Approximating Asymmetric TSP in Exponential Time

Alexander Golovnev

New York University
alex.golovnev@gmail.com

Abstract. Let G be a complete directed graph with n vertices and integer edge weights in range $[0, M]$. It is well known that an optimal Traveling Salesman Problem (TSP) in G can be solved in 2^n time and space (all bounds are given within a polynomial factor of the input length, i.e., $\mathrm{poly}(n, \log M)$) and this is still the fastest known algorithm. If we allow a polynomial space only, then the best known algorithm has running time $4^n n^{\log n}$. For TSP with bounded weights there is an algorithm with $1.657^n \cdot M$ running time. It is a big challenge to develop an algorithm with 2^n time and polynomial space. Also, it is well-known that TSP cannot be approximated within any polynomial time computable function unless P=NP.
In this short note we propose a very simple algorithm that, for any $0 < \varepsilon < 1$, finds $(1+\varepsilon)$-approximation to asymmetric TSP in $2^n\varepsilon^{-1}$ time and $\varepsilon^{-1} \cdot \mathrm{poly}(n, \log M)$ space. Thereby, for any fixed ε, the algorithm needs 2^n steps and polynomial space to compute $(1+\varepsilon)$-approximation.

Keywords: TSP; traveling salesman problem; asymmetric traveling salesman problem; approximation scheme; exponential algorithm.

1 Introduction

1.1 Problem Statement

A cycle in a graph is called Hamiltonian if it visits every vertex exactly once. The Traveling Salesman Problem (TSP) is, given a complete graph $G = (V, E)$ with nonnegative integer edge weights, to find a Hamiltonian cycle of minimal weight. Metric TSP is TSP restricted to graphs with edge costs satisfying the triangle inequality. Directed (or asymmetric) TSP is a version of TSP on directed graphs.

1.2 Exact Algorithms

Bellman [1], Held and Karp [2] developed a dynamic programming algorithm for TSP. It solves TSP in $O^*(2^n)$[1] time and space. This is

[1] $O^*(\cdot)$ suppresses polynomial factors of the input length, that is $\mathrm{poly}(n, \log M)$ where n is the number of vertices in the input graph and M is the maximal edge weight.

still the fastest algorithm for general TSP. The best known polynomial-space algorithm has running time $O^*(4^n n^{\log n})$ (Gurevich and Shelah [3], Björklund and Husfeldt [4]).

Koivisto and Parviainen [5] described a way to solve TSP in time $O^*(2^{2n-t}n^{\log(n-t)})$ and space $O^*(2^t)$ for any $t = n, n/2, n/4....$ They also showed time-space trade-off such that for any $\sqrt{2} < S < 2$, TSP can be solved in $O(T^n)$ time and $O(S^n)$ space with $TS < 4$.

Kohn, Gottlieb and Kohn [6], Karp [7], Bax and Franklin [8] presented an algorithm for TSP with $O^*(2^n \cdot M)$ running time and $O^*(M)$ space where M is the maximal edge weight. Lokshtanov and Nederlof [9] described a way to solve TSP in $O^*(2^n \cdot M)$ time with only $O^*(1) = \text{poly}(n)$ space. Björklund [10] developed a Monte Carlo algorithm for symmetric TSP with running time $O(1.657^n \cdot M)$ and exponentially small probability of error.

No better bounds are known for Metric TSP. It is a big challenge to develop $O^*(2^n) = 2^n \cdot \text{poly}(n, \log M)$-time algorithm for TSP with polynomial space (see Open Problem 2.2.b, Woeginger [11]).

Eppstein [12] proposed an algorithm for TSP in cubic graphs with running time $O(1.260^n)$ as well as an algorithm solving TSP in graphs of degree 4 with running time $O(1.890^n)$. Iwama and Nakashima [13] improved the bound for TSP in cubic graphs to $O(1.251^n)$. Gebauer [14] proposed an algorithm for TSP in graphs of degree 4 with running time $O(1.733^n)$ and exponential space.

Björklund, Husfeldt, Kaski and Koivisto [15] developed an algorithm with running time $O((2-\varepsilon)^n)$ for TSP in bounded-degree graphs (ε depends only on the degree of a graph).

