Heuristic function

- A heuristic function is a function that maps from problem state descriptions to measures of desirability, usually represented as numbers
- Well designed heuristic function play an important role in guiding a search process toward a solution.

- The purpose of heuristic function is to guide the search process in profitable direction by suggesting which path to follow first when more than one is available.
- There is a trade off between the cost of evaluating a heuristic function and the savings in search time that the function provides.

- A heuristic is a technique that improves the efficiency of a search process by sacrificing claims of completeness. Heuristics are like our tour guides.
- Using good heuristics we can get good possible non optimal solutions to hard problems in less than exponential time.

Algorithm : Generate-and-Test

1. Generate a possible solution.
2. Test to see if this is actually a solution by comparing the chose point or the endpoint of the chosen path to the set of acceptable goal states.
3. If a solution has been found, quit. Otherwise, return to step 1.

GENERATE-AND-TEST

- Acceptable for simple problems.
  - Eg : 1. finding key of a 3 digit lock.
    2. 8-puzzle problem

- Inefficient for problems with large space.
- Use DFS as all possible solution generated, before they can be tested.

GENERATE-AND-TEST

- Generate solution randomly: British museum algorithm;

  wandering randomly.

- Exhaustive generate-and-test. : consider each case in depth

- Heuristic generate-and-test: not consider paths that seem unlikely to lead to a solution.
- Plan generate-test:

□□– Create a list of candidates.

□□– Apply generate-and-test to that list on the basis of constraint-satisfaction.

Ex – DENDERAL, which infers the structure of organic compounds using mass spectrogram and nuclear magnetic resonance (NMR) data.

HILL CLIMBING

- Generate-and-test + direction to move (feedback from test procedure).
- Test function + heuristic function = Hill Climbing
- Heuristic function (objective function) to estimate how close a given state is to a goal state.
- Hill climbing is often used when a good heuristic function is available for evaluating states but when no other useful knowledge is available.

SIMPLE HILL CLIMBING

- Evaluation function as a way to inject task-specific knowledge into the control process.
- Key difference between Simple Hill climbing and Generate-and-test is the use of evaluation function as a way to inject task specific knowledge into the control process.
- Better : higher value of heuristic function

  Lower value

Algorithm : Simple Hill-Climbing

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.

2. Loop until a solution is found or until there are no new operators left to be applied in the current state:

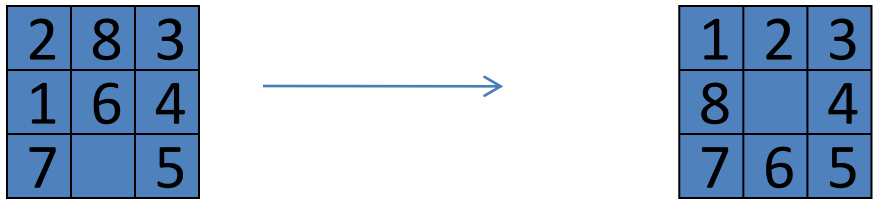
(a) Select an operator that has not yet been applied to the current state and apply it to produce a new state.

(b) Evaluate the new state.

