QXD0037 - Inteligência Artificial

Trabalho de Implementação II – Especificação
O Problema do Caxeiro Viajante via Algoritmo A*
Prof. Samy Sá, 2016.2

PUBLICAÇÃO: 21/10/2016
PRAZO: 04/11/2016

Neste projeto, você deverá utilizar o algoritmo A* para atacar o problema do Caxeiro Viajante. Este documento determinará o formato de entrada e saída que seu programa deve seguir e detalha a representação de estados e operadores que devem ser implementados.

O Problema do Caixeiro Viajante (TSP)

Enunciado: Dado um conjunto de cidades, as distâncias entre cada par destas cidades, e uma cidade inicial, encontre o menor caminho que passa exatamente uma vez por cada cidade e retorna à cidade inicial.

Para simplificar o problema, suponha que todas as cidades são conectadas. Desta forma, o mapa das cidades deve ser considerado um grafo G = (V,E) completo em que os nós são as cidades e cada aresta representa o caminho entre as cidades em suas extremidades.

Representação do Problema

Para aplicar o A* ao TSP, considere que:

- As cidades são pontos com coordenadas ***x***, ***y*** em um mapa (um plano).
- A distância entre quaisquer duas cidades é dada diretamente pela distância geométrica entre os pontos do plano em que as cidades estão situadas. Ou seja, dadas duas cidades A,B com coordenadas (x_1,y_1) e (x_2,y_2), a distância entre as cidades será dada por $\text{sqrt}((x_1 - x_2)^2 + (y_1 - y_2)^2)$. Atribua esta distância como peso à aresta (A,B).
- Os estados do espaço de busca são como no Trabalho de Implementação I, mas esta técnica explorará o espaço de busca construindo sequência de cidades. Para tal, cada nó da árvore construída se referirá a uma das cidades na lista, apenas.
- A função heurística a ser utilizada será o custo da Árvore Geradora Mínima (AGM) do subgrafo induzido pelas cidades não visitadas e a cidade inicial. As definições seguem:
  - Dado G = (V,E) e um $V' \subseteq V$, o subgrafo induzido por V' será G' = (V',E'), onde $E' = \{(x,y) \in E \mid x, y \in V'\}$.
  - Dado G = (V,E), uma árvore geradora é um subgrafo qualquer G' = (V', E') conexo e acíclico (uma árvore) tal que V' = V. A árvore geradora mínima é a árvore geradora cuja soma dos pesos das arestas é mínimo. Forneceremos nas seções a seguir um algoritmo eficiente para calcular a AGM de um grafo.
- O estado inicial é uma cidade qualquer do mapa.
- O único operador disponível consiste em seguir da cidade atual para uma das cidades que ainda não foi visitada. Quando todas as cidades já tiverem sido visitadas uma vez em um caminho qualquer, o operador consiste em seguir da cidade atual para a cidade inicial, fechando o circuito.

Formatos de entrada e saída

A entrada do programa será um arquivo com coordenadas de pontos. O arquivo terá duas linhas: a primeira terá todas as coordenadas ***x*** de cada cidade no plano e a segunda terá todas as coordenadas

y de cada cidade no plano. Por exemplo, a entrada do programa pode conter o conteúdo:

0.0	1.0	2.0	0.0	1.0	1.0	3.0
0.0	2.0	0.0	2.0	1.0	3.0	2.0

O conteúdo acima indica uma entrada com 7 cidades em que as coordenadas da primeira são ***x*** = 0.0 e ***y*** = 0.0, as coordenadas da segunda cidade são ***x*** = 1.0 e ***y*** = 2.0, e assim por diante.

Para facilitar os cálculos de custos de circuitos no seu programa, utilize a entrada para calcular uma matriz M[i,j] das distâncias entre cada par de cidades i, j.

Para a saída do seu programa base, exiba na tela, a cada iteração do algoritmo A*, a cidade ***s*** que foi visitada nesta iteração e os custos das funções g(***s***), h(***s***), e f(***s***) = g(***s***) + h(***s***).

Algoritmo para AGM

Você pode utilizar qualquer algoritmo para o cálculo da AGM. Uma sugestão é utilizar o Algoritmo de Kruskal, que executa em O(|E|.log |E|), onde |E| é tamanho do conjunto de arestas. O pseudo-código do algoritmo segue:

Entrada: um grafo G
Saída: um subgrafo S = (V', E') de G, tal que V' = V e S é uma árvore.

1. ordene as arestas de G em ordem de custo crescente
2. crie um subgrafo S = (V', E') de G tal que V' = V e E' é inicialmente vazio
3. enquanto S for desconexo, faça:
4. selecione a aresta ***e*** de custo mínimo
5. se os nós nos extremos de ***e*** estiverem desconectados, adicione ***e*** a S
6. retorne S.

