Create account Log in

Article Talk Read Edit More Search

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

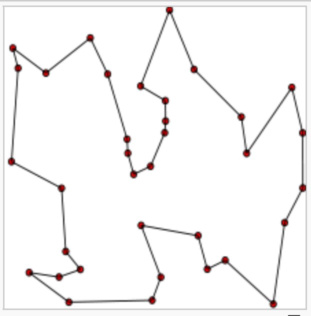
Languages
العربية
Català
Čeština
Dansk
Deutsch
Español
Euskara
فارسی
Français
한국어
Italiano
עברית
Lietuvių
Magyar
Монгол
Nederlands
日本語
Polski
Português
Русский
Simple English
Slovenčina
Slovenščina
Српски / srpski
Srpskohrvatski / српскохрватски
Suomi
Svenska
Türkçe
Українська
Tiếng Việt
中文
Edit links

Travelling salesman problem

From Wikipedia, the free encyclopedia
(Redirected from Traveling salesman problem)

The **travelling salesman problem** (**TSP**) asks the following question: *Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?* It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

TSP is a special case of the travelling purchaser problem and the Vehicle routing problem.


Solution of a travelling salesman 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 superpolynomially (perhaps, specifically, 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%.[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. The TSP also appears in astronomy, as astronomers observing many sources will want to minimise the time spent slewing the telescope between the sources. In many applications, additional constraints such as limited resources or time windows may be imposed.

Contents [hide]

History [edit]

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.[2]

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 Icosian Game was a recreational puzzle based on finding a Hamiltonian cycle.[3] 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. [4]

William Rowan Hamilton

It was first considered mathematically in the 1930s by Merrill Flood who was looking to solve a school bus routing problem.[5] Hassler Whitney at Princeton University introduced the name *travelling salesman problem* soon after.[6]

In the 1950s and 1960s, the problem became increasingly popular in scientific circles in Europe and the USA after the RAND Corporation in Santa Monica, offered prizes for steps in solving the problem.[5] Notable contributions were made by George Dantzig, Delbert Ray Fulkerson and Selmer M. Johnson from the RAND Corporation, who expressed the problem as an integer linear program and developed the cutting plane method for its solution. They wrote what is considered the seminal paper on the subject in which 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. Dantzig, Fulkerson and Johnson, however, speculated that given a near optimal solution we may be able to find optimality or prove optimality by adding a small amount of extra inequalities (cuts). They used this idea to solve their initial 49 city problem using a string model. They found they only needed 26 cuts to come to a solution for their 49 city problem. While this paper did not give an algorithmic approach to TSP problems, the ideas that lay within it were indispensable to later creating exact solution methods for the TSP, though it would take 15 years to find an algorithmic approach in creating these cuts.[5] As well as cutting plane methods, Dantzig, Fulkerson and Johnson used branch and bound algorithms perhaps for the first time.[5]

In the following decades, the problem was studied by many researchers from mathematics, computer science, chemistry, physics, and other sciences. In the 1960s however a new approach was created, instead of finding optimal solutions, people tried to instead find the worst solutions and in doing so, created lower bounds for the problem. These may then be used with branch and bound approaches. One method of doing this was to create the minimum spanning tree of the graph and then multiply the cost of this by 2.[5]

Christofides made a big advance in this approach of giving an approach for which we know the worst-case scenario. His algorithm given in 1976, at worst is 1.5 times longer than the optimal solution. As the algorithm was so simple and quick, many hoped it would give way to a near optimal solution method. However, until 2011 when it was beaten by less than a billionth of a percent, this remained the method with the best worst-case scenario.[7]

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.

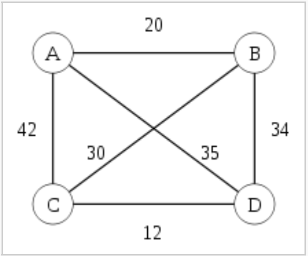
In the 1990s, Applegate, Bixby, Chvátal, and Cook developed the program *Concorde* that has been used in many recent record solutions. Gerhard Reinelt published the TSPLIB in 1991, a collection of benchmark instances of varying difficulty, which has been used by many research groups for comparing results. 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.[8]

The problem is sometimes, especially in newer publications, referred to as *Travelling Salesperson Problem*.

Description [edit]

As a graph problem [edit]

TSP can be modelled as an undirected weighted graph, such that cities are the graph's vertices, paths are the graph's edges, and a path's distance is the edge's length. It is a minimization problem starting and finishing at a specified vertex after having visited each other vertex exactly once. Often, the model is a complete graph (*i.e.* each pair of vertices is connected by an edge). If no path exists between two cities, adding an arbitrarily long edge will complete the graph without affecting the optimal tour.



Symmetric TSP with four cities

Asymmetric and symmetric [edit]

In the *symmetric TSP*, the distance between two cities is the same in each opposite direction, forming an undirected graph. This symmetry halves the number of possible solutions. In the *asymmetric TSP*, paths may not exist in both directions or the distances might be different, forming a directed graph. Traffic collisions, one-way streets, and airfares for cities with different departure and arrival fees are examples of how this symmetry could break down.

Related problems [edit]

- An equivalent formulation in terms of graph theory is: Given a complete weighted graph (where the vertices would represent the cities, the edges would represent the roads, and the weights would be the cost or distance of that road), find a Hamiltonian cycle with the least weight.

