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

Markus Bläser [1]

Computer Science, Saarland University

L. Shankar Ram

Institut für Theoretische Informatik, ETH Zürich

Maxim Sviridenko

IBM T.J. Watson Research Center

Abstract

We consider an APX-hard variant (Δ-Max-ATSP) and an APX-hard relaxation (Max-3-DCC) of the classical traveling salesman problem. We present a $\frac{31}{40}$-approximation algorithm for Δ-Max-ATSP and a $\frac{3}{4}$-approximation algorithm for Max-3-DCC with polynomial running time. The results are obtained via a new way of applying techniques for computing undirected cycle covers to directed problems.

Key words: Approximation algorithm, Traveling Salesman Problem, cycle cover, blossom inequalities.

1 Introduction

The class of traveling salesman problems (TSP) has been studied for many decades. The usual setting for the problems is, informally, the following: given a

⋆ A preliminary version of the paper appeared in the proceedings of WADS 2005, LNCS 3608, pp. 350–359. We correct an error that appears in the proceedings version.

[1] Corresponding author. Address: Computer Science, Saarland University, Postfach 151150, D-66041 Saarbrücken, Germany. Email: mblaeser@cs.uni-saarland.de

complete graph with a weight function on the edges, find an optimal (minimum or maximum) weight tour visiting all the vertices exactly once. There are several variations possible in the above description giving rise to different problems. Usually, one deals with minimization variants, that is, one wants to compute a minimum weight tour. Still, the corresponding maximization variants have also been investigated. At a first glance, computing a tour of maximum weight seems to be unnatural, but problems with such an objective function find their applications, for instance, in maximum latency delivery problems [7] or in the computation of shortest common superstrings [6]. In this paper, we study two maximization problems—one of them being a TSP problem and the other, a related cycle cover problem.

More formally, let $G = (V, E)$ be a complete loopless directed graph and $w : E \to \mathbb{Q}_{\geq 0}$ be a weight function that assigns to 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 the weights 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). It is known that most of the TSP minimization problems are NPO–hard (see [22] for further details). This result urged researchers to study those problems in which the weight function w is restricted. A natural restriction is the triangle inequality, that is,

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

With this additional assumption, several approximation algorithms [12,2,15,10] have been designed for ATSP achieving performance guarantees of the order of $\log_2(|V|)$. The special case that G is undirected or, equivalently, that w is symmetric has received even more attention. In this case, a 1.5-factor approximation algorithm exists due to Christofides [9].

The maximization variant of TSP—given G, find a Hamiltonian tour of maximum weight—has also been studied. Note that constant factor approximation algorithms exist for the problem even without any restrictions on the weight function w. However, here we study the variant of Maximum ATSP where w fulfills the triangle inequality. We denote this problem Δ-Max-ATSP. For a survey of results on Maximum TSP, the reader is referred to [1].

A *cycle cover* of G is a collection of node-disjoint cycles such that each node is part of exactly one cycle. A cycle cover in an undirected graph is also called 2-factor. 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 called 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.

1.1 Previous Results

For the general Maximum ATSP, Nemhauser, Fisher, and Wolsey [11] present a $\frac{1}{2}$-approximation algorithm with polynomial time. Kosaraju, Park, and Stein [16], Bläser [3], Lewenstein and Sviridenko [19], and Kaplan et al. [15] 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 [17] provide a $\frac{3}{4}$-approximation algorithm with polynomial running time. Kaplan et al. [15] improve on this by giving a polynomial time $\frac{10}{13}$-approximation algorithm. Recently, Chen and Nagoya [8] gave a $\frac{27}{35}$-approximation algorithm for Δ-Max-ATSP.

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 minimization variant given by Papadimitriou and Yannakakis [20].

The problem of computing a maximum weight 2-cycle cover is solvable in polynomial time, as will be discussed in the section describing our decomposition technique. But, the problem of computing maximum weight 3-cycle covers is APX-hard, even if w attains only two different values [5]. Bläser and Manthey [4] give a $\frac{3}{5}$-approximation algorithm with polynomial running time for Max-3-DCC. Kaplan et al. [15] improve on this by giving a $\frac{2}{3}$-approximation algorithm with polynomial running time.

