Approximation Algorithms For Traveling Salesman Problem

K Vamsi Krishna
Supervisor: Dr. T. Kavitha
Department of Computer Science and Automation
Indian Institute of Science

Abstract

The Traveling Salesman Problem (TSP) is one of the well studied combinatorial optimization problems. Because of its simple specification (yet notoriously hard) and wide applicability it has attracted interest of researchers both from academics and industry. Another ascept of TSP is the number of variations it has, and suprisingly even the most simplest looking variants were found to be NP-hard.

In this report, we give a brief survey of the TSP and its variants, specifically various approximation algorithms that have been given for it.

1 Introduction

In layman terms, TSP can be stated as "Consider a salesman who has to visit a specified list of cities. Assuming (s)he has the information of all the intercity traveling costs, what is the cheapest way to visit all the cities and return to the home city?".

The earliest reference to this problem goes back to 1830s, but Karl Menger seems to have popularized it in 1920s. See *Ch. 1* of [1] and [2] for an interesting history of TSP.

1.1 Problem

Let us formalize the problem in terms of graphs, "Given a complete weighted graph, $G = (V, E)$ and weight function, $w : E \rightarrow \mathbb{R}^+$, find the minimum weight Hamiltonian cycle". G is a directed graph in general.

Let the vertex set be $V = \{1, 2, \cdots, n\}$ and weight of an edge (i, j) be w_{ij}. Let OPT denote the cost of an optimal TSP tour, T_{OPT}. Consider the instance of TSP in Figure 1, it has $OPT = 22$ and $T_{OPT} = \langle 1, 4, 3, 2, 1 \rangle$.

For an integer programming formulation see [1].

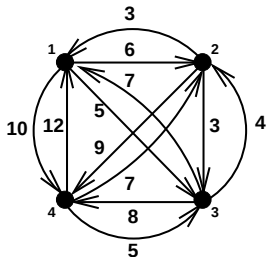

Figure 1: An instance of TSP

1.2 Motivation

As stated, TSP has wide applications in the sense that many problems can be cast as TSP problems. Apart from the obvious application in transportation and logistics, it has appeared in disguise in many fields like compilers [3], genetics [4], etc. For a feel of it, consider the following scheduling problem.

Consider a manufacturing environment where the same set of machines are used to produce n different commodities. Also, when the machines are changed from producing commodity i to commodity j, a change-over cost of c_{ij} is incurred. Since production continues cyclically, we need to consider the change over cost of last commodity to the first commodity also. It is clear that minimizing total change-over cost is same as solving TSP on the graph with commodities as verticies and weight of edges as change-over costs.

1.3 Outline of the Report

We start with describing some of the important variants of TSP in Section 2, and then in Section 3 indicate various methods of solving TSP, followed by approximation algorithms and synopsis of some algorithms that we have looked at in Section 4 and we conclude in Section 5.

2 Variants of TSP

TSP may be classified in various dimensions, some of which are described below.

Symmetric/Asymmetric: In the symmetric case the given graph is undirected (or equivalently, $w_{ij} = w_{ji}$) and in the asymmetric case it is a directed graph.

Minimization/Maximization: In general the optimization is to minimize the total weight of the tour, however, we can consider to maximize the total weight (see it as *profit* instead of *cost*). Note that with respect to hardness both minimization and maximization versions are equivalent, replace w_{ij} by $-w_{ij}$ (add a suitable constant to make weights positive). However, with respect to approximability both the versions differ as will be seen in later sections.

Metric/Non-metric: In the metric case the weights satisfy *triangle inequality* i.e.,

$$w_{ij} \leq w_{ik} + w_{kj}, \forall i, j, k \in V$$

Again note that any non-metric instance can be converted into a metric instance, replace w_{ij} by $w_{ij} + M$, where M is suitably large constant.

Parameterized triangle inequality: In this case the triangle inequality is weakened and it is enough for the weights to satisfy

$$w_{ij} \leq \beta(w_{ik} + w_{kj}), \forall i, j, k \in V$$

where β is the parameter. For a given instance, we can easily compute β as $\max\left\{\frac{w_{ij}}{w_{ik}+w_{kj}}\right\}, \forall i, j, k \in V$

Bounded metric (a, b)**:** In this case the weights (only integral) are taken from the interval $[a, b]$. The specific cases $(1, 2)$ for minimization and $(0, 1)$ for maximization have been well studied. The $(1, 2)$ case for minimization is of special importance as it is metric and is the one that is generally used to provide hardness results [5].