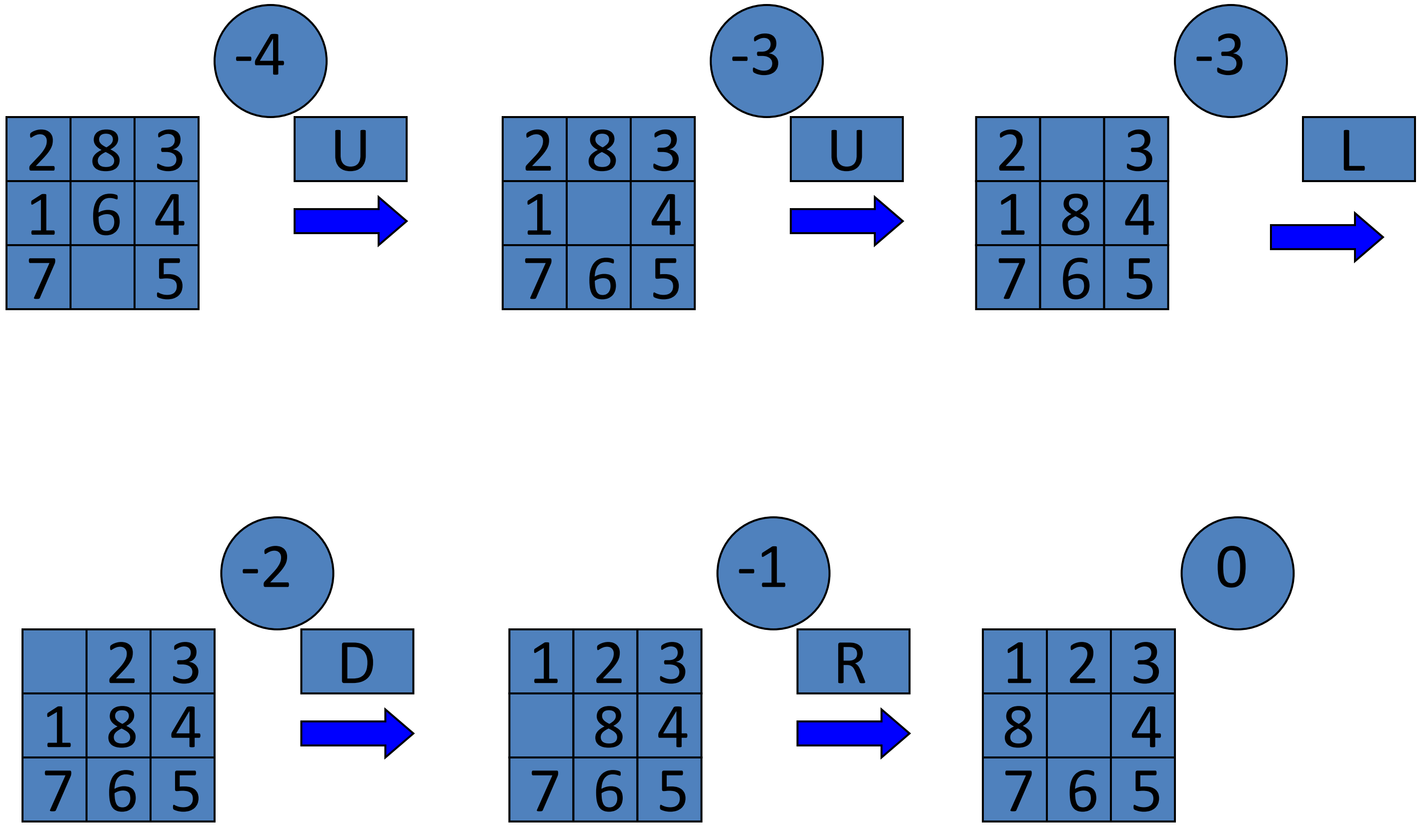
- **(i) If it is a goal state, then return it and quit.**
- **(ii) If it is not a goal state but it is <u>better</u> than the current state, then make it the current state.**
- **(iii) If it is not better than the current state, then continue in the loop.**

EX

1) Use heuristic function as measure of how far off the number of tiles out of place.
2) Choose rule giving best increase in function.



Example


-4
2 8 3
1 6 4
7 5
U
-3
2 8 3
1 4
7 6 5
U
-3
2 3
1 8 4
7 6 5
L
-2
2 3
1 8 4
7 6 5
D
-1
1 2 3
8 4
7 6 5
R
0
1 2 3
8 4
7 6 5

STEEPEST-ASCENT HILL CLIMBING

- Considers all the moves from the current state.
- Selects the best one as the next state.
- Also known as Gradient Search.

Algorithm : Steepest-Ascent Hill Climbing or gradient search

1. **Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.**
2. **2. Loop until a solution is found or until a complete iteration produces no change to current state:**

(a) Let *SUCC* be a state such that any possible successor of the current state will be better than *SUCC.*

(b) For each operator that applies to the current state do:

  (i) Apply the operator and generate a new state.

  (ii) Evaluate the new state. If it is a goal state, then return it and quit. If not, compare it to *SUCC.* If it is better, then set *SUCC* to this state. If it is not better, leave *SUCC* alone.

(c) If the SUCC is better than current state, then set current state to SUCC.

