Space D*: Um Algoritmo para Path Planning Multi-Robôs

Renan Maffei[1], Silvia Botelho[1], Luan Silveira[1], Paulo Drews Jr.[1], Nelson Duarte Filho[1], Alessandro Bicho[1], Felipe Almeida[1], Matheus Longaray[1]

[1]Centro de Ciências Computacionais (C3)
Universidade Federal do Rio Grande (FURG) – Rio Grande – RS – Brazil

silviacb@furg.br

Abstract. *This paper describes a new method of path planning for multiple robots in unknown environments. The method, called Space D*, is based on two algorithms: the D*, which is a incremental graph search algorithm and the Space Colonization algorithm, previously used to simulate crowd behaviors. The major contribution of the proposed method is the focus on generating paths in spacious environments to help facilitate the control of robots, and thus presenting itself in a viable way for use in real environments. The results obtained validates the approach and show the advantage in relation to D* method.*

Resumo. *Este trabalho descreve um novo método de path planning para múltiplos robôs em ambientes desconhecidos. O método, chamado Space D*, é baseado em dois algoritmos: o D*, que é um algoritmo de busca em grafos incremental, e o algoritmo de Colonização do Espaço, utilizado previamente para a simulação do comportamento de multidões. A maior contribuição do método proposto é o enfoque sobre a geração de caminhos por ambientes espaçosos, visando facilitar o controle dos robôs, e, dessa forma, apresentando-se de forma viável para a utilização em ambientes reais. Os resultados obtidos validam a abordagem e mostram a vantagem em relação ao método D*.*

1. Introdução

Planejar movimentos livres de colisão para robôs autônomos situados em ambientes com obstáculos é um dos principais problemas da robótica [Latombe 1991] [LaValle 2006].

Uma área do planejamento de caminhos bastante estudada é a de ambientes com múltiplos robôs independentes. Para esse tipo de problema, dois tipos de abordagens podem ser efetuadas: a abordagem centralizada e a desacoplada. Na abordagem centralizada, o planejamento é feito considerando-se um grande e único robô formado por vários segmentos (cada um dos robôs do ambiente), situado dentro de um grande espaço de configuração (correspondente à união dos espaços de configuração de cada robô), onde cada segmento tem uma configuração final desejada (o objetivo de cada robô). O problema deste tipo de abordagem é que o espaço de configuração composto pelos robôs cresce exponencialmente com o aumento do número deles, o que se torna altamente inviável.

Em contra partida, apesar de incompleta, a abordagem desacoplada reduz o problema em planejar a movimentação de cada robô individualmente e posteriormente adequá-las. Além disso, uma alternativa viável são técnicas desacopladas distribuídas, considerando-se a facilidade de uso de técnicas de processamento distribuído para a implementação de tais abordagens. Nessas técnicas, cada robô planeja a sua

movimentação baseando-se no seu conhecimento local e nas interações com outros robôs. A diferença principal das abordagens distribuídas para as técnicas somente desacopladas é que a etapa de adequação do caminho deve ser feita de forma distribuída e em tempo real. Técnicas distribuídas apresentam maior robustez, pois aceitam melhor as falhas e incertezas na atuação de cada robô, ou seja, um único robô defeituoso não interrompe o funcionamento do todo.

Outra complexidade associada ao problema de planejamento de caminhos refere-se ao nível de conhecimento sobre o ambiente, isto é, a certeza sobre a presença de obstáculos no cenário. Uma estratégia de navegação utilizada quando não se possui o conhecimento exato do ambiente é o chamado *path planning* baseado em sensoriamento [Choset and Burdick 1994]. Neste tipo de técnica planeja-se o menor caminho baseado na configuração atual do ambiente e conforme novos obstáculos são detectados, novos caminhos devem ser planejados.

Um dos algoritmos mais populares para o tratamento deste tipo de problema é o algoritmo D* (Focussed Dynamic A*) [Stentz 1995], que adapta a otimalidade do A* para a dinamicidade do ambiente, mesclando heurísticas com buscas incrementais, e alcançando consideráveis ganhos sobre execuções repetidas do A*.

Diversos trabalhos foram propostos baseando-se no D* e, posteriormente, no D* Lite [Koenig and Likhachev 2002], que implementa a mesma estratégia do D*, porém de forma simplificada e com um grau de eficiência igual ou até maior do que o D*. Em [Ferguson and Stentz 2005], foi proposta uma extensão para o D* e D* Lite utilizando interpolação linear para produzir caminhos mais suaves, contornando, assim, a questão de poucas possibilidades de transição entre as células. Outra extensão do D* Lite foi proposta em [Likhachev et al. 2005], onde o algoritmo foi adaptado para sistemas com escassez de tempo. Neste tipo de técnica, como em [van den Berg et al. 2006], o planejamento é feito incrementalmente e conforme se dispõe de mais tempo, refina-se a solução obtida.

