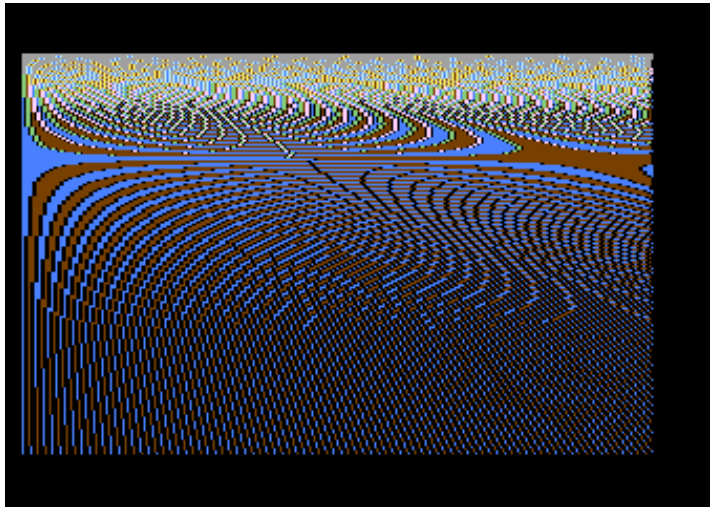


Figura: Exemplo de tela do Atari 8 bits. Resolução 320x192

PURO

Histórico da GPU(*Graphics Processing Unit*)

- Primeiros controladores gráficos aparecem nos anos 80, com o Atari 8 bits
- Chips acelaradores 2D começam a surgir nos anos 90

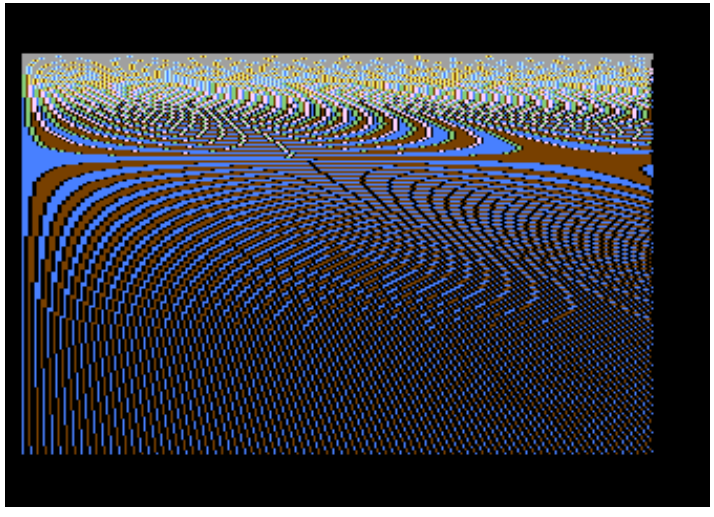


Figura: Exemplo de tela do Atari 8 bits. Resolução 320x192

PURO

Histórico da GPU(*Graphics Processing Unit*)

- Primeiros controladores gráficos aparecem nos anos 80, com o Atari 8 bits
- Chips acelaradores 2D começam a surgir nos anos 90
- Soluções 3D integrando CPU e GPU em 97

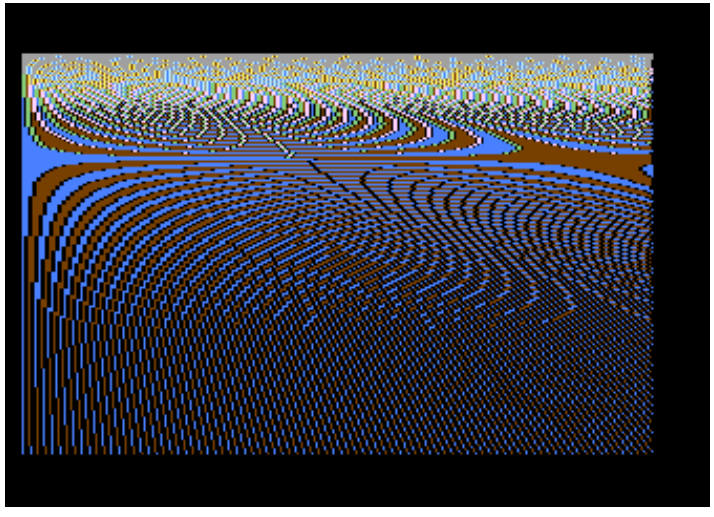


Figura: Exemplo de tela do Atari 8 bits. Resolução 320x192

PURO

Histórico da GPU(*Graphics Processing Unit*)

- Primeiros controladores gráficos aparecem nos anos 80, com o Atari 8 bits
- Chips acelaradores 2D começam a surgir nos anos 90
- Soluções 3D integrando CPU e GPU em 97
- Surgem API's como OpenGL e DirectX

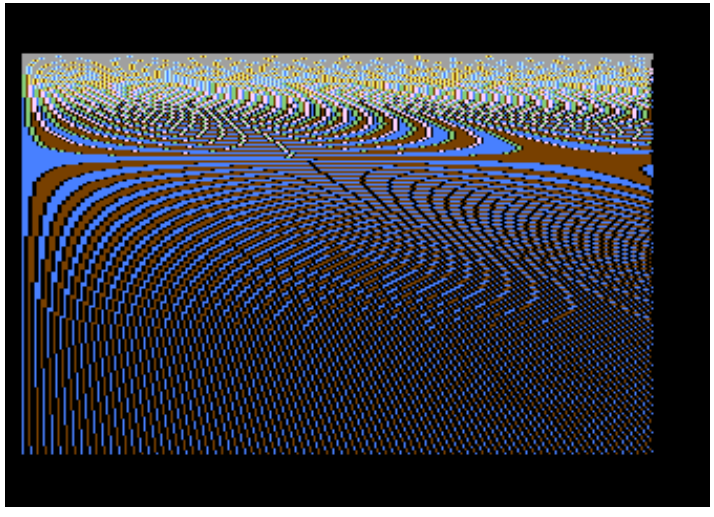


Figura: Exemplo de tela do Atari 8 bits. Resolução 320x192

PURO

Histórico da GPU(*Graphics Processing Unit*)

- Primeiros controladores gráficos aparecem nos anos 80, com o Atari 8 bits
- Chips acelaradores 2D começam a surgir nos anos 90
- Soluções 3D integrando CPU e GPU em 97
- Surgem API's como OpenGL e DirectX
- Em 1999 surge a primeira arquitetura com Shaders

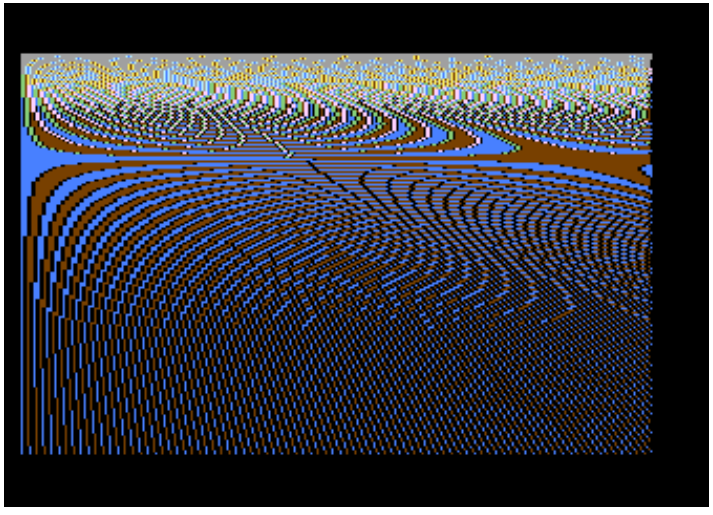
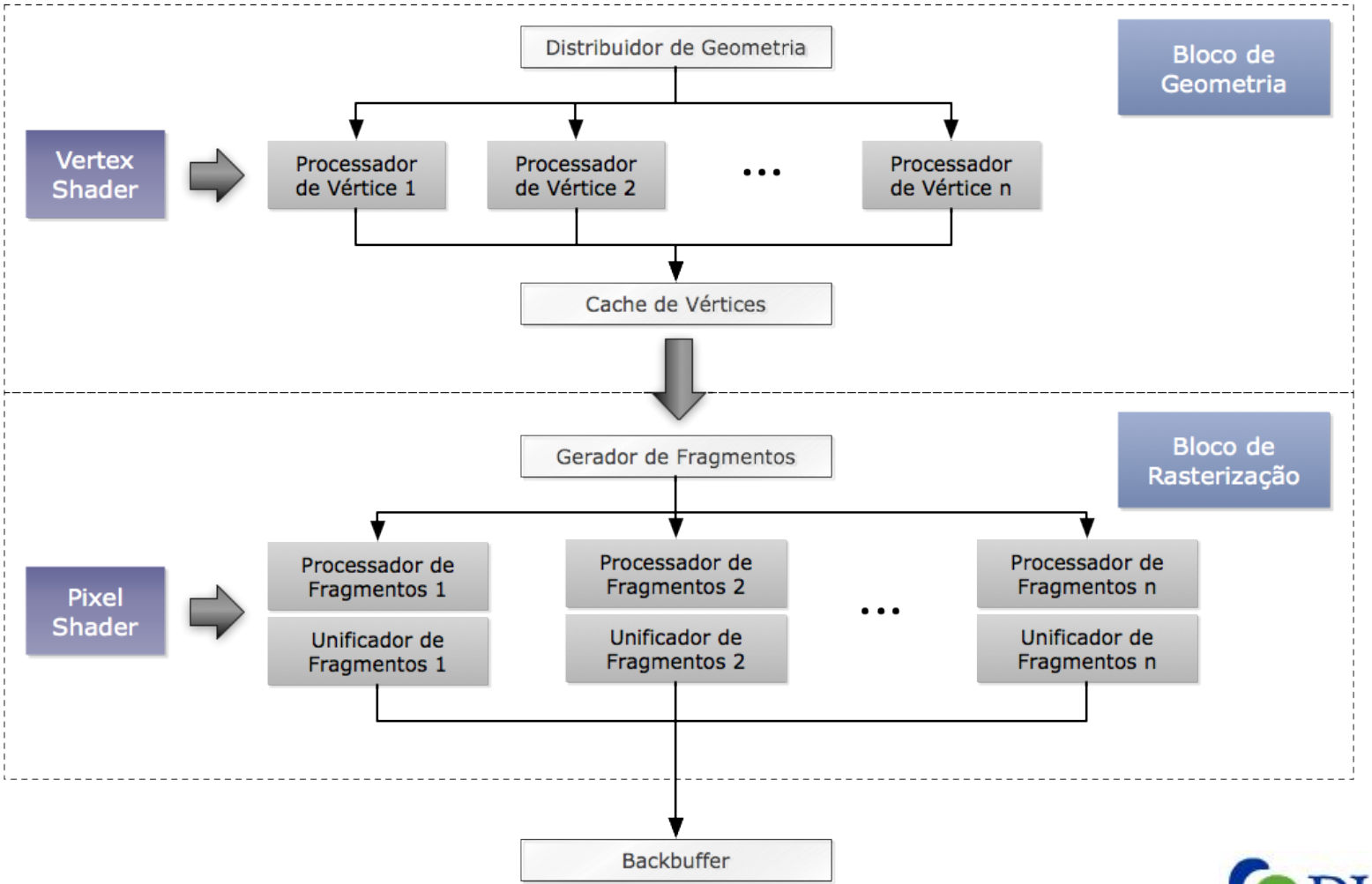


Figura: Exemplo de tela do Atari 8 bits. Resolução 320x192

PURO

Arquitetura de GPU com Shaders



GPGPU(*General Purpose Programming using a Graphics Processing Unit*)

- Até 2006 era difícil criar um programa não-gráfico para GPU
- A cada ano a diferença entre GPUxCPU em GFlops aumenta

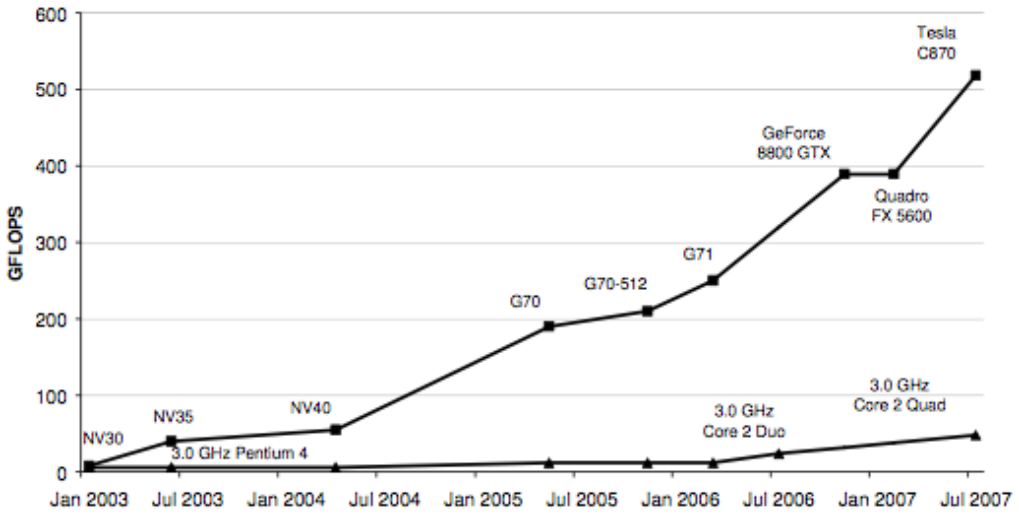


Figura: Diferença de Performance CPUxGPU em GFlops

PURO

GPGPU(*General Purpose Programming using a Graphics Processing Unit*)

- Até 2006 era difícil criar um programa não-gráfico para GPU
- A cada ano a diferença entre GPUxCPU em GFlops aumenta
- Quanto mais escaláveis e quanto maior carga de trabalho, melhor
- Em 2007 Temos o CUDA [NVIDIA, 2007]