For the Undirected Hamiltonian Cycle Problem (HC) the $O^*(2^n)$ barrier was improved. Björklund [10] discovered an algorithm solving HC in $O(1.657^n)$ steps using only polynomial space. The algorithm has $O^*(2^{n/2})$ running time on bipartite graphs.

1.3 Approximation Algorithms

General TSP It is well-known that general TSP cannot be approximated within any polynomial time computable function unless P=NP (Sahni and Gonzalez [16]). TSP and Metric TSP are known to be strongly NP-hard problems (see, e.g., Garey and Johnson [17]).

Undirected Metric TSP There is a 2-approximation algorithm for Metric TSP (due to Rosenkrantz, Stearns and Lewis [18]). The best known approximation guarantee 3/2 is achieved by Christofides' algorithm [19].

The best known lower bound is due to Karpinski, Lampis and Schmied [20]: Undirected Metric TSP cannot be approximated in polynomial time with a ratio better than $\frac{123}{122}$, unless P=NP.

Directed Metric TSP Frieze, Galbiati, Maffioli [21] showed that directed metric TSP can be approximated with $\log n$ ratio, Bläser [22]

improved this bound to $0.999 \log_2 n$. Kaplan, Lewenstein, Shafrir and Sviridenko [23] developed a $\frac{4}{3} \log_3 n$-approximation algorithm. Feige and Singh [24] provided $\frac{2}{3} \log_2 n$ approximation. Finally, Asadpour, Goemans, Madry, Gharan and Saberi [25] improved the ratio to $O(\frac{\log n}{\log \log n})$.

Karpinski, Lampis and Schmied [20] proved that Directed Metric TSP cannot be approximated in polynomial time with a ratio better than $\frac{75}{74}$, unless P=NP.

Other Special Cases Gharan, Saberi and Singh [26] developed an algorithm with approximation factor $3/2 - \varepsilon$ for graphic TSP. Mömke and Svensson [27] gave a 1.461-approximation algorithm for graphic TSP. Mucha [28] improved the approximation factor to 1.444. A major breakthrough was obtained by Arora [29],[30] and Mitchell [31], where a polynomial-time approximation scheme (PTAS) was found for Euclidean TSP. Very recently Bartal et al. [32] found PTAS for Doubling TSP.

(1,2)-TSP is TSP restricted to instances where all weights are equal to one or two. Even this problem is known to be MAX-SNP hard (Papadimitriou, Yannakakis [33]). Berman and Karpinski [34] designed a 8/7-approximation algorithm for (1,2)-TSP.

Approximating TSP in exponential time Boria, Bourgeois, Escoffier, and Paschos [35] developed exponential-time approximation scheme for TSP. They suggested the following $(1 + \varepsilon)$-approximation algorithms for Metric TSP:

- using $O^*(4^{(1-\varepsilon/2)n} n^{\log n})$ time and polynomial space for any $\varepsilon \leq 2/3$,
- using $O^*(2^{(1-\varepsilon/2)n})$ time and exponential space for any $\varepsilon \leq 2/5$.

1.4 Our Results

In this note, we present a very simple algorithm that, for any fixed ε, needs $O^*(2^n)$ steps and polynomial space to compute a $(1 + \varepsilon)$-approximation. As stated above the best known approximation ratio achievable in polynomial time for Undirected Metric TSP is 1.5, while for Directed Metric TSP the best known approximation ratio achievable in polynomial time is $O(\frac{\log n}{\log \log n})$. And we know that if P≠NP then we cannot find $\frac{117}{116}$-approximation in polynomial time (see section 1.3). The presented algorithm is able to find, e.g., a 1001/1000-approximation in $O^*(2^n)$ steps using only a polynomial space.

The Knapsack problem is, given a set of objects with specified sizes and profits, find a subset of objects whose total size is bounded by knapsack capacity and total profit is maximized. Our algorithm is inspired by FPTAS for the Knapsack problem due to Ibarra and Kim [36]. They use the fact that Knapsack can be solved by a simple pseudopolynomial algorithm working in time $O(nW)$ where n is the number of items and W is the total weight. The algorithm first divides all input weights by some $\alpha(\varepsilon, n, W)$ and then invokes the pseudopolynomial algorithm. The resulting answer might not be optimal as the weights are not just divided by α, but also rounded after that. However a simple analysis shows that rounding does not affect the result too much.

