Utilização de ambientes paralelos no processo de aprendizado de algoritmos de busca de caminho em tempo real

Vinicius Marques Terra
Orientador: Prof. Luiz Chaimowicz
Co-orientador: Prof. Renato Ferreira

Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

30 de Junho de 2010

Sumário

Contextualização

Estado atual dos videogames

- O avanço das tecnologias usadas nos jogos permite criação de ambientes cada vez mais realistas e imersivos;
- A evolução dos jogos está atrelada ao avanço dos *hardwares* nos videogames;
  - GPUs - Computação Gráfica;
  - Unidades de processamento de física - simulações físicas;
  - Processadores multinúcleo - interação com o jogador, etc.;
- Incorporação de processadores multinúcleo muda o paradigma de programação de jogos;
- Aplicar essa mudança à **Inteligência Artificial** de um jogo é uma maneira de se aproveitar tais recursos providos.

Contextualização

Definição do escopo

- Utilizar ambientes paralelos para otimizar algoritmos Inteligência Artificial:
- Algoritmos de busca de caminho em tempo real;
- Implementação da paralelização em arquitetura presente em console de última geração:
- Cell BroadBand Engine - Playstation 3;

Contextualização

Inteligência Artificial

Entidades não controladas pelo jogador são chamadas NPCs (*non-player characters*);

- A movimentação de NPCs faz parte do projeto da I.A. de um jogo;
- Neste contexo, temos os problema da busca de caminho;
- Encontrar o caminho mais eficiente, minimizando custos (tempo, distância, combustível, dinheiro etc.);

Contextualização

Busca de caminho

Como a forma de representação de ambientes mais comum em jogos é a discretização em grafos, esse problema se resolve com algoritmos de busca em grafos.

- Para a maioria dos jogos, são utilizados algoritmos de busca de caminho como o **A*** (Hart et al., 1968);
- Nestes algoritmos, o caminho completo é computado antes que o NPC execute sua primeira ação;
- Consequentemente, o tempo de planejamento aumenta juntamente com o tamanho do problema;
- Utilizados em ambientes previamente conhecidos e controlados;

Busca por uma solução completa até o estado objetivo	Executa

Contextualização

Avanço dos jogos X busca de caminho

Por um outro lado, o avanço dos jogos faz com que os ambientes se tornem cada vez maiores e mais complexos, com uma quantidade cada vez maior de NPCs atuando nestes ambientes:

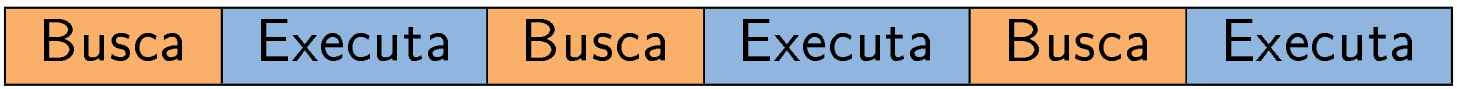
- mapas de propoções "continentais";
- milhares de NPCs atuando simultaneamente nesses ambientes;
- ambientes dinâmicos;

Neste caso, a utilização de algoritmos de busca de caminho pode prejudicar o desempenho do jogo, pois quebra o compromisso com respostas em tempo real;

Contextualização

Busca de caminho em tempo real

- A solução é a utilização de algoritmos de busca de caminho em tempo real;
- Intercalam planejamento (delimitado por um limite de tempo) e execução;
- Este limite de tempo independe do tamanho dos mapas e das buscas a serem realizadas;
- Mantém-se então o compromisso com respostas em tempo real;



Contextualização

Busca de caminho em tempo real

- Busca de caminho em tempo real = busca de caminho + um limite de ações?

Contextualização

Busca de caminho em tempo real

- Busca de caminho em tempo real = busca de caminho + um limite de ações?
- **NÃO.** Somente impor um limite constante acarreta perda de completude;
- Deve existir também um componente de **aprendizado**;
- Evita mínimos locais e reconhece modificações no ambiente;

Processo de convergência:

"O mecanismo de aprendizado garante que a realização de uma mesma busca sucessivamente resulte no caminho de custo mínimo." (R. E. Korf)

Objetivos

O problema

O compromisso com respostas em tempo real destes algoritmos faz com que:

- O limite de tempo disponível seja pequeno;
- A região do mapa explorada por vez seja pequena;
- As buscas apresentem maior sub-otimalidade, retardando o processo de convergência.

Objetivos

Solução proposta

Como acelerar o processo de convergência, mantendo o compromisso com tempo real?

Objetivos

Solução proposta

Como acelerar o processo de convergência, mantendo o compromisso com tempo real?

- Utilizando ambientes **paralelos**;
- Uma busca, denominada principal, mantém a restrição de tempo real;
- Outras buscas, denominadas auxiliares, são realizadas sem tal restrição;
- Todas as buscas compartilham o mesmo aprendizado adquirido;

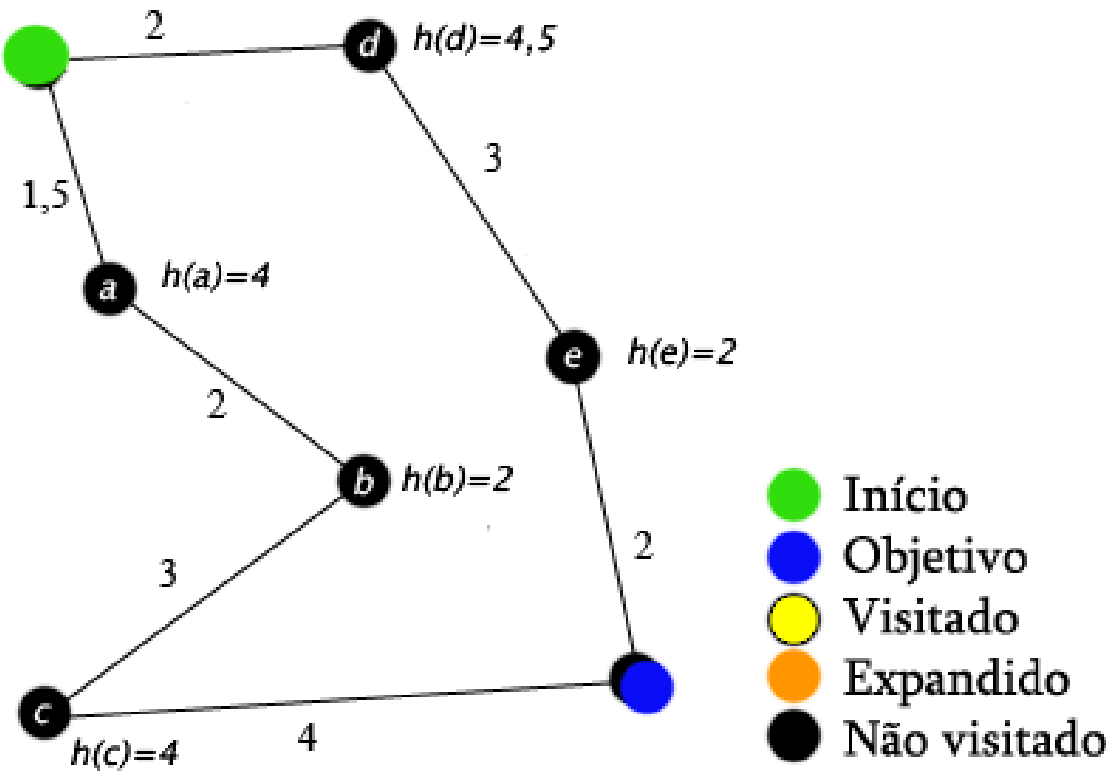
Algoritmos de Busca de Caminho clássicos

A*

- **A*** (Hart et al., 1968): algoritmo amplamente utilizado para busca de caminho;
- Algoritmo bastante flexível;
- Apresenta melhor performance que *DFS* e *BFS*;
- Propriedades:
  - Busca é por **heurística**: *best-first search*;
  - Uma heurística **admissível** garante que a solução de custo ótimo seja encontrada;
  - Estados expandidos de acordo com função $f(x) = g(x) + h(x)$;

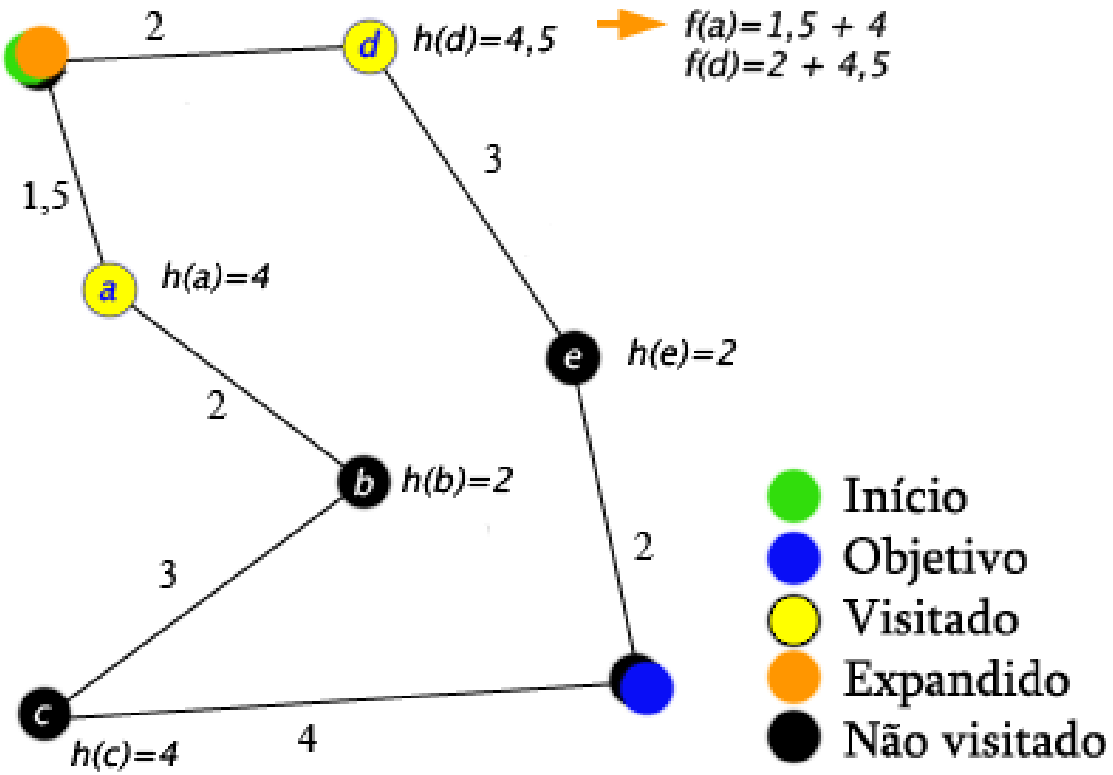
Algoritmos de Busca de Caminho clássicos

A*



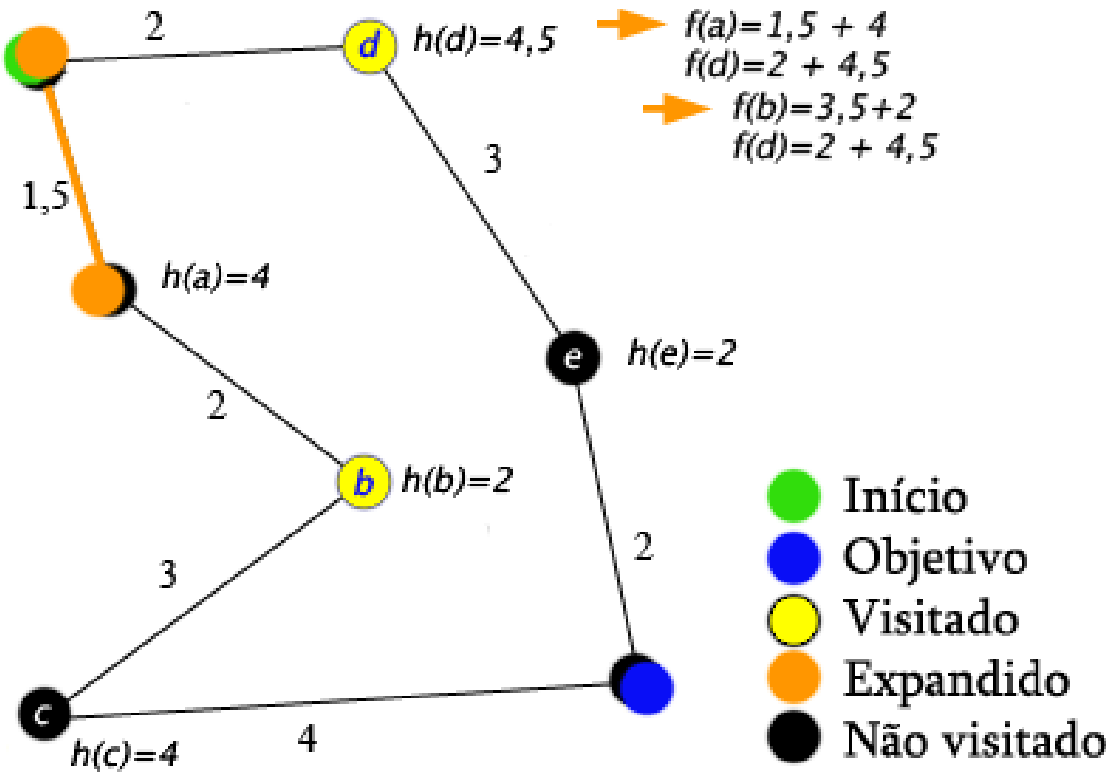
Algoritmos de Busca de Caminho clássicos

