Department of Computer Engineering

University of Peradeniya

CO541: Artificial Intelligence
Assignment 3

1)w = 0 gives f(n) = 2g(n). This behaves exactly like uniform-cost search the factor of two makes no difference in the ordering of the nodes. w = 1 gives A^* search. w = 2 gives

f(n) = 2h(n), i.e., greedy best-first search. We also have

$$f(n) = (2 - w)\left[g(n) + \frac{w}{2-w} h(n)\right]$$

which behaves exactly like A^* search with a heuristic $\frac{w}{2-w} h(n)$

For w <= 1, this is always less than h(n) and hence admissible, provided h(n) is itself admissible.

2)

3)

a.
b.
c. If we relax the constraint for TSP such that each city can be visited more than one time (that is, there may be some cities visited more than one time) and the cost for repeated edges are not count, we can get a solution for MST problem.
d.

4)Well, since h1(n) ≥ h2(n) we could trivially say that if the two heuristics are equal then h2 will not result in expanding less nodes than h1, but I assume that is not the intent of the question, so let's consider h1(n) < hs(n).

For any given node n, it will be placed in the open list with a values of f(n) = g(n) + h(n). Since f1(n) ≥ f2(n), every node evaluated by f1 functions will have an estimated at least as large as f2. Since the node with the smallest value is chosen for expansion, one of two possibilities exit

- f1(n) ≥ f2(n) and n has the smallest value. In this case the same node is expanded by both heuristics.
- f1(n) ≥ f2(n) and there exits a node n 0 in the open list such that f1(n) ≥ f1(n 0) and f2(n) ≤ f2(n 0). In this case, A∗ will select node n 0 first, and, if that node lead to a solution state, node n will not be expanded. Note that the inequlity for f2 is valid since it just shows that f2 under estimated the path cost to the goal from node n 0 .

Thus we have shown that A∗ with heuristic h1 con only expand as many or less nodes than h2.

5)

In this suppose n' = S2 and n = S1

- Here our heuristic are admissible as they satisfy h(n) ≤ c(n)
  h(n) =10 < 10+2;
  h(n') = 5 < 10.

- Now c(n,a,n') = 2; h(n') = 5; h(n) = 10
  As h(n) =10 and c(n,a,n') + h(n') = 7
  or 10 > 7 so our graph is inconsistent.
- Now f(n) = g(n) + h(n)
  So f(n) = 10 + 1 = 11 and f(n') = 5+5 = 10
  Based on this from initial state we will go to n' as f(n') < f(n)
  Now from n' it will go to G(goal state) f(G) = 10+0 = 10.
  Which is smallest in all
  So total cost = 5 + 10 = 15
  But this is non optimal as there exist an optimal path through n with cost 13 (1+2+10).
- So here A* graph-search returns a suboptimal solution with an h(n) function that is admissible but inconsistent.

6)

7) It is optimal, if enough memory is available to store the shallowest optimal solution path. Otherwise, it returns the best solution (if any) that can be reached with the available memory.

8)

9)(a)Local beam search with k=1

* We would randomly generate 1 start state
* At each step we would generate all the successors, and retain the 1 best state
* Equivalent to HILL-CLIMBING

(b) Local beam search with k=∞

* 1 initial state and no limit of the number of states retained
* We start at initial state and generate all successor states (no limit how many)
* If one of those is a goal, we stop
* Otherwise, we generate all successors of those states (2 steps from the initial state), and continue
* Equivalent to BREADTH-FIRST SEARCH

(c) Simulated annealing with T = 0 at all times

* If T is very small, the probability of accepting an arbitrary neighbor with lower value is approximately 0
* This means that we choose a successor state randomly and move to that state if it is better than the current state
* Equivalent to FIRST-CHOICE HILL CLIMBING

(d) Genetic algorithm with population size N = 1

* If selection step necessarily chooses the single population member twice, so the crossover steo does nothing.
* Moreover, if we think of the mutation step as selecting a successor at random, there is no guarantee that the successor is an improvement over the parent
* Equivalent to RANDOM WALK