Various aspects of Travelling Salesman Problems

# Main objective:

Main objective is to find out and describe the various aspects of Travelling Salesman Problems, exact problem definition, detailed background theory, suggested solutions/ methods/algorithms/methodologies, algorithm, Implementation, and findings along with the conclusion.

# Exact problem definition And Detailed background theory:

TSP is a special case of the travelling purchaser problem.

In the theory of computational complexity, the decision version of the TSP (where, given a length *L*, the task is to decide whether the graph has any tour shorter than *L*) belongs to the class of NP-complete problems. Thus, it is possible that the worst-case running time for any algorithm for the TSP increases super polynomially (or perhaps exponentially) with the number of cities.

The problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, a large number of heuristics and exact methods are known, so that some instances with tens of thousands of cities can be solved completely and even problems with millions of cities can be approximated within a small fraction of 1%.

The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing. In these applications, the concept *city* represents, for example, customers, soldering points, or DNA fragments, and the concept *distance* represents travelling times or cost, or a similarity measure between DNA fragments. In many applications, additional constraints such as limited resources or time windows may be imposed.

The origins of the travelling salesman problem are unclear. A handbook for travelling salesmen from 1832 mentions the problem and includes example tours through Germany and Switzerland, but contains no mathematical treatment.

The travelling salesman problem was mathematically formulated in the 1800s by the Irish mathematician W. R. Hamilton and by the British mathematician Thomas Kirkman. Hamilton’s Icosia Game was a recreational puzzle based on finding a Hamiltonian cycle.  The general form of the TSP appears to have been first studied by mathematicians during the 1930s in Vienna and at Harvard, notably by Karl Menger, who defines the problem, considers the obvious brute-force algorithm, and observes the non-optimality of the nearest neighbour heuristic:

We denote by *messenger problem* (since in practice this question should be solved by each postman, anyway also by many travelers) the task to find, for ﬁnitely many points whose pairwise distances are known, the shortest route connecting the points. Of course, this problem is solvable by finitely many trials. Rules which would push the number of trials below the number of permutations of the given points, are not known. The rule that one first should go from the starting point to the closest point, then to the point closest to this, etc., in general does not yield the shortest route

Hassler Whitney at Princeton University introduced the name *travelling salesman problem* soon after.[[5]](http://en.wikipedia.org/wiki/Travelling_salesman_problem#cite_note-5)

In the 1950s and 1960s, the problem became increasingly popular in scientific circles in Europe and the USA. Notable contributions were

made by George Dantzig, Delbert Ray Fulkerson and Selmer M. Johnson at the RAND Corporation in Santa Monica, who expressed the problem as an integer linear program and developed the cutting plane method for its solution. With these new methods they solved an instance with 49 cities to optimality by constructing a tour and proving that no other tour could be shorter. In the following decades, the problem was studied by many researchers from mathematics, computer science, chemistry, physics, and other sciences.

Richard M. Karp showed in 1972 that the Hamiltonian cycle problem was NP-complete, which implies the NP-hardness of TSP. This supplied a mathematical explanation for the apparent computational difficulty of finding optimal tours.

Great progress was made in the late 1970s and 1980, when Grötschel, Padberg, Rinaldi and others managed to exactly solve instances with up to 2392 cities, using cutting planes and branch-and-bound.

In the 1990s, Applegate, Bixby, Chvátal, and Cook developed the program *Concorde* that has been used in many recent record solutions. Gerhard Reinelt published the TSPLIB in 1991, a collection of benchmark instances of varying difficulty, which has been used by many research groups for comparing results. In 2006, Cook and others computed an optimal tour through an 85,900-city instance given by a microchip layout problem, currently the largest solved TSPLIB instance. For many other instances with millions of cities, solutions can be found that are guaranteed to be within 2-3% of an optimal tour.

# Suggested solution/algorithms And algorithm along with the dry execution:

The traditional lines of attack for the NP-hard problems are the following:

* Devising algorithms for finding exact solutions (they will work reasonably fast only for small problem sizes).
* Devising "suboptimal" or heuristic algorithms, i.e., algorithms that deliver either seemingly or probably good solutions, but which could not be proved to be optimal.
* Finding special cases for the problem ("subproblems") for which either better or exact heuristics are possible.

**Computational complexity**

The problem has been shown to be NP-hard (more precisely, it is complete for the complexity class FPNP ), and the decision problem version ("given the costs and a number *x*, decide whether there is a round-trip route cheaper than *x*") is NP-complete. The bottleneck travelling salesman problem is also NP-hard. The problem remains NP-hard even for the case when the cities are in the plane with Euclidean distances, as well as in a number of other restrictive cases. Removing the condition of visiting each city "only once" does not remove the NP-hardness, since it is easily seen that in the planar case there is an optimal tour that visits each city only once (otherwise, by the triangle inequality, a shortcut that skips a repeated visit would not increase the tour length).

**Complexity of approximation**

In the general case, finding a shortest travelling salesman tour is NPO-complete. If the distance measure is a metric and symmetric, the problem becomes APX-completeand Christofides’s algorithm approximates it within 1.5.