A*



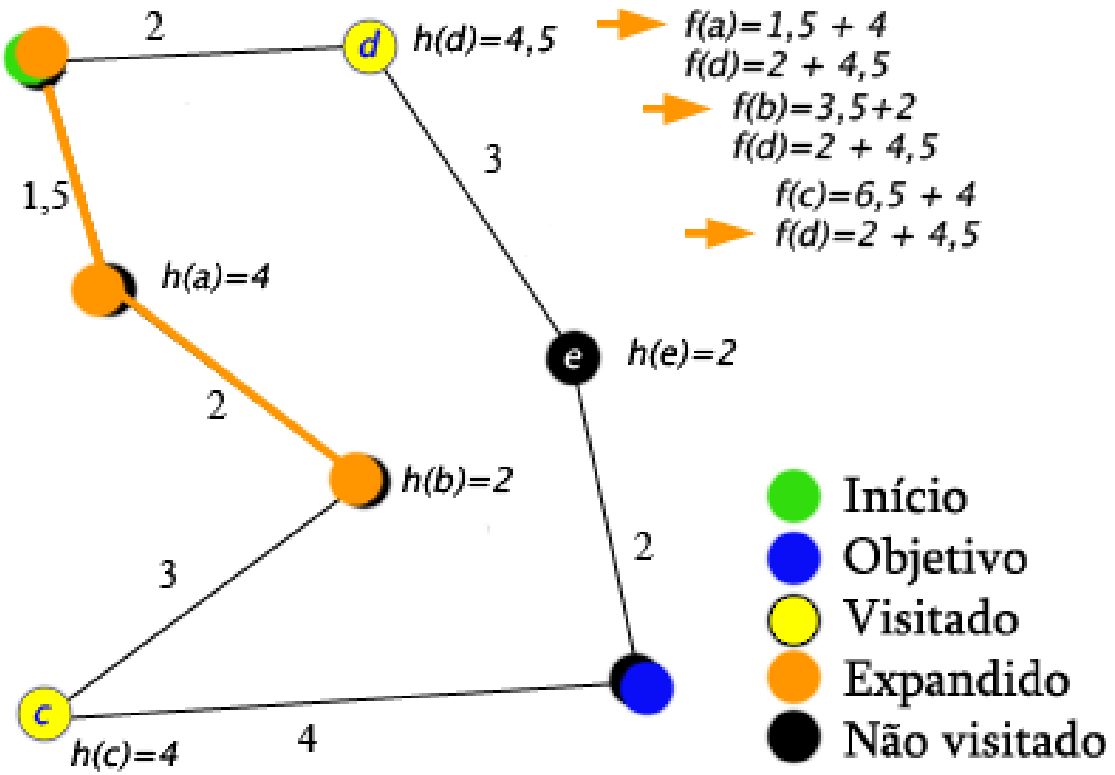
Algoritmos de Busca de Caminho clássicos

A*



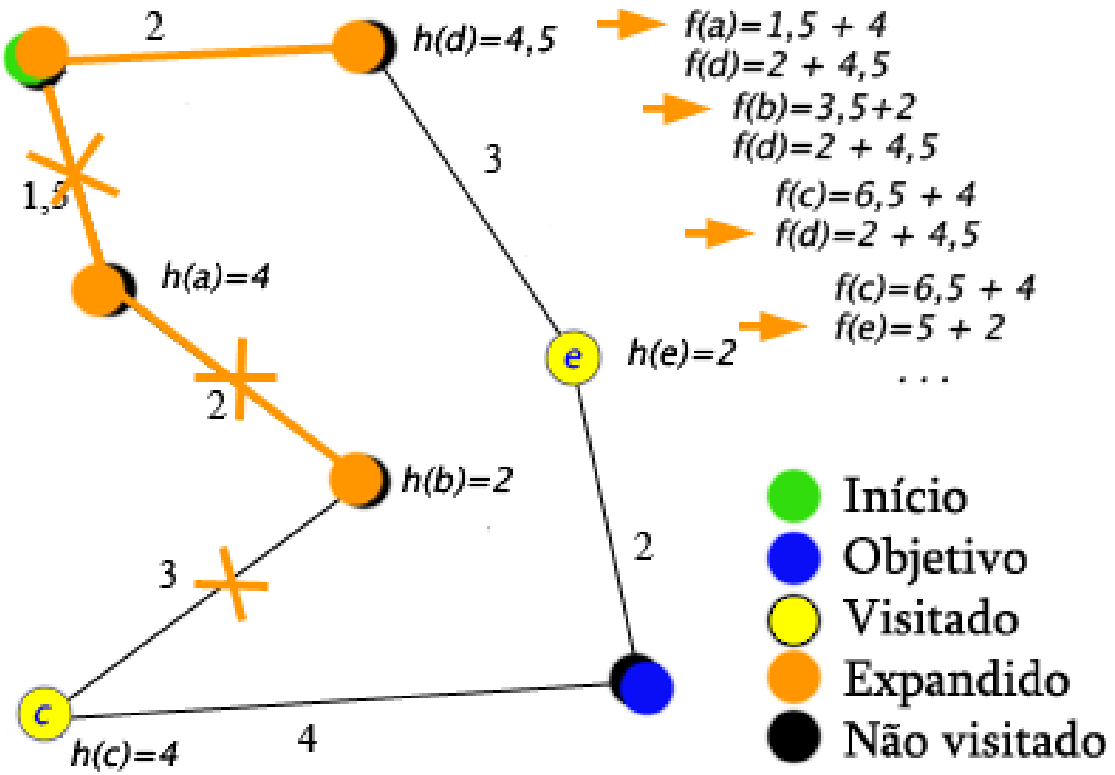
Algoritmos de Busca de Caminho clássicos

A*



Algoritmos de Busca de Caminho clássicos

A*



Algoritmos de busca de caminho em tempo real

Propriedades básicas

- Espaço de busca local;
- Espaço de aprendizado local;
- Regra de aprendizado;
- Solução inicial não é ótima: converge para otimalidade após repetidas execuções (processo de convergência);

Algoritmos de busca de caminho em tempo real

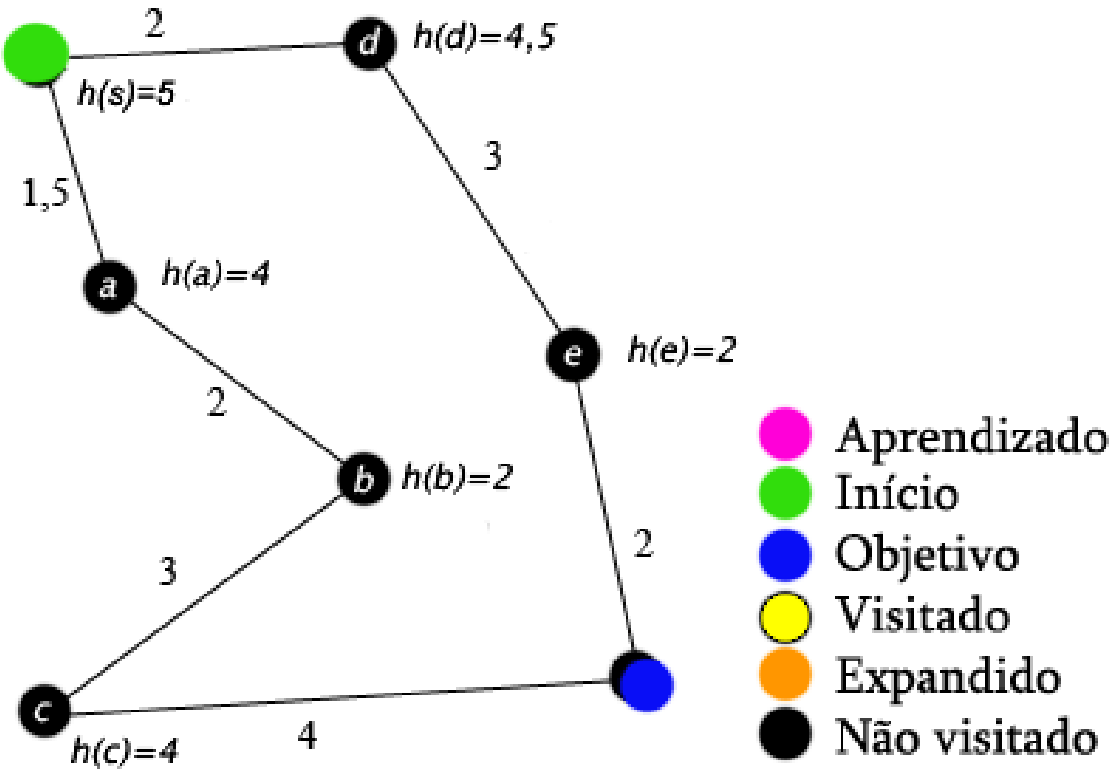
LSS-LRTA*

- **Local Search Space LRTA***(Koenig & Sun, 2009): variação do LRTA*(Korf, 1990) com maiores espaços de busca e aprendizado;
- Utiliza o A* para determinar tais espaços;
- A profundidade da busca(*lookahead*) é um parâmetro $>= \mathbf{1}$;
- Utiliza uma busca Dijkstra(1959) para atualizar os estados no espaço de aprendizado local;
- Possui menor tempo de convergência;

Algoritmos de busca de caminho em tempo real

LSS-LRTA*

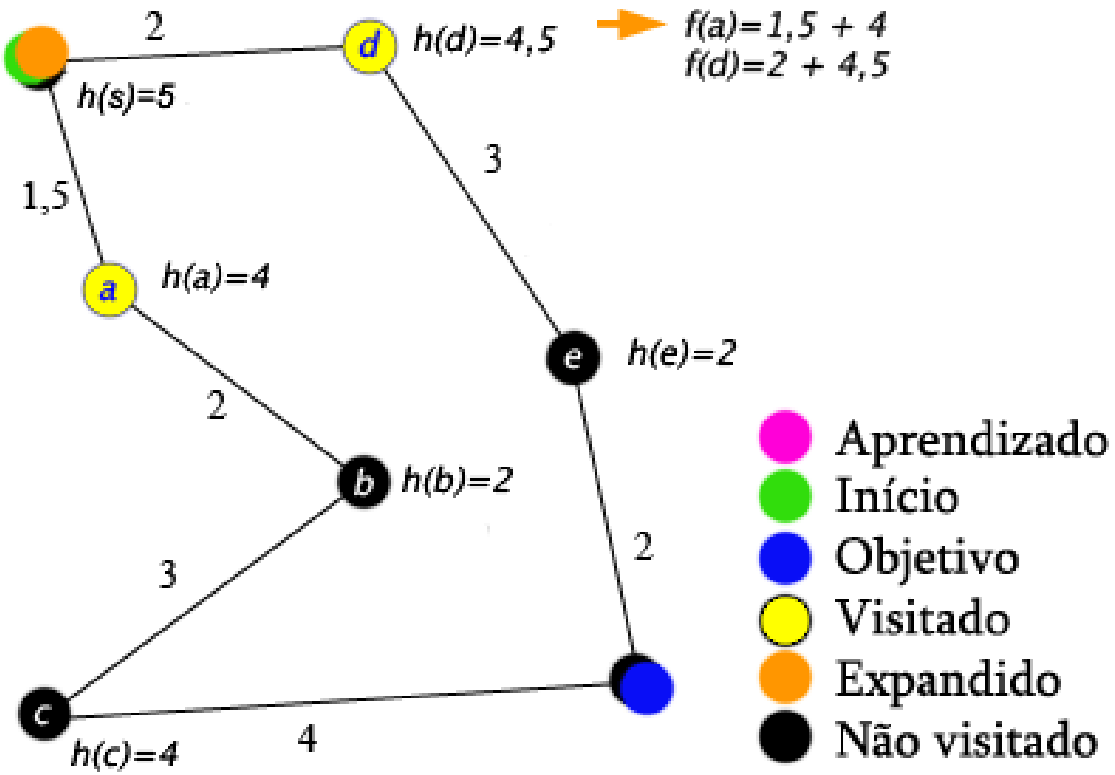
Algoritmo com *lookahead* 2:



Algoritmos de busca de caminho em tempo real

LSS-LRTA*

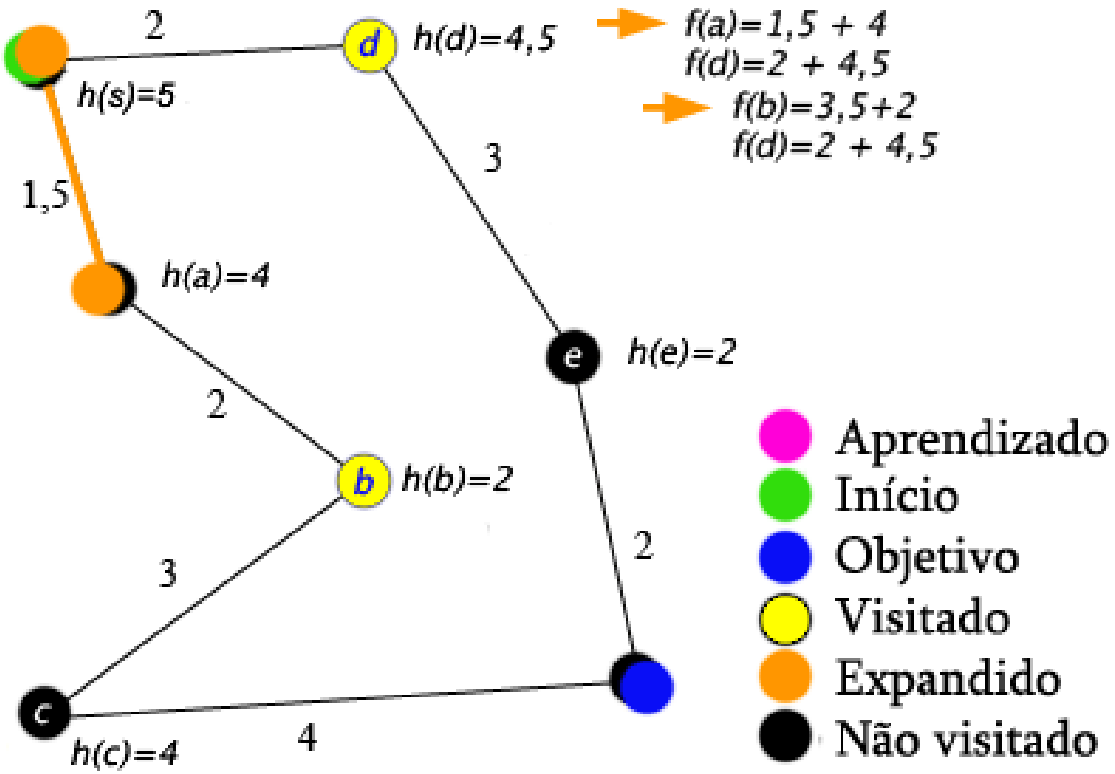
Algoritmo com *lookahead* 2:



