Trabalho 1 - Algoritmos de Busca

Inteligência Artificial - 2024.2

Prof. Samy Sá

Universidade Federal do Ceará
Campus de Quixadá

Obs.: Para completar este trabalho, será necessário implementar alguns algoritmos, experimentar extensivamente com eles e escrever um relatório com suas conclusões. Em caso de plágio em qualquer parte deste trabalho, os envolvidos terão suas notas automaticamente zeradas. Em caso de reincidência, os envolvidos serão imediatamente reprovados na disciplina com nota 0,0 e 64 faltas.

1 Introdução

Este trabalho propoõe a aplicação dos principais algoritmos de busca utilizados para planejamento de agentes em Inteligência Artificial. Nosso problema envolve uma cidade fictícia com alguns pontos de interesse (universidade, a casa do agente, farmácias, bibliotecas, academias, etc.) e um agente que precisa navegá-la, sempre buscando os caminhos de *menor custo*. O objetivo deste trabalho é explorar variações do problema de *planejamento com menor custo* e estudar como os diferentes algoritmos que estudamos se comportam nestas variações.

2 A Cidade (Ambiente) e o Agente

Ambiente. Consideraremos uma cidade cujo mapa consiste em grade com ruas nos sentidos horizontal e vertical seguindo as coordenadas inteiras x, y no intervalo $[0, 30]$ (números naturais até 30, portanto). Por consequência, os cruzamentos destas ruas correspondem a alguns dos pontos no quadrante superior direito do plano cartesiano com coordenadas em números naturais. Por simplicidade, consideraremos que todos os endereços relevantes estão nos cruzamentos das vias, ou seja, nestes pontos de coordenadas inteiras.

Agente. Consideraremos um agente simples que planeja suas ações antes de começar a executá-las. Neste caso, as ações do agente compreendem apenas movimentos entre coordenadas vizinhas na cidade, de forma que cada plano do agente corresponde a uma rota entre dois pontos da cidade. Dado o formato da cidade, as ações do agente correspondem, portanto, às operações

$$f_1(x, y) = (x - 1, y)$$
$$f_2(x, y) = (x + 1, y)$$
$$f_3(x, y) = (x, y - 1)$$
$$f_4(x, y) = (x, y + 1)$$

Mais à frente, nos referiremos a estes operadores como

$$(-1, +0), (+1, +0), (+0, -1), (+0, +1).$$

Por exemplo, se agente iniciar uma rota no ponto $(3, 4)$, após executar a ação f_4, ele chegará ao ponto $(3 + 0, 4 + 1) = (3, 5)$. Na sequência, se o agente executar a ação f_1, ele chegará ao ponto $(3-1, 5+0) = (2, 5)$. Lembre-se ainda que as coordenadas podem limitar a aplicação de alguns operadores, já que todas as coordenadas estão limitadas aos inteiros no intervalo $[0, 30]$.

3 Planejamento Como Busca

Estados do Espaço de Busca. Num problema de navegação em mapas, os estados do espaço de busca correspondem às coordenadas que podem ser alcançadas. Neste caso, temos infinitos estados correspondendo aos diferentes pares de coordenadas naturais do plano cartesiano. Na descrição do mapa, alguns destes pontos podem receber algum rótulo indicando o tipo de endereço. Entre as opções, admitiremos alguns rótulos únicos, tais como *casa*, *trabalho*, *ufc*, e alguns rótulos gerais, tais como *farmácia*, *mercado*, *academia*, *posto de gasolina*, etc. Desta forma, os estados do espaço de busca são pares de números naturais acompanhados de um rótulo opcional (uma string).

Vizinhança Entre Estados no Espaço de Busca. Considerando as ações disponíveis ao agente, dos estados são vizinhos se eles compartilham o mesmo valor em uma de suas coordenadas e a diferença na outra coordenada é de apenas uma unidade. Por exemplo, os vizinhos do estado na coordenada $(1, 3)$ são $f_1(1, 3) = (0, 3)$, $f_2(1, 3) = (2, 3)$, $f_3(1, 3) = (1, 2)$, e $f_4(1, 3) = (1, 4)$.

Custos das Ações. Alguns dos algoritmos que pretendemos explorar assumem que as ações do agente têm custos diferentes. Por esta razão, consideraremos alguns cenários com funções de custo que avaliam ações no sentido horizontal e vertical do mapa de formas diferentes:

C1. Todas as ações (f_1, f_2, f_3, f_4) têm custo 10, independente da direção.