O trabalho atual propõe uma extensão do algoritmo D* para ambientes multi-robôs, visando criar uma forma de colaboração entre os robôs e assim facilitar o processo de planejamento de caminhos. O algoritmo, denominado Space D*, combina o D* com o algoritmo de Colonização do Espaço, projetado originalmente para a simulação da modelagem de plantas [Runions et al. 2005], e posteriormente adaptado para a simulação do comportamento de multidões [Bicho 2009]. A principal característica do algoritmo de Colonização do Espaço é a preferência de deslocamento por espaços livres, o que evita o risco de colisão dado a incerteza e a escalabilidade de obstáculos fixos e móveis.

Por focar-se na criação de caminhos por ambientes com maior espaço livre, o método visa facilitar o controle dos robôs, apresentando-se de forma viável para a utilização em ambientes reais. Métodos que simplesmente buscam o menor caminho tendem a se aproximar dos obstáculos, e isto implica em reduções na velocidade dos robôs, e consequentemente, aumento no tempo total de execução (apesar de obterem menores distâncias totais percorridas). Por outro lado, alguns métodos que buscam ambientes livres, como campos potenciais, não garantem a obtenção de caminhos (caindo em mínimos locais), diferentemente do Space D*, que herda esta característica do D*.

Além disso, através da troca de informações, busca-se reduzir as distâncias per-

corridas por cada robô, visto que, ao adquirir conhecimento maior sobre o ambiente, o planejamento de caminhos torna-se mais preciso.

2. Trabalhos Relacionados

Grande parte dos trabalhos sobre planejamento de caminhos de múltiplos robôs, como [van den Berg et al. 2009] [Clark et al. 2003], baseiam-se em ambientes que devem estar totalmente determinados. Desta forma, a aplicação de técnicas como planejamento priorizado e coordenação de caminhos fixos em ambientes desconhecidos pode ser realizada desde que juntamente com alguma abordagem para tratamento de incerteza.

Por sua vez, os algoritmos de planejamento de caminhos em ambientes desconhecidos, como [Stentz 1995] [Koenig and Likhachev 2002] [Ferguson and Stentz 2005], são desenvolvidos para robôs independentes. No entanto, podem claramente ser aplicados em problemas com múltiplos robôs considerando os outros robos no cenário como obstáculos dinâmicos.

Porém, existem alguns trabalhos que se aproveitam da presença de múltiplos robôs com a capacidade de se comunicar e desenvolvem técnicas colaborativas para o planejamento de caminhos lidando com a incerteza. A grande maioria dos trabalhos de planejamento colaborativo de caminhos para múltiplos robôs situados em ambientes desconhecidos focam-se no problema de exploração do ambiente. Em [Rekleitis et al. 1997], o cenário é separado em faixas que são exploradas por times de robôs, sendo que dentro de cada time somente um robô se move de cada vez, visando reduzir erros de odometria. Em [López de Màntaras et al. 1997], robôs exploram aleatoriamente o ambiente e trocam informações sobre obstáculos quando se encontram.

Entretanto, existem alguns trabalhos que visam o planejamento de caminhos em si. Em [Pereira et al. 2003], utilizando funções de navegação, os robôs mantêm-se próximos uns dos outros para poderem trocar informações, enquanto rumam em direção aos seus objetivos. Em [Chen and Li 2005], robôs que precisam ir a um destino comum, são coordenados por um robô específico, enquanto os restantes responsabilizam-se por coletar as informações do ambiente. Em [Okada et al. 2007], é proposto um algoritmo baseado em *Roadmaps* Probabilísticos (PRM), onde os robôs possuem diferentes prioridades, requisitadas para prevenção de colisões, e são capazes de trocarem informações sobre suas trajetórias. Em [Pallottino et al. 2007], são utilizadas regras sociais para realizar o planejamento local em robôs, evitando a ocorrência de colisões de forma totalmente descentralizada. Já em [Jung et al. 2010], robôs com objetivos em comum trocam informações sobre o mapa quando se cruzam, para evitar que áreas sejam revisitadas.

3. Técnicas Utilizadas

O Space D* conforme mencionado anteriormente tem como base dois algoritmos que serão comentados brevemente a seguir: o primeiro é o D* e o segundo é o algoritmo de Colonização do Espaço.