HILL CLIMBING: DISADVANTAGES

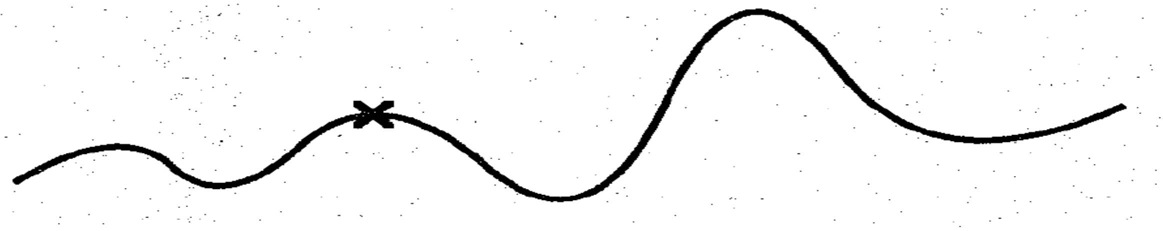
- Fail to find a solution
- Either Algorithm may terminate not by finding a goal state but by getting to a state from which no better state can be generated.

- This happen if program reached
  - Local maximum: A state that is better than all of its neighbours, but not better than some other states far away.
  - Plateau: A flat area of the search space in which all neighbouring states have the same value.
  - Ridge: Special kind of local maximum.

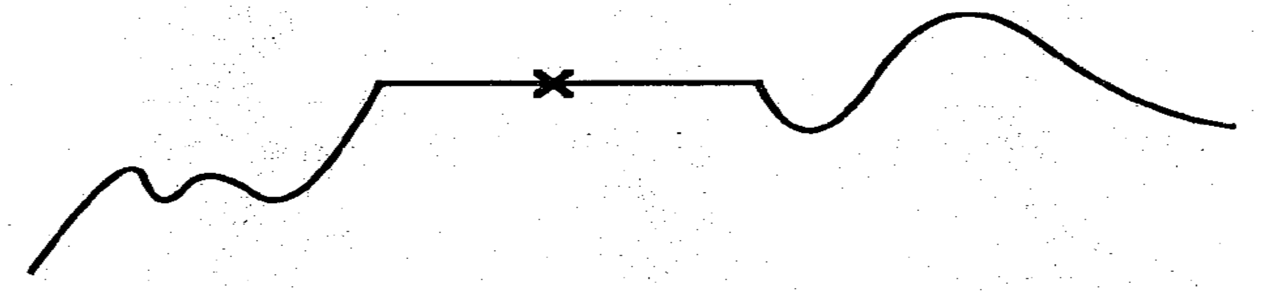
    The orientation of the high region, compared to the set of available moves, makes it impossible to climb up.