Algoritmos de busca de caminho em tempo real

LSS-LRTA*

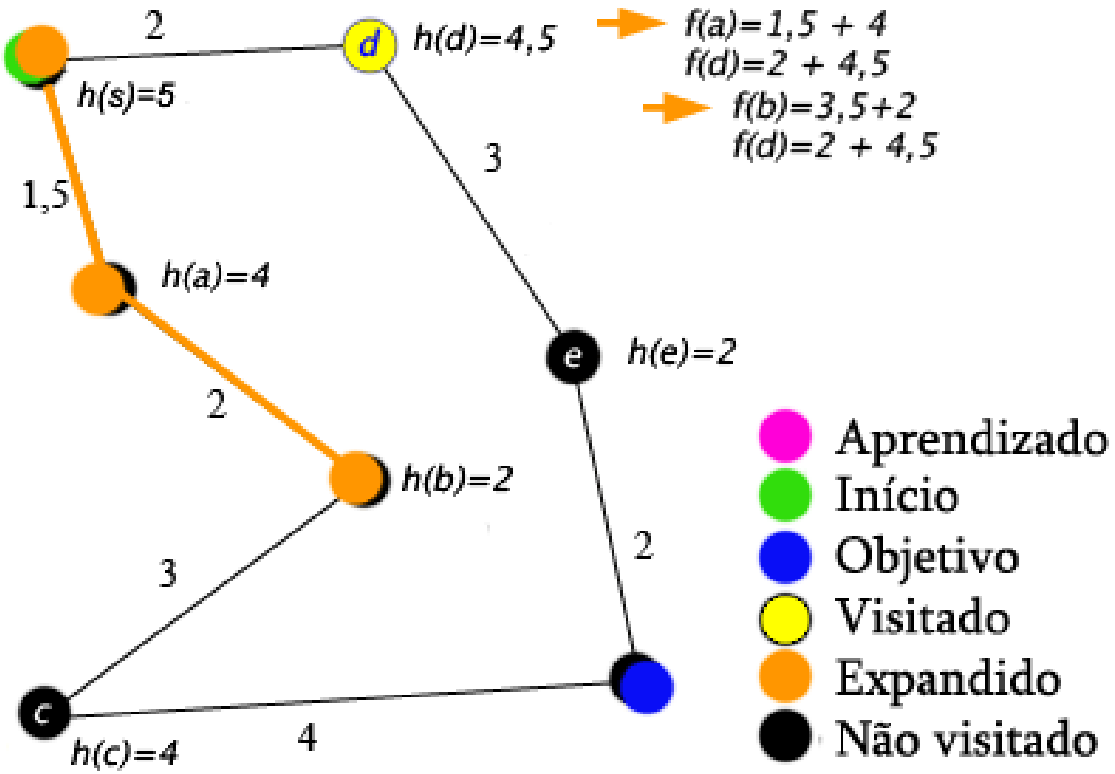
Algoritmo com *lookahead* 2:



Algoritmos de busca de caminho em tempo real

LSS-LRTA*

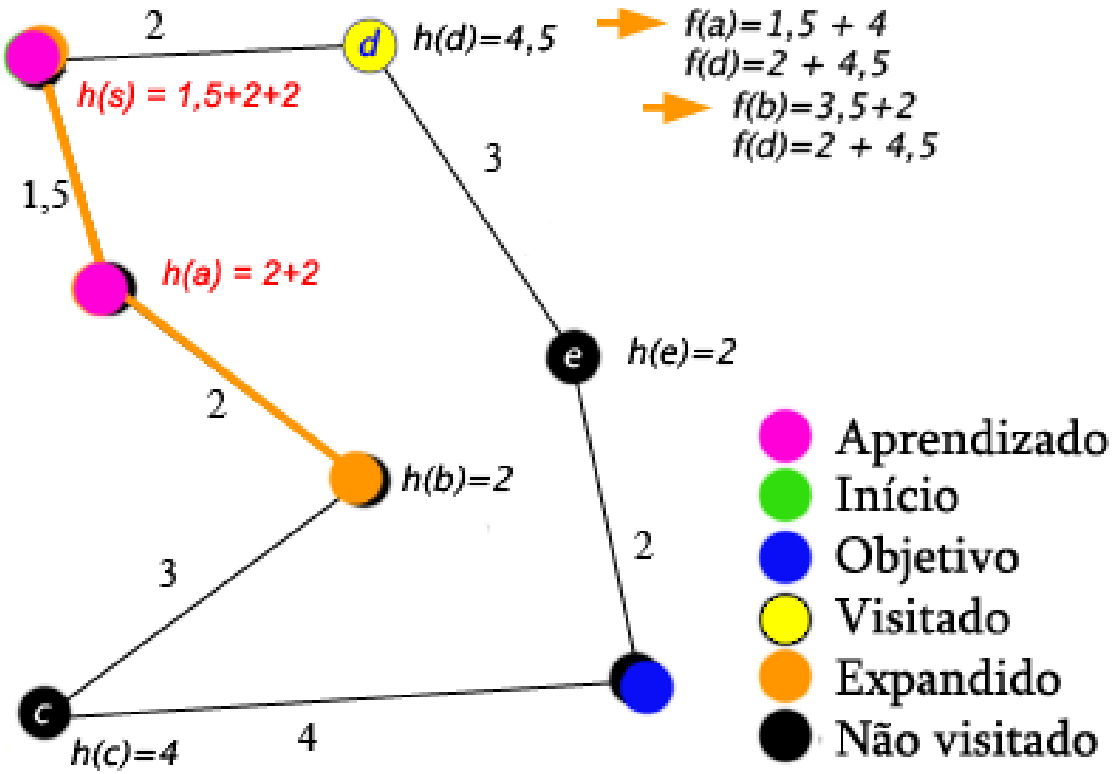
Algoritmo com *lookahead* 2:



Algoritmos de busca de caminho em tempo real

LSS-LRTA*

Algoritmo com *lookahead* 2:



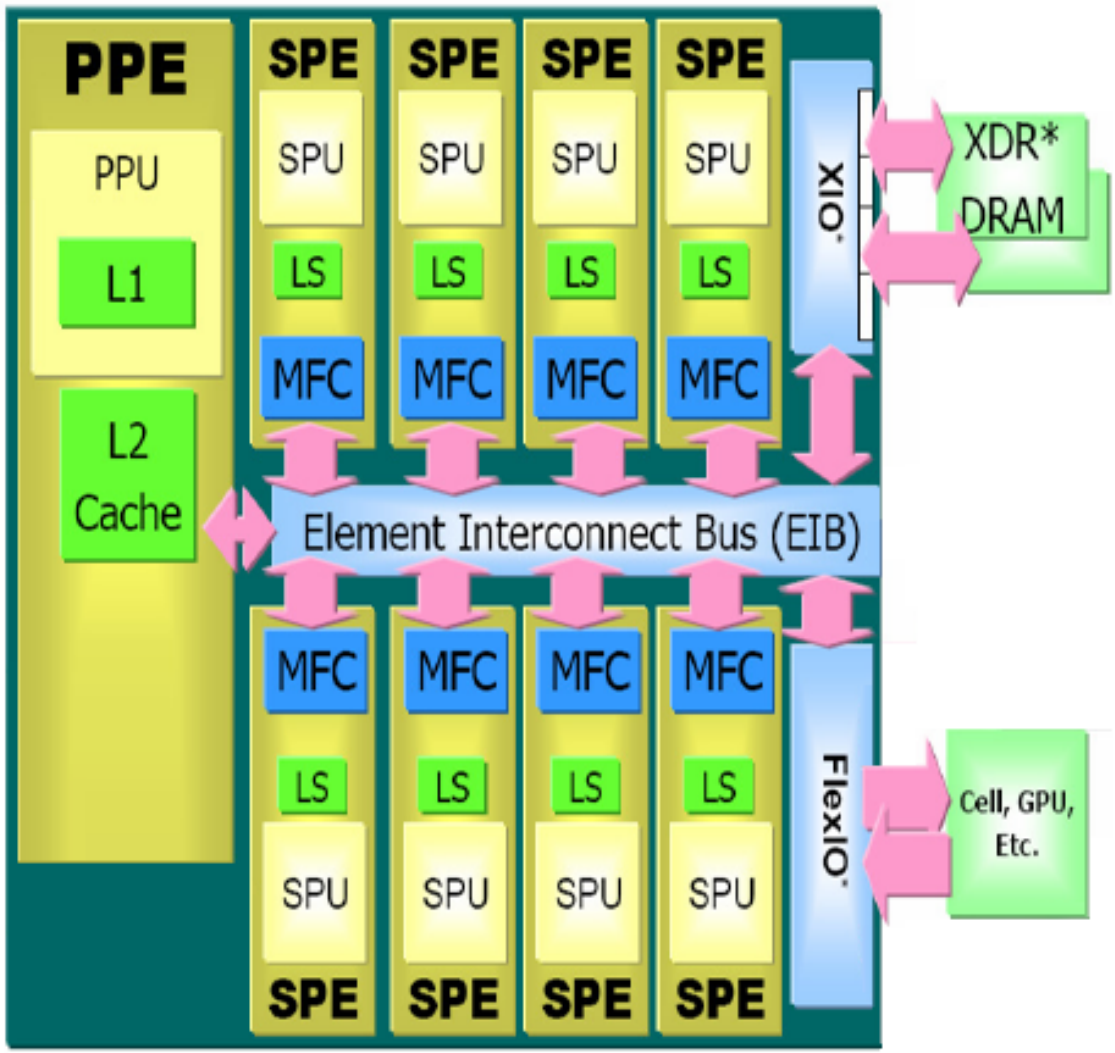
A Arquitetura Cell Broadband Engine

Características

- **Cell BroadBand Engine** foi projetado para ser uma ponte entre os processadores de propósito geral e os processadores específicos;
- Composto por nove processadores operando em uma memória compartilhada e coerente:
  - Um **Power Processor Element**(PPE): processador *multithreaded* da família Power Architecture™ de 64-bits;
  - Oito **Synergistic Processing Elements**(SPEs) independentes: processador RISC com organização SIMD 128-bits de alta performance, com 256K de LS;
- Operações de DMA realizadas através de um barramento chamado **Element Interconnect Bus**;
- Comunicação feita também através de **Mailboxes**: registradores de mensagens de 32-bits;

A Arquitetura Cell Broadband Engine

Diagrama da arquitetura



A Arquitetura Cell Broadband Engine

Considerações

- Memória limitada dos SPEs: código-fonte, estruturas de dados e pilha de execução;
- Ausência de cache: latência das transferências de dados - *double buffer*;

Visão geral

- Manter restrição de tempo real em uma busca principal, no PPE;
- Efetuar trechos complementares a esta busca em paralelo nos SPEs;
- Todas as buscas compartilham o mesmo aprendizado adquirido;

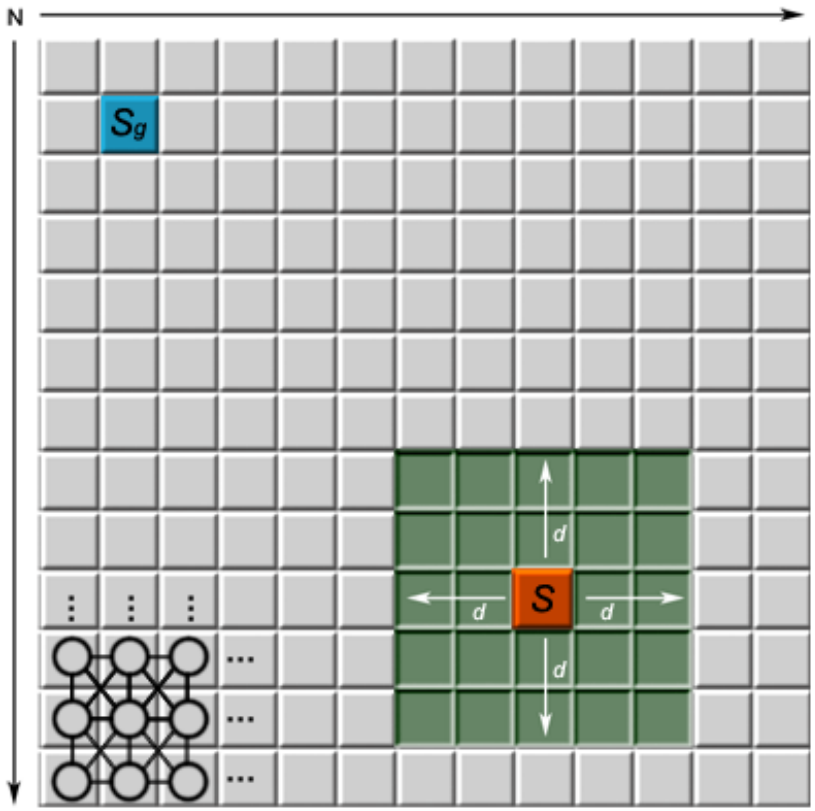
Fluxo de execução no PPE

Visão Geral

- Realiza um trecho da busca em tempo real principal;
- Cria tarefas correspondentes aos trechos de buscas auxiliares;
- Sincroniza com SPEs para distribuir tarefas;

Fluxo de execução no PPE

Estrutura de uma tarefa



Fluxo de execução no PPE

Criação de tarefas

Lookahead das buscas auxiliares (d) é múltiplo do *lookahead* da principal;

Fluxo de execução no PPE

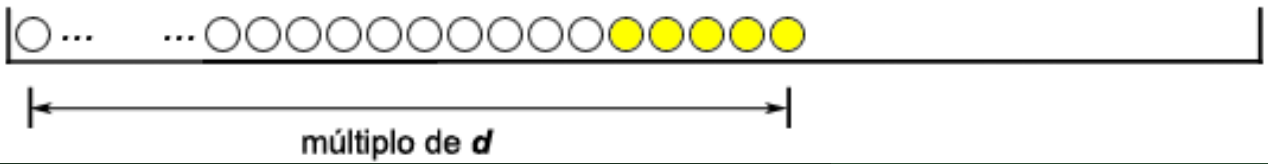
Criação de tarefas

Lookahead das buscas auxiliares (d) é múltiplo do *lookahead* da principal;