- The requirement of returning to the starting city does not change the computational complexity of the problem, see Hamiltonian path problem.
- Another related problem is the bottleneck travelling salesman problem (bottleneck TSP): Find a Hamiltonian cycle in a weighted graph with the minimal weight of the weightiest edge. The problem is of considerable practical importance, apart from evident transportation and logistics areas. A classic example is in printed circuit manufacturing: scheduling of a route of the drill machine to drill holes in a PCB. In robotic machining or drilling applications, the "cities" are parts to machine or holes (of different sizes) to drill, and the "cost of travel" includes time for retooling the robot (single machine job sequencing problem).[9]
- The generalized travelling salesman problem, also known as the "travelling politician problem", deals with "states" that have (one or more) "cities" and the salesman has to visit exactly one "city" from each "state". One application is encountered in ordering a solution to the cutting stock problem in order to minimise knife changes. Another is concerned with drilling in semiconductor manufacturing, see e.g., U.S. Patent 7,054,798. Surprisingly, Behzad and Modarres demonstrated that the generalised travelling salesman problem can be transformed into a standard travelling salesman problem with the same number of cities, but a modified distance matrix.
- The sequential ordering problem deals with the problem of visiting a set of cities where precedence relations between the cities exist.
- The travelling purchaser problem deals with a purchaser who is charged with purchasing a set of products. He can purchase these products in several cities, but at different prices and not all cities offer the same products. The objective is to find a route between a subset of the cities, which minimizes total cost (travel cost + purchasing cost) and which enables the purchase of all required products.

Integer linear programming formulation [edit]

TSP can be formulated as an integer linear program.[10][11][12] Label the cities with the numbers 0, ..., *n* and define:

$$x_{ij} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

For *i* = 0, ..., *n*, let u_i be an artificial variable, and finally take c_{ij} to be the distance from city *i* to city *j*. Then TSP can be written as the following integer linear programming problem:

$$\begin{aligned} \min \sum_{i=0}^{n} \sum_{j\ne i,j=0}^{n} c_{ij}x_{ij} \\ 0 \le x_{ij} \le 1 && i,j = 0,\cdots,n \\ u_i \in \mathbf{Z} && i = 0,\cdots,n \\ \sum_{i=0,i\ne j}^{n} x_{ij} = 1 && j = 0,\cdots,n \\ \sum_{j=0,j\ne i}^{n} x_{ij} = 1 && i = 0,\cdots,n \\ u_i - u_j + nx_{ij} \le n-1 && 1 \le i \ne j \le n \end{aligned}$$

The first set of equalities requires that each city be arrived at from exactly one other city, and the second set of equalities requires that from each city there is a departure to exactly one other city. The last constraints enforce that there is only a single tour covering all cities, and not two or more disjointed tours that only collectively cover all cities. To prove this, it is shown below (1) that every feasible solution contains only one closed sequence of cities, and (2) that for every single tour covering all cities, there are values for the dummy variables u_i that satisfy the constraints.

To prove that every feasible solution contains only one closed sequence of cities, it suffices to show that every subtour in a feasible solution passes through city 0 (noting that the equalities ensure there can only be one such tour). For if we sum all the inequalities corresponding to $x_{ij} = 1$ for any subtour of *k* steps not passing through city 0, we obtain:

$$nk \le (n-1)k,$$

which is a contradiction.

It now must be shown that for every single tour covering all cities, there are values for the dummy variables u_i that satisfy the constraints.

Without loss of generality, define the tour as originating (and ending) at city 0. Choose $u_i = t$ if city *i* is visited in step *t* (*i*, *t* = 1, 2, ..., n). Then

$$u_i - u_j \le n-1,$$

since u_i can be no greater than *n* and u_j can be no less than 1; hence the constraints are satisfied whenever $x_{ij} = 0.$ For $x_{ij} = 1,$ we have:

$$u_i - u_j + nx_{ij} = (t) - (t+1) + n = n-1,$$

satisfying the constraint.

Computing a solution [edit]

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

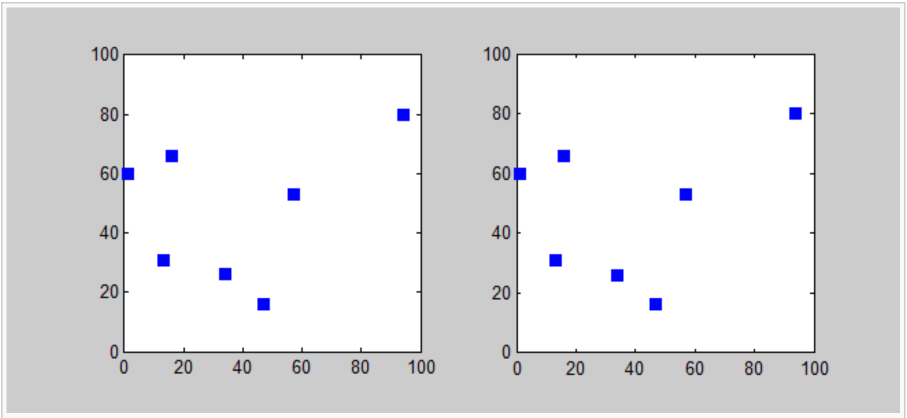
- Devising algorithms for finding exact solutions (they will work reasonably fast only for small problem sizes).
- Devising "suboptimal" or heuristic algorithms, i.e., algorithms that deliver either seemingly or probably good solutions, but which could not be proved to be optimal.
- Finding special cases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

Exact algorithms [edit]

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 of dynamic programming is the Held–Karp algorithm that solves the problem in time $O(n^2 2^n)$.[13]

