TSP Heuristics

Luis Goddyn, Math 408

There are two types of heuristic methods for finding optimal TSP tours. *Scratch methods* produce an initial tour which is hopefully fairly close (say 10%) from being optimal. *Improvement methods* (sometimes called *post-optimization methods*) attempt to improve a given tour. A well accepted general strategy is to begin with many scratch solutions, and to post-optimize each of these to a reasonable extent. The best of these is selected to be the final tour.

We assume throughout that we are solving an instance (V, d) where V is a set of n *nodes* and the distance function d is non-negative valued and satisfies the triangle inequality. The length of a tour T for (V, d) is denoted $\ell(T)$, while the length of an optimal tour is written ℓ_{opt}.

1 Scratch Methods

1.1 Nearest Neighbour

Initially we randomly select a node $v_1 \in V$. At any stage we have a list of distinct nodes $(v_1, v_2, \ldots, v_k)$. If $k = n$ we output the tour $T = (v_0, v_1, \ldots, v_n, v_1)$ and STOP. Otherwise we extend the list by selecting v_{k+1} to be a closest node to v_k not yet in the list.

This greedy-type heuristic does not perform well in practice. Even if the distance function is non-negative and satisfies the triangle inequality, it is not hard to find examples for which $\ell(T)/\ell_{\text{opt}} = (\log_2 n)/3$. On the other hand, it is known that this ratio never exceeds $(\log_2 n)/2$.

1.2 Insertion Heuristics

Three variations of the insertion heuristic are in common use: *nearest, furthest* and *cheapest* insertion. All follow the same strategy, differing only in the details. Here is the general strategy.

1. Select two nodes v_1, v_2 and let C be the cycle (v_1, v_2, v_1).
2. At any stage we have a cycle C through k nodes, $2 \le k \le n$. If $k = n$, then output the tour $T = C$ and STOP. Otherwise select a node $v \in V - C$. Insert v into C in the best possible way. That is, of the k places in C into which v may be inserted, choose the one resulting in the shortest length cycle.

The following table specifies the details and the corresponding performance guarantees, for each variation. In the following $d(v, C) = \min\{d(v, x) : x \in C\}$ is the distance from a node v to the set of nodes in C.

Variation	Selection of v_1, v_2 in Step 1	Selection of v in Step 2	Performance Guarantee
Nearest Insertion	Nearest pair of nodes in V	Pick v s.t. $d(v, C)$ is minimum	$\ell(T)/\ell_{\text{opt}} \le 2$
Furthest Insertion	Furthest pair of nodes in V	Pick v s.t. $d(v, C)$ is maximum	$\ell(T)/\ell_{\text{opt}} \le \log_2 n$
Cheapest Insertion	Nearest pair of nodes in V	Pick v s.t. the increase in $\ell(C)$ caused by inserting v is minimum	$\ell(T)/\ell_{\text{opt}} \le \log_2 n$

In practice, furthest insertion consistently outperforms the other two variations. It frustrates theoreticians that this fact is not reflected by the currently known performance guarantees!

1.3 Christofide's Heuristic

This heuristic has the best performance guarantee of all known scratch methods. It also works well in practice. Unfortunately it requires the high overhead of a matching heuristic.

1. Find a minimum weight spanning tree R in (V, d). Let $V_1 \subseteq V$ be those nodes having odd degree in R.
2. Find a minimum weight perfect matching M of V_1.
3. Find an Euler tour S of the Eulerian graph $(V, R \cup M)$.
4. Convert S into a TSP tour T by taking *shortcuts*. That is, we obtain T by deleting from S all but one copy of each vertex in V.

Theorem 1 *Christofide's heuristic produces a tour T for which $\ell(T)/\ell_{\text{opt}} \leq 3/2$.*

Proof. Let T^* be an optimal tour of (V, d). Deleting an edge e from T^* results in a spanning tree of V. Since $d(e) \geq 0$ and R is a minimum weight spanning tree we have $\ell(R) \leq \ell(T^*)$.

Let C by the cycle through V_1 obtained by visiting the vertices in V_1 in the same order as they appear in T^*. By triangle inequality we have $\ell(C) \leq \ell(T^*)$. Since C has even length (why?), C is the disjoint union of two perfect matchings on V_1. Both of these matchings must be at least as long as the minimum length matching M. Thus $\ell(C) \geq 2\ell(M)$, so $l(M) \leq \ell(C)/2 \leq \ell(T^*)/2$. By triangle inequality we have

$$l(T) \leq l(S) = \ell(R) + \ell(M) \leq \ell(T^*) + \ell(T^*)/2 = 3\ell_{\text{opt}}/2.$$

Q.E.D.

2 Improvement Methods

Once we have a tour T, we try to improve on it using methods such as those described bellow.

2.1 k-opt

The idea here is to check some or all $\binom{n}{k}$ k-subsets of edges in T. For each such subset S, we test whether an improved tour can be obtained by replacing the edges in S with k new edges. Such a replacement is called a *k-switch*. One must take care that a k-switch actually results in a tour rather than a set of two or more cycles. A tour with no improving k-switches is said to be *k-optimum*. The simplest strategy is *k-opt*. That is, we successively perform k-switches until a k-optimum tour is obtained.

This heuristic k-opt is usually applied for $k = 2$ or $k = 3$. The problem is that the number of k-subsets grows very quickly with k. Furthermore any k-switch which is made can introduce numerous new k-subsets which have improving k-switches. Indeed if unfortunate choices are made, it can take an exponential number of 2-switches before obtaining a 2-optimum tour!

2.2 Lin-Kernighan

This is essencially a k-opt-type algorithm, but differs from k-opt in two ways. First, the value of k is allowed to vary. Second, when an improvement is found, it is not necessarily used immediately.

A *δ-path* is obtained from a path by adding a new edge from one end-node to one of the other nodes in the path. Thus a δ-path looks like a cycle with possibly a "tail" attached to it. From any δ-path P we may obtain a tour $T(P)$ by replacing this final edge with a new edge joining the endpoint of the resulting path. The heuristic consists of $2n$ *edge-scans*. One for each pair (v, e) where $v \in V$ and $e = vu$ is one of the two edges of the current tour T adjacent with v. The following is a description of Lin-Kernighan taken from an unpublished account by Bill Cook. Please refer to the practice sheet when trying to read it.