Fluxo de execução no PPE

Criação de tarefas

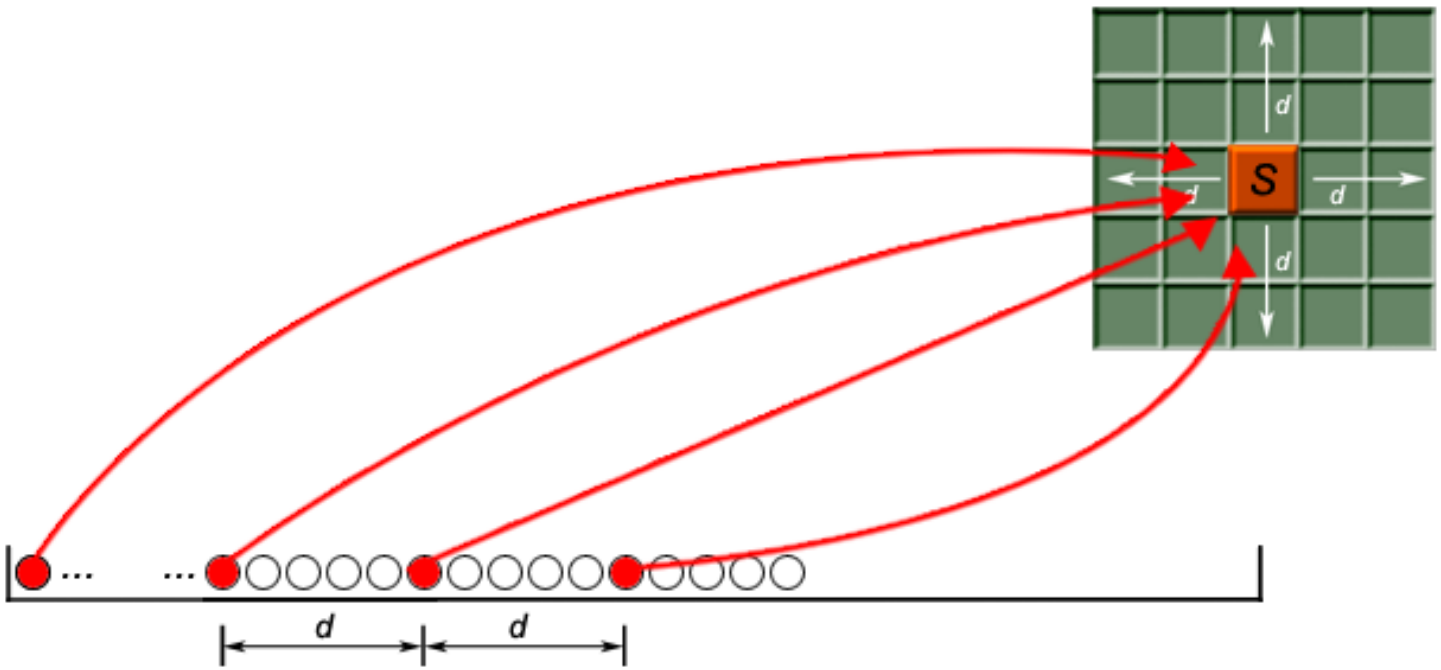
Lookahead das buscas auxiliares (*d*) é múltiplo do *lookahead* da principal;



Fluxo de execução no PPE

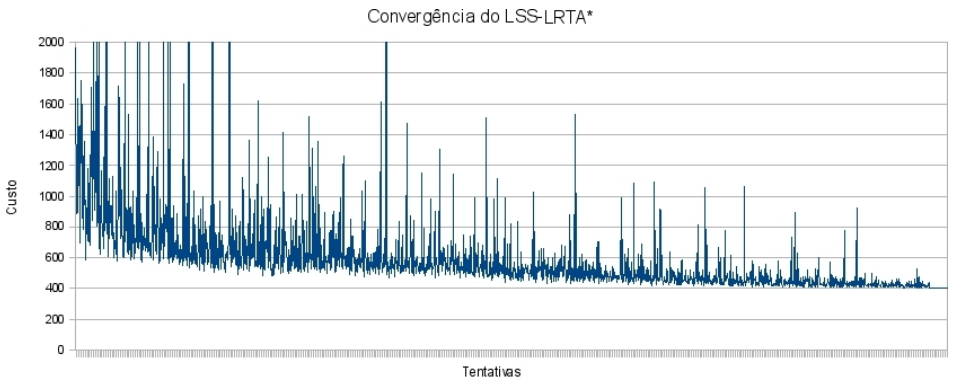
Criação de tarefas

Lookahead das buscas auxiliares (*d*) é múltiplo do *lookahead* da principal;



Fluxo de execução no PPE

Ordenação das tarefas



Objetivo da ordenação é premiar a regularidade do processo de convergência:

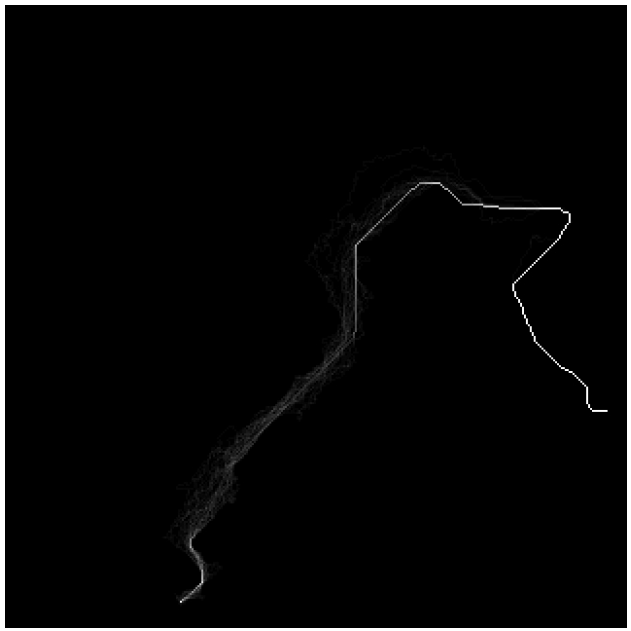
- Classificação de estados: baseado em colônias de formigas;
- Se o custo da última busca for **menor** que o custo armazenado, são premiados os estados deste caminho;
- Estados desta busca recebem mais "feromônios";
- O custo armazenado passa a ser o custo desta busca;

Fluxo de execução no PPE

Ordenação das tarefas

- Tarefas são inseridas em um *heap*, ordenadas pela quantidade de feromônios;

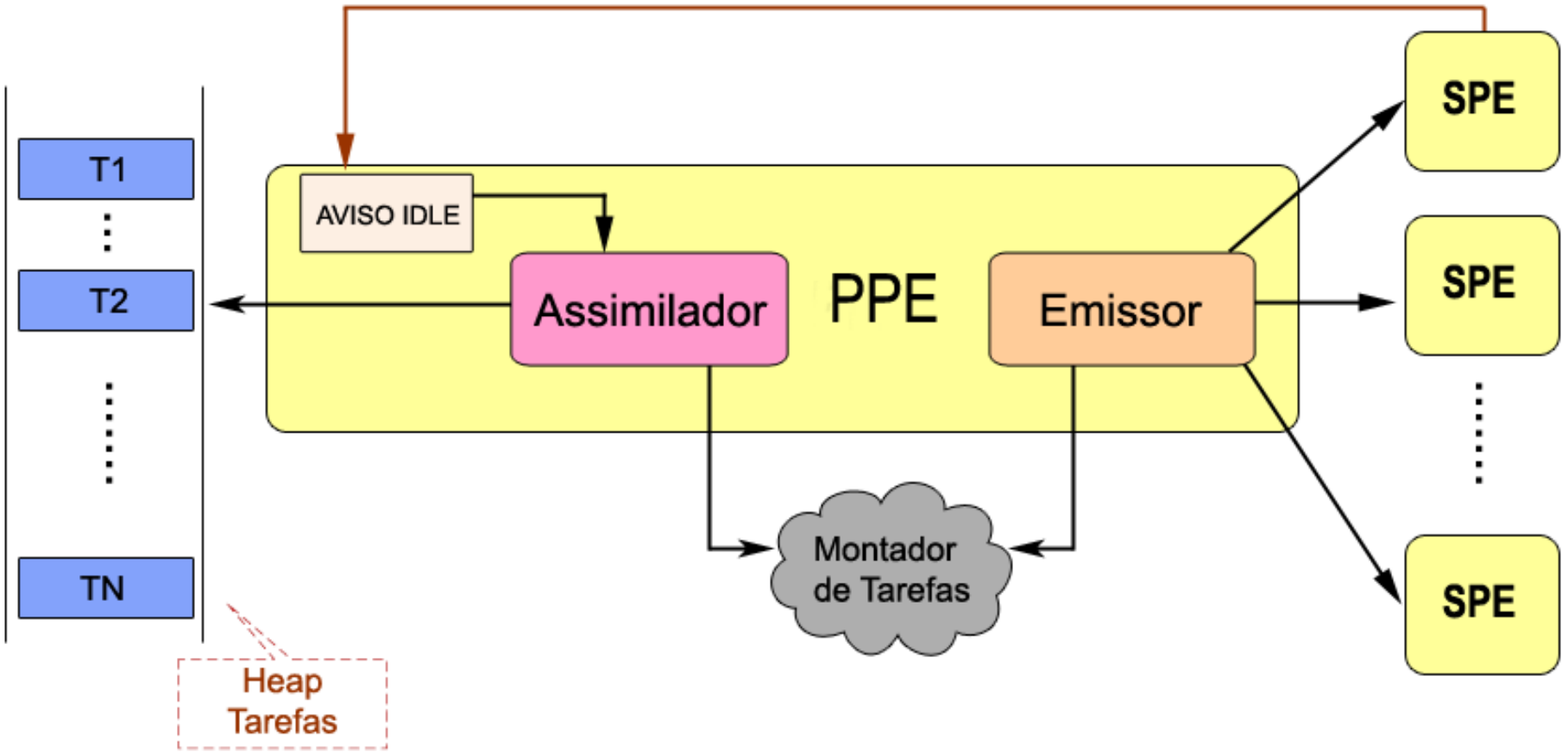




Fluxo de execução no PPE

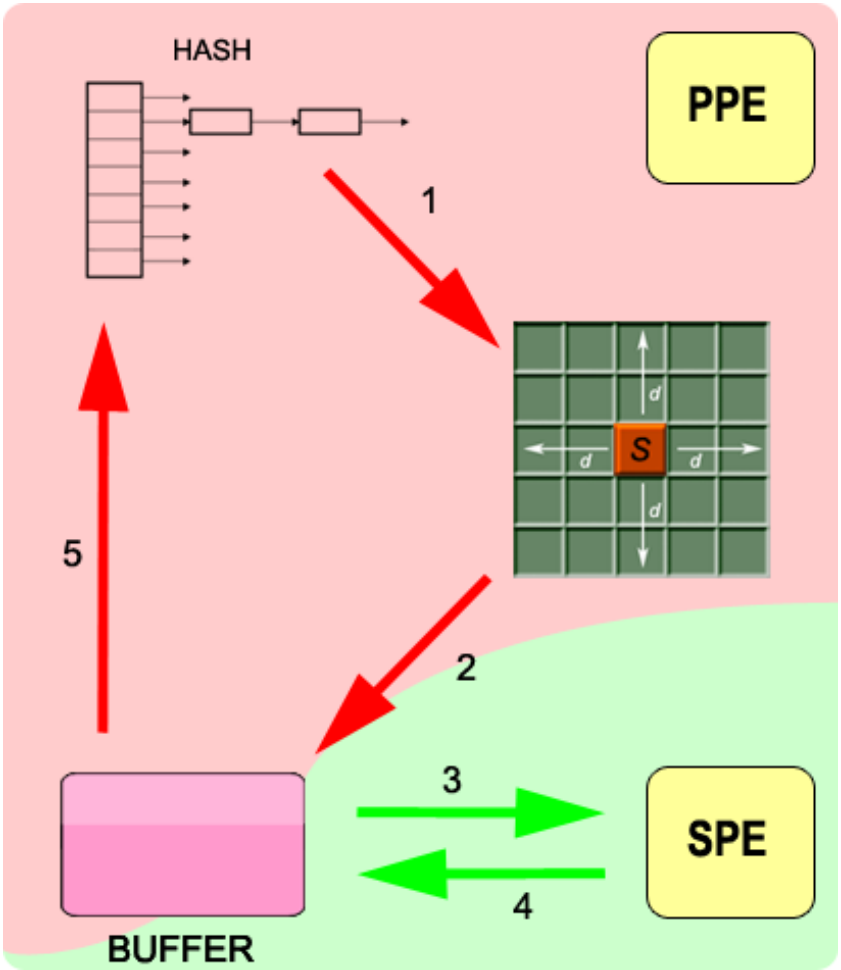
Sincronização com SPEs

- Troca de mensagens por *mailboxes*, tarefas enviadas por DMA;



