Heuristic Optimisation

Part 4: Local search

Sándor Zoltán Németh

http://web.mat.bham.ac.uk/S.Z.Nemeth

s.nemeth@bham.ac.uk

University of Birmingham

Overview

- Local search algorithm
- Local search for SAT
- Local search for TSP
- Local search for NLP

Local search versus exhaustive search

- Exhaustive search enumerates all the possible solutions
- Local search focuses on a neighbourhood of some particular solution

Algorithm of local search

1. Pick a solution from the search space. Define this as the current solution.
2. Apply a transformation to the current solution to obtain a new solution.
3. If the new solution is better than the current one, replace current solution by the new solution.
4. Repeat steps 2.-3. until no improvement is possible.

The transformation in step 2

Can range from

returning a random solution from the whole search space
(might be worse than simple enumeration)

to

returning the current solution
(no change)

- Small neighbourhood - quick, but we might get stuck in local optima
- Large neighbourhood -less chance for getting stuck, but takes longer

Local search for SAT

GSAT:

1. Repeat steps 2.-3. MAX_TRIES times:
2. Assign values to $X =< x_1, \ldots, x_n >$ randomly
3. Repeat MAX_FLIPS times:
   If the formula is satisfied, return X
   else make a flip
4. Return “No solution found”

Make a flip

The SAT formula must be given in conjunctive normal form (a conjunction of clauses):

$$(x_1 \vee x_4 \vee x_5) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_6)$$

For each possible bit flip (from the current solution) the decrease in the number of unsatisfied (false) clauses is calculated

The flip with the largest decrease is made (which can actually be an increase!)

Example for GSAT

$$F = F_1 \bigwedge F_2 \bigwedge F_3 \bigwedge F_4$$

$F_1 = x_1 \bigvee x_2$, $F_2 = x_2$, $F_3 = \overline{x}_1 \bigvee \overline{x}_2$, $F_4 = x_3$

START: $(x_1, x_2, x_3) = (0, 0, 0)$

	F_1	F_2	F_3	F_4	
flip	0	0	1	0	decrease
x_1	1	0	1	0	$+1$
x_2	1	1	1	0	$+2$
x_3	0	0	1	1	$+1$

FLIP x_2

$(x_1, x_2, x_3) = (0, 1, 0)$

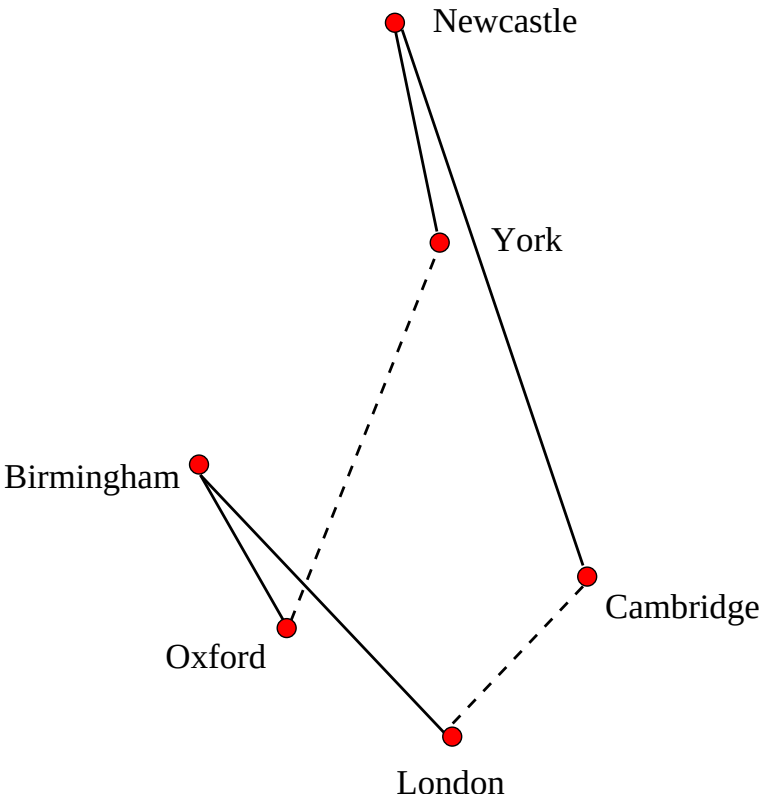
	F_1	F_2	F_3	F_4	
flip	1	1	1	0	decrease
x_1	1	1	0	0	-1
x_2	0	0	1	0	-2
x_3	1	1	1	1	$+1$