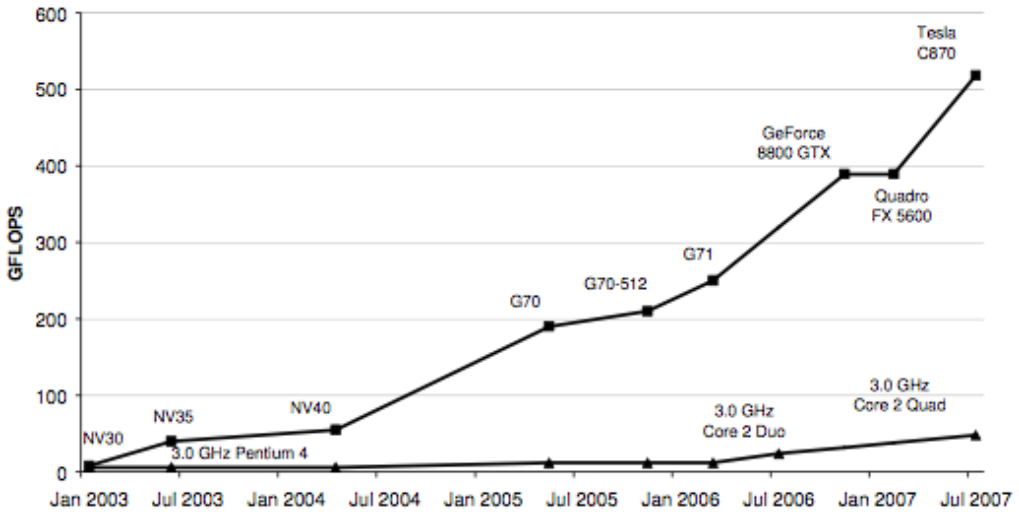
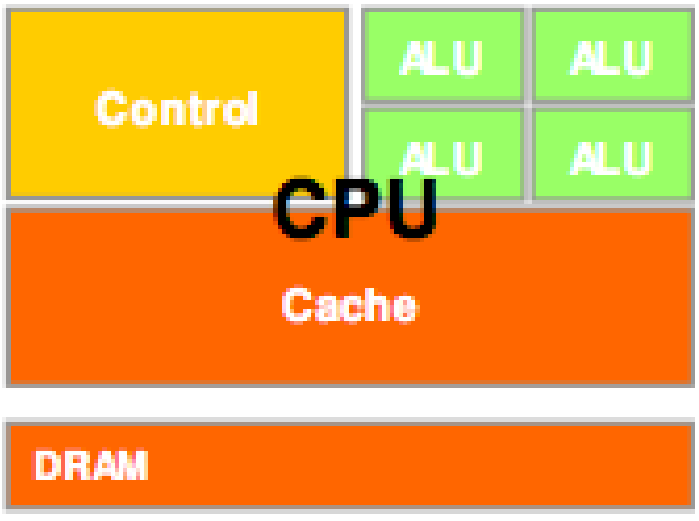
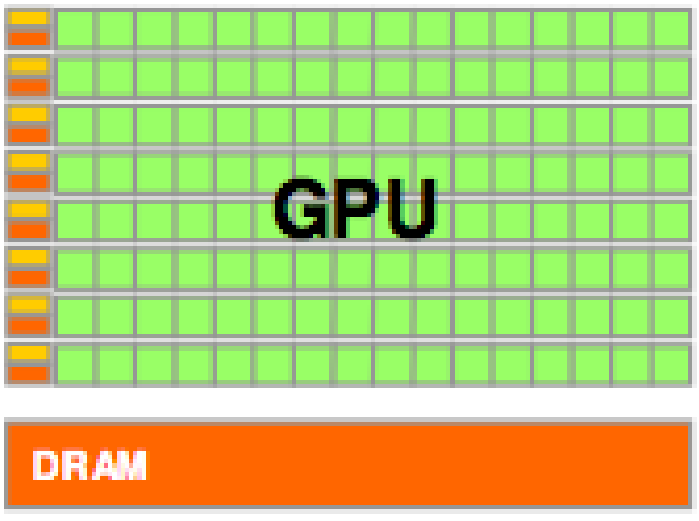


Figura: Diferença de Performance CPUxGPU em GFlops

PURO

Diferença de Arquitetura entre CPU e GPU





PURO

Características

- Extensão de C

Características

- Extensão de C
- *Host* $\times$ *Device*
- 3 abstrações principais:

Características

- Extensão de C
- *Host* $\times$ *Device*
- 3 abstrações principais:
  - Hierarquia de Threads
  - Hierarquia de Memória
  - Barreiras de Sincronização

Características

- Extensão de C
- *Host* × *Device*
- 3 abstrações principais:
  - Hierarquia de Threads
  - Hierarquia de Memória
  - Barreiras de Sincronização
- Funções especiais chamadas de *kernels*
- Uso de palavras-chave para determinar tipos de variáveis e tipos de função
- Não tem Recursão ou Alocação Dinâmica

Hierarquia de Threads

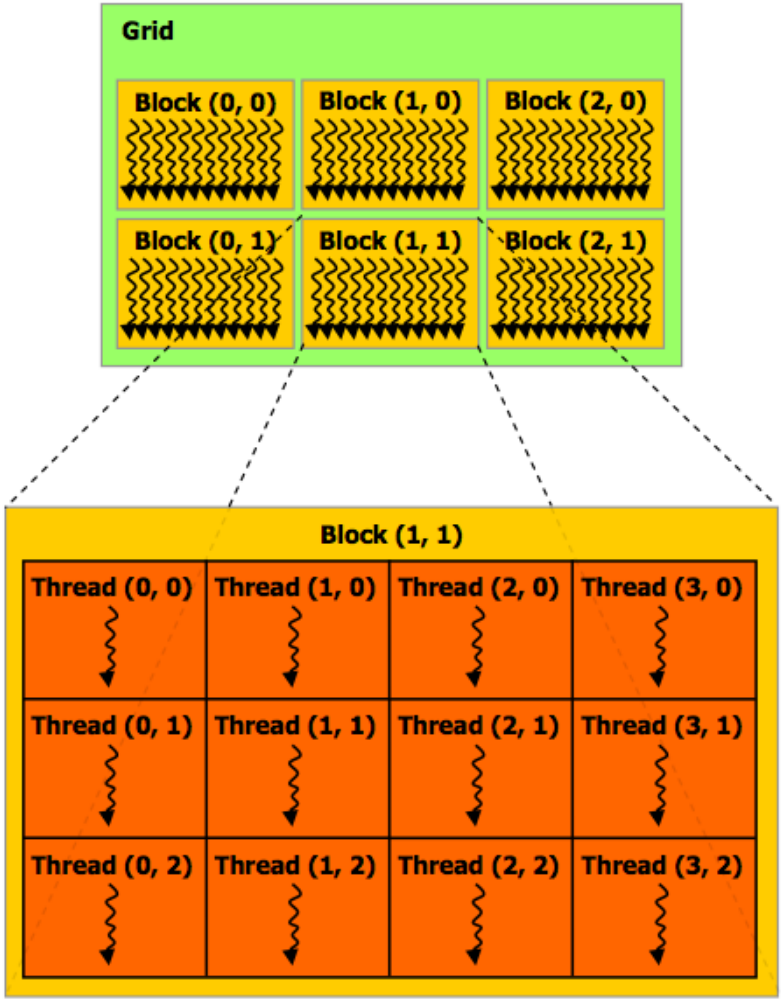
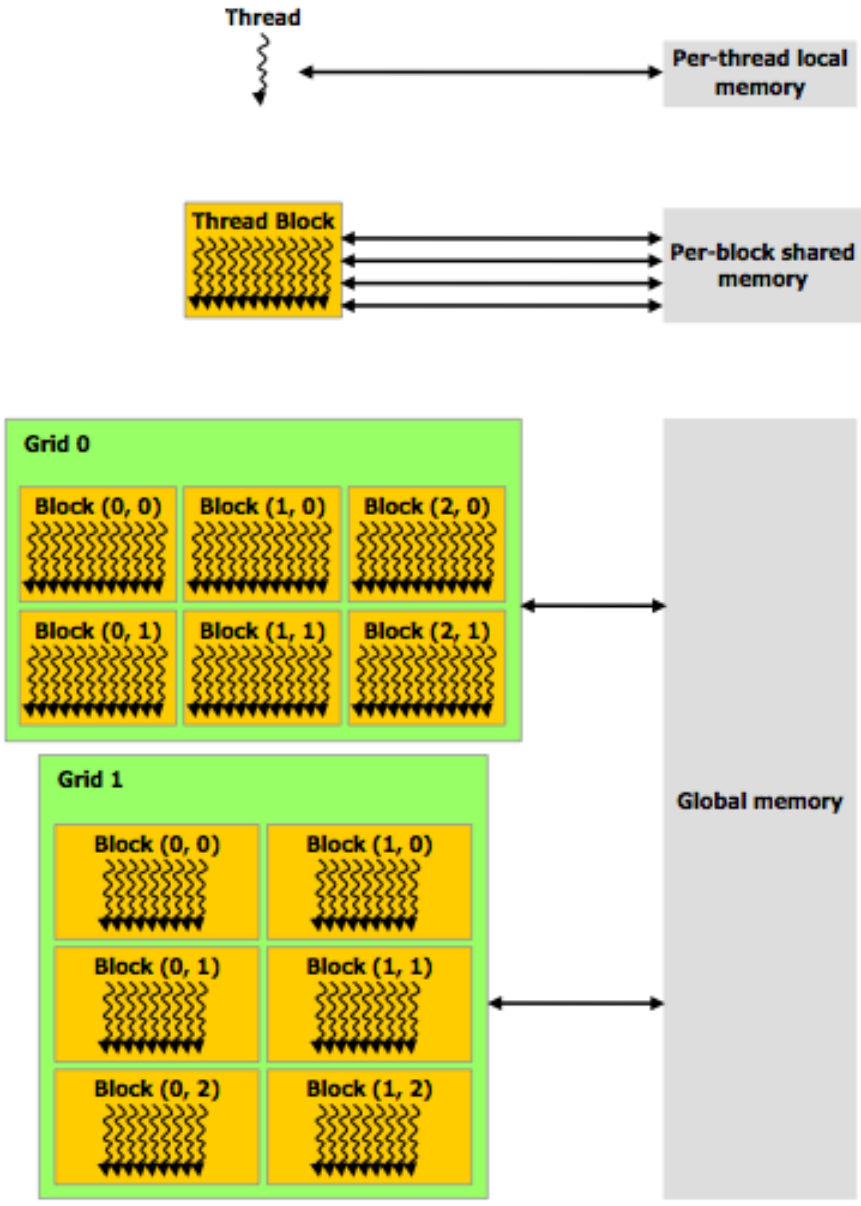


Figura: Grid de Threads

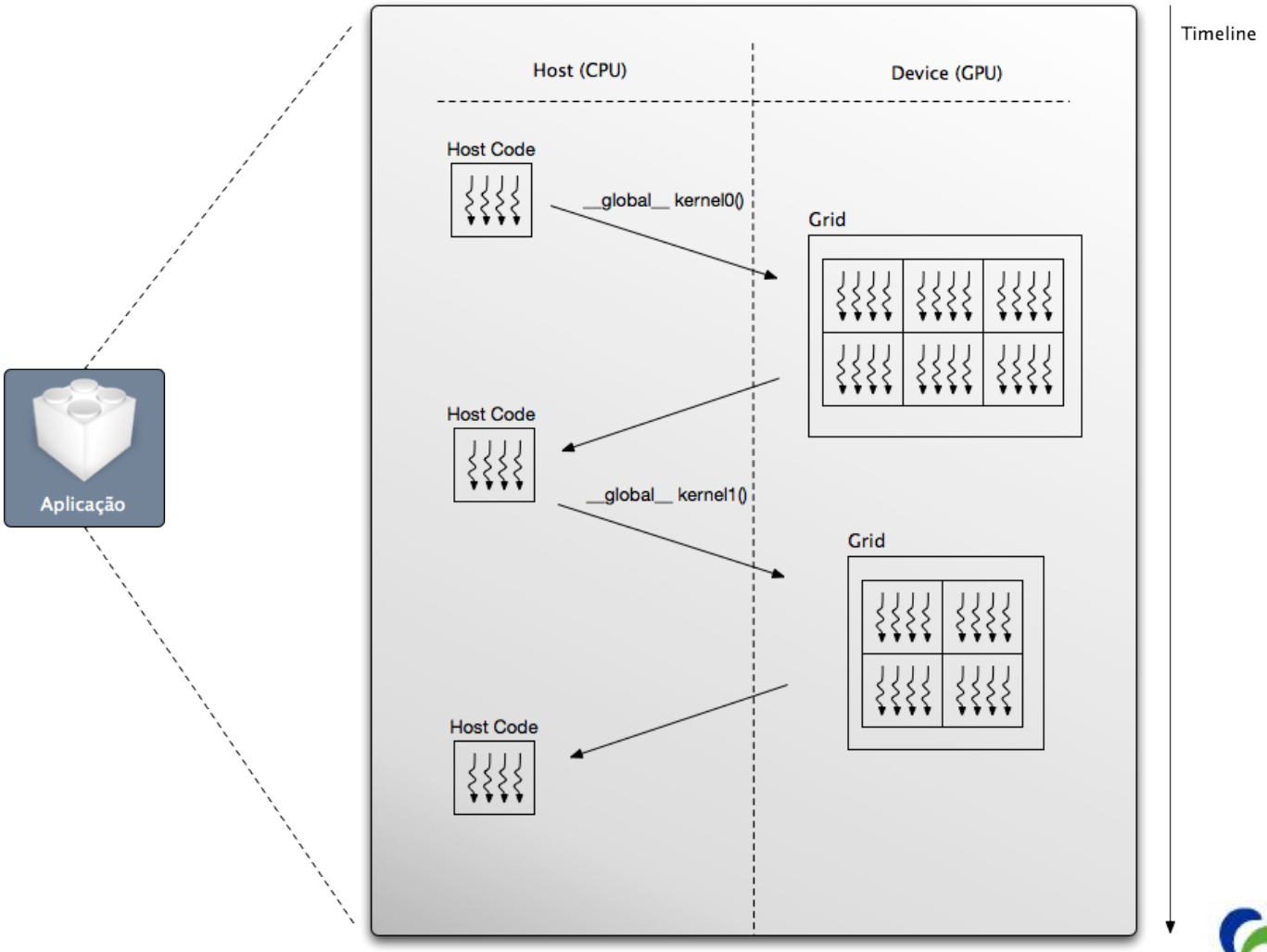
PURO

Hierarquia de Memória



PURO

Programação Heterogênea



PURO

Exemplo de Código

```
// Definicao da variavel compartilhada entre as threads
__shared__ float vet[N];

// Definicao do Kernel
__global__ void corners(float vet[N]){
int index = threadIdx.x;

__syncthreads()
if((index + 1) < N && (index - 1) > 0){
vet[index] = vet[index]+ vet[index+1] + vet[index-1];
}
}

int main(){
...
// Chamada do Kernel
corners<<<1,N>>>(vet[N]);
}
```

Sumário Principal

PURO

Motivação e Trabalhos Relacionados

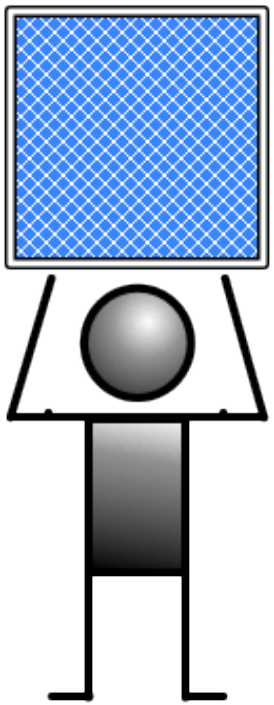
- Muitos trabalhos utilizando Teoria de Agentes, como:
  - Criação de veículos autônomos [Stone and Veloso, 1997]
  - Coordenação de sinais de trânsito [Bazzan, 2005]
  - Sistemas de evacuação [Dimakis et al., 2009]
- Convergência da entre IA e GPGPU, como:
  - Crowd Simulation [Passos et al., 2008]
- Objetivos desse trabalho:

Motivação e Trabalhos Relacionados

- Muitos trabalhos utilizando Teoria de Agentes, como:
  - Criação de veículos autônomos [Stone and Veloso, 1997]
  - Coordenação de sinais de trânsito [Bazzan, 2005]
  - Sistemas de evacuação [Dimakis et al., 2009]
- Convergência da entre IA e GPGPU, como:
  - Crowd Simulation [Passos et al., 2008]
- Objetivos desse trabalho:
  - Mapear um sistema multiagente criado a partir de uma arquitetura distribuída, para arquiteturas GPGPU
  - Mostrar diferenças, vantagens, desvantagens e uma análise de desempenho

Problema dos Caixotes

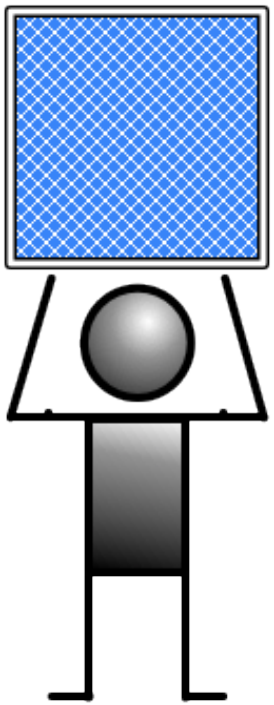
- Os agentes devem passar os caixotes de um lado para o outro
- Interação em um mapa unidimensional
- Objetivos do Estudo:



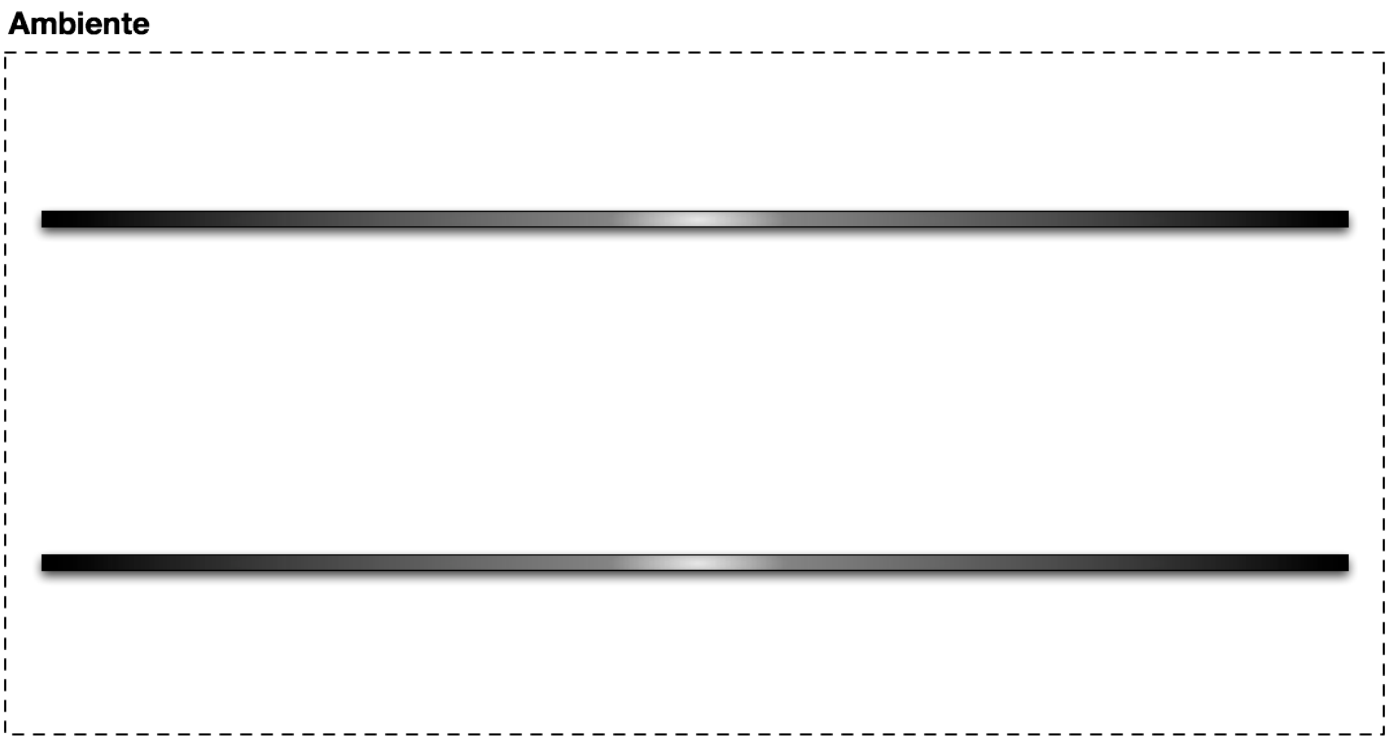
PURO

Problema dos Caixotes

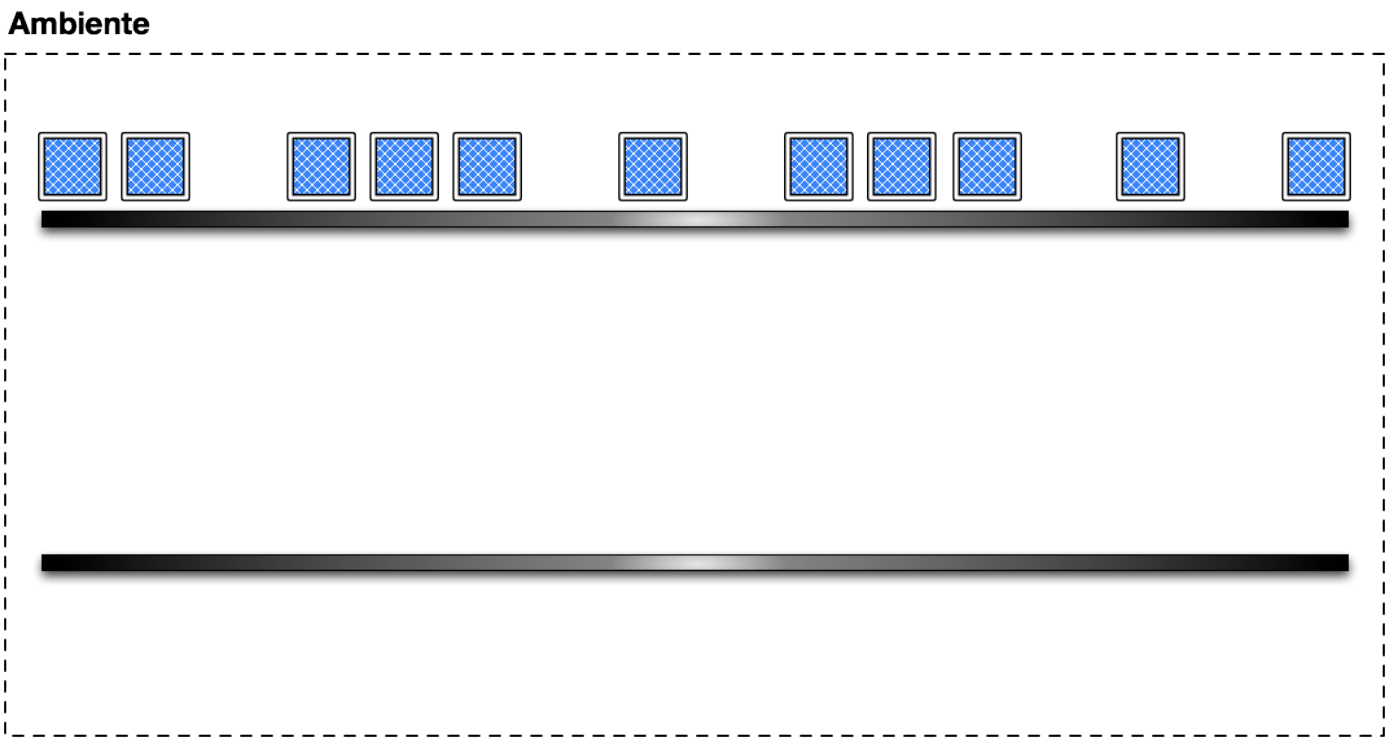
- Os agentes devem passar os caixotes de um lado para o outro
- Interação em um mapa unidimensional
- Objetivos do Estudo:
  - Criação do ambiente
  - Interação dos agentes
  - Número elevado de agentes



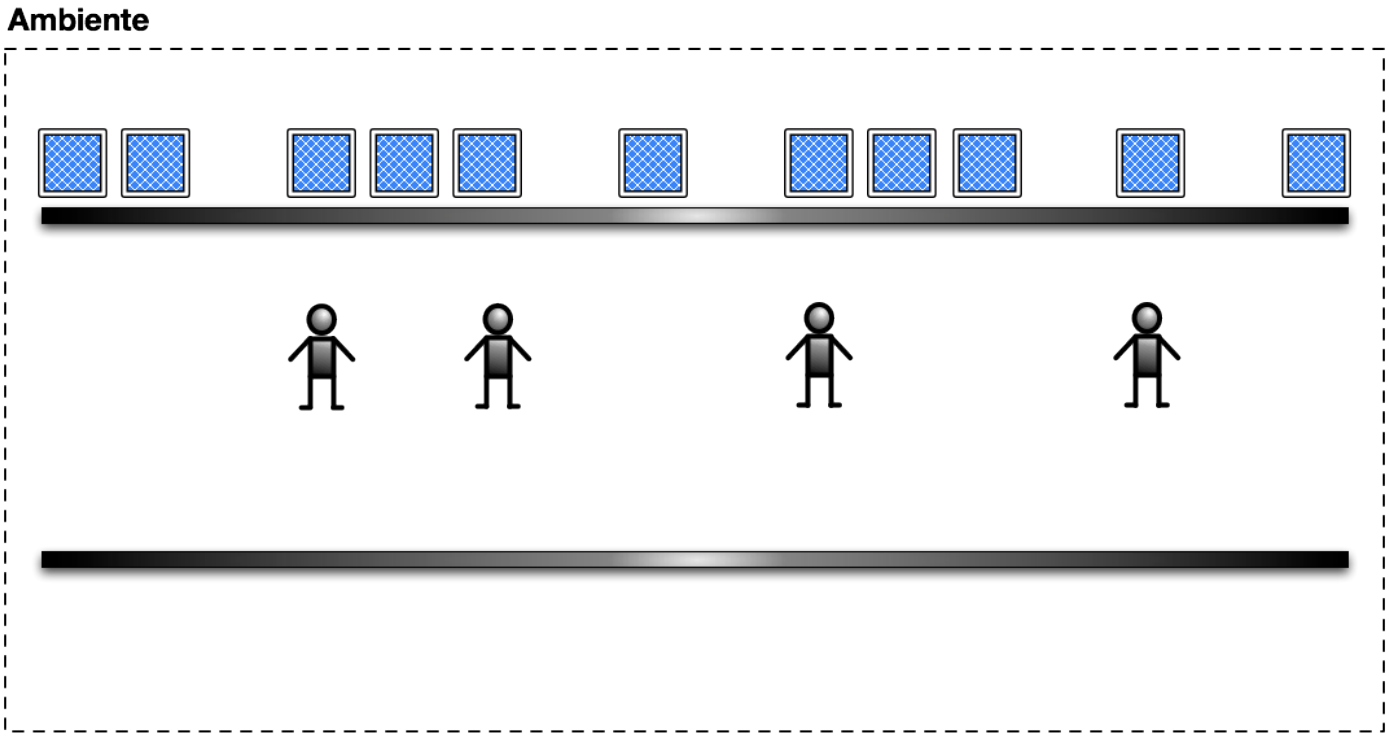
Representação do Sistema



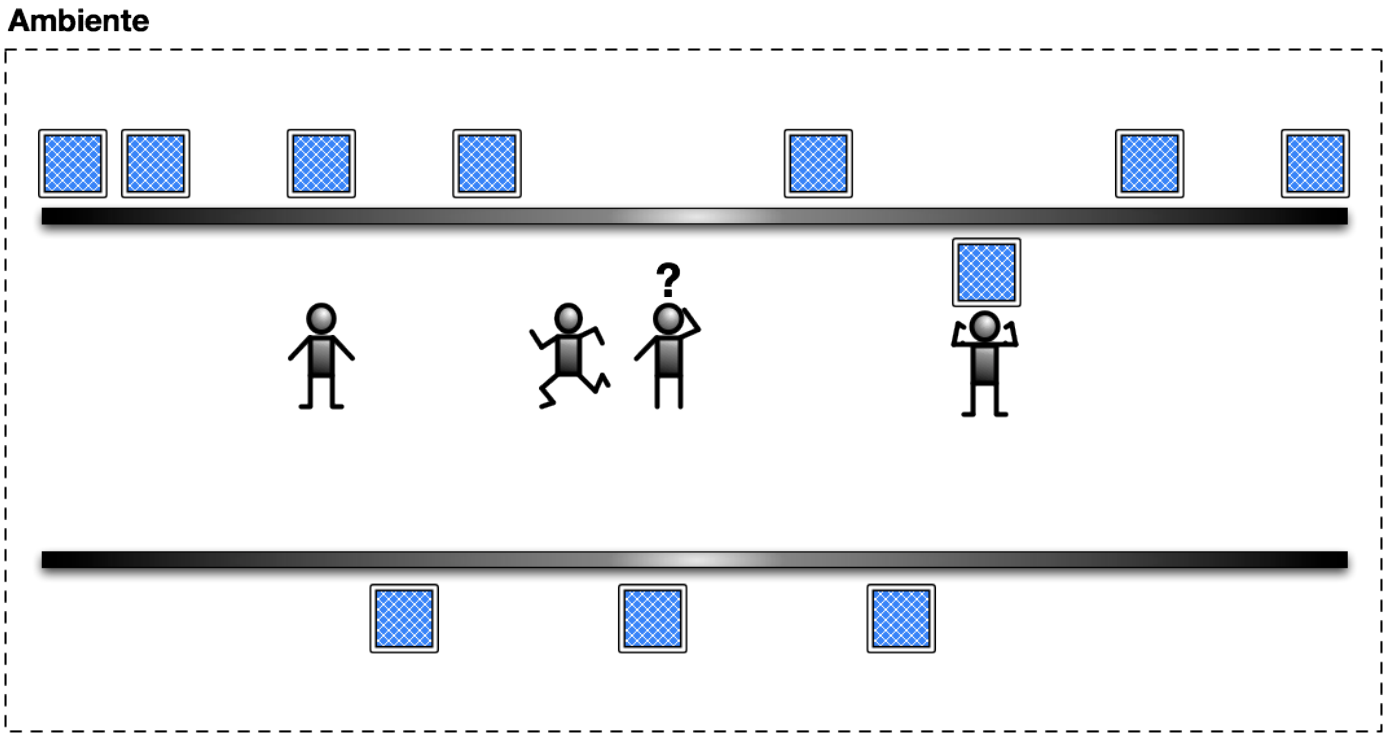
Representação do Sistema



Representação do Sistema



Representação do Sistema



Mapeando de JADE para CUDA

JADE

- Agentes:
- Inteligência:
- Cenário:

CUDA

- Agentes:
- Inteligência:
- Cenário:

PURO

Mapeando de JADE para CUDA

JADE

- Agentes:
  - Criados a partir de uma classe que herda atributos da classe `Agent` e alocados em um `AgentContainer`.
- Inteligência:
- Cenário:

CUDA

- Agentes:
  - Vetor Unidimensional, onde cada índice do vetor representa a ID do agente.
- Inteligência:
- Cenário:

PURO

Mapeando de JADE para CUDA

JADE

- Agentes:
  - Criados a partir de uma classe que herda atributos da classe `Agent` e alocados em um `AgentContainer`.
- Inteligência:
  - Classe privada que herda atributos da classe `Behaviour`, e contem métodos `action()` e `done()`
- Cenário:

CUDA

- Agentes:
  - Vetor Unidimensional, onde cada índice do vetor representa a ID do agente.
- Inteligência:
  - Programada diretamente no *kernel*
- Cenário:

PURO

Mapeando de JADE para CUDA

JADE

- Agentes:
  - Criados a partir de uma classe que herda atributos da classe `Agent` e alocados em um `AgentContainer`.
- Inteligência:
  - Classe privada que herda atributos da classe `Behaviour`, e contem métodos `action()` e `done()`
- Cenário:
  - Classe `Scenario`

CUDA

- Agentes:
  - Vetor Unidimensional, onde cada índice do vetor representa a ID do agente.
- Inteligência:
  - Programada diretamente no *kernel*
- Cenário:
  - Vetor unidimensional com variáveis booleanas

PURO

Mapeando de JADE para CUDA

JADE

- Vantagens:
- Desvantagens:

CUDA

- Vantagens:
- Desvantagens:

PURO

Mapeando de JADE para CUDA

JADE

- Vantagens:
  - Ferramental provido pelo *framework* e pela linguagem JAVA
  - Alto nível Abstração
- Desvantagens:

CUDA

- Vantagens:
  - Criação de agentes na chamada do *kernel*
  - Alta escalabilidade
- Desvantagens:

PURO

Mapeando de JADE para CUDA

JADE

- Vantagens:
  - Ferramental provido pelo *framework* e pela linguagem JAVA
  - Alto nível Abstração
- Desvantagens:
  - Muito trabalho em problemas simples
  - Criação dos agentes

CUDA

- Vantagens:
  - Criação de agentes na chamada do *kernel*
  - Alta escalabilidade
- Desvantagens:
  - Grande uso de estruturas de dados
  - Alocação na GPU(`cudaMalloc()` e `cudaMemCpy()`)

PURO

Análise de Performance

Máquina utilizada:

- CPU Intel Core 2 Duo 2.26GHz
- 4GB de RAM DDR3
- GPU NVidia Geforce 9400M
- 256MB DDR3 para a GPU

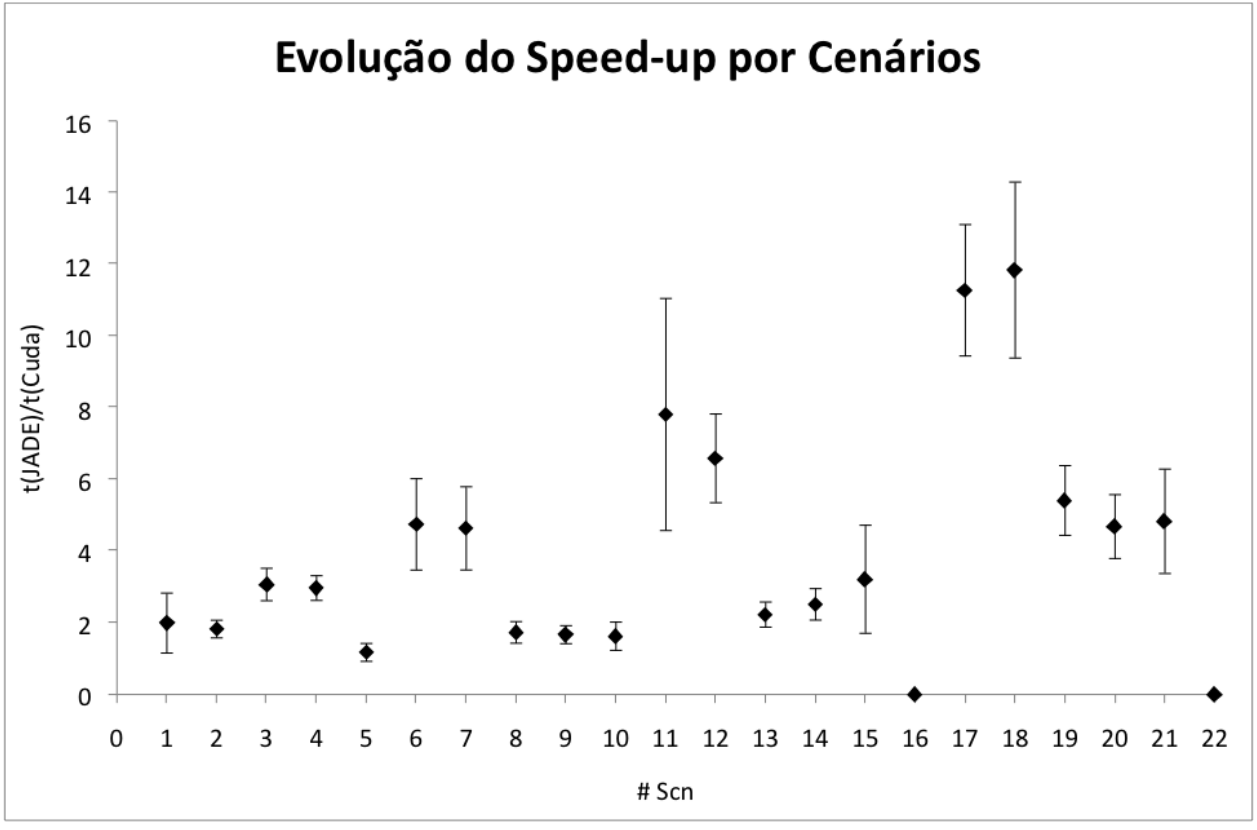
Análise de Performance: Casos de Teste

Scn	# Agentes	# Caixotes
1	250	500
2	500	500
3	250	1,000
4	500	1,000
5	1,000	1,000
6	250	2,000
7	500	2,000
8	1,000	2,000
9	1,500	2,000
10	2,000	2,000
11	250	5,000

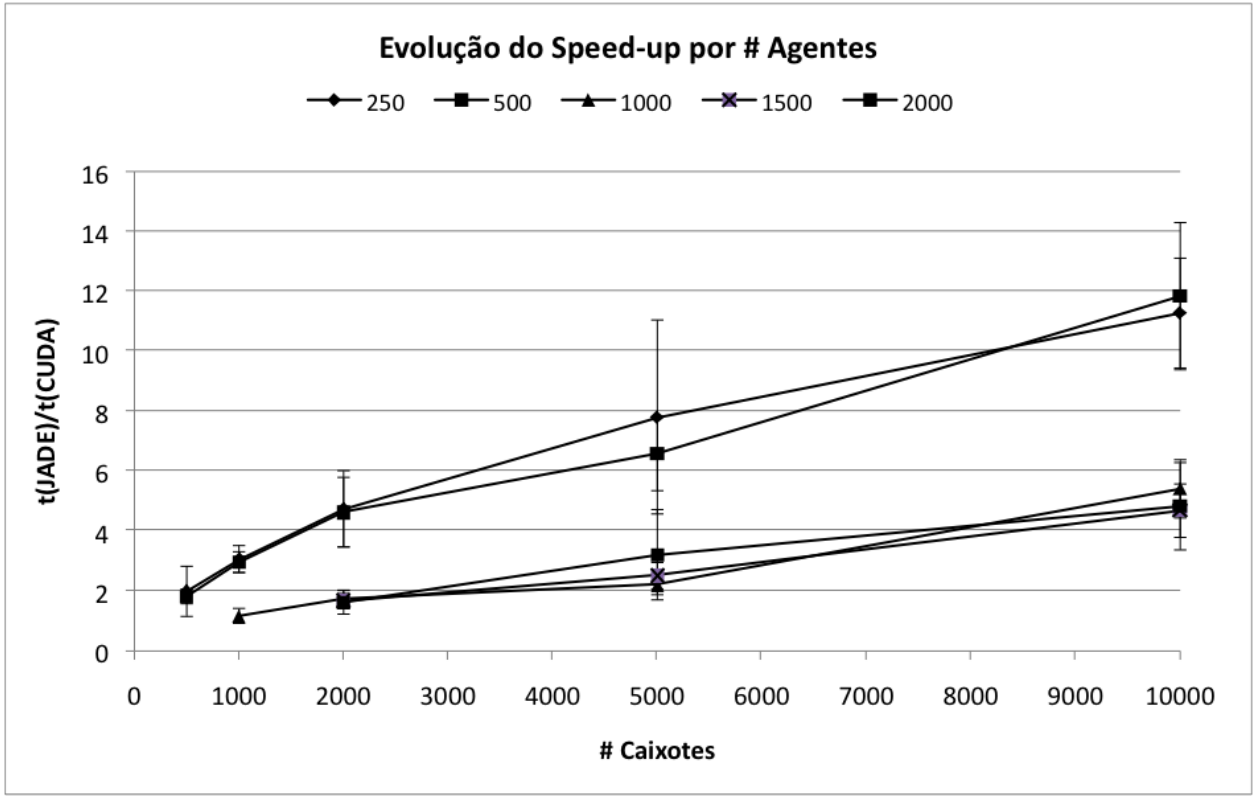
Scn	# Agentes	# Caixotes
12	500	5,000
13	1,000	5,000
14	1,500	5,000
15	2,000	5,000
16	5,000	5,000
17	250	1,0000
18	500	1,0000
19	1,000	10,000
20	1,500	10,000
21	2,000	10,000
22	5,000	10,000

