Simpler Approximation of the Maximum Asymmetric Traveling Salesman Problem

Katarzyna Paluch*[1], Khaled Elbassioni[2], and Anke van Zuylen[2]

1 **Institute of Computer Science, University of Wroclaw**
ul. Joliot-Curie 15, room 304, 50-383 Wroclaw, Poland
`abraka@cs.uni.wroc.pl`

2 **Max Planck Institute for Informatics**
Campus E 13, Saarbrücken, Germany
`{elbassio,anke}@mpi-inf.mpg.de`

Abstract

We give a very simple approximation algorithm for the maximum asymmetric traveling salesman problem. The approximation guarantee of our algorithm is 2/3, which matches the best known approximation guarantee by Kaplan, Lewenstein, Shafrir and Sviridenko. Our algorithm is simple to analyze, and contrary to previous approaches, which need an optimal solution to a linear program, our algorithm is combinatorial and only uses maximum weight perfect matching algorithm.

1998 ACM Subject Classification I.1.2 Algorithms

Keywords and phrases approximation algorithm, maximum asymmetric traveling salesman problem

Digital Object Identifier 10.4230/LIPIcs.STACS.2012.501

1 Introduction

In this paper, we study the maximum asymmetric traveling salesman problem (MAX ATSP). The input to the MAX ATSP is a directed complete graph $G = (V, A)$ with edge weights $w(e) \geq 0$ for all $e = (i, j) \in A$. The objective is to find a tour of maximum weight. This problem is known to be APX-hard [11], and research has focused on finding good approximation algorithms for this problem. Good approximation algorithms are of particular interest because they imply good approximations for a number of related problems. For example, it was shown by Breslauer, Jiang and Jiang [3] that an α-approximation algorithm for MAX ATSP implies a $(\frac{7}{2} - \frac{3}{2}\alpha)$-approximation algorithm for the shortest superstring problem; a problem that arises in DNA sequencing and data compression. Any α-approximation algorithm for MAX ATSP implies an algorithm with the same guarantee for the maximal compression problem defined by Tarhio and Ukkonen [12].

The current best approximation algorithm for MAX ATSP is due to Kaplan, Lewenstein, Shafrir and Sviridenko [5] and achieves an approximation guarantee of $\frac{2}{3}$. Their algorithm needs the optimal solution of an LP relaxation of the max ATSP, which is scaled up to an integral solution by multiplying it by the least common denominator of all variables. From the scaled up solution, a pair of cycle covers is extracted that have a combined weight of at least twice the weight of the optimum solution. Finally the obtained pair of cycle covers

* Supported by MNiSW grant number N N206 368839, 2010-2013.

is processed using a rather complicated coloring lemma. Previous results on MAX ATSP appeared among others in [4], [6] [1], [9]. In this paper, we give a very simple approximation algorithm that achieves the same guarantee as the algorithm by Kaplan et al. Our algorithm is combinatorial in nature: it constructs a certain matching instance and computes a maximum weight matching. We then give a simple procedure that uses this matching to form three tours, and we show that the average weight of these tours is at least $\frac{2}{3}$ times the optimum.

Other variants of the maximum traveling salesman problem that have been considered are among others: the maximum symmetric traveling salesman problem (MAX TSP), in which the underlying graph is undirected - currently the best known approximation ratio is $\frac{7}{9}$ [10], the maximum metric symmetric traveling salesman problem, in which the edge weights satisfy the triangle inequality - the best approximation factor is $\frac{7}{8}$ [7], the maximum asymmetric traveling salesman problem with triangle inequality - the best approximation ratio is $\frac{35}{44}$ [8].