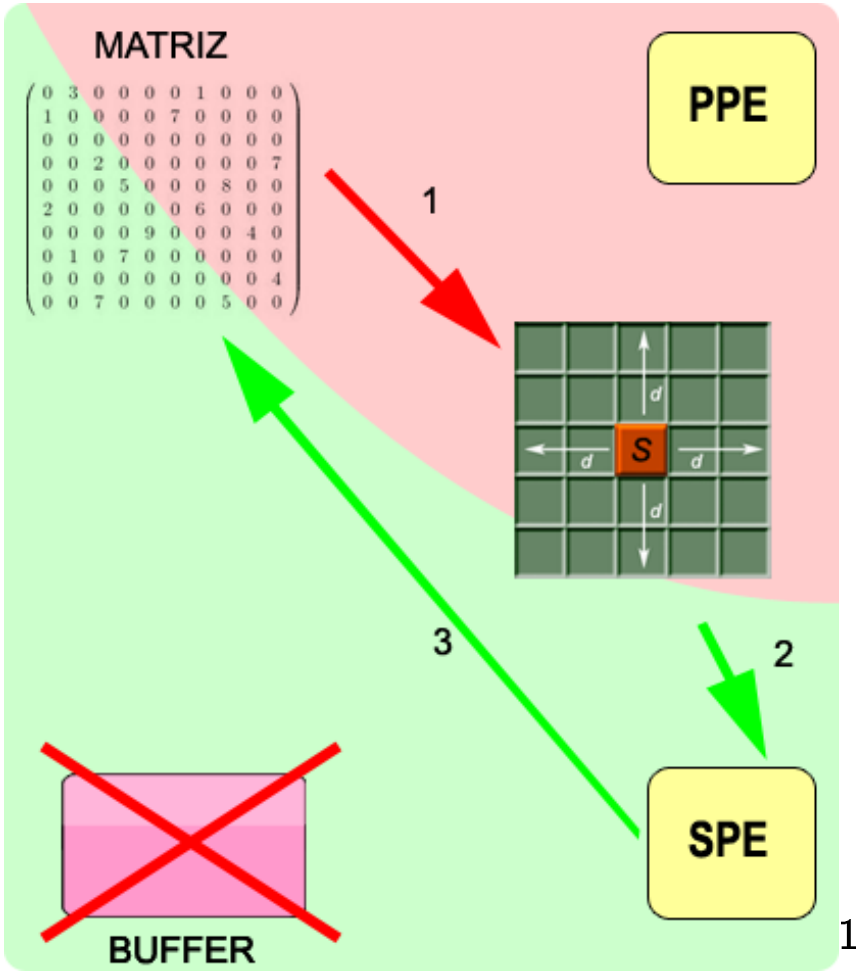
Montagem das tarefas

Primeira abordagem: compartilhamento da tabela Hash de heurísticas



Montagem das tarefas

Segunda abordagem: matriz esparsa no lugar da tabela Hash

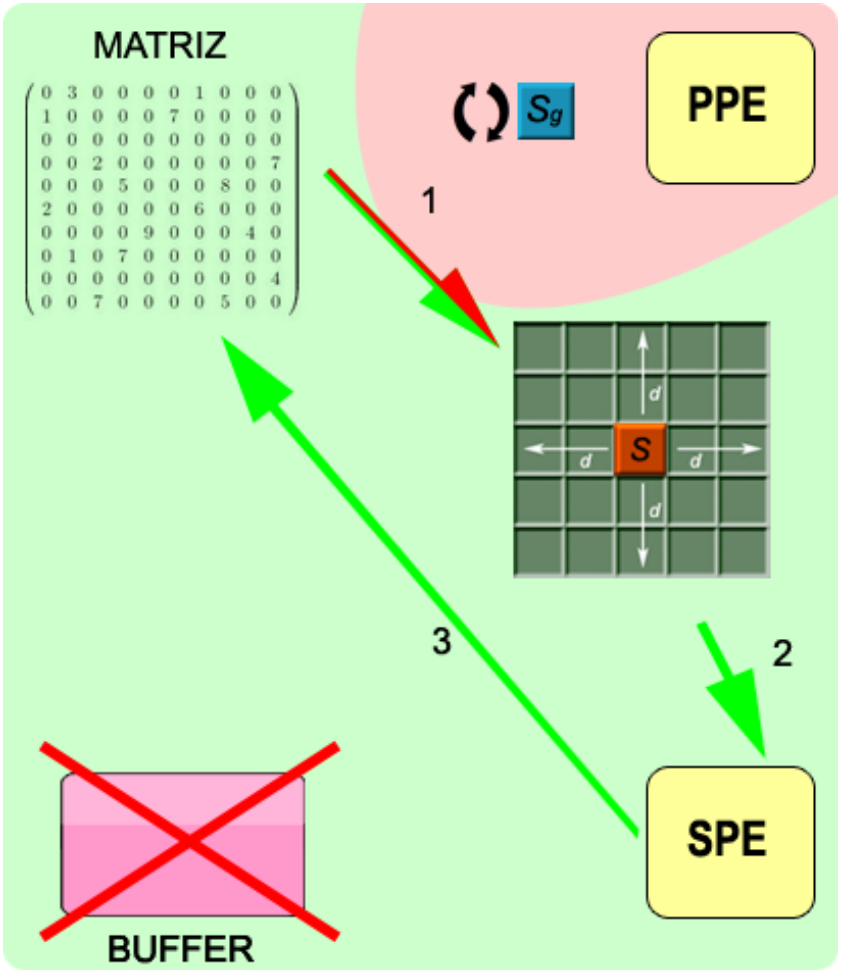


[1]

[1] *Trade-off* processamento × espaço

Montagem das tarefas

Terceira abordagem: reduzindo custo de sincronização PPE-SPEs

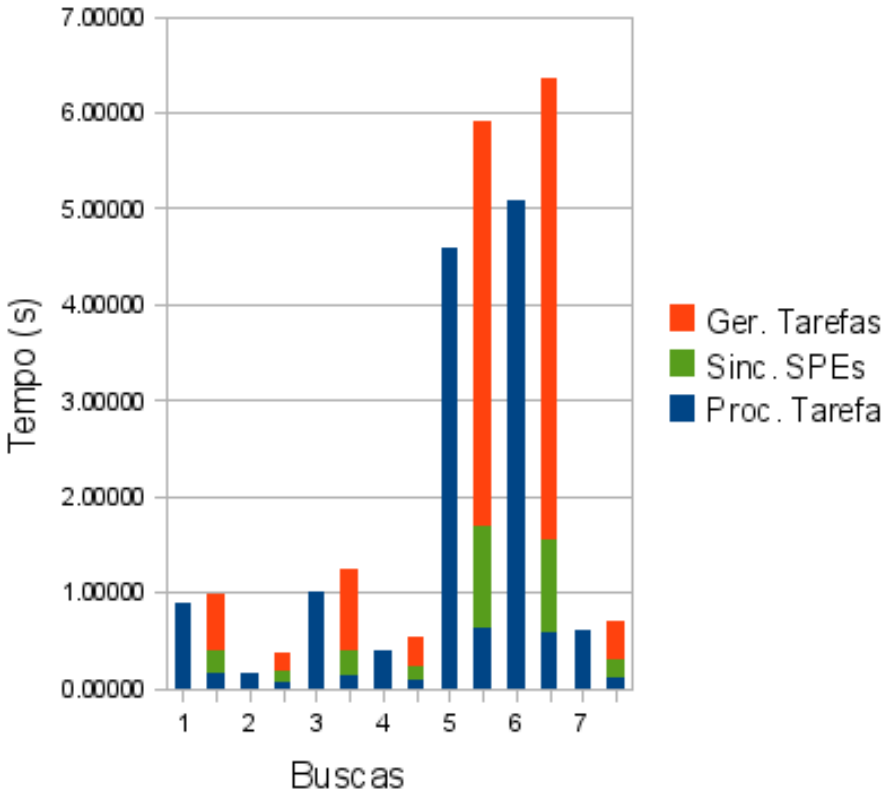


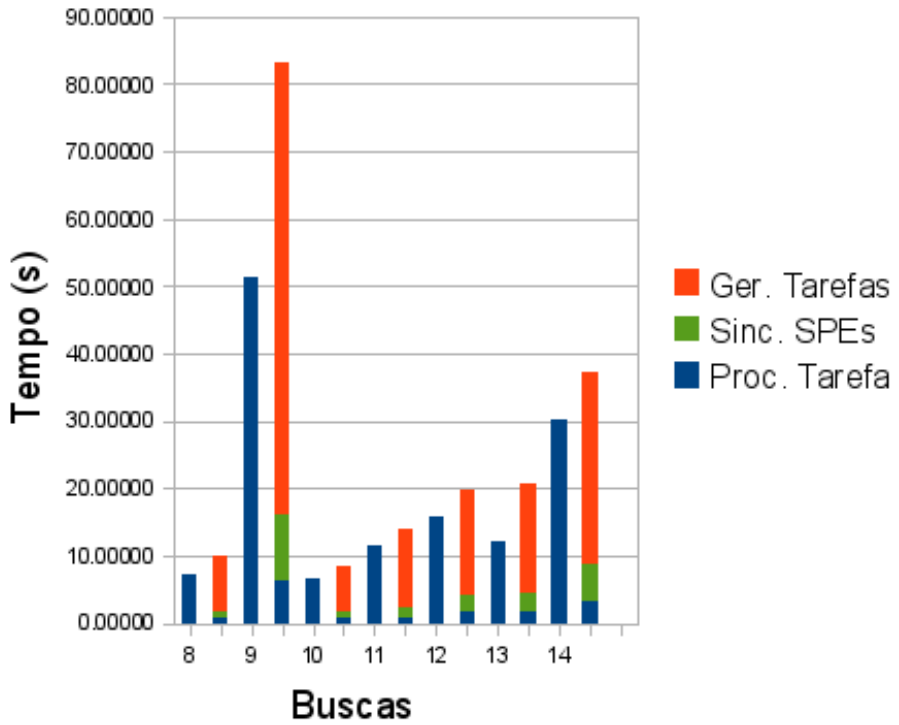
Avaliação experimental

- Avaliações apresentadas:
  - tempo de Execução(para as 3 abordagens de montagem de tarefas);
  - *throughput*;
  - ambiente simulado de um jogo;
- Avaliações realizadas em *blades* IBM BladeCenter QS20;
- Foram feitas 14 buscas, executadas em mapas de jogos reais;
- As 7 primeiras buscas possuem menor tempo de convergência e as 7 últimas um tempo maior (versão sequencial);
- Algoritmo utilizado: LSS-LRTA*;

Tempo de execução até a convergência

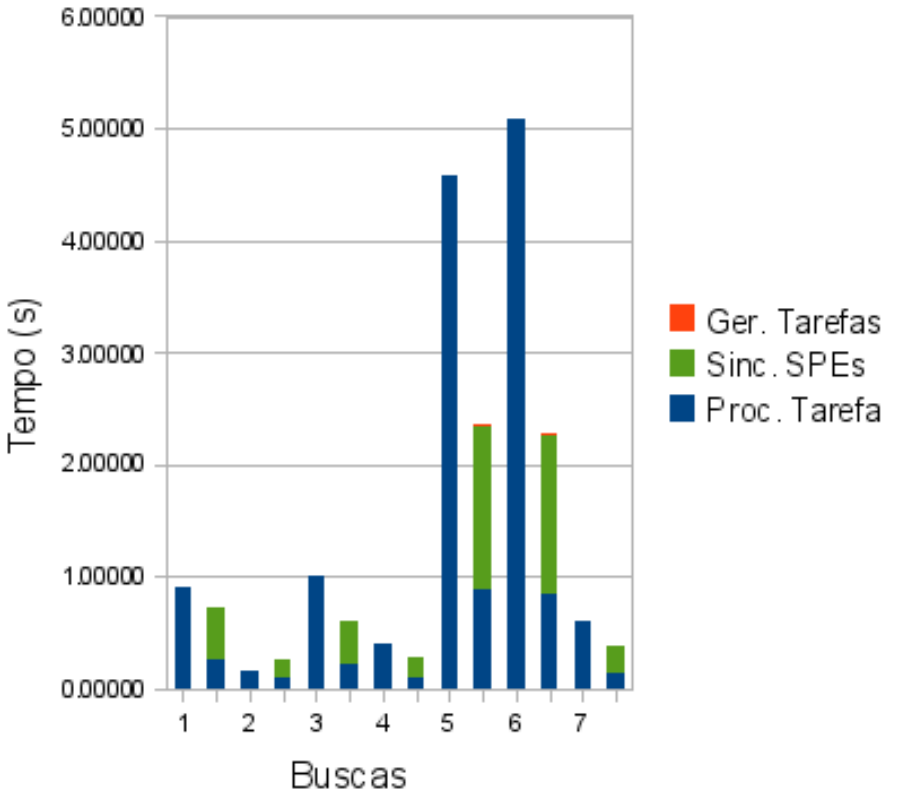
Primeira abordagem

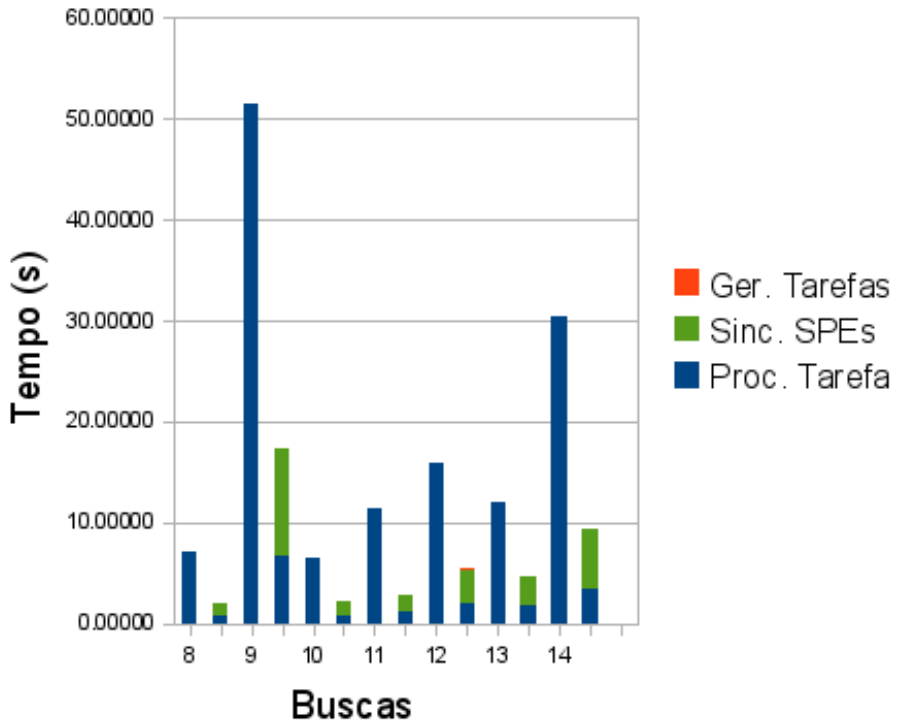




Tempo de execução até a convergência

Segunda abordagem[2]

