Improved Approximation Algorithms for Metric Maximum ATSP and Maximum 3-Cycle Cover Problems

Markus Bläser[1], L. Shankar Ram[1], and Maxim Sviridenko[2]

[1] Institut für Theoretische Informatik, ETH Zürich,
CH-8092 Zürich, Switzerland
{mblaeser, lshankar}@inf.ethz.ch
[2] IBM T.J. Watson Research Center
sviri@us.ibm.com

Abstract. We consider an APX-hard variant (Δ-Max-ATSP) and an APX-hard relaxation (Max-3-DCC) of the classical traveling salesman problem. Δ-Max-ATSP is the following problem: Given an edge-weighted complete loopless directed graph G such that the edge weights fulfill the triangle inequality, find a maximum weight Hamiltonian tour of G. We present a $\frac{31}{40}$-approximation algorithm for Δ-Max-ATSP with polynomial running time. Max-3-DCC is the following problem: Given an edge-weighted complete loopless directed graph, compute a spanning collection of node-disjoint cycles, each of length at least three, whose weight is maximum among all such collections. We present a $\frac{3}{4}$-approximation algorithm for this problem with polynomial running time. In both cases, we improve on the previous best approximation performances. The results are obtained via a new decomposition technique for the fractional solution of an LP formulation of Max-3-DCC.

1 Introduction

Travelling salesman problems have been studied for many decades. Classically, one deals with minimization variants, that is, one wants to compute a shortest (i.e., minimum weight) Hamiltonian tour. But also the corresponding maximization variants have been investigated. At a first glance, computing a tour of maximum weight seems to be unnatural, but this problems has its applications for instance in maximum latency delivery problems [5] or in the computation of shortest common superstrings [4].

1.1 Notations and Definitions

Let $G = (V, E)$ be a complete loopless directed graph and $w : E \to \mathbb{Q}_{\geq 0}$ be a weight function that assigns each edge a nonnegative weight. A cycle of G is a (strongly) connected subgraph such that each node has indegree and outdegree one. (Since G has no loops, every cycle has length at least two.) The weight $w(c)$

of a cycle c is the sum of weigths of the edges contained in it. A *Hamiltonian tour* of G is a cycle that contains all nodes of G. The problem of finding a Hamiltonian tour of minimum weight is the well-studied asymmetric traveling salesman problem (ATSP). The problem is called asymmetric, since G is directed. The special case that G is undirected or, equivalently, that w is symmetric has received even more attention. But also the maximization variant—given G, find a Hamiltonian tour of maximum weight—has been studied. This problem is for instance used for maximum latency delivery problems [5] and as a blackbox in shortest common superstring computations [4]. We here study the variant of Maximum ATSP where w in addition fulfills the triangle inequality, that is,

$$w(u,v) + w(v,x) \geq w(u,x) \qquad \text{for all nodes } u, v, x.$$

We call this problem Δ-Max-ATSP.

A *cycle cover* of G is a collection of node-disjoint cycles such that each node is part of exactly one cycle. Every Hamiltonian tour is obviously a cycle cover. We call a cycle a *k-cycle* if it has *exactly k* edges (and nodes). A cycle cover is a *k-cycle cover* if each cycle in the cover has *at least k* edges. We call the problem of finding a maximum weight 3-cycle cover Max-3-DCC. Note that we here do not require w to fulfill the triangle inequality.

1.2 Previous Results

For the general Maximum ATSP, Nemhauser, Fisher, and Wolsey [6] present a $\frac{1}{2}$-approximation algorithm with polynomial time. Kosaraju, Park, and Stein [8], Bläser [1], Levenstein and Sviridenko [10], and Kaplan et al. [7] improve on this by giving polynomial time approximation algorithm with performances $\frac{38}{63}$, $\frac{8}{13}$, $\frac{5}{8}$, and $\frac{2}{3}$, respectively. For Δ-Max-ATSP, Kostochka and Serdyukov [9] provide a $\frac{3}{4}$-approximation algorithm with polynomial running time. Kaplan et al. [7] improve on this by giving a polynomial time $\frac{10}{13}$-approximation algorithm.