The key idea in our approach to MAX ATSP is the following. A natural first idea that has been very fruitful when designing an approximation algorithm for maximum traveling salesman problems is to start with a *cycle cover* of maximum weight, i.e., a maximum weight collection of edges such that each node is incident to exactly two edges (in the undirected case) or exactly one incoming and one outgoing edge (in the directed case). Such a cycle cover can be found in polynomial time by a reduction to maximum weight matching. Since the optimal tour is a cycle cover, the weight of the maximum weight cycle cover is at least the weight of the optimal tour. By removing the lightest edge from each cycle, we obtain a collection of node disjoint paths, which can be arbitrarily connected to form a tour. For the asymmetric case, this approach will give a $\frac{1}{2}$-approximation algorithm, since the cycle cover may contain cycles of length two (2-cycles). If we could find a maximum weight cycle cover without 2-cycles, then we would achieve a $\frac{2}{3}$-approximation, but, unfortunately, finding a maximum weight cycle cover without 2-cycles is APX-hard [2].

Our key observation is the fact that we can exclude 2-cycles from the cycle cover, if we allow the cycle cover to contain "half edges": We split each edge (i, j) into two half edges; the head of (i, j) (which is incident to j but not to i) and the tail of (i, j) (which is incident to i, but not to j). A half edge has weight equal to half the weight of the original edge. Now, for a pair of edges $(i, j), (j, i)$, we ensure that the solution does not contain both edges, but we do allow the solution to contain the heads of both (i, j) and (j, i) or the tails of both (i, j) and (j, i). We show that such a cycle cover with half edges can be computed by an appropriate reduction to a maximum weight perfect matching problem. Finally, we show how to use the cycle cover with half edges to extract three tours with total weight at least twice the weight of the cycle cover.

2 Cycle Covers without 2-Cycles but with Half-Edges

We begin by introducing the maximum weight cycle cover problem without 2-cycles, but with half-edges, and showing it can be computed in polynomial time. Given a complete directed graph $G = (V, E)$ and weights $w(e) \geq 0$ for each $e \in E$, a cycle cover is a subset C of the edges, so that each $i \in V$ has exactly one outgoing and one incoming edge in C. In the maximum weight cycle cover problem with half-edges, we allow the solution C to contain "only the head or only the tail" of an edge (i, j). Such a half-edge has weight $\frac{1}{2}w(i, j)$ and is thought to be incident to only one endpoint of the edge (i, j). We introduce these half-edges so that we can ensure that our cycle cover does not have 2-cycles. This gives rise to the following problem:

▶ **Definition 1.** Given a directed graph $G = (V, E)$ with edge weights $w(i, j) \geq 0$ for every $(i, j) \in E$, let $\tilde{G} = (\tilde{V}, \tilde{E})$ be the graph obtained from G by replacing each $(i, j) \in E$ by a node $v_{(i,j)}$ and two edges $(i, v_{(i,j)})$ and $(v_{(i,j)}, j)$, each with weight $\frac{1}{2}w(i, j)$. A *cycle cover without 2-cycles but with half-edges* is a subset $\tilde{C} \subseteq \tilde{E}$ such that

(i) each node in V has exactly one outgoing and one incoming edge in $\tilde{C}$;
(ii) for each $(i, j) \in E$, $\tilde{C}$ contains either zero edges from $\{(i, v_{(i,j)}), (v_{(i,j)}, j), (j, v_{(j,i)}), (v_{(j,i)}, i)\}$, or it contains exactly one edge incident to i and one edge incident to j.

▶ **Lemma 2.** *We can find a maximum weight cycle cover without 2-cycles but with half-edges in polynomial time.*

Proof. We reduce the problem of finding $\tilde{C}$ of maximum weight to a maximum weight perfect matching problem in the following undirected graph G':

For each node $i \in V$, we create two nodes, in_i and out_i. For an edge $e = (i, j) \in E$, we create two nodes $\text{head}_{(i,j)}$ and $\text{tail}_{(i,j)}$, and we have three undirected edges $\{\text{out}_i, \text{tail}_{(i,j)}\}, \{\text{tail}_{(i,j)}, \text{head}_{(i,j)}\}$ and $\{\text{head}_{(i,j)}, \text{in}_j\}$. The weight of the first and third edge is $\frac{1}{2}w(i, j)$, and the weight of the second edge is 0.