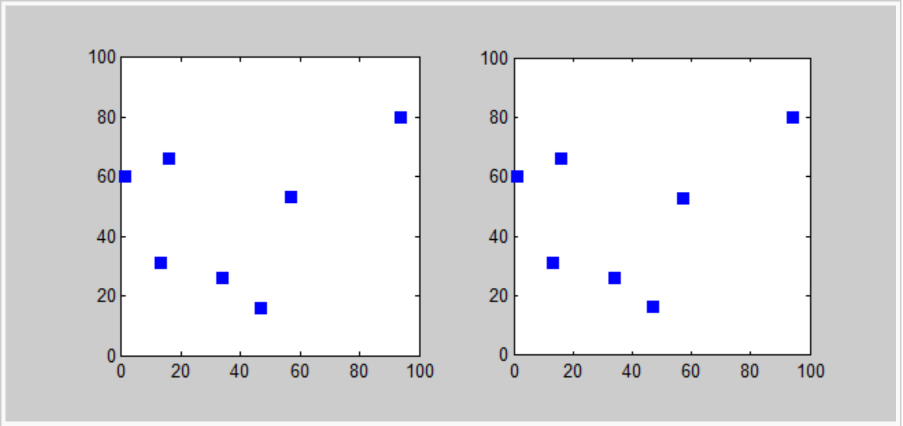
Improving these time bounds seems to be difficult. For example, it has not been determined whether an exact algorithm for TSP that runs in time $O(1.9999^n)$ exists.[14]



Solution to a symmetric TSP with 7 cities using brute force search. Note: Number of permutations: (7-1)!/2 = 360

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[15]); 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).



Solution of a TSP with 7 cities using a simple Branch and bound algorithm. Note: The number of permutations is much less than Brute force search

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.[16] 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 [edit]

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.[8]

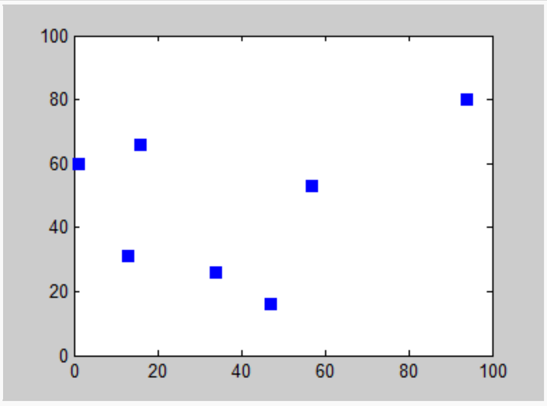
Several categories of heuristics are recognized.

Constructive heuristics [edit]

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.[17] 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.[18] 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.

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 [19]), 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.

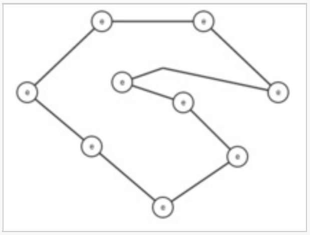

Nearest Neighbor algorithm for a TSP with 7 cities. The solution changes as the starting point is changed

Christofides' algorithm for the TSP [edit]

The Christofides algorithm follows a similar outline but combines the minimum spanning tree with a solution of another problem, minimum-weight perfect matching. This gives a TSP tour which is at most 1.5 times the optimal. The Christofides algorithm was one of the first approximation algorithms, and was in part responsible for drawing attention to approximation algorithms as a practical approach to intractable problems. As a matter of fact, the term "algorithm" was not commonly extended to approximation algorithms until later; the Christofides algorithm was initially referred to as the Christofides heuristic.

This algorithm looks at things differently by using a result from graph theory which helps improve on the LB of the TSP which originated from doubling the cost of the minimum spanning tree. Given an Eulerian graph we can find an Eulerian tour in *O(n)* time.[5] So if we had an Eulerian graph with cities from a TSP as vertices then we can easily see that we could use such a method for finding an Eulerian tour to find a TSP solution. By triangular inequality we know that the TSP tour can be no longer than the Eulerian tour and as such we have a LB for the TSP. Such a method is described below.

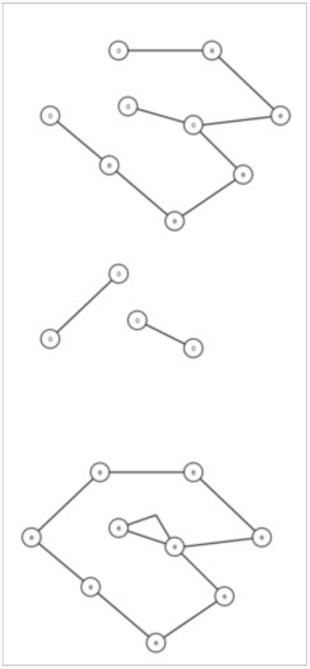
1. Find a minimum spanning tree for the problem
2. Create duplicates for every edge to create an Eulerian graph
3. Find an Eulerian tour for this graph
4. Convert to TSP: if a city is visited twice, create a shortcut from the city before this in the tour to the one after this.


Using a shortcut heuristic on the graph created by the matching below

To improve our lower bound, we therefore need a better way of creating an Eulerian graph. But by triangular inequality, the best Eulerian graph must have the same cost as the best travelling salesman tour, hence finding optimal Eulerian graphs is at least as hard as TSP. One way of doing this that has been proposed is by the concept of minimum weight matching for the creation of which there exist algorithms of $O(n^3)$.[5]

To make a graph into an Eulerian graph, one starts with the minimum spanning tree. Then all the vertices of odd order must be made even. So a matching for the odd degree vertices must be added which increases the order of every odd degree vertex by one.[5] This leaves us with a graph where every vertex is of even order which is thus Eulerian. Now we can adapt the above method to give Christofides' algorithm,

1. Find a minimum spanning tree for the problem
2. Create a matching for the problem with the set of cities of odd order.
3. Find an Eulerian tour for this graph
4. Convert to TSP using shortcuts.


