Problem #1

Question: Provide an example of boards in which Local Beam Search with two starting points does not give the same result as Hill Climbing with a single restart.

The local beam search algorithm begins with *k* randomly generated states. At each step, all successors of the *k* states are generated. The *k* successors from across all the boards with the *k* best scores are passed to the next round. This process is repeated until all successors have lower values than the best solution in the current round at which point the algorithm terminates. Unlike with the Hill Climbing algorithm (both with and without restart), local beam search allows sharing of successor quality information between the *k* states.

Overview of this Example

Heuristic Cost Function (*h*): The number of pairs of attacking queens. In this case, the algorithms are trying to minimize the heuristic function (i.e. $h = 0$). This is not Hill Climbing in the strictest sense. If we wanted to maximize a parameter, we could use heuristic, h', where h' is defined as:

$$h' = \sum_{i=1}^{8} (i - 1) - h$$

The summation:

$$\sum_{i=1}^{8} (i - 1)$$

is used because it represents the maximum number of possible attacking pairs (i.e. all the queens in one row):

$$7 + 6 + 5 + 4 + 3 + 2 + 1 = 28$$

You could then maximize h' to 28. For simplicity, I will discuss minimizing h to 0 due to its greater simplicity. However, the concept is identical regardless of which of the two approaches is used.

Queen Symbol on the Board: ♠

State Transition Definition: Per round, one queen can be moved from its current position to any other square within the same column. For example, a queen in square A1 (see below) can move to A2, A3, A4, A5, A6, A7, and A8 only.

Board Overview

Board # 1:

	A	B	C	D	E	F	G	H
8							♠	
7					♠			
6		♠						
5				♠				
4						♠		
3								♠
2			♠					
1	♠							

Board # 1 - Initial Board for the Hill Climbing Algorithm and Initial State #1 for Local Beam Search

Description: This board comes from the textbook *Artificial Intelligence: A Modern Approach* by Russell and Norvig (see page 123 in the third edition).

Heuristic Value of this Board:

$$h = 1$$

List of Pairs of Attacking Queens:

(D5, G8)

Importance of this Board:

Per Russell and Norvig, this board is a local minima.[1] Any movement of a queen results in a higher heuristic function (i.e. $h > 1$). Hence, if Hill Climbing was run on this board, it would immediately terminate since it is at a local minimum.

Board # 2:

	A	B	C	D	E	F	G	H
8	18	12	14	13	13	12	14	14
7	14	16	13	15	12	14	12	16
6	14	12	18	13	15	12	14	14
5	15	14	14	♠	13	16	13	16
4	♠	14	17	15	♠	14	16	16
3	17	♠	16	18	15	♠	15	♠
2	18	14	♠	15	15	14	♠	16
1	14	14	13	17	12	14	12	18

Board # 2 – Restart Board for the Hill Climbing Algorithm and Initial State #2 for Local Beam Search.

Description: This board also comes from the textbook *Artificial Intelligence: A Modern Approach* by Russell and Norvig (see page 123 in the third edition). Those squares that do not contain a queen have numbers describing the heuristic cost if a queen in that column was moved into that space. For example, if the queen in cell B4 was moved to B8, the heuristic cost, h, would be 12.

Heuristic Value of this Board:

$$h = 17$$

List of Pairs of Attacking Queens:

(A4, B3)	(A4, C2)	(A4, E4)	(B3, C2)	(B3, D5)	(B3, F3)	(B3, H3)	(C2, E4)	(C2, G2)
(D5, E4)	(D5, F3)	(D5, G2)	(E4, F3)	(E4, G2)	(F3, G2)	(F3, H3)	(G2, H3)	

Importance of this Board:

Per Russell and Norvig, the minimum heuristic cost by moving one queen within its column is 12 (see cells: B8, B6, E7, E1, F8, F6, G7, and G1).

[1] This stated here without proof based off Russell and Norvig's assertion. This statement could be easily proven using the brute force approach of generating the 56 (7 * 8) successor states.

Board Traversal Using Hill Climbing

Step #1: The hill climbing algorithm examines all possible successors of Board # 1. Per Russell and Norvig, no successors have lower heuristic costs. Hence, the algorithm terminates. Since it is not a goal state, it generates another random (i.e. restart) board (in this case Board # 2).

