*Busca e Aprendizagem em Tempo Real – Um Ambiente de Testes Para Algoritmos Baseados no LRTA**

França, Tiago Peres[1]; Reis, Pedro Augusto da Silva[2]; Lopes, Carlos Roberto[3]
[1, 2, 3] Universidade Federal de Uberlândia
pedro_augusto@comp.ufu.br
crlopes@facom.ufu.br

1 - Introdução

Aplicações com mapas são comuns hoje em dia. Elas estão presentes, por exemplo, em sistemas de localização, jogos e em redes de computadores. É de praxe, por exemplo, antes de viajar acessar algum serviço na web que trace a rota mais curta entre duas localizações.

Algoritmos de busca, dentre os quais pode-se mencionar o algoritmo de Dijkstra, varrem todo o espaço de possibilidades, sendo inviáveis para certas aplicações, como o planejamento do caminho entre dois pontos citado anteriormente. Por esse motivo, faz-se necessário o uso de algoritmos de busca em tempo real.

Algoritmos de busca em tempo real não procuram a solução ótima, mas sim uma que se aproxime desta. Para melhorar o resultado encontrado por esses algoritmos, é uma boa ideia dotá-los de aprendizagem. Assim, a cada execução, o algoritmo tentará obter uma melhor solução com base em soluções obtidas anteriormente.

Existem vários algoritmos de busca e aprendizagem em tempo real, dentre eles, os mais conhecidos são o LRTA* e suas variantes. Dado um determinado problema, uma das grandes dificuldades é decidir qual deles utilizar e com quais parâmetros [Bulitko & Lee, 2006].

Procurando resolver este problema, propõe-se neste trabalho um ambiente que facilite a comparação e o entendimento desses algoritmos bem como dos parâmetros utilizados pelos mesmos.

2 - Materiais e métodos

2.1 – Descrição do Problema

Dado um mapa M, uma posição inicial S0, e uma meta G o objetivo consiste em encontrar um caminho que ligue S0 a G. Aqui estamos considerando que tal caminho será obtido para que um agente (no caso um **robô**) possa segui-lo.

2.2 – O Mapa

O mapa consiste de uma malha bidimensional que possui largura W e altura H. Cada posição é identificada por um par (0 < i < H, 0 < j < W), onde i identifica uma linha e j uma coluna do mapa. Essas posições assumem um de três estados possíveis: *bloqueado* - a posição é inacessível -, *livre* - a posição é acessível - e *objetivo* - a posição caracteriza um objetivo a ser alcançado pelo robô.

2.3 - Caminho, posição inicial e objetivos

O caminho é a saída do programa e representa uma sequência de posições assumidas desde a posição inicial do robô até uma posição pertencente ao conjunto de objetivos.

2.4 - Robô

Por último, o robô é o agente que percorre o mapa, pode-se movimentar para norte (i ← i – 1), sul (i ← i + 1), leste (j ← j + 1) e oeste (j ← j – 1).

*2.5 – LRTA**

Proposto por Korf (1990), o LRTA* é um algoritmo de busca e aprendizagem em tempo real que propõe uma maneira inteligente de se encontrar um caminho entre dois vértices em um grafo, a partir de heurísticas com valores iniciais pré-calculados.

Dada uma posição p, o LRTA* procura a posição adjacente p' com o menor f = g + h, onde g é o custo de se sair de p e chegar em p', e h é a heurística de p'. Assim, o agente assume a posição p' de menor f, e a heurística (h) de p é atualizada para o valor de f, se f > h.

A heurística de cada estado do mapa é pré-calculada de acordo com a distância

Manhatan deste ao objetivo mais próximo, sendo atualizada ao decorrer da execução do LRTA*.

O algoritmo converge para a melhor solução quando executado várias vezes sobre o mesmo problema. A convergência ocorre no decorrer de sucessivas atualizações nos valores das heurísticas.