C2. Ações na vertical (f_3, f_4) têm custo 10 e ações na horizontal (f_1, f_2) têm custo 15.

C3. Ações na vertical (f_3, f_4) têm custo 10 e ações na horizontal (f_1, f_2) têm custo que varia com o tempo de trajeto segundo a função

$$c_3(t) = 10 + (|5 - t| \textbf{ mod } 6),$$

onde t é o número passos (arestas) no caminho da raiz da árvore de busca até o estado que está sendo avaliado.

C4. Ações na vertical (f_3, f_4) têm custo 10 e ações na horizontal (f_1, f_2) têm custo que varia com o tempo de trajeto segundo a função

$$c_4(t) = 5 + (|10 - t| \textbf{ mod } 11),$$

onde t é o número de passos (arestas) no caminho da raiz da árvore de busca até o estado que está sendo avaliado.

Estes valores servirão para simular condições de tráfego ou más condições em diferentes trechos das vias.

Obs.: Na implementação da estrutura de *nó* da árvore de busca, convém adicionar um atributo *profundidade* para simular a passagem do tempo e facilitar os cálculos dos custos de ações nos cenários **C3** e **C4**.

Algoritmos. Trabalharemos com as principais variações do Algoritmo de Melhor Primeiro:

A1. Busca em Largura
A2. Busca em Profundidade
A3. Busca de Custo Uniforme (Algoritmo de Dijkstra)
A4. Busca Gulosa
A5. Busca A*

Todos os algoritmos foram discutidos em sala e são abordados em profundidade no Capítulo 3 do livro-texto.

Funções Heurísticas. Alguns dos algoritmos listados dependem de funções heurísticas para operar conforme esperado, em particular a Busca Gulosa e a Busca A*. Nossas heurísticas serão baseadas em duas medidas comumente usadas para esse tipo de problema:

- Distância Euclidiana: $\texttt{euc}((x_1, y_1), (x_2, y_2)) = \sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2}$
- Distância Manhattan: $\texttt{man}((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$

Ambas as medidas são baseadas em resultados de geometria. A distância euclidiana refere-se à menor distância entre dois pontos em um plano, capturanado a distância em linha reta entre esses pontos. Corresponde à medida da hipotenusa em um triângulo retângulo formado pelos pontos $(x_1, y_1), (x_2, y_2)$ e um terceiro ponto que pode ser escolhido entre (x_1, y_2) ou (x_2, y_1). A somas dos catetos deste triângulo é o que nos dá a distância manhatthan. Note que distâncias são medidas não-negativas, daí a necessidade de utilizar o módulo numérico nas diferenças entre as coordenadas.[1]

Multiplicaremos os valores destas funções por 10 devido à maneira como construímos as funções de custo na seção anterior. Dessa forma, obtemos as heurísticas

H1. $10 \times \texttt{euc}((x_1, y_1), (x_2, y_2)) = 10 \times \sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2}$
H2. $10 \times \texttt{man}((x_1, y_1), (x_2, y_2)) = 10 \times (|x_1 - x_2| + |y_1 - y_2|)$

Note que, dependendo da função de custo, as heurísticas podem ser ou não admissíveis.

[1] Numa situação em que $x_1 = x_2$ ou $y_1 = y_2$, um dos catetos tem medida nula e o triângulo colapsa em um segmento de reta. Os cálculos continuam funcionando nestes casos.

4 Implementação

Inicialmente, você deve implementar funções distintas para cada algoritmo (**A1**, **A2**, **A3**, **A4**, **A5**), função de custo (f_1, f_2, f_3, f_4) e função heurística (**H1**, **H2**) conforme propostos acima.

Depois organize chamadas que serão utéis para os experimentos pedidos. É desejado que você possa executar com facilidade qualquer cenário envolvendo um algoritmo, uma função de custo e uma heurística (quando aplicável) com parâmetros de entrada diferentes. A lista completa de cenários dos experimentos será fornecida na Seção 5.

É necessário que possamos executar qualquer experimento manualmente, escolhendo os parâmetros que passaremos como entrada para testes.