Euclidean: In the euclidean case the nodes are considered as points in R^d and weights are given by their euclidean distance. Euclidean TSP is NP-complete [6].

Reoptimization: In this case we are given a graph along with the optimum tour and we are asked to find the optimum tour when small modifications (adding a new node, deleting a node) are made to the graph.

In addition to those mentioned above, there are many other variants like Generalized TSP, Vehicle routing problem [7], Planar TSP [8], Online TSP [9], etc.

Problem	Notation
min, metric, sym	Δ-TSP
min, metric, asym	Δ-ATSP
min, non-metric, sym	TSP
min, non-metric, asym	ATSP
min, weights 1 & 2, sym	(1,2)-TSP
min, weights 1 & 2, asym	(1,2)-ATSP
min, parameterized tri. ineq., sym	Δ_β-TSP
min, parameterized tri. ineq., asym	Δ_γ-ATSP
min, reoptimization, sym	RTSP
max, metric, sym	Δ-MaxTSP
max, metric, asym	Δ-MaxATSP
max, non-metric, sym	MaxTSP
max, non-metric, asym	MaxATSP
max, weights 0 & 1, sym	(0,1)-MaxTSP
max, weights 0 & 1, asym	(0,1)-MaxATSP
max, reoptimization, sym	MaxRTSP
min, euclidean	ETSP

Table 1: Notations for various TSPs

3 Solving TSP

3.1 Exact Algorithms

The straight forward *brute force* approach of solving TSP by enumerating all possible tours is clearly not feasible, even for a moderate value of n, as the number of tours is $O(n!)$.

However, by using the *dynamic programming* approach, it can be solved in $O(n^2 2^n)$ which is better than $O(n!)$. The applicability of dynamic programming follows from the observation below

$$c(i, S) = \min_{j \in S}\{w_{ij} + c(j, S - \{j\})\}$$

where $c(i, S)$ represents the total weight of the shortest path from vertex i to the starting vertex, say 1, going through all the vertices in S. Now, the original problem can be solved by computing $c(1, V - \{1\})$. See [10] for analysis.

Another approach that is generally used to attack NP-Hard problems is *branch and bound*. This is similar to the brute force approach in the sense that we start out enumerating (branch) all possible tours, however, we also start with a bound on the optimal tour cost (and keep updating it as we find better bounds), and skip all those tours about which we are sure that their cost crosses the bound. Branch and bound has been found to perform very well for practical TSP instances. For further information on branch and bound refer [1].

3.2 Heuristics

For most practical situations it is sufficient if we can provide a *near optimal* tour instead of the optimal. This motivates us to try out heuristics which are fast to implement and perform well in general. Early research in TSP has resulted in various heuristics. As a sample we will look at the most natural heuristic and also give an instance for which it outputs the worst tour.

The nearest neighbour heuristic (Table 2) always chooses the next vertex as the one that is nearest to the present vertex. The algorithm, when invoked with $\sigma(1) = 1$, returns the tour $\langle 1, 3, 2, 4, 1\rangle$ (of weight 30 which is the worst tour) when run on the instance in Figure 1.

- Input: Graph G with weights w_{ij} and a starting vertex $\sigma(1)$.
- Ouput: TSP tour $\langle \sigma(1), \sigma(2), \cdots, \sigma(n), \sigma(1)\rangle$.

1. Initialize $S = \{1, 2, \cdots, n\} - \{\sigma(1)\}$.
2. For $k = 2, 3, \cdots, n$
   (a) Choose $\sigma(k)$ such that
   $$w_{\sigma(k-1),\sigma(k)} = \min_{s \in S}\{w_{\sigma(k-1),s}\}$$
   (b) $S = S - \{\sigma(k)\}$.

Table 2: Nearest Neighbour for ATSP

3.3 Other Approaches

TSP, because of its wide applicability, has been a testbed for various optimization methodoligies like heuristics (see *Ch 4.6* of [11]), genetic algorithms [12], neural nets [13], etc.