If the distances are restricted to 1 and 2 (but still are a metric) the approximation ratio becomes 8/7. In the asymmetric, metric case, only logarithmic performance guarantees are known, the best current algorithm achieves performance ratio 0.814 log(*n*)  it is an open question if a constant factor approximation exists.

The corresponding maximization problem of finding the *longest* travelling salesman tour is approximable within 63/38. If the distance function is symmetric, the longest tour can be approximated within 4/3 by a deterministic algorithm and within \tfrac{1}{25}(33+\varepsilon) by a randomised algorithm.

**Exact algorithms**:

The most direct solution would be to try all permutations (ordered combinations) and see which one is cheapest (using brute force search). The running time for this approach lies within a polynomial factor of O(n!), the factorial of the number of cities, so this solution becomes impractical even for only 20 cities. One of the earliest applications ofdynamic programming is the Held–Karp algorithm that solves the problem in time O(n^2 2^n).

Improving these time bounds seems to be difficult. For example, it has not been determined whether an exact algorithm for TSP that runs in time O(1.9999^n) exists.

Other approaches include:

* Various branch-and-bound algorithms, which can be used to process TSPs containing 40–60 cities.
* Progressive improvement algorithms which use techniques reminiscent of linear programming. Works well for up to 200 cities.
* Implementations of branch-and-bound and problem-specific cut generation (branch-and-cut); this is the method of choice for solving large instances. This approach holds the current record, solving an instance with 85,900 cities, see Applegate et al. (2006).

An exact solution for 15,112 German towns from TSPLIB was found in 2001 using the cutting-plane method proposed by George Dantzig,Ray Fulkerson, and Selmer M. Johnson in 1954, based on linear programming. The computations were performed on a network of 110 processors located at Rice University and Princeton University (see the Princeton external link). The total computation time was equivalent to 22.6 years on a single 500 MHz Alpha processor. In May 2004, the travelling salesman problem of visiting all 24,978 towns in Sweden was solved: a tour of length approximately 72,500 kilometers was found and it was proven that no shorter tour exists.

In March 2005, the travelling salesman problem of visiting all 33,810 points in a circuit board was solved using *Concorde TSP Solver*: a tour of length 66,048,945 units was found and it was proven that no shorter tour exists. The computation took approximately 15.7 CPU-years (Cook et al. 2006). In April 2006 an instance with 85,900 points was solved using *Concorde TSP Solver*, taking over 136 CPU-years, see Applegate et al. (2006).

**Heuristic and approximation algorithms**:

Various heuristics and approximation algorithms, which quickly yield good solutions have been devised. Modern methods can find solutions for extremely large problems (millions of cities) within a reasonable time which are with a high probability just 2–3% away from the optimal solution.

Several categories of heuristics are recognized.

#### Constructive heuristics

The nearest neighbor (NN) algorithm (or so-called greedy algorithm) lets the salesman choose the nearest unvisited city as his next move. This algorithm quickly yields an effectively short route. For N cities randomly distributed on a plane, the algorithm on average yields a path 25% longer than the shortest possible path. However, there exist many specially arranged city distributions which make the NN algorithm give the worst route (Gutin, Yeo, and Zverovich, 2002). This is true for both asymmetric and symmetric TSPs (Gutin and Yeo, 2007). Rosenkrantz et al. [1977] showed that the NN algorithm has the approximation factor \Theta(\log |V|) for instances satisfying the triangle inequality. A variation of NN algorithm, called Nearest Fragment (NF) operator, which connects a group (fragment) of nearest unvisited cities, can find shorter route with successive iterations. The NF operator can also be applied on an initial solution obtained by NN algorithm for further improvement in an elitist model, where only better solutions are accepted.

Constructions based on a minimum spanning tree have an approximation ratio of 2. The Christofides algorithm achieves a ratio of 1.5.

The bitonic tour of a set of points is the minimum-perimeter monotone polygon that has the points as its vertices; it can be computed efficiently by dynamic programming.

Another constructive heuristic, Match Twice and Stitch (MTS) (Kahng, Reda 2004 ), performs two sequential matchings, where the second matching is executed after deleting all the edges of the first matching, to yield a set of cycles. The cycles are then stitched to produce the final tour.

**Iterative improvement**

**Pairwise exchange**

The pairwise exchange or *2-opt* technique involves iteratively removing two edges and replacing these with two different edges that reconnect the fragments created by edge removal into a new and shorter tour. This is a special case of the *k*-opt method. Note that the label *Lin–Kernighan* is an often heard misnomer for 2-opt. Lin–Kernighan is actually the more general k-opt method.

