Relatório 2

Wallace Mielniczki[1]

[1]Centro de ciências tecnológicas – Universidade do Estado de Santa Catarina (UDESC)
Caixa Postal 631 – 89.219-710 – Joinville – SC – Brazil

Abstract. *This report will demonstrate techniques, analysis, results and conclusion in the implementation of a robot that aims to navigate in an environment collecting parts to repair several factories.*

Resumo. *Este relatório irá demonstrar técnicas, análise, resultados e conclusão na implementação de um robô que tem como objetivo navegar em um ambiente coletando peças para consertar várias fabricas.*

1. Introdução

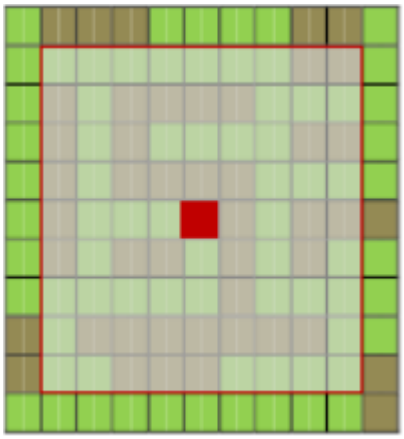
O trabalho consiste em implementar um sistema de navegação automática de um robô que realiza diferentes manutenções em unidades fabris. Para isto, deve-se utilizar o algoritmo de busca A* para o cálculo do custo da rota.

O robô tem a informação da posição das fábricas a serem realizadas as manutenções, mas está desprovido de ferramentas para realizá-las. As ferramentas estão dispersas no ambiente e o robô deve capturá-las utilizando seu radar para poder se dirigir para a fábrica. Ou seja, o agente não tem a informação da localidade das ferramentas e só pode realizar a manutenção se possuir a ferramenta necessária.

Existem 5 tipos de ferramentas dispersas no ambiente:

- Bateria de carga elétrica;
- Braço de solda;
- Bomba de sucção;
- Dispositivo de refrigeração;
- Braço pneumático;

A posição inicial das ferramentas é desconhecida. O agente deve utilizar seu radar para localizá-las. O radar possui um alcance máximo de 4 regiões adjacentes em todas as direções.



O total de cada tipo de ferramenta é o seguinte:

- 20 baterias de carga elétrica;
- 10 braços de solda;
- 8 bombas de sucção;
- 6 dispositivos de refrigeração;
- 4 braços pneumáticos.

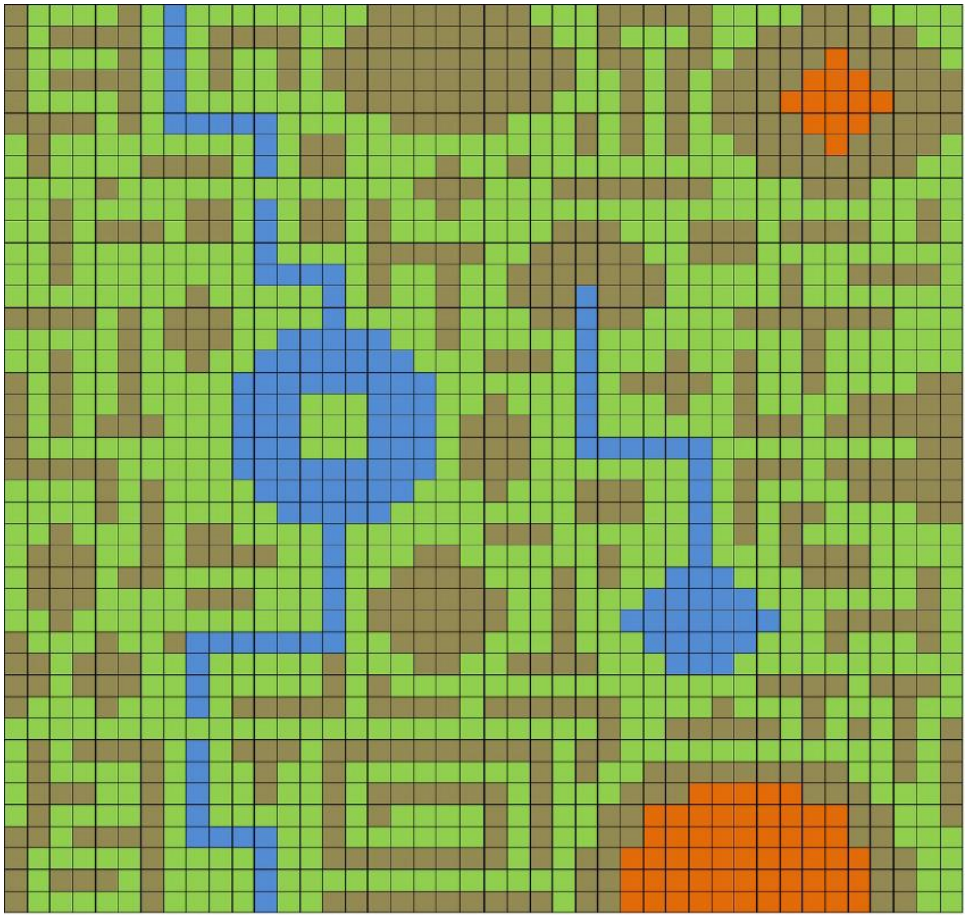
Existem 5 tipos de fábricas que necessitam manutenção e que se relacionam com as ferramentas:

- Indústria de melhoramento genético de grãos com falta de energia elétrica nas incubadoras. Necessita de 8 baterias;
- Empresa de manutenção de cascos de embarcações. Necessita de 5 braços de solda;
- Indústria petrolífera com dutos entupidos. Necessita de 2 bombas;
- Fábrica de fundição com superaquecimento nas caldeiras. Necessita de 5 refrigeradores;
- Indústria de vigas de aço com falta de braços mecânicos para moldagem. Necessita de 2 braços pneumáticos.

O ambiente por onde o robô vai navegar, representado através de uma matriz n x n, é formado por diversos tipos de terrenos e em cada tipo de terreno o robô tem um grau de dificuldade diferente para andar. A Figura 1 mostra o ambiente a ser explorado pelo robô. Por exemplo, o robô consegue passar facilmente por um terreno sólido e plano, porém terá dificuldades para andar em um terreno montanhoso ou em um pântano.

Os tipos de terrenos que compõem o ambiente são (ver cores na Figura 2):

- Sólido e plano (verde) – Custo: 1
- Montanhoso (marrom) – Custo: 5
- Pântano (azul) – Custo: 10
- Árido (vermelho) – Custo: 15
- Obstáculo (preto) – Intransponível



2. Metodologia e Desenvolvimento

Primeiramente comecei analisando qual seria a ação do robô quando nenhuma peça estivesse em seu radar e não desse para consertar nenhuma fábrica. Pensei em algumas possibilidades: (1) Andar aleatoriamente, (2) Se movimentar somente pelos campos verdes para se ter menor custo, (3) ter uma rota padrão para cobrir todo o mapa pelo radar.

Se movimentar somente pelos campos verdes não cobriria com o radar todos os campos que conteriam as peças e o robô ficaria procurando essas peças para sempre, então podemos descartar essa opção. Ter uma rota padrão meio que quebraria a ideia de o robô funcionar para qualquer ambiente, pois a rota padrão seria sob medida para aquele ambiente em específico. Então escolhi andar aleatoriamente quando o robô não tiver nenhum objetivo. O problema dessa abordagem é que tem a possibilidade do custo do caminho feito pelo robô ser altíssimo, então vamos ter que planejar bem o algoritmo de rotas e tomada de decisões para que isso não aconteça.

Quando o robô encontra uma ou várias peças em seu radar ele deve verificar antes se ele precisa delas para consertar uma fábrica, se a fábrica correspondente a peça encontrada no radar já foi consertada ou eu já peguei a quantidade necessária daquela

peça para consertar a fábrica não tem o porquê eu ir até ela, então simplesmente a ignoramos, assim diminuímos processamento e custo de rota.

Devemos também desenvolver algumas regras de prioridades entre os objetivos. Primeiramente pensei em simplesmente andar aleatoriamente pelo mapa, e quando fosse detectado uma peça no radar eu iria até ela, pegava elas sem uma ordem específica e se pudesse consertar uma fábrica iria até a fábrica e a consertava. Isso funciona, mas na maioria dos casos o robô andava aleatoriamente pelo mapa sem um objetivo, causando processamento e custo de rota alto. Para melhorar o desempenho do robô, quando ele estivesse caminhando em direção a um objetivo, a cada passo verificar ao redor se tem peças para ele pegar, por exemplo se o robô tivesse que se deslocar de uma ponta a outra do mapa para consertar uma fábrica ele não iria diretamente até ela, ele iria no caminho pegando outras peças necessárias que aparecessem em seu radar durante o trajeto, dessa forma as chances de andar aleatoriamente pelo mapa são reduzidas. Mas aí um outro problema surge, quando eu tenho um objetivo principal e encontro outras peças no radar, qual a melhor ordem e rota deve se seguir para minimizar os custos?