1.2 Our Results

Our main technial contribution ist the use of undirected techniques, namely the so-called blossom inequalities, for computing directed cycle covers. The new idea is to formulate a linear program which includes 2-cycles eliminations constraints and the blossom inequalities, solve it, and decompose the optimum fractional solution into an undirected cycle cover. This means, that after ignoring directions, the subgraph is a cycle cover. It has of course the drawback that when viewed as a directed graph, the edges of the cycles might not point into the direction along a directed cycle. The advantage, on the other hand is, that all cycles in the obtained cycle cover 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 use this approach to design better approximation algorithms for 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 due to [8] which has an approximation performance $\frac{27}{35}$. Note that $\frac{31}{40} = 0.775$ and $\frac{27}{35} \approx 0.771$.

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

2 The Decomposition Technique

Let $G = (V, E)$ be the given complete loopless directed graph with weight function $w : E \to \mathbb{Q}_{\geq 0}$ on the edges. The following integer program computes a maximum weight cycle cover in G:

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

We relax the integrality constraints to $y_{u,v} \geq 0$. It is well-known that this relaxed linear program has always an integral solution which can be found in polynomial time, see e.g. [21, Section 18.3]. In order to find a good approximation for traveling salesman problem, we want to have cycle covers with long cycles. A first step is to add the so-called *2-cycle elimination constraints*:

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

If these constraints are added to the linear program relaxation of (1), then it does not always have an integral optimum solution.

In undirected graphs, however, we can find optimum cycle cover that contain no 2-cycles, so called *2-factors*. 2-factors can be computed by the *blossom inequalities*:

$$
\begin{aligned}
x_{\{u,v\}} &= y_{u,v} + y_{v,u} && \text{for all } \{u,v\}. \\
x(\delta(W)\backslash F) - x(F) &\geq 1 - |F| && \text{for all } W \subseteq V,\ F \text{ matching}, \\
& && F \subseteq \delta(W),\ |F| \text{ odd}
\end{aligned} \tag{3}
$$

In the formulation above, $x_{\{u,v\}}$ are the indicator variables when we disregard the directions of the edges; $\delta(W)$ denotes the set of edges whose one endpoint lies in W and for any set A of unordered pairs of nodes, $x(A) := \sum_{e \in A} x_e$. There are exponentially many blossom inequalities, but Letchford et al. [18] construct a separation oracle for the blossom inequalities with running time $O(|V|^4)$.

We now solve the relaxed linear program (1) together with the constraints (2) and (3). We can find an optimum solution if there is a polynomial time separation oracle, that is, an algorithm that determines which inequalities are violated by a given assignment to the variables, see Grötschel et al. [13]. Of course, if the number of violated inequalities is exponential then one will not be able to find all of them in polynomial time, but finding one is sufficient. For the blossom inequalities, we use the oracle by Letchford et al. All other inequalities can be checked trivially, since there are only polynomially many. Thus an optimal fractional solution $z^\star := (y^\star_{u,v}, x^\star_{\{u,v\}})$ can be found in polynomial time.

Let L be the largest common denominator of all $y^\star_{u,v}, x^\star_{\{u,v\}}$, $u, v \in V$. We build a directed multigraph $M^\star$ as follows: For all $u, v \in V$, we add $y_{u,v} \cdot L$ many copies of the edge (u, v) to $M^\star$. The number L can be exponentially large in the input size, but its bit size is polynomial. We do not add $y_{u,v} \cdot L$ copies of an edge explicitly but use a counter to store the number of copies. Next, we normalize the solution $z^\star$ and the multigraph $M^\star$ as follows: Construct an 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 at least three. The multiplicity of those cycles is $\min\{y^\star_{u,v} : (u, v) \in c_1\} \cdot L$ and $\min\{y^\star_{u,v} : (u, v) \in c_2\} \cdot L$. We can assume that the weight of c_1 is not larger than the weight of c_2. We delete $\min\{y^\star_{u,v} : (u, v) \in c_1\} \cdot L$ copies of c_1 from $M^\star$ and add $\min\{y^\star_{u,v} : (u, v) \in c_1\} \cdot L$ copies of c_2. We also change the current solution $(y^\star_{u,v}, x^\star_{\{u,v\}})$ to reflect the change in $M^\star$. The new solution is also an optimal solution of the linear program since we did not decrease the value of the solution and we do not violate the constraints (1) and (2). This transformation does not change the values of $x^\star_{\{u,v\}}$ at all, and therefore the constraints (3) are still fulfilled, too. Repeating the procedure $O(|V|^2)$ times, we can guarantee that we have an optimal solution $(y^\star_{u,v}, x^\star_{\{u,v\}})$ such that

graph $M^\star$ does not have oppositely oriented cycles of length larger than two.

