*4. Give the name that results from each of the following special cases:*

*a. Local beam search with k=1.*

a. Local beam search with k = 1 is hill-climbing search.

*b. Local beam search with one initial state and no limit on the number of states retained.*

b. Local beam search with k = ∞: strictly speaking, this doesn’t make sense. The idea is that if every successor is retained (because k is unbounded), then the search resembles breadth-first search in that it adds one complete layer of nodes before adding the next layer. Starting from one state, the algorithm would be essentially identical to breadth-first search except that each layer is

generated all at once.

*c. Simulated annealing with T=0 at all times (and omitting the termination test).*

c. Simulated annealing with T = 0 at all times: ignoring the fact that the termination step

would be triggered immediately, the search would be identical to first-choice hill climbing because every downward successor would be rejected with probability 1.

*d. Simulated annealing with T=infinity at all times.*

d. Simulated annealing with T = infinity at all times: ignoring the fact that the termination step would never be triggered, the search would be identical to a random walk because every successor would be accepted with probability 1. Note that, in this case, a random walk is approximately equivalent to depth-first search.

*e. Genetic algorithm with population size N=1.*

e. Genetic algorithm with population size N = 1: if the population size is 1, then the two selected parents will be the same individual; crossover yields an exact copy of the individual; then there is a small chance of mutation. Thus, the algorithm executes a random walk in the space of individuals.

*2. (20 points total, 5 pts off for each wrong answer, but not negative) Label the following as T (= True) or F (= False). Unless stated otherwise, assume a finite branching factor, step costs ≥ ε > 0, and at least one goal at a finite depth. You may be in either a tree or a graph.*

*a. (5 pts) An admissible heuristic NEVER OVER-ESTIMATES the remaining cost (or distance) to the goal.*

TRUE, by definition of admissible.

*b. (5 pts) Best-first search when the queue is sorted by f(n) = g(n) + h (n) is both complete and optimal when the heuristic is admissible and the total cost estimate f(n) is monotonic increasing on any path ~~to a goal node~~.*

TRUE, because the search described is A\* and the heuristic described is both admissible and consistent.

*c. (5 pts) Most search effort is expended while examining the interior branch nodes of a search tree.*

FALSE. Most search effort is expended while examining leaf node of the tree.

*d. (5 pts) Uniform-cost search (sort queue by g(n)) is both complete and optimal when the path cost never decreases.*

TRUE, because uniform-cost search is A\* search with h(n) = 0, which is admissible.

*e. (5 pts) Greedy best-first search (sort queue by h(n)) is both complete and optimal when the heuristic is admissible and the path cost never decreases.*

FALSE. Your book gives a counter-example (Fig. 3.23, 3rd ed.; Fig. 4.2, 2nd ed.).

*f. (5 pts) Beam search uses O(bd) space and O(bd) time.*

FALSE. For a beam search in a tree using k nodes total, the space used is O(bk) and the time is O(bmk). For a beam search in a graph, the space is again O(bk) but it can waste time in loops.

*g. (5 pts) Simulated annealing uses O(constant) space and can escape from local optima.*

TRUE. The space is constant and it accepts bad moves with probability exp(-delta(Value)).

*h. (5 pts) Genetic algorithms use O(constant) space and can escape from local optima.*

TRUE. The space is constant and it can accept bad moves by creating bad offspring.

*i. (5 pts) Gradient descent uses O(constant) space and can escape from local optima.*

FALSE. The space is constant, but it generally moves toward, and gets stuck on, a local optima.

*3. (20 points total, 5 pts each) Perform Simulated Annealing search to maximize value in the following search space.*

*Recall that a good move (increases value) is always accepted (P = 1.0); a bad move (decreases value) is accepted with probability P = eΔVAL/T , where ΔVAL = VAL(Next) − VAL(Current).*

A (VAL=15)

B (VAL=5)

C (VAL=45)

D (VAL=44)

E (VAL=48)

F (VAL=47.9)

G (VAL=48.3)

*Use this temperature schedule:*

|  |  |  |  |
| --- | --- | --- | --- |
| *Time Step* | *1–100* | *101–200* | *201–300* |
| *Temperature (T)* | *10* | *1.0* | *0.1* |

*This table of values of e may be useful:*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *x* | *0.0* | *−1.0* | *−4.0* | *-4.3* | *−40.0* | *−43.0* |
| *ex* | *1.0* | *≈0.37* | *≈0.018* | *≈0.014* | *≈4.0\*10−18* | *≈2.1\*10−19* |

*a. (5 points total, 1 pt off for each wrong answer, but not negative)Analyze the following possible moves in the search. The first one is*

*done for you as an example.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Time* | *From* | *To* | *T* | *ΔVAL* | *ΔVAL/T* | *P* |
| *57* | *A* | *B* | *10* | *−10* | *−1* | *0.37* |
| *78* | *C* | *B* | 10 | −40 | −4 | ≈0.018 |
| *132* | *C* | *B* | 1.0 | −40 | −40 | ≈4.0\*10−18 |
| *158* | *C* | *D* | 1.0 | −1 | −1 | ≈0.37 |
| *194* | *E* | *D* | 1.0 | −4 | −4 | ≈0.018 |
| *194* | *E* | *B* | 1.0 | −43 | −43 | ≈2.1\*10−19 |
| *238* | *E* | *D* | 0.1 | −4 | −40 | ≈4.0\*10−18 |
| *263* | *E* | *F* | 0.1 | −0.1 | −1 | ≈0.37 |
| *289* | *G* | *F* | 0.1 | −0.4 | −4 | ≈0.018 |
| *289* | *G* | *D* | 0.1 | −4.3 | −43 | ≈2.1\*10−19 |

*b. (5 pts) At Time=100, is the search more likely to be in state A or in state C? (ignore E, G)*

C.

*c. (5 pts) At Time=200, is the search more likely to be in state A, C, or E? (ignore G)*

E.

*d. (5 pts) At Time=300, is the search more likely to be in state A, C, E, or G?*

G.