1 Greedy algorithm

The greedy algorithm is a simple and fast algorithm that is commonly used to solve optimization problems. It always makes the best/greedy choice at present, and hopes to get the optimal solution to the problem, at last, based on each greedy choice. More specifically, it is to traverse all reachable nodes based on the current node and select the nearest node as the next node. The basic idea is to traverse all reachable next nodes from a node, select the nearest node as the next node, then mark the current node has passed, the next node as the current node, repeat the greedy strategy, and so on Until all nodes are marked as having gone the end of the node.

2 Hill-climbing algorithm

Hill-climbing algorithm is a locally preferred method. It is an improvement of depth-first search. It uses feedback information to help generate the solution. Each time the algorithm chooses an optimal solution from the solution space of the current solution as the current solution until it reaches a local optimal solution.

3 The basic idea of ​​Simulated annealing algorithm

Simulated annealing is a random search algorithm based on the principle of annealing. It differs from the hill-climbing method. It selects the next node both based on the difference after choosing the node and the probability. When the current solution is increased based on selecting the neighbor node, it will accept the improved solution as a new current solution. Contradictory, when compared with the current solution, the worse solution is accepted as a new current solution with a certain probability. The simulated annealing algorithm has been theoretically proved to be a global optimization algorithm that converges to the global optimal solution with a probability of 1.

3 Dynamic programming algorithm

Its basic idea is to decompose the problem to several sub-problems. First solve the sub-problems, and then obtain the solution of the original problem from the solutions of these sub-problems.

Let V' denote a set of nodes. Assuming starting from the node s, d (i, V') represents the minimum cost for the current arrival of node i through all nodes in the V' set.

(1) When V’ is an empty set, d(i, V’) means that it returns to s from node i without going through any node, as shown in the figure above. In this case d(i, V’)=Cis (is the distance from node i to node s).

(2) If V' is not empty, it is the optimal solution to the subproblem. You must try each one in the V’ set of cities and find the optimal solution--d(i, V’)=min{Cik + d(k, V’-{k})}. Cik represents the distance between your chosen node and node i. d(k, V’-{k}) is a sub-problem.