Improved Deterministic Approximation Algorithms for Max TSP

Zhi-Zhong Chen [*] Yuusuke Okamoto [†] Lusheng Wang [‡]

Abstract

We present an $O(n^3)$-time approximation algorithm for the maximum traveling salesman problem whose approximation ratio is asymptotically $\frac{61}{81}$, where n is the number of vertices in the input complete edge-weighted (undirected) graph. We also present an $O(n^3)$-time approximation algorithm for the metric case of the problem whose approximation ratio is asymptotically $\frac{17}{20}$. Both algorithms improve on the previous bests.

1 Introduction

The *maximum traveling salesman problem* (Max TSP) is to compute a maximum-weight Hamiltonian circuit (called a *tour*) in a given complete edge-weighted (undirected) graph. The problem is known to be Max-SNP-hard [1] and there have been a number of approximation algorithms known for it [4, 5, 10]. In 1984, Serdyukov [10] gave an $O(n^3)$-time approximation algorithm for Max TSP that achieves an approximation ratio of $\frac{3}{4}$. Serdyukov's algorithm is very simple and elegant, and it tempts one to ask if a better approximation ratio can be achieved for Max TSP by a polynomial-time approximation algorithm. However, previous to our work, there was no (deterministic) polynomial-time algorithm with an approximation ratio better than $\frac{3}{4}$. Interestingly, Hassin and Rubinstein [5] showed that with the help of randomization, a better approximation ratio for Max TSP can be achieved. More precisely, they gave a randomized $O(n^3)$-time approximation algorithm (H&R-algorithm) for Max TSP whose *expected* approximation ratio is asymptotically $\frac{25}{33}$. The expected approximation ratio $\frac{25}{33}$ of H&R-algorithm does not guarantee that with (reasonably) high probability (say, a constant), the weight of its output tour is at least $\frac{25}{33}$ times the optimal. So, it is much more desirable to have a (deterministic) approximation algorithm for Max TSP that achieves an approximation ratio better than $\frac{3}{4}$ (and runs at least as fast as Serdyukov's algorithm). In this paper, we give the first such (deterministic) approximation algorithm for Max TSP; its approximation ratio is asymptotically $\frac{61}{81}$ and its running time is $O(n^3)$. While this improvement is small (as Hassin and Rubinstein said about their algorithm), it at least demonstrates that the ratio of $\frac{3}{4}$ can be improved and further research along this line is encouraged. Our algorithm is basically a nontrivial derandomization of H&R-algorithm.

We note in passing that Chen and Wang [3] have recently improved H&R-algorithm to a randomized $O(n^3)$-time approximation algorithm whose expected approximation ratio is asymptotically $\frac{251}{331}$. Their new algorithm is complicated and even more difficult to derandomize.

[*] Department of Mathematical Sciences, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan. Email: chen@r.dendai.ac.jp. Part of work done during a visit at City University of Hong Kong.

[†] Department of Mathematical Sciences, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan.

[‡] Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong.

The *metric* case of Max TSP has also been considered in the literature. In this case, the weights on the edges of the input graph obey the triangle inequality. What is known for this case is very similar to that for Max TSP. In 1985, Kostochka and Serdyukov [8] gave an $O(n^3)$-time approximation algorithm for metric Max TSP that achieves an approximation ratio of $\frac{5}{6}$. Their algorithm is very simple and elegant. Tempted by improving the ratio $\frac{5}{6}$, Hassin and Rubinstein [6] gave a randomized $O(n^3)$-time approximation algorithm (H&R2-algorithm) for metric Max TSP whose *expected* approximation ratio is asymptotically $\frac{7}{8}$. In this paper, by nontrivially derandomizing H&R2-algorithm, we give a (deterministic) $O(n^3)$-time approximation algorithm for metric Max TSP whose approximation ratio is asymptotically $\frac{17}{20}$, an improvement over the previous best ratio (namely, $\frac{5}{6}$). Our algorithm also has the advantage of being easy to parallelize.

2 Basic Definitions

Throughout this paper, a graph means a simple undirected graph (i.e., it has neither parallel edges nor self-loops), while a multigraph may have parallel edges but no self-loops.

Let G be a graph. We denote the vertex set of G by $V(G)$, and denote the edge set of G by $E(G)$. The *degree* of a vertex v in G is the number of edges incident to v in G. A *cycle* in G is a connected subgraph of G in which each vertex is of degree 2. A *path* in G is either a single vertex of G or a connected subgraph of G in which exactly two vertices are of degree 1 and the others are of degree 2. The *length* of a cycle or path C is the number of edges in C. A *tour* (also called a *Hamiltonian cycle*) of G is a cycle C of G with $V(C) = V(G)$. A *cycle cover* of G is a subgraph H of G with $V(H) = V(G)$ in which each vertex is of degree 2. A *subtour* of G is a subgraph H of G in which each connected component is a path. Two edges of G are *adjacent* if they share an endpoint. A *matching* of G is a (possibly empty) set of pairwise nonadjacent edges of G. A *perfect matching* of G is a matching M of G such that each vertex of G is an endpoint of an edge in M. For a subset F of $E(G)$, $G - F$ denotes the graph obtained from G by deleting the edges in F.