2.6 – Descrição do Ambiente

Este trabalho propõe um ambiente construído em Java Swing com o objetivo de facilitar o entendimento do comportamento de LRTA* e suas variantes (SLA e *γ-trap*) . O *software* é formado de um editor de mapas e um visualizador de comportamento para os algoritmos.

2.7 – Visualizador de Comportamento

O comportamento do algoritmo pode ser visualizado de duas maneiras diferentes, pelo visualizador de comportamento: visualizando a execução através de uma animação ou gerando estatísticas sobre as execuções. Na janela de dados estatísticos é possível visualizar três seções separadas por abas: *Path Length*, *Execution Cost* e *Planning Cost*.

Path length (comprimento do caminho) é a medida que conta o número de passos que o robô levou para sair da posição inicial e chegar ao objetivo. A contagem é feita para cada execução, de cada robô, até que haja convergência. Esta medida é representada por um gráfico de linhas que relaciona cada execução com um comprimento de caminho. Há várias linhas neste gráfico, cada uma representando um algoritmo e parâmetros diferentes.

Execution cost (custo de execução) é a medida que conta o número de estados assumidos da posição inicial do robô, em sua primeira execução, ao estado final em sua última execução. Esta medida representa a soma dos comprimentos de caminho de cada execução, até a convergência. O custo de execução é representado por um gráfico de barras, onde cada barra é um algoritmo com seus parâmetros. A altura da barra é definida pelo custo de execução do robô.

Planning Cost (custo de planejamento) é a média do custo para se planejar uma ação do robô. O custo para se planejar uma ação é dado pelo número de estados considerados pelo robô ao decidir qual será a próxima posição do mapa a ser ocupada por ele. Esta medida é representada por um gráfico de barras, onde cada barra é um algoritmo e seus parâmetros. A altura da barra é definida pelo custo de planejamento do robô.

2.8 - Experimentos

Para a realização dos testes, será utilizado o mapa M1 representado na Figura 1.

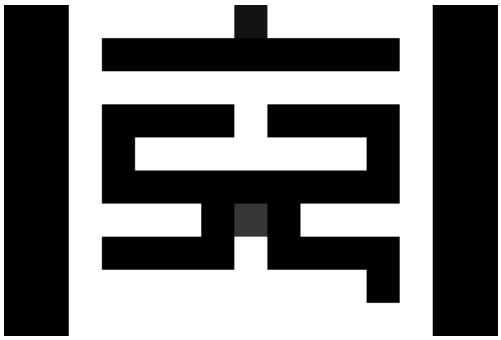


Figura 1 – Mapa M1, posição inicial representada pelo ponto cinza escuro

A avaliação dos algoritmos (LRTA*, SLA e γ-trap) será composta pelas métricas especificadas na sessão 2.7. Detalhes sobre estes algoritmos podem ser encontrados em [Bulitko, 2004; Bulitko & Lee, 2006].

3 - Resultados e discussão

*3.1 Execução do LRTA**

Na Figura 2 vemos um gráfico com a variação do caminho gerado pelo LRTA* no mapa M1.

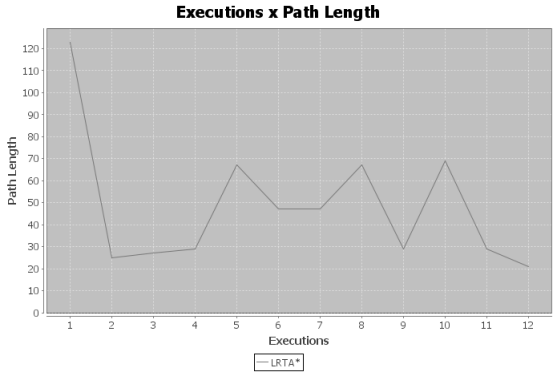


Figura 2 - Variação no comprimento do caminho gerado pelo LRTA* em M1.

Pode-se perceber que, primeiramente, o LRTA* toma um longo caminho para chegar

ao objetivo. Na segunda execução, o número de passos cai bastante. Nas execuções seguintes o LRTA* tenta encontrar outros caminhos que possam levar o agente a chegar ao objetivo mais rapidamente. A convergência acontece na execução 12.

