O Mundo do Aspirador de Pó

Elvis Herandes Ribeiro Jan Pierre Agenciano da Silva Rocha
Lucas Leite de Oliveira

Outubro 2018

Resumo

Esse trabalho tem como objetivo simular um mundo de aspirador de pó melhor do que foi mostrado para nós em sala de aula, e para isso usamos de alguns artifícios da Inteligência Artificial como uma busca baseada no algoritmo de Dijkstra, para achar o melhor caminho possível para recolher o máximo de sujeira, e métodos para simular a sujeira em uma sala que ficasse o mais próximo de realidade possível.

Sumário

1 Introdução

O mundo do aspirador de pó estudado em sala, consiste em um robô que não conhece nada mais do que seu estado atual e se esse espaço possui sujeira, e a partir disso ele tem apenas três ações para ser feito, 2 de movimentação, que é mover-se para direita ou esquerda e a opção de sugar caso no lugar exista alguma sujeira.

Na nossa adaptação além da opção de sugar, o robô pode ter 8 opções de movimento, e aprendizado, que através de um algoritmo de busca, define quais os lugares de uma sala sujam mais e que devem ser limpos com uma frequência maior do que os outros lugares, podendo fazer uma adaptação nesse caminho caso a sujeira repentinamente mude de lugar.

2 Desenvolvimento

Nesse tópico vamos falar um pouco mais de pontos críticos e que são importantes para a nossa simulação.

2.1 Ambiente

Um ambiente em inteligência artificial significa o lugar em que o agente vai atuar.

O nosso ambiente é representado por duas salas, uma representa a sala real que é o que realmente vemos e que realmente possui o valor da sujeira, e a outra sala que chamamos de sala virtual que representa uma estimativa da porcentagem de sujeira que pode ter em cada lugar.

Para representarmos mais especificamente cada lugar, é preciso uma abstração melhor do que é cada sala, então para isso, dividimos as salas em quadrantes, pois a partir de agora podemos ter dados mais preciso de cada lugar no ambiente.

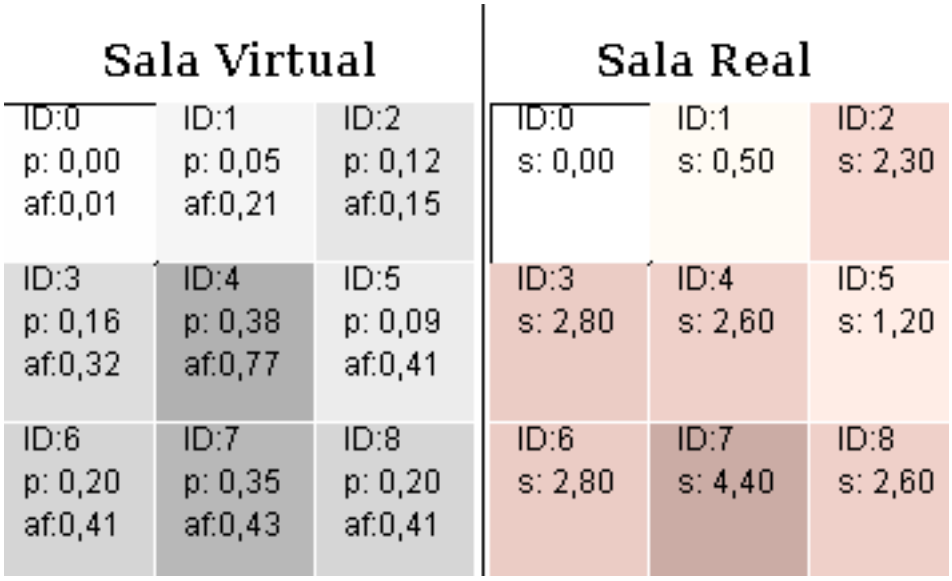

Figura 1: À esquerda, a estimativa que o agente faz sobre a sala. À direita, a sala real.

Cada quadrante possui seus atributos, que se difere para cada tipo de sala. No caso da sala real, cada quadrante possui o seu Id, a sujeira que possui no chão, e um multiplicador de sujeira, que é utilizado para incrementar a sujeira a cada espaço de tempo e simular

o acumulo em um determinado local. Já na sala virtual, eles possuem a porcentagem de sujeira e alguns outros atributos que são usados para que o robô utilize na busca de um caminho ideal de limpeza, o que gera a aprendizagem dele.

2.2 Agente

Um Agente é tudo que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por meio de atuadores. No projeto, o robô é o nosso agente e ele usa alguns sensores para saber o que acontece no ambiente, e de acordo com o que esses sensores fazem, as ações são tomadas pelos atuadores que agem no ambiente.

2.2.1 Sensores

Os sensores são uados para que o agente se comunique com o ambiente. Nós utilizamos dois tipos de sensores, o HPS (Home Positioning System), que ajuda o agente a se localizar no ambiente normalmente, e um sensor de sujeira, que é responsável por saber o quanto de sujeira ele coletou em cada quadrante que ele passa da sala real.