Step #2: The hill climbing algorithm examines Board # 2 and observes the eight successor states that have identical minimum value (i.e. B8, B6, E7, E1, F8, F6, G7, and G1). The algorithm chooses one of the successors and then generates subsequent successors from that one state until a local minimum is found or the goal is reached.

Board Traversal Using Local Beam Search

Step #1: The local beam algorithm generates all possible successors for Board # 1 and Board # 2. Per Russell and Norvig, there are no moves within the same column of Board # 2 that yields a heuristic cost of less than 12. Per Russell and Norvig, Board # 1 has no successor states with a heuristic cost of less than or equal to 1. Therefore, all successor states from the two starting boards have heuristic costs greater than the current minimum. As such, the algorithm immediately terminates.

Summary

In Hill Climbing, Board # 1 was immediately passed over in the first round (since it was a local minima), but Hill Climbing does traverse through successor states in Board # 2. As such, Hill Climbing could still find a solution from Board # 2 if such a solution exists and is reachable. In contrast, Local Beam Search immediately terminates after the first round/generation (since all successors in both boards have lower heuristic costs than the current generation). Local beam search could never find a solution to these boards (if one existed).

Problem #2

Question: Provide a schedule for a simulated annealing algorithm that could be used to solve the eight queens problem. Give an example of a situation where rather than going in the direction of a solution, the algorithm instead goes from a better state to a less fit one.

Below is modified version of the pseudocode from Russell and Norvig for simulated annealing. In their example code, a heuristic function is being maximized. In contrast, I am using the heuristic function I described in Problem #1 where I am trying to minimize a heuristic (i.e. $h = 0$); this difference necessitated minor adjustments to the code. This algorithm runs until the temperature (T) falls below some minimum value (T_{Min}) or the solution is at the goal (i.e. no attacking queens).

function Simulated-Annealing($problem, schedule, T_{Min}$) **returns** a final state
// $problem$ – Problem to be solved
// $schedule$ – Temperature cool down scalar
// T_{Min} – Terminating Temperature
$current = Make\_Node(problem.\text{INITIAL\_STATE})$
for t = 0 **to** ∞
    $T = schedule(t)$
    if $T < T_{Min}$ **or** $current.is\_at\_goal()$ **return** *current*
    next = a randomly selected next state
    $\Delta E = next.value - current.value$
    if $\Delta E < 0$ **then** $current = next$
    // $random()$ is pseudorandom number generator that generates numbers between 0 and 1 with a uniform distribution
    else if $random() < e^{\frac{-\Delta E}{T}}$ **then** $current = next$

Overview of this Example

Same as my answer to Problem #1; see section "Overview of this Example" for the complete description. In brief, the heuristic is the number of pairs of attacking queens. A successor state is one where a single queen from the current state is moved within its column.

Example Board

Same as my answer to Problem #1; see section "Board # 1". In brief, the board shown below has a heuristic cost of $h = 1$. Per Russell and Norvig, this board is a local minima, and all successor states have higher (i.e. worse) heuristic cost.

	A	B	C	D	E	F	G	H
8							♠	
7					♠			
6		♠						
5				♠				
4						♠		
3								♠
2			♠					
1	♠							

Since there are no greater value solutions using the previously specified transition paradigm, a lesser value solution will be selected with probability, P, where P is:

$$P = e^{\frac{-\Delta E}{T}}$$

In the pseudocode, T was defined as:

$$T = schedule(t)$$

Hence, the equation for P can be rewritten as:

$$P = e^{\frac{-\Delta E}{schedule(t)}}$$

By definition, any probabilities (including P) must be between 0 and 1. Therefore, the previous equation can be rewritten:

$$0 \leq e^{\frac{-\Delta E}{schedule(t)}} \leq 1$$

By taking the natural logarithm of both sides, you get:

$$-\infty < \frac{-\Delta E}{schedule(t)} \leq 0$$

Any real number is greater than negative infinity so the left side of the inequality can be ignored. Moreover, the change in energy (ΔE) can be simplified to a constant k (where $k \geq 0$ because of the pseudocode structure). This allows the above equation to be rewritten as:

$$\frac{-k}{schedule(t)} \leq 0$$

Multiplying both sides by $schedule(t)$ (with the condition that $schedule(t) \neq 0$), you further simplify to:

$$-k \cdot schedule(t) \leq 0$$

$$\therefore schedule(t) \geq 0$$

Therefore, the schedule function must always be greater than 0 and should degrade slowly enough (but not too slowly as to become a random walk) to increase the likelihood of exiting local minimum. A function that fills this condition well is the logarithm function. Hence, the schedule could be:

$$schedule(t) = 1 + c \cdot \log\left(\frac{d}{t}\right)$$

where $0 < d \leq 1$ and $0 < c$. By adjusting c and d, it is possible to accelerate (decelerate) the rate at which the $schedule(t)$ decreases as t increases. This equation fulfills the condition of always be greater than 0 with the above specified conditions.

