Artificial Intelligence - 2017
Local Search Algorithms

Prepared By;
Vijani Supeshala Piyawardana | BSc (Hons) Computer Science |
Demonstrator - COMP3004L, COMP3008L - UCD |
NSBM Green University Town

Local Search Algorithms and Optimisation

Local Search

- To search faster
- Do not keep track of many paths
- Typically keeps only one state in memory and move to neighbours of that state
- Advantage:
    - Very little memory
    - Can find solution in immense or infinite state spaces where optimal strategies would not stand a chance
- Can be used for optimization as well as standard search

Optimization Problem

- A problem we don’t know a solution, but we have an objective function to weigh states.

Optimization
(meaning :- optimal, optimize, optimization)

- To find state with the lowest weight.
- find optimal configuration is the algorithm goal, (ex: TSP-Travelling Sales Problem)
- Or at least one, with a suitably low weight.

Iterative Improvements

- Many local search algorithms use iterative improvements
- In many optimization problems, path is irrelevant, the goal state itself, is the solution (ex: n-queen problem)
- We can have a solution and have improvements to that solution.
- Like TSP, we do not have to keep the track of path to the solution. Local search is ideal for such problems.
- But, in “n-puzzle” problem, we need to know how to reach the goal. Final state do not give any information about the path. Local search is not good for such problems.

To the problems which the path cost is not important, Local search is useful.

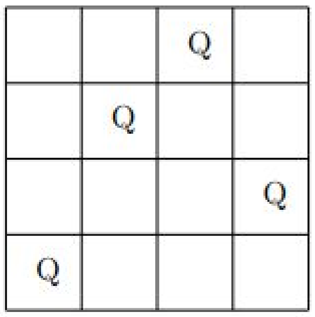
N-queen problem

- We have nXn chessboard and n queens
- We have to put n queens on the board that, none of the queens attack each other.
- Here, we know the final configuration, we know the solution.
- So, we can define n-queen problem as an optimization problem.

Using Iterative methods

- Start with a solution
- Improve it towards a good solution

Goal of n-queen : put n queens on an nXn board with no 2 queens on the same row, column or diagonal
Start with any state.
(h is the weight of the state/no of conflicts, need to find a state with low weight)



1: Illustration of the 4-queens problem

Take a 4X4 chess board and 4 queens. Our job is to find arrangement where no two queens attack each other also no two queens in same column.

https://www.youtube.com/watch?v=0DeznFqrgAI

A function

```
bool solveNQueen( int board[N][N], int col ) {
      if( col >= N) {
            Return true;
      }
      for(int i=0; i<N; i++){
            if(isSafe(board, i, col))
            {
                    board[i][col] = 1;
                    if(solveNQueen(board, col+1 ))
                         return true;
                    board[i][col]=0;
          }
}
```

1. Hill Climbing Search - a local search method
   (or gradient ascent/descent)

- Idea : we start with the current configuration, we move to a new configuration to maximize the value
  Ex: in n-queens, our objective is to minimize the no of conflicts

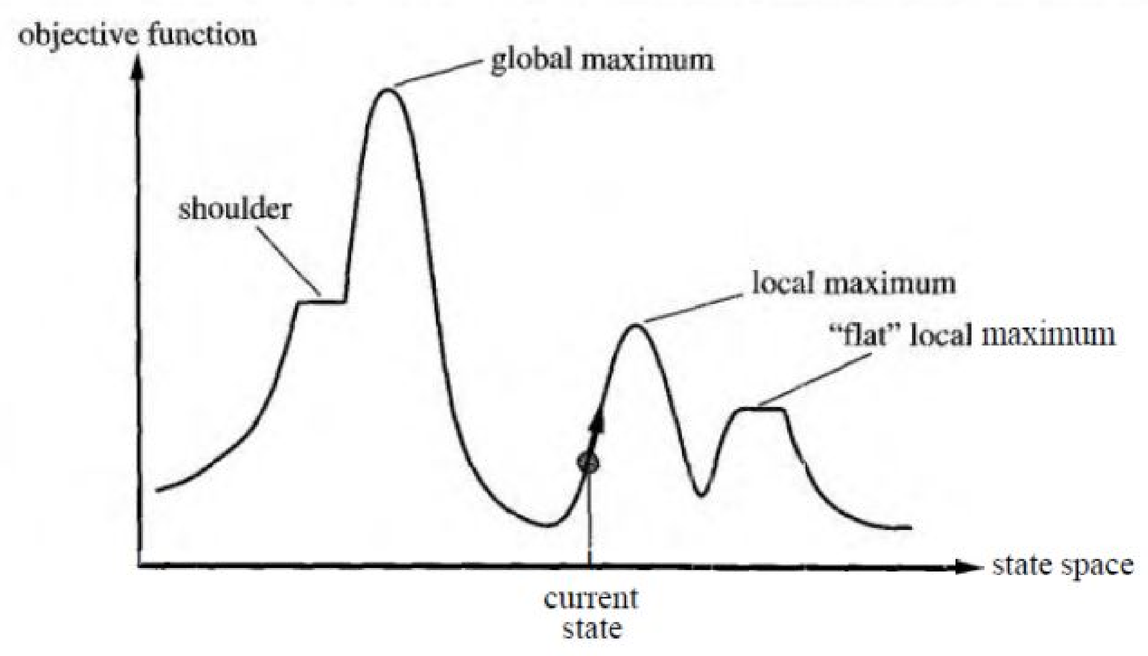
- Iteratively maximize “value” of current state by replacing it by successor state that has highest value, as long as possible.
- We go to the neighbour with minimum no of conflicts, the neighbour which is best, we continue until we got to a state where all the neighbours are worse than that.