The other direction in which these problems are handled is to look at certain special classes of graphs for which they may turn out to be polynomially solvable. The special classes that are of interest are those that are rich and include most instances that come up in the application that is under consideration.

4 Approximation Algorithms

Approximation algorithms can be regarded as formalization of heuristics. That is, for approximation algorithms one wants a provable bound on the quality of the solution and the running time. There are various notions of quality. We consider the definition given below.

Definition 1 *Let Π be a minimization problem with the objective function obj_Π, and let δ be a function, $\delta : \mathbb{Z}^+ \to \mathbb{Q}^+$, with $\delta \geq 1$. An algorithm $\mathcal{A}$ is said to be a factor δ approximation algorithm for Π if, on each instance I, $\mathcal{A}$ produces a feasible solution s for I such that $obj_\Pi(I, s) \leq \delta(|I|).OPT(I)$, and the running time of $\mathcal{A}$ is polynomial in $|I|$.*

Similar definition can be given for approximation algorithm for maximization problem. Note that the closer δ is to 1, the better is the approximation algorithm.

NP-hard problems vary greatly in their approximability in the sense that certain problems are not approximable for any polynomially computable δ, some with δ that is function of $|I|$, some with a constant δ, and some that are approximable to any required constant δ. The approximation algorithms for a problem that achieve any given approximation factor are refered as *polynomial time approximation schemes* or *PTAS*.

Theorem 1 *For any polynomial time computable function $\alpha(n)$, TSP cannot be approximated within a factor of $\alpha(n)$, unless $P = NP$.*

For a proof of Theorem 1 and more on approximation algorithms see Vazirani [14].

4.1 Tools

The following tools have been repeatedly exploited in various approximation algorithms for TSP. For the remainder of this section assume the minimization TSP, similar arguments apply for maximization TSP also.

Minimum spanning tree: A minimum weight tree that spans all the vertices of a given graph is called its minimum spanning tree (MST). Note that $w(MST) \leq OPT$, as removing any edge from T_{OPT} makes it a spanning tree.

Minimum matching: A matching with minimum possible weight and which leaves at most one unmatched vertex is called a mimimum matching (M). Note that $w(M) \leq OPT/2$, as T_{OPT} can be seen as a collection of two matchings.

Minimum cycle cover: A cycle cover is a collection of disjoint cycles that spans all the verties of the given graph. A cycle cover with minimum possible weight is called a minimum cycle cover (C). Note

that $w(C) \leq OPT$, as T_{OPT} is also a cycle cover. The problem of finding minimum cycle cover is often refered as *assignment problem.*

Subtour patching: Subtour patching is the technique of joining the cycles in a cycle cover into a single tour. Careful analysis of the weight of edges involved in the patching often provides good approximations.

There are polynomial time algorithms for finding MST, M and C [15].

4.2 Classical Algorithms

Christofides algorithm [16] for Δ-TSP and Repeated assignment algorithm [17] for Δ-ATSP have been the best approximation algorithms for nearly 30 years.

1. Find a minimum spanning tree of G, say MST.
2. Compute a minimum weight matching, M, on the set of odd-degree vertices of MST. Add M to T and obtain an Eulerian graph.
3. Find an Euler tour, T_e, of this graph.
4. Output the tour that visits vertices of G in the order of their first apperance in T_e. Let T_o be this tour.

Table 3: Christofides for Δ-TSP

Theorem 2 *Christofides algorithm (Table 3) achieves a factor 3/2 approximation for Δ-TSP.*

Proof:

- $w(T_e) \leq w(MST) + w(M) \leq 3OPT/2$
- $w(T_o) \leq w(T_e)$ because of triangle inequality
- $\Rightarrow w(T_o) \leq 3OPT/2$ □

Theorem 3 *Repeated assignment algorithm (Table 4) achieves a factor $O(\log n)$ approximation for Δ-ATSP.*

Proof:

- Let S_j denote the cycle cover found in j^{th} iteration of the while loop.
- $w(S_j) \leq OPT$
- $w(T_o) \leq \sum w(S_j)$
- $\Rightarrow w(T_o) \leq \log_2(n)OPT$, as the while loop can repeat for at most $\log_2(n)$ times. □

- ASSIGN(.) solves the assignment problem i.e., finds the minimum cycle cover.
- TOUR(.) makes a valid tour out of edges found so far.

