**Hill Climbing Simulated Annealing and Best First Search**

**Steepest Ascent Hill Climbing**

This differs from the basic Hill climbing algorithm by choosing the best successor rather than the first successor that is better. This indicates that it has elements of the breadth first algorithm.

The algorithm proceeds

Steepest ascent Hill climbing algorithm

1 Evaluate the initial state

2 If it is goal state Then quit

otherwise make the current state this initial state and proceed;

3 Repeat

set Target to be the state that any successor of the current state can better;

for each operator that can be applied to the current state

apply the new operator and create a new state

evaluate this state

If this state is goal state Then quit

Otherwise compare with Target

If better set Target to this value

If Target is better than current state set current state to Target

Until a solution is found or current state does not change

Both the basic and this method of hill climbing may fail to find a solution by reaching a state from which no subsequent improvement can be made and this state is not the solution.

Local maximum state is a state which is better than its neighbours but is not better than states faraway. These are often known as foothills. Plateau states are states which have approximately the same value and it is not clear in which direction to move in order to reach the solution. Ridge states are special typesw of local maximum states. The surrounding area is basically unfriendly and makes it difficult to escape from, in single steps, and so the path peters out when surrounded by ridges. Escape relies on: backtracking to a previous good state and proceed in a completely different direction--- involves keeping records of the current path from the outset; making a gigantic leap forward to a different part of the search space perhaps by applying a sensible small step repeatedly, good for plateaux;

applying more than one rule at a time before testing, good for ridges.

None of these escape strategies can guarantee success.

**Example**

Consider the letter blocks in figs 3.1 and 3.2 ; the rules are as before only one block can be moved at a time and the table is infinite in size. A useful heuristic might be add one for every block that is resting on the required block and subtract one for every block that is not. Sum over all towers on the table. This heuristic leads to a local maximum and is not helpful . However if the whole structure supporting any block is examined we get a more pwerful heuristic function and it enables the algorithm to reach the correct goal state.

**SIMULATED ANNEALING**

This is a variation on hill climbing and the idea is to include a genral survey of the scene to avoid climbing false foot hills. The whole space is explored initailly and this avoids the danger of being caught on a plateau or ridge and makes the procedure less sensitive to the starting point. There are two additional changes; we go for minimisation rather than creating maxima and we use the term objective function rather than heuristic. It becomes clear that we are valley descending rather than hill climbing. The title comes from the metallurgical process of heating metals and then letting them cool until they reach a minimal energy steady final state.

The probability that the metal will jump to a higher energy level is given by

p = e-[[Delta]]E/kT

where k is Boltzmann's constant.

The rate at which the system is cooled is called the annealing schedule.

[[Delta]]E is called the change in the value of the objective function and kT is called T a type of temperature. An example of a problem suitable for such an algorithm is the travelling salesman.

The SIMULATED ANNEALING algorithm is based upon the physical process which occurs in metallurgy where metals are heated to high temperatures and are then cooled. The rate of cooling clearly affects the finished product. If the rate of cooling is fast, such as when the metal is quenched in a large tank of water the structure at high temperatures persists at low temperature and large crystal structures exist, which in this case is equivalent to a local maximum. On the other hand if the rate of cooling is slow as in an air based method then a more uniform crystalline structure exists equivalent to a global maximum.The probability of making a large uphill move is lower than a small move and the probability of making large moves decreases with temperature. Downward moves are allowed at any time.

The implementation of the simulated annealing algorithm requires an annealing schedule which has three components. The first component is temperature; the setting of the temperature is critical and is generally chosen so that the average value of the other parameters gives a p of 0.5. As the value of T approaches zero so the possibility of accepting a worse state also goes to zero and the algorithm becomes simple hill climbing. The second is when should the temperature be reduced which affects the changes that are introduced. The third is the amount of the change in temperature at each change. A fourth component concerns when to quit from this process.

The three major differences between this algorithm and hill climbing are:

the annealing schedule must be maintained;

moves to worse states are allowed, more worse moves are allowed initally;

previous good states are remembered, to use as safe havens in case of subsequent disaster.

This algorithm can be implemented in the field of neural networks.

**The SIMULATED ANNEALING algorithm**

1 Evaluate the initial state

2 If it is goal state Then quit

otherwise make the current state this initial state and proceed;

3 Make BESTSOFAR to current state

4 Set T from the annealing schedule

5 Repeat

select an operator that can be applied to the current state

apply the new operator and create a new state

evaluate this new state

find [[Delta]]E = difference between the values of current and new states

If this new state is goal state Then quit

Otherwise compare with the current state

If better set BESTSOFAR to the value of this state and make the current the new state

If it is not better then make it the current state with probability p'. This involves generating a random number in the range 0 to 1 and comparing it with a half, if it is less than a half do nothing and if it is greater than a half accept this state as the next current be a half.

