An implementation of a symmetric-TSP solver

Matthias Köppe*

September 16, 1997

Abstract

The symmetric TSP can be solved exactly using branch and bound methods. Implemented is both an assignment-problem relaxation with original costs and a 1-tree relaxation with Lagrangean costs, using various branching rules. Moreover, a simple simulated-annealing method is implemented.

1 Branch and bound

Branch and bound methods are enumerative methods that work by breaking up the feasible set into smaller subsets, calculating lower bounds for these subsets and discarding those subsets having too big lower bounds. For a more detailed discussion, cf. [1, chapter 10].

Lower bounds are achieved by solving a relaxed problem. In the `branch.cc` program, we use the assignment problem as a relaxation. It is solved using the Hungarian algorithm. We split a feasible set A into two subsets A_1 and A_2 by selecting one edge and forcing it to be included into all tours of A_1 and to be excluded from all tours of A_2. The edge is selected such that the lower bound of A_2 grows maximally. The tree of feasible subsets can be processed either in depth-first order or in lowest-bound-first order. – Since the lower bounds are very weak, a lot of splits must be done, and the program will require very much memory. On a PC with approx. 100 MB virtual memory, the `cont26.dat` problem could be solved. You can't expect results for larger problems.

The `1tree.cc` program uses a more advanced technique. As a relaxation, minimal 1-trees (that is a spanning tree plus an additional edge) are calculated. This is done using the algorithm of Prim-Dijkstra, applied on the TSP graph with vertex 1 removed; then two edges from vertex 1 are added. Instead of using the original TSP costs, the costs are updated in a Lagrangean fashion; for any choice of the Lagrange vector λ, we receive a valid lower bound. A subgradient iteration procedure is then used to find the best λ (for the strongest lower bound). For details, cf. [1, section 10.3]. Two different branching rules have been implemented, namely the one of Smith and Thompson (1977) and the one of Volgenant and Jonker (1982); they are very similar in their performance. The tree of feasible subsets can be processed either in depth-first order or in lowest-bound-first order. As an upper-bounding device and tour-building heuristics, we are using a simulated-annealing algorithm. – With this implementation, even the `cont101.dat` problem could be solved within approx. 4 minutes on a PC.

*`mkoeppe@csmd.cs.uni-magdeburg.de`

2 Simulated annealing

The `anneal.h` file contains a simple algorithm for general iteration in the configuration space. Two different neighbourhoods of tours are implemented. One simply selects two cities in the tour and exchanges them. The other neighbourhood selects two pairs of adjacent cities $A - B$, $C - D$ and reconnects them as $A - C$, $B - D$; this procedure is known as a LIN2OPT step. There are two acceptance mechanisms available; for local optimization, a move to a neighbour configuration is accepted iff the tour becomes shorter. For simulated annealing, also moves that increase the tour length are accepted with a certain probability. An exponential cooling scheme is used.

At low temperature, the rejection probability is very high, such that movement in the configuration space will be very slow. This problem can be solved using a rejection-free (or "faster-than-the-clock") algorithm; a very simple implementation can be found in `anneal.cc`. It works by calculating the acceptance probabilities of all neighbour configurations, choosing a neighbour according to the effective probability and increasing the clock by an exponentially distributed time step.

References

[1] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem*. Wiley-Interscience series in discrete mathematics. John Wiley & Sons Ltd., Chichester – New York – Brisbane – Toronto – Singapore, 1992.