Minha primeira abordagem foi sempre que era detectado uma ou mais peças no caminho eu ia da posição que eu estava até ela e voltava para a rota principal, mas essa abordagem não é a melhor, primeiro porque eu posso ter outro campo da minha rota principal que me dê um caminho menos custoso até a peça, outra que o processo de ir e voltar gera alto processamento e custo de rota. Mas para eu encontrar a melhor rota até uma peça partindo de algum campo do caminho principal eu teria que aplicar o algoritmo A* de cada campo do caminho principal até a peça que eu detectei, isso geraria um enorme custo de processamento.

A abordagem que utilizei e que deu ótimos resultados foi, sempre que eu tiver um objetivo principal eu vou até ele, se no caminho eu encontrar alguma peça necessária eu vou imediatamente até ela e após pegá-la, recalculo a rota para meu objetivo principal. Por isso a utilização de uma pilha de objetivos, toda vez que eu encontro uma peça ou posso consertar uma fábrica em adiciono no topo da pilha e vou em direção ao objetivo do topo. Isso gera um efeito bola de neve pois as peças não estão muito distantes uma das outras e na maioria dos casos quando eu vou em direção a uma peça eu encontro outra, e se as fabricas estiverem bem espalhadas isso ajuda ainda mais, pois no caminho as chances de eu encontrar peças são grandes. Assim as chances de eu ficar sem objetivo são diminuídas.

3. Descrição de Experimentos/Simulações e Resultados Obtidos

Vamos comparar os resultados obtidos com a utilização de outros dois algoritmos de busca, o algoritmo de busca gulosa e de custo uniforma. Dentre esses algoritmos vamos analisar o custo de rota e a quantidade de nós visitados por eles.

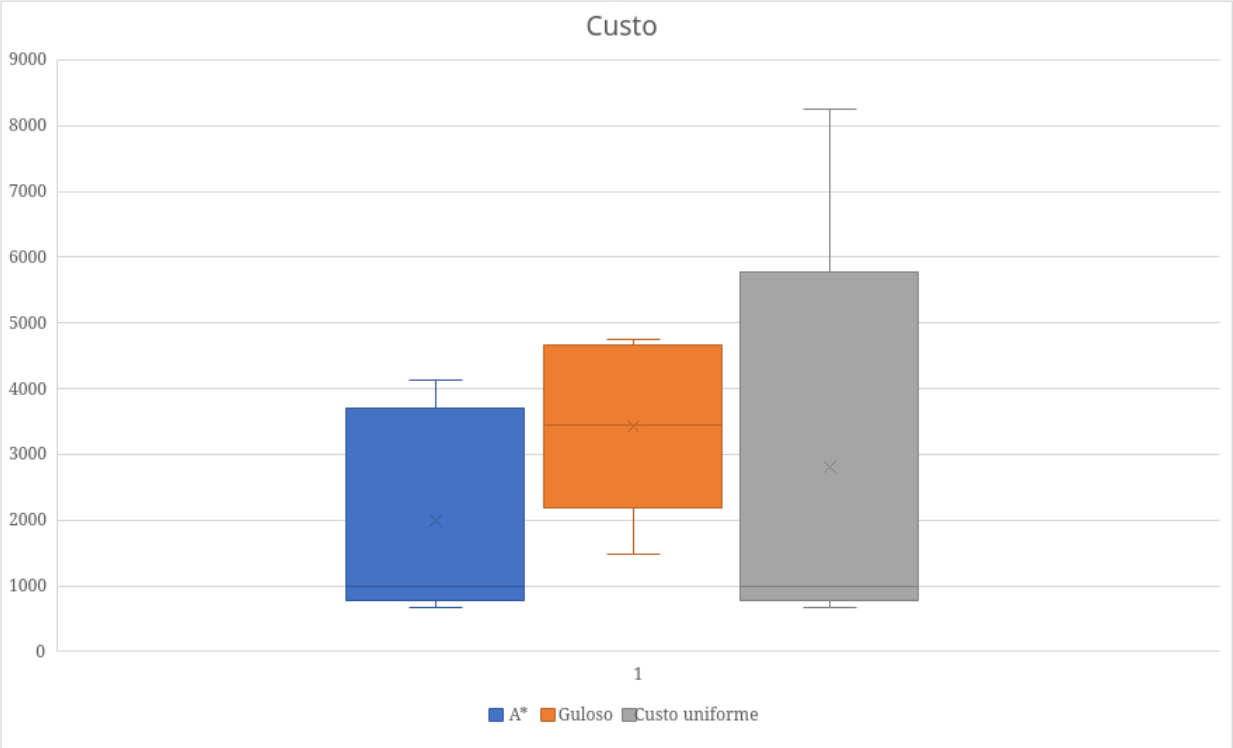
Com isso obtemos os seguintes dados:

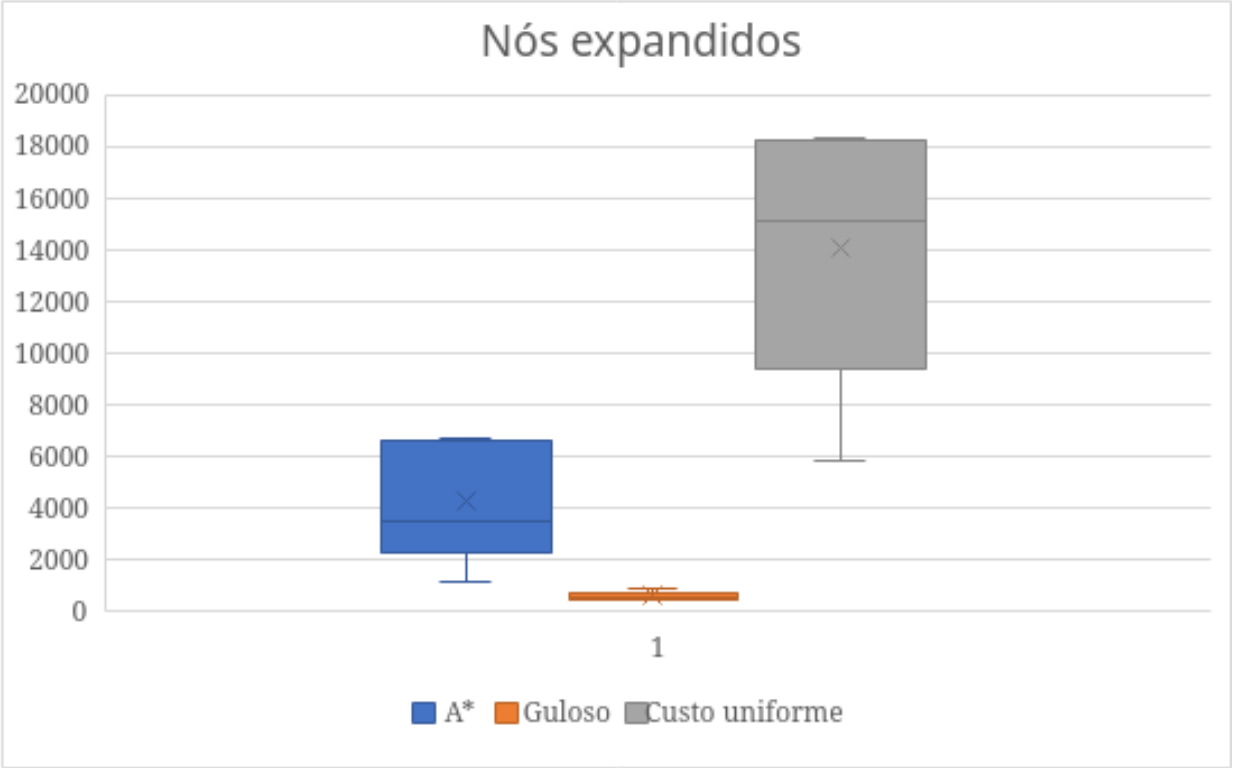
A*	
custo	Nós expandidos

670	3509
868	3350
4126	1183
3284	6529
995	6687
Média: 1988.6	Média: 4251.6
Desvio padrão: 1599.0	Desvio padrão: 2339.7

Guloso	
custo	Nós expandidos
1476	915
3447	560
4743	432
4593	631
2882	543
Média: 3428;2	Média: 616.2
Desvio padrão: 1341.1	Desvio padrão: 181.6

Custo uniforme	
custo	Nós expandidos
670	15175
868	12876
8259	5886
3284	18368
995	18168
Média: 2815.2	Média: 14094.6
Desvio padrão: 3223.4	Desvio padrão: 5120





4. Análise dos resultados obtidos

Analisando os gráficos de box-plot percebemos que o algoritmo guloso tem uma grande vantagem na quantidade de nós expandidos comparado aos demais, mas ele não nos traz um ótimo caminho. Já o algoritmo A* percorre menos nós

comparado ao algoritmo de custo uniforme, ou seja, é mais rápido, e os custos de caminho do A* são iguais ou menores que o algoritmo de custo uniforme.

5. Conclusões e Trabalhos Futuros

Percebemos que o algoritmo A* nos traz um ótimo caminho com custo mediano de processamento comparado aos demais algoritmos analisados nesse trabalho, portanto para este caso ele é a melhor escolha.

Uma ideia de trabalho futuro seria o agente mapear os campos que já foram detectados e quando não tiver objetivo explorar os campos que ainda não foram analisados, com isso ele não precisaria andar aleatoriamente e com certeza daria um custo de caminho muito menor.

Referências