The following graph shows the relationship between $schedule(t)$ and P when $\Delta E = 1$, $c = 0.25$, and $d = 0.5$.

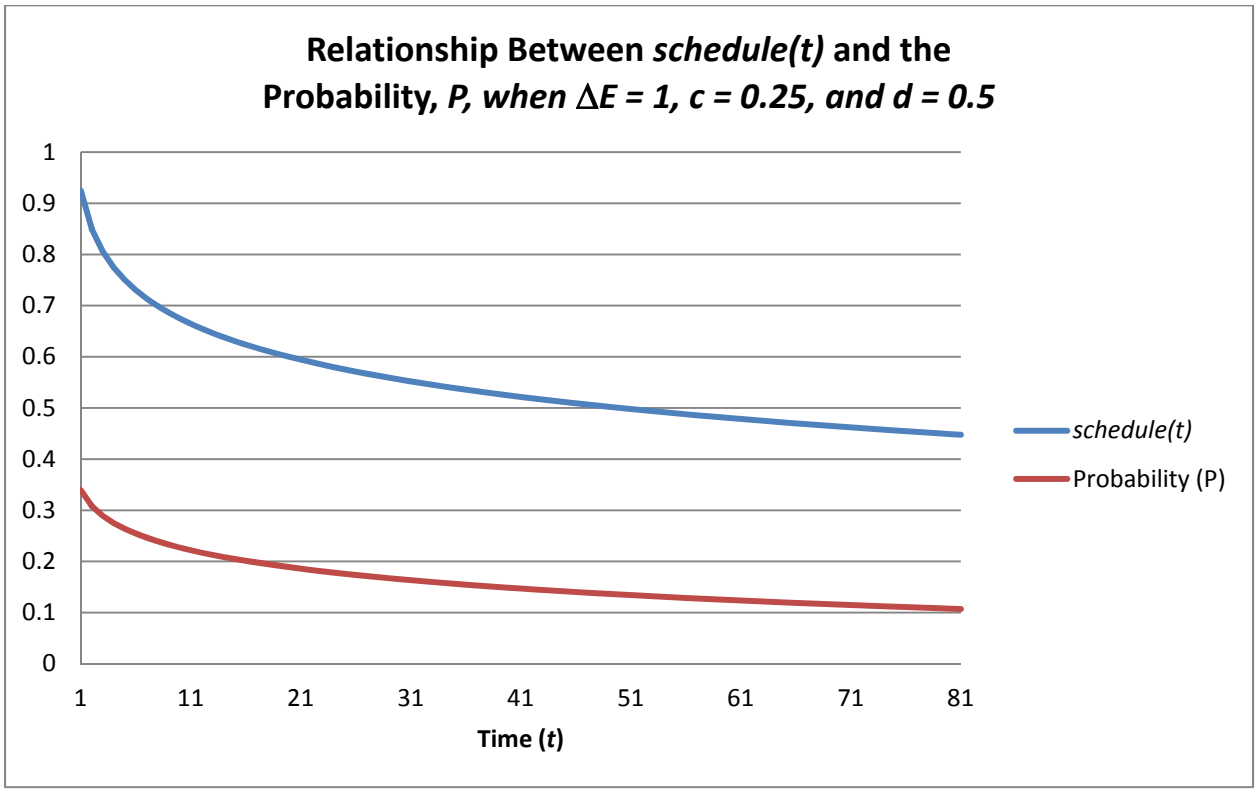

Figure 1 – Relationship between the Probability of Selecting a Lower Value State Using the Specified Schedule Function when $\Delta E = 1$, $c = 0.25$, and $d = 0.5$.

As t increases, P decreases; the rate at which P decreases slows as t increases. By the time P approaches 0, ideally the solution (if it exists) would have been found.