Saídas. Tenha em mente que cada combinação de parâmetros deverá ser executada várias vezes, possivelmente em um loop no seu próprio código. A cada vez que rodar um destes algoritmo, indique na saída os valores de

1. Estado Inicial
2. Objetivo da Busca
3. Caminho Retornado (ou um código de erro, caso se aplique)
4. Custo do Caminho Encontrado ($+\infty$ ou `null`, em caso de erro)
5. Quantidade de nós que foram *gerados* na busca
6. Quantidade de nós que foram *visitados* na busca

Além disso, pode ser útil listar por nome qual algoritmo, qual função de custo, heurística e demais parâmetros usados como entrada, caracterizando dessa forma qual experimento está sendo rodado.

Documentação Dos Códigos. É importante que o seu código seja legível e bem comentado para a sua avaliação. Além de comentar seu código, adicione um arquivo `readme.txt` ao folder com seus códigos com as instruções necessárias para executá-lo e para rodar os diferentes experimentos. Caso se aplique, indique a versão do compilador/interpretador utilizado e se é necessário instalar alguma biblioteca ou recurso adicional para rodar seu código.

5 Experimentação

Espera-se que vocês passem bastante tempo nessa fase do trabalho, de forma que é importante concluir as implementações com folga para os experimentos.

Parte 0: Experimentação Livre

Inicialmente, procure executar, para as mesmas coordenadas de origem e destino, todos os algoritmos, variando as funções de custo utilizadas e, quando se aplicar, a heurística utilizada. Isso permitirá a você observar algumas coisas no comportamento de cada algoritmo individualmente. Por exemplo, permitirá observar que as buscas em profundidade e largura não são afetadas pela escolha da função de custo. É

interessante que você procura variar as entradas e combinações a partir de cada algoritmo e tente observar como as buscas progridem e como essas mudanças afetam os algoritmos, especialmente as buscas em profundidade e em largura. Procure entender que tipo de variações nas coordenadas de origem e destino afetam mais a execução de cada algoritmo.

Parte 1: Largura vs. Profundidade vs. Custo Uniforme

Repita os seguintes passos 50 vezes (em um laço), guardando em um arquivo os valores de saída para cada vez que executar um dos algoritmos:

Gere aleatoriamente um par (x_1, y_1) de coordenadas para servir de estado inicial e outro par (x_2, y_2) de coordenadas para o estado objetivo. Para cada par de pontos gerados, faça:

i. Execute a Busca em Largura para encontrar um caminho do ponto (x_1, y_1) até o ponto (x_2, y_2) usando a função de custo f_1. Repita a mesma busca usando as funções f_2, f_3, e f_4.

ii. Execute a Busca em Profundidade para encontrar um caminho do ponto (x_1, y_1) até o ponto (x_2, y_2) usando a função de custo f_1. Repita a mesma busca usando as funções f_2, f_3, e f_4.

iii. Execute a Busca de Custo Uniforme para encontrar um caminho do ponto (x_1, y_1) até o ponto (x_2, y_2) usando a função de custo f_1. Repita a mesma busca usando as funções f_2, f_3, e f_4.

Obs.: Como a escolha das funções não afeta a execução dos algoritmos, poderíamos executá-los à revelia das funções de custo e utilizá-las apenas ao final para calcular o custo dos caminhos retornados de acordo com cada função. Essa estratégia, porém, não é recomendada: a execução independente destes conjuntos de parâmetros ajudará a verificar se suas implementações destes algoritmos estão corretas: espera-se que os valores de caminho mudem de acordo com a função utilizada, mas as sequências de ações retornadas pela busca em profundidade devem ser as mesmas para cada execução com as mesmas entradas. O mesmo vale para a busca em largura. Nesse sentido, encare este primeiro experimento como um teste de consistência pros seus algoritmos de base.

Este experimento vai gerar 600 registros para você avaliar, sendo 50 com cada combinação de algoritmo e função de custo. Este é um número pequeno de registros, mas funcionará bem para o nosso trabalho. Lembre-se de anotar no arquivo qual é o algoritmo e a função utilizada para gerar cada resultado. Procure reorganizar estes dados de diferentes maneiras para observar o que ocorre nas execuções de cada algoritmo para as mesmas entradas. Adicione suas conclusões e os arquivos em que você se baseou ao relatório.