Δ-Max-ATSP is APX-hard, even if the weight function is $\{1, 2\}$-valued. This follows basically from the hardness proof of the corresponding minization variant given by Papadimitriou and Yannakakis [12]

The problem of computing a maximum weight 2-cycle cover is solvable in polynomial time, see Section 2.1. But already the problem of computing maximum weight 3-cycle covers is APX-hard, even if w attains only two different values [3]. Bläser and Manthey [2] give a $\frac{3}{5}$-approximation algorithm with polynomial running time for Max-3-DCC. Kaplan et al. [7] improve on this by giving a $\frac{2}{3}$-approximation algorithm with polynomial running time.

1.3 New Results

As a main technical contribution, we present a new decomposition technique for the fractional solution of an LP for computing maximum weight 3-cycle covers. The new idea is to ignore the directions of the edges and decompose the fractional solution into undirected cycle covers, that means, that after ignoring directions, the subgraph is a cycle cover. This has of course the drawback that viewed as a directed graph, the edges of the cycles might not point into the same direction. The advantage, on the other hand is, that all cycles in the cycle covers obtained have length at least three. In previous approaches such a fractional solution always was decomposed into directed cycle covers in which every cycle could have length two (but one had some additional knowledge about the distribution of the 2-cycles in the covers.) We apply this method to Δ-Max-ATSP and Max-3-DCC.

For Δ-Max-ATSP, this results in a $\frac{31}{40}$-approximation algorithm improving on the previous best algorithm which has approximation performance $\frac{10}{13}$. Note that $\frac{31}{40} = 0.775$ and $\frac{10}{13} \approx 0.769$.

For Max-3-DCC, we get a $\frac{3}{4}$-approximation algorithm. This improves the previous best algorithm which has performance $\frac{2}{3}$.

2 Computing Undirected 3-Cycle Covers

Let G be a complete directed graph without loops with n nodes and let w be a weight function on the edges of G.

2.1 LP for 3-Cycle Covers

Maximum weight cycle covers can be computed by solving the following well-known LP:

$$\begin{aligned}
&\text{Maximize } \sum_{(u,v)} w(u,v)x_{u,v} \text{ subject to}\\
&\sum_{u\in V} x_{u,v} = 1 \text{ for all } v \in V, \quad \text{(indegree constraints)}\\
&\sum_{v\in V} x_{u,v} = 1 \text{ for all } u \in V, \quad \text{(outdegree constraints)}\\
&x_{u,v} \geq 0 \qquad \text{for all } (u,v).
\end{aligned} \tag{1}$$

The variable $x_{u,v}$ is the indicator variable of the edge (u,v). The matrix corresponding to (1) is totally unimodular (see e.g. [11]), thus any optimum basic solution of (1) is integer valued (indeed $\{0,1\}$ valued). When one wants to use cycle covers as a relaxation for approximating Hamiltonian tours, the worst case is a cycle cover that consists solely of cycles of length two, so-called *2-cycles*. To avoid this, one can add 2-cycle elimination constraints to the LP:

$$x_{u,v} + x_{v,u} \leq 1 \quad \text{for all } (u,v) \quad \text{(2-cycle elimination)} \tag{2}$$

These constraints are a subset of the so-called *subtour elimination constraints.*

If we consider the LP above as an integer LP, then the 2-cycle elimination constraints ensure that there are no 2-cycles in a feasible solution. However, after adding the 2-cycle elimination constraints, the basic feasible solutions of the relaxed LP may not be integral anymore.