Note that a perfect matching in this graph corresponds to a cycle cover of G of the same weight and vice versa: for each edge (i, j) of G, a perfect matching of G' either contains edge $\{\text{tail}_{(i,j)}, \text{head}_{(i,j)}\}$ or edges $\{\text{out}_i, \text{tail}_{(i,j)}\}, \{\text{head}_{(i,j)}, \text{in}_j\}$. Of course, if we are only interested in computing a cycle cover of G, then it suffices to split each node i into two: in_i and out_i and connect in_j and out_i via an edge whenever $(i, j) \in G$ and compute a perfect matching in the thus obtained bipartite undirected graph. We will now show how to modify G' so that a perfect matching in G' corresponds to a maximum weight cycle cover without 2-cycles but with half-edges.

A perfect matching in G' defines a set of half-edges $\tilde{C} \subseteq \tilde{E}$ of the same weight that satisfies property (i) in Definition 1, but $\tilde{C}$ may contain four half-edges that correspond to a 2-cycle in G. To enforce that $\tilde{C}$ also satisfies property (ii), we add two additional nodes, $i_{\{i,j\}}$ and $j_{\{i,j\}}$ for each pair of edges $(i, j), (j, i)$. We add an edge from $i_{\{i,j\}}$ to $\text{tail}_{(i,j)}$ and $\text{head}_{(j,i)}$, and we add an edge from $j_{\{i,j\}}$ to $\text{head}_{(i,j)}$ and $\text{tail}_{(j,i)}$. These edges all have weight 0. The fact that the additional nodes need to be matched ensures that $\tilde{C}$ does not contain all four half-edges corresponding to (i, j) and (j, i).

For a pair of edges $(i, j), (j, i)$ in G, the matching instance G' thus has a gadget containing 10 edges. See Figure 1. We now verify that a perfect matching in G' corresponds to a cycle

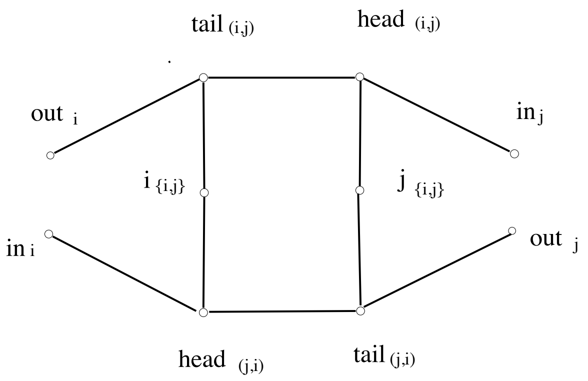

Figure 1 A gadget corresponding to a 2-cycle on vertices i and j.

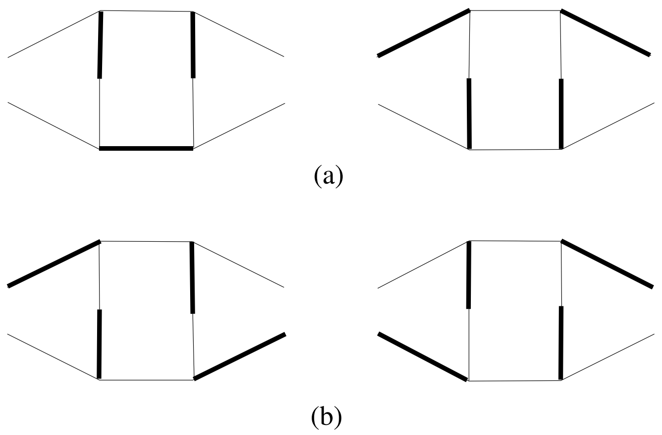

Figure 2 Possible ways of matching vertices $i_{\{i,j\}}, j_{\{i,j\}}, \mathrm{head}_{(i,j)}, \mathrm{tail}_{(i,j)}, \mathrm{head}_{(j,i)}, \mathrm{tail}_{(j,i)}$.