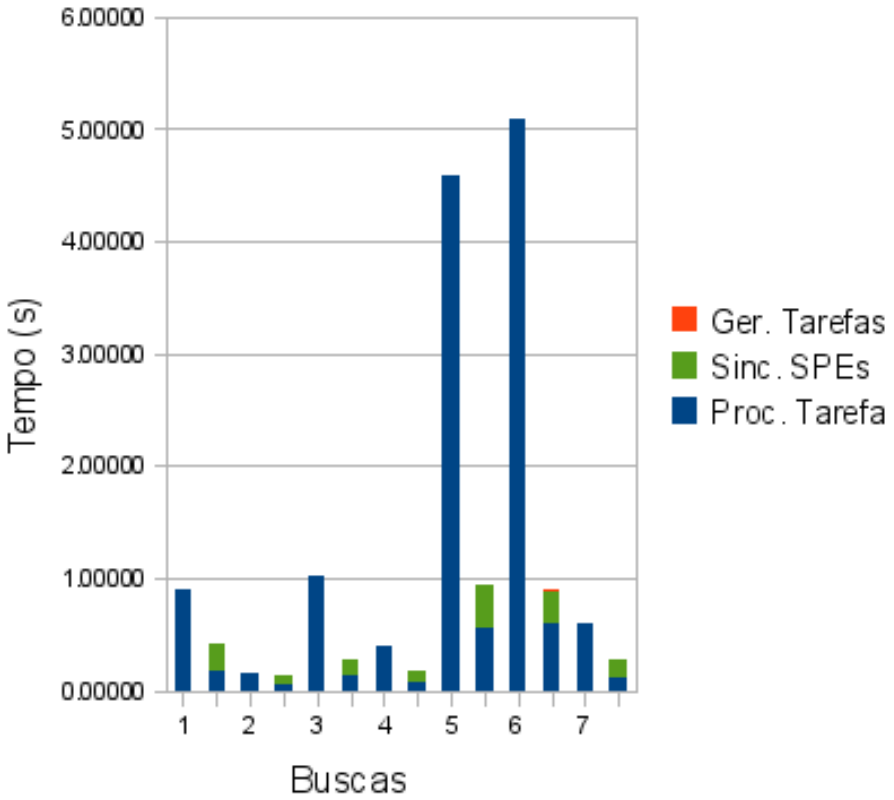


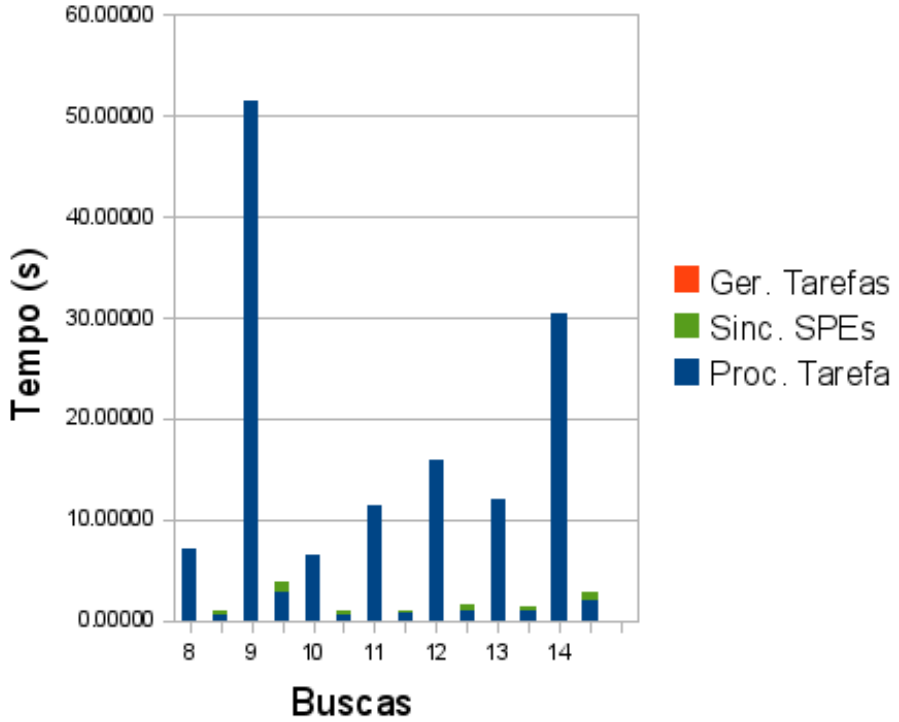


[2] Atenção à escala do eixo Y

Tempo de execução até a convergência

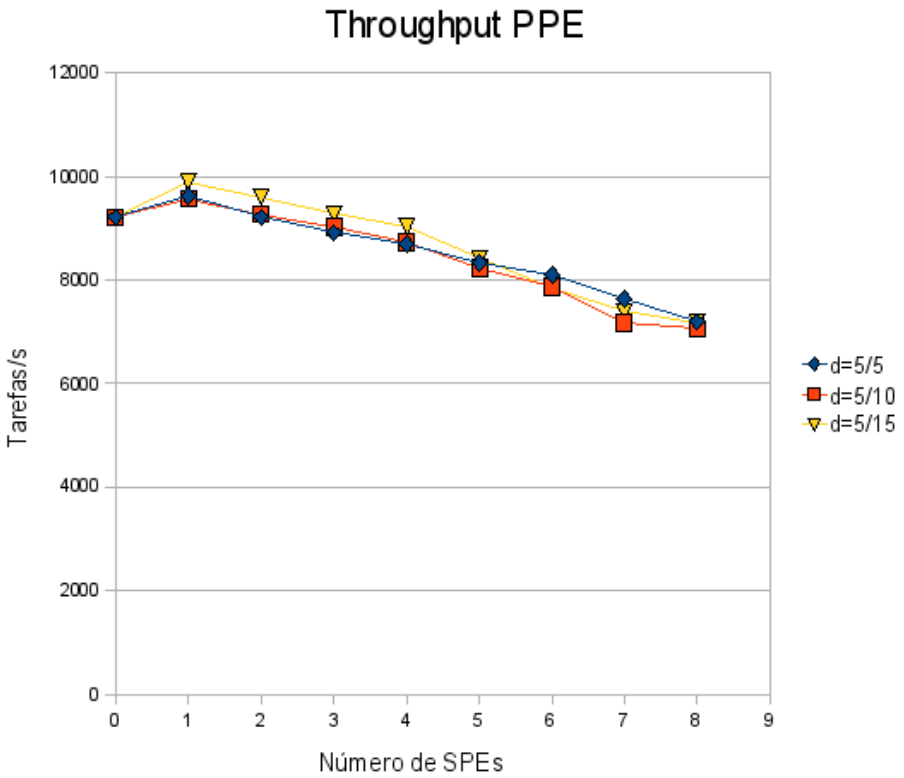
Terceira abordagem

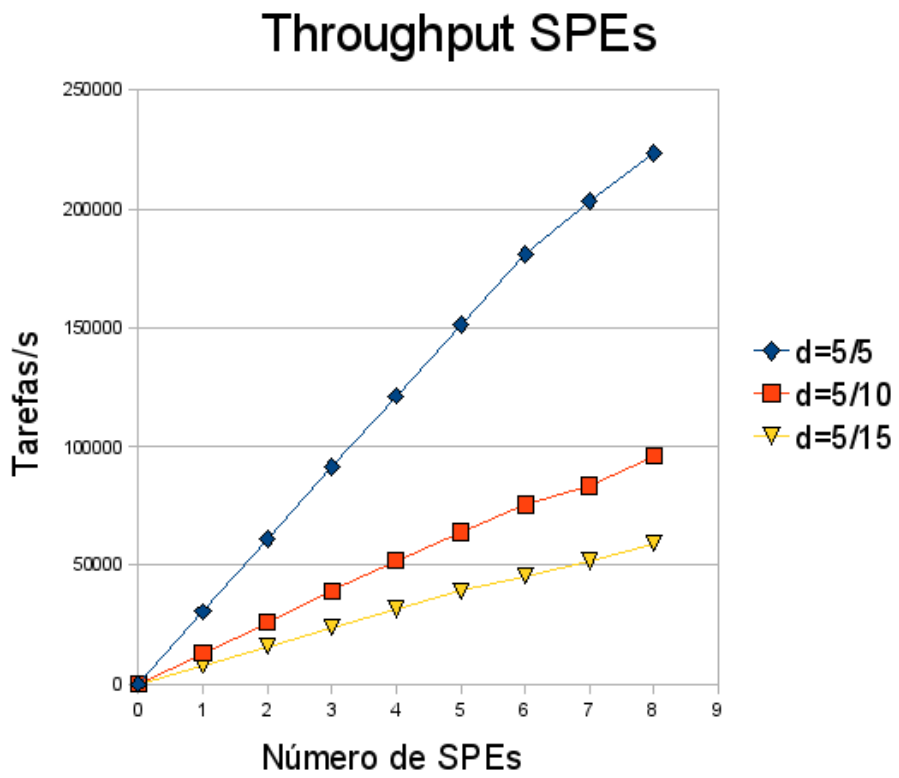




Throughput

- Medido em tarefas processadas por segundo;
- Foi utilizada a busca 13, cuja versão sequencial converge em 675 tentativas;

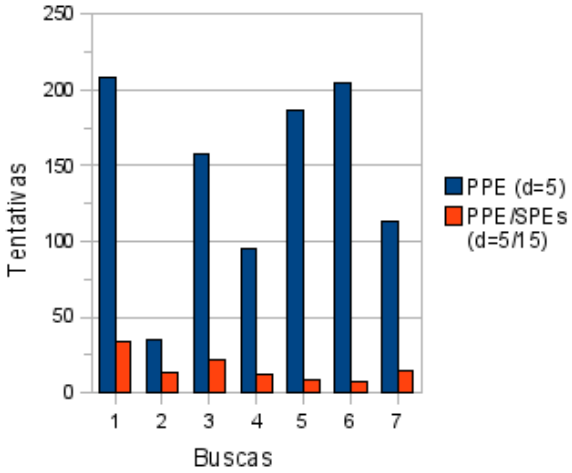


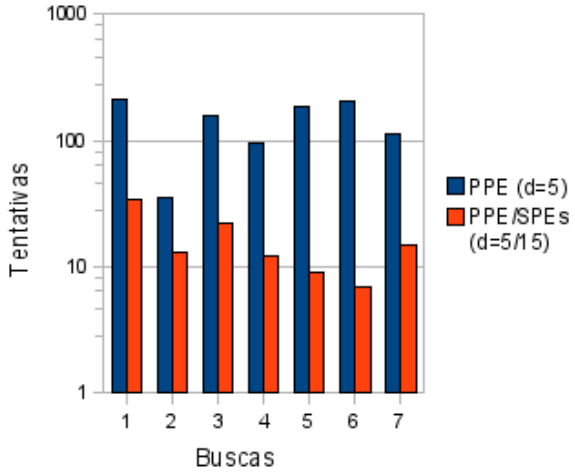


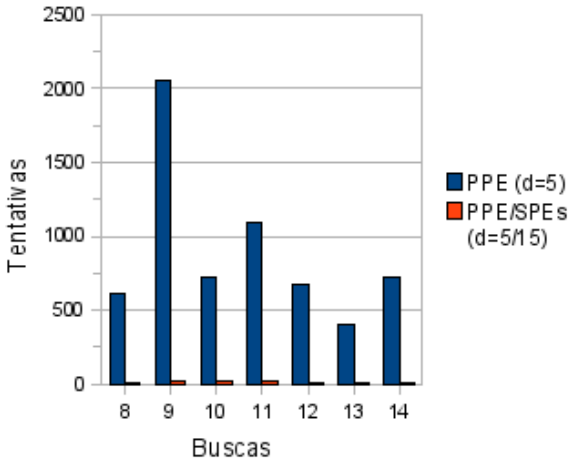
Execução em ambiente simulado

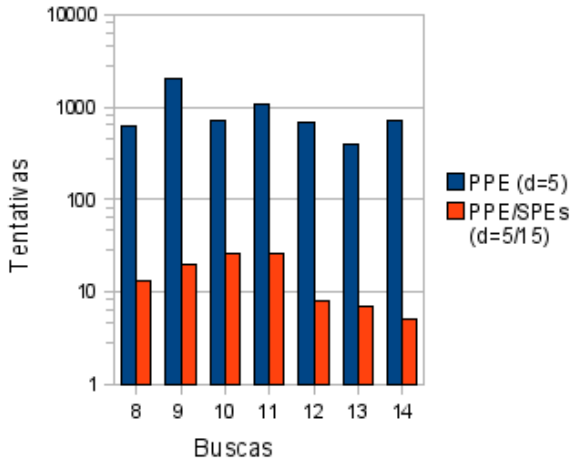
- Condições de tempo similares a de um jogo: intercalando busca de caminho com ação dos NPCs;
- Jogos de consoles atuais possuem taxa de atualização fixa em 60Hz;
- No pior dos casos, é necessário uma chamada do algoritmo por quadro: 1/60s entre as requisições;

Execução em ambiente simulado









Sumário dos resultados

- A primeira abordagem, que preserva a tabela Hash, não mostrou desempenho satisfatório;
- A abordagem utilizando matriz esparsa elimina a etapa de preenchimento dos *buffers* pelo PPE;
- A última abordagem, acrescentando maior autonomia aos SPEs, apresentou a melhor performance;
- A ordenação das tarefas, baseada na classificação dos estados, reduz a área de exploração das buscas;

Limitações e trabalhos futuros

- Algumas observações:
  - A sucessiva sobreposição das áreas atualizadas pelas tarefas pode comprometer o desempenho;
  - Consequência da maior concentração de buscas em regiões com mais mínimos locais;
- Outros contextos para aplicação:
  - CDNs, previsão de tráfego, rede de sensores sem fio, redes *ad hoc*, P2P, etc.

Perguntas ?