3.1. D*

Conforme citado anteriormente, os algoritmos D* [Stentz 1995] e D* Lite [Koenig and Likhachev 2002] são uma das abordagens mais utilizadas e eficientes para

o problema de planejamento de caminhos em ambientes desconhecidos. De fato, atualmente, o D* Lite passou a ser utilizado em lugar do D* pela grande maioria dos pesquisadores, e é nele que o Space D* se baseia.

O comportamento básico de ambos algoritmos consiste em recalcular o menor caminho para o destino sempre que os custos de alguma célula variarem (algum obstáculo ou espaço livre for descoberto). Isto pode ser visualizado na Figura 1, onde a posição atual do robô é o nó marcado com um ponto, o destino é o nó marcado com um "x" e o caminho computado são os nós hachurados. Os nós preto representam obstáculos já descobertos.

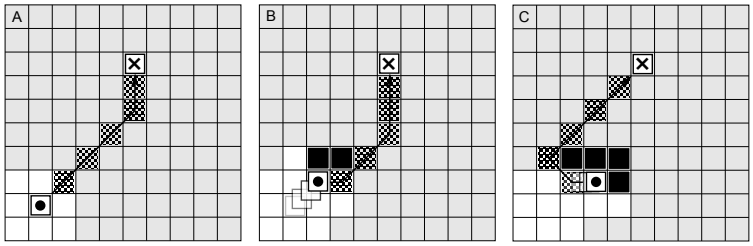

Figure 1. Funcionamento do D*/D* Lite. a)O robo sente o ambiente e calcula o caminho mínimo. b) Move-se para o próximo nodo no caminho, le os sensores, detecta um obstáculo e recalcula o caminho. c) Vai para a direita e executa o mesmo procedimento de (b).

O funcionamento do D* Lite é inspirado no A*, sendo que a cada atualização de uma célula, apenas as células que forem influenciadas por essa mudança serão recalculadas. Após esse processo, o menor caminho até o objetivo é calculado. Mais informações sobre o algoritmo podem ser encontradas em [Koenig and Likhachev 2002];

3.2. Algoritmo de Colonização do Espaço

Baseado em estudos da biologia, que atribuíram o desenvolvimento das nervuras a um hormônio chamado auxina presente nas folhas, o algoritmo de Colonização do Espaço foi desenvolvido em [Runions et al. 2005] para simular o crescimento das nervuras de uma folha em relação à distribuição das auxinas nesta folha.

O algoritmo é composto basicamente por três processos. O primeiro deles é o crescimento da lâmina da folha. Paralelamente a ele ocorrem os outros dois: o desenvolvimento do padrão de nervuras, que é influenciado diretamente pela distribuição das fontes de auxinas na folha (representando espaços livres), e a geração de novas auxinas, que por sua vez é afetada pelo desenvolvimento das nervuras.

O processo de crescimento da nervura começa através da associação de cada auxina ao nodo da nervura mais próximo. Em seguida, os nodos se expandem na direção das auxinas a que se associaram. Conforme o padrão cresce e se aproxima das auxinas, estas vão sendo removidas, visto que o espaço que elas representam deixa de ser livre. Depois, a lâmina da folha cresce e novas auxinas são geradas aleatoriamente, distribuídas de forma uniforme pelo espaço livre. Então, o ciclo recomeça com a associação auxinas-nodos e continua até a folha atingir o tamanho máximo. Como é possível perceber, o algoritmo é caracterizado como uma competição dos nodos da nervura pelas auxinas, representantes de espaços livres, visto que quanto mais auxinas um nodo consegue se associar, mais espaço ele tem para se expandir.

Em [Bicho 2009], o algoritmo de Colonização do Espaço foi adaptado para a simulação do comportamento de multidões. Os conceitos de nodos da nervura e auxinas foram transformados em agentes e marcadores de espaços livres, ou seja, o algoritmo de Colonização do Espaço passou a ser utilizado para simular a competição de agentes pelos espaços livres do ambiente.

Ainda em [Bicho 2009], algumas modificações sobre o algoritmo foram propostas. A primeira é que os marcadores se tornaram persistentes, isto é, não são mais "consumidos" pelos agentes, mas sim alocados quando algum agente se aproxima e liberados quando se afasta. Outra modificação é que apenas os marcadores dentro de certa região ao redor de um agente podem influenciá-lo (antes qualquer auxina podia influenciar qualquer nodo da folha, desde que este fosse o mais próximo dela). Uma terceira modificação é que a direção de movimento também passa a ser guiada pela intenção de se alcançar um destino específico e não apenas por grandes disponibilidades de marcadores. E uma última é que a velocidade de movimentação dos agentes passa a variar conforme o espaço livre que ele dispõe (no algoritmo original a velocidade era uniforme).