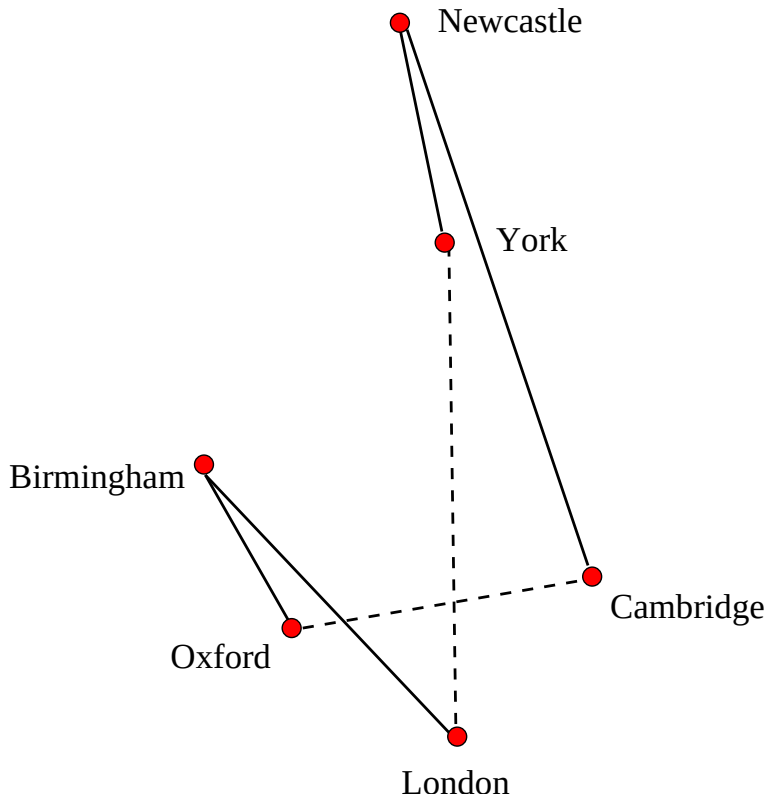
FLIP x_3

SOLUTION: $(x_1, x_2, x_3) = (0, 1, 1)$

Algorithm 2-opt for TSP

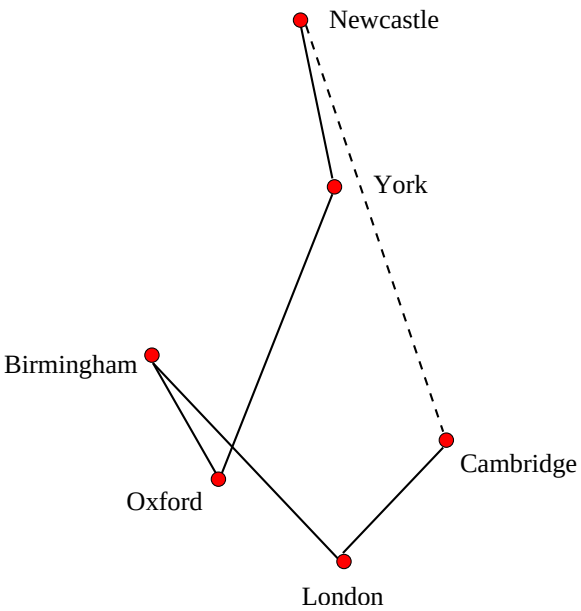
The transformation consists of changing two non-adjacent edges in the tour