2.2 Decomposition Into Directed 2-Cycle Covers

Let $(x^\star_{u,v})$ denote an optimal fractional solution of the relaxed LP (1) together with (2). Let $W^\star$ be its weight. Let N be the smallest common multiple of all denominators of the $x^\star_{u,v}$. From $(x^\star_{u,v})$, we build a directed multigraph $M^\star$: Each edge (u, v) in $M^\star$ has multiplicity $x^\star_{u,v} \cdot N$. By using standard scaling and rounding we may also assume that N is a power of two, i.e. $N = 2^\nu$ for some integer ν polynomially bounded in the input length (see the journal version of [7] for details).

We change the solution $(x^\star_{u,v})$ and corresponding multigraph $M^\star$ in the following way. Construct undirected graph H by defining an undirected edge $\{u, v\}$ if $M^\star$ contains edges between vertices u and v in both directions. If H contains a cycle C then $M^\star$ contains two corresponding oppositely oriented cycles C_1 and C_2 of length ≥ 3. The multiplicity of those cycles is $\min\{x^\star_{u,v} : (u, v) \in C_1\} \cdot N$ and $\min\{x^\star_{u,v} : (u, v) \in C_2\} \cdot N$. W.l.o.g assume that the weight of C_1 is no more than the weight of C_2. We delete $\min\{x^\star_{u,v} : (u, v) \in C_1\} \cdot N$ copies of C_1 from $M^\star$ and add $\min\{x^\star_{u,v} : (u, v) \in C_1\} \cdot N$ copies of C_2. We also change the current solution $(x^\star_{u,v})$ to reflect the change in $M^\star$. The new solution is also an optimal solution of the LP (1) together with (2) since we did not decrease the value of the solution and did not violate the LP constraints during the transformation. Repeating the procedure $O(n^2)$ times we could guarantee that we have an optimal solution $(x^\star_{u,v})$ such that graph $M^\star$ does not have oppositely oriented cycles of length larger than two.

Lewenstein and Sviridenko [10] showed how to compute a collection of cycle covers $C_1, \ldots, C_N$ from $M^\star$ with the following properties:

(P1) $M^\star$ is the union of $C_1, \ldots, C_N$, considered as a multigraph. Thus the total weight of $C_1, \ldots, C_N$ equals $N \cdot W^\star$.
(P2) Between any pair of nodes u and v, the total number of edges in $C_1, \ldots, C_N$ between u and v is at most N.

The number N might be exponential, however, Lewenstein and Sviridenko also gave a succinct representation consisting of at most n^2 cycle covers with appropriate multiplicities. This will be sufficient for our algorithms.

The discussion above implies that $C_1, \ldots, C_N$, fulfill the additional property:

(P3) Let H be the undirected graph that contains an edge $\{u, v\}$ iff u and v are contained in a 2-cycle in at least one of $C_1, \ldots, C_N$. Then H is acyclic.

2.3 Decomposition Into Undirected 3-Cycle Covers

For our algorithms, we now redistribute the edges of $C_1, \ldots, C_N$. Each copy of an edge (u, v) in $M^\star$ appears in exactly one of the $C_1, \ldots, C_N$. Color an edge of $M^\star$ red, if it occurs in a 2-cycle in the particular C_i. Otherwise color it blue. Note that by (P3), red edges cannot form a cycle of length strictly larger than two.

Lemma 1. *Let U be an undirected $2N$-regular multigraph that has at most N copies of any edge where $N = 2^\nu$ is a power of two. Then there are undirected*

3-cycle covers $D_1, \dots, D_N$ such that U is the union of $D_1, \dots, D_N$. This decomposition can be performed in polynomial time.

Proof. The proof is by induction on N: If $N = 1$, then U is a 3-cycle cover.

Assume that $N > 1$. We will now decompose U into two N-regular multigraph U_1 and U_2, each of them containing at most $N/2$ copies of each edge. By the induction hypothesis, these multigraphs can be decomposed into $N/2$ 3-cycle covers each. This proves the lemma.