cover without 2-cycles but with half-edges $\tilde{C}$ of the same weight. If a perfect matching M of G' matches nodes $i_{\{i,j\}}$ and $j_{\{i,j\}}$ as shown in Figure 2(a), then for one of the edges $(i,j),(j,i)$, both of the corresponding half-edges are excluded from $\tilde{C}$ and for the other edge, either both of the corresponding half-edges, or neither of the corresponding half-edges will be in $\tilde{C}$. If nodes $i_{\{i,j\}}$ and $j_{\{i,j\}}$ are matched as in Figure 2(b), M must contain appropriately either $\{\mathrm{out}_i, \mathrm{tail}_{(i,j)}\}, \{\mathrm{out}_j, \mathrm{tail}_{(j,i)}\}$ or $\{\mathrm{in}_i, \mathrm{head}_{(j,i)}\}, (\mathrm{in}_j, \mathrm{head}_{(i,j)})$, i.e., $\tilde{C}$ contains either $(i, v_{(i,j)})$ and $(j, v_{(j,i)})$ or $(v_{(j,i)}, i)$ and $(v_{(i,j)}, j)$. So indeed, condition (ii) of Definition 1 is satisfied. ◀

▶ **Lemma 3.** *Given a graph G, let $\tilde{C}$ be a cycle cover without 2-cycles but with half-edges. Then, we can construct three sets of node disjoint paths $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ in G, with total weight at least twice the weight of $\tilde{C}$.*

Proof. We use $\tilde{C}$ to construct a set F of directed and undirected edges with endpoints in V, and we then decompose F into three paths. For a pair of vertices i, j, if both $(i, v_{(i,j)})$ and $(v_{(i,j)}, j)$ are in $\tilde{C}$, then we replace it by the edge (i,j). If both $(i, v_{(i,j)})$ and $(j, v_{(j,i)})$, or $(v_{(i,j)}, j)$ and $(v_{(j,i)}, i)$ are in $\tilde{C}$, then we add undirected edge $\{i,j\}$. We think of an undirected edge as having two tails (and no heads) in the first case, and two heads (and no tails) in the second case. Note that it is then the case that each node is the head of one edge and the tail of one edge in F, by property (i) of Definition 1.

We will show how to find three sets of node-disjoint paths, such that each edge $(i,j) \in F$ is in exactly two of the paths, and for an undirected edge $\{i,j\} \in F$, there is one path that contains (i,j) and one path that contains (j,i). Note that the sum of the weights of these three sets of paths is exactly equal to twice the weight of $\tilde{C}$.

We consider a weakly connected component in (V, F). Note that a component has at least three nodes by property (ii) of Definition 1. If the component has no undirected edges, then it is a directed cycle, since each node is the head of one edge and the tail of another edge. Let F' be the edges in the component. We take two adjacent edges e_1, e_2 and make this the first path, the second path is $F' \backslash \{e_1\}$ and the third path is $F' \backslash \{e_2\}$.

Otherwise, if F' does contain undirected edges, note that (V, F') is a cycle if we ignore the direction of all edges. Moreover, it is the case that the number of undirected edges in the cycle is even and an undirected edge having two heads (resp. two tails) is followed on the cycle by an undirected edge having two tails (resp. two heads). To see this, recall that each

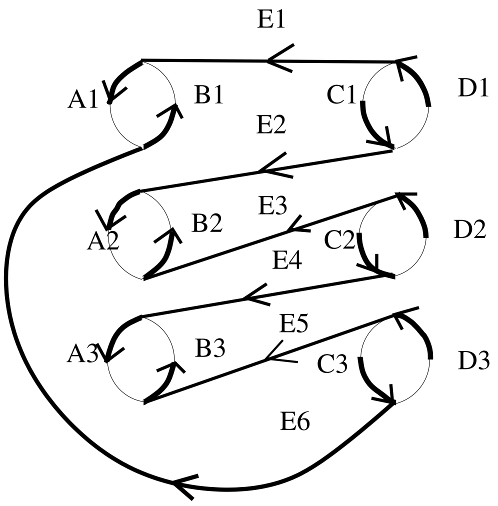

Figure 3 $E_1, E_2, \dots, E_6$ represent directed paths. To P_1 we add edges on paths E_1, E_3, E_5 as well as edges $A_1, D_1, B_2, D_2, B_3, D_3$. To P_2 we add edges on paths E_2, E_4, E_6 as well as edges $B_1, C_1, A_2, C_2, A_3, C_3$. To P_3 we add edges on paths $E_1, E_2, \dots, E_6$.

node is the head and the tail of one edge, and that undirected edges have either two heads or two tails.