Creating a matching

Iterative improvement [edit]

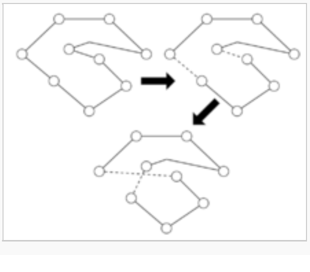
Pairwise exchange

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.

For Euclidean instances, 2-opt heuristics give on average solutions that are about 5% better than Christofides' algorithm. If we start with an initial solution made with a greedy algorithm, the average number of moves greatly decreases again and is *O(n)*. For random starts however, the average number of moves is *O(n* log*(n))*. However whilst in order this is a small increase in size, the initial number of moves for small problems is 10 times as big for a random start compared to one made from a greedy heuristic. This is because such 2-opt heuristics exploit `bad' parts of a solution such as crossings. These types of heuristics are often used within Vehicle routing problem heuristics to reoptimise route solutions.[20]

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



An example of a 2-opt iteration

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 [edit]

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.

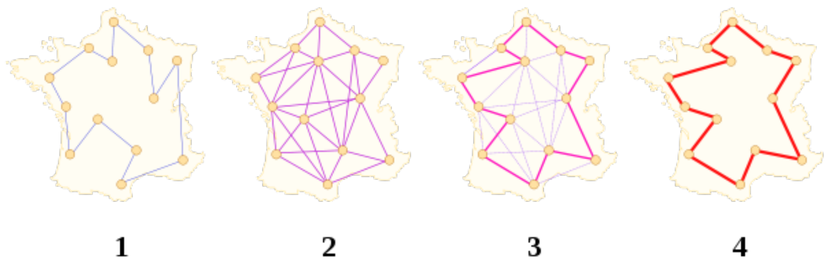
TSP is a touchstone for many general heuristics devised for combinatorial optimization such as genetic algorithms, simulated annealing, Tabu search, ant colony optimization, river formation dynamics (see swarm intelligence) and the cross entropy method.

Ant colony optimization [edit]

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).[21] It models behavior observed in real ants to find short paths between food sources and their nest, an emergent behaviour 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.



Ant Colony Optimization Algorithm for a TSP with 7 cities: Red and thick lines in the pheromone map indicate presence of more pheromone

Special cases of the TSP [edit]

Metric TSP [edit]

In the *metric TSP*, also known as *delta-TSP* or Δ-TSP, the intercity distances satisfy the triangle inequality.

A very natural restriction of the TSP is to require that the distances between cities form a metric to satisfy the triangle inequality; that is the direct connection from *A* to *B* is never farther than the route via intermediate *C*:

$$d_{AB} \leq d_{AC} + d_{CB}.$$

The edge spans then build a metric on the set of vertices. When the cities are viewed as points in the plane, many natural distance functions are metrics, and so many natural instances of TSP satisfy this constraint.

The following are some examples of metric TSPs for various metrics.

- In the Euclidean TSP (see below) the distance between two cities is the Euclidean distance between the corresponding points.
- In the rectilinear TSP the distance between two cities is the sum of the differences of their *x*- and *y*-coordinates. This metric is often called the Manhattan distance or city-block metric.
- In the maximum metric, the distance between two points is the maximum of the absolute values of differences of their *x*- and *y*-coordinates.

The last two metrics appear for example in routing a machine that drills a given set of holes in a printed circuit board. The Manhattan metric corresponds to a machine that adjusts first one co-ordinate, and then the other, so the time to move to a new point is the sum of both movements. The maximum metric corresponds to a machine that adjusts both co-ordinates simultaneously, so the time to move to a new point is the slower of the two movements.

In its definition, the TSP does not allow cities to be visited twice, but many applications do not need this constraint. In such cases, a symmetric, non-metric instance can be reduced to a metric one. This replaces the original graph with a complete graph in which the inter-city distance d_{AB} is replaced by the shortest path between *A* and *B* in the original graph.

Euclidean TSP [edit]

The **Euclidean TSP**, or **planar TSP**, is the TSP with the distance being the ordinary Euclidean distance.

The Euclidean TSP is a particular case of the metric TSP, since distances in a plane obey the triangle inequality.

Like the general TSP, the Euclidean TSP is NP-hard. With discretized metric (distances rounded up to an integer), the problem is NP-complete.[22] However, in some respects it seems to be easier than the general metric TSP. For example, the minimum spanning tree of the graph associated with an instance of the Euclidean TSP is a Euclidean minimum spanning tree, and so can be computed in expected O (*n* log *n*) time for *n* points (considerably less than the number of edges). This enables the simple 2-approximation algorithm for TSP with triangle inequality above to operate more quickly.

In general, for any *c* > 0, where *d* is the number of dimensions in the Euclidean space, there is a polynomial-time algorithm that finds a tour of length at most (1 + 1/*c*) times the optimal for geometric instances of TSP in

$$O\left(n(\log n)^{(O(c\sqrt{d}))^{d-1}}\right),$$

time; this is called a polynomial-time approximation scheme (PTAS).[23] Sanjeev Arora and Joseph S. B. Mitchell were awarded the Gödel Prize in 2010 for their concurrent discovery of a PTAS for the Euclidean TSP.

In practice, simpler heuristics with weaker guarantees continue to be used.

Asymmetric TSP [edit]

In most cases, the distance between two nodes in the TSP network is the same in both directions. The case where the distance from *A* to *B* is not equal to the distance from *B* to *A* is called asymmetric TSP. A practical application of an asymmetric TSP is route optimisation using street-level routing (which is made asymmetric by one-way streets, slip-roads, motorways, etc.).

