35/44-approximation for Asymmetric Maximum TSP with Triangle Inequality [Extended Abstract]

Łukasz Kowalik and Marcin Mucha*

Institute of Informatics, Warsaw University, Warsaw, Poland
{kowalik,mucha}@mimuw.edu.pl

Abstract. We describe a new approximation algorithm for the asymmetric maximum traveling salesman problem (ATSP) with triangle inequality. Our algorithm achieves approximation factor 35/44 which improves on the previous 31/40 factor of Bläser, Ram and Sviridenko [2].

1 Introduction

The Traveling Salesman Problem and its variants are among the most intensively researched problems in computer science and arise in a variety of applications. In its classical version, given a set of vertices V and a symmetric weight function $w : V^2 \to \mathbb{R}$ one has to find a Hamiltonian cycle of minimum weight. This problem is probably the most widely known example of an inapproximable NP-hard problem. However, there is a lot of research on approximation of several natural variants of TSP. These variants are still NP-hard, but allow approximation. One of the most important problems in this category is the maximization version (maxTSP for short), where w is assumed to have only nonnegative values (otherwise minTSP would reduce to it). There are several variants of maxTSP, e.g. the weight function can be symmetric or asymmetric, it can satisfy the triangle inequality or not, etc. (For some results on maxTSP variants see e.g. [3, 4, 6, 8]).

In this paper, we are concerned with the variant, where the weight function is asymmetric (in other words, the graph is directed) and satisfies the triangle inequality. This variant is often called *the semimetric maxTSP*.

The first approximation algorithm for this problem was proposed by Kostochka and Serdyukov [9] in 1985 and had approximation ratio of $\frac{3}{4}$. Quite recently, Kaplan, Lewenstein, Shafrir and Sviridenko [5] provided a very general and powerful framework for approximating asymmetric TSP variants and gave improved approximation ratios for 3 different problems: $\frac{4}{3}\log_3 n$ for semimetric minTSP, $\frac{10}{13}$ for semimetric maxTSP and $\frac{2}{3}$ for asymmetric maxTSP. Using a different approach, Bläser et. al obtained a $\frac{31}{40}$-approximation algorithm for semimetric maxTSP.

* Part of this work was done while both authors were staying at the Max Planck Institute in Saarbruecken, Germany. This research is partially supported by a grant from the Polish Ministry of Science and Higher Education, project N206 005 32/0807.

We show that in the case of semimetric maxTSP the ideas of Kaplan et al. can be combined with a new patching procedure yielding a $\frac{35}{44}$-approximation.

Overview of the paper The semimetric max-TSP approximation algorithm of Kaplan et al. combines two ideas: Kostochka and Serdyukov's "patching" algorithm for the same problem and a new framework based on pairs of cycle covers. In Section 2 we briefly review both ideas and the way they can be combined. In Section 3 we introduce a new patching procedure based on Kaplan et al.'s framework. This immediately leads to a relatively simple $\frac{11}{14}$-approximation for semimetric maxTSP. In Section 4 we describe a more elaborate patching method which improves the approximation ratio to $\frac{35}{44}$ by lowerbounding the weight of almost every edge used to form a Hamiltonian cycle.

2 Preliminaries

Throughout the remainder of this paper we assume all graphs to be directed and weighted with a nonnegative weight function w satisfying the triangle inequality.

2.1 Kostochka and Serdyukov's Algorithm

Many approximation algorithms for TSP problems begin with finding a minimum (maximum) cycle cover and then patch it to a Hamiltionian cycle. The following theorem shows how this is done in Kostochka and Serdyukov's algorithm.

Theorem 1. *Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a cycle cover in a directed weighted graph G with edge weights satisfying the triangle inequality. Let m_i be the number of edges in C_i and let $w_i = w(C_i)$ be the weight of C_i. Given the cycle cover $\mathcal{C}$, we can find in polynomial time a Hamiltonian cycle of weight $\sum_{i=2}^{k} \left(1 - \frac{1}{2m_i}\right) w_i$.*

A slightly weaker version of the above theorem is due to Kostochka and Serdyukov [7]. The version in this paper is taken from Kaplan et al. [5].

Maximum weight cycle cover (possibly containing 2-cycles) can be found in polynomial time. Such cover has weight at least as large as the maximum weight Hamiltonian cycle. From Theorem 1 it follows that

Theorem 2. *There exists a $\frac{3}{4}$-approximation algorithm for semimetric maxTSP.*

2.2 The Algorithm of Kaplan et al.

The 2-cycles are the obvious bottleneck of the above approach. If we could find, in polynomial time, a maximum weight cycle cover with no 2-cycles, we would get a 5/6-approximation algorithm. Unfortunately, finding such a cover is an NP-hard problem (see e.g. [1]). Kaplan et al. [5] proposed the following alternative approach.