Since $z^\star$ fulfills the constraints (1), the adjacency matrix of $M^\star$ scaled down by $1/L$ is doubly stochastic, that is, its row and columns sums are all 1. By the Birkhoff–von Neumann theorem (see e.g. [21, Cor. 18.1a]), we can write this matrix as a convex combination of at most n^2 permutation matrices. Equivalently, we can write $M^\star$ as the union of at most n^2 cycle covers, each having a multiplicity. We can compute such cycle covers $C_1, \ldots, C_N$ and multiplicities $\gamma_1, \ldots, \gamma_N$ in polynomial time. Note that $L = \gamma_1 + \cdots + \gamma_N$. We color (a particular copy of) an edge e in $M^\star$ red, if it is contained in a 2-cycle in the unique cycle cover C_i that e belongs to. All other edges are colored blue. Let W_2 be the sum of all red edges divided by L and W_3 be the sum of all blue edges divided by L.

From $M^\star$, we get an undirected multigraph graph $M_u^\star$ by replacing every edge by an undirected one. We still keep all the copies, in particular, a 2-cycle is replaced by two undirected parallel edges. Since $x^\star_{\{u,v\}}$ fulfills the blossom inequalities and hence lies in the 2-factor polytope (refer to [21, p. 530]), $M_u^\star$ is the sum of a number of 2-factors. For the moment, we give every copy of the red edge $\{u, v\}$ the weight $w(u, v) + w(v, u)$. The total weight of all edges under this new weight function is $2W_2 \cdot L + W_3 \cdot L$, since we double the weight of the red edges. We compute a maximum weight 2-factor F_u in $M_u^\star$. This can be done in polynomial time; it is sufficient to consider only one copy between two nodes of $M_u^\star$ during this computation. The weight of F_u is at least $2W_2 + W_3$ under the new weight function, because $M_u^\star$ is the sum of L 2-factors of total weight $2W_2 \cdot L + W_3 \cdot L$. Let F be the directed graph that we get from F_u be taking the directions into account again. Next, for every red edge (u, v), we add the corresponding edge (v, u). Let D be the resulting directed graph. D has weight at least $2W_2 + W_3$ under the old weight function, since for each red edge we added exactly the weight that is lost when going back to the old weight function. Note that D is a simple graph. The weakly connected components of D are cycles, but not all edges point in the same direction and if the edge is red, also the other edge between the two nodes is present. Every weakly connected component of D contains at least one blue edge. (If D had a cycle without a blue edge, then $M^\star$ would have two cycles of length at least three that are oppositely directed, a contradiction.)

We now apply our technique to the two problems Δ-Max-ATSP and Max-3-DCC. For each of them, we design two algorithms: One uses $C_1, \ldots, C_N$ and works well if W_2 is small, the other one uses D is favourable if W_2 is large.