PURO

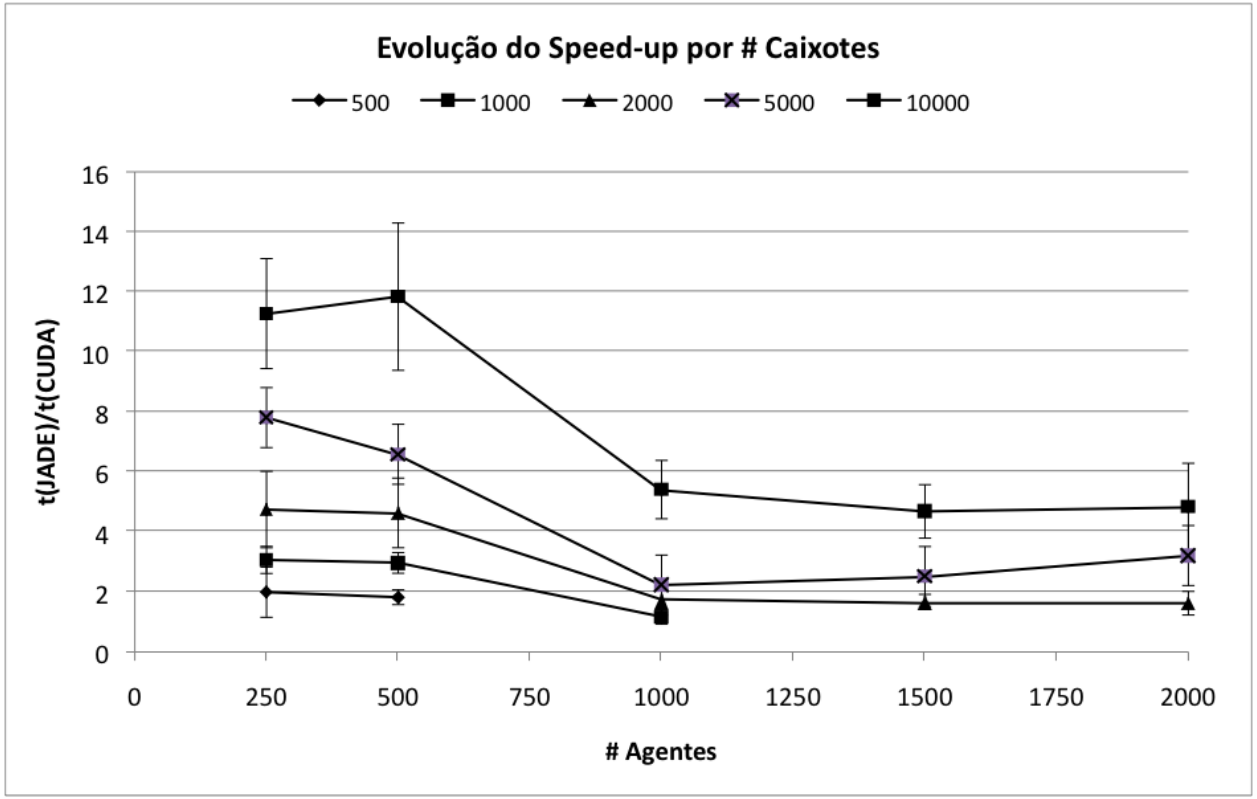
Análise de Performance: Gráficos



Análise de Performance: Gráficos

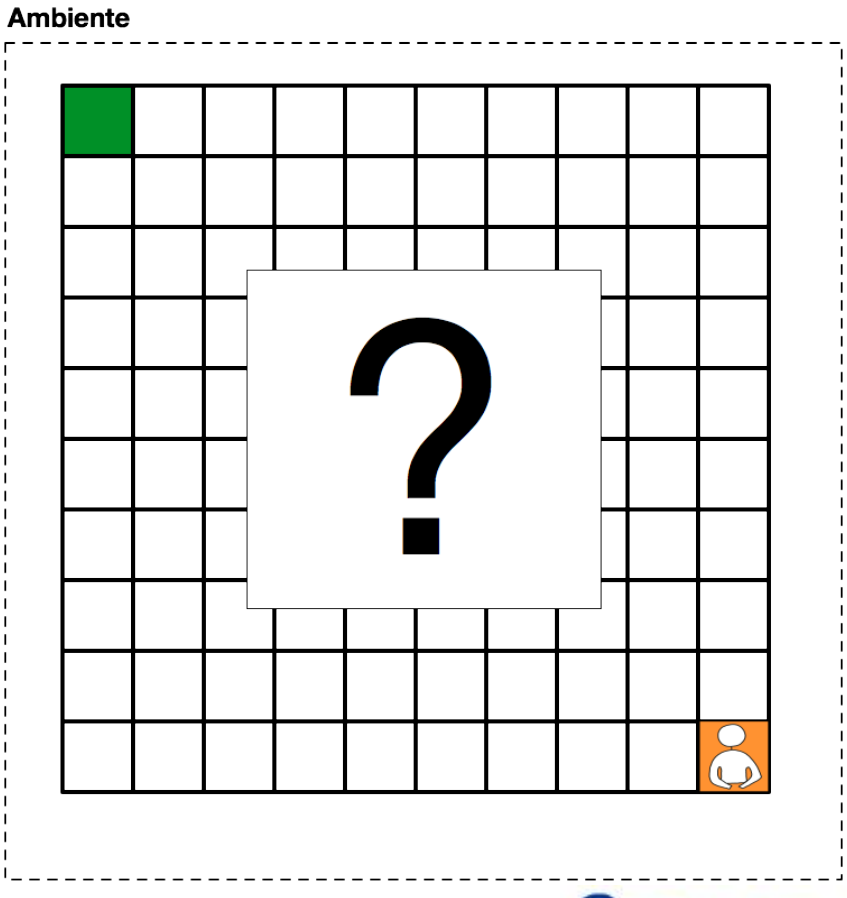


Análise de Performance: Gráficos



Pathfiding A*

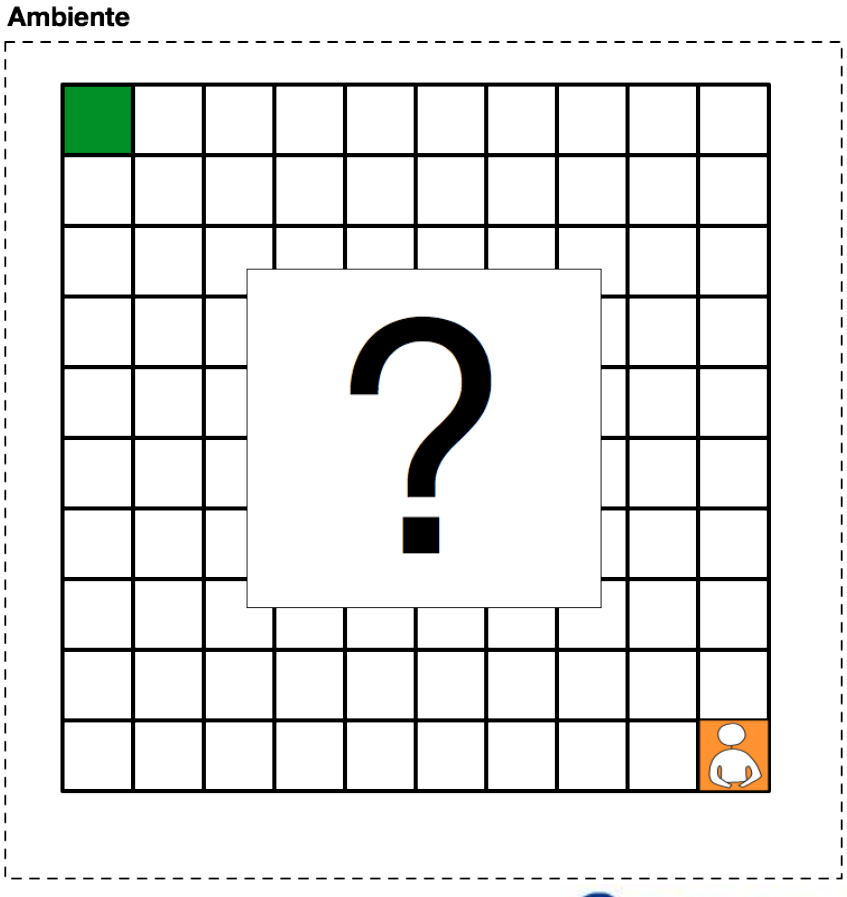
- O agente deverá encontrar o caminho da sua origem até o final (*checkpoint*)
- Interação em um mapa bidimensional
- Objetivos do Estudo:



PURO

Pathfiding A*

- O agente deverá encontrar o caminho da sua origem até o final (*checkpoint*)
- Interação em um mapa bidimensional
- Objetivos do Estudo:
  - Avaliar situações mais complexas
  - Sincronização
  - Busca pelo caminho



PURO

Algoritmo A*

- Algoritmo de busca heurística [Nilson, 1971]

Algoritmo A*

- Algoritmo de busca heurística [Nilson, 1971]
- Avalia o nó n a partir de uma função $f(n) = g(n) + h(n)$, onde:

Algoritmo A*

- Algoritmo de busca heurística [Nilson, 1971]
- Avalia o nó n a partir de uma função $f(n) = g(n) + h(n)$, onde:
  - $g(n)$ é o caminho estendido a partir da soma de custo, ou distância ao longo do caminho
  - $h(n)$ é a estimativa até o nó objetivo
- Recebe como entrada o Mapa, os pontos iniciais e final, e retorna um caminho organizado em uma lista de pontos

Algoritmo A*

- Algoritmo de busca heurística [Nilson, 1971]
- Avalia o nó n a partir de uma função $f(n) = g(n) + h(n)$, onde:
  - $g(n)$ é o caminho estendido a partir da soma de custo, ou distância ao longo do caminho
  - $h(n)$ é a estimativa até o nó objetivo
- Recebe como entrada o Mapa, os pontos iniciais e final, e retorna um caminho organizado em uma lista de pontos
- Possui complexidade média $O(N \exp(C\phi(N))$ [Pearl, 1984], onde:

Algoritmo A*

- Algoritmo de busca heurística [Nilson, 1971]
- Avalia o nó n a partir de uma função $f(n) = g(n) + h(n)$, onde:
  - $g(n)$ é o caminho estendido a partir da soma de custo, ou distância ao longo do caminho
  - $h(n)$ é a estimativa até o nó objetivo
- Recebe como entrada o Mapa, os pontos iniciais e final, e retorna um caminho organizado em uma lista de pontos
- Possui complexidade média $O(N \exp(C\phi(N))$ [Pearl, 1984], onde:
  - $\phi(N) = log(N)^k$
  - N é o número de partições até o nó objetivo
  - k é o número de níveis

PURO

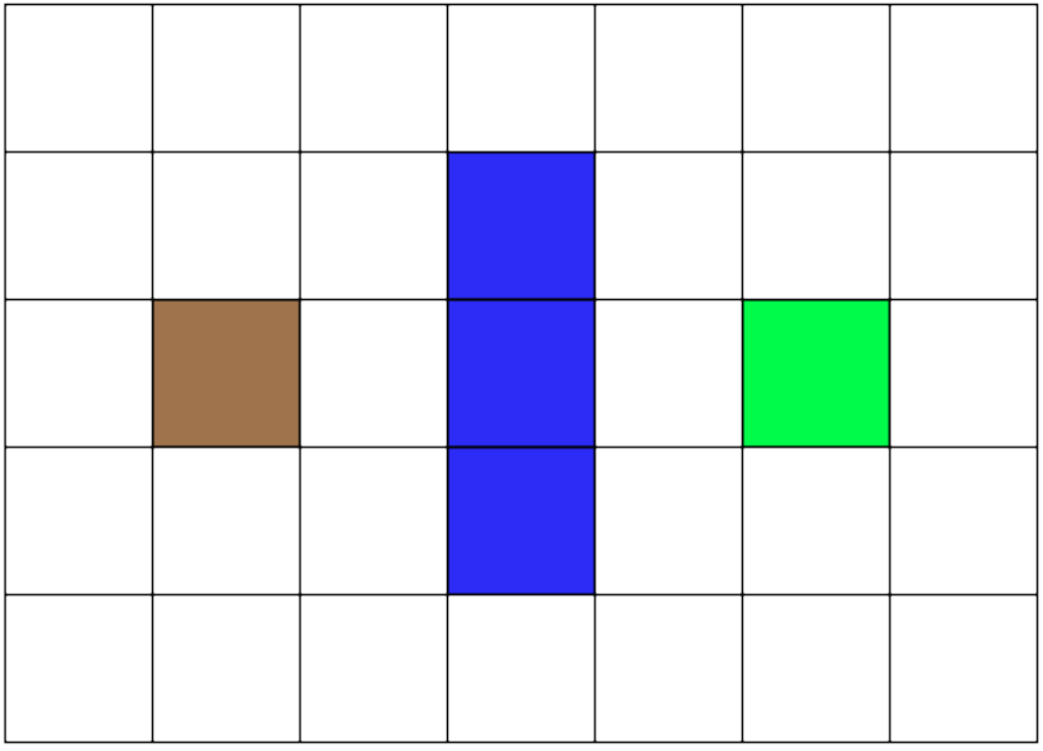
Algoritmo A*

Requer: *Map*: $m \times n$ Mapa Bidimensional.
Requer: *Pos*: Posição do Agente.
Requer: *Checkpoint*: Posição desejada do Agente.
Condição: $ListaAberta = ListaFechada = \{\phi\}$.

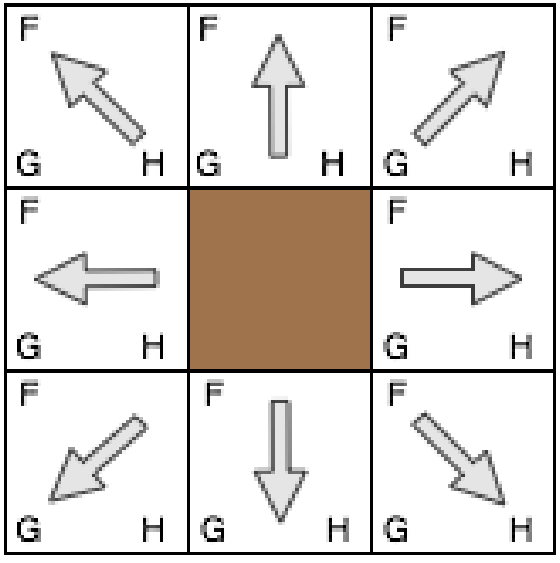
1: $ListaAberta \leftarrow Pos$;
2: **enquanto** $Checkpoint \notin ListaFechada$ and $ListaAberta \neq \{\phi\}$ **faça**
3: $A \leftarrow ListaAberta[0]$;
4: $ListaFechada \leftarrow A$;
5: Remover(A,$ListaAberta$);
6: **para todo** Adjecentes A_i de A **faça**
7: **se** $A_i x, y$ não é um obstáculo e $0 \leq x \leq m$, $0 \leq y \leq n$ **então**
8: **se** $\neg(\exists P,\ P\{x,y\} = A_i\{x,y\}$ e $P \in ListaAberta)$ **então**
9: $ListaAberta \leftarrow A_i$;
10: Ordenar($OpenList$);
11: **senão se** $A_i\{G\} < P\{G\}$ **então**
12: Remover(P,$OpenList$);
13: $OpenList \leftarrow A_i$;
14: Ordenar($OpenList$);
15: **fim se**
16: **fim se**
17: **fim para**
18: **fim enquanto**
19: $Path \leftarrow$Organizar($ClosedList$); // Pegar todos os pontos necessários do caminho.
20: **Retorne** *Path*;