Hill-Climbing Dangers

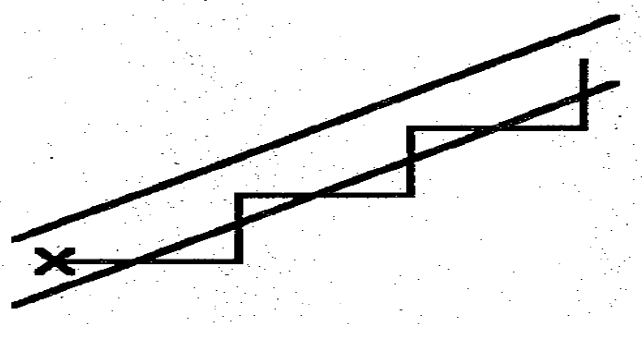
- Local maximum



- Plateau



- Ridge



HILL CLIMBING: DISADVANTAGES

Ways Out

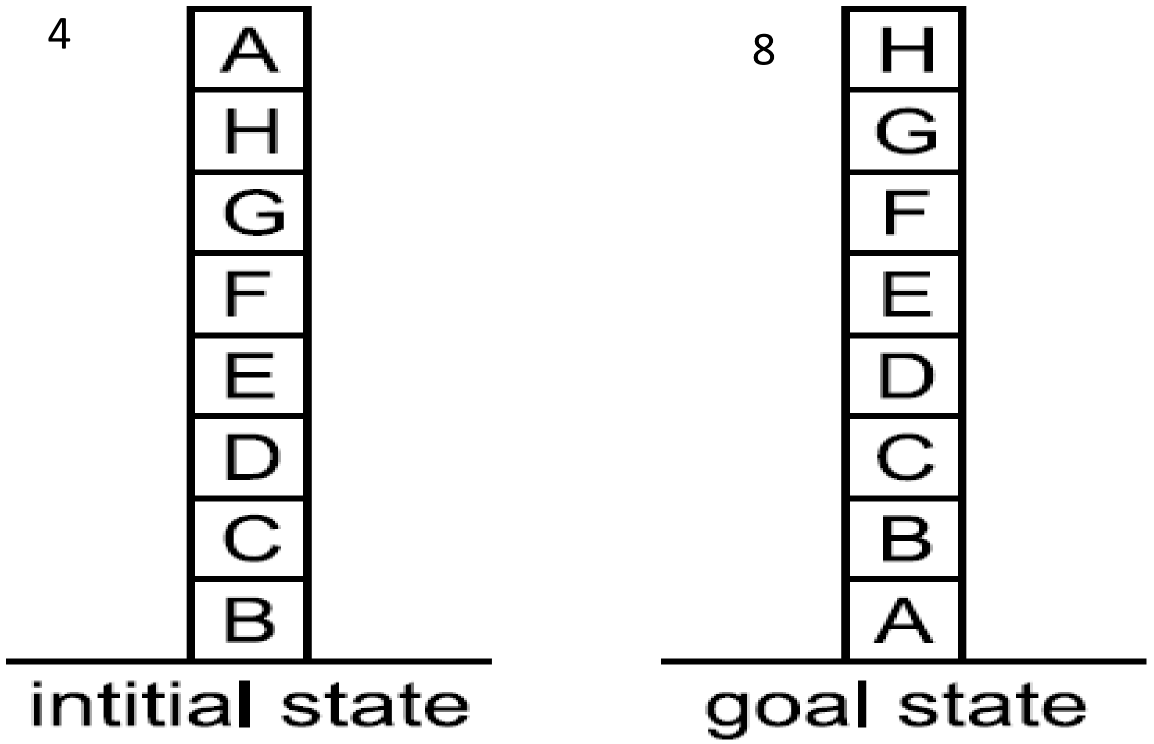
- Backtrack to some earlier node and try going in a different direction. (good way in dealing with local maxima)
- Make a big jump to try to get in a new section. (good way in dealing with plateaus)
- Moving in several directions at once. (good strategy for dealing with ridges)

HILL CLIMBING: DISADVANTAGES

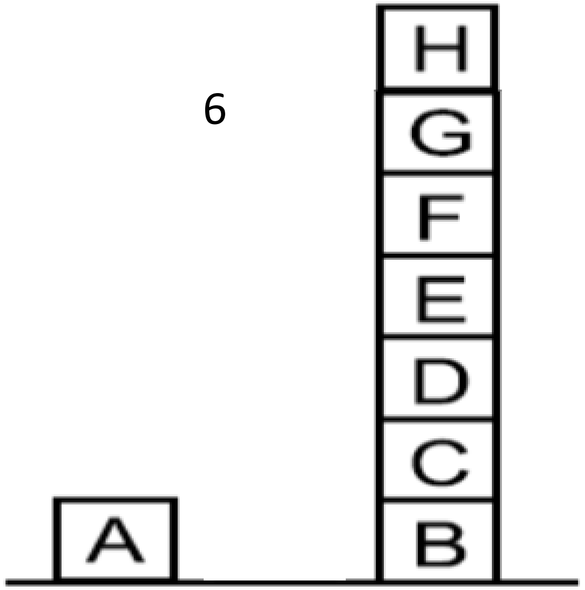
- Hill climbing is a local method:
  Decides what to do next by looking only at the “immediate” consequences of its choices rather than by exhaustively exploring all the consequences.
- Global information might be encoded in heuristic functions.