Let e be an edge of U and let m be its multiplicity. We move $\lfloor m/2 \rfloor$ copies to U_1 and $\lfloor m/2 \rfloor$ to U_2. If m is even, this distributes all copies of e between U_1 and U_2. If m is odd, then one copy remains, that is, the multigraph U' that remains after treating all edges in this way is indeed a graph. Since U is $2N$-regular and we remove an even number of copies of each edge, the degree of each node in U' is even. Therefore, each connected component of U' is Eulerian. For each such component, we compute an Eulerian tour. We take the edges of each component of U' in the order induced by the Eulerian tour and move them in an alternating fashion to U_1 and U_2. In this way, the degree at each node in U' is "halved", therefore both U_1 and U_2 are N-regular.

It remains to show that every edge in U_1 and U_2, respectively, has multiplicity at most $N/2$. This is clearly true if the multiplicity m of an edge e in U was even. If m is odd, then $m < N$, since N is even. Thus U_1 and U_2 get $\lfloor m/2 \rfloor < N/2$ copies. The last copy is then either moved to U_1 or U_2, but in both cases, the multiplicity is thereafter $\leq N/2$. □

Let W_2 be the average weight of all 2-cycles and W_3 be the average weight of all cycles of length at least three in $C_1, \dots, C_N$.

We now consider $M^\star$ as an undirected graph. It is $2N$-regular and by (P2), there are at most N edges between any pair of nodes. By Lemma 1, we can decompose $M^\star$ into undirected 3-cycle covers $D_1, \dots, D_N$. As already mentioned, none of the cycles in $D_1, \dots, D_N$ solely consists of red edges. Now we view $D_1, \dots, D_n$ as directed graphs again. They may not be directed cycle covers anymore, since the cycles may not be directed cycles. For all i and for any red edge in D_i, we add the other edge of the corresponding 2-cycle to D_i. Let $\hat{D}_1, \dots, \hat{D}_N$ be the resulting graphs. The average weight of $\hat{D}_1, \dots, \hat{D}_N$ is

$$\frac{w(\hat{D}_1) + \dots + w(\hat{D}_N)}{N} = 2W_2 + W_3, \qquad (3)$$

because every edge in a 2-cycle was added a second time.

3 Metric Maximum ATSP

Throughout this section, we assume that w fulfills the triangle inequality, i.e.,

$$w(u, v) + w(v, x) \geq w(u, x) \qquad \text{for all nodes } u, v, x.$$

Our goal is to find a Hamiltonian tour of maximum weight.

3.1 First Algorithm

By exploiting an algorithm due to Kostochka and Serdyukov [9], Kaplan et al. [7] show how to compute a Hamiltonian tour of weight at least $\frac{3}{4}W_2 + \frac{5}{6}W_3$.

Theorem 1. *There is an algorithm with polynomial running time that given $C_1, \ldots, C_N$, computes a Hamiltonian tour of weight $\frac{3}{4}W_2 + \frac{5}{6}W_3$.* □

This will be the first algorithm that we use. It is favorable if W_2 is small. Next we design an algorithm that works well if W_2 is large.

3.2 Second Algorithm

Lemma 2. *Let K be a connected component of $\hat{D}_i$. After deleting one blue edge of K, we can construct in polynomial time two node-disjoint directed paths P_1 and P_2 such that*

1. *P_1 and P_2 span the same nodes as K,*
2. *P_1 can be transformed into P_2 by reversing all directions of its edges and vice versa,*
3. *except the discarded blue edge, all edges of K are in $P_1 \cup P_2$, and*
4. *the discarded edge connects the two end-points of P_1 and P_2, respectively.*

Proof. The component corresponding to K in D_i is an undirected cycle. At least one of the edges on this cycle is blue by (P3). Discard one blue edge. Let $v_1, \ldots, v_\ell$ be the nodes of K (in this order) and assume that the edge between v_ℓ and v_1 was discarded. Between any two nodes v_λ and $v_{\lambda+1}$, there are at most two edges and if there are two edges, then these edges are red and point into different directions, since we added the other edge of the 2-cycles. Therefore, the paths $v_1, v_2, \ldots, v_\ell$ and $v_\ell, v_{\ell-1}, \ldots, v_1$ contain all edges of K except the one that we discarded. □

By applying Lemma 2 to each component of $\hat{D}_i$, we obtain two collections of node-disjoint paths $P_{i,1}$ and $P_{i,2}$ such that each connected component of $\hat{D}_i$ corresponds to two oppositely directed paths in $P_{i,1}$ and $P_{i,2}$ respectively. Next we are going to construct Hamiltonian tours $H_{i,1}$ and $H_{i,2}$ out of $P_{i,1}$ and $P_{i,2}$.