Correlation with the Example Board

If the specified schedule was applied to the example board and the ΔE was assumed to be fixed at 1 (it would not necessarily be fixed, but this is done to simplify the model), then the probability the algorithm would have transitioned from the local minimum to a worse state by time t is shown in Figure 2. Below is a table of this probability of transition to a lower value state with respect to time, t. As t approaches infinity, the probability of transition to a lower value state approaches 1 (i.e. certainty)

Time (t)	Probability of Transition to a Lower Value State
1	0.339
5	0.826
10	0.956
20	0.996
50	0.99997

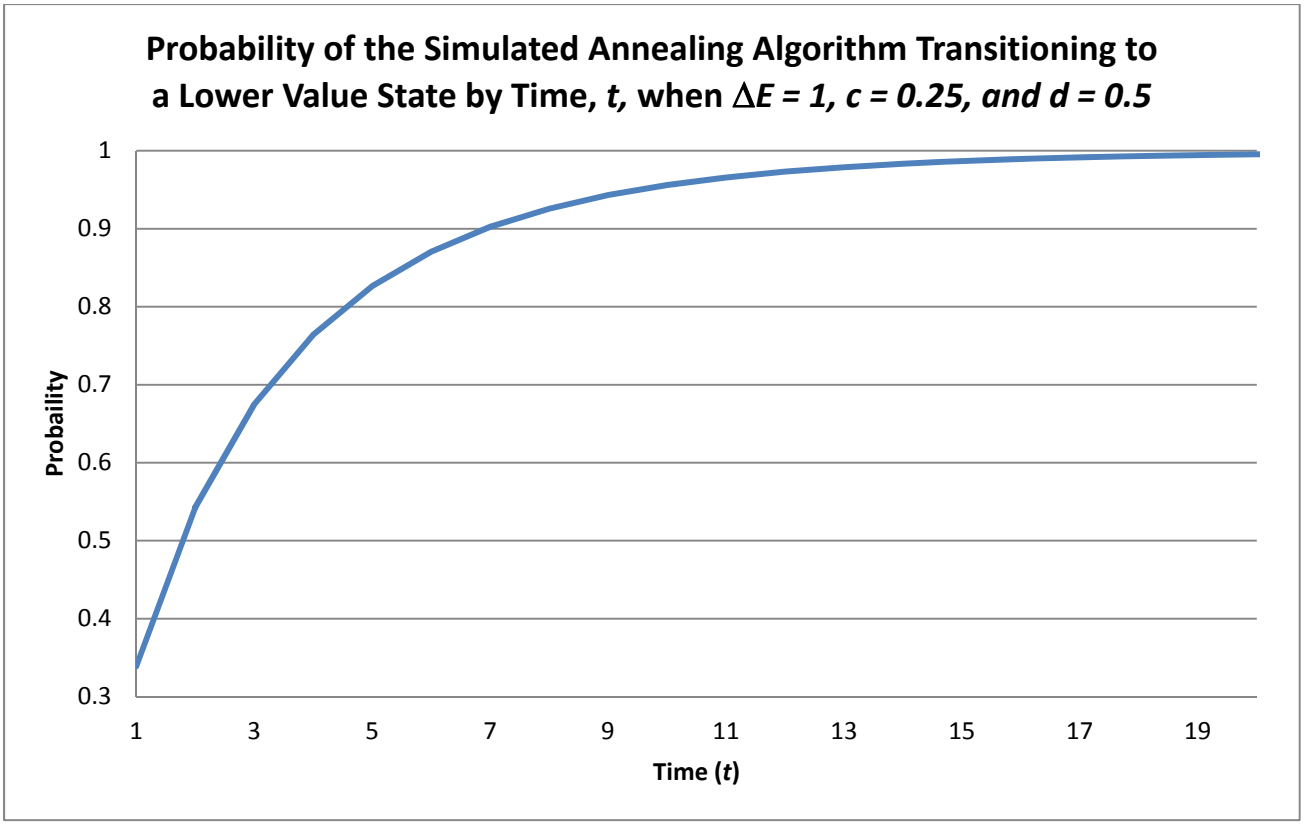

Figure 2 – Likelihood of a Transition to a Lower Value State by Time *t* when $\Delta E = 1$, $c = 0.25$, and $d = 0.5$.