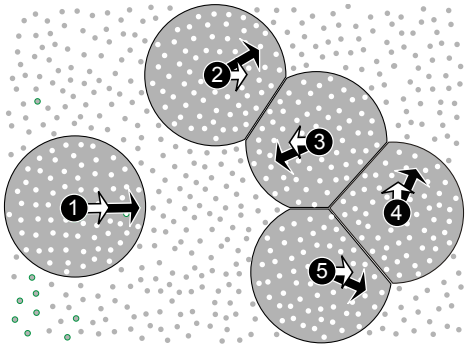

Figure 2. Algoritmo de Colonização do Espaço para simulação do comportamento de multidões.

Um exemplo de execução do algoritmo pode ser observado na Figura 2. Os agentes são os círculos pretos numerados e as setas brancas indicam a direção de seus destinos. As áreas maiores são as regiões de marcadores que cada agente foi capaz de alocar. Como se pode observar, as restrições nas regiões alocadas influenciam diretamente o vetor de movimento de cada agente, representado pela seta preta. Cada agente busca a direção do seu destino, mas evitando a proximidade com outros agentes ou obstáculos. Contudo, o algoritmo de Colonização do Espaço sozinho não resolve o problema de planejamento de caminhos em ambientes com obstáculos, mas a sua característica de busca por espaços livres motivou a mesclagem do algoritmo com o D* Lite para o planejamento de caminhos para múltiplos robôs em ambientes desconhecidos.

4. Space D*

O algoritmo desenvolvido neste trabalho para o planejamento de caminhos de múltiplos robôs em ambientes desconhecidos relaciona o algoritmo D* ao algoritmo de Colonização do Espaço e por esse motivo recebeu o nome de Space D*. O funcionamento resumido do método é o seguinte:

- O ambiente é representado por um *grid* formado por marcadores, que indicam os custos para um robô atravessá-lo (se está livre ou se é um obstáculo). No início, o

mapa é vazio e conforme obstáculos são detectados, o mapa é preenchido.

- O algoritmo D* Lite é aplicado por cada robô sobre o respectivo conjunto de marcadores, atualizando as distâncias para o destino de cada um;
- A navegação do robô é feita usando uma modificação do algoritmo de Colonização do Espaço. O robô aloca todos os marcadores livres dentro de sua área de alcance e com as distâncias para o destino desses marcadores calcula a sua direção de movimento, que sempre se dará dentro da zona alocada;
- Quando os robôs se cruzam, são trocadas informações sobre seus respectivos mapas, permitindo a atualização dos custos dos marcadores. Além disso, o algoritmo de Colonização do Espaço busca evitar colisões entre eles, pois cada robô aloca apenas os marcadores mais próximos de si;

Uma das principais contribuições do método proposto é a adição, ao D*, da preferência por ambientes mais espaçosos, o que conduz a uma robustez maior. Na sua definição original, o D* gera caminhos que tendem a passar muito perto de obstáculos, a fim de diminuir a distância total percorrida, no entanto, isso acaba exigindo um controle muito preciso na hora de executar a movimentação dos robôs.

Outra vantagem do método proposto é a troca de informações entre os robôs, enfoque de alguns trabalhos relacionados, como [Jung et al. 2010]. Como cada robô precisa mapear o ambiente por onde passa (para realizar o cálculo das distâncias para o destino), as informações sobre a presença de obstáculos podem ser trocadas quando robôs se encontrarem. Isto gera a possibilidade de se evitar caminhos que estejam bloqueados e que só futuramente o robô perceberia.

Contudo as mesmas restrições existentes no D*, continuam presentes no Space D*. Há a necessidade de discretização do mundo, podendo esta ter uma densidade variável(número de células no mapa). Por ser uma algoritmo de busca em grafo, quanto maior a densidade, mais demorada será a sua execução.

Outra importante restrição do método, é que, em sua atual implementação, a etapa de colonização do espaço considera os robôs como infinitesimais, ou seja, os seus corpos são considerados pontos. No entanto, para o correto funcionamento do algoritmo deve-se garantir que todos os robôs nunca saiam das respectivas regiões de marcadores alocados e que, conforme essas regiões diminuem, os seus deslocamentos também diminuam. Para robôs reais, o que o algoritmo garante é que o centróide do robô nunca sairá da região delimitada, porém isto pode ocasionar problemas quando a área alocada de um robô for muito pequena.

