Comparison between Hill Climbing search and Simulated annealing algorithms

A hill-climbing algorithm is a local search algorithm that moves continuously upward (decreasing heuristic value in our 8 puzzle problem) until the best solution is attained. This algorithm comes to an end when the peak is reached. So hill climbing finds optimal solutions for convex problems – for other problems it will find only local optima (solutions that cannot be improved upon by any neighbouring configurations), which are not necessarily the best possible solution (the global optimum) out of all possible solutions (the search space).

To avoid getting stuck in local optima, one could use a probabilistic technique for approximating the global optimum of a given function.

Simulated annealing combines hill climbing with random walk, i.e. it randomly choses the neighbour of a node with some probability, even if change in energy is positive(heuristic/ energy value of new node is greater than previous node) i.e. choosing a bad neighbour

This probability of choosing bad neighbour randomly, decreases as the parameter deciding the probability decreases, i.e. T temperature)

So as our algorithm proceed its search, after every iteration we decrease the temperature by applying some cooling function, hence further as iterations increase/time passes , it will choose bad neighbours with lesser probabilities.

```
Start State

1 2 3
5 6 0
7 8 4
```

```
Goal State

1 2 3
5 8 6
0 7 4
```

	Hill Climbing search (Manhattan distance)	Hill Climbing (#Misplaced Tiles)
No. of states generated	10	10
No. of states explored	3	3
No. of states to optimal path	3	3
Optimal path cost	3	3
Time taken	0.0009963512420654297	0.0029947757720947266

Following result for simulated annealing is best of all successes over multiple trial runs of algorithm

	Simulated Annealing (Manhattan distance)	Simulated Annealing (#Misplaced Tiles)
No. of states explored	3	5
Optimal path cost	3	5
Time taken	0.0	0.0

	Time Complexity	Space Complexity
Hill Climbing	O(d)	O(b)
Simulated Annealing	O(d)	O(b)

b : branching factor , d:depth of search space tree

e) Constraints to be checked

ii) If we make a new heuristics h3 (n) = h1 (n) * h2 (n) then, chances of converging faster are more as more heuristic information has been taken into consideration.

But, during experimentation we found that during multiple runs it did gave solutions close to optimal, but also quite frequently in some cases it went into rigorous computation, which was not the case when misplaced heuristics were used alone or when Manhattan was used less frequently success was obtained but never went into rigorous computation or say looping.

From our understanding we can say it may be because of deltaE variation from other two heuristic functions, as "boltz = exp(-float(deltaE)/temperature)" might be computing out to be very low so random condition "r <= boltz" where r is a random no. between (0,1), will be false most of the times and it will iterate over and over for accepting a neighbour state that is bad in nature with some probability value which is coming out very low.

iii) If we consider the blank tile as another tile:
If we assume that swaps are possible among numbered tiles then it will make search space larger as no. of possible moves at each state would be more, which in turn will make our search more rigorous as branching factor(b) and depth(d) will increase considerably.

iv) if the search algorithm got stuck into Local optimum? Is there any way to get out of this?

As we saw in the case of new heuristic h3 = h1*h2, sometimes it was getting stuck in local optimum and was not able to select any neighbour having very less probabilities under random behaviour of algorithm.
We could do few things to overcome this problem:
a) Simply apply a time restriction and when that limit surpasses assign the neighbour state(with whatever less probability) as current state without comparing with random no. OR just limit the no. of iterations done for a particular state while randomly choosing one of its neighbours as current.

Conclusion:

Hill climbing and Simulated Annealing both have very less time and space complexity but both doesn’t guarantee the optimality and completeness, also simulated annealing may converge over a single run or it may not because of its random walk property, and also it generates different results over multiple runs of its algorithm.

Simulated annealing have advantage over hill climbing as it can solve the problem of local maxima which occurs in hill climbing, so chances of its convergence towards a global maxima are more than that in hill climbing, which could achieve a global maxima only if its initial state is very close to it i.e. only one steep ascent/descent away as the case may be maximization/minimization problem.