A Hill-Climbing Problem

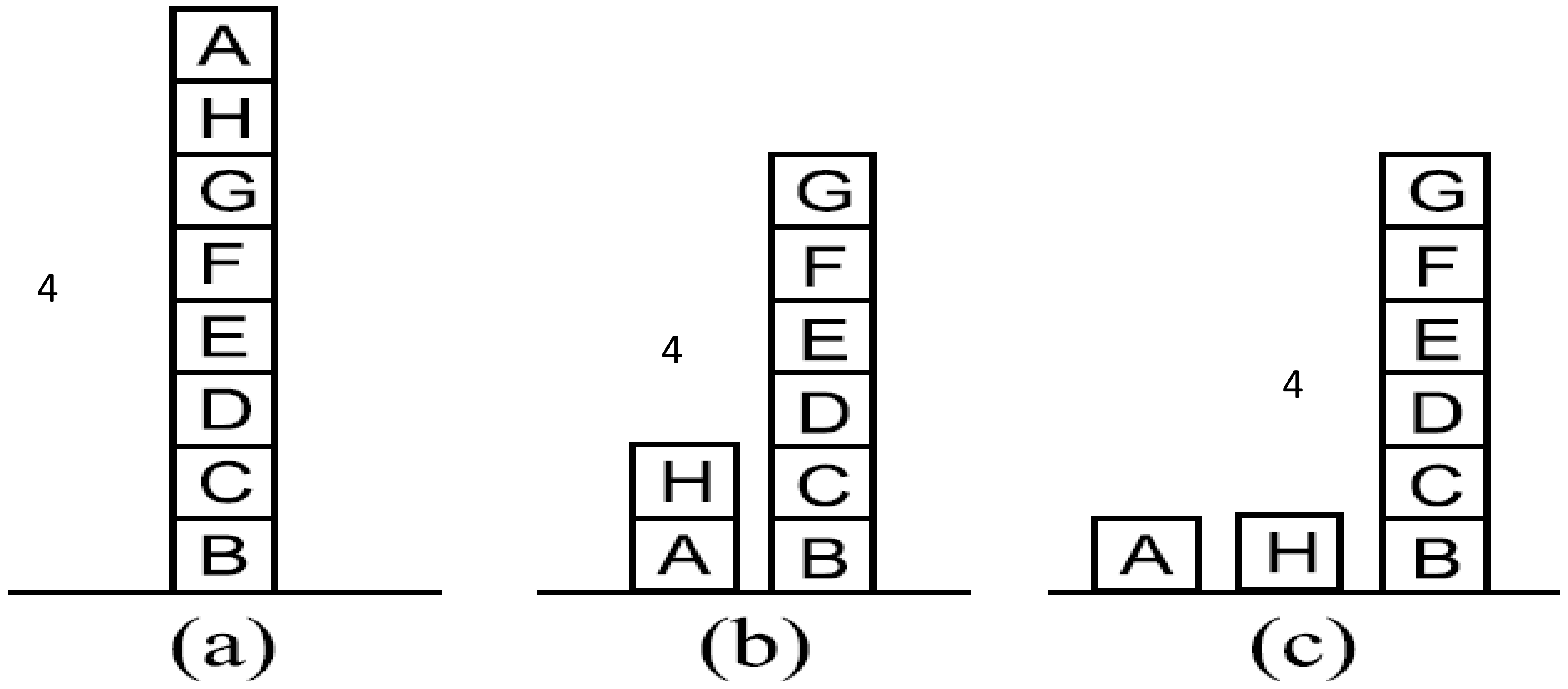
Local: Add one point for every block that is resting on thing it is supposed to be resting on. Subtract one point from every block that is sitting on wrong thing.