Parte 2: Custo Uniforme vs. A*

Repita os seguintes passos 50 vezes (em um laço), guardando em um arquivo os valores de saída para cada vez que executar um dos algoritmos:

Gere aleatoriamente um par (x_1, y_1) de coordenadas para servir de estado inicial e outro par (x_2, y_2) de coordenadas para o estado objetivo. Para cada par de pontos gerados, faça:

i. Execute a Busca de Custo Uniforme para encontrar um caminho do ponto (x_1, y_1) até o ponto (x_2, y_2) usando a função de custo f_1. Repita a mesma busca usando as funções f_2, f_3, e f_4.

ii. Execute a Busca A* para encontrar um caminho do ponto (x_1, y_1) até o ponto (x_2, y_2) usando a função de custo f_1 e função heurísta H_1. Repita a mesma busca usando as combinações restantes das funções f_2, f_3, e f_4 com as heurísticas H_1 e H_2.

Este experimento vai gerar 600 registros para você avaliar, sendo 50 com cada combinação de algoritmo e funções empregadas. Neste caso, serão 200 registros para custo uniforme e 400 registros para o A*. Como no experimento anterior, lembre-se de anotar no arquivo qual é o algoritmo e as funções utilizadas para gerar cada resultado. Procure reorganizar estes dados de diferentes maneiras para observar o que ocorre nas execuções de cada algoritmo para as mesmas entradas. Adicione suas conclusões e os arquivos em que você se baseou ao relatório.

Parte 3: Busca Gulosa vs. A*

Repita os seguintes passos 50 vezes (em um laço), guardando em um arquivo os valores de saída para cada vez que executar um dos algoritmos:

Gere aleatoriamente um par (x_1, y_1) de coordenadas para servir de estado inicial e outro par (x_2, y_2) de coordenadas para o estado objetivo. Para cada par de pontos gerados, faça:

i. Execute a Busca Gulosa para encontrar um caminho do ponto (x_1, y_1) até o ponto (x_2, y_2) usando a função heurística H_1. Calcule o custo do caminho encontrado de acordo com cada função f_1, f_2, f_3, f_4. Repita a mesma busca usando a função H_2. Mais uma vez, calcule o custo do caminho encontrado de acordo com cada função f_1, f_2, f_3, f_4.

ii. Execute a Busca A* para encontrar um caminho do ponto (x_1, y_1) até o ponto (x_2, y_2) usando a função de custo f_1 e função heurísta H_1. Repita a mesma busca usando as combinações restantes das funções f_2, f_3, e f_4 com as heurísticas H_1 e H_2.

Este experimento vai gerar 100 registros da busca gulosa para você avaliar, sendo 50 com cada heurística, mas custos reais diferentes de caminho de acordo com cada função empregada. Lembre-se que a busca gulosa utiliza apenas a função heurística para decidir em que sequência visitará os nós, ignorando a função de custo ao longo da sua execução. A função de custo só será utilizada ao final para calcular o custo real do caminho retornado. Além destes, o experimento vai gerar outros 400 registros para o algoritmo A*, da mesma forma que tivemos na **Parte 2**, compreendendo todas as combinações de função de custo e função heurística disponíveis.

Como nos experimentos anteriores, lembre-se de anotar no arquivo qual é o algoritmo e as funções utilizadas para gerar cada resultado. Procure reorganizar estes dados de diferentes maneiras para observar o que ocorre nas execuções de cada algoritmo para as mesmas entradas e heurísticas. Adicione suas conclusões e os arquivos em que você se baseou ao relatório.

Parte 4: Largura vs. Profundidade com Randomização da Vizinhança

Os experimentos até agora assume que a sua função para gerar vizinhos utiliza os operadores $(-1, +0)$, $(+1, +0)$, $(+0, -1)$, $(+0, +1)$ sempre na mesma ordem. Dessa maneira, cada execução do algoritmo de Busca em Profundidade (respectivamente, Largura) indo das coordenadas (x_1, y_1) até (x_2, y_2) será idêntica, pois esse algoritmo visita novos nós usando apenas a ordem em que eles foram gerados. Para observarmos melhor o comportamente desses algoritmos, consideraremos aplicar estes mesmos operadores, mas embaralhando a ordem de aplicação desses operadores a cada vez que precisarmos gerar vizinhos de um nó na árvore de busca.