Theorem 3. *Let $G = (V, E)$ be a directed weighted graph. We can find in polynomial time a pair of cycle covers $\mathcal{C}_1, \mathcal{C}_2$ such that (i) $\mathcal{C}_1$ and $\mathcal{C}_2$ share no 2-cycles, (ii) total weight $w(\mathcal{C}_1) + w(\mathcal{C}_2)$ of the two covers is at least $2OPT$, where OPT is the weight of the maximum weight Hamiltonian cycle in G.*

We will call such pairs of cycle covers *nice pairs of cycle covers.*

Observation 1 (Kaplan et al.) *In the above theorem, we can assume that the graph consisting of all the 2-cycles of $\mathcal{C}_1$ and $\mathcal{C}_2$ does not contain oppositely oriented cycles. For if it does contain such cycles, say C and its opposite $\hat{C}$, we can remove all the 2-cycles forming C and $\hat{C}$ from $\mathcal{C}_1$ and $\mathcal{C}_2$ and instead add C to $\mathcal{C}_1$ and $\hat{C}$ to $\mathcal{C}_2$.*

Theorem 4. *There exists a $\frac{10}{13}$-approximation algorithm for semimetric maxTSP.*

The proof of the above theorem can be found in [5]. Since our approach extends that of Kaplan et al., we include it here for completeness. Let us first introduce a few definitions. A *bipath* is a pair of oppositely oriented paths, i.e. a path and its opposite. As a special case, a *biedge* is a single edge together with its opposite edge. A *bicycle* is a pair of oppositely oriented cycles. Finally, a *Hamiltonian bicycle* is a pair of oppositely oriented Hamiltonian cycles.

Proof (of Theorem 4). Let $\mathcal{C}_1$, $\mathcal{C}_2$ be a nice pair of cycle covers. Applying Theorem 1 to $\mathcal{C}_1$ and $\mathcal{C}_2$, we get two Hamiltonian cycles H_1, H_2 with total weight $w(H_1) + w(H_2) \geq \frac{3}{4}W_2 + \frac{5}{6}W_{3+}$, where W_2 is the total weight of 2-cycles in $\mathcal{C}_1$ and $\mathcal{C}_2$ and W_{3+} is the total weight of all the other cycles.

Another way to construct a Hamiltonian cycle using $\mathcal{C}_1$ and $\mathcal{C}_2$ is to consider the graph H consisting of all the 2-cycles of $\mathcal{C}_1$ and $\mathcal{C}_2$. It follows from Observation 1 that H is a sum of disjoint bipaths. We can patch these bipaths arbitrarily to get a Hamiltonian bicycle $\hat{H}$ of weight $w(\hat{H}) \geq W_2$.

Picking the heaviest cycle out of H_1, H_2 and the two cycles of $\hat{H}$ gives a Hamiltonian cycle of weight at least $\frac{1}{2}\max\left\{\frac{3}{4}W_2 + \frac{5}{6}W_{3+}, W_2\right\}$. Since $W_2 + W_{3+} \geq 2\mathrm{OPT}$, easy calculation (or solving a corresponding linear program) shows that the weight of this heaviest cycle is at least $\frac{10}{13}\mathrm{OPT}$. □

3 Spanning Bitrees and 11/14-approximation

Kaplan et al.'s algorithm (see Theorem 4) balances two solutions. The first one is based on Kostochka and Serdyukov's algorithm and the second one on Kaplan et al.'s approach of constructing a nice pair of cycle covers. However, from these cycle covers they pick only the 2-cycles. The basic idea of our approach is to partially incorporate longer cycles into this second solution by constructing additional bipaths and/or extending existing ones.

Remark 1. Cycles of length > 2 do not contain pairs of opposite edges. Hence, not all the new bipath edges will belong to some cycle.