One Possible Moves



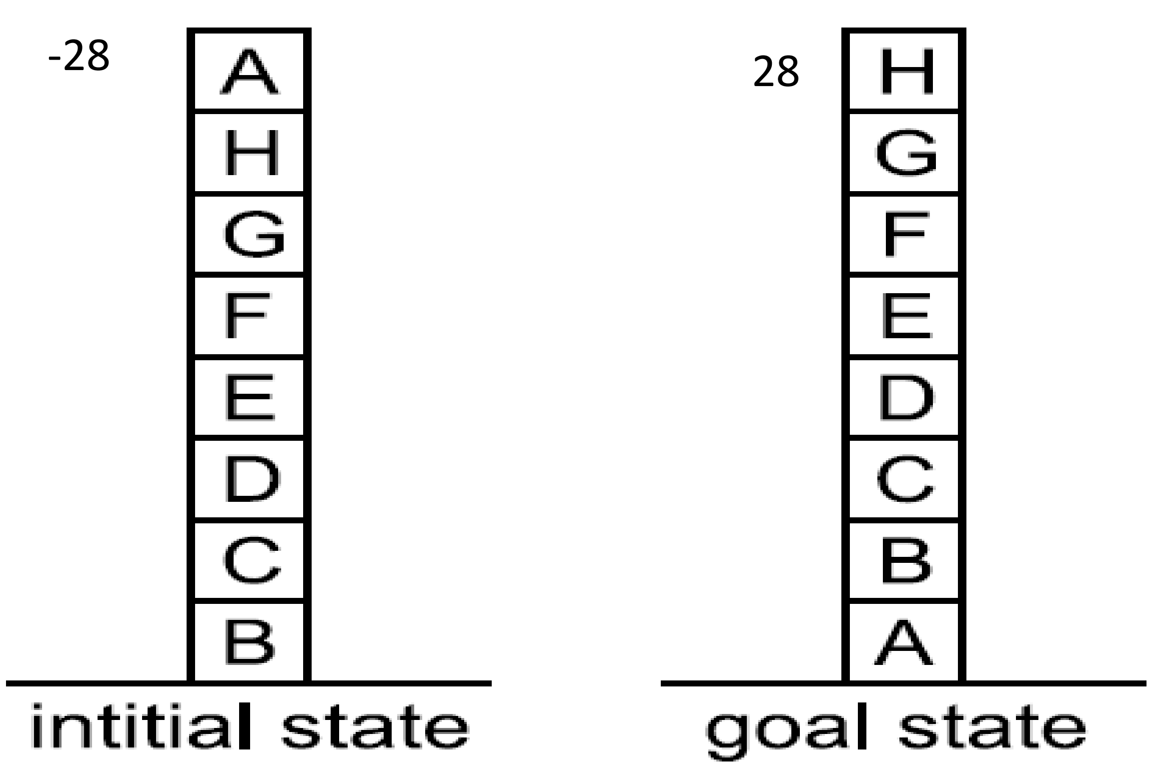
Three Possible Moves



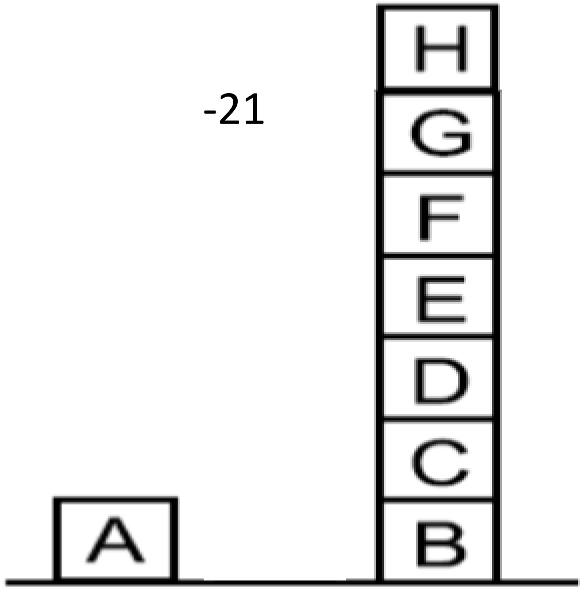
Hill Climbing will Halt because all states have lower score than the Current state.