4.1. Funcionamento do Space D*

O Algoritmo 1 descreve o funcionamento do Space D*. À seguir cada uma das etapas é explicada.

Mapeamento do Ambiente: A etapa inicial do processo consiste no mapeamento do ambiente, onde cada robô percebe a presença de obstáculos e define os custos dos marcadores associados a cada posição.

Algoritmo 1: Space-D*

```
Mapeia o ambiente.
se Detectar Outros Robos então
    Troca informações com os outros robôs.
fim se
Atualiza os custos dos marcadores.
Calcula D* Lite.
Aloca os marcadores mais próximos.
Calcula a direção de movimento.
```

Detecção de outros robôs e troca de informações: Em seguida cada robô detecta a existência de robôs ao seu redor. Para isso ser possível cada robô possui um identificador único, o que faz com que quando um robô se aproximar de outro, ele possa perceber qeu o obstáculo é, na verdade, um robô e ainda qual robô. Quando dois robôs se encontrarem, ambos param por um momento e trocam informaços correspondentes às suas descrições do mapa.

Atualização dos custos dos marcadores: Após o mapeamento e troca de informações entre os robôs, deve-se atualizar os custos dos marcadores. Vale destacar que na atual implementação, todos os obstáculos são estáticos (excetuando-se os próprios robôs), assim a atualização dos custos dos marcadores consiste em verificar se no mapa recebido existem novos obstáculos que no mapa atual ainda não foram descobertos.

Cálculo do D* Lite: Na sequência são atualizadas as distâncias para o destino partindo de cada marcador de interesse do robô. Com as distâncias atualizadas calcula-se o menor caminho para o destino, segundo o algoritmo D* Lite [Koenig and Likhachev 2002].

Alocação de marcadores mais próximos Na próxima etapa, os robôs alocam os marcadores mais próximos de si. Esta etapa é descentralizada, pois cabe a cada robô alocar o conjunto dos marcadores situados dentro de seu alcance, considerando apenas aqueles que se encontram mais perto de si do que de outros robôs. Uma restrição reside sobre o tamanho do raio da região de alocação. Para que não ocorram colisões entre os robôs é necessário que o alcance do sensor de percepção de robôs seja sempre maior que o raio da região de marcadores alocados. O ideal é que o alcance do sensor seja no mínimo o dobro do raio de alocação, pois assim garante-se que no exato momento em que dois robôs se encontrarem, cada um estará alocando apenas os marcadores mais próximos de si.

Cálculo da direção de movimento Na última etapa, para calcular a sua direção de movimentação, cada robô utilizará as distâncias para o destino dos marcadores de sua região de interesse. O cálculo do vetor de movimento é uma soma vetorial ponderada dos vetores que ligam o robô a cada marcador associado. Porém, diferentemente de [Bicho 2009], onde o tamanho de cada vetor era relacionado à direção do destino, agora os módulos dos vetores baseiam-se na distância entre marcador e destino, de forma que

quanto menor a distância, maior deve ser o módulo do vetor. A definição da função f que determina o módulo do vetor de um marcador s associado a um robô r é:

$$f(s) = max_{p \in S(r)}(g(p)) - g(s)$$

Ou seja, o tamanho de cada vetor é a subtração da maior distância ($g(p)$) de um marcador pertencente ao conjunto $S(r)$ do robô, pelo valor de g de cada marcador.

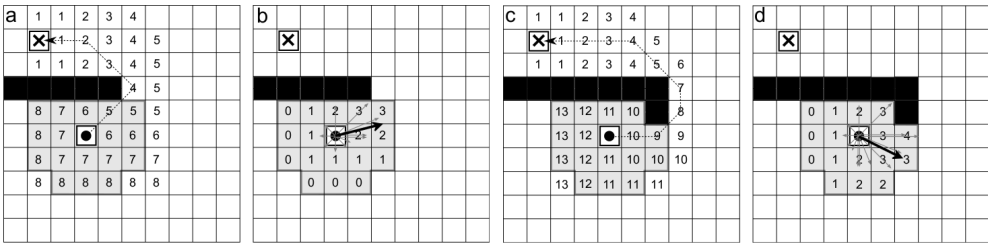

Figure 3. Exemplo de execução do Space D*. A subfigura (a) demonstra os passos 5 e 6 do processo. O passo 7 é mostrado na subfigura (b). (c) e (d) demonstram os mesmos passos na próxima iteração do algoritmo, depois de encontrar outros obstáculos.