Revise T in the annealing schedule dependent on number of nodes in tree

Until a solution is found or no more new operators

6 Return BESTSOFAR as the answer

**Best First Search**

So far we have considered only two systematic searches, depth first and breadth first. Now best first is a combination of each is considered.

Depth first is good because a solution can be found without computing all nodes and breadth first is good because it does not get trapped in dead ends. The best first search allows us to switch between paths thus gaining the benefit of both approaches. At each step the most promising node is chosen. If one of the nodes chosen generates nodes that are less promising it is possible to choose another at the same level and in effect the search changes from depth to breadth. If on analysis these are no better then this previously unexpanded node and branch is not forgotten and the search method reverts to the descendants of the first choice and proceeds, backtracking as it were. See fig 3.3 the node A generates B C and D of which D is the best, where best is least. E and F are now generated and are worse according to the heuristic function and so B is considered. Thus nodes G and H are generated and evaluated give worse values so the path through D to E is reconsidered. The better of E and F are considered and E generates I and J giving another value of 1 perhaps a good trail.

This process is very similar to steepest ascent, but in hill climbing once a move is chosen and the others rejected the others are never reconsidered whilst in best first they are saved to enable revisits if an impasse occurs on the apparent best path. Also the best available state is selected in best first even its value is worse than the value of the node just explored whereas in hill climbing the progress stops if there are no better successor nodes. The best first search algorithm will involve an OR graph which avoids the problem of node duplication and assumes that each node has a parent link to give the best node from which it came and a link to all its successors. In this way if a better node is found this path can be propagated down to the successors. This method of using an OR graph requires 2 lists of nodes

OPEN is a priority queue of nodes that have been evaluated by the heuristic function but which have not yet been expanded into successors. The most promising nodes are at the front.

CLOSED are nodes that have already been generated and these nodes must be stored because a graph is being used in preference to a tree.

Heuristics

In order to find the most promising nodes a heuristic function is needed called f' where f' is an approximation to f and is made up of two parts g and h' where g is the cost of going from the initial state to the current node; g is considered simply in this context to be the number of arcs traversed each of which is treated as being of unit weight. h' is an estimate of the initial cost of getting from the current node to the goal state. The function f' is the approximate value or estimate of getting from the initial state to the goal state. Both g and h' are positive valued variables.

Best First

The Best First algorithm is a simplified form of the A\* algorithm. From A\* we note that

f' = g+h' where g is a measure of the time taken to go from the initial node to the current node and h' is an estimate of the time taken to solution from the current node. Thus f' is an estimate of how long it takes to go from the initial node to the solution. As an aid we take the time to go from one node to the next to be a constant at 1.

**Best First Search Algorithm**

1 Start with OPEN holding the initial state

2 Repeat

pick the best node on OPEN

generate its successors

For each successor Do

If it has not been genrated before evaluate it add it to OPEN and record its parent

If it has been generated before change the parent if this new path is better and in that case update the cost of getting to any successor nodes

3 Until a goal is found or no more nodes left in OPEN

Commentary on the A\* algorithm

We can make some interesting remarks about this algorithm by considering the role of the function g. By making g=0 we are interested only in finding a solution regardless of cost. If cost matters then g is not constant and truly reflects the price of getting from node to node. By including g in f' we can choose the next node to be expanded as the best path so far rather than as the nearest node to the solution. If h' is a perfect estimator of h then A\* wil immediately converge to the goal without search a direct solution. If h' is 0 then the search is controlled by g and if g is also zero then the search is random whereas if g=1 then the search becomes breadth first and is guaranteed to find a solution if one exists.

If h' is a perfect estimator of h then A\* will converge to the goal without a search. The better h' is the quicker the goal is reached. If g is 0 the search is random and if g is 1 then it is breadth first. If h' never overestimates h then A\* is guaranteed to find the optimal path if one exists.

In fig 3.4 where all arcs count as 1 and f' is the sum of g and h', B has the lowest value of f' 4=3+1. By expanding B we get E, 5 =3+2 an extra arc, but the same as C so E is expanded and then Fis generated, 6=3+3 another arc and now worse than C. Although we have wasted time in going down B we are still O K as we have reached C and the value of h' at B did not exceed that at F.

In fig 3.5 we expand B as a first step. After expanding E and then F and then G we converge on a solution of cost 4 as h' becomes 0 at G. But suppose that there is a direct path from D to a solution then we will never reach this solution of cost 2 as the estimate on D was 5 too high. Thus worse solutions will be explored without realising that D was the best because the value of h' was such an overestimate. Practically we cannot achieve the fact that h' must never over estimate h unless we make h'=0 so it is not much help in this form as it takes us back to a breadth first.