1. Intialize $P = \phi, k = 0, G' = G$.
2. While $k \neq 1$
   (a) Let $S = \{P_1, P_2, \cdots, P_k\}$ be the solution obtained by calling $ASSIGN(G')$.
   (b) $V' = \phi$.
   (c) For $i = 1$ to k
      i. Choose $v_i \in P_i$.
      ii. $V' = V' \cup \{v_i\}$.
      iii. $P = P \cup \{P_i\}$.
   (d) G' be the induced graph on V'.
3. Return the tour, T_o, obtained by calling $TOUR(P)$.

Table 4: Repeated Assignment for Δ-ATSP

5 Literature Survey

Approximation algorithms with improved factors are being continously found for almost all variants given in Section 2. Table 5 lists the best approximation factors available to date for some of the variants.

In the remainder of this section we briefly describe a sample of 4 algorithms highlighting the essential ideas in each case. Being short of space, proofs have been avoided or are kept brief. For complete details see the corresponding references.

5.1 7/12 algorithm for (0,1)-MaxATSP

This algorithm, given by Vishwanathan [30], is the first algorithm to achieve non-trivial factor for (0,1)-MaxATSP. The basic tool is subtour patching. We mention that 2-cycles i.e., cycles of length two in the cycle cover are the main cause of poor approximations for asymmetric case when compared to symmetric case. The paper gives a nice way of handling 2-cycles. The essential idea is to give two algorithms, one that performs well when there are lot of 2-cycles and one that performs well when there are very few 2-cycles.

We say vertices u and v are *balanced* if $w_{uv} = w_{vu}$.

Problem	Factor
Δ-TSP	$3/2$ [16]
Δ-ATSP	$4/3 \log_3 n$ [18]
(1,2)-TSP	$8/7$ [19]
(1,2)-ATSP	$5/4$ [20]
Δ_β-TSP, $\beta < 1$	$3/2$, depending on β [21]
Δ_γ-ATSP, $\gamma < 1$	$\frac{1}{1-\frac{1}{2}(\gamma+\gamma^3)}$ [22]
Δ_β-TSP, $\beta > 1$	$min\{4\beta, \frac{3}{2\beta^2}\}$ [23, 24, 25]
Δ_γ-ATSP, $\gamma > 1$	$min\{\frac{3\gamma^2+\gamma}{2}, 4\gamma\}$ [24, 25]
Δ-MaxTSP	$7/8 - o(1)$ [26]
Δ-MaxATSP	$31/40$ [27]
MaxTSP	$61/81 - o(1)$ [28]
MaxATSP	$2/3$ [18]
(0,1)-MaxTSP	$3/4$ [20]
(0,1)-MaxATSP	$3/4$ [20]
ETSP	$PTAS$ [29]

Table 5: Best Approximation Factors

Let l denote the number of weight one edges between balanced pairs of vertices.

1. Modify wieghts:
   if ($w(u \to v) = 1$ & $w(v \to u) = 0$)
   then set $w(u \to v) = 4/3$
2. Find maximum weight cycle cover.
3. Patch the cycles.

Table 6: Algorithm 1 for (0,1)-MaxATSP

Lemma 1 *The weight of the tour produced by Algorithm 1 (Table 6) is at least* $2OPT/3 - l/6$.

Lemma 2 *The weight of the tour produced by Algorithm 2 (Table 7) is at least* $OPT/2 + l/6$.

Theorem 4 *There is a polynomial approximation algorithm for (0,1)-MaxATSP which finds a tour of length at least* $7/12$ *of the optimal.*

Proof:

- The proof of the Lemma 1 and Lemma 2 is based on the observation that the weight of the optimum tour in the modified graph is at least $4(OPT - l)/3 + l$ in the case of Algorithm 1 and at least $4l/3 + (OPT - l)$ in the case of Algorithm 2.
- At least one of the algorithms performs as well as their average i.e., $\geq (7/12).OPT$. □

1. Create an undirected graph $G' = (V', E')$ with weight function w':
   $V' = \{u' | u \in V\}$
   $w'_{u'v'} = 4/3$ if $\langle u, v \rangle$ are balanced with wt. 1
   $w'_{u'v'} = 0$ if $\langle u, v \rangle$ are balanced with wt. 0
   $w'_{u'v'} = 1$ if $\langle u, v \rangle$ are not balanced
2. Find maximum weight cycle cover in G'. (Note that each cycle corresponds to two directed cycle covers - clockwise & counter clockwise.)
3. Select the direction that gives more weight.
4. Patch the cycles.

