E/15/366
S.THINESH

CO541: Artificial Intelligence
Assignment 3

1)

a. 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),greedy best-first search.

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

3)

a. The MST heuristic is admissible as it is always shorter than or equal to a closed loop.
b.
c. The TSP is the search for the cheapest closed loop to visit all given cities. Relaxing this without the need to close the loop, a minimal spanning tree is a relaxed sub-problem which connects all cities with minimal connection cost. The MST heuristic is admissible as it is always shorter than or equal to a closed loop.
d.

4)

Well, since $h1(n) \geq h2(n)$ 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 $f1(n) \geq f2(n)$, 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

- $f1(n) \geq f2(n)$ and n has the smallest value. In this case the same node is expanded by both heuristics.
- $f1(n) \geq f2(n)$ and there exits a node $n\ 0$ in the open list such that $f1(n) \geq f1(n\ 0)$ and $f2(n) \leq f2(n\ 0)$. 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 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 $h(n) \leq c(n)$
  $h(n) = 10 < 10+2$;
  $h(n') = 5 < 10$.

- 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)

If we assume the comparison function to be transitive, then we can always sort the nodes using it and choose the node that is at the top of the sorted list.

Assuming the comparison function is a total ordering, we can still do best first search by sorting the queue the comparison function. However, since we don't have information about how much better a given node is than another we cant combine the results of the comparison function with other information. So we cant do A*. We also give up the optimality guarantees that come with A* search. Also like greedy search, we loose completeness.

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)

SMA* or Simplified Memory Bounded A* is a shortest path algorithm based on the A* algorithm. The main advantage of SMA* is that it uses a bounded memory, while the A* algorithm might need exponential memory.

SMA*-

It will utilize whatever memory is made available to it. It avoids repeated states as far as its memory allows. It is complete if the available memory sufficient to store the shallowest solution path. It is optimal if enough memory is available to store the shallowest optimal solution path. Otherwise, it returns the best solution that can be reached, with the available memory.

A* algorithm –

A* uses best-first search and finds the least-cost path from a given initial node to the goal node (out of one or more possible goals) .It uses a distance-plus-cost heuristic function (denoted by f(n)) f(n) = g(n) + h(n) . Heuristic function should be admissible (not over estimate) and monotone A* always finds a cheapest solution path if the heuristic is admissible

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= α

- 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 neighbour 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 step 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