Repita os seguintes passos 20 vezes (em um laço), guardando em um arquivo os valores de saída para cada vez que executar um dos algoritmos:

> Gere aleatoriamente um par (x_1, y_1) de coordenadas para servir de estado inicial e outro par (x_2, y_2) de coordenadas para o estado objetivo. Para cada par de pontos gerados, faça:
>
> i. Execute 10 vezes a Busca em Largura gerando os vizinhos de cada nó em ordem aleatória para encontrar um caminho do ponto (x_1, y_1) até o ponto (x_2, y_2). A cada execução, anote o caminho encontrado e calcule o seu custo total usando as funções f_1, f_2, f_3, f_4.
>
> ii. Execute 10 vezes a Busca em Profundidade gerando os vizinhos de cada nó em ordem aleatória para encontrar um caminho do ponto (x_1, y_1) até o ponto (x_2, y_2). A cada execução, anote o caminho encontrado e calcule o seu custo total usando as funções f_1, f_2, f_3, f_4.

Este experimento vai gerar 400 registros para você avaliar, sendo 10 com cada combinação de algoritmo e parâmetros de estado inicial e objetivo. Lembre-se de

anotar no arquivo qual é o algoritmo e a função utilizada para gerar cada resultado. Procure reorganizar estes dados de diferentes maneiras para observar o que ocorre nas execuções de cada algoritmo para as mesmas entradas. Compare qual é o melhor caminho encontrado por cada algoritmo para as mesmas entradas de acordo com cada função de custo. Adicione suas conclusões e os arquivos em que você se baseou ao relatório.

Parte 5: Caminho Mínimo Com Uma Parada A Mais

Esta parte da experimentação é nosso objetivo maior, pois é mais difícil prever o comportamento dos algoritmos e a execução de cada um poderá se alongar. Consideraremos que o agente deseja retornar do trabalho para casa, mas precisa passar em uma farmácia no caminho. Há várias farmácia na cidade e, portanto, muitos caminhos possíveis para o agente resolver o seu trajeto. Entre isso e os custos diferentes que as ações do agente pode ter ao longo da busca, é difícil prevermos qual das coordenadas aleatórias fornecerá ao agente o caminho de menor custo.

Em termos de implementação, precisaremos guardar uma lista de coordenadas de farmácias e, ao gerar um novo nó na árvore de busca, deveremos verificar se o estado a que esse nó se refere contém uma farmácia. Um caminho que resolve esse problema deverá começar no trabalho do agente (estado inicial), encontrar uma farmácia e depois seguir para a casa do agente (objetivo). Nosso objetivo é encontrar o caminho de menor custo que satisfa essas condições.

Este experimento utilizará exclusivamente a Busca A*, mas recomendo que você rode também o algoritmo de Custo Uniforme para observar como essa busca se comportará, especialmente se estiver usando algum código adicional para visualizar os caminhos explorados por cada algoritmo em sua execução. Como antes, utilizaremos coordenadas aleatórias para os locais de interesse.

Repita os seguintes passos 25 vezes (em um laço), guardando em um arquivo os valores de saída para cada vez que executar o algoritmo. Para este experimento, além do estado incial e objetivo, liste também os pontos onde há farmácias em cada instância do experimento. A cada iteração do laço:

Gere aleatoriamente um par (x_1, y_1) de coordenadas para servir de estado inicial e outro par (x_2, y_2) de coordenadas para o estado objetivo. O primeiro ponto representará o endereço do trabalho do agente e o segundo ponto representará o endereço da sua casa.

Gere aleatoriamente mais quatro pares de coordenadas (x_3, y_3), (x_4, y_4), (x_5, y_5), (x_6, y_6) para representar os endereços das farmácias disponíveis na região. Garanta que não ocorrem endereços repetidos.

Para cada conjuntos de pontos gerados, faça:

i. Execute a Busca A* para encontrar um caminho do ponto (x_1, y_1) até o ponto (x_2, y_2) que passe por ao menos um dos endereços de farmácia usando a função de custo f_1 e função heurísta H_1.

ii. Repita a Busca A* para os mesmos pontos usando as combinações restantes das funções f_2, f_3, e f_4 com as heurísticas H_1 e H_2.

