Hill Climbing

Hill climbing is a variant of generate-and-test in which feedback from the test procedure is used to help the generator decide which direction to move in the search space.

The test function of a generate-and-test procedure responds with only a yes or no. But if the test function is augmented with a heuristic function that provides an estimate of how close a given state is to a goal state, the generate procedure can exploit it as shown in the procedure below. Such a provision sounds attractive as the computation of the heuristic function can be done at almost no cost at the same time that the test for a solution is being performed. Hill climbing is often used when a good heuristic function is available for evaluating states but when no other useful knowledge is available. For example, suppose you are in an unfamiliar city without a map and you want to get downtown. You simply aim for the tall buildings. The heuristic function is just distance between the current location and the location of the tall buildings and the desirable states are those in which this distance is minimized.

Absolute solutions exist whenever it is possible to recognize a goal state just by examining it. Getting downtown is an example of such a problem. For these problems, hill climbing can terminate whenever a goal state is reached. Only relative solutions exist, however, for maximization (or minimization) problems where there is no *a priori goal* state, such as the travelling salesman problem.

Simple Hill Climbing

The simplest way to implement hill climbing is as follows:

Algorithm: Simple Hill Climbing

1. Evaluate the initial state. If it is also a goal state, then return 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 and quit.
        ii. If it is not a goal state but it is better 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.

The use of an evaluation function as a way to inject task-specific knowledge into the control process is the key difference between this algorithm and generate-and- test algorithm. It is the use of such knowledge that makes this and the other methods discussed in the rest of this unit *heuristic* search methods, and it is that same knowledge that gives these methods their power to solve some otherwise intractable problems.

The appearance of relatively vague question like, "Is one state better than another?" is quite noticeable in these algorithms. Hence a precise definition of better must 'be provided. In some cases, it means a higher value of the heuristic function while in others the least. Hardly matter which, until a particular hill-climbing program is consistent in its interpretation.

To see how hill climbing works, let's return to the puzzle of the four colored blocks. To solve the problem, we first need to define a heuristic function that describes how close a particular configuration is to being a solution. One such function is simply the sum of the number of different colors on each of the four sides. A solution to the puzzle will have a value of 16. Next we need to define a set of rules that describe ways of transforming one configuration into another. Actually, one rule will suffice. It says simply pick a block and rotate it 90 degrees in any direction. Having provided these "definitions, the next step is to generate a starting configuration. This can either be done at random or with the aid of the heuristic function described in the last section. Now hill climbing can begin. We generate a new state by selecting a block and rotating it. If the resulting state is better, then we keep it. If not, we return to the previous state and try a different perturbation.

Generate-and-Test

Example: coloured blocks

Heuristic: if there are more red faces than other colours then, when placing a block with several red faces, use few of them as possible as outside faces**.**

Hill Climbing

- Searching for a goal state = Climbing to the top of a hill
- Generate-and-test + direction to move.
- Heuristic function to estimate how close a given state is to a goal state.

Simple Hill Climbing

Algorithm

1. Evaluate the initial state.
2. Loop until a solution is found or there are no new operators left to be applied:
   - Select and apply a new operator
   - Evaluate the new state:
     goal → quit
     better than current state → new current state

- Evaluation function as a way to inject task-specific knowledge into the control process.

Example: coloured blocks

- Heuristic function: the sum of the number of different
- colours on each of the four sides (solution = 16).

Steepest-Ascent Hill Climbing (Gradient Search)

- Considers all the moves from the current state.
- Selects the best one as the next state.

Algorithm
Evaluate the initial state.
Loop until a solution is found or a complete iteration produces no change to current state:
- SUCC = a state such that any possible successor of the current state will be better than SUCC (the worst state).
- For each operator that applies to the current state, evaluate the new state:
  goal → quit
  better than SUCC → set SUCC to this state
- SUCC is better than the current state → set the current state to SUCC.

Hill Climbing: Disadvantages

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
The orientation of the high region, compared to the set of available moves, makes it impossible to climb up. However, two moves executed serially may increase the height.

Ways Out

- Backtrack to some earlier node and try going in a different direction.
- Make a big jump to try to get in a new section.
- Moving in several directions at once.
- Hill climbing is a local method:
  - Decides what to do next by looking only at the "immediate" consequences of its choices.
  - Global information might be encoded in heuristic functions.
- Can be very inefficient in a large, rough problem space.
- Global heuristic may have to pay for computational complexity.
- Often useful when combined with other methods, getting it started right in the right general neighbourhood.