***k*-opt heuristic, or Lin–Kernighan heuristics**

Take a given tour and delete *k* mutually disjoint edges. Reassemble the remaining fragments into a tour, leaving no disjoint subtours (that is, don't connect a fragment's endpoints together). This in effect simplifies the TSP under consideration into a much simpler problem. Each fragment endpoint can be connected to 2*k* − 2 other possibilities: of 2*k* total fragment endpoints available, the two endpoints of the fragment under consideration are disallowed. Such a constrained 2*k*-city TSP can then be solved with brute force methods to find the least-cost recombination of the original fragments. The *k*-opt technique is a special case of the *V*-opt or variable-opt technique. The most popular of the *k*-opt methods are 3-opt, and these were introduced by Shen Lin of Bell Labs in 1965. There is a special case of 3-opt where the edges are not disjoint (two of the edges are adjacent to one another). In practice, it is often possible to achieve substantial improvement over 2-opt without the combinatorial cost of the general 3-opt by restricting the 3-changes to this special subset where two of the removed edges are adjacent. This so-called two-and-a-half-opt typically falls roughly midway between 2-opt and 3-opt, both in terms of the quality of tours achieved and the time required to achieve those tours.

***V*-opt heuristic**

The variable-opt method is related to, and a generalization of the *k*-opt method. Whereas the *k*-opt methods remove a fixed number (*k*) of edges from the original tour, the variable-opt methods do not fix the size of the edge set to remove. Instead they grow the set as the search process continues. The best known method in this family is the Lin–Kernighan method (mentioned above as a misnomer for 2-opt). Shen Lin and Brian Kernighan first published their method in 1972, and it was the most reliable heuristic for solving travelling salesman problems for nearly two decades. More advanced variable-opt methods were developed at Bell Labs in the late 1980s by David Johnson and his research team. These methods (sometimes called Lin–Kernighan–Johnson) build on the Lin–Kernighan method, adding ideas from tabu search and evolutionary computing. The basic Lin–Kernighan technique gives results that are guaranteed to be at least 3-opt. The Lin–Kernighan–Johnson methods compute a Lin–Kernighan tour, and then perturb the tour by what has been described as a mutation that removes at least four edges and reconnecting the tour in a different way, then *V*-opting the new tour. The mutation is often enough to move the tour from the local minimum identified by Lin–Kernighan. *V*-opt methods are widely considered the most powerful heuristics for the problem, and are able to address special cases, such as the Hamilton Cycle Problem and other non-metric TSPs that other heuristics fail on. For many years Lin–Kernighan–Johnson had identified optimal solutions for all TSPs where an optimal solution was known and had identified the best known solutions for all other TSPs on which the method had been tried.

**Randomised improvement**:

Optimized Markov chain algorithms which use local searching heuristic sub-algorithms can find a route extremely close to the optimal route for 700 to 800 cities.

TSP is a touchstone for many general heuristics devised for combinatorial optimization such as genetic algorithms, simulated annealing, Tabu search, ant colony optimization, river formation dynamics (see swarm intelligence) and the cross entropy method.

**Ant colony optimization**:

Main article: Ant colony optimization algorithms

Artificial intelligence researcher Marco Dorigo described in 1997 a method of heuristically generating "good solutions" to the TSP using a simulation of an ant colony called *ACS*(Ant Colony System). It models behavior observed in real ants to find short paths between food sources and their nest, an emergent behavior resulting from each ant's preference to follow trail pheromones deposited by other ants.

ACS sends out a large number of virtual ant agents to explore many possible routes on the map. Each ant probabilistically chooses the next city to visit based on a heuristic combining the distance to the city and the amount of virtual pheromone deposited on the edge to the city. The ants explore, depositing pheromone on each edge that they cross, until they have all completed a tour. At this point the ant which completed the shortest tour deposits virtual pheromone along its complete tour route (*global trail updating*). The amount of pheromone deposited is inversely proportional to the tour length: the shorter the tour, the more it deposits.

Analyst's travelling salesman problem

There is an analogous problem in geometric measure theory which asks the following: under what conditions may a subset *E* of Euclidean space be contained in a rectifiable curve (that is, when is there a curve with finite length that visits every point in *E*)? This problem is known as the analyst's travelling salesman problem or the geometric travelling salesman problem.

# Free software for solving TSP:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name (alphabetically)** | **License** | **API language** | **Brief info** |
| Concorde | free for academic | only executable | requires a linear solver installation for its MILP subproblem |
| DynOpt | ? | C | an ANSI C implementation a dynamic programming based algorithm developed by Balas and Simonetti, approximate solution |
| LKH | research only | C | an effective implementation of the Lin-Kernighan heuristic for Euclidean traveling salesman problem |
| OpenOpt | BSD | Python | exact and approximate solvers, STSP / ATSP, can handle multigraphs, constraints, multiobjective problems, see its TSP page for details and examples |
| OptaPlanner | Apache License | Java | Open Source Java constraint solver with TSP and VRP examples. |
| R TSP package | GPL | R | infrastructure and solvers for STSP / ATSP, interface to Concorde |
| TSP Solver and Generator | GPL | C++ | branch and bound algorithm |
| MJC2 Free Vehicle Routing Software | ? | C++ | executable only |
| TSPGA | ? | C | approximate solution of the STSP using the "pgapack" package |