Este experimento vai gerar 200 registros para você avaliar, sendo 25 com cada combinação de algoritmo e funções empregadas. Como nos experimentos anteriores, lembre-se de anotar no arquivo que está usando o algoritmo A* e quais funções foram utilizadas para gerar cada resultado. Procure reorganizar estes dados de diferentes maneiras para observar o que ocorre nas execuções do algoritmo para as mesmas entradas e funções. Adicione suas conclusões e os arquivos em que você se baseou ao relatório.

6 Relatório

Procurem ser concisos e técnicos no que forem relatar. Discutam que tipo de testes vocês fizeram para avaliar se as implementações estão corretas e o que a equipe foi capaz de observar em cada experimento. Divida seu relatório de acordo com as seções:

1. **Implementação** - Linguagem utilizadas, bibliotecas e frameworks relevantes, o que é necessário para rodar seus algoritmos e como eu devo proceder para rodá-los com parâmetros de teste escolhidos por mim.

2. **Experimentação** - Divida em subseções nomeadas com **Parte 1** até **Parte 5**. Em cada uma destas, discuta o que foi possível observar na execução dos experimentos e nos dados gerados. Conforme o necessário, havendo arquivos diferentes relacionados à sua análise, indique quais são para que eu possa avaliar suas conclusões.

3. **Considerações Finais** - Caso desejem, podem adicionar uma seção com discussão de resultados de forma mais ampla: comparações entre as diferentes partes do experimento, outras variações que vocês tenham considerado, etc. Esta seção é inteiramente opcional.

7 Pontuação

Para uma avaliação adequada, é necessário que eu possa repetir os experimentos com seus códigos e observar como cada algoritmo se comporta ao longo de uma execução. Será necessário executar os algoritmos e suas variações com valores de entrada que serão escolhidos com o propósito de avaliar as implementações. Desta forma, é imperativo que eu possa executar qualquer um dos algoritmos com as funções necessárias passando

coordenadas de estado inicial e objetivo escolhidos por nós, bem como uma lista de coordenadas de farmácias no caso do último experimento.

- 1,0 ponto: todos os requisitos implementados (experimentos em loop e possibilidade de repetí-los manualmente)
- 1,0 ponto: documentação adequada dos códigos
- 2,0 pontos: implementação correta dos algoritmos
- 5,0 pontos: até um para cada parte de 1 a 5 no relatório (avaliação independente)
- 1,0 ponto: arquivos com dados da experimentação disponíveis e legíveis

Para evitar excesso no número de trabalhos para correção, é mandatório que cada equipe seja composta por três pessoas, enquanto possível. Para reforçar essa diretriz, a soma dos pontos obtidos será multiplicada por:

- $1,00$ para trabalhos submetidos por uma equipe com três pessoas
- $0,75$ para trabalhos submetidos por uma equipe com apenas duas pessoas
- $0,50$ para trabalhos submetidos por uma equipe com apenas uma pessoa
- $0,50$ para trabalhos submetidos por uma equipe com mais que três pessoas

Como a quantidade de estudantes da turma não é um múltiplo de três, é possível que tenhamos uma ou duas equipes com menos que três pessoas ao final. Esses casos só serão aceitos quando não houverem mais colegas disponíveis para formar equipes e tratados diretamente comigo. Equipes nessa situação não serão penalizadas por terem menos membros.

8 Outras Variações Interessantes

Nos experimentos que discutimos, foram propostas diversas funções de custo. As funções f_3 e f_4, em particular, fornecem custos variáveis às ações do agente na direção horizontal. Em ambos os casos, os valores começam *maiores* que o valor constante das ações da direção vertical. Nesse sentido, você pode propor funções semelhantes que comecem fornecendo custos menos na direção horizontal para ver como isso afeta as soluções dos algoritmos.

Similar a essa proposta, você pode permitir uma variação dos experimentos com as funções f_3 e f_4 em que o parâmetro de *tempo* é um fornecido como entrada. Dessa maneira, dependendo do tempo, os valores retornados pelas funções f_3 e f_4 para as primeiras ações na horizontal poderão variar bastante, podendo resultar em caminhos diferentes como respostas nos algoritmos de Busca de Custo Uniforme e Busca A*.

Sinta-se à vontade para propor outras funções, variações, e discutí-las no relatório. Contribuições interessantes nesse sentido poderão ajudar na avaliação geral do trabalho, compensando possíveis falhas nos pontos anteriores do trabalho.