Solving by conversion to symmetric TSP [edit]

Solving an asymmetric TSP graph can be somewhat complex. The following is a 3×3 matrix containing all possible path weights between the nodes *A*, *B* and *C*. One option is to turn an asymmetric matrix of size *N* into a symmetric matrix of size 2*N*.[24]

Asymmetric path weights

	A	***B***	***C***
A		1	2
B	6		3
C	5	4	

To double the size, each of the nodes in the graph is duplicated, creating a second *ghost node*. Using duplicate points with very low weights, such as −∞, provides a cheap route "linking" back to the real node and allowing symmetric evaluation to continue. The original 3×3 matrix shown above is visible in the bottom left and the inverse of the original in the top-right. Both copies of the matrix have had their diagonals replaced by the low-cost hop paths, represented by −∞.

Symmetric path weights

	A	***B***	***C***	***A′***	***B′***	***C′***
A				−∞	6	5
B				1	−∞	4

C				2	3	−∞
A′	−∞	1	2			
B′	6	−∞	3			
C′	5	4	−∞			

The original 3×3 matrix would produce two Hamiltonian cycles (a path that visits every node once), namely *A*-*B*-*C*-*A* [score 9] and *A*-*C*-*B*-*A* [score 12]. Evaluating the 6×6 symmetric version of the same problem now produces many paths, including *A*-*A′*-*B*-*B′*-*C*-*C′*-*A*, *A*-*B′*-*C*-*A′*-*A*, *A*-*A′*-*B*-*C′*-*A* [all score 9 – ∞].

The important thing about each new sequence is that there will be an alternation between dashed (*A′*,*B′*,*C′*) and un-dashed nodes (*A*, *B*, *C*) and that the link to "jump" between any related pair (*A*-*A′*) is effectively free. A version of the algorithm could use any weight for the *A*-*A′* path, as long as that weight is *lower* than all other path weights present in the graph. As the path weight to "jump" must effectively be "free", the value zero (0) could be used to represent this cost—if zero is not being used for another purpose already (such as designating invalid paths). In the two examples above, non-existent paths between nodes are shown as a blank square.

Analyst's travelling salesman problem [edit]

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

TSP path length for random sets of points in a square [edit]

Suppose $X_1, \ldots, X_n$ are n independent random variables with uniform distribution in the square $[0,1]^2$, and let L_n^* be the shortest path length (i.e. TSP solution) for this set of points, according to the usual Euclidean distance. It is known[25] that, almost surely,

$$\frac{L_n^*}{\sqrt{n}} \to \beta \qquad \text{when } n \to \infty,$$

where β is a positive constant that is not known explicitly. Since $L_n^* \le 2\sqrt{n}+2$ (see below), it follows from bounded convergence theorem that $\beta = \lim_{n\to\infty} \mathbb{E}[L_n^*]/\sqrt{n}$, hence lower and upper bounds on β follow from bounds on $\mathbb{E}[L_n^*]$.

Upper bound [edit]

- One has $L^* \le 2\sqrt{n}+2$, and therefore $\beta \le 2$, by using a naive path which visits monotonically the points inside each of $\sqrt{n}$ slices of width $1/\sqrt{n}$ in the square.
- Few [26] proved $L_n^* \le \sqrt{2n}+1.75$, hence $\beta \le \sqrt{2}$, later improved by Karloff (1987): $\beta \le 0.984\sqrt{2}$.
- The currently [27] best upper bound is $\beta \le 0.92\ldots$.

Lower bound [edit]

- By observing that $\mathbb{E}[L_n^*]$ is greater than n times the distance between X_0 and the closest point $X_i \ne X_0$, one gets (after a short computation)

  $$\mathbb{E}[L_n^*] \ge \tfrac{1}{2}\sqrt{n}.$$

- A better lower bound is obtained[25] by observing that $\mathbb{E}[L_n^*]$ is greater than $\frac{1}{2}n$ times the sum of the distances between X_0 and the closest and second closest points $X_i, X_j \ne X_0$, which gives

  $$\mathbb{E}[L_n^*] \ge \left(\tfrac{1}{4}+\tfrac{3}{8}\right)\sqrt{n} = \tfrac{5}{8}\sqrt{n},$$

- The currently [27] best lower bound is

  $$\mathbb{E}[L_n^*] \ge \left(\tfrac{5}{8}+\tfrac{19}{5184}\right)\sqrt{n},$$

- Held and Karp[28] gave a polynomial-time algorithm that provides numerical lower bounds for L_n^*, and thus for $\beta(\simeq L_n^*/\sqrt{n})$ which seem to be good up to more or less 1%.[29] In particular, David S. Johnson[30] obtained a lower bound by computer experiment:

  $$L_n^* \gtrsim 0.7080\sqrt{n}+0.522,$$

where 0.522 comes from the points near square boundary which have fewer neighbors, and Christine L. Valenzuela and Antonia J. Jones [31] obtained the following other numerical lower bound:

$$L_n^* \gtrsim 0.7078\sqrt{n}+0.551.$$

Computational complexity [edit]

The problem has been shown to be NP-hard (more precisely, it is complete for the complexity class FP^{NP}; see function problem), 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 [edit]

In the general case, finding a shortest travelling salesman tour is NPO-complete.[32] If the distance measure is a metric and symmetric, the problem becomes APX-complete[33] and Christofides's algorithm approximates it within 1.5.[34]