- Hill climbing search keeps only 1 state in memory, tiny memory use, simple.
- Only go uphill (or only downhill) as much as you can. That means, we can talk about the hill climbing of minimizing value function or maximizing value function.
- One moves from a state by generating all its neighbours and picking the best one.

Optimization & landscape

Consider the above semantic diagram of state space of hill climbing. We’re starting at a particular position and we want to move to a neighbour, whose value is larger.
The optimum solution is at Global Maximum.

Problems we face when we use Hill climbing



1. Local Maximum

Start from 1, move until we get here A whose neighbours are all worse than this. This is the solution that we have obtained and this is also happened to be an optimum solution.

But suppose you start from 2, move to best neighbour, best neighbour….. And you'll reach B where all neighbours are worse, but it's not the optimum solution to the problem.

So, we can get stuck at local maximum, instead of landing up on the global optimum. Depending on initial state, you may get stuck in local maximum.

2. Flat Local Maximum
3. Local Minima
4. Shoulder
5. Global Minimum

Hill Climbing variations

1. Random restart - if you get stuck, start again from a different state.

   If your chances of succeeding in one attempt are p, then you expect it to take you approximately 1/p starts on average to get a solution.
   Greatly increases the chances of solution even though;

   If p is tiny, it might take a while.
   It multiplies the complexity.

2. Stochastic hill climbing - pick a successor at random among uphill ones

3. First choice hill climbing - generate random successors until you find one that is uphill.

Both 2 and 3 mitigate the local maximum problem.
3 makes it possible to apply hill-climbing to infinite-dimensional problems.

2. Local Beam Search Technique

- Basically a variation of hill-climbing search technique.
- It is a hill climbing variation with k-states instead of 1-state.
- So keep track of k-states.
  - Initially we have k randomly selected states.
  - Next, we'll determine/generate all successors/neighbours of all of k-states.
  - If any of successors is the goal then search is finished.
  - else , select k-best from successors and repeat.
    (Here, "best" means the successor with the highest objective function value.)

- Major difference with random-restart hill climbing search:
  - Information is shared among k search threads (interplay)

- Tricky issues;
  1. Who is k?
     Pick k out of the list at random but with a probability that depends on their quality.
  2. Can suffer from lack of diversity. How to prevent diversity from quickly disappearing? The answer is :- Stochastic beam search.

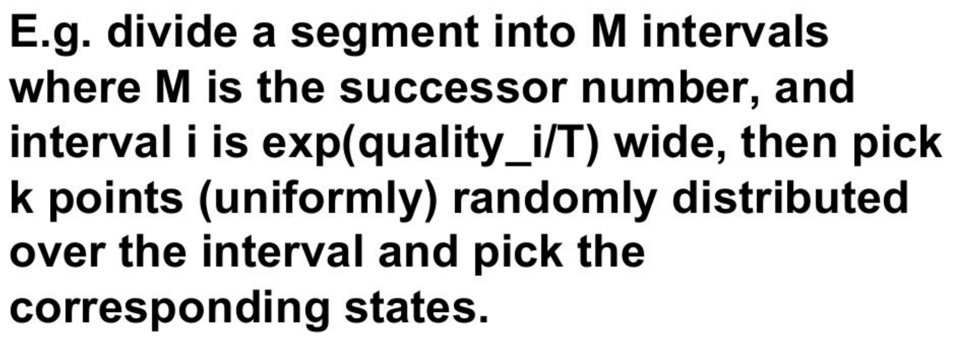
3. Stochastic beam search

Choose k successors proportional to state quality. That means;

Only points directly next to the current point are looked at. k points are chosen randomly at the beginning of the algorithm. At each iteration, all points adjacent to the current points are found, of these, only k points are chosen to be the next data set.

k points are selected based on the value of the points. If a point is higher, it is more likely to be selected than a lower point.

It is important to note that the algorithm can select lower points in order to manoeuvre out of local maximum. This way, the search usually finds the global maximum.



E.g. divide a segment into M intervals where M is the successor number, and interval i is exp(quality_i/T) wide, then pick k points (uniformly) randomly distributed over the interval and pick the corresponding states.