Let P be a family of disjoint bipaths. We say that set of biedges S is *allowed* w.r.t. P, if S is disjoint from P and the edge sum of P and S is a family of disjoint bipaths (e.g. adding S does not create a bicycle in P). A biedge e is *allowed* w.r.t P if $\{e\}$ is allowed w.r.t. P, otherwise e is *forbidden.*

The following is the skeleton of the algorithm, that we will develop in the remainder of the paper:

Algorithm 3.1 MAIN ALGORITHM

```
1: Let $\mathcal{C}_1$, $\mathcal{C}_2$ be a nice pair of cycle covers
2: Let $P$ be the family of bipaths constructed in Kaplan et al.'s Algorithm
3: Mark all 2-cycles as processed
4: for all unprocessed cycles $C$ in $\mathcal{C}_1$ and $\mathcal{C}_2$ do
5:     use $C$ to construct a heavy set $S$ of biedges, allowed w.r.t. $P$
6:     $P := P \cup S$
7:     mark $C$ as processed
8: arbitrarily patch $P$ to a Hamiltonian bicycle
```

Let the degree $\deg_P(v)$ of a vertex v in a family P of bipaths be the number of biedges in P incident with v (and not the number of edges). In the above algorithm S will always be chosen in such a way that the following is satisfied:

Invariant 1 *For any vertex v, $\deg_P(v)$ is not greater than the number of processed cycles containing v.*

How do we construct a heavy set of biedges S using a cycle C? In this section, S will contain only a single biedge e with both ends in C. When choosing $S = \{e\}$, we could pick e to be any of the biedges allowed w.r.t. P. However, we want e to have a large weight.

Let *bitree* be a connected set of biedges with no bicycles. Let C be a cycle and let the vertices of C be numbered $1, \ldots, k$ along the cycle. A bitree T is *plane* w.r.t. C if T does not contain two biedges u_1u_2, v_1v_2 such that $u_1 < v_1 < u_2 < v_2$ (intuitively, this means that if we make a planar drawing of C, we can complete it to a planar drawing of $C \cup T$). We say that T is a *plane spanning bitree* of C if T is plane w.r.t. C and connects all vertices of C. Plane spanning bitrees are interesting because they have large weight.[1]

Lemma 1. *Let T be a plane spanning bitree of a cycle C. Then $w(T) \geq w(C)$.*

Proof. The proof relies on the triangle inequality. The weight of every edge of C is upperbounded by the weight of a certain path in T. Figure 1 shows how this is done. The solid paths incident to a region marked with number i upperbound the weight of the cycle edge i. □

[1] All the plane spanning bitrees we use in this paper are in fact bipaths. We believe, however, that the more general setting might be beneficial in attempts to improve the results of this paper.

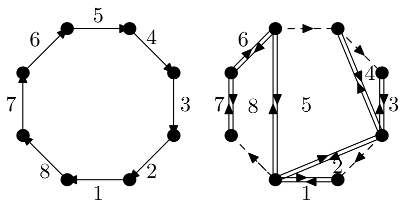

Fig. 1. The proof idea of Lemma 1

Observation 2 *Consider an execution of the Main Algorithm and let C be an unprocessed cycle. If P satisfies Invariant 1, then the set of biedges that have both endpoints in C and are forbidden w.r.t P forms a matching.*

Lemma 2. *Consider an execution of the Main Algorithm, let C be an unprocessed cycle, and let P satisfy Invariant 1. Then, there exists T, a plane spanning bitree w.r.t. C (in fact, a bipath), whose all biedges are allowed w.r.t P.*

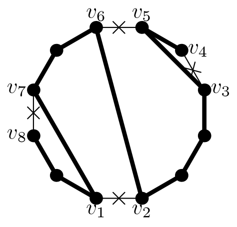

Fig. 2. Finding a plane bipath avoiding forbidden edges.

Proof. The path T is constructed as follows. First, for each edge (u, v) of cycle C put biedge uv in T whenever it is allowed. Note that at this point T already contains all vertices of C (because forbidden biedges with endvertices on C form a matching). Let k be the number of forbidden biedges corresponding to edges in $E(C)$. If $k = 0$ we remove any biedge from T and we are done. Otherwise enumerate the endvertices of the k biedges on C from v_1 to v_{2k} along the cycle C. Finally, for every $i = 1, \ldots, k-1$ add edge $v_i v_{2k-i}$ to T. (See Fig. 2). All these edges are allowed since their endvertices are endvertices of distinct forbidden edges and forbidden edges with ends on C form a matching. Also, T forms a path, since all its vertices are of degree 2 except for v_k and v_{2k}, which are of degree 1. Finally, path T is plane: the only edges that may cross are chords of C, however, for any pair of such distinct chords $v_i v_{2k-i}$, $v_j v_{2k-j}$ either $i < j < 2k-j < 2k-i$ or $j < i < 2k-i < 2k-j$. This proves the claim. □