We use the same idea. By OPT we denote the cost of an optimal solution. To get a polynomial-space approximation algorithm for Asymmetric TSP we first divide all edge weights by a big enough number α and then use an algorithm based on inclusion-exclusion. Then it turns out that the running time of inclusion-exclusion algorithm is $2^n \cdot \mathrm{OPT}/\alpha$ and the weight of the resulting cycle is at most $\mathrm{OPT} + \alpha n$. We choose α satisfying the following two inequalities:

$$\frac{\varepsilon \mathrm{OPT}}{\mathrm{poly}(n)} \leq \alpha \leq \frac{\varepsilon \mathrm{OPT}}{n}\,.$$

We show that the former inequality ensures that the running time of the algorithm is $O^*(2^n \varepsilon^{-1})$ and the space is $O^*(\varepsilon^{-1})$. The latter inequality guarantees that the algorithm gives a $(1+\varepsilon)$-approximation. Finally, we show that such α can be found within the required time and space.

To compare the obtained approximation with the result by Boria et al. [35] we note that, for any fixed $\varepsilon > 0$, the algorithm by Boria et al. either requires exponential space or has running time $O^*(c^n)$ where $c > 2$.

2 Algorithm

Throughout the paper we assume that edge weights are positive integers. By $\mathrm{OPT}(G)$ we denote the weight of an optimal traveling salesman cycle of the graph G. Our goal is to find a tour of weight $\leq (1+\varepsilon)\mathrm{OPT}(G)$ for a given ε. We omit G if it is clear from the context.

2.1 Inclusion-Exclusion Algorithm

Inclusion-exclusion algorithms are based on the following well-known formula (the proof can be found, e.g., in Bax [37]).

Theorem 1. *Let X be a finite set and f, g be two functions defined on all subsets of X. Let also for each $A \subseteq X$*

$$g(A) = \sum_{B \subseteq A} f(B)\,.$$

Then for each $A \subseteq X$

$$f(A) = \sum_{B \subseteq A} (-1)^{|A|-|B|} g(B). \tag{1}$$

E.g., to check whether a graph $G(V, E)$ contains a Hamiltonian path from s to t in $O^*(2^n)$ time and polynomial space one defines, for $A \subseteq V$, $f(A)$ as the number of walks (a walk is a path that can pass through the same vertex more than once) of length $n-1$ that go through all vertices of A and contain no vertex outside of A. Hence, $f(V)$ is the number of Hamiltonian paths from s to t. It is not difficult to see that then $g(A)$ is the number of walks of length $n-1$ such that all their vertices belong to A. It remains to note that $g(A)$ can be computed in polynomial time (by

a dynamic programming algorithm or by raising the adjacency matrix of the graph induced by A to the $(n-1)$-th power).

Kohn, Gottlieb and Kohn [6], Karp [7], Bax and Franklin [8] showed how to solve TSP in $O^*(2^n \cdot M)$ time and $O(n \cdot M + n^2)$ space where M is the maximum edge weight. First, we need to show that the running time of inclusion-exclusion algorithm is in fact $O^*(2^n \cdot \mathrm{OPT})$ and the space is $O(n \cdot \mathrm{OPT} + n^2)$.

The decision version of TSP is to determine if a given graph G contains a Hamiltonian cycle of weight $\leq k$ for a given decision parameter k. We show that the decision version of TSP can be solved in time $O^*(2^n k) = 2^n k \cdot \mathrm{poly}(n, \log M)$ and space $O^*(k) = k \cdot \mathrm{poly}(n)$.