2.2.2 Atuadores

Os atuadores são os responsáveis por receber determinada ação e colocar em prática então, como atuadores nós possuímos as rodinhas, pois são elas que possibilitam o robô se mover no ambiente nas posições definidas na imagem 1.0, e a ação de sugar uma sujeira de um determinado local, que é de muita importância para o objetivo do robô.

2.3 Função do Agente

A função do agente define o comportamento do agente em determinada situação, ela mapeia qualquer sequência de percepções específica para uma ação, e é isso que a nossa busca faz.

2.4 Programa do Agente

Os programas de agentes é o responsável por implementar a função do agente e ele tem como entrada somente a percepção atual, como a entrada dos sensores, e é dele que se espera a ação que será feita e passada para que os atuadores executem.

Para isso nós aplicamos um algoritmo de busca, que tende a achar um caminho ideal para coletar sujeira que deve ser percorrido toda vez pelo robô.

2.4.1 Busca

A nossa busca é baseada em um algoritmo que é aplicado em grafos chamado Dijkstra, então para isso a primeira coisa que precisamos fazer é transformar a matriz que possuímos e que simboliza uma sala em um grafo direcionado, onde todos os quadrantes adjacentes são vizinhos dele, e a porcentagem de ter sujeira no lugar é o peso da aresta.

O algoritmo de Dijktra basicamente nos retorna a distância do quadrante atual para todas os outros quadrantes, então precisamos calcular o inverso da porcentagem já que queremos o caminho mais sujo, sendo assim ele sabe o caminho que deve ser percorrido

para chegar em um determinado lugar e que tenha o maior número de sujeira, então basta o robô percorrer esse caminho e ir limpado todas as casas que ele passar. Acontece que a medida que o algoritmo vai rodando, o robô começa a aprender um caminho que tende ao ótimo, graças a aprendizagem que implementamos.

2.4.2 Método de Penalidade

A penalidade é algo importante pois graças a ela podemos quantificar se o caminho que o robô percorreu foi bom ou ruim.

No nosso projeto, a primeira coisa que o robô faz é limpar a sala completamente para que podemos incluir o α inicial em cada quadrante da sala virtual, a partir daí o robô começa a executar a busca e vai percorrer vários caminhos coletando sujeira, à medida que o robô coleta a sujeira na sala real, nós verificamos se o tanto de sujeira que nós coletamos correspondia a sujeira que a gente esperava que tivesse naquele lugar.

Se o valor for próximo, nós não ajustamos a porcentagem, agora caso a sujeira tenha uma discrepância muito grande do esperado, nós aumentamos o αda sujeira naquele lugar, sendo assim o nosso robô vai começar a passar mais vezes no quadrante, e vai adaptando de acordo com o tempo.

2.5 Aprendizado

O aprendizado na inteligência artificial é algo muito importante, pois todo agente racional precisa aprender a partir do que ele percebe para que ele possa ter uma autonomia e compensar um conhecimento prévio e incorreto.

Sendo assim o nosso robô possui um aprendizado, graças a abstração de sala que construímos no tópico anterior, pois ele não pode acessar a sala real, somente a sala virtual, então é através da sala que calculamos uma probabilidade que chamamos de α, é ela que define pra qual lugar o robô vai, dentro da sala virtual, e esse α ele vai adaptando a sala real, até que chega um ponto que o robô será capaz de achar o caminho ótimo para limpar a sala.

3 Execução e Testes

Para simular o desempenho do robô ao longo do tempo, assumimos que o ambiente está sempre se sujando. O objetivo é que no infinito ele consiga adaptar a sala virtual de modo que fique idêntica à sala real.

Como é difícil analisar as duas salas e definir uma norma que calcule o quanto uma sala está distante da outra, usamos um fator de comparação que é dado pela seguinte razão:

$$\frac{sujeiraColetada}{distanciaPercorrida} \tag{1}$$

, ou seja, a sujeira coletada influencia positivamente na pontuação e a distância percorrida, negativamente.

Para ilustrar a eficiencia do agente com a Busca citada na seção 2.4.1, comparemos com a busca Subida de Encosta.

Ambas as buscas começarão com o mesmo cenário retratado na figura 3

Com um teste de 1 minuto obtemos os seguintes resultados:

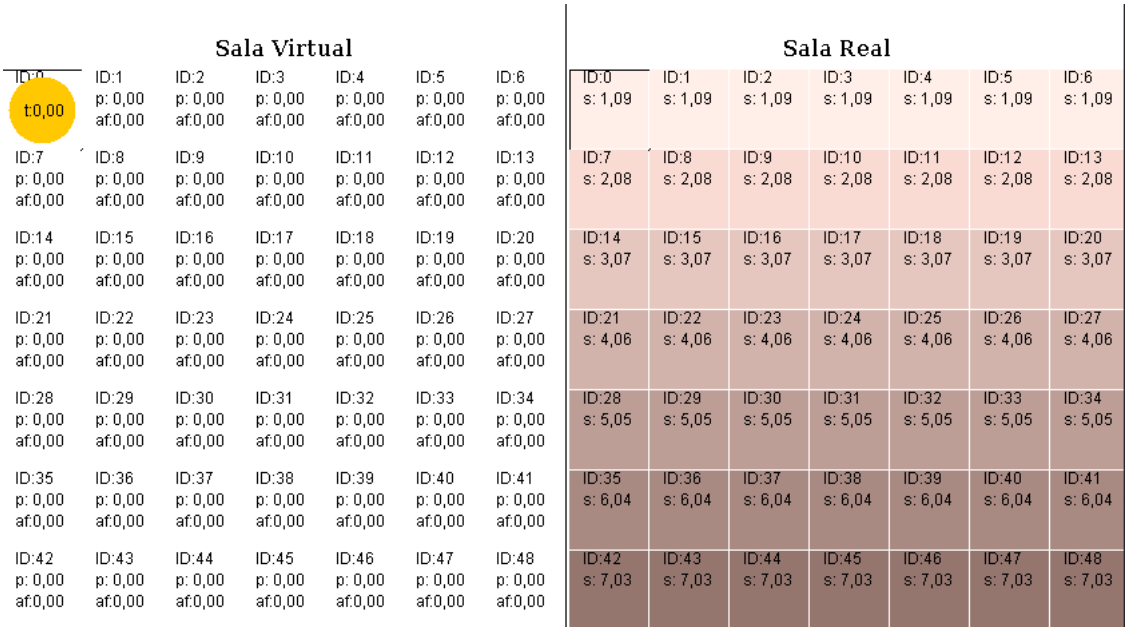

Figura 2: Cenário inicial para ambas as buscas.

Tempo(s)	Rendimento(fórmula 1)	
	Dijkstra	Subida de Encosta
10	0.53170	0.43959
20	0.74868	0.80468
30	0.79132	0.73555
40	0.87517	0.67400
50	0.91933	0.81358
60	0.93009	0.76911
70	0.94247	0.86631
80	0.93870	0.81006
90	0.94807	0.91759
100	0.96556	0.87719

Tabela 1: Evolução dos dois algoritmos ao decorrer do tempo.

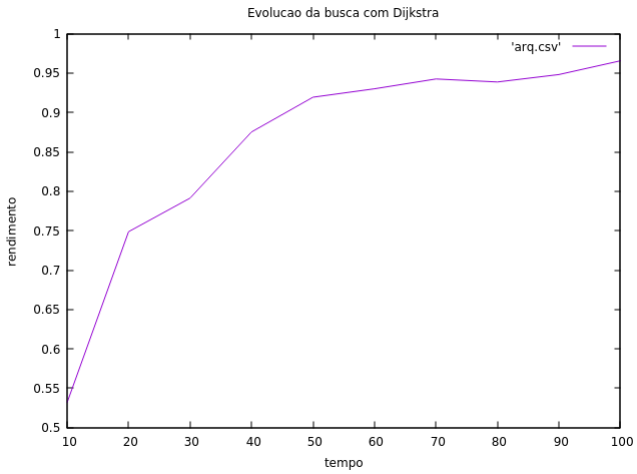

Grafico 1:Evolução da busca com Dijkstra ao decorrer do tempo.

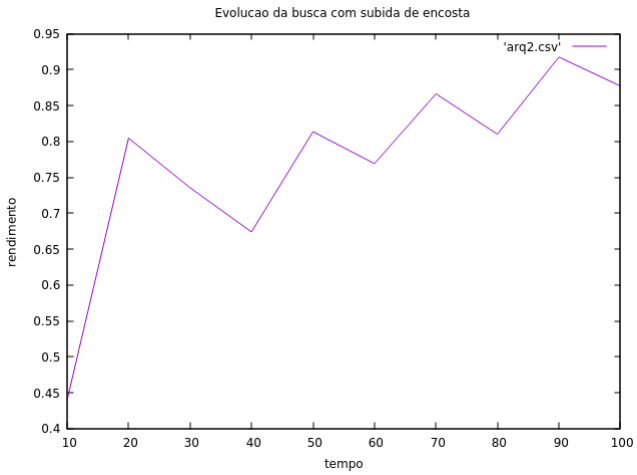

Grafico 2:Evolução da busca com subida de encosta ao decorrer do tempo.

Podemos observar pela tabela e pelos gráficos que a busca com Dijkstra está em aprendizado linear, enquanto a busca em subida de encosta é um tanto quanto imprevisível. Com um coeficiente de rendimento melhor, a busca com Dijkstra percorre menos espaço e aspira uma quantidade de sujeira maior.

4 Conclusão

Depois de construir o nosso projeto e visualizar o comportamento do robô concluímos que o método de busca usado é válido (como comprovado no gráfico 1) e que ele representa bem o comportamento do robô em um mundo real, sendo assim melhor do que o proposto pelo livro.