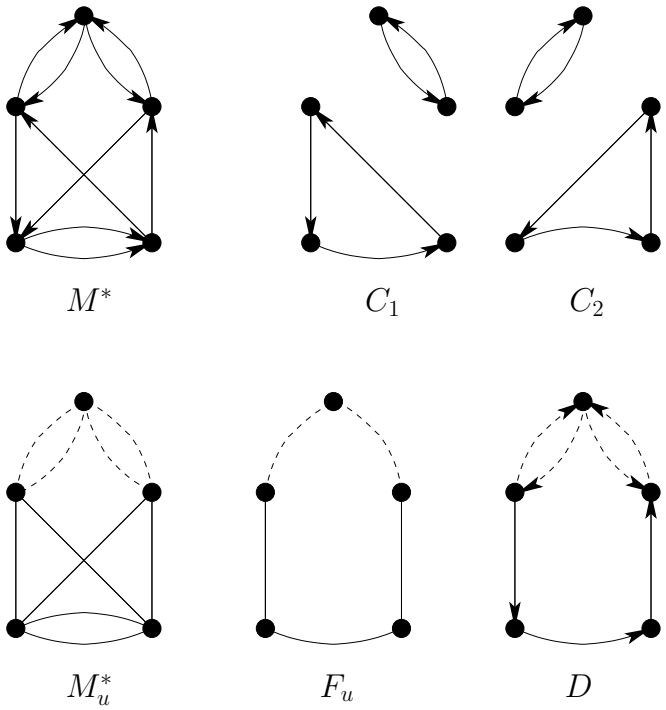

Fig. 1. An example. The leftmost graph in the upper line is the graph $M^\star$. Here, $L = 2$. Next to it, we see the decomposition of $M^\star$ into cycle covers. Both cycle covers have the multiplicity 1 in this example. The four edges incident with the node on the top will be colored red, since they appear in 2-cycles in C_1 and C_2, respectively. All other edges are colored blue. Red edges will be drawn dashed in the following. In the second row, we see the graph $M_u^\star$ that we get from $M^\star$ by disregarding directions. Next to it, we see the 2-factor F_u. The rightmost graph is D. Directions are re-introduced and for the two red edges, the corresponding edge with opposite direction is added.

3 Metric Maximum ATSP

Throughout this section, we assume that w fulfills the triangle inequality. Our goal is to find a Hamiltonian tour of maximum weight.

By exploiting an algorithm due to Kostochka and Serdyukov [17], Kaplan et al. [15] show how to compute a Hamiltonian tour of weight at least $\frac{3}{4}W_2+\frac{5}{6}W_3$.

Lemma 1 *There is an algorithm with polynomial running time that computes a Hamiltonian tour of weight at least* $\frac{3}{4}W_2 + \frac{5}{6}W_3$. □

This will be the first algorithm that we use. The second algorithm needs the following lemma.

Lemma 2 *Let K be a weakly connected component of D. 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,
(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 the underlying undirected graph is a cycle of length at least three. At least one of the edges on this cycle is blue. 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 D, we obtain two collections of node-disjoint paths P_1 and P_2 such that each weakly connected component of D corresponds to two oppositely directed paths in P_1 and P_2 respectively. Next we are going to construct Hamiltonian tours H_1 and H_2 out of P_1 and P_2. The proof of the following lemma uses an idea of Hassin and Rubinstein [14].

Lemma 3 *We can construct two Hamiltonian tours H_1 and H_2 in polynomial time such that H_1 and H_2 contain all the weight of the red edges of D and $1/2$ of the weight of the blue edges of D.*

PROOF. Let $p_{j,1}, \ldots, p_{j,k}$ be the paths of P_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 and estimate the expected weight of the two tours H_1 and H_2. We select paths $q_1, \ldots, q_k$: q_κ is $p_{1,\kappa}$ or $p_{2,\kappa}$, both with probability $1/2$, $1 \leq \kappa \leq k$. All coin flips are independent. The cycle H_1 is obtained by patching the paths together in the order $q_1, \ldots, q_k$, and the cycle H_2 by patching the paths together in the opposite order $q_k, \ldots, q_1$.

There are two "types" of edges in H_1 and H_2. Edges that already appear in P_1 or P_2 and edges that are introduced during the patching process. Edges of the first type correspond to red edges of D or blue edges of D that are not discarded when building P_1 and P_2.

Every edge of P_1 and P_2 is included with probability $1/2$ and if it is included, then it is included twice, namely once in H_1 and once in H_2. Thus every edge e of P_1 and P_2 contributes weight $w(e)$ to H_1 and H_2 in expectation. Thus the

expected weight of the edges of the first type in H_1 and H_2 is $w(P_1) + w(P_2)$ by linearity of expectation.

The discarded blue edges are of course not included in H_1 or H_2, but we can bound the weight of the edges of the second type in terms of the weight of the discarded blue edges. We do this analysis locally, i.e., we estimate the expected weight of the edges of the second type between to consecutive paths q_κ and $q_{\kappa+1}$. Thereafter, we exploit the linearity of expectation.

Figure 3 shows two discarded blue edges e and f corresponding to paths q_κ and $q_{\kappa+1}$. The direction of q_κ is determined randomly with probability $1/2$. The same is true for $q_{\kappa+1}$. Thus, there are four possibilities how q_κ and $q_{\kappa+1}$ can be directed. Since the coin flips are independent, each occurs with probability $1/4$. The edges introduced by the patching are x_j and y_j, $j = 1, \ldots, 4$ respectively. Each pair belongs to one of the four possiblities above. The edge x_j is introduced when patching in the order $q_1, \ldots, q_\ell$, i.e, when building H_1. The edge y_j is introduced, when building H_2. The expected weight that we get from the edges of the second type introduced between q_κ and $q_{\kappa+1}$ is

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

By the triangle inequality, we have

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

It follows that

$$z \geq \tfrac{1}{4}(w(e) + w(f)).$$

By the same argument, we get another $\frac{1}{4}w(e)$ from the edges of the second type introduced between $q_{\kappa-1}$ and q_κ. And from the edges of the second type introduced between $q_{\kappa+1}$ and $q_{\kappa+2}$, we get another $\frac{1}{4}w(f)$. Thus the total expected weight of the edges of the second type 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. □

Lemma 4 *There is an algorithm with polynomial running time that computes a Hamiltonian tour of weight at least* $W_2 + \frac{1}{4}W_3$.