We denote the number of circuits (closed walks) in $U \subseteq V$ of weight $\leq k$ with exactly n edges by $c(U)$. Note that $c(U)$ can be computed by dynamic programming in time $O(kn^4)$ and space $O(kn^3)$. For this, let D be an $n \times k \times n$ matrix such that $D[v, l, t]$ is the number of walks w such that w has total weight l and exactly t edges and ends in the vertex v. The matrix D can be filled in using the following simple observation:

$$D[v, l, t] = \sum_{u \in U} D[u, l - l(u, v), t - 1]\,.$$

Thus, each cell can be computed in $O(n)$ using the previously computed values. Therefore the total running time will be $O(kn^3)$ and the space is $O(kn^2)$ (the given upper bounds on the running time and space are w.r.t. arithmetic operations. Since all the entries in D does not exceed 2^n, the number of bit operations can only be n times larger). By noting that the layer $D[\cdot, \cdot, t]$ only depends on the layer $D[\cdot, \cdot, t-1]$, the space complexity can be reduced by a factor of n. Finally, note that $c(U) = \sum_i D[i, k - e(i, 1), n - 1]$.

We apply (1) for $g(U) := c(U)$ and for $f(U)$ denoting the number of circuits of weight at most k with exactly n edges containing all vertices of U and no vertices outside of U.

Algorithm 1 InclExcl-Decision-TSP — solving decision version of Asymmetric TSP in $O^*(2^n k)$ time and $O^*(k)$ space.

Input: $G = (V, E)$ — complete weighted directed graph, k — decision parameter.
Output: the number of Hamiltonian cycles of weight at most k.

1: $res = 0$
2: **for** $U \subseteq V$ **do**
3: $\quad res = res + (-1)^{|V|-|U|} c(U)$
4: **end for**
5: **return** res

The correctness of the algorithm InclExcl-Decision-TSP follows from (1). The time complexity of the algorithm is $O^*(2^n k)$ and the space complexity is $O^*(k)$.

Binary search on k gives us the following algorithm (InclExcl-TSP) for optimization version of asymmetric TSP.

Algorithm 2 InclExcl-TSP — solving Asymmetric TSP in $O^*(2^n \cdot \mathrm{OPT})$ time and $O^*(\mathrm{OPT})$ space.

Input: $G = (V, E)$ — complete weighted directed graph.
Output: the weight of the shortest Hamiltonian cycle.

1: set $k = 1$ and double k until InclExcl-Decision-TSP$(G, k) > 0$
2: use binary search on the found interval to find the minimum k, s.t. InclExcl-Decision-TSP$(G, k) > 0$

Lemma 1. *The algorithm* InclExcl-TSP *solves Asymmetric TSP in time* $O^*(2^n \cdot OPT)$ *and space* $O^*(OPT)$.

Proof. We use binary search to find the minimum value k, such that InclExcl-Decision-TSP returns a positive number of cycles. The number of calls of InclExcl-Decision-TSP with $k \leq 2\mathrm{OPT}$ is $O(\log OPT)$ that is polynomial in the input size. The running time is $O^*(2^n \cdot \mathrm{OPT})$ and space is $O^*(\mathrm{OPT})$. □

Remark 1. The algorithm InclExcl-TSP is due to Karp [7]. We just emphasize here that the running time and space can be bounded by $O^*(2^n \cdot \mathrm{OPT})$ and $O^*(\mathrm{OPT})$, respectively.

Remark 2. The algorithm InclExcl-TSP can be adapted to produce a cycle itself rather than simply establishing its existence.

2.2 An Algorithm for Metric TSP

We denote the $(\log n)$-approximation algorithm mentioned in Subsection 1.3 for solving the Directed Metric TSP by Approx-Asym-TSP. We suggest the following approximation scheme for Symmetric and Asymmetric Metric TSP.

Algorithm 3 Approx-Metric-TSP — $(1+\varepsilon)$-approximation of Metric TSP in time $O^*(2^n\varepsilon^{-1})$ and space $O^*(\varepsilon^{-1})$.

Parameter: $\varepsilon > 0$ — approximation ratio.
Input: $G = (V, E)$ — complete weighted directed graph.
Output: a Hamiltonian cycle of weight at most $(1+\varepsilon)\mathrm{OPT}(G)$.