PURO

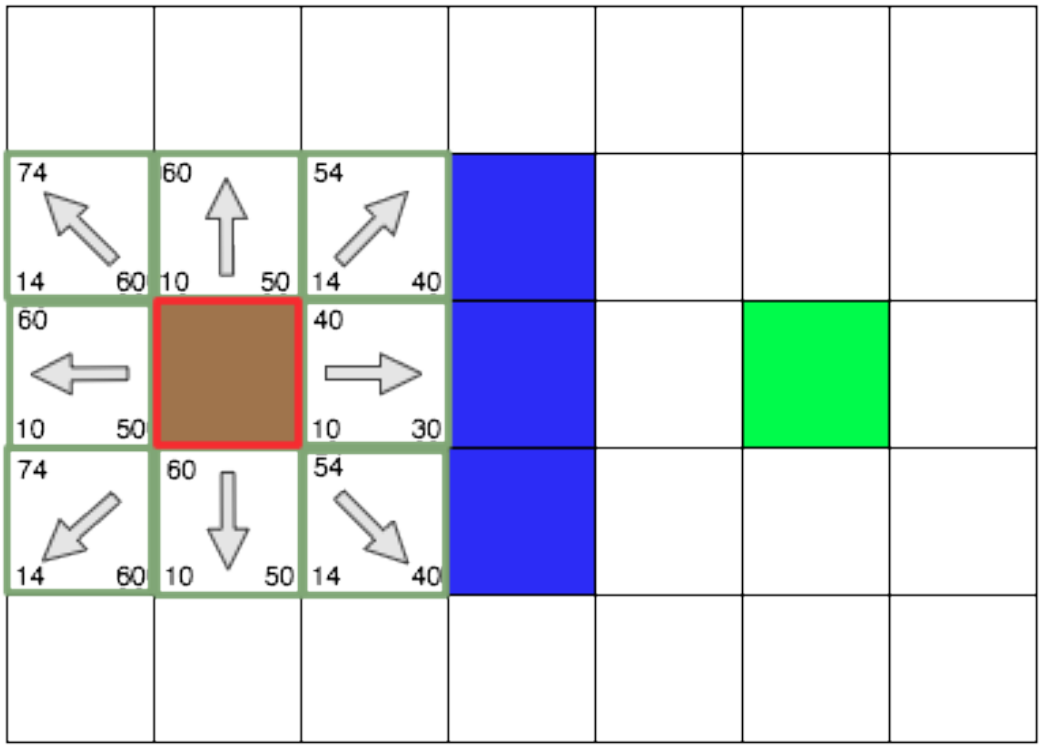
Algoritmo A*



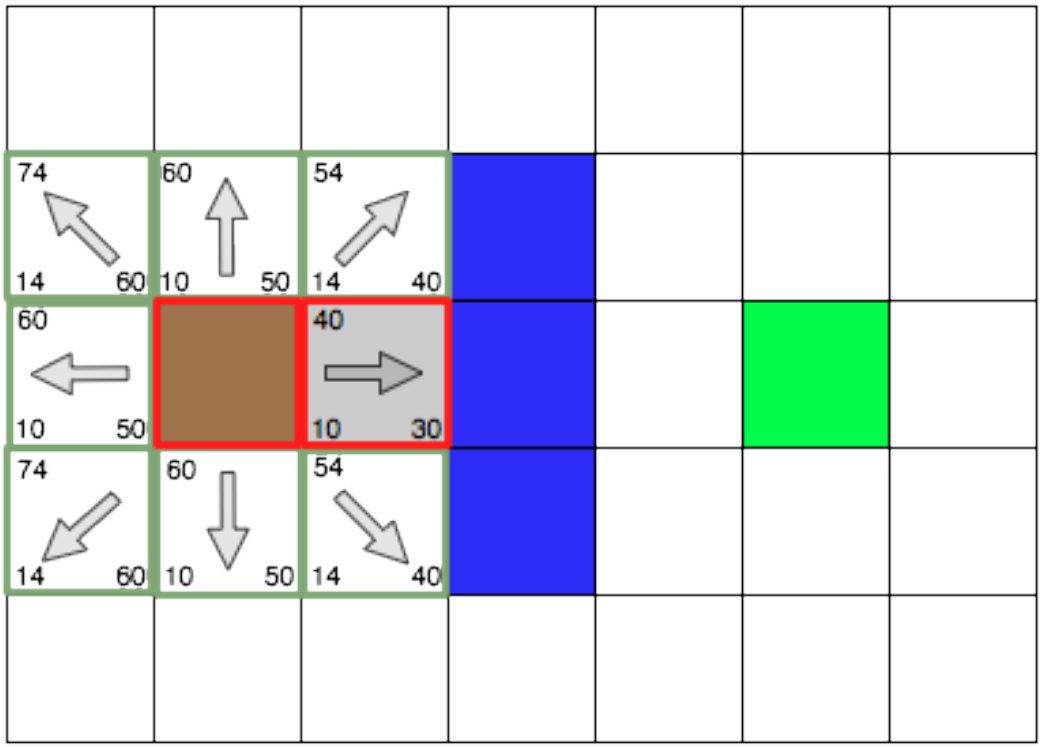
Algoritmo A*



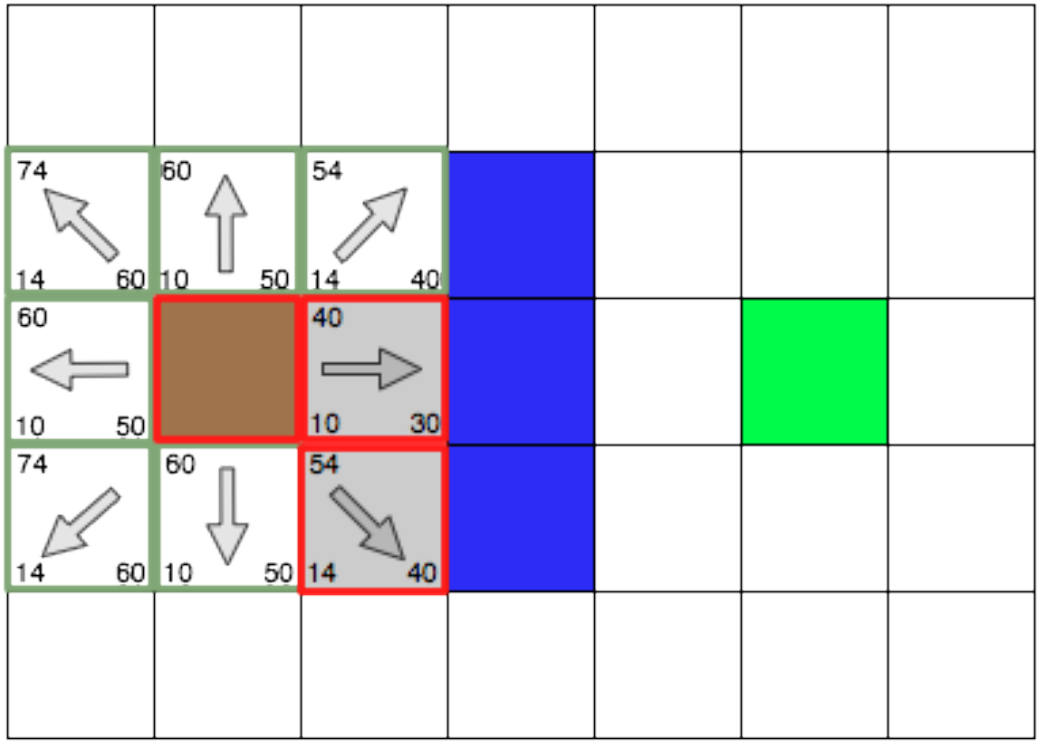
Algoritmo A*



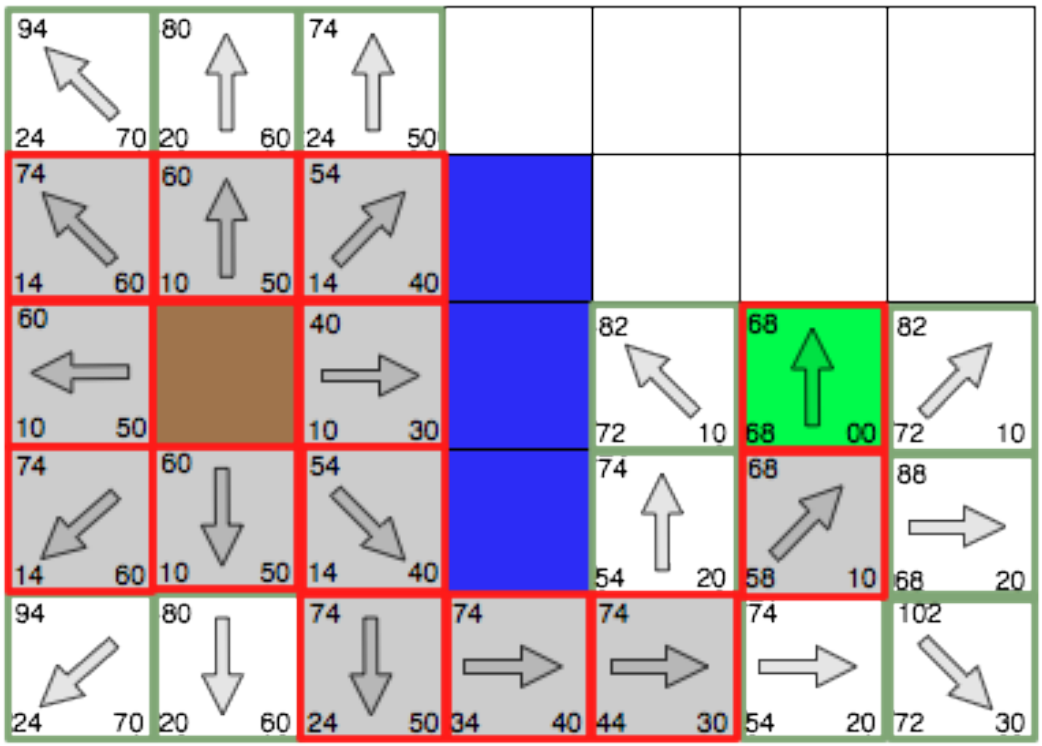
Algoritmo A*



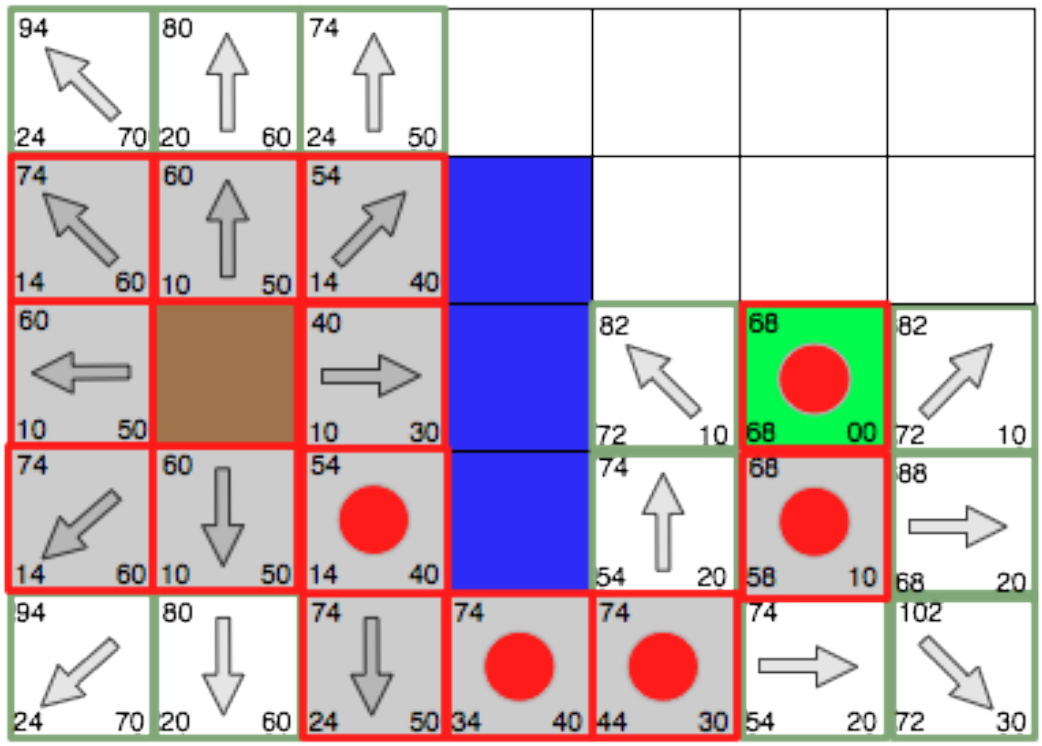
Algoritmo A*



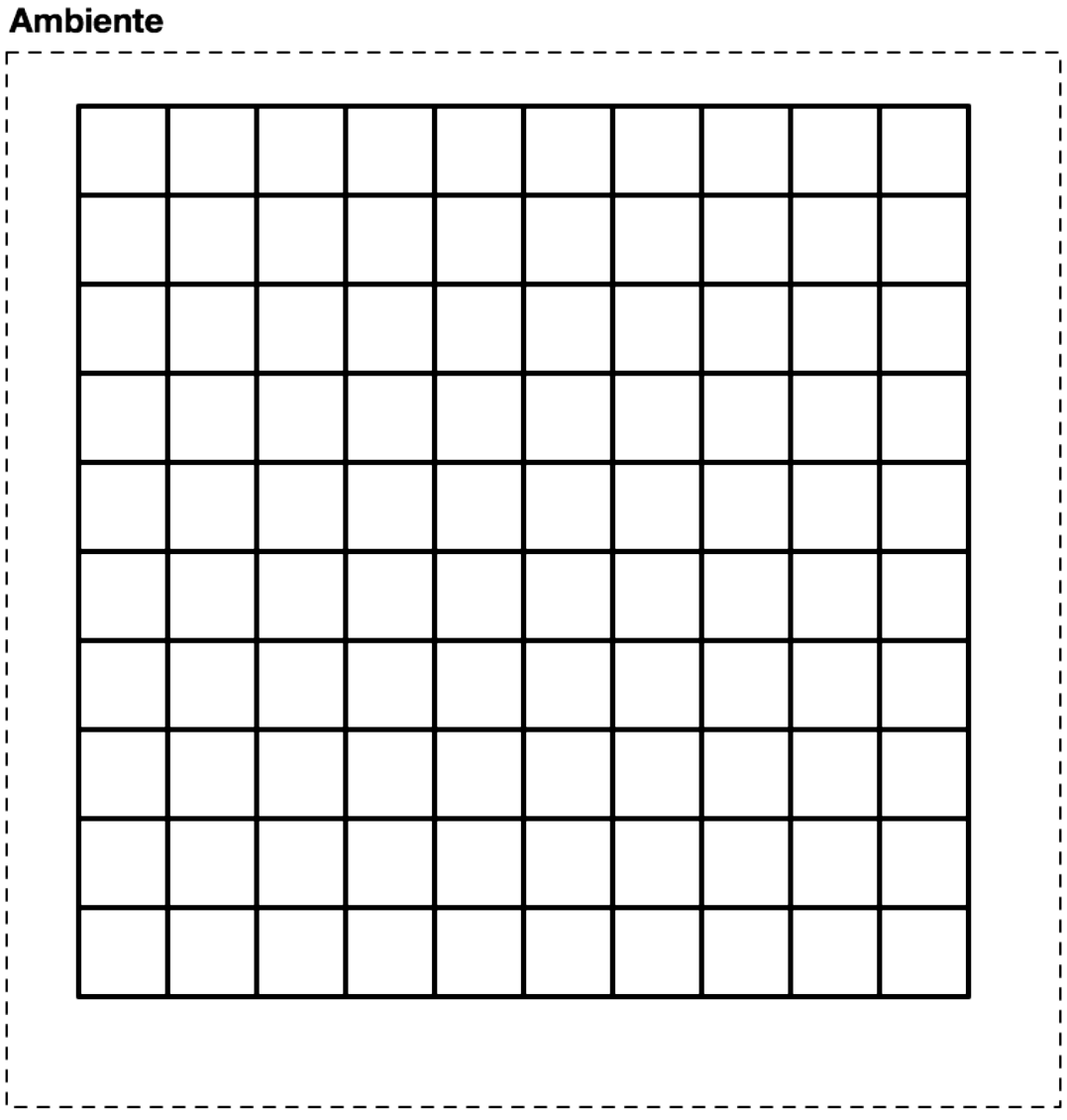
Algoritmo A*


94 24 70
80 20 60
74 24 50
74 14 60
60 10 50
54 14 40
60 10 50
40 10 30
82 72 10
68 68 00
82 72 10
74 14 60
60 10 50
54 14 40
74 54 20
68 58 10
88 68 20
94 24 70
80 20 60
74 24 50
74 34 40
74 44 30
74 54 20
102 72 30

Algoritmo A*

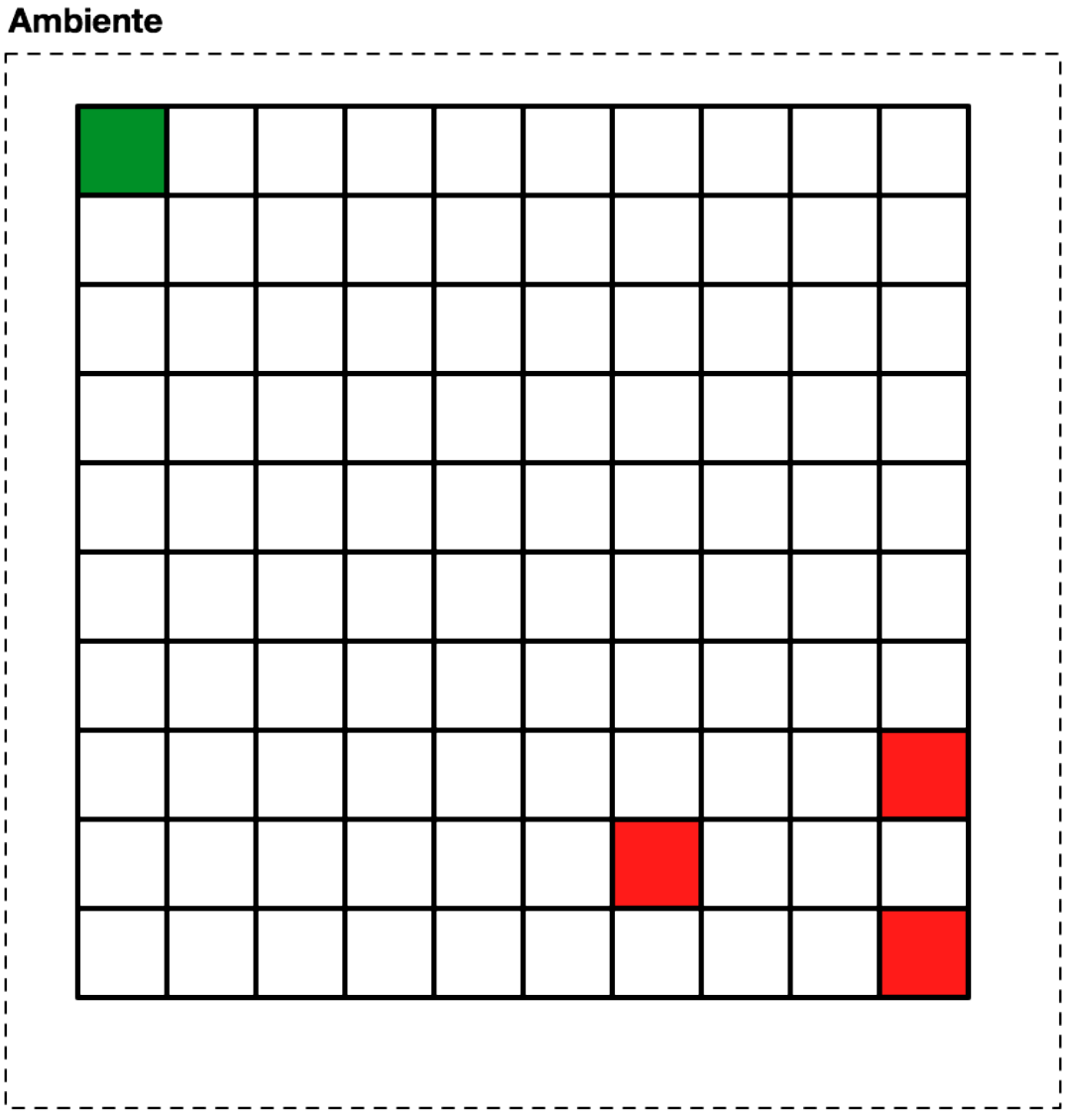

94 24 70
80 20 60
74 24 50
74 14 60
60 10 50
54 14 40
60 10 50
40 10 30
82 72 10
68 68 00
82 72 10
74 14 60
60 10 50
54 14 40
74 54 20
68 58 10
88 68 20
94 24 70
80 20 60
74 24 50
74 34 40
74 44 30
74 54 20
102 72 30