Table 7: Algorithm 2 for (0,1)-MaxATSP

5.2 $\frac{\gamma}{1-\gamma} + \gamma$ algorithm for Δ_γ-ATSP, $\gamma < 1$

This algorithm, given by Chandran and Ram [31], is the first constant factor approximation for Δ_γ-ATSP, $\gamma < 1$. Again, subtour patching is the basic tool. The essential idea is to try various patchings and return the best of them.

Lemma 3 *Let* $C = \{v_1, v_2, \cdots, v_k, v_{k+1} = v_1\}$ *be a directed cycle in* G *with* $2 \leq k \leq n$ *and* $\gamma \in [1/2, 1)$. *Let* $u \in V - C$, *then* $\exists (v_m, u)$ *such that* $w_{v_m,u} = \max\{w_{v_i,u} : i = 1, \cdots, k\}$ *and* $w_{v_m,u} \leq (\frac{\gamma}{1-\gamma}) w_{v_m,v_{m+1}}$.

Lemma 4 *If a cycle cover* C *is input to Hamiltonian Path algorithm (Table 8), it outputs an Hamiltonian path* H *of cost* $\leq (\frac{\gamma}{1-\gamma}) w(C)$. *In particular, if* C *is a minimum cost cycle cover,* $w(H) \leq (\frac{\gamma}{1-\gamma}) OPT$.

Theorem 5 *The Hamiltonain Path algorithm (Table 8) can be used to get a tour of weight at most* $(\frac{\gamma}{1-\gamma} + \gamma) OPT$.

Proof:

- Lemma 3 is proved by assuming that the required edge doesnot exist leading to contradiction with the triangle inequality.
- Lemma 4 follows from Lemma 3.
- Run the Hamiltonian Path algorithm n times, once with each vertex i, thus obtaining an Hamiltonian path H_i ending at vertex i.

- Input: A cycle cover $C = \{C_1, C_2, \cdots, C_k\}$; a vertex $u_1 \in V$. Assume $u_1 \in C_1$.
- Ouput: An Hamiltonian path of G ending at vertex u_1.

1. Consider the edge $(u_1, v_1) \in C_1$ and remove it. Now, we have a path from v_1 to u_1.
2. For $i = 2 \cdots k$ do:
   (a) Get an edge (u_i, v_i) in C_i such that $w_{u_i,v_{i-1}} = \max\{w_{x,v_{i-1}} : \forall x \in C_i\}$ and remove it.
   (b) Add the edge (u_i, v_{i-1}). We have a path from v_i to u_1.
3. We have an Hamiltonian path from v_k to u_1. Output this path.

Table 8: Hamiltonian Path

- Each H_i is converted into a tour T_i by adding the missing edge.
- Select the minimum wieght tour among $T_i s$.
- Because of the triangle inequality we are assured that weight of at least one of the missing edges is at most γOPT. This, together with the Lemma 4, implies the theorem. □

5.3 3/4 algorithm for `(0,1)-MaxATSP`

This algorithm was given by Blaser [20]. The basic tool is linear programming. The idea is to find an admissible multigraph that is 2-path colorable.

Definition 2 *A multigraph is called 2-path colourable if its edges can be coloured with two colors such that each color class is a collection of node disjoint paths.*

Definition 3 *A directed multigraph is called admissable if (i) the indegree and outdegree of each node is at most two, (ii) the total degree of each node is at most three, and (iii) between each pair of nodes, there are at most two edges (counted with multiplicities).*

Lemma 5 *If there are no 2-cycles on a cycle in an admissible multigraph G, then G is 2-path colorable, where 2-cycle on a cycle is a directed cycle $v_1, \ldots, v_k, v_1$ with $k \geq 3$ such that (v_i, v_{i-1}) is also an edge for some i.*

1. Solve the relaxed LP for admissible multigraph, giving H^*.
2. Round H^* to an admissible multigraph S.
3. Find a 2-path colorable admissible graph S' with the same weight as S.
4. Find a 2-path coloring for S', partitioning the edges into two sets.
5. Return the set with larger weight after patching.