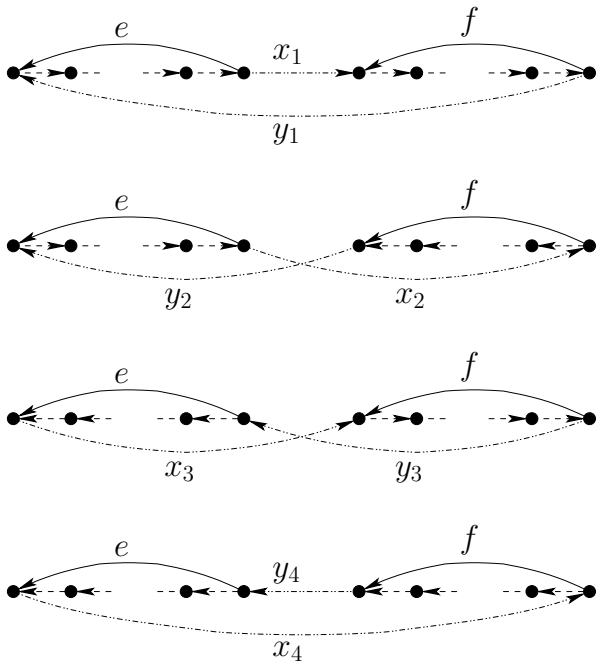

Fig. 2. Two discarded blue edges e and f (drawn solid) of two consecutive path (drawn dashed). There are four possibilities how the paths can be chosen. x_j and y_j are the edges used for the patching, the x_j are used when patching from left to right, the y_j when patching from right to left.

PROOF. The algorithm decomposes the graph D into two collection of paths P_1, P_2 as in Lemma 2. From P_1, P_2, it computes Hamiltonian tours H_1 and H_2 as in Lemma 3. It then outputs the tour with the largest weight. D has weight at least $2W_2 + W_3$. Let a be the weight of the deleted blue edges. Then $w(P_1) + w(P_2) \geq 2W_2 + W_3 - a$. The total weight of H_1 and H_2 is at least

$$w(P_1) + w(P_2) + a/2 \geq 2W_2 + W_3 - a/2 \geq 2W_2 + W_3/2,$$

since $a \leq W_3$. Thus the heavier one of H_1 and H_2 has weight at least $W_2 + W_3/4$.

Finally, we run the algorithms of Lemmas 1 and 4 and output the heavier tour.

Theorem 5 *There is a $\frac{31}{40}$-approximation algorithm for Δ-Max-ATSP with polynomial running time.*

PROOF. The first algorithms returns a Hamiltonian tour of weight at least $\frac{3}{4}W_2 + \frac{5}{6}W_3$. The second one returns a tour of weight at least $W_2 + \frac{1}{4}W_3$. Since

$$\tfrac{31}{40}(W_2 + W_3) = \tfrac{9}{10}(\tfrac{3}{4}W_2 + \tfrac{5}{6}W_3) + \tfrac{1}{10}(W_2 + \tfrac{1}{4}W_3)$$

is a convex combination of these, the heavier of the two tours has weight at least $\frac{31}{40}(W_2 + W_3)$. Since $W_2 + W_3$ is an upper bound for the weight of an optimum Hamiltonian tour, we are done.

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.

Bläser and Manthey [4] show how to compute a 3-cycle cover of weight $\frac{1}{2}W_2 + W_3$, from $C_1, \ldots, C_N$.

Lemma 6 *There is an algorithm with polynomial running time that computes a 3-cycle cover of weight at least $\frac{1}{2}W_2 + W_3$.*

This will be our first algorithm. Next, we design an algorithm that is favorable if W_2 is large. We start with a lemma similar to Lemma 2.

