Intelligent Systems (Sheet #1)

Marius Gavrilescu

1. (a) A state is a tuple (m, c, p) of two integers and a position (LEFT or RIGHT). The starting state is (3, 3, LEFT). The goal state is (0, 0, RIGHT). Valid states are those where $0 \leq m, c \leq 3$ and ($m = 0$ or $m = 3$ or $m = c$). From state (m, c, LEFT) we can go to (m', c', RIGHT) where $m' \leq m, c' \leq c, c - c' + m - m' \in \{1, 2\}$, and from (m', c', RIGHT) we can go to (m, c, LEFT) with the same restrictions.

   (b) One optimal solution is $(3, 3, LEFT); (3, 1, RIGHT); (3, 2, LEFT); (3, 0, RIGHT); (3, 1, LEFT); (1, 1, RIGHT); (2, 2, LEFT); (0, 2, RIGHT); (0, 3, LEFT); (0, 1, RIGHT); (0, 2, LEFT); (0, 0, RIGHT)$.

   It is better to use a graph search because actions are reversible, so a tree search would perform very slowly.

   (c) Most people would not approach this problem by formulating the state space and actions precisely. They would instead try to guess a solution, which is not always a good way of solving a problem.

2. (a) Suppose we have a tree with n nodes in a chain (in other words, branching factor 1 and depth n). A depth first search would do $O(n)$ steps, while a iterative deepening search would do $1 + 2 + 3 + \ldots + n = O(n^2)$ steps.

   (b) If the edges have different weights, iterative deepening will not necessarily find an optimal solution since it always finds the same solution as a breadth-first search. Hence, if the optimal solution is on the third level of the tree but there is another (suboptimal) solution on the second level, BFS and iterative deepening would both (incorrectly) find the latter, while UCS will find the former.

3. If a node has been marked visited, then all its neighbours have been added to the frontier previously, so they are either visited or will be visited soon. Hence both searches will consider all nodes.

   Additionally, both searches will consider the nodes in increasing order of cost. Therefore the first solution found will necessarily be the one with the lowest cost.

   Since all nodes will be considered, and the result of the search is the goal node with the lowest cost, both searches find an optimal solution.

4. (a) i. L[244=0+244]
   ii. M[311=70+241], T[440=111+329]
   iii. L[384=140+244], D[387=145+242], T[440=111+329]
   iv. D[387=145+242], T[440=111+329], M[451=210+241], T[580=251+329]
   v. C[425=265+160], T[440=111+329], M[451=210+241], M[461=220+241], T[580=251+329]
   vi. T[440=111+329], M[451=210+241], M[461=220+241], P[503=403+100], T[580=251+329], R[604=411+193], D[627=385+242]
   vii. M[451=210+241], M[461=220+241], L[466=222+244], P[503=403+100], T[580=251+329], A[595=229+366], R[604=411+193], D[627=385+242]
   viii. M[461=220+241], L[466=222+244], P[503=403+100], L[524=280+244], D[527=285+242], T[580=251+329], A[595=229+366], R[604=411+193], D[627=385+242]
   ix. L[466=222+244], P[503=403+100], L[524=280+244], D[527=285+242], L[534=290+244], D[537=295+242], T[580=251+329], A[595=229+366], R[604=411+193], D[627=385+242]
   x. P[503=403+100], L[524=280+244], D[527=285+242], M[533=292+241], L[534=290+244], D[537=295+242], T[580=251+329], A[595=229+366], R[604=411+193], D[627=385+242], T[662=333+329]
   xi. B[504=504+0], L[524=280+244], D[527=285+242], M[533=292+241], L[534=290+244], D[537=295+242], T[580=251+329], A[595=229+366], R[604=411+193], D[627=385+242], T[662=333+329], R[693=500+193], C[701=541+160]

   (b) The same solution would be returned by the graph search (?). The algorithm would be faster, but more memory will be used (although the frontier will be smaller, since we don't have nodes in it twice, we need to remember the visited nodes).

5. (a) Assume a heuristic h is consistent. Let h^* be the true cost function. Then $h(n) \leq c(n, a, n') + h(n')$, for any neighbouring nodes n and n'. We want to prove the heuristic does not overestimate for any node. We know the heuristic does not overestimate for the goal nodes (because $h(g) = h^*(g) = 0$).