Table 9: 3/4 algorithm for `(0,1)-MaxATSP`

Theorem 6 *The algorithm in Table 9 is factor* 3/4 *approximation algorithm for `(0,1)-MaxATSP`.*

Proof:

- A simple modification of optimum TSP tour gives an admissible multigraph of weight at least $(3/2)OPT$, assuming n is even.
- This, with Lemma 5, implies the theorem. □

5.4 PTAS for `ETSP`

This algorithm, given by Arora [29], is a breakthrough result in the area of approximations. The basic idea of the algorithm is to perform recursive geometric partitioning of the instance and solve the subinstances thus produced using dynamic programming.

1. For each possible shift (a, b)
   (a) Consider the (a, b)-shifted dissection.
   (b) Compute the optimum portal-tour.
   (c) Let $T_{(a,b)}$ be the tour after removing the portals.
2. Return the minimum weight tour over all $T_{(a,b)}$s.

Table 10: PTAS for `ETSP`

Definition 4 *Bounding box is the smallest axis aligned square that contains all the input nodes.*

Definition 5 *A dissection of the bounding box is a recursive partitioning into smaller squares, until they are of unit size.*

Definition 6 *Let a, b be integers. An (a, b)-shifted dissection is defined as the dissection obtained by shifting the dissection lines modulo L, where a is the shift for vertical lines and b is for horizontal lines. Note that some of the squares are wrapped-around in the shifted dissection.*

Definition 7 *Portals are the designated points on the edges of the squares through which a tour can cross the boundary of any region in the dissection. Each square has a portal at each of its 4 corners and $m = O((1/\epsilon)\log L)$ equally spaced portals on each edge.*

Definition 8 *A tour is said to be a portal-tour with respect to a dissection if it crosses each edge of each square in the dissection at most $r = O(1/\epsilon)$ times and always through one of the portals, however it is allowed to go through a portal multiple times.*

Each subproblem is specified by (i) a square (ii) a multiset of $\leq r$ portals for each edge and (iii) a pairing of these $\leq 4r$ portals. We can look at (ii) and (iii) as specifying an interface for the square. Dynamic programming can be used because of the observation below

$$c(S, I) = \min_{I_1, I_2, I_3, I_4} \sum_{i=1}^{4} c(S_i, I_i)$$

Theorem 7 *For a randomly picked (a, b)-shifted dissection,*

$$Pr\big(OPT_{(a,b)} \leq (1+\epsilon)OPT\big) \geq 1/2$$

where $OPT_{(a,b)}$ is the cost of an optimum portal-tour with respect to (a, b)-shifted dissection.

Proof:
The proof is based on estimating the increase in weight by *modifying* an optimum tour into a portal tour. □

Theorem 8 *There exists a PTAS for ETSP.*

Proof:
The size of the look-up table in the dynamic program and the time needed to compute an entry of the table are poylnomial in n, for any given ϵ. This, together with Theorem 7, implies a *PTAS* for ETSP. □

6 Conclusions and Future Work

Even after continous effort and improved approximations for various versions of TSP, there still remains a lot to be achieved in this respect. One of the intriguing problem being "Does there exist a constant factor approximation for ATSP?"

Our future work would mainly focus on

1. Improving bound for (1,2)-TSP [19].
2. Looking for an interesting special class of graphs for which TSP would be polynomially solvable [32].
3. Improving bounds for RTSP and MaxRTSP [33].
4. Performing computational experiments with various heuristics.

References

[1] E. L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Khan, and D. B. Shmoys, editors. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* John Wiley & Sons, 1985.

[2] Alexander Schrijver. On the history of combinatorial optimization (till 1960). In K. Aardal, G.L. Nemhauser, and R. Weismantel, editors, *Handbook of Discrete Optimization*, pages 1–68. Elsevier, 2005.

[3] Charikar, Motwani, Raghavan, and Silverstein. Constrained TSP and low-power computing. In *WADS*, 1997.