Lemma 7 *Let K be a weakly connected component of D. We can construct in polynomial time two 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,
(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 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. □

Lemma 8 *There is an algorithm with polynomial running time that computes a 3-cycle cover of weight at least $W_2 + \frac{1}{2}W_3$.*

PROOF. The algorithm computes the graph D and decomposes them into two 3-cycle covers by treating each component as in Lemma 7. It then outputs the 3-cycle cover with the largest weight. D has weight $2W_2 + W_3$. The heaviest of the two 3-cycle covers has weight at least $W_2 + \frac{1}{2}W_3$.

If we run the algorithms of Lemmas 6 and 8 and output the heavier tour, we get a 3/4 approximation algorithm.

Theorem 9 *There is a $\frac{3}{4}$-approximation algorithm for Max-3-DCC with polynomial running time.*

Acknowledgements

We thank A. Czumaj for pointing to an error in a previous version of the paper. The comments of a referee helped at lot to improve the presentation of the paper.

References

[1] A. Barvinok, E. Gimaldi and A. Serdyukov. The Maximum TSP. *The Traveling Salesman Problem and Its Variations*, 585–607, Comb. Optim. 12, Kluwer Acad. Publ., Dordrecht, 2002.

[2] M. Bläser. A New Approximation Algorithm for the Asymmetric TSP with Triangle Inequality. *Proc. of the 14th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA)*, 639–647, 2003.

[3] M. Bläser. An $\frac{8}{13}$-approximation Algorithm for the Asymmetric Maximum TSP. *J. Algorithms*, 50(1):23–48, 2004.

[4] M. Bläser and B. Manthey. Two Approximation Algorithms for 3-cycle Covers. *Proc. 5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, volume 2462 of *Lecture Notes in Comput. Sci.*, pages 40–50, 2002.

[5] M. Bläser and B. Manthey. Approximating Maximum Weight Cycle Covers in Directed Graphs with Edge Weights Zero and One. *Algorithmica*, 2005.

[6] D. Breslauer, T. Jiang, and Z. Jiang. Rotations of Periodic Strings and Short Superstrings. *J. Algorithms*, 24:340–353, 1997.

[7] P. Chalasani and R. Motwani. Approximating Capacitated Routing and Delivery Problems. *SIAM J. Comput*, 28:2133–2149, 1999.

[8] Z. Chen and T. Nagoya. Improved Approximation Algorithms for Metric Max TSP. *Proc. 13th Ann. European Symp. on Algorithms (ESA)*, pages 179–190, 2005.

[9] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *Tech. Report, GSIA, Carnegie Mellon University*, 1976.

[10] U. Feige and M. Singh. Improved Approximation Ratios for Traveling Salesperson Tours and Paths in Directed Graphs. *Proc. 10th Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2007.

[11] 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.

[12] A. Frieze, G. Galbiati and F. Maffioli. On the worst–case performance of some algorithms for the asymmetric travelling salesman problem. *Networks*, 12:23–39, 1982.

[13] M. Grötschel, L. Lovasz and A. Schrijver. Geometric Algorithms and Combinatorial Optimization. *Algorithms and Combinatorics: Study and Research Texts, 2*, Springer - Verlag, Berlin, 1988.

[14] R. Hassin and S. Rubinstein. A $\frac{7}{8}$-approximation Algorithm for Metric Max TSP. *Information Processing Letters*, 81:247–251, 2002.

[15] H. Kaplan, M. Lewenstein, N. Shafrir and M. Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *J. ACM*, 52(4):602–626, 2005.

[16] S. R. Kosaraju, J. K. Park, and C. Stein. Long Tours and Short Superstrings. *Proc. 35th Ann. IEEE Symp. on Foundations of Comput. Sci. (FOCS)*, 1994.

[17] 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).

[18] A. N. Letchford, G. Reinelt and D. O. Theis. A Faster Exact Separation Algorithm for Blossom Inequalities. *Proc. 10th Integer Programming and Combinatorial Optimization (IPCO)*, 2004.

[19] M. Lewenstein and M. Sviridenko. A 5/8-approximation Algorithm for the Maximum Asymmetric TSP. *SIAM J. Disc. Math.*, 17(2):237–248, 2003.

[20] C. H. Papadimitriou and M. Yannakakis. The Traveling Salesman Problem with Distances One and Two. *Math. Operations Research*, 18:1–11, 1993.

[21] A. Schrijver. Combinatorial Optimization - Polyhedra and Efficiency. *Springer-Verlag*, Berlin, 2003.

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