Theorem 5. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be a nice pair of cycle covers of G. Then, there exists a Hamiltonian bicycle in G with weight at least $\sum_{i=2}^{\infty} \frac{W_k}{k-1}$, where W_k is the total weight of k-cycles in $\mathcal{C}_1$ and $\mathcal{C}_2$.*

Proof. We use the Main Algorithm. When processing a cycle C of length k, we use Lemma 2 to construct T, a plane spanning bitree w.r.t C, whose all biedges are allowed w.r.t. P. Then we set $S = \{e\}$, where e is the heaviest biedge of T. By Lemma 1 $w(e) \geq \frac{w(C)}{k-1}$, which proves the claim. □

Theorem 6. *There exists a $\frac{11}{14}$-approximation algorithm for semimetric maxTSP.*

Proof. As in the proof of Theorem 4 we construct a nice pair of cycle covers $\mathcal{C}_1$, $\mathcal{C}_2$ and use Theorem 1 to get Hamiltonian cycles H_1, H_2 with total weight

$$w(H_1) + w(H_2) \geq \sum_{i=2}^{\infty} \left(1 - \frac{1}{2k}\right) W_k.$$

Next, by Theorem 5 to get two more Hamiltonian cycles H_3, H_4 with total weight

$$w(H_3) + w(H_4) \geq \sum_{i=2}^{\infty} \frac{1}{k-1} W_k.$$

Picking the heaviest cycle out of all the H_i gives a Hamiltonian cycle H of weight

$$w(H) \geq \frac{1}{2} \max \left\{ \sum_{i=2}^{\infty} \left(1 - \frac{1}{2k}\right) W_k, \sum_{i=2}^{\infty} \frac{1}{k-1} W_k \right\}.$$

From $\sum_{i=2}^{\infty} W_k \geq 2\mathrm{OPT}$, it follows that $w(H) \geq \frac{11}{14}\mathrm{OPT}$. This can be proved by solving a corresponding LP (details omitted in this extended abstract). □

4 Making Ends Meet and 35/44-approximation

In this section we introduce two improvements. First, we will add more than one biedge to the family P of bipaths, while processing a single cycle C. This is possible if C is long enough. Moreover, recall that in the last step of the algorithm from the previous section we construct a Hamiltonian cycle by patching the bipaths with arbitrary edges. The endvertices of these edges could belong to distinct cycles and we do not lowerbound their weight in any way. The second improvement we are going to present here is to partially incorporate the patching process into the main algorithm in order to be able to lowerbound this weight. We use this approach for processing short cycles.

4.1 Long cycles

Lemma 3. *Let P be a family of disjoint bipaths satisfying Invariant 1 and let C be an unprocessed cycle of length at least 5. Then there exists an allowed family of biedges S, such that (i) after processing C, the family $P \cup S$ satisfies Invariant 1, (ii) $w(S) \geq \frac{1}{4}w(C)$, (iii) if $|C| \leq 7$ then $w(S) \geq \frac{1}{3}w(C)$, (iv) if $|C| = 5$ then $w(S) \geq \frac{1}{2}w(C)$.*

Proof. In order to keep Invariant 1 satisfied, we make S a set of vertex-disjoint allowed biedges with endvertices in C. Let Q be the plane bipath spanning C with no forbidden biedges, which exists by Lemma 2. We color the edges of Q with two colors: a and b, so that incident biedges get distinct colors. Adding all biedges of one color, say a, to P may create one or more bicycles (note that such a bicycle contains at least two biedges from Q). For each such bicycle we pick one biedge from Q and we recolor it to a new color c. Similarly, we recolor some biedges from b to d.

It is clear that each of the four color classes is an allowed family of biedges. Let S be the heaviest of these four sets. Clearly $w(S) \geq \frac{1}{4}w(Q)$. Since $w(Q) \geq w(C)$ by Lemma 1, we get (ii).

Now, let $|C| \leq 7$. Again, we find the bipath Q and we 2-color it. Suppose that adding all the biedges of color a to P gives a bicycle. Since there are at most 3 biedges colored a and any bicycle contains at least 2 such biedges, we can only get one such bicycle. Similarly, at most one bicycle is formed by P and biedges colored b. Suppose that both bicycles exist (the remaining cases are trivial). We need to recolor one (colored) biedge from each cycle to a new color, so that the recolored edges are not adjacent.