Lemma 3. *Given $P_{i,1}$ and $P_{i,2}$, we can construct in polynomial time, two Hamiltonian tours $H_{i,1}$ and $H_{i,2}$ such that $H_{i,1}$ and $H_{i,2}$ contain all the weight of the red edges and $1/2$ of the weight of the blue edges of $\hat{D}_i$.*

Proof. Let $p_{j,1}, \ldots, p_{j,k}$ be the paths of $P_{i,j}$ for $j = 1, 2$. Assume that $p_{1,\kappa}$ and $p_{2,\kappa}$ span the same nodes but have opposite directions. Let $p_{1,\kappa}$ be the path that forms a cycle with the discarded blue edge.

We first describe a randomized algorithm. We first select paths $q_1, \ldots, q_k$: q_κ is $p_{1,\kappa}$ or $p_{2,\kappa}$, both with probability $1/2$. All coin flips are independent. The cycle $H_{i,1}$ is obtained by patching the paths together in the order $q_1, \ldots, q_k$, and the cycle $H_{i,2}$ by patching the paths together in the opposite order $q_k, \ldots, q_1$.

With the exception of the discarded blue edges in Lemma 2, an edge of $\hat{D}_i$ is included twice with probability 1/2, namely once in $H_{i,1}$ and once in $H_{i,2}$. Thus we get all the weight of these edges in expectation.

We now show that we can get some weight of the discarded blue edges back (in expectation) during the patching process. Figure 3.2 shows two discarded blue edges e and f. There are four possibilities how the corresponding paths can be directed. Each occurs with probability 1/4. The edges introduced by the patching are x_j and y_j, respectively. The expected weight we get is

$$\tfrac{1}{4}(w(x_1) + w(y_1) + w(x_2) + w(y_2) + w(x_3) + w(y_3) + w(x_4) + w(y_4)).$$

By the triangle inequality, we have

$$\begin{aligned} w(e) &\leq w(x_1) + w(y_2), \\ w(e) &\leq w(x_2) + w(y_1), \\ w(f) &\leq w(x_3) + w(y_1), \\ w(f) &\leq w(x_1) + w(y_3). \end{aligned}$$

Thus we recover $\frac{1}{4}(w(e) + w(f))$ in expectation.[1] But we will recover $\frac{1}{4}w(e)$ on the lefthand side of the path of e and $\frac{1}{4}w(f)$ on the righthand side of the path of f. Thus the total weight is 1/2 of the weight of the discarded blue edges.

The above randomized procedure can be easily derandomized by exploiting the method of conditional expectations. □

Theorem 2. *There is an algorithm with polynomial running time that given $C_1, \dots, C_N$, computes a Hamiltonian tour of weight at least $W_2 + \frac{1}{4}W_3$.*

Proof. The algorithm computes the graphs $\hat{D}_1, \dots, \hat{D}_N$ and decomposes them into $2N$ collections of paths $P_{1,1}, P_{1,2}, \dots, P_{N,1}, P_{N,2}$ as in Lemma 2. From each pair $P_{i,1}, P_{i,2}$, it computes Hamiltonian tours $H_{i,1}$ and $H_{i,2}$ as in Lemma 3. It then outputs the tour with the largest weight. By (3), $\hat{D}_1, \dots, \hat{D}_N$ have weight $N \cdot (2W_2 + W_3)$. When constructing the collections of paths, we might loose up to weight $N \cdot W_3$. But when forming the tours, we get half of it back by Lemma 3. Altogether, there are $2N$ Hamiltonian tours. The heaviest of them has weight at least $W_2 + \frac{1}{4}W_3$. □