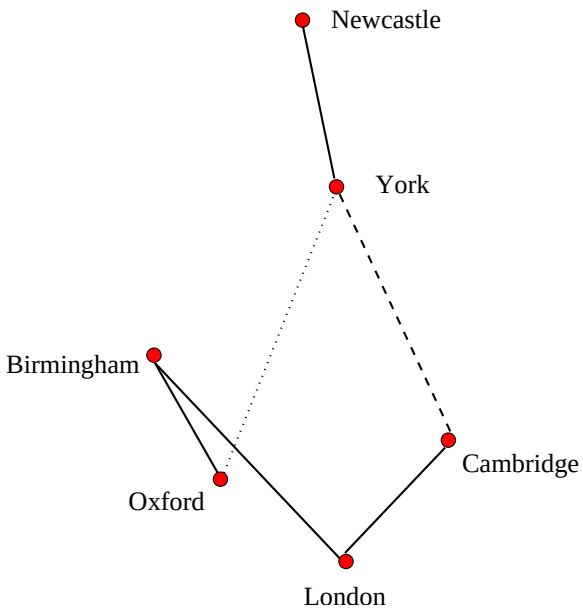


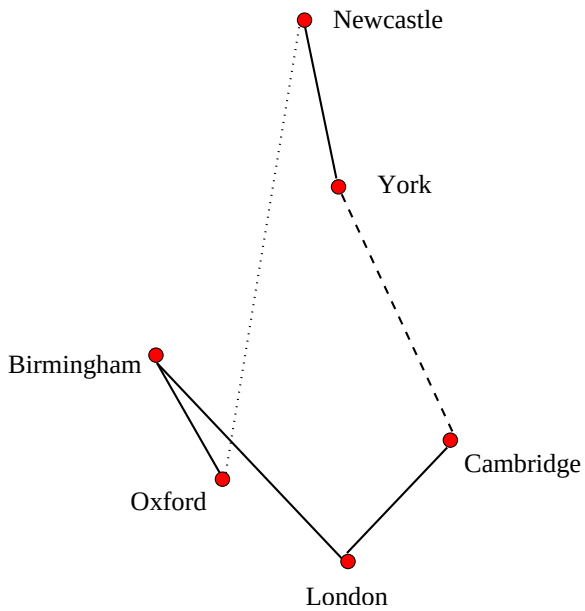


The algorithm should be restarted several times for better results!

A δ-path for TSP







In a δ-path all nodes appear exactly once with the exception of the last one which appears twice

By replacing one edge a δ path can be

- repaired to a legal tour, or
- modified to another δ path (switch)

The Lin-Kernighan algorithm for TSP

1. Generate a tour T and store it as best.

2. For each node k on the tour T and for each edge (i, k)

   If there is a node j with $cost(i, j) \leq cost(i, k)$ then create the δ-path p where edge (i, j) replaces (i, k)

   (a) Repair p into a tour T_1, store it, if better than best.

   (b) If there is a switch from p to p_1 with $cost(p_1) \leq cost(T_1)$, replace p with p_1.

   (c) Repeat from (a).

3. Return best.

It is the best algorithm for large fully-connected symmetric TSP problems, as it is fast and can find near-optimal solutions.

Local search for NLP

The majority of numerical optimisation algorithms for the NLP are based on local search

Given $f : \mathbb{R}^n \to \mathbb{R}$
optimise $f(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$
subject to $x \in F \subset S \subset \mathbb{R}^n$

The search space $S \subset \mathbb{R}^n$:

$$l_i \leq x_i \leq u_i, \ 1 \leq i \leq n$$

The feasible region $F \subset S$:

$$g_j(\mathbf{x}) \leq 0, \ 1 \leq j \leq q$$

$$h_j(\mathbf{x}) = 0, \ q < j \leq m$$

The gradient method of minimisation

- We use the gradient of the evaluation function at a particular candidate solution
- We use this information to direct the search toward improved solutions (in a local sense)
- It is essential to find the right step size to guarantee the best rate of convergence over the iterations

The gradient method cont'd

The maximum decrease occurs in the direction of the negative gradient $-\nabla f(\mathbf{x})$

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]_{\mathbf{x}}$$

Steepest descent:

1. Start with an initial candidate solution $\mathbf{x_1}$. Let $k = 1$.
2. Generate a new solution
   $\mathbf{x_{k+1}} = \mathbf{x_k} - \alpha_k \nabla f(\mathbf{x_k})$
3. If $\|\mathbf{x_{k+1}} - \mathbf{x_k}\| > \varepsilon$ let $k = k + 1$ and go to step 2.

Newton's method

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (H(f(\mathbf{x_k})))^{-1} \nabla f(\mathbf{x_k})$$

The Hessian matrix of f:

$$H(f(\mathbf{x})) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Calculating the inverse of the Hessian matrix can be very time-consuming!

Summary

The algorithms that work on complete solutions can be stopped at any time, they give a potential solution, provided the chosen algorithm matches the problem

For a smooth, unimodal evaluation function the gradient method is great

For a combinatorial problem (TSP) there are many local operators that take advantage of characteristics of the problem domain

But what if you don't know much about the problem or it does not fit within the problem class of any of these algorithms?

Recommended reading

Z. Michalewicz & D.B. Fogel
How to Solve It: Modern Heuristics

Section 3.2. Local search