[4] S. R. Kosaraju, J. K. Park, and C. Stein. Long tours and short superstrings. In *FOCS*, 1994.

[5] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18:1–11, 1993.

[6] C. H. Papadimitriou. The euclidean traveling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237–244, 1977.

[7] G. Gutin and A. P. Punnen, editors. *The Traveling Salesman Problem and Its Variations.* Kluwer, 2002.

[8] Michelangelo Grigni, Elias Koutsoupias, and C. H. Papadimitriou. An approximation scheme for planar graph TSP. In *FOCS*, 1995.

[9] Ausiello, Bonifaci, and Laura. The on-line asymmetric traveling salesman problem. In *WADS*, 2005.

[10] Ellis Horowitz and Sartaj Sahni. *Fundamentals of Computer Algorithms.* Galgotia Publications, 1998.

[11] Jonathan L. Gross and Jay Yellen, editors. *Handbook of Graph Theory.* CRC, 2003.

[12] J. Y. Potvin. Genetic algorithms for the traveling salesman problem. *Annals for Operations Research*, 63(3):337–370, 1996.

[13] J. Y. Potvin. The traveling salesman problem: a neural network perspective. *ORSA Journal on Computing*, 5(4):328–348, 1993.

[14] Vijay V. Vazirani. *Approximation Algorithms.* Springer-Verlag, 2001.

[15] Douglas B. West. *Introduction to Graph Theory.* Prentice-Hall, 1996.

[16] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical report, GSIA, Carnegie-Mellon University, Pittsburgh, 1976.

[17] A. M. Frieze, G. Galbiati, and F. Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982.

[18] Kaplan, Lewenstein, Shafrir, and Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *Journal of the ACM*, 52, 2005.

[19] Piotr Berman and Marek Karpinski. 8/7 approximation algorithm for (1, 2)-TSP. In *SODA*, 2006.

[20] Markus Bläser. A 3/4-approximation algorithm for maximum ATSP with weights zero and one. In *APPROX*, 2004.

[21] Böckenhauer, Hromkovic, Klasing, Seibert, and Unger. Approximation algorithms for the TSP with sharpened triangle inequality. *Information Processing Letters*, 75:133–138, 2000.

[22] Markus Bläser. An improved approximation algorithm for the asymmetric TSP with strengthened triangle inequality. In *ICALP*, 2003.

[23] Böckenhauer, Hromkovic, Klasing, Seibert, and Unger. Towards the notion of stability of approximation for hard optimization tasks and the traveling salesman problem. *ECCC*, 6(31), 1999.

[24] Bender and Chekuri. Performance guarantees for the TSP with a parameterized triangle inequality. In *WADS*, 1999.

[25] Thomas Andreae and Hans-Jurgen Bandelt. Performance guarantees for approximation algorithms depending on parametrized triangle inequalities. *SIAM Journal on Discrete Mathematics*, 8(1):1–16, 1995.

[26] Chen and Nagoya. Improved approximation algorithms for metric max TSP. In *ESA*, 2005.

[27] Markus Bläser, L. Shankar Ram, and Maxim Sviridenko. Improved approximation algorithms for metric maximum ATSP and maximum 3-cycle cover problems. In *WADS*, 2005.

[28] Zhi-Zhong Chen, Yuusuke Okamoto, and Lusheng Wang. Improved deterministic approximation algorithms for max TSP. *Information Processing Letters*, 95(2):333–342, 2005.

[29] S. Arora. Polynomial-time approximation scheme for Euclidean TSP and other geometric problems. In *FOCS*, 1996.

[30] Sundar Vishwanathan. An approximation algorithm for the asymmetric travelling salesman problem with distances one and two. *Information Processing Letters*, 44(6):297–302, 1992.

[31] L. Sunil Chandran and L. Shankar Ram. Approximations for ATSP with parametrized triangle inequality. In *STACS*, 2002.

[32] J. Mark Keil. Finding Hamiltonian circuits in interval graphs. *Information Processing Letters*, 20(4):201–206, 1985.

[33] Giorgio Ausiello, Bruno Escoffier, Jérôme Monnot, and Vangelis Th. Paschos. Reoptimization of minimum and maximum traveling salesman's tours. In *SWAT*, 2006.