Let us start at one end of Q and go along Q until we encounter a colored cycle biedge. Assume w.l.o.g. that its color is a. Then, we can recolor both this biedge and the furthest cycle biedge colored b to a new color c. Clearly, each of the three color classes is an allowed family of biedges. Again, we let S be the heaviest of them, obtaining $w(S) \geq \frac{1}{3}w(C)$.

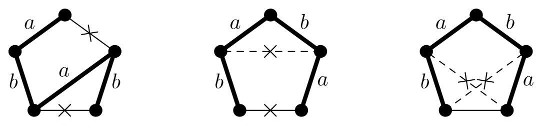

Fig. 3. Coloring a bipath spanning a 5-cycle. Crossed out edges are forbidden.

Finaly, consider the case of $|C| = 5$. W.l.o.g. we can assume that there are two forbidden biedges with endvertices on C (if not, we can just "forbid" additional biedges). Figure 3 shows all three possible configurations of these biedges together with our choice of the bipath Q in each case. As before, we 2-color Q, and then set S to be the heavier of the two color classes. This gives $w(S) \geq \frac{1}{2}w(C)$. Observe that in each case both color classes contain a biedge with an endvertex not adjacent to a forbidden biedge. Such a biedge cannot be a part of a bicycle in $P \cup S$, so S is allowed. □

4.2 Short cycles

To get the approximation ratio better than $\frac{11}{14}$ we need to extract more weight from the 3- and 4-cycles when constructing the bipaths in the Main Algorithm.

Unfortunately, it turns out that it is impossible to take more than one edge from *each* such cycle. Note however, that when only a single biedge is put into P when processing a cycle C, at least one vertex v of C becomes a *loose end*, i.e. $\deg_P(v)$ is smaller than the number of processed cycles containing v.

Remark 2. If $\deg_P(v) = 0$ and both cycles containing v have already been processed, we consider v to be *two* loose ends.

We can link loose ends from distinct cycles without violating Invariant 1. Surprisingly, it is possible to lowerbound the weight of such links. First let us see how loose ends are created.

Lemma 4. *Let P be a family of disjoint bipaths satisfying Invariant 1 and let C be an unprocessed k-cycle. Then there exists an allowed family of biedges S such that (i) $w(S) \geq \frac{1}{k-1}w(C)$, (ii) after processing C family $P \cup S$ satisfies Invariant 1, and (iii) the number of loose ends increases by $k-2$.*

Proof. We use the approach described in the previous section, i.e. $S = \{e\}$ where e is the heaviest biedge of the plane spanning bipath of C. All the vertices of C except for the two endvertices of e become loose ends. $\square$

The following two lemmas show how loose ends can be used to extract more weight from 3-cycles and 4-cycles.

Lemma 5. *Let P be a family of disjoint bipaths satisfying Invariant 1 with at least 2 loose ends and let C be an unprocessed 3-cycle. Then there exists an allowed family of biedges S such that (i) $w(S) \geq \frac{3}{4}w(C)$, (ii) after processing C, the family $P \cup S$ satisfies Invariant 1, and (iii) the number of loose ends decreases by 1.*

Proof. Our plan here is to make S contain one biedge with both endvertices in C and one biedge linking the remaining vertex of C with one of the loose ends. This obviously satisfies *(ii)* and *(iii)*. We only need to guarantee that S is allowed and that it has weight at least $\frac{3}{4}w(C)$. We consider one of the following two cases, depending on whether or not there exists a loose end v that is not connected to C with a bipath in P (this bipath might have length 0 in which case one of the vertices of C is a loose end).

Case 1. There exists such v. Let a, b, c be the vertices of C and suppose $Q = abc$ is a plane spanning bipath of C with no forbidden edges. Consider two possibilities for S: $S_1 = \{ab, cv\}$ (ab and cv denote biedges here) and $S_2 = \{bc, av\}$. Both are allowed. For example, if we add S_1 to P, cv lies on a bipath (not a bicycle) because v is not connected with C in P, and ab by itself cannot form a bicycle because it is allowed as a biedge of Q. Similar argument works for S_2. We also have

$$\begin{aligned} w(S_1) + w(S_2) = w(ab) + w(bc) + w(cv) + w(va) \geq w(ab) + w(bc) + w(ca) \geq \\ \geq \tfrac{1}{2}[(w(ab) + w(bc)) + (w(bc) + w(ca)) + (w(ca) + w(ab))] \geq \tfrac{3}{2}w(C), \end{aligned}$$

where the second inequality follows from the triangle inequality and the last inequality follows from Lemma 1. Taking S to be the heavier of S_1 and S_2 we get the required lower bound of $\frac{3}{4}w(C)$.