- Stochastic Beam Search resembles (asexual) natural selection.
- An individual has a better chance of surviving if it is fitter than the others.
- This doesn't guarantee it will survive and the less fit individuals won't.
- Less promising solutions are kept around, and their successors still have a chance of looking more promising.
- This alleviates many of the problems of standard beam search, because it makes it less likely for the k-states to converge quickly.
- However it adds a tricky free parameter (T).
- Too high and your search is random.
- Too low and it becomes a very expensive hill climbing.

-

Two other approaches in problem solving, which are **Metaheuristic Algorithms**

1. Simulated annealing
2. Genetic algorithms

There are sometimes called **metaheuristics** as they are general problem solving strategies, that can be applied to many different problems.

-

Combinatorial Optimization Problems

Combinatorial : combination+al (meaning is something like that)

A minimization or maximization problem that is specified by a set of problem instances.
An instance of a combinatorial optimization problem is a pair **(S,f)** where;

The solution space or state space **S** denotes the finite set of all possible solutions and the cost function **f** is a mapping defined as;

$$f : S \dashrightarrow R$$

In the case of minimization, the problem is to find a solution $i_{opt} \in S$ which satisfies;

$$f(i_{opt}) \leq f(i) \quad \forall \quad i \in S$$

Or, in the case of maximization, the problem is to find a solution $i_{opt} \in S$ which satisfies;

$$f(i_{opt}) \geq f(i) \quad \forall \quad i \in S$$

Let f_{opt} denote the optimal cost and S_{opt} the set of optimal solutions.

Example : TSP - Travelling Salesman Problem

The Travelling Salesman starts tour with a given city/edge, visits each of the specific group of edges/cities and returns to the original point of departure.
TSP is the shortest walk in a circuit provided that it passes from all cities/vertices only once.

https://www.youtube.com/watch?v=7B8Sx_nAxLk
https://www.youtube.com/watch?v=HWHZAtQI1vI

- Given **n** cities and an **nXn** matrix d_{pq} whose elements denote the distance between each pair **p** and **q** of the n cities.
- A tour is defined as a closed path visiting each city exactly once.
- The problem is to find a tour of minimal length.
- A state for this problem is given by a cyclic permutation.
- Cyclic permutation is;

$$\pi = (\pi(1) \ldots \pi(n))$$

Where $\pi(k)$ denotes the successor of city **k** with;

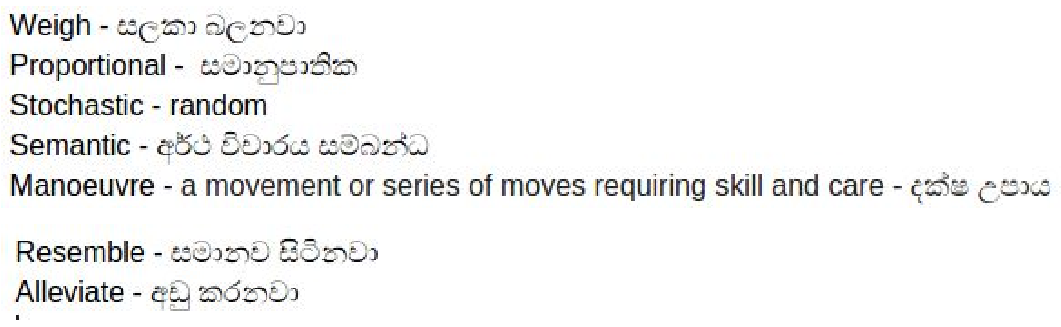
$$\pi'(k) \neq k \; \forall \; l \in \{1,\ldots,n-1\}$$

$\pi^n(k) \neq k$

Algorithm

```
INPUT An integer $n \geq 3$ and an nXn distance matrix non-negative integers
OUTPUT The shortest tour of n cities
Begin
for all cyclic permutations of $\pi$ of (1,2,...,n) {
        calculate cost
        If (cost < min) {
                min := cost
                best_tour := $\pi$
        }
}
output best_tour
end
```

Vocabulary



Weigh - සලකා බලනවා
Proportional - සමානුපාතික
Stochastic - random
Semantic - අර්ථ විචාරය සම්බන්ධ
Manoeuvre - a movement or series of moves requiring skill and care - දක්ෂ උපාය

Resemble - සමානව සිටිනවා
Alleviate - අඩු කරනවා

Past exam questions

2014 Question 3

(a) In the context of combinatorial optimization, what is meant by a local search algorithm and a neighbourhood function.

Local search algorithm
Neighbourhood function

(b) Give an appropriate neighbourhood function for the travelling salesman problem.

2016 Question 3

(a) I. What is meant by a metaheuristic algorithm?

A general purpose problem solving framework which is not specific to a particular application domain.
(further explanation : metaheuristics algorithms are general problem solving strategies, that can be applied to many different problems)

Next : Simulated Annealing and Genetic Algorithms PDF