If the distances are restricted to 1 and 2 (but still are a metric) the approximation ratio becomes 8/7.[35] In the asymmetric, metric case, only logarithmic performance guarantees are known, the best current algorithm achieves performance ratio 0.814 log(n);[36] 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.[37] If the distance function is symmetric, the longest tour can be approximated within 4/3 by a deterministic algorithm[38] and within $\frac{1}{25}(33+\varepsilon)$ by a randomised algorithm.[39]

Human performance on TSP [edit]

The TSP, in particular the Euclidean variant of the problem, has attracted the attention of researchers in cognitive psychology. It has been observed that humans are able to produce good quality solutions quickly.[40] These results suggest that computer performance on the TSP may be improved by understanding and emulating the methods used by humans for these problems, and have also led to new insights into the mechanisms of human thought.[41] The first issue of the *Journal of Problem Solving* was devoted to the topic of human performance on TSP,[42] and a 2011 review listed dozens of papers on the subject.[41]

Benchmarks [edit]

For benchmarking of TSP algorithms, **TSPLIB** is a library of sample instances of the TSP and related problems is maintained, see the TSPLIB external reference. Many of them are lists of actual cities and layouts of actual printed circuits.

.

Popular culture [edit]

Travelling Salesman, by director Timothy Lanzone, is the story of four mathematicians hired by the U.S. government to solve the most elusive problem in computer-science history: P vs. NP.[43]

See also [edit]

- Canadian traveller problem
- Route inspection problem (also known as "Chinese postman problem")
- Set TSP problem
- Seven Bridges of Königsberg
- Tube Challenge
- Vehicle routing problem
- Graph Exploration

Notes [edit]

1. ^ See the TSP world tour problem which has already been solved to within 0.05% of the optimal solution. [1]
2. ^ "Der Handlungsreisende – wie er sein soll und was er zu tun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein – von einem alten Commis-Voyageur" (The travelling salesman — how he must be and what he should do in order to get commissions and be sure of the happy success in his business — by an old *commis-voyageur*)
3. ^ A discussion of the early work of Hamilton and Kirkman can be found in Graph Theory 1736–1936
4. ^ Cited and English translation in Schrijver (2005). Original German: "Wir bezeichnen als *Botenproblem* (weil diese Frage in der Praxis von jedem Postboten, übrigens auch von vielen Reisenden zu lösen ist) die Aufgabe, für endlich viele Punkte, deren paarweise Abstände bekannt sind, den kürzesten die Punkte verbindenden Weg zu finden. Dieses Problem ist natürlich stets durch endlich viele Versuche lösbar. Regeln, welche die Anzahl der Versuche unter die Anzahl der Permutationen der gegebenen Punkte herunterdrücken würden, sind nicht bekannt. Die Regel, man solle vom Ausgangspunkt erst zum nächstgelegenen Punkt, dann zu dem diesem nächstgelegenen Punkt gehen usw., liefert im allgemeinen nicht den kürzesten Weg."
5. ^ *a b c d e f g h* al.], edited by E.L. Lawler ... [et (1985). *The Traveling salesman problem : a guided tour of combinatorial optimization* (Repr. with corrections. ed.). Chichester [West Sussex]: Wiley. ISBN 0471904139.
6. ^ A detailed treatment of the connection between Menger and Whitney as well as the growth in the study of TSP can be found in Alexander Schrijver's 2005 paper "On the history of combinatorial optimization (till 1960). Handbook of Discrete Optimization (K. Aardal, G.L. Nemhauser, R. Weismantel, eds.), Elsevier, Amsterdam, 2005, pp. 1–68.PS,PDF
7. ^ Klarreich, Erica. "Computer Scientists Find New Shortcuts for Infamous Traveling Salesman Problem". *WIRED*. Simons Science News. Retrieved 2015-06-14.
8. ^ *a b* Rego, César; Gamboa, Dorabela; Glover, Fred; Osterman, Colin (2011), "Traveling salesman problem heuristics: leading methods, implementations and latest advances", *European Journal of Operational Research* **211** (3): 427–441, doi:10.1016/j.ejor.2010.09.010, MR 2774420.
9. ^ Behzad, Arash; Modarres, Mohammad (2002), "New Efficient Transformation of the Generalized Traveling Salesman Problem into Traveling Salesman Problem", *Proceedings of the 15th International Conference of Systems Engineering (Las Vegas)*
10. ^ Papadimitriou, C.H.; Steiglitz, K. (1998), *Combinatorial optimization: algorithms and complexity*, Mineola, NY: Dover , pp.308-309.
11. ^ Tucker, A. W. (1960), "On Directed Graphs and Integer Programs", IBM Mathematical research Project (Princeton University)
12. ^ Dantzig, George B. (1963), *Linear Programming and Extensions*, Princeton, NJ: PrincetonUP, pp. 545–7, ISBN 0-691-08000-3, sixth printing, 1974.
13. ^ Bellman (1960), Bellman (1962), Held & Karp (1962)
14. ^ Woeginger (2003)
15. ^ Padberg & Rinaldi (1991)
16. ^ Work by David Applegate, AT&T Labs – Research, Robert Bixby, ILOG and Rice University, Vašek Chvátal, Concordia University, William Cook, University of Waterloo, and Keld Helsgaun, Roskilde University is discussed on their project web page hosted by the University of Waterloo and last updated in June 2004, here [2]
17. ^ Johnson, D.S. and McGeoch, L.A.. "The traveling salesman problem: A case study in local optimization", Local search in combinatorial optimization, 1997, 215-310
18. ^ Ray, S. S.; Bandyopadhyay, S.; Pal, S. K. (2007). "Genetic Operators for Combinatorial Optimization in TSP and Microarray Gene

Ordering". *Applied Intelligence* **26** (3): 183–195. doi:10.1007/s10489-006-0018-y.