Case 2. Such v does not exists, so we have two loose ends u, v connected to two different vertices of C, say u connected to a, and v connected to b. Let c be the remaining vertex of C. Notice that all biedges of C are allowed. For if any of them, call it xy, were not allowed, then x and y would be connected with a bipath in P, and that cannot happen, since we know that either the bipath starting in x or the bipath starting in y ends in a loose end.

Consider the two solutions defined in the previous case: $S_1 = \{ab, cv\}$ and $S_2 = \{bc, av\}$. They are both allowed. For example, adding S_1 to P forms a bipath $\ldots cv \ldots ba \ldots u$ ending in a loose end u, so no bicycles are formed. Similar argument works for S_2. The weight argument is the same as in Case 1. □

Lemma 6. *Let P be a family of disjoint bipaths satisfying Invariant 1 with at least 2 loose ends and let C be an unprocessed 4-cycle. Then there exists an allowed family of biedges S such that (i) $w(S) \geq \frac{1}{2}w(C)$, (ii) after processing C, the family $P \cup S$ satisfies Invariant 1, and (iii) the number of loose ends does not change.*

Proof. Our plan is to make S contain two biedges with both endvertices on C or one biedge with both endvertices on C and one biedge linking a vertex of C with one of the loose ends. This satisfies *(ii)* and *(iii)* and again we only need to guarantee that S is allowed and that it has weight at least $\frac{1}{2}w(C)$. We consider the same two cases as in the previous lemma.

Case 1. There exists a loose end v not connected to C in P.

Let $C = abcd$ and let Q be a plane spanning bipath of C with no forbidden edges. We consider all solutions of the following form: a biedge of Q and a biedge connecting one of the remaining vertices of C and v. There a six such solutions since Q has 3 edges and there are always 2 remaining vertices. All these solutions are allowed. That is because the bipath edge is allowed by itself, and the linking edge cannot form a cycle in P since v is not connected with C in P.

Let us now bound the total weight of these six solutions. Consider a pair of solutions corresponding to a single biedge of Q, say xy. The total weight of these two solutions is $2w(xy) + w(vz) + w(vw) \geq 2w(xy) + w(zw)$ (by triangle inequality), where z, w are the two remaining vertices. So we get twice the weight of the bipath biedge and the weight of the complementary biedge. Now, notice that for any plane spanning bipath Q of a 4-cycle, the complementary biedges of biedges of Q also form a plane spanning bipath. It follows from Lemma 1 that the total weight of all six solutions is at least $3w(C)$. Taking S to be the heaviest of the six solutions gives the required lower bound of $\frac{1}{2}w(C)$.

Case 2. Such v does not exists, so we have two loose ends u, v connected to two different vertices of C. Let $C = abcd$. We have two cases.

Case 2a. v and u are connected to two successive cycle vertices, say u is connected to a and v is connected to b. Consider two solutions: $S_1 = \{da, bc\}$ and

$S_2 = \{ab, cv\}$ (here cv is a dummy biedge, added only to keep the number of loose ends constant for simplicity). Both solutions are allowed, because if we add any of them to P, each of the added biedges lies on a bipath ending in a loose end. Also $w(S_1) + w(S_2) \geq w(C)$ by Lemma 1, because $\{da, bc, ab\}$ is a plane spanning bitree of C.

Case 2b. v and u are connected to opposite cycle vertices, say u is connected to a and v is connected to c. Consider two solutions: $S_1 = \{ab, cd\}$ and $S_2 = \{ad, bc\}$. The rest of the argument is the same as in the previous Case 2a. □

For technical reasons, that will become clear in the proof of Theorem 7, the very last cycle needs to be processed even more effectively. This is possible, because when processing the last cycle we can make P a Hamiltonian bicycle. To deal with this special case we use the following lemmas (we defer their proofs to the full version of the paper).