Algoritmos de Busca de Caminho incrementais

Busca de caminho incremental:

- Possui algumas caraterísticas similares à busca clássica;
- Ideal para ambientes desconhecidos;

Busca por uma solução completa partindo do estado inicial	Executa
Se inviável, procura solução partindo do estado corrente	Executa

... itera até encontrar o objetivo

Algoritmos de Busca de Caminho incrementais

Variações do A*

- **Local Repair A***(Stout, 1996): algoritmo incremental, realiza uma busca A* sob suposição de espaço livre, até o objetivo ou obstáculo;
- **Iterative-Deepening A***(Korf, 1985): realiza série de buscas de diferentes profundidades, atreladas a um limite de custo;
- **D***(Stentz, 1995) e **Lifelong Planning A***(Koenig & Likhachev, 2002): utilizam informações da busca anterior para o replanejamento;

Paralelização de algoritmos de busca de caminho

- **Asynchronous Parallel IA***(Reinefeld & Schnecke): diferentes partes do espaço de busca processadas de forma assíncrona, em paralelo, pela rotina sequencial mais rápida disponível;
- **Parallel Retracting A***(Evett et al., 1995): retrai estados já expandidos pelo algoritmo cuja heurística é menos promissora e distribui os estados gerados pela busca entre os processadores, onde cada um expande o estado local mais promissor;
- **Single agent parallel window search**(Powley & Korf, 1989) realiza diversas buscas IDA* simultaneamente, cada uma em uma janela de busca, com limites de custos diferentes, onde todos processos compartilham a mesma ordenação dos estados;

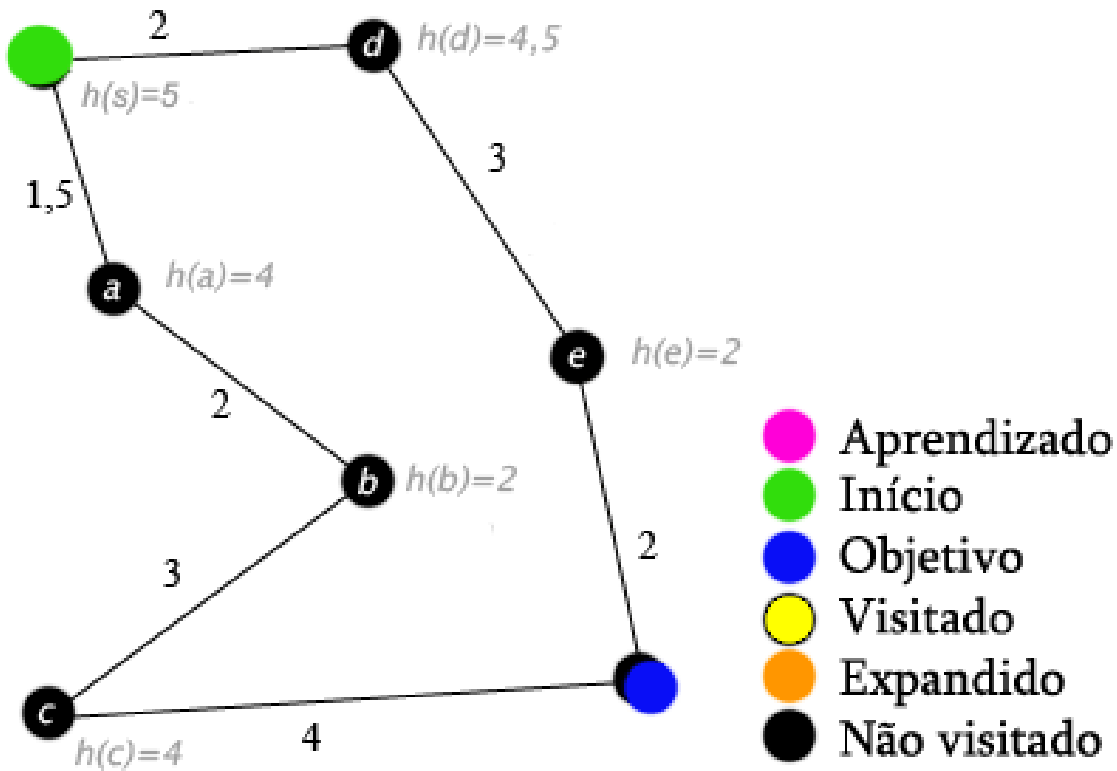
Algoritmos de busca de caminho em tempo real

LRTA*

- **Learning Real-Time A*** (Korf, 1990) foi o primeiro algoritmo de busca em tempo real a ser criado;
- O espaço de busca local corresponde somente aos vizinhos imediatos do estado corrente;
- A profunidade da busca (*lookahead*) é igual a 1;
- Espaço de aprendizado local é o próprio estado corrente;

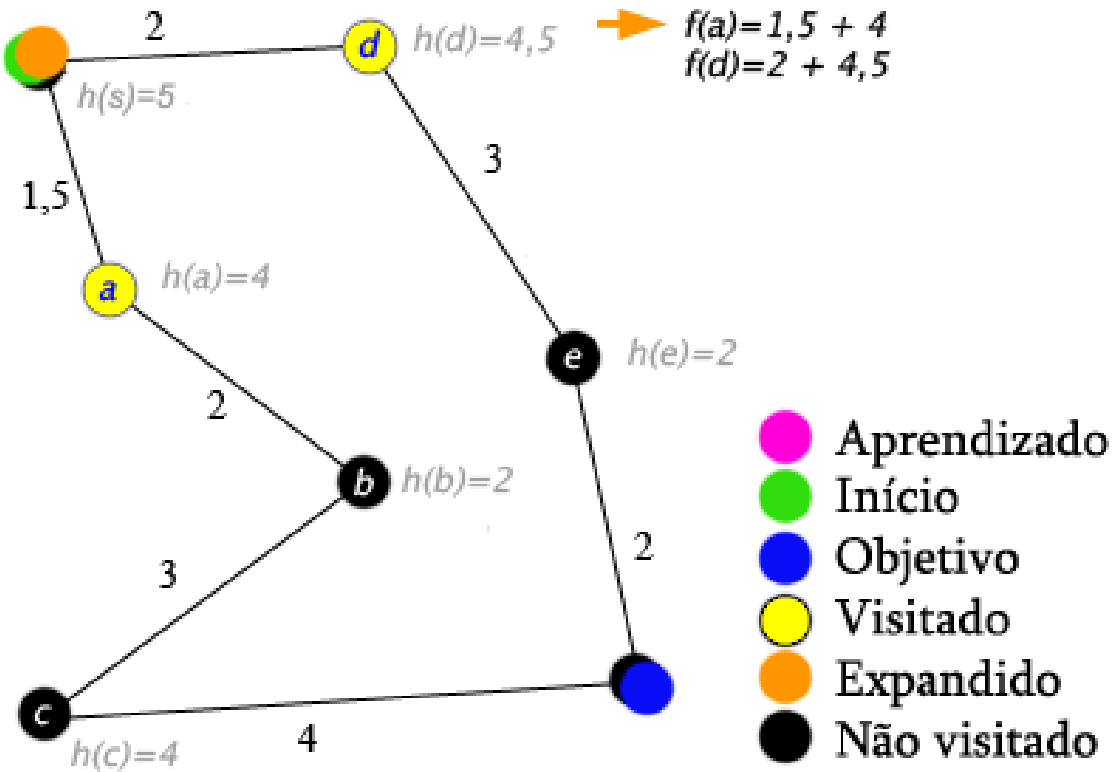
Algoritmos de busca de caminho em tempo real

LRTA*



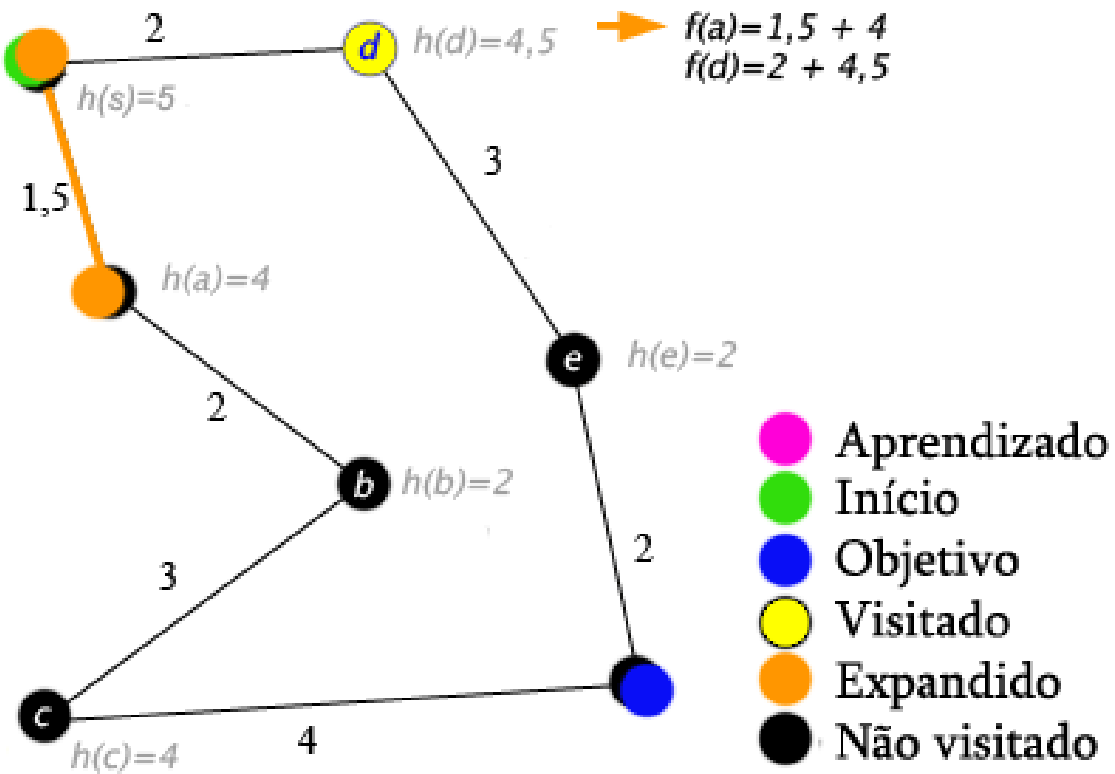
Algoritmos de busca de caminho em tempo real

LRTA*



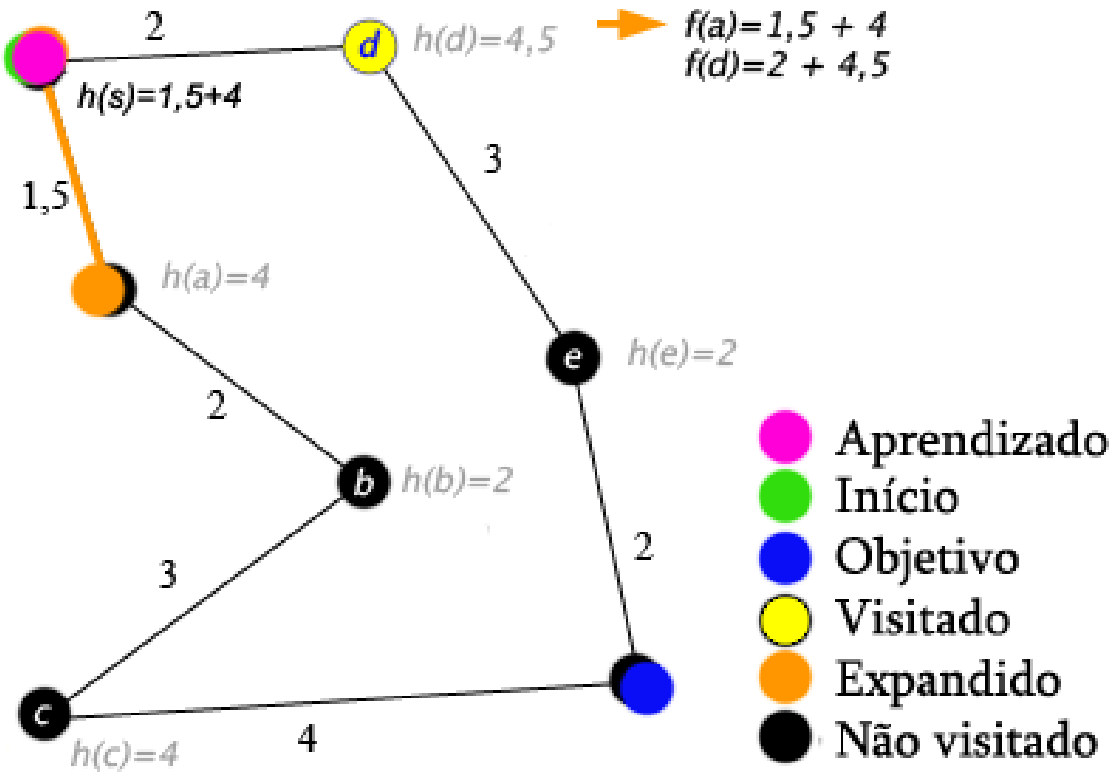
Algoritmos de busca de caminho em tempo real

LRTA*



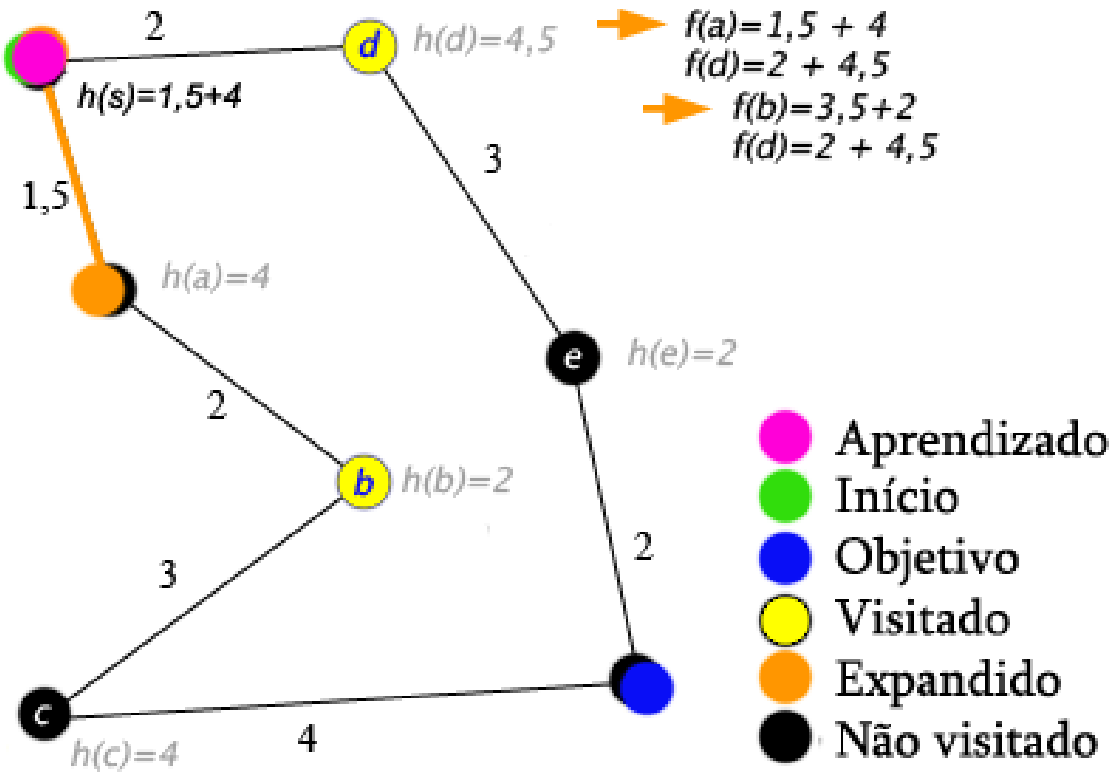
Algoritmos de busca de caminho em tempo real