1: $\beta =$ Approx-Asym-TSP(G)
2: divide all edge weights by α: $w(e) = \lceil w(e)/\alpha \rceil$, where

$$\alpha = \frac{\beta \cdot \varepsilon}{n \log n}$$

3: **return** InclExcl-TSP(G)

Lemma 2. *The algorithm* Approx-Metric-TSP *finds a Hamiltonian cycle of weight* $\leq (1+\varepsilon)OPT(G)$ *in a graph* G.

Proof. Let OPT and OPT$'$ denote the weight of an optimal cycle in G before and after dividing the weights by α, respectively.
The weight RES of the found cycle is less or equal than $\alpha \cdot \mathrm{OPT}'$. Denote by $I = (e_1, \ldots, e_n)$ an optimal cycle of the input graph G. Then OPT$'$ is not greater than $\sum_{i \in I} \lceil w(e_i)/\alpha \rceil$. Thereby,

$$\mathrm{RES} \leq \alpha \cdot \mathrm{OPT}' \leq \alpha \cdot \sum_{i \in I} \lceil w(e_i)/\alpha \rceil \leq \sum_{i \in I} (w(e_i) + \alpha) = \mathrm{OPT} + \alpha n.$$

Since $\beta \leq \mathrm{OPT} \cdot \log n$, $\alpha n \leq \varepsilon \mathrm{OPT}$ and $\mathrm{RES} \leq (1+\varepsilon) \cdot \mathrm{OPT}$. □

Lemma 3. *The running time of the algorithm* Approx-Metric-TSP *is* $O^*(2^n + 2^n\varepsilon^{-1})$*, the space is* $O^*(\varepsilon^{-1})$.

Proof. The running time of step 1 is polynomial (see section 1.3).
By Lemma 1, the running time of step 3 is $O^*(2^n \cdot \mathrm{OPT}')$ and the space is $O^*(\mathrm{OPT}')$. The inequalities $\mathrm{OPT}' \leq \frac{\mathrm{OPT}}{\alpha} + n$ and $\beta \geq \mathrm{OPT}$ imply

$$\mathrm{OPT}' \leq \frac{\mathrm{OPT}}{\alpha} + n = \frac{\mathrm{OPT} n \log n}{\beta\varepsilon} + n \leq \frac{n \log n}{\varepsilon} + n = O^*(\varepsilon^{-1}).$$

Therefore, the running time of Approx-Metric-TSP is $O^*(2^n\varepsilon^{-1})$ and the space is $O^*(\varepsilon^{-1})$. □

2.3 An Algorithm for General TSP

In the previous subsection we considered only the metric version of TSP because the algorithm invoked a polynomial-time approximation algorithm to compute the bound β. Now we show a way to avoid using of approximation algorithm and extend the result to the nonmetric case of TSP. The only difference between the algorithms are steps 2-5 of finding 4-approximation of general TSP in exponential time.

Lemma 4. *The algorithm* Approx-TSP *finds a Hamiltonian cycle of weight* $\leq (1+\varepsilon)OPT(G)$ *in a graph* G.

Proof. We only need to show that after step 5, the value of β provides a 4-approximation of OPT, i.e., $\mathrm{OPT} \leq \beta \leq 4\mathrm{OPT}$. Then we use the same routine as in Approx-Metric-TSP.

Let β_{end} be the value of beta in the last iteration of the repeat-loop (before line 4). Since the G' of the final iteration of the repeat-loop contains a cycle of weight $\leq 2n$, G contains a cycle of weight $\leq 2n\alpha_{end} = 2\beta_{end}$, thus $\mathrm{OPT} \leq 2\beta_{end}$. For proving that $\mathrm{OPT} \geq \beta_{end}/2$ let $\beta_{prev} := \beta_{end}/2$ and let $\alpha_{prev} = \beta_{prev}/n$. Note that β_{prev} and α_{prev} are the values of α and β in the previous iteration of the repeat-loop. If G had a cycle of weight at most $\beta_{end}/2$ then in the previous round, the G' had a cycle of weight $\beta_{prev}/\alpha_{prev} + n = 2n$, contradicting the fact that the termination condition was not fulfilled.

Thereby, $2\beta_{end}$ is 4-approximation of OPT. □

Algorithm 4 APPROX-TSP — $(1+\varepsilon)$-approximation of TSP in time $O^*(2^n\varepsilon^{-1})$ and space $O^*(\varepsilon^{-1})$.