A Hill-Climbing Problem

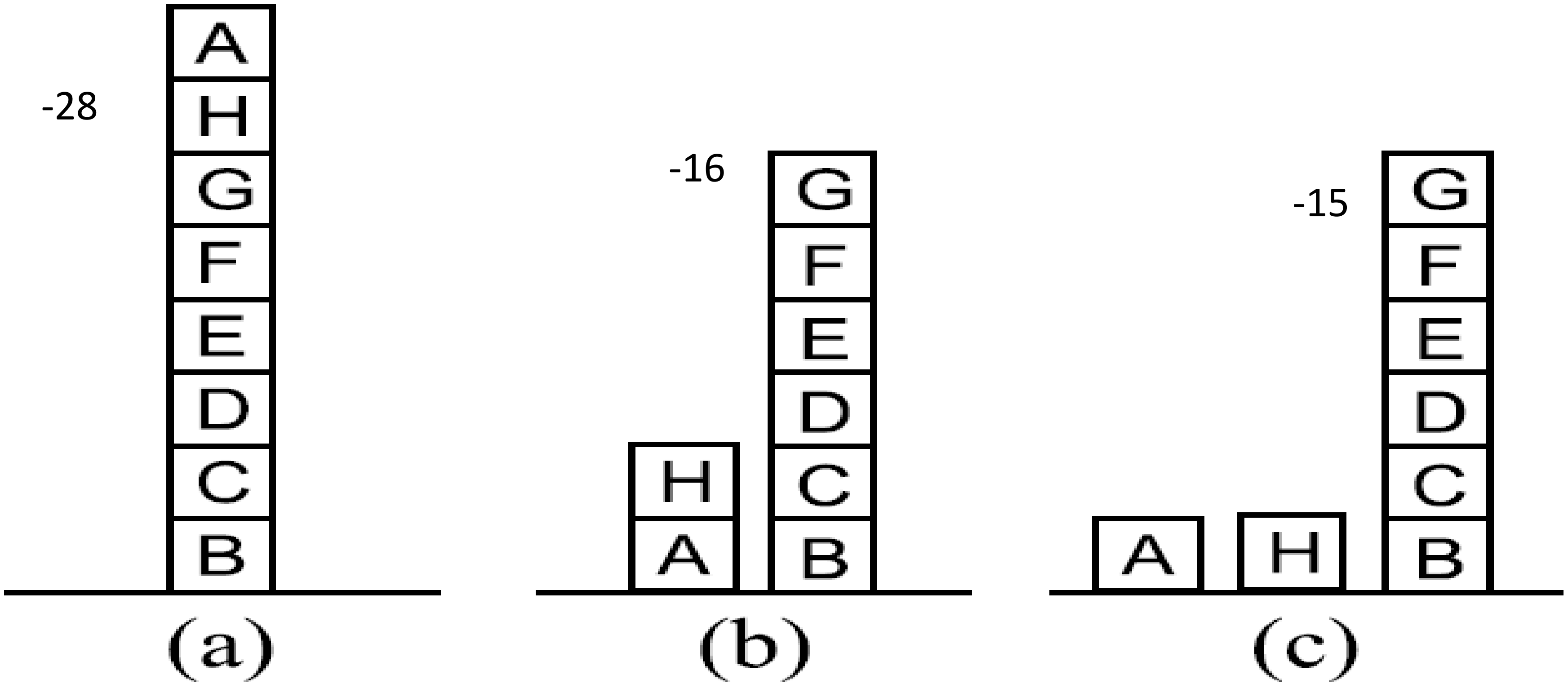
Global: Add one point for every block in correct support structure, subtract one point for every block in existing support structure.



One Possible Moves



Three Possible Moves



SIMULATED ANNEALING

- A variation of hill climbing in which, at the beginning of the process, some downhill moves may be made.
- To do enough exploration of the whole space early on, so that the final solution is relatively insensitive to the starting state.
- Lowering the chances of getting caught at a local maximum, or plateau, or a ridge.
- The term **simulated annealing** derives from the roughly analogous physical process of **heating** and then **slowly cooling** a substance to obtain a strong crystalline structure.
    - The simulated annealing process lowers the temperature by slow stages until the system ``freezes" and no further changes occur.

SIMULATED ANNEALING

- Probability of transition to higher energy state is given by function:
  - $P = e^{-\Delta E/kT}$