3.3 Final Algorithm

Theorem 3. *There is a $\frac{31}{40}$-approximation algorithm for Δ-Max-ATSP with polynomial running time.*

Proof. The algorithm runs the algorithms of Theorems 1 and 2 and outputs the heavier tour. Balancing the approximation performances of both algorithms

[1] Note that we need $w(x_1)$ and $w(y_1)$ twice, but $w(x_4)$ and $w(y_4)$ is not of any use for us.

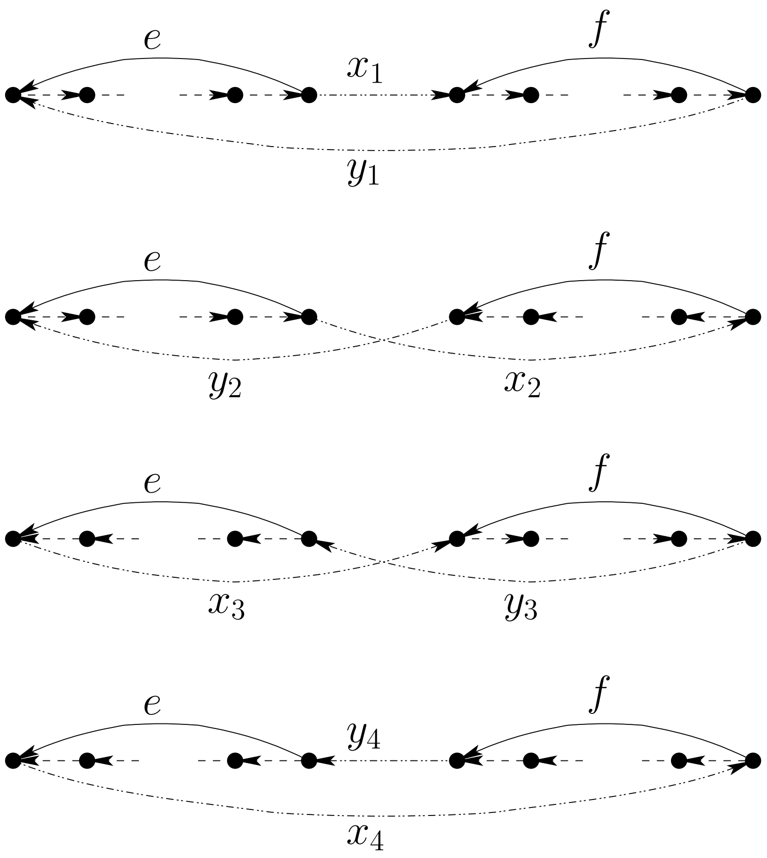

Fig. 1. Two discarded blue edges e and f (drawn solid) of two consecutive path (drawn dashed). There are four possibilities how the paths can be chosen. x_j and y_j are the edges used for the patching, the x_j are used when patching from left to right, the y_j when patching from right to left

yields the desired result. This is easiliy seen by the following probabilistic argument: Choose the output of the first algorithm with probability 9/10 and the output of the second one with probability 1/10. An easy calculation shows that the expected weight is $\frac{31}{40}(W_2 + W_3)$. The weight of the heavier tour is certainly at least as large as the expected weight. $\square$

4 Maximum 3-Cycle Cover

In this section, we only assume that w is nonnegative. In particular, w is *not* required to fulfill the triangle inequality. Our goal is to compute a directed 3-cycle cover of maximum weight.

4.1 First Algorithm

Bläser and Manthey [2] show how to compute a 3-cycle cover of weight $\frac{1}{2}W_2+W_3$.

Theorem 4. *There is an algorithm with polynomial running time that given $C_1,\ldots,C_N$, computes a 3-cycle cover of weight $\frac{1}{2}W_2+W_3$.* □

This will be our first algorithm. In the next subsection, we design an algorithm that is favorable if W_2 is large.