3.2 – Execução do SLA

Na Figura 3 temos um gráfico comparativo entre o custo de execução e o custo de planejamento do LRTA* e do SLA.

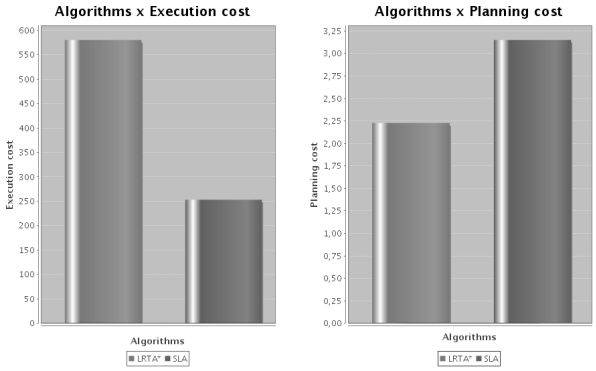


Figura 3 – custo de execução do LRTA* comparado ao SLA e à direita custo de planejamento do LRTA* comparado ao SLA.

No SLA, a primeira execução leva a muito mais passos que o LRTA*, mas em compensação, já na segunda execução, devido à realização do *backtracing*, o SLA toma o caminho de menor custo. Quanto ao custo de planejamento, pode-se perceber que o SLA é mais custoso que o LRTA*. Isso pode ser explicado pela estratégia de *backtracking*, pois além de analisar as próximas posições, o SLA também considera voltar para o estado anterior. Um alto custo de planejamento significa que o robô levará mais tempo numa posição decidindo qual ação tomar.

3.3 – Execução do y-trap

A execução do γ-trap foi realizada com os seguintes parâmetros: com d=4 e γ=1 e γ-trap com d=4 e γ=3 em M1. Onde d é a profundidade do *lookahead* e y é o peso.

A profundidade do *lookahead* (d) é a quantidade de posições que o agente pode visualizar a sua frente, no LRTA* esse valor é 1, no Y-trap ele pode variar no intervalo [1, +∞).

O parâmetro γ é o peso dado a distância entre duas posições no cálculo do custo de movimento. Este parâmetro varia no intervalo (0, 1]. O cálculo de custo de movimento da posição p para a posição p' é feita através da equação c(p') = γ * dist(p, p') + h(p'), onde h(x) é a heurística da posição x e dist(p, p') é a distância entre as posições p e p'.

γ-trap com γ=1 percorreu um caminho maior que o LRTA*. Isso se deve ao fato de o *γ-trap* fazer *backtracking*, assim como o SLA. O número de passos depende também do lookahead (parâmetro d), O *lookahead* do LRTA* é um, enquanto o do *γ-trap*, neste caso, é quatro. Quanto maior o *lookahead*, mais rápido o robô encontra seu objetivo e mais curto é o caminho. Em função disto, apesar do *backtracking*, a diferença do comprimento do caminho na primeira execução entre os dois algoritmos foi pequena.

4 - Considerações finais

Através do software desenvolvido foi possível testar os algoritmos num ambiente que facilita a visualização do comportamento, das vantagens e das desvantagens em se utilizar cada um deles.

Não é do nosso conhecimento a existência de uma ferramenta similar. Pretendemos torná-la pública e atualizá-la de modo a incorporar novos algoritmos que sejam propostos. Entendemos que tal ambiente é valioso para usuários que queiram usar LRTA* e tenham dificuldade na escolha de uma abordagem apropriada.

5 - Referências

Bulitko, V. & Lee, G. (2006) “Learning in Real-Time Search: A Unifying Framework”, In: Journal of Artificial Intelligence Research 25 (2006) 119-157.

Bulitko, V. (2004). “Learning for Adaptive Real-time Search”, In: CoRR cs.AI/0407016.

Agradecimentos

Os autores agradecem ao Progama de Educação Tutorial / MEC pela oportunidade de realizar este trabalho.