Lemma 7. *Let P be a family of disjoint bipaths satisfying Invariant 1 with exactly 1 loose end. Assume that all cycles have been processed except for one 3-cycle C. Then there exists an allowed family of biedges S such that (i) $P \cup S$ is a Hamiltonian bicycle, (ii) $w(S) \geq \frac{3}{4}w(C)$.*

Lemma 8. *Let P be a family of disjoint bipaths satisfying Invariant 1 with exactly 2 loose ends. Assume that all cycles have been processed except for one 4-cycle C. Then there exists an allowed family of biedges S such that (i) $P \cup S$ is a Hamiltonian bicycle, (ii) $w(S) \geq \frac{2}{3}w(C)$.*

Lemma 9. *Let P be a family of disjoint bipaths satisfying Invariant 1 with no loose ends. Assume that all cycles have been processed except for one 4-cycle C. Then there is an allowed family of biedges S such that (i) $P \cup S$ is a Hamiltonian bicycle, (ii) $w(S) \geq \frac{1}{2}w(C)$.*

4.3 Putting It All Together

Theorem 7. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be a nice pair of cycle covers of G. Then, there exists a Hamiltonian bicycle in G with weight at least $W_2 + \frac{5}{8}W_3 + \frac{1}{2}W_4 + \frac{1}{2}W_5 + \frac{1}{3}W_6 + \frac{1}{3}W_7 + \frac{1}{4}W_{8+}$, where W_k is the total weight of k-cycles in $\mathcal{C}_1$ and $\mathcal{C}_2$ and W_{8+} is the total weight of cycles of length at least 8 in $\mathcal{C}_1$ and $\mathcal{C}_2$.*

Proof. We use the Main Algorithm and process all the long (i.e. of length at least 5) cycles before the 3- and 4-cycles. Long cycles are processed using Lemma 3. As a result we get a family P of bipaths satisfying Invariant 1 and such that $w(P) \geq W_2 + \frac{1}{2}W_5 + \frac{1}{3}W_6 + \frac{1}{3}W_7 + \frac{1}{4}W_{8+}$. Depending on the number of loose ends in P, we continue in one of the following ways.

Case 1. There are at least 2 loose ends. Then we first process 4-cycles, in any order, using Lemma 6 for each cycle. Note that $w(P)$ increases by at least $\frac{1}{2}W_4$ during this phase. Next we process 3-cycles in order of decreasing weight. The first 3-cycle A is processed using Lemma 5. As a result the number of loose ends drops by 1 and $W(P)$ increases by $\frac{3}{4}w(A)$. Then we process the second 3-cycle B using Lemma 4. We get one loose end and $W(P)$ increases by $\frac{1}{2}w(B)$. We process all the 3-cycles in this way, alternating between Lemmas 5 and 4. Clearly that overall $W(P)$ increases by at least $\frac{5}{8}W_3$, hence after patching P to a Hamiltonian bicycle we get its total weight as claimed.

Case 2. There are no loose ends. Note that, when a cycle C is processed, the number of loose ends increases by $|C| - 2|S|$. Hence, at any time, the parity of the number of loose ends equals the parity of the sum of lengths of the processed cycles. It follows that if there are no loose ends then the sum of lengths of the processed cycles is even. On the other hand, the sum of lengths of all cycles in $\mathcal{C}_1$ and $\mathcal{C}_2$ is $2n$, hence also the sum of lengths of the unprocessed cycles is even. It implies that the number of 3-cycles is even. Now we will consider subcases regarding the number of 3-cycles and 4-cycles.

Case 2a. There are at least two 4-cycles. Then we start by processing the lightest 4-cycle using Lemma 4. This gives us 2 loose ends. Next, all 3-cycles and all but one remaining 4-cycles are processed using the algorithm from Case 1. Again, since the number of 3-cycles is even, we still have 2 loose ends when this phase is finished. It follows that the remaining 4-cycle can be processed using Lemma 8. We see that in total $w(P)$ increases by $\frac{1}{3}$ of the weight of the lightest 4-cycle, $\frac{2}{3}$ of the weight of some other 4-cycle, $\frac{1}{2}$ of the weight of all the other 4-cycles and by $\frac{5}{8}W_3$, which is at least $\frac{5}{8}W_3 + \frac{1}{2}W_4$, as required.

Case 2b. There are at least four 3-cycles. Then we start by processing the two lightest 3-cycles using Lemma 4. This gives us 2 loose ends and $w(P)$ increases by $\frac{1}{2}$ of the weight of these 3-cycles. Next, all 4-cycles and all but two remaining 3-cycles are processed using the algorithm from Case 1. This increases $w(P)$ by $\frac{5}{8}$ of the weight of the triangles processed in this phase and by $\frac{1}{2}W_4$. Note that since the number of 3-cycles is even, we still have 2 loose ends after this phase. The two remaining 3-cycles are processed using Lemma 5 and Lemma 7, respectively. Then $w(P)$ increases by $\frac{3}{4}$ of their weight. During the processing of all short cycles $w(P)$ increases by at least $\frac{5}{8}W_3 + \frac{1}{2}W_4$, as required.