Na Figura 3, pode ser observado um exemplo da execução do Space D*. A posição do robô é o nó indicado com um círculo, o destino é o nó indicado com um "x" e os obstáculos são os nós pretos. Em (a), a área demarcada representa a região alocada pelo robô, com as distâncias para o destino de cada marcador (em pontilhado está o menor caminhado calculado pelo D* Lite). Em (b), são gerados os vetores na direção dos marcadores com o módulo calculado por f, com os quais se obtém o vetor de movimento (vetor maior). Em (c), após deslocar-se na direção indicada anteriormente, o robô percebe novos obstáculos e recalcula as distâncias para o destino. Em (d), um novo vetor de movimento é gerado.

5. Testes e Validação

O algoritmo Space D* foi validado junto a um conjunto de testes simulados de ambientes multi-robôs nos softwares Player/Stage. Ambos fazem parte do Player Project [Gerkey et al. 2003]. Os mapas utilizados podem ser vistos na Figura 4.

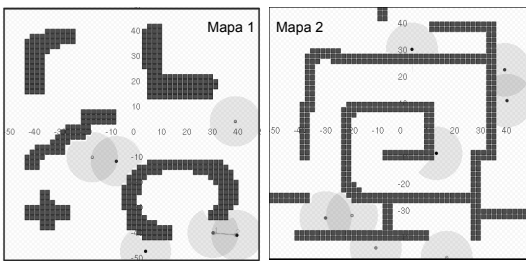

Figure 4. Mapas usados nos testes.

A validação foi realizada comparando-se as soluções obtidas na execução do Space D* com os resultados da execução do D* Lite. Verificou-se que, assim como o D* Lite, o Space D* gera trajetórias que sempre alcançam o destino sem a ocorrência de

colisões com obstáculos, mas além disso, conforme foi previsto, ele busca manter o robô afastado de qualquer obstáculo, seja dinâmico (outros robôs) ou estático (paredes).

Para o método utilizado para a comparação, a direção de movimentação dos robôs é a indicada pelo menor caminho computado pelo algoritmo D*. Como se pode deduzir, se não existirem erros e imprecisões no controle dos robôs, tal técnica funcionará perfeitamente. No entanto, muitas vezes ocorrem colisões ao se gerar caminhos próximos de obstáculos, a não ser que as velocidades de deslocamento dos robôs sejam reduzidas nessas localidades.

Na Figura 5 é mostrada uma comparação entre as trajetórias geradas na execução do Space D* e do D* no mapa 1 da Figura 4. São analisados os caminhos que três robôs dispersos aleatoriamente percorreram. É possível ver que as trajetórias obtidas com o Space D* passam longe o suficiente dos obstáculos, permitindo que a velocidade se mantenha próxima do limite estabelecido. Por outro lado, utilizando apenas o D*, a trajetória tangencia os obstáculos e, em virtude disso, em velocidade bem abaixo da máxima.

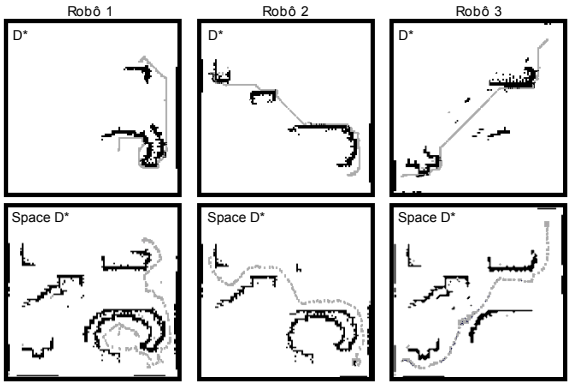

Figure 5. Comparação entre caminhos gerados pelo D* e pelo Space D* para três robôs. Os segmentos pretos são obstáculos e os cinzas são as trajetórias geradas pelos algoritmos.

Na tabela 1 foram comparados os tempos médios de execução, as distâncias totais médias e as velocidades médias dos robôs entre a simulação utilizando o Space D* e a simulação utilizando apenas o D*. Em todos os casos de teste o ambiente tinha 100x100 metros (e utilizou-se uma discretização de 100x100 marcadores) e a velocidade máxima dos robôs era de 5 m/s. Foram avaliados três conjuntos de teste (10, 15 e 20 robôs) cada um com cinco configurações aleatórias de posições iniciais e finais para cada robô.