19. ^ Kahng, A. B.; Reda, S. (2004). "Match Twice and Stitch: A New TSP Tour Construction Heuristic". *Operations Research Letters* **32** (6): 499–509. doi:10.1016/j.orl.2004.04.001.
20. ^ Johnson, D. S.; McGeoch, L. A. (1997). *The Traveling Salesman Problem: A case study in local optimisation* (PDF). London: John Wiley and Sons. pp. 215–310.
21. ^ Marco Dorigo. "Ant Colonies for the Traveling Salesman Problem. IRIDIA, Université Libre de Bruxelles. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66. 1997. http://citeseer.ist.psu.edu/86357.html
22. ^ Papadimitriou (1977).
23. ^ Arora (1998).
24. ^ Jonker, Roy; Volgenant, Ton. "Transforming asymmetric into symmetric traveling salesman problems". *Operations Research Letters* **2** (161–163): 1983. doi:10.1016/0167-6377(83)90048-2.
25. ^ *a* *b* Beardwood, Halton & Hammersley (1959)
26. ^ Few, L. (1955). "The shortest path and the shortest road through n points". *Mathematika* **2** (02): 141–144. doi:10.1112/s0025579300000784.
27. ^ *a* *b* Steinerberger, S. (2015). "New bounds for the traveling salesman constant". *Advances in Applied Probability* **47.1**.
28. ^ Held, M.; Karp, R.M. (1970). "The Traveling Salesman Problem and Minimum Spanning Trees". *Operations Research* **18**: 1138–1162. doi:10.1287/opre.18.6.1138.
29. ^ Goemans, M.; Bertsimas, D. (1991). "Probabilistic analysis of the Held and Karp lower bound for the Euclidean traveling salesman problem". *Mathematics of operation research* **16** (1): 72–89. doi:10.1287/moor.16.1.72.
30. ^ David S. Johnson
31. ^ Christine L. Valenzuela and Antonia J. Jones
32. ^ Orponen (1987)
33. ^ Papadimitriou (1983)
34. ^ Christofides (1976)
35. ^ Berman & Karpinski (2006).
36. ^ Kaplan (2004)
37. ^ Kosaraju (1994)
38. ^ Serdyukov (1984)
39. ^ Hassin (2000)
40. ^ Macgregor, J. N.; Ormerod, T. (June 1996), "Human performance on the traveling salesman problem", *Perception & Psychophysics* **58** (4): 527–539, doi:10.3758/BF03213088.
41. ^ *a* *b* MacGregor, James N.; Chu, Yun (2011), "Human performance on the traveling salesman and related problems: A review", *Journal of Problem Solving* **3** (2).
42. ^ *Journal of Problem Solving* 1(1), 2006, retrieved 2014-06-06.
43. ^ Geere, Duncan. "'Travelling Salesman' movie considers the repercussions if P equals NP". Wired. Retrieved 26 April 2012.

References [edit]

- Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J. (2006), *The Traveling Salesman Problem*, ISBN 0-691-12993-2.
- Arora, Sanjeev (1998), "Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems", *Journal of the ACM* **45** (5): 753–782, doi:10.1145/290179.290180, MR 1668147.
- Beardwood, J.; Halton, J.H.; Hammersley, J.M. (1959), "The Shortest Path Through Many Points", *Proceedings of the Cambridge Philosophical Society* **55**: 299–327, doi:10.1017/s0305004100034095.
- Bellman, R. (1960), "Combinatorial Processes and Dynamic Programming", in Bellman, R., Hall, M., Jr. (eds.), *Combinatorial Analysis, Proceedings of Symposia in Applied Mathematics 10*, American Mathematical Society, pp. 217–249.
- Bellman, R. (1962), "Dynamic Programming Treatment of the Travelling Salesman Problem", *J. Assoc. Comput. Mach.* **9**: 61–63, doi:10.1145/321105.321111.
- Berman, Piotr; Karpinski, Marek (2006), "8/7-approximation algorithm for (1,2)-TSP", *Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA '06)*, pp. 641–648, doi:10.1145/1109557.1109627, ISBN 0898716055, ECCC TR05-069.
- Christofides, N. (1976), *Worst-case analysis of a new heuristic for the travelling salesman problem*, Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh.
- Hassin, R.; Rubinstein, S. (2000), "Better approximations for max TSP", *Information Processing Letters* **75** (4): 181–186, doi:10.1016/S0020-0190(00)00097-1.
- Held, M.; Karp, R. M. (1962), "A Dynamic Programming Approach to Sequencing Problems", *Journal of the Society for Industrial and Applied Mathematics* **10** (1): 196–210, doi:10.1137/0110015.
- Kaplan, H.; Lewenstein, L.; Shafrir, N.; Sviridenko, M. (2004), "Approximation Algorithms for Asymmetric TSP by Decomposing Directed Regular Multigraphs", *In Proc. 44th IEEE Symp. on Foundations of Comput. Sci*, pp. 56–65.
- Kosaraju, S. R.; Park, J. K.; Stein, C. (1994), "Long tours and short superstrings'", *Proc. 35th Ann. IEEE Symp. on Foundations of Comput. Sci*, IEEE Computer Society, pp. 166–177.