LRTA*



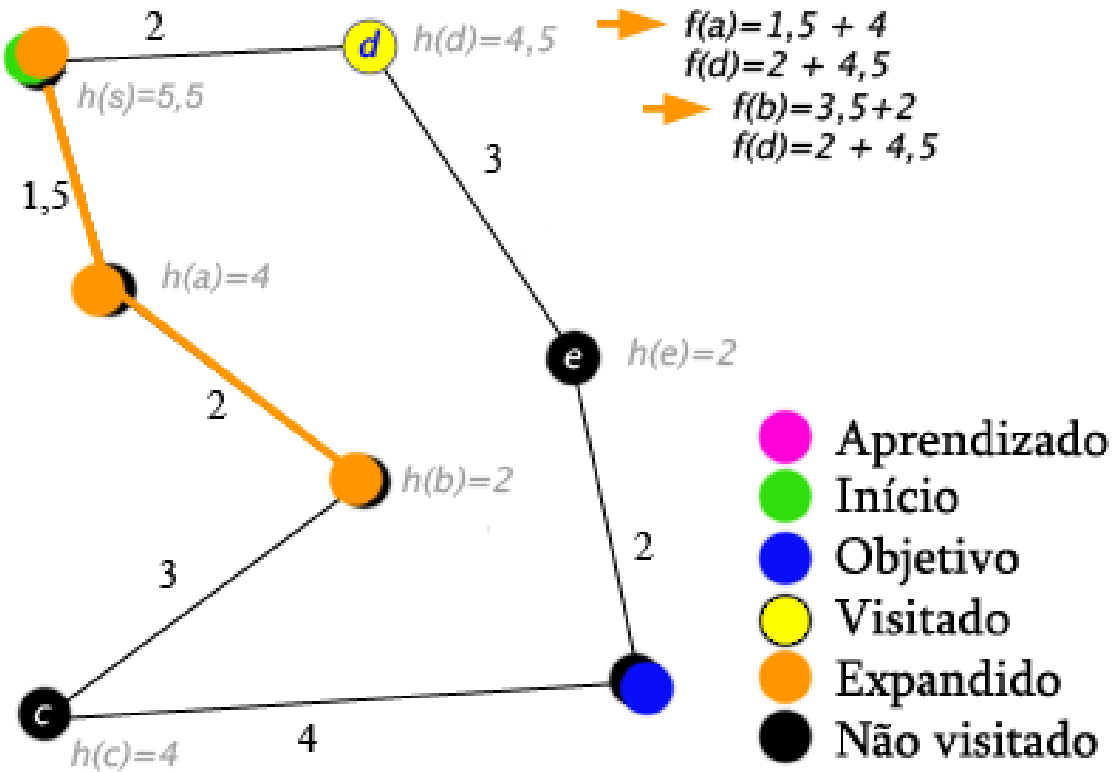
Algoritmos de busca de caminho em tempo real

LRTA*



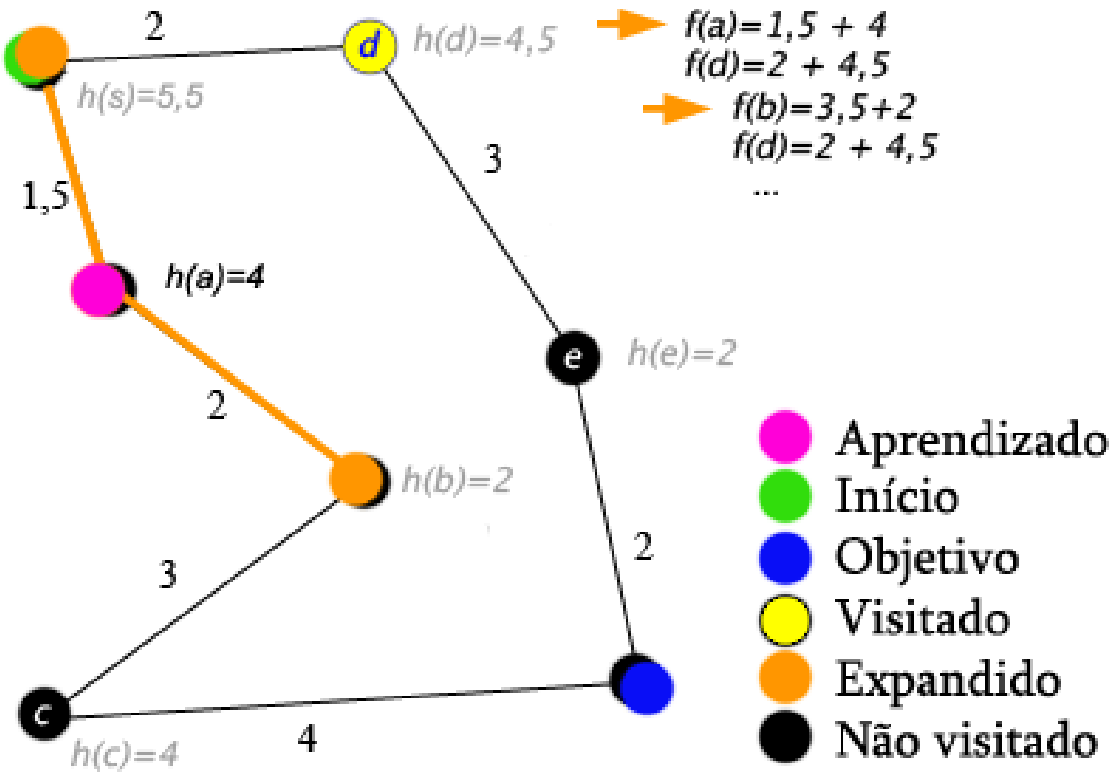
Algoritmos de busca de caminho em tempo real

LRTA*



Algoritmos de busca de caminho em tempo real

LRTA*



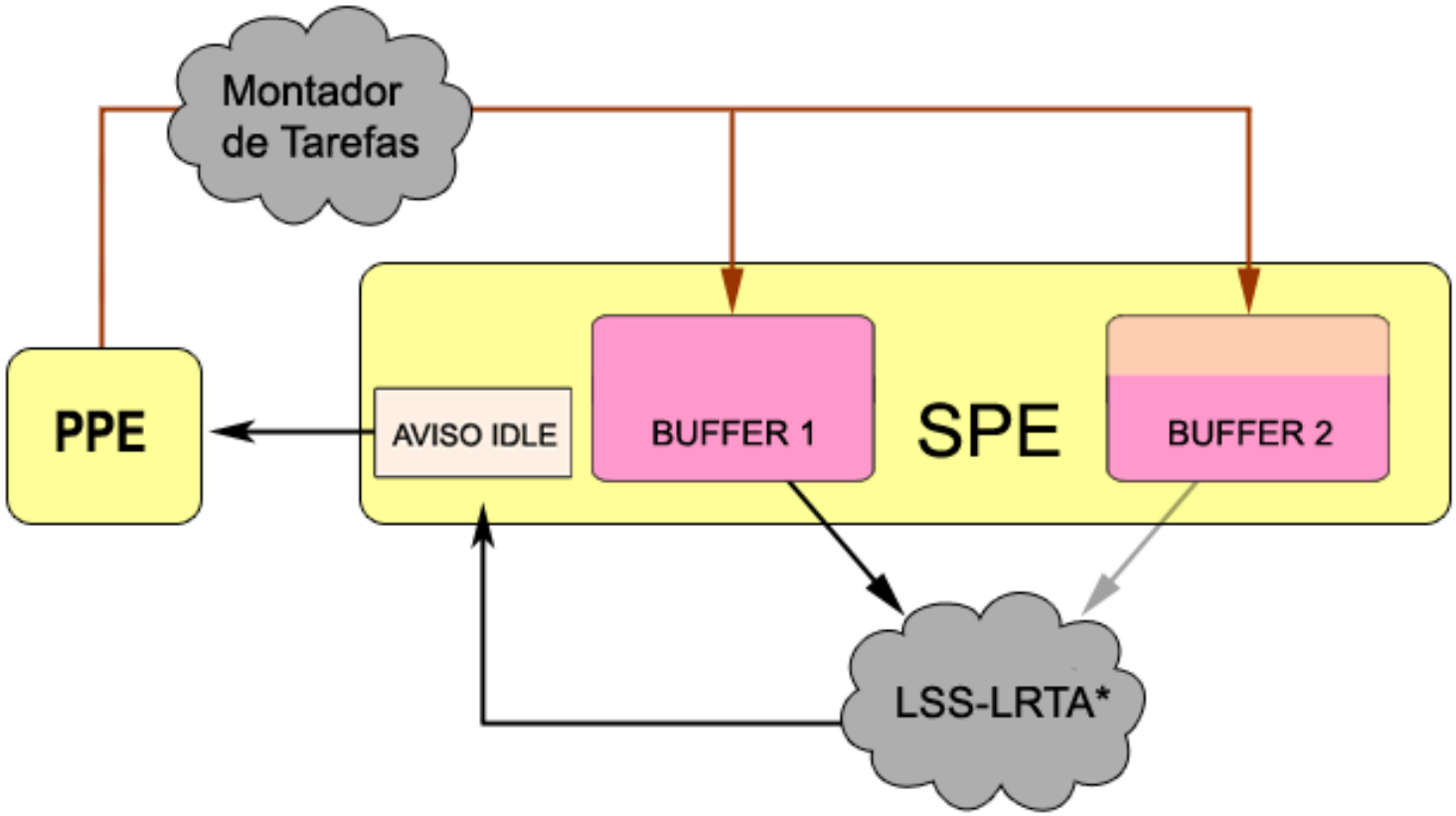
Algoritmos de busca de caminho em tempo real

Outros algoritmos

- **LRTA*(k)**(Hernández & Meseguer, 2005) é uma variação do LRTA* com aprendizado realizado por método chamado propagação limitada;
- **Path Refinement Learning Real-time Search**(Bulitko et al., 2007) pré computa abstrações baseadas em cliques de grafos, realizando então diferentes níveis de busca, onde as buscas em níveis mais altos delimitam **corredores** e *subgoals* para as buscas inferiores;
- **kNN LRTA***(Bulitko & Bjornsson, 2009) se baseia no kNN para construir um banco de dados de *subgoals*, verificando no banco de dados casos mais similares ao problema em questão;

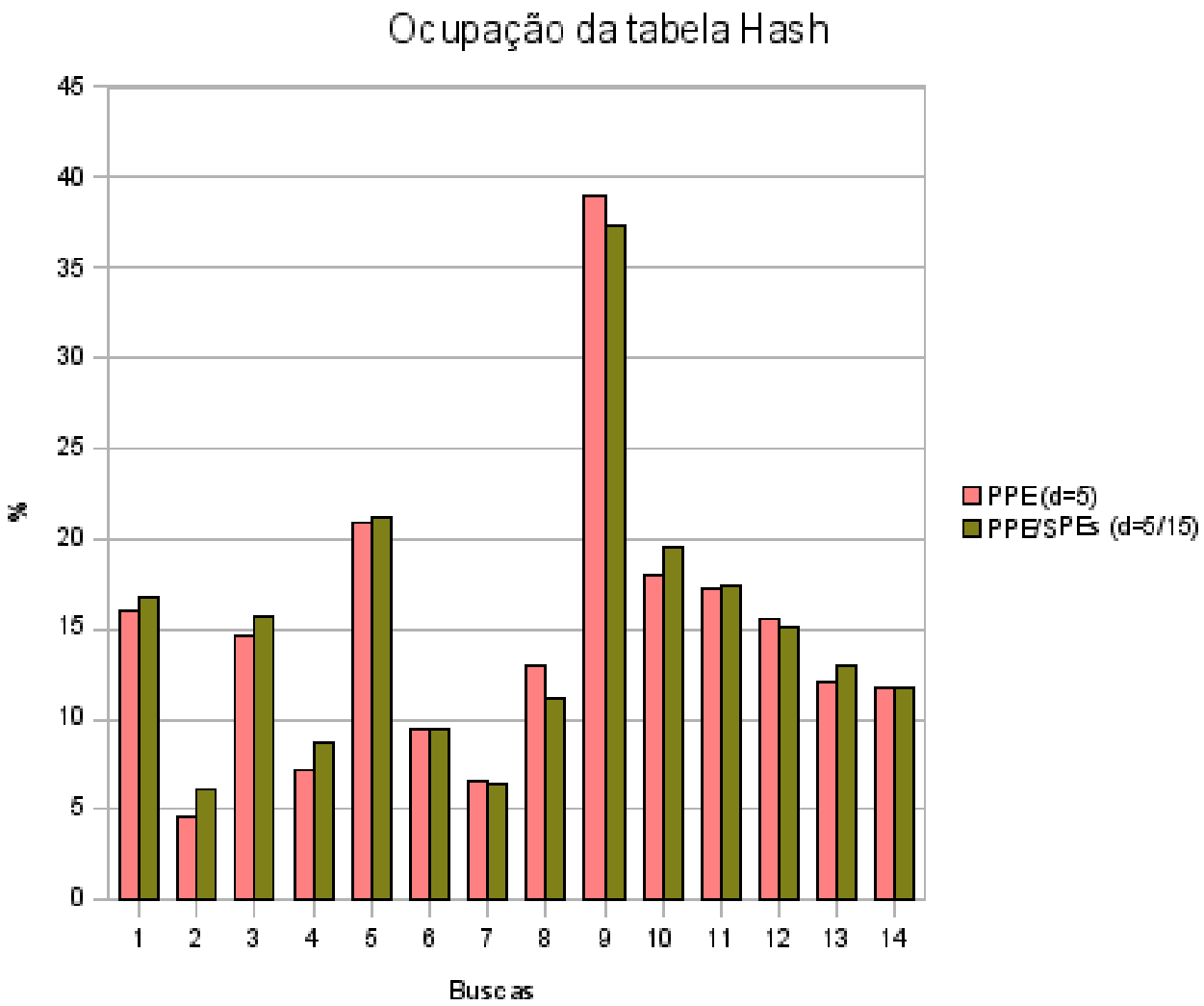
Fluxo de execução no SPE

Diagrama de execução



Avaliação experimental

Ocupação do hash



Variação do número de SPEs

- Tempo para a busca 13;
- *Lookaheads* **5**(PPE) e **15**(SPEs);
- Tempos semelhantes 5-8 SPEs: **sobreposição** de tarefas;