É preciso ressaltar que em termos de tempo de execução de algoritmo, por efetuar um processamento maior, o Space D* é mais lento que o D* Lite. De fato, utilizando-se o D* Lite cada ciclo de simulação foi de aproximadamente metade do tempo do ciclo do Space D*. Contudo, por evitar a aproximação de obstáculos o Space D* permite uma taxa de atualização menor. Com isso, o que acaba influenciando majoritariamente os tempos de simulação são as velocidades médias de cada robô, associadas às distâncias percorridas.

O resultado obtido foi que com o Space D* o tempo médio de simulação se mostrou sempre menor (cerca de 30%), isto porque, apesar de o D* obter trajetórias

	Utilizando o Space D*					
Nº de Robôs	Tempo (s)		Distância (m)		Velocidade (m/s)	
	Média	DP	Média	DP	Média	DP
10	25.6	7.6	89.74	38.16	3.71	0.82
15	29.7	12.2	88.91	36.66	3.03	0.7
20	33.2	16.8	86.29	35.07	2.69	0.61
	Utilizando apenas o D*					
Nº de Robôs	Tempo (s)		Distância (m)		Velocidade (m/s)	
	Média	DP	Média	DP	Média	DP
10	33.2	10.3	83.21	29.61	2.57	0.64
15	42.4	16.0	85.68	31.07	2.13	0.58
20	50.9	21.8	86.91	33.48	1.74	0.39

Table 1. Comparação entre o Space D* e o método usando apenas o D*.

menores (como pode-se observar pelas distâncias médias), a velocidade média do Space D* é consideravelmente maior. Com isso, conclui-se que, em situações realistas, o uso do Space D* se mostrou mais aconselhável, visto que apesar de o D* tratar a incerteza sobre o espaço de configuração do ambiente, ele desconsidera a incerteza no controle dos robôs.

Na sequência, resolveu-se analisar se a existência de troca de informações entre os robôs contribui de forma significativa para o desempenho do Space D*. Para esse teste os dois mapas da Figura 4 foram utilizados, com três conjuntos de teste (10, 15 e 20 robôs) cada um com cinco configurações aleatórias de posições iniciais e finais para cada robô. Os resultados podem ser vistos na tabela 2. Para o primeiro mapa, a troca de informações não contribuiu para uma melhora no tempo de execução, pelo contrário, se mostrou menos eficiente que o método sem troca de informações para o exemplo com 10 robôs. No entanto, nos testes com 20 robôs, o método com trocas de informações obteve um melhor resultado. Entretanto, para o segundo mapa, o método com troca de informações permitiu que os robôs alcançassem o alvo de 10% à 30% mais rápido e com uma distância percorrida 20% a 50% menor.

O que pode-se inferir é que quando os obstáculos são esparsos e passíveis de serem contornados, a troca de informações não faz muita diferença. Porém, em um ambiente similar a um labirinto, a escolha do caminho a ser tomado, quando se alcança um obstáculo, é crítica. Quanto mais informações sobre o ambiente cada robô possuir, maior será a certeza na tomada dessa decisão e menor será a distância percorrida até o destino.

6. Conclusão

O algoritmo Space D* foca a geração de caminhos por ambientes espaçosos, mantendo o robô afastado de obstáculos dinâmicos e desconhecidos(paredes, obstáculos e outros robos. Dessa forma, o método se apresenta aplicável em situações realistas, onde o uso somente do D* não se mostra uma alternativa viável. Além disso, a troca de informações entre robôs apresenta ganhos no tempo de execução em ambientes onde os obstáculos formam trechos de caminhos sem saída.

Como trabalho futuro, existe a ideia de associar níveis de confiabilidade para os marcadores, bem como estender o algoritmo para o tratamento de obstáculos dinâmicos. Outra possibilidade é incluir no cálculo dos custos dos marcadores a concentração de

Mapa 1

Nº de Robôs	Método com Troca de Informações					
	Tempo (s)		Distância (m)		Velocidade (m/s)	
	Média	DP	Média	DP	Média	DP
10	25.6	7.6	89.74	38.16	3.71	0.82
15	29.7	12.2	88.91	36.66	3.03	0.7
20	33.2	16.8	86.29	35.07	2.69	0.61
	Método sem Troca de Informações					
Nº de Robôs	Tempo (s)		Distância (m)		Velocidade (m/s)	
	Média	DP	Média	DP	Média	DP
10	23.8	6.5	89.23	38.56	3.91	0.88
15	29.6	11.8	90.86	39.9	3.12	0.69
20	35.4	17.9	92.15	41.08	2.74	0.63

