1a)

* **Min { h1(n), h2(n) }** is admissible.

For a heuristic to be admissible h(n) <= C\*(n).

Since, h1(n) <= C\*(n)

h2(n) <= C\*(n)

Therefore, min{h1(n), h2(n)} <= C\*(n)

Hence the heuristic is admissible.

**Min { h1(n), h2(n) }** is not consistent.

For a heuristic to be consistent h(n) <= C(n, a, n’) + h(n’)

Now, with c(n, a, n’) remaining constant,

The below will not hold

Min { h1(n), h2(n) } <= c(n, a, n’) + Min { h1(n’), h2(n’) }

As, Min { h1(n’), h2(n’) }

Hence the heuristic is consistent

* **w h1(n) + (1 -w)h2(n)** (0<=w<=1) is admissible.

Since, h1(n) <= C\*(n)

h2(n) <= C\*(n)

and, w h1(n) + (1 -w)h2(n) <= max {h1(n), h2(n)}

Therefore, w h1(n) + (1 -w)h2(n) <= C\*(n) ( See below)

Hence the heuristic is admissible

**w h1(n) + (1 -w)h2(n)** is consistent.

Since, h1(n) <= c(n, a, n’) + h1(n’) …1

And h2(n) <= c(n,a,n’) +h2(n) …2

Multiplying …1 by (w) and …2 by (1-w)

We get, w h1(n) + (1 -w)h2(n) <= c(n,a,n’) + w h1(n) + (1 -w)h2(n)

Hence, the heuristic is consistent.

* **Max { h1(n), h2(n) }** is admissible.

Since, h1(n) <= C\*(n)

h2(n) <= C\*(n)

Therefore, max{h1(n), h2(n)} <= C\*(n)

Hence the heuristic is admissible.

**Max { h1(n), h2(n) }** is consistent.

Now again since c(n, a, n’) will remain constant,

the below statement will hold

Max { h1(n), h2(n) } <= c(n, a, n’) + Max { h1(n’), h2(n’) }

Hence the heuristic is consistent.

1b)

For w=1, the algorithm is guaranteed to be optimal.

For w=0, it performs Breadth First Search

For w=1, it performs A\*algorithm

For w=2, it performs Greedy Best First Search Algorithm

2.

a) For problems where we are just looking for a solution and not the best solution Hill climbing is better. For example in real time path finding we would want a solution fast which might not be the best solution.

b) For some problems like 8-puzzle problem half the states are unreachable from the other half states. In such cases where some of the states cannot be reached from the current state, random restart can give better results.

c) Simulated annealing is should be used when we have continuous value function with multiple local Maximas.

d)We should store the best value state after each iteration. At the end we should compare the last state with the best value state and return the best value state.

e) We can use genetic algorithm along with simulated annealing by keeping a population of solutions. New states can be derived by crossover along with the simulated annealing.