Parameter: $\varepsilon > 0$ — approximation ratio.
Input: $G = (V, E)$ — complete weighted directed graph.
Output: a Hamiltonian cycle of weight at most $(1+\varepsilon)\mathrm{OPT}(G)$.

1: $\beta = 1$
2: **repeat**
3: let G' be a graph obtained from G by dividing all edge weights by α': $w'(e) = \lceil w(e)/\alpha' \rceil$, where
$$\alpha' = \frac{\beta}{n}$$
4: $\beta = 2\beta$
5: **until** INCLEXCL-DECISION-TSP$(G', 2n) > 0$
6: divide all edge weights by α: $w(e) = \lceil w(e)/\alpha \rceil$, where
$$\alpha = \frac{\beta \cdot \varepsilon}{4n}$$
7: **return** INCLEXCL-TSP(G)

Lemma 5. *The running time of the algorithm* APPROX-TSP *is* $O^*(2^n + 2^n\varepsilon^{-1})$*, the space is* $O^*(\varepsilon^{-1})$.

Proof. The running time of step 5 is $O^*(2^n \cdot 2n) = O^*(2^n)$, the space is $O^*(2n) = O^*(1)$. The number of calls of INCLEXCL-DECISION-TSP is $O(\log_2(4 \cdot \mathrm{OPT})) = \mathrm{poly}(n, \log M)$. Along the lines of Lemma 3, the running time of APPROX-TSP is $O^*(2^n + 2^n\varepsilon^{-1})$ and the space is $O^*(\varepsilon^{-1})$. □

Remark 3. Note that it must be difficult to improve the running time of such an exponential approximation algorithm to $O^*(2^n \cdot \log \varepsilon^{-1})$ since using such an algorithm with $\varepsilon = (nM+1)^{-1}$ would solve TSP exactly in time $O^*(2^n)$ and polynomial space.

References

1. Bellman, R.: Dynamic Programming Treatment of the Travelling Salesman Problem. J. ACM 9 (1962) 61–63
2. Held, M., Karp, R.M.: A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied Mathematics 10(1) (1962) 196–210
3. Gurevich, Y., Shelah, S.: Expected Computation Time for Hamiltonian Path Problem. SIAM J. Comput. 16 (1987) 486–502
4. Björklund, A., Husfeldt, T.: Exact Algorithms for Exact Satisfiability and Number of Perfect Matchings. Algorithmica 52 (2008) 226–249
5. Koivisto, M., Parviainen, P.: A Space-Time Tradeoff for Permutation Problems. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms. SODA'10, Philadelphia,

PA, USA, Society for Industrial and Applied Mathematics (2010) 484–492
6. Kohn, S., Gottlieb, A., Kohn, M.: A Generating Function Approach to the Traveling Salesman Problem. In: ACN'77: Proceedings of the 1977 annual conference, New York, NY, USA (1977) 294–300
7. Karp, R.M.: Dynamic Programming Meets the Principle of Inclusion and Exclusion. Operations Research Letters 1(2) (1982) 49–51
8. Bax, E., Franklin, J.: A Finite-Difference Sieve to Count Paths and Cycles by Length. Inf. Process. Lett. 60 (1996) 171–176
9. Lokshtanov, D., Nederlof, J.: Saving space by algebraization. In: Proceedings of the 42nd ACM symposium on Theory of computing. STOC '10, ACM (2010) 321–330
10. Björklund, A.: Determinant Sums for Undirected Hamiltonicity. In: Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. FOCS'10, Washington, DC, USA, IEEE Computer Society (2010) 173–182
11. Woeginger, G.J.: Open Problems Around Exact Algorithms. Discrete Appl. Math. 156 (2008) 397–405
12. Eppstein, D.: The Traveling Salesman Problem for Cubic Graphs. In: Algorithms and Data Structures. Volume 2748 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2003) 307–318
13. Iwama, K., Nakashima, T.: An Improved Exact Algorithm for Cubic Graph TSP. In: Computing and Combinatorics. Volume 4598 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2007) 108–117
14. Gebauer, H.: Finding and Enumerating Hamilton Cycles in 4-Regular Graphs. Theoretical Computer Science 412(35) (2011) 4579–4591
15. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: The Travelling Salesman Problem in Bounded Degree Graphs. In: Automata, Languages and Programming. Volume 5125 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2008) 198–209
16. Sahni, S., Gonzalez, T.: P-Complete Approximation Problems. J. ACM 23 (1976) 555–565
17. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)
18. Rosenkrantz, D.J., Stearns, R.E., Lewis, P.M.: An Analysis of Several Heuristics for the Traveling Salesman Problem. SIAM J. Comput. 6(3) (1977) 563–581
19. Christofides, N.: Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem. Technical Report 338, Graduate School of Industrial Administration, CMU (1976)
20. Karpinski, M., Lampis, M., Schmied, R.: New inapproximability bounds for tsp. CoRR abs/1303.6437 (2013)
21. Frieze, A.M., Galbiati, G., Maffioli, F.: On the Worst-Case Performance of Some Algorithms for the Asymmetric Traveling Salesman Problem. Networks 12(1) (1982) 23–39
22. Bläser, M.: A New Approximation Algorithm for the Asymmetric TSP with Triangle Inequality. In: Proceedings of the fourteenth