Mapa 2

Nº de Robôs	Método com Troca de Informações					
	Tempo (s)		Distância (m)		Velocidade (m/s)	
	Média	DP	Média	DP	Média	DP
10	37.9	18.3	126.26	63.71	3.43	0.74
15	39.4	19.0	113.05	52.12	2.89	0.66
20	40.3	19.7	99.74	44.48	2.41	0.42
	Método sem Troca de Informações					
Nº de Robôs	Tempo (s)		Distância (m)		Velocidade (m/s)	
	Média	DP	Média	DP	Média	DP
10	42.2	22.1	142.65	71.09	3.51	0.83
15	47.9	24.9	145.33	75.39	3.07	0.7
20	51.8	27.2	146.14	78.95	2.82	0.61

Table 2. Comparação entre os métodos com e sem troca de informações, para os mapas da Figura 4.

robôs em áreas do ambiente, de forma que ao trocar informações seja possível determinar os lugares com maiores concentrações de robôs, e, quem sabe, evitá-los. Além disso pode-se usar os vários robôs na realização de tarefas em cooperação.

References

Bicho, A. L. (2009). *Da modelagem de plantas à dinâmica de multidões: um modelo de animação comportamental bio-inspirado.* Doutorado em engenharia elétrica, Universidade Estadual de Campinas - UNICAMP.

Chen, J. and Li, L.-R. (2005). Path planning protocol for collaborative multi-robot systems. In *Proceedings 2005 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 721–726, Espoo, Finland.

Choset, H. and Burdick, J. W. (1994). Sensor-based planning and nonsmooth analysis. In *ICRA*, pages 3034–3041.

Clark, C. M., Rock, S. M., and Latombe, J.-C. (2003). Dynamic networks for motion planning in multi-robot space systems. In *In Intl. Symp. of Artificial Intelligence, Robotics and Automation in Space*.

Ferguson, D. and Stentz, A. T. (2005). Field d*: An interpolation-based path planner and replanner. In *Proceedings of the International Symposium on Robotics Research (ISRR)*.

Gerkey, B. P., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323.

Jung, J. H., Park, S., and Kim, S.-L. (2010). Multi-robot path finding with wireless multihop communications. *Comm. Mag.*, 48:126–132.

Koenig, S. and Likhachev, M. (2002). Improved fast replanning for robot navigation in unknown terrain. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pages 968 – 975 vol.1.

Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, U.K. Also available at http://planning.cs.uiuc.edu/.

Likhachev, M., Ferguson, D. I., Gordon, G. J., Stentz, A., and Thrun, S. (2005). Anytime dynamic a*: An anytime, replanning algorithm. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), June 5-10 2005, Monterey, California, USA*, pages 262–271. AAAI.

López de Màntaras, R., Amat, J., Esteva, F., López, M., and Sierra, C. (1997). Generation of unknown environment maps by cooperative low-cost robots. In *Proceedings of the first international conference on Autonomous agents*, AGENTS '97, pages 164–169, New York, NY, USA. ACM.

Okada, T., Beuran, R., Nakata, J., Tan, Y., and Shinoda, Y. (2007). Collaborative motion planning of autonomous robots. In *Proceedings of the 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 328–335, Washington, DC, USA. IEEE Computer Society.

Pallottino, L., Scordio, V. G., Frazzoli, E., and Bicchi, A. (2007). Decentralized cooperative policy for conflict resolution in multi-vehicle systems. *IEEE Trans. on Robotics*, 23(6):1170–1183.

Pereira, G. A. S., Das, A. K., Kumar, V., and Campos, M. F. M. (2003). Decentralized motion planning for multiple robots subject to sensing and communication constraints. In *in Proceedings of the Second MultiRobot Systems Workshop*, pages 267–278. Kluwer Academic Press.

Rekleitis, I. M., Dudek, G., and Milios, E. E. (1997). Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *Proceedings of the Fifteenth international joint conference on Artifical intelligence - Volume 2*, pages 1340–1345, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Runions, A., Fuhrer, M., Lane, B., Federl, P., Rolland-Lagan, A.-G., and Prusinkiewicz, P. (2005). Modeling and visualization of leaf venation patterns. *ACM Trans. Graph.*, 24(3):702–711.

Stentz, A. T. (1995). The focussed d* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

van den Berg, J., Snoeyink, J., Lin, M., and Manocha, D. (2009). Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Proceedings of Robotics: Science and Systems*, Seattle, USA.

van den Berg, J. P., Ferguson, D., and Kuffner, J. (2006). Anytime path planning and replanning in dynamic environments. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 15-19, 2006, Orlando, Florida, USA*, pages 2366–2371.