   We will do a uniform-cost graph search on the transpose graph, starting from the goal nodes (for which the heuristic does not overestimate). Whenever we expand from node x to node y, we know that the heuristic does not overestimate for node x (in other words $h(x) \leq h^*(x)$) and that $h(y) \leq c(y, a, x) + h(x) \leq c(y, a, x) + h^*(x)$. But since uniform-cost graph search is optimal, we know $c(y, a, x) + h^*(x) = h^*(y)$. So $h(y) \leq h^*(y)$. Thus the heuristic does not overestimate for node y.

   By induction we get that the heuristic does not overestimate for any node, so the heuristic is admissible.

   (b) The heuristic is admissible, since it does not overestimate any distance, but it is not consistent since $h(B) > 1 + h(A)$

   (c) `S[9=0+9],`
   `A[5=4+1], B[9=2+7]`
   `B[9=2+7], C[10=6+4]`
   `A[8=7+1], C[10=6+4]`
   `C[9=5+4], C[10=6+4]`
   `G[9=9+0], C[10=6+4]`

   Yes, the solution is optimal.

6. (a) A state is a tuple (v, c) where v is a set of not yet visited nodes, and c is the current node. Let v_0 be a random element of V. The start state is (V, v_0). From (v, c) we can go to $(v \cup \{x\}, x)$ with cost $w(\langle c, x\rangle)$ for any $x \in V \setminus v$ such that $\langle c, x\rangle \in E$. The goal state is $(\{\}, v_0)$.

   (b) The path cost is $t = w(\langle c, x\rangle)$ as described in the previous part. UCS will find the tour with the lowest t (= the optimal tour). Since the state space is so large, UCS will perform very slowly.

   (c) $h^*((v, c)) = min(w(\langle c, x_1\rangle) + w(\langle x_1, x_2\rangle) + \ldots + w(\langle x_k, v_0\rangle))$ where $\langle c, x_1\rangle, \langle x_1, x_2\rangle, \ldots, \langle x_k, v_0\rangle \in E$, $x_i = x_j \implies i = j$ and $v = \{x_1, x_2, \ldots, x_k\}$.

   This means that $h^*((v, c))$ is the length of the shortest path from c to v_0, passing through all of then nodes in v and no others, without passing more than once through a node.

   The A* algorithm would go straight via the optimal path if we're using this ideal heuristic. However, the heuristic is not feasible because there is no reasonably quick way to compute it.

   (d) For a given state, we compute the MSF of the subgraph of the TSP graph which only includes unvisited nodes. Let S be the sum of the edges of the MSF. Clearly, the real cost of visiting them all is more than S. Then let A be the length of the shortest path from the current node to an unvisited node, and B be the length of the shortest path from an unvisited node to the destination.

   Then $h(n) = S + A + B$.

   $h(n)$ is easy to compute unlike h^*, while still being a consistent heuristic.

   (e) The heuristic is certainly admissible, because the sum of the edges of the MSF cannot be larger than the cost to tour its nodes.

   If we go from (v, c) to $(v \setminus x, x)$ then the cost increase is $w(\langle c, x\rangle)$. For consistency we need to prove $h((v, c)) \leq h((v \setminus x, x)) + w(\langle c, x\rangle)$, which is equivalent to $A + S + B \leq A' + S' + B' + W$.

   The B component of the heuristic cannot decrease, because it is the shortest edge from v to v_0, and we removed a node from v. So $B \leq B'$. Hence we only need to prove $A + S \leq A' + S' + W$

   A is the shortest edge from c to a node in v. Hence, $A \leq W$. So we only need to prove $S \leq A' + S'$

   But A' is the length of the shortest edge from x to a node in $v \setminus x$, so adding that edge to the new MSF we get a forest spanning v of edge sum $A' + S'$. As S is the edge sum of MSF spanning v, we get $S \leq A' + S'$.

   Therefore our heuristic is both admissible and consistent, so it will always find an optimal solution when plugged into A* (of either the tree or the graph variety).