Case 2c. There are two 3-cycles and one 4-cycle. Then we consider two methods of processing these cycles and we choose the more profitable one. Method 1: process the 3-cycles using Lemma 4 and obtaining 2 loose ends, then process the 4-cycle using Lemma 8. In this case $w(P)$ increases by $\frac{1}{2}W_3 + \frac{2}{3}W_4$. Method 2: process the 4-cycle using Lemma 4 and obtaining 2 loose ends, then process the 3-cycles using Lemma 5 for the first one and Lemma 7 for the second one. In this case $w(P)$ increases by $\frac{3}{4}W_3 + \frac{1}{3}W_4$. Clearly the better method gives us $\max\{\frac{1}{2}W_3 + \frac{2}{3}W_4, \frac{3}{4}W_3 + \frac{1}{3}W_4\} \geq \frac{5}{8}W_3 + \frac{1}{2}W_4$, as required .

Case 2d. There are no 3-cycles and there is one 4-cycle. We use Lemma 9.

Case 2e. There are two 3-cycles and no 4-cycles. We process the lighter 3-cycle A using Lemma 4 which gives us 1 loose end. Then the second 3-cycle B can be processed using Lemma 7. This increases $w(P)$ by at least $\frac{1}{2}w(A) + \frac{3}{4}w(B) \geq \frac{5}{8}W_3$ as required.

Case 3. There is exactly one loose end. By the parity argument from Case 2., the number of 3-cycles is odd. We can treat the single loose end as an imaginary 3-cycle I of weight 0. This way the number of 3-cycles becomes even and we again arrive at Case 2. Note that in the algorithms from subcases 2a, 2b and 2e the imaginary triangle would be processed using Lemma 4. If we just do nothing while processing I we get the same effect: $w(P)$ grows by $\frac{1}{2}w(I) = 0$ and we get an additional loose end. Case 2d does not apply since we do have 3-cycles. The only case left is a counterpart of Case 2c: there is one 3-cycle and one 4-cycle. Similarly to Case 2c we consider 2 methods and we choose the more profitable one. Method 1 is: process the 3-cycle using Lemma 4 obtaining the second loose end and then process the 4-cycle using Lemma 8. Method 2 is: process the 4-cycle using Lemma 4 obtaining two more loose ends and then process the 3-cycle using Lemma 5. Performing the same calculations as in Case 2c, we see that $w(P)$ increases by at least $\frac{5}{8}W_3 + \frac{1}{2}W_4$, as required. □

Theorem 8. *There exists a $\frac{35}{44}$-approximation algorithm for semimetric maxTSP.*

Proof. Similarly to the algorithm in Theorem 6, our algorithm chooses the heaviest of the four Hamiltonian cycles: two constructed by Kostochka and Serdukov's algorithm and the two cycles of the bicycle from Theorem 7. Again, by simple LP reasoning, one can show that the resulting cycle has weight $\geq \frac{35}{44}\text{OPT}$. □

Acknowledgments The authors would like to thank Kasia Paluch for many helpful discussions.

References

1. M. Bläser and B. Manthey. Two approximation algorithms for 3-cycle covers. In *APPROX'02*, pages 40–50, 2002.
2. M. Bläser, S. Ram, and M. Sviridenko. Improved approximation algorithms for metric maximum ATSP and maximum 3-cycle cover problems. In *WADS'05*, pages 350–359, 2005.
3. R. Hassin and S. Rubinstein. Better approximations for max TSP. *Inf. Process. Lett.*, 75(4):181–186, 2000.
4. R. Hassin and S. Rubinstein. A 7/8-approximation algorithm for metric Max TSP. *Inf. Process. Lett.*, 81(5):247–251, 2002.
5. 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.
6. S. R. Kosaraju, J. K. Park, and C. Stein. Long tours and short superstrings (preliminary version). In *FOCS'94*, pages 166–177, 1994.
7. A. V. Kostochka and A. I. Serdyukov. Polynomial algorithms with the estimates 3/4 and 5/6 for the traveling salesman problem of the maximum (in Russian). *Upravlyaemye Sistemy*, 26:55–59, 1985.
8. M. Lewenstein and M. Sviridenko. A 5/8 approximation algorithm for the maximum asymmetric TSP. *SIAM J. Discrete Math.*, 17(2):237–248, 2003.
9. A. I. Serdyukov. The traveling salesman problem of the maximum (in Russian). *Upravlyaemye Sistemy*, 25:80–86, 1984.