4.2 Second Algorithm

Lemma 4. *Let K be a connected component of $\hat{D}_i$. We can construct in polynomial time two node-disjoint directed cycles Z_1 and Z_2 such that*

1. *Z_1 and Z_2 span the same nodes as K,*
2. *Z_1 can be transformed into Z_2 by reversing all directions of its edges and vice versa,*
3. *all edges of K are in $Z_1 \cup Z_2$,*
4. *and the length of Z_1 and Z_2 is at least three.*

Proof. The component corresponding to K in D_i is an undirected cycle of length at least three. After possibly adding some edges to K, K consists of two oppositely oriented directed cycles of length at least three. □

Theorem 5. *There is an algorithm with polynomial running time that given $C_1,\ldots,C_N$, computes a 3-cycle cover of weight $W_2+\frac{1}{2}W_3$.*

Proof. The algorithm computes the graphs $\hat{D}_1,\ldots,\hat{D}_N$ and decomposes them into $2N$ collections of 3-cycle covers by treating each component as in Lemma 4. It then outputs the 3-cycle cover with the largest weight. By (3), $\hat{D}_1,\ldots,\hat{D}_N$ have weight $N\cdot(2W_2+W_3)$. There are $2N$ 3-cycle covers. The heaviest of them has weight at least $W_2+\frac{1}{2}W_3$. □

4.3 Final Algorithm

Theorem 6. *There is a $\frac{3}{4}$-approximation algorithm for Max-3-DCC with polynomial running time.*

Proof. The algorithm runs the algorithms of Theorems 4 and 5 and outputs the heavier 3-cycle cover. Balancing the approximation performances of both algorithms yields the desired result. □

References

1. Markus Bläser. An $\frac{8}{13}$-approximation algorithm for the asymmetric maximum tsp. *J. Algorithms*, 50(1):23–48, 2004.
2. Markus Bläser and Bodo Manthey. Two approximation algorithms for 3-cycle covers. In *Proc. 5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, volume 2462 of *Lecture Notes in Comput. Sci.*, pages 40–50, 2002.

3. Markus Bläser and Bodo Manthey. Approximating maximum weight cycle covers in directed graphs with edge weights zero and one. *Algorithmica*, 2005.
4. Dany Breslauer, Tao Jiang, and Zhigen Jiang. Rotations of periodic strings and short superstrings. *J. Algorithms*, 24:340–353, 1997.
5. P. Chalasani and R. Motwani. Approximating capacitated routing and delivery problems. *SIAM J. Comput*, 28:2133–2149, 1999.
6. M. L. Fisher, L. Nemhauser, and L. A. Wolsey. An analysis of approximations for finding a maximum weight Hamiltonian circuit. *Networks*, 12(1):799–809, 1979.
7. H. Kaplan, M. Lewenstein, N. Shafrir, and M. Sviridenko. Approximation algorithms for asymmetric tsp by decomposing directed regular multigraphs. In *Proc. 44th Ann. IEEE Symp. on Foundations of Comput. Sci. (FOCS)*, pages 56–65, 2003.
8. S. Rao Kosaraju, James K. Park, and Clifford Stein. Long tours and short superstrings. In *Proc. 35th Ann. IEEE Symp. on Foundations of Comput. Sci. (FOCS)*, 1994.
9. A. V. Kostochka and A. I. Serdyukov. Polynomial algorithms with the estimates $\frac{3}{4}$ and $\frac{5}{6}$ for the traveling salesman problem of the maximum. *Upravlyaemye Sistemy*, 26:55–59, 1985. (in Russian).
10. Moshe Lewenstein and Maxim Sviridenko. A 5/8 approximation algorithm for the maximum asymmetric TSP. *SIAM J. Disc. Math.*, 17(2):237–248, 2003.
11. C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
12. C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Math. Operations Research*, 18:1–11, 1993.