- Orponen, P.; Mannila, H. (1987), "On approximation preserving reductions: Complete problems and robust measures'", *Technical Report C-1987–28, Department of Computer Science, University of Helsinki*.
- Padberg, M.; Rinaldi, G. (1991), "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems", *Siam Review*: 60–100, doi:10.1137/1033004.
- Papadimitriou, Christos H. (1977), "The Euclidean traveling salesman problem is NP-complete", *Theoretical Computer Science* **4** (3): 237–244, doi:10.1016/0304-3975(77)90012-3, MR 0455550.
- Papadimitriou, C. H.; Yannakakis, M. (1993), "The traveling salesman problem with distances one and two", *Math. Oper. Res.* **18**: 1–11, doi:10.1287/moor.18.1.1.
- Serdyukov, A. I. (1984), "An algorithm with an estimate for the traveling salesman problem of the maximum'", *Upravlyaemye Sistemy* **25**: 80–86.
- Steinerberger, Stefan (2015), "New Bounds for the Traveling Salesman Constant", *Advances in Applied Probability* **47**.
- Woeginger, G.J. (2003), "Exact Algorithms for NP-Hard Problems: A Survey", *Combinatorial Optimization – Eureka, You Shrink! Lecture notes in computer science, vol. 2570*, Springer, pp. 185–207.

Further reading [edit]

- Adleman, Leonard (1994), "Molecular Computation of Solutions To Combinatorial Problems" (PDF), *Science* **266** (5187): 1021–4, Bibcode:1994Sci...266.1021A, doi:10.1126/science.7973651, PMID 7973651.
- Arora, S. (1998), "Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems" (PDF), *Journal of the ACM* **45** (5): 753–782, doi:10.1145/290179.290180.
- Babin, Gilbert; Deneault, Stéphanie; Laportey, Gilbert (2005), *Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem*, Cahiers du GERAD, G-2005-02, Montreal: Group for Research in Decision Analysis.
- Cook, William (2011), *In Pursuit of the Travelling Salesman: Mathematics at the Limits of Computation*, Princeton University Press, ISBN 978-0-691-15270-7.
- Cook, William; Espinoza, Daniel; Goycoolea, Marcos (2007), "Computing with domino-parity inequalities for the TSP", *INFORMS Journal on Computing* **19** (3): 356–365, doi:10.1287/ijoc.1060.0204.
- Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. (2001), "35.2: The traveling-salesman problem", *Introduction to Algorithms* (2nd ed.), MIT Press and McGraw-Hill, pp. 1027–1033, ISBN 0-262-03293-7.
- Dantzig, G. B.; Fulkerson, R.; Johnson, S. M. (1954), "Solution of a large-scale traveling salesman problem", *Operations Research* **2** (4): 393–410, doi:10.1287/opre.2.4.393, JSTOR 166695.
- Garey, M. R.; Johnson, D. S. (1979), "A2.3: ND22–24", *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, pp. 211–212, ISBN 0-7167-1045-5.
- Goldberg, D. E. (1989), "Genetic Algorithms in Search, Optimization & Machine Learning", *Reading: Addison-Wesley* (New York: Addison-Wesley), Bibcode:1989gaso.book.....G, ISBN 0-201-15767-5.
- Gutin, G.; Yeo, A.; Zverovich, A. (2002), "Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP", *Discrete Applied Mathematics* **117** (1–3): 81–86, doi:10.1016/S0166-218X(01)00195-0.
- Gutin, G.; Punnen, A. P. (2006), *The Traveling Salesman Problem and Its Variations*, Springer, ISBN 0-387-44459-9.
- Johnson, D. S.; McGeoch, L. A. (1997), "The Traveling Salesman Problem: A Case Study in Local Optimization", in Aarts, E. H. L.; Lenstra, J. K., *Local Search in Combinatorial Optimisation*, John Wiley and Sons Ltd, pp. 215–310.
- Lawler, E. L.; Lenstra, J. K.; Rinnooy Kan, A. H. G.; Shmoys, D. B. (1985), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, ISBN 0-471-90413-9.
- MacGregor, J. N.; Ormerod, T. (1996), "Human performance on the traveling salesman problem" (PDF), *Perception & Psychophysics* **58** (4): 527–539, doi:10.3758/BF03213088.
- Mitchell, J. S. B. (1999), "Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, *k*-MST, and related problems", *SIAM Journal on Computing* **28** (4): 1298–1309, doi:10.1137/S0097539796309764.
- Rao, S.; Smith, W. (1998), "Approximating geometrical graphs via 'spanners' and 'banyans'", *Proc. 30th Annual ACM Symposium on Theory of Computing*, pp. 540–550.
- Rosenkrantz, Daniel J.; Stearns, Richard E.; Lewis, Philip M., II (1977), "An Analysis of Several Heuristics for the Traveling Salesman Problem", *SIAM Journal on Computing* **6** (5): 563–581, doi:10.1137/0206041.
- Vickers, D.; Butavicius, M.; Lee, M.; Medvedev, A. (2001), "Human performance on visually presented traveling salesman problems", *Psychological Research* **65** (1): 34–45, doi:10.1007/s004260000031, PMID 11505612.
- Walshaw, Chris (2000), *A Multilevel Approach to the Travelling Salesman Problem*, CMS Press.
- Walshaw, Chris (2001), *A Multilevel Lin-Kernighan-Helsgaun Algorithm for the Travelling Salesman Problem*, CMS Press.

External links [edit]

- Traveling Salesman Problem at University of Waterloo
- TSPLIB at the University of Heidelberg
- *Traveling Salesman Problem* by Jon McLoone at the Wolfram Demonstrations Project
- Traveling Salesman movie (on IMDB)

Wikimedia Commons has media related to ***Traveling salesman problem***.

Categories: Travelling salesman problem | NP-complete problems | NP-hard problems | Operations research | Graph algorithms | Computational problems in graph theory | Hamiltonian paths and cycles