A manutenção do grafo auxiliar S deve ser percebida como a manutenção de uma floresta, ou seja, um conjunto de árvores, cada qual um subgrafo de G. No início do algoritmo, cada nó é uma árvore. A cada iteração do laço que adicionar uma aresta a S, estamos conectando das florestas restantes, de forma que o número de florestas diminui em 1. Eventualmente, teremos apenas duas florestas e a próxima aresta de menor custo que for adicionada tornará S conexo.

Admissibilidade da Heurística Utilizada

Observe que a AGM de um grafo G, sendo uma árvore, escolhe as ***n – 1*** arestas de menor custo que conectam os nós de G, onde ***n*** é o número de nós. Um circuito nestes ***n*** nós, por sua vez, utilizará ***n*** arestas e estas precisam estar organizadas de forma que cada nó seja extremidade de exatamente duas arestas. Destaque uma cidade (um nó) **A** desse circuito para ser a cidade de partida e considere que a cidade visitada por último, logo antes de retornar a **A** seja a cidade **B**. O caminho de **A** até **B** tem, portanto, ***n* – 1** arestas. Observe que este caminho é um subgrafo conexo e acíclico de G, ou seja, uma árvore. Mais que isso, dado que este caminho passa por todos os nós de G, trata-se de uma *árvore geradora* de G. Naturalmente, o custo do caminho será maior ou igual à da AGM, uma vez que a AGM tem, por definição, o custo *mínimo*. Concluímos que a AGM como estimativa de custo de caminho mínimo entre dois nós (passando por todos os demais do grafo) jamais superará o custo real deste caminho e é, portanto, uma heurísitica admissível.

Experimentação

Para esta atividade, suas execuções do algoritmo devem visar responder a duas perguntas, as quais devem ser comentadas no relatório da atividade:

Q1: A escolha da cidade inicial influencia a execução do algoritmo?

Fundamentação: Uma vez fixada uma entrada para o programa, a resposta do circuito mínimo será a mesma. Este circuito, por definição, para por todas as cidades e consiste de um caminho circular, então devemos ser capazes de encontrá-lo independetemente da cidade que escolhermos como partida. Será, então, que a escolha da cidade inicial afeta a performance do A* quando o grafo é completo?

Método: Para responder a esta pergunta, execute o algoritmo repetidas vezes, iniciando-o por cidades diferentes. Se o mapa utilizado tiver até 10 cidades, repita a busca iniciando uma vez por cada cidade. Em cada execução, anote o número de estados visitados e o número de estados gerados. Quando utilizar um mapa com mais que 10 cidades, escolha aleatoriamente 10 ou mais cidades de partida para comparação, guardando sempre os mesmos dados: número de estados visitados e número de estados gerados. Você deve experimentar várias execuções com diferentes mapas e avaliar os dados em busca de uma conclusão para esta pergunta.

Q2: A heurística de AGM permanece eficiente se excluírmos a cidade inicial de nossas estimativas?

Fundamentação: No enunciado original da heurística, calcularemos a AGM do subgrafo induzido pelas cidades não visitadas + a cidade de partida **A**. O motivo de adicionarmos a cidade de partida ao subgrafo é, por intuição, que estaremos estimando o custo do menor caminho que passa por todas as cidades não visitadas e retorna à partida. Intuitivamente, se removermos essa cidade das estimativas, nossa heurística ficará menos informada. Será que essa diferença de qualidade é relevante? Será que os números de estados visitados e gerados muda significativamente? E o tempo de execução do algoritmo, teria uma melhoria por reduzirmos a entrada do Kruskal, executado a cada novo nó que geramos?

Método: Para responder a esta pergunta, para cada entrada de mapa e seleção da cidade de partida, execute o A* com as duas variações da heurística AGM, a dizer, incluindo e excluindo a cidade de partida das estimativas. Compare os dados dos conjuntos de execuções que fizer para cada mapa guiando-se pela fundamentação acima e registre suas conclusões.

Comparação Entre as Variações

Para cada arquivo de entrada que utilizar no seu algoritmo, registre os número gerados em uma tabela com uma linha para cada exercução do algoritmo indicando a cidade inicial, a versão da heurística utilizada e os totais de cidades visitadas e geradas. Faça testes com arquivos de entradas distintos e quantidade crescente de número de cidades, indentificando cada tabela de acordo com os respectivos arquivos de entrada.

Em um relatório curto (~2 páginas), descreva a sua rotina de experimentação, indicando quantas execuções e números de cidades mínimo e máximo que avaliou. Responda às perguntas indicadas na seção anterior com base nos seus experimentos.

É encorajado que outras perguntas sejam levantadas e que seus experimentos adequem-se para respondê-las, seja durante a rotina sugerida ou com uma nova bateria de testes. Se este for o caso, comente suas perguntas adicionais, indicando como adequou os experimentos e a que conclusões você chegou.

Envie o seu relatório e todos os arquivos produzidos (código, dados de execução, etc.) em um arquivo compactado quando da entrega do trabalho.