Representação do Sistema



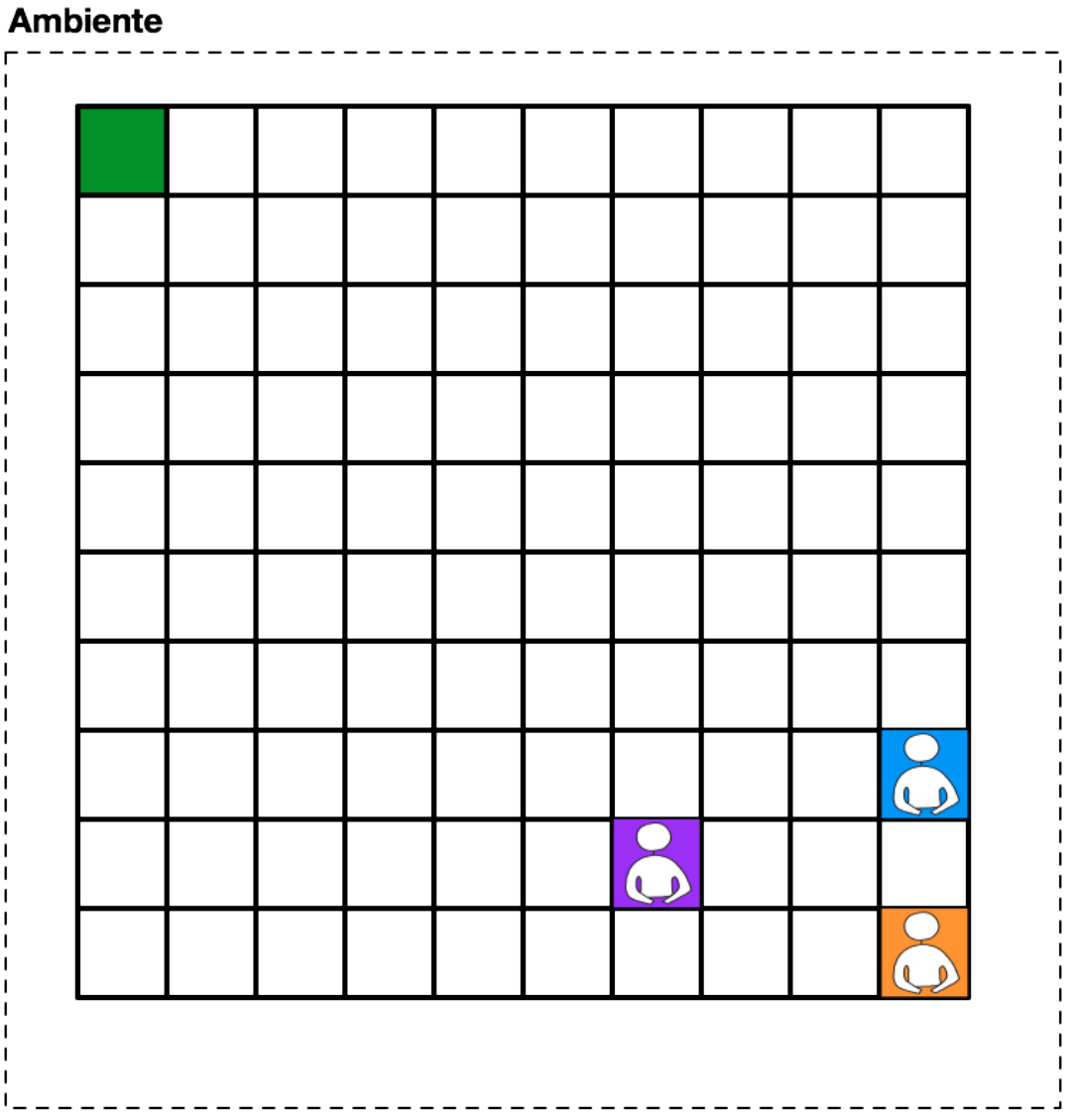
PURO

Representação do Sistema



PURO

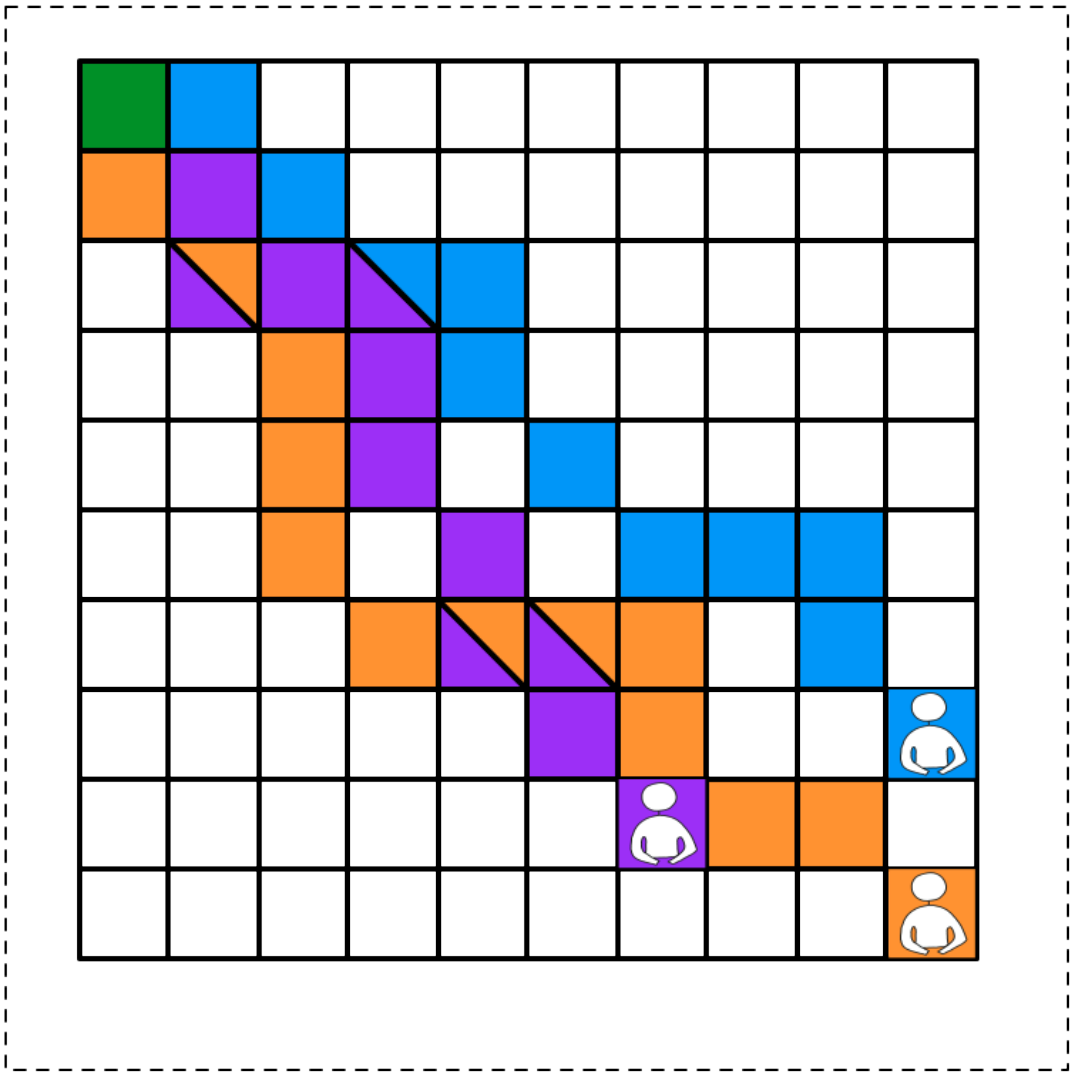
Representação do Sistema



PURO

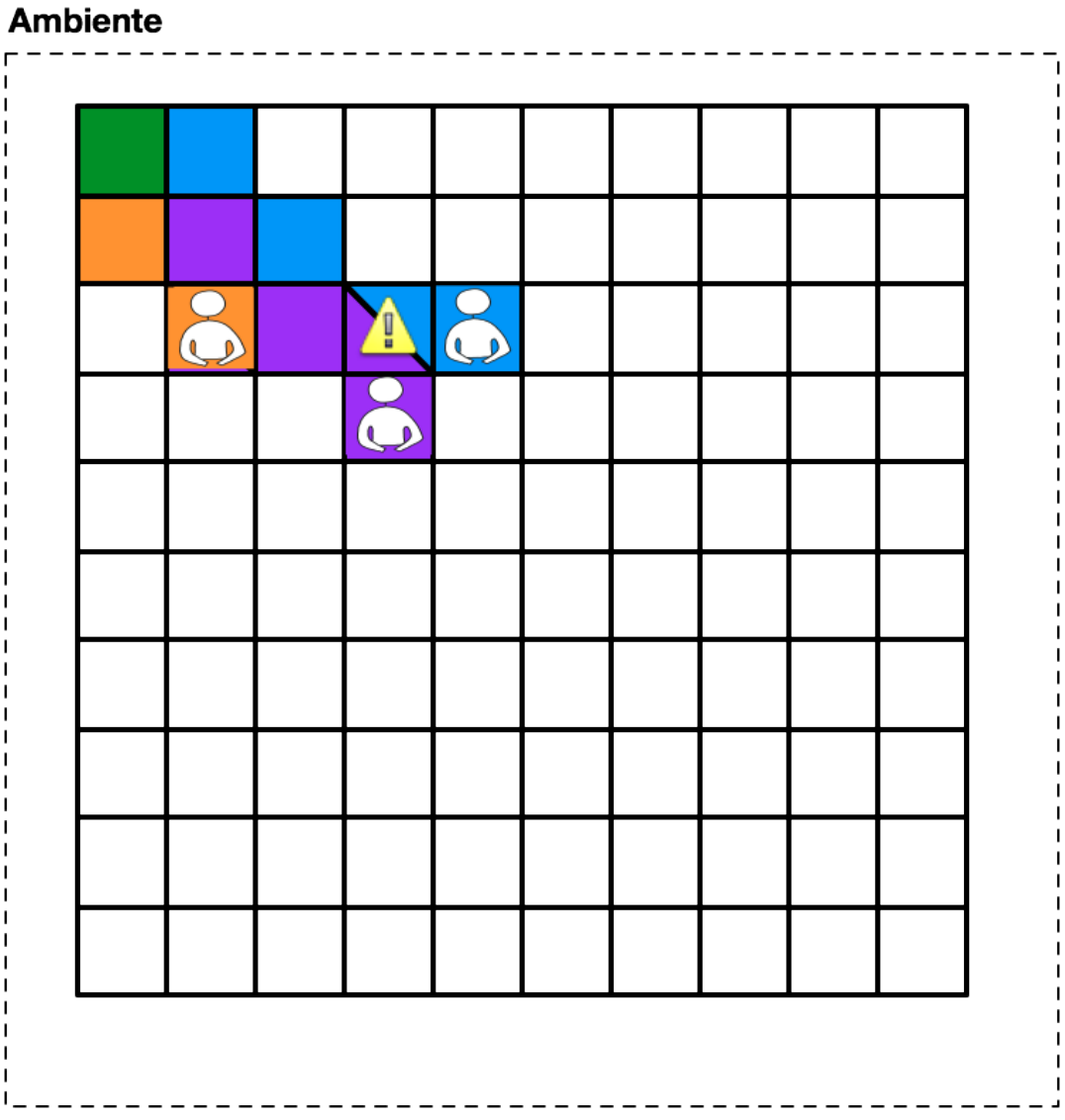
Representação do Sistema

Ambiente



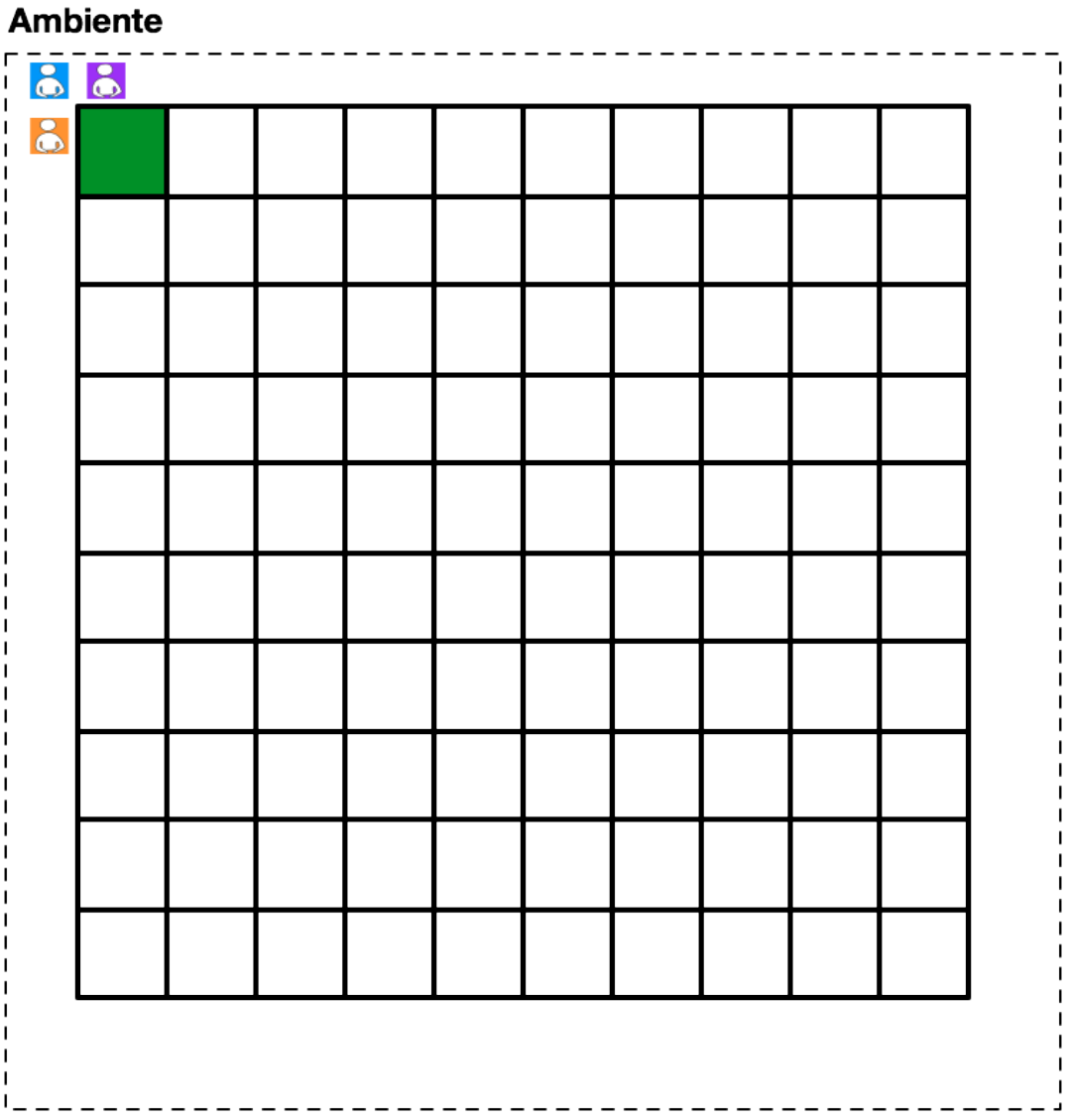
PURO

Representação do Sistema



PURO

Representação do Sistema



PURO

Mapeando de JADE para CUDA

JADE

- Agentes:
- Inteligência:
- Cenário:

CUDA

- Agentes:
- Inteligência:
- Cenário:

Mapeando de JADE para CUDA

JADE

- Agentes:
  - Criados a partir de uma classe que herda atributos da classe `Agent` e alocados em um `AgentContainer`.
- Inteligência:
- Cenário:

CUDA

- Agentes:
  - 2 vetores unidimensionais, onde o índice dos vetores representam o agente
- Inteligência:
- Cenário:

Mapeando de JADE para CUDA

JADE

- Agentes:
  - Criados a partir de uma classe que herda atributos da classe `Agent` e alocados em um `AgentContainer`.
- Inteligência:
  - Classe privada que herda atributos da classe `Behaviour`, e contem métodos `action()` e `done()`
- Cenário:

CUDA

- Agentes:
  - 2 vetores unidimensionais, onde o índice dos vetores representam o agente
- Inteligência:
  - Criada diretamente no *kernel*
- Cenário:

Mapeando de JADE para CUDA

JADE

- Agentes:
  - Criados a partir de uma classe que herda atributos da classe `Agent` e alocados em um `AgentContainer`.
- Inteligência:
  - Classe privada que herda atributos da classe `Behaviour`, e contem métodos `action()` e `done()`
- Cenário:
  - Classe `Scenario`

CUDA

- Agentes:
  - 2 vetores unidimensionais, onde o índice dos vetores representam o agente
- Inteligência:
  - Criada diretamente no *kernel*
- Cenário:
  - Matriz $m \times n$ representada em um vetor unidimensional de tamanho $m \times n$

Mapeando JADE para CUDA

JADE

- Vantagens:
  - Ferramental provido pelo *framework* e pela linguagem JAVA
  - Alocação dinâmica
  - Alto nível Abstração
- Desvantagens:
  - Criação dos agentes

CUDA

- Vantagens:
  - Criação de agentes na chamada do *kernel*
- Desvantagens:
  - Grande uso de estruturas de dados
  - Alocação Estática
  - Alocação na GPU(`cudaMalloc()` e `cudaMemCpy()`)
  - Ineficiência do A*

PURO

Análise de Performance

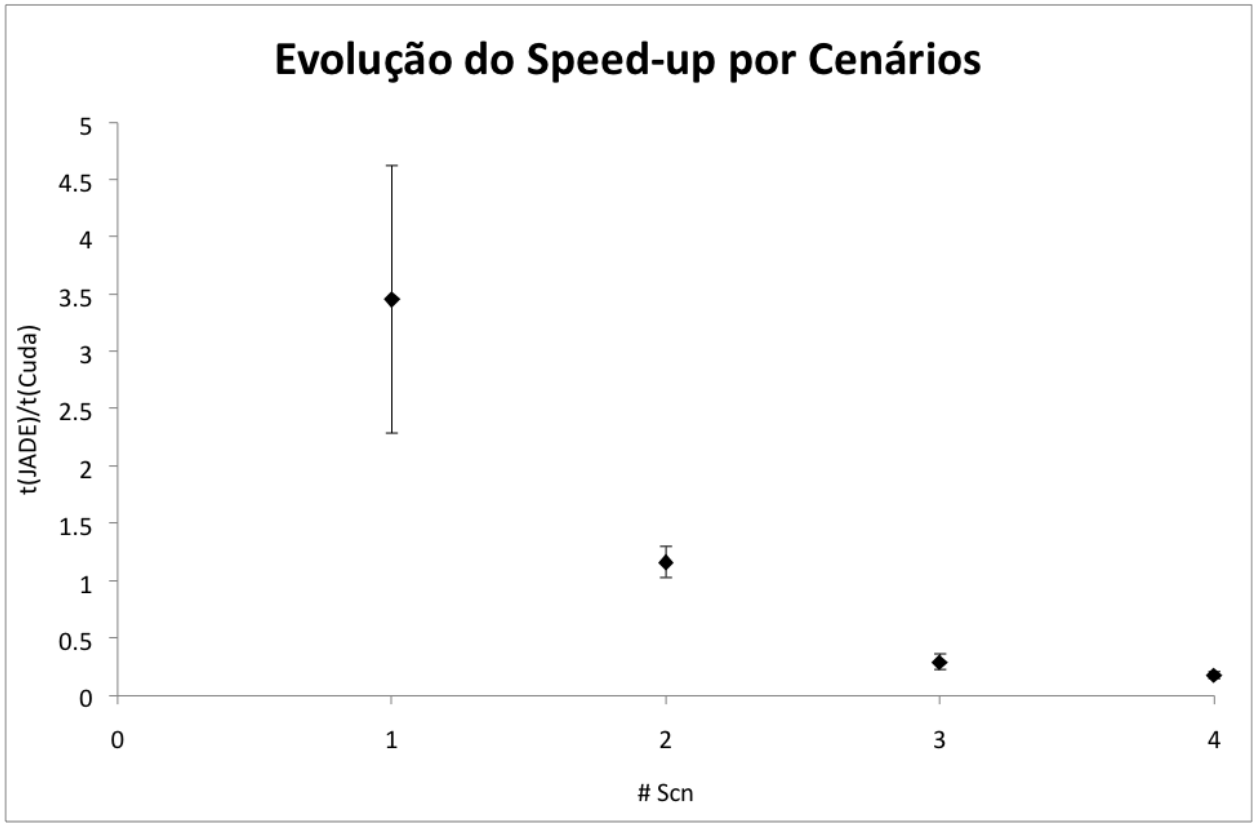
Máquina utilizada:

- CPU Intel Core 2 Duo 2.26GHz
- 4GB de RAM DDR3
- GPU NVidia Geforce 9400M
- 256MB DDR3 para a GPU

Análise de Performance: Casos de Teste

Scn	# Agentes	# Mapa
1	3	6x6
2	5	10x10
3	10	20x20
4	43	40x40

Análise de Performance: Gráficos



Sumário Principal

PURO

Conclusões e Trabalhos Futuros

- Esse estudo mostrou vantagens, desvantagens, dificuldades e similaridades ao mapear o mesmo problema de um paradigma AOP para o paradigma GPGPU

Conclusões e Trabalhos Futuros

- Esse estudo mostrou vantagens, desvantagens, dificuldades e similaridades ao mapear o mesmo problema de um paradigma AOP para o paradigma GPGPU
- Foi observada uma dificuldade no JADE ao trabalhar com muitos agentes
- Assim como há uma dificuldade no CUDA ao trabalhar com algoritmos complexos

Conclusões e Trabalhos Futuros

- Esse estudo mostrou vantagens, desvantagens, dificuldades e similaridades ao mapear o mesmo problema de um paradigma AOP para o paradigma GPGPU
- Foi observada uma dificuldade no JADE ao trabalhar com muitos agentes
- Assim como há uma dificuldade no CUDA ao trabalhar com algoritmos complexos
- Faltou avaliar:

- Trabalhos Futuros:

PURO

Conclusões e Trabalhos Futuros

- Esse estudo mostrou vantagens, desvantagens, dificuldades e similaridades ao mapear o mesmo problema de um paradigma AOP para o paradigma GPGPU
- Foi observada uma dificuldade no JADE ao trabalhar com muitos agentes
- Assim como há uma dificuldade no CUDA ao trabalhar com algoritmos complexos
- Faltou avaliar:
  - Comunicação
  - Outros modelos de programação orientada a agente, como o BDI [Rao and Georgeff, 1995]
  - Agentes Heterogênos
- Trabalhos Futuros:

PURO

Conclusões e Trabalhos Futuros

- Esse estudo mostrou vantagens, desvantagens, dificuldades e similaridades ao mapear o mesmo problema de um paradigma AOP para o paradigma GPGPU
- Foi observada uma dificuldade no JADE ao trabalhar com muitos agentes
- Assim como há uma dificuldade no CUDA ao trabalhar com algoritmos complexos
- Faltou avaliar:
  - Comunicação
  - Outros modelos de programação orientada a agente, como o BDI [Rao and Georgeff, 1995]
  - Agentes Heterogênos
- Trabalhos Futuros:
  - Framework de programação de agentes utilizando CUDA
  - Sistemas de Emergência
  - Simulação de multidões

PURO

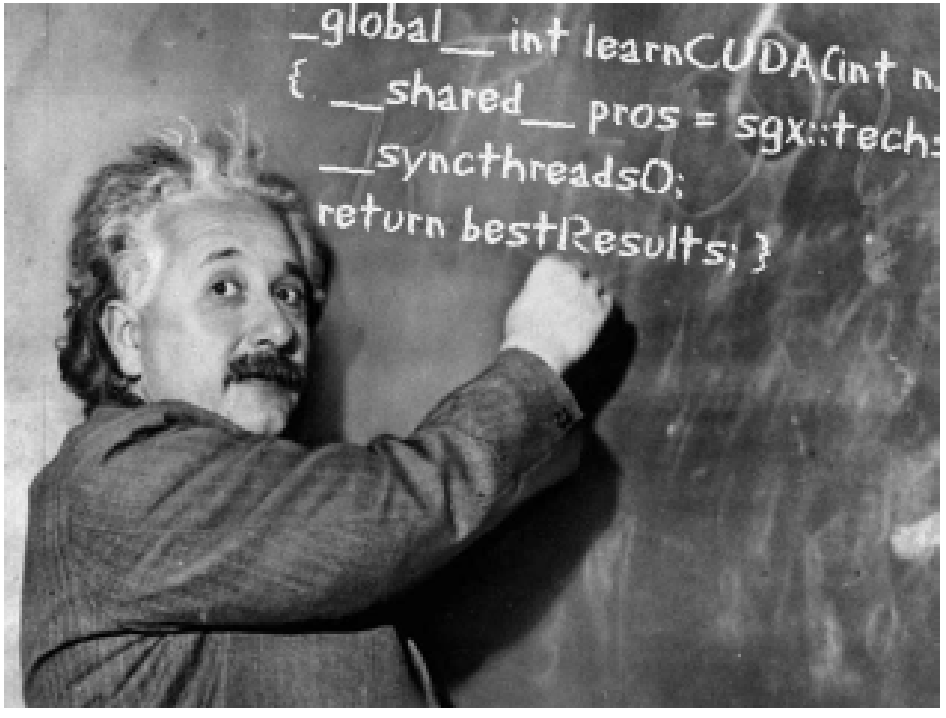
Considerações Finais e Agradecimentos

- Toda a tipografia da monografia e da apresentação foi feita utilizando LaTeX
- Esse trabalho teve um artigo aceito na conferência de Videojogos 2010 em Portugal
- Agradecer:
  - À Plateia Presente
  - Aos professores do Pólo Universitário de Rio das Ostras
  - À minha orientadora, Flávia Bernardini
  - Ao pessoal do MediaLab

PURO

Fim

Perguntas?



"O estado de espírito de uma pessoa se revela em seus sonhos. Devemos nos esforçar para fazer dos sonhos nossos aliados." (Yamamoto Tsunetomo - Hagakure, O Código Samurai)

PURO


Asimov, I. (1950).
I, robot.
Série dividida em 9 contos, publicados separadamente.


Bazzan, A. L. C. (2005).
A distributed approach for coordination of traffic signal agents.
In *Autonomous Agents and Multiagent Systems*, volume 10, pages 131–164. Springer Link.


Buchanan, B. G. (2005).
A (very) brief history of artificial intelligence.
In *American Association for Artificial Intelligence 25th Anniversary Issue*, pages 53–60.


Dimakis, N., Filippoupolitis, A., and Gelenbe, E. (2009).
Distributed building evacuation simulator for smart emergency management.
The Computer Journal.

PURO

Accepted to be published. Available at http://sa.ee.ic.ac.uk/publications/DBES_CJ.pdf.


Evans, T. (1968).
A program for the solution of a class of geometric–analogy inteligence-test questions.
In *Semantic Information about knowledge*, pages 271–353. MIT Press, Cambridge, Massachusetts.


Garcia, A. C. B. and Sichman, J. S. (2003).
Sistemas Inteligentes, Fundamentos e Aplicações, chapter Agentes e Sistemas Multiagentes.
Manole.


Gelertner, H. (1959).
Realization of a geometry-theorem proving machine.
In *Proceedings of an International Conference on Information Processing*, pages 273–282.


Hebb, D. O. (1949).

PURO

The Organization of Behavior.
Psychology Press; New edition edition (June 15, 2002).


Newell, A., Simon, H., and Simon, J. C. (1959).
Report on a general problem-solving program.
In *Proceedings of the International Conference on Information Processing*, pages 256–264.


Nilson, N. J. (1971).
Problem-solving methods in Artificial Inteligence.
McGraw-Hill.


NVIDIA (2007).
CUDA description.
`http://www.nvidia.com/cuda`.


Passos, E. B., Joselli, M., Zamith, M., Rocha, J., Clua, E. W. G., Montenegro, A., Conci, A., and Feijo, B. (2008).
Supermassive crowd simulation on gpu based on emergent behavior.

PURO

In *SBGames 2008*.


Pearl, J. (1984).
Heuristics: Intelligent Search Strategies for Computer Problem Solving.
Addison-Wesley.


Rao, A. S. and Georgeff, M. P. (1995).
Bdi agents: from theory to practice.
In *Proceedings of the 1st International Conference on Multi-Agent Systems*, page 312–319.


Russell, S. and Norvig, P. (1995).
Artificial Intelligence, A Modern Approach.
Prentice Hall, second edition edition.


Shortliffe, E. H. and Buchanan, B. G. (1984).
Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.
Addison-Wesley.

PURO


Slagle, J. (1963).
A heuristic program that solves symbolic integration problems in freshman calculus.
Jornal of the Association for Computing Machinery, 10(4).


Stone, P. and Veloso, M. (1997).
Multiagent systems: A survey from a machine learning perspective.
Autonomous Robotics, 8:345–383.


Turing, A. (1950).
Computing machinery and intelligence.
Journal of the Mind Association, LIX:433–460.


Wolldridge, M. (2002).
An Introduction to Multiagent Systems.
John Wiley and Sons.