The paths to be added to $\mathcal{P}_1, \mathcal{P}_2$ and $\mathcal{P}_3$ are now constructed as follows: to $\mathcal{P}_1$ we add the directed edges that point in clockwise direction and we add the undirected edges, which we make directed by directing them in clockwise direction. To $\mathcal{P}_2$ we add the directed edges that point in anticlockwise direction, and we add the undirected edges and direct them in anticlockwise direction. Finally, we also add all directed edges from F' to $\mathcal{P}_3$. An example of this procedure is shown in Figure 3.

There is one exception to the above construction. If the directed edges in F' form one path (possibly an empty path), then at least one of the two sets $\mathcal{P}_1, \mathcal{P}_2$ contains a cycle. In this case, we take one undirected edge in F' and reverse its direction in both $\mathcal{P}_1$ and $\mathcal{P}_2$.

Clearly, the paths are node disjoint, each directed edge is contained in two of the three sets $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$, and for each undirected edge $\{i, j\}$, there is one set of paths that contains (i, j) and one set of paths that contains (j, i). ◀

▶ **Corollary 4.** *There exists a $\frac{2}{3}$-approximation algorithm for max ATSP.*

Proof. The optimal tour gives a set $\tilde{C}$ that is a cycle cover without 2-cycles but with half-edges, if for each edge (i, j) in the optimal tour, we include $(i, v_{(i,j)})$ and $(v_{(i,j)}, j)$ in $\tilde{C}$. Hence the maximum weight set $\tilde{C}$ has weight at least OPT, and it can be computed in polynomial time using Lemma 2. Using Lemma 3, we can thus find three sets of node disjoint paths, $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$, of total weight $2OPT$. We can arbitrarily connect the paths in $\mathcal{P}_i$ into a tour without decreasing the weight, for $i = 1, 2, 3$ and hence, one of these tours has weight at least $\frac{2}{3}OPT$. ◀

Acknowledgements

The authors thank Marcin Mucha for useful comments on an earlier version of this paper.

References

1 Markus Bläser. An 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 *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, volume 2462 of *Lecture Notes in Computer Science*, pages 40–50. Springer, 2002.

3 Dany Breslauer, Tao Jiang, and Zhigen Jiang. Rotations of periodic strings and short superstrings. *J. Algorithms*, 24(2):340–353, 1997.

4 M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for finding a maximum weight Hamiltonian circuit. *Oper. Res.*, 27(4):799–809, 1979.

5 Haim Kaplan, Moshe Lewenstein, Nira Shafrir, and Maxim Sviridenko. Approximation algorithms for asymmetric tsp by decomposing directed regular multigraphs. *J. ACM*, 52(4):602–626, 2005. Preliminary version appeared in FOCS'03.

6 S. Rao Kosaraju, James K. Park, and Clifford Stein. Long tours and short superstrings (preliminary version). In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 166–177, 1994.

7 Lukasz Kowalik and Marcin Mucha. Deterministic 7/8-approximation for the metric maximum tsp. *Theor. Comput. Sci.*, 410(47-49):5000–5009, 2009.

8 Lukasz Kowalik and Marcin Mucha. 35/44-approximation for asymmetric maximum tsp with triangle inequality. *Algorithmica*, 59(2):240–255, 2011.

9 Moshe Lewenstein and Maxim Sviridenko. A 5/8 approximation algorithm for the maximum asymmetric tsp. *SIAM J. Discrete Math.*, 17(2):237–248, 2003.

10 Katarzyna E. Paluch, Marcin Mucha, and Aleksander Madry. A 7/9 - approximation algorithm for the maximum traveling salesman problem. In *APPROX-RANDOM*, pages 298–311, 2009.

11 Christos H. Papadimitriou and Mihalis Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18:1–11, 1993.

12 Jorma Tarhio and Esko Ukkonen. A greedy approximation algorithm for constructing shortest common superstrings. *Theor. Comput. Sci.*, 57:131–145, 1988.