annual ACM-SIAM symposium on Discrete algorithms. SODA'03, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2003) 638–645
23. Kaplan, H., Lewenstein, M., Shafrir, N., Sviridenko, M.: Approximation Algorithms for Asymmetric TSP by Decomposing Directed Regular Multigraphs. J. ACM 52 (2005) 602–626
24. Feige, U., Singh, M.: Improved Approximation Ratios for Traveling Salesperson Tours and Paths in Directed Graphs. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. Volume 4627 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2007) 104–118
25. Asadpour, A., Goemans, M.X., Madry, A., Gharan, S.O., Saberi, A.: An $O(\log n/\log\log n)$-approximation Algorithm for the Asymmetric Traveling Salesman Problem. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms. SODA'10, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2010) 379–389
26. Gharan, S.O., Saberi, A., Singh, M.: A Randomized Rounding Approach to the Traveling Salesman Problem. In: FOCS. (2011) 550–559
27. Mömke, T., Svensson, O.: Approximating Graphic TSP by Matchings. In: Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. FOCS'11, Washington, DC, USA, IEEE Computer Society (2011) 560–569
28. Mucha, M.: Improved Analysis for Graphic TSP Approximation via Matchings. CoRR abs/1108.1130 (2011)
29. Arora, S.: Polynomial Time Approximation Schemes for Euclidean TSP and Other Geometric Problems. In: In Proceedings of the 37th IEEE Symposium on Foundations of Computer Science (FOCS'96). (1996) 2–11
30. Arora, S.: Nearly Linear Time Approximation Schemes for Euclidean TSP and Other Geometric Problems. In: In Proceedings of the 38th IEEE Symposium on Foundations of Computer Science (FOCS'97). (1997) 554–563
31. Mitchell, J.S.B.: Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric TSP, k-MST, and Related Problems. SIAM J. Comput. 28 (1999) 1298–1309
32. Bartal, Y., Gottlieb, L.A., Krauthgamer, R.: The Traveling Salesman Problem: Low-Dimensionality Implies a Polynomial Time Approximation Scheme. CoRR abs/1112.0699 (2011)
33. Papadimitriou, C.H., Yannakakis, M.: The Traveling Salesman Problem with Distances One and Two. Math. Oper. Res. 18 (1993) 1–11
34. Berman, P., Karpinski, M.: 8/7-approximation Algorithm for (1,2)-TSP. In: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm. SODA'06, New York, NY, USA, ACM (2006) 641–648
35. Boria, N., Bourgeois, N., Escoffier, B., Paschos, V.T.: Exponential Approximation Schemata for Some Network Design Problems. Cahier du LAMSADE 303, LAMSADE, Universite Paris-Dauphine (2011)

36. Ibarra, O.H., Kim, C.E.: Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. J. ACM 22 (1975) 463–468
37. Bax, E.T.: Recurrence-Based Reductions for Inclusion and Exclusion Algorithms Applied to #P Problems. (1996)