  Where ΔE is the positive change in the energy level

  T is the temperature

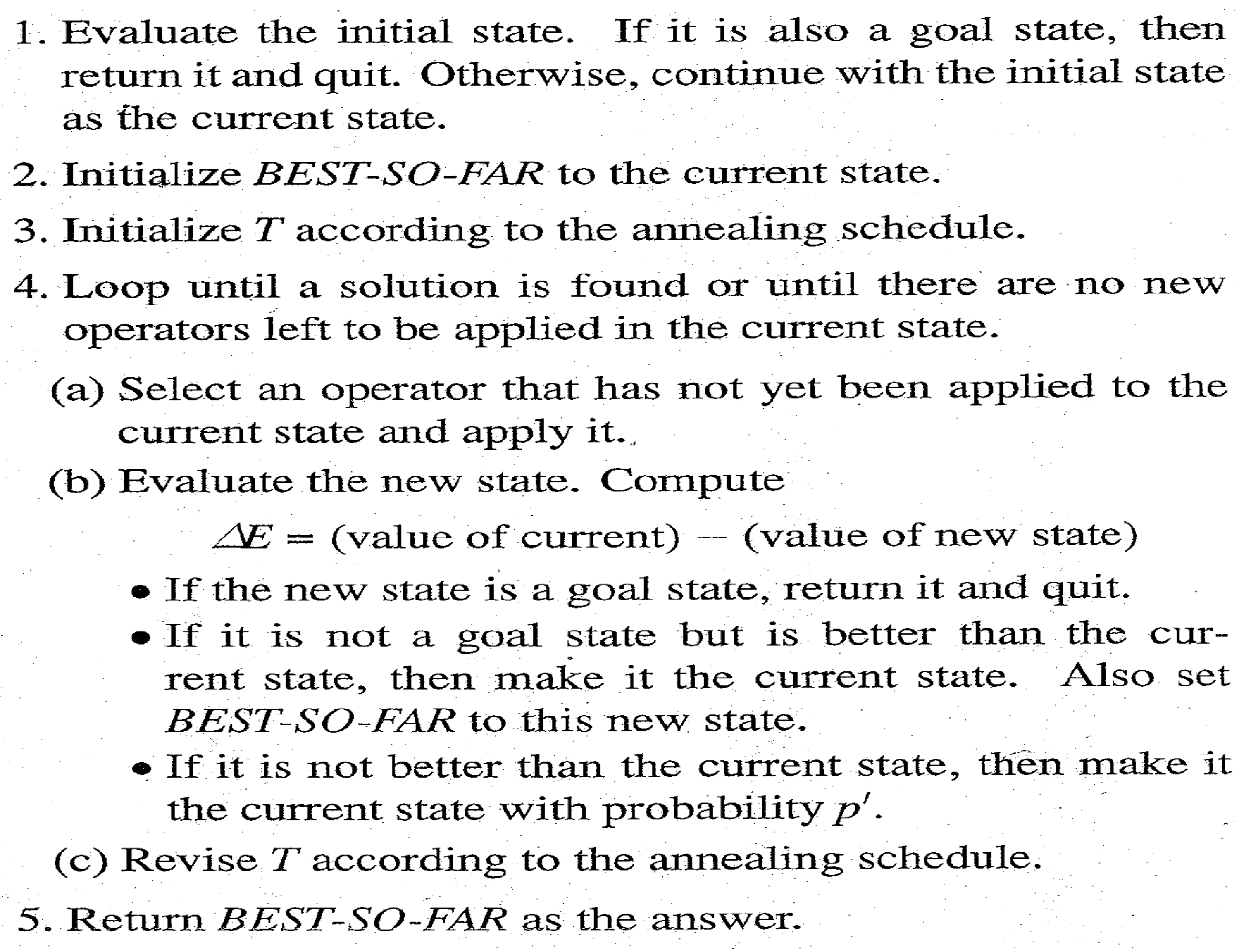
  K is Boltzmann constant.

  Suppose k=1,

  $P' = e^{-\Delta E/T}$

  Annealing schedule: if the temperature is lowered sufficiently slowly, then the goal will be attained.

Algorithm : Simulated Annealing



1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.
2. Initialize *BEST-SO-FAR* to the current state.
3. Initialize *T* according to the annealing schedule.
4. Loop until a solution is found or until there are no new operators left to be applied in the current state.
   - (a) Select an operator that has not yet been applied to the current state and apply it.
   - (b) Evaluate the new state. Compute

     $$\Delta E = \text{(value of current)} - \text{(value of new state)}$$

     - If the new state is a goal state, return it and quit.
     - If it is not a goal state but is better than the current state, then make it the current state. Also set *BEST-SO-FAR* to this new state.
     - If it is not better than the current state, then make it the current state with probability p'.
   - (c) Revise *T* according to the annealing schedule.
5. Return *BEST-SO-FAR* as the answer.

SIMULATE ANNEALING: IMPLEMENTATION

- It is necessary to select an annealing schedule which has three components:
  - Initial value to be used for temperature
  - Criteria that will be used to decide when the temperature will be reduced
  - Amount by which the temperature will be reduced.

Thank You!!!