Throughout the rest of the paper, fix an instance (G, w) of Max TSP, where G is a complete (undirected) graph and w is a function mapping each edge e of G to a nonnegative real number $w(e)$. For a subset F of $E(G)$, $w(F)$ denotes $\sum_{e \in F} w(e)$. The *weight* of a subgraph H of G is $w(H) = w(E(H))$. Our goal is to compute a tour of large weight in G. We assume that $n = |V(G)|$ is odd; the case where n is even is simpler. For a random event A, $\Pr[A]$ denotes the probability that A occurs. For a random variable X, $\mathcal{E}[X]$ denotes the expected value of X.

3 Algorithm for Max TSP

This section is divided into three subsections. In Section 3.1, we sketch H&R-algorithm. In Section 3.2, we describe our derandomization of H&R-algoirthm and analyze its approximation ratio. In Section 3.3, we give the details that are omitted in Section 3.2.

3.1 Sketch of H&R-algorithm

H&R-algorithm starts by computing a maximum-weight cycle cover $\mathcal{C}$. If $\mathcal{C}$ is a tour of G, then we are done. Throughout the rest of this section, we assume that $\mathcal{C}$ is not a tour of G. Suppose that T is a maximum-weight tour of G. Let T_{int} denote the set of all edges $\{u, v\}$ of T such that some cycle C in $\mathcal{C}$ contains both u and v. Let T_{ext} denote the set of edges in T but not in T_{int}. Let $\alpha = w(T_{\text{int}})/w(T)$.

H&R-algorithm then computes three tours T_1, T_2, T_3 of G and outputs the one of the largest weight. Based on an idea in [4], T_1 is computed by modifying the cycles in $\mathcal{C}$ as follows. Fix a parameter $\epsilon > 0$. For each cycle C in $\mathcal{C}$, if $|E(C)| > \epsilon^{-1}$, then remove the minimum-weight edge; otherwise, replace C by a maximum-weight path P in G with $V(P) = V(C)$. Then, $\mathcal{C}$ becomes a subtour and we can extend it to a tour T_1 in an arbitrary way. As observed by Hassin and Rubinstein [5], we have:

Fact 3.1 $w(T_1) \geq (1-\epsilon)w(T_{\text{int}}) = (1-\epsilon)\alpha w(T)$.

When $w(T_{\text{ext}})$ is large, $w(T_{\text{int}})$ is small and $w(T_1)$ may be small, too. The two tours T_2 and T_3 together are aimed at the case where $w(T_{\text{ext}})$ is large. By modifying Serdyukov's algorithm, T_2 and T_3 are computed as shown in Figure 1:

1. Compute a maximum-weight matching M in G.
2. Compute a maximum-weight matching M' in a graph H, where $V(H) = V(G)$ and $E(H)$ consists of those $\{u, v\} \in E(G)$ such that u and v belong to different cycles in $\mathcal{C}$.
3. Let $C_1, \ldots, C_r$ be an arbitrary ordering of the cycles in $\mathcal{C}$.
4. Initialize a set N to be empty.
5. For $i = 1, 2, \ldots, r$ (in this order), perform the following two steps:
   (a) Compute two disjoint nonempty matchings A_1 and A_2 in C_i such that each vertex of C_i is incident to an edge in $A_1 \cup A_2$ and both graphs $(V(G), M \cup N \cup A_1)$ and $(V(G), M \cup N \cup A_2)$ are subtours of G.
   (b) Select $h \in \{1, 2\}$ uniformly at random, and add the edges in A_h to N.
6. Complete the graph $(V(G), M \cup N)$ to a tour T_2 of G by adding some edges of G.
7. Let M'' be the set of all edges $\{u, v\} \in M'$ such that both u and v are of degree at most 1 in $\mathcal{C} - N$. Let G'' be the graph obtained from $\mathcal{C} - N$ by adding the edges in M''. (Comment: For each edge $\{u, v\} \in M'$, $\Pr[\{u, v\} \in M''] \geq \frac{1}{4}$. So, $\mathcal{E}[w(M'')] \geq w(M')/4$. Moreover, G'' is a collection of vertex-disjoint cycles and paths; each cycle in G'' must contain at least two edges in M''.)
8. For each cycle C in G'', select one edge in $E(C) \cap M''$ uniformly at random and delete it from G''. (Comment: After this step, $\Pr[\{u, v\} \in E(G'')] \geq \frac{1}{8}$ for each edge $\{u, v\} \in M'$, and hence $\mathcal{E}[w(M' \cap E(G''))] \geq w(M')/8$.)
9. Complete G'' to a tour T_3 of G by adding some edges of G.

Figure 1. Computation of tours T_2 and T_3 in H&R-algorithm.

3.2 Derandomization of H&R-algorithm

Only Steps 5 and 8 in Figure 1 need randomness. Derandomizing Step 8 is easy. However, derandomizing Step 5 is hard, because it may need $\Omega(n)$ random choices which heavily depend on each other. We next show that Step 5 can be derandomized at the cost of making the approximation ratio slightly worse.

For clarity, we transform each edge $\{u, v\} \in M'$ to an ordered pair (u, v), where the cycle C_i in $\mathcal{C}$ with $u \in V(C_i)$ and the cycle C_j in $\mathcal{C}$ with $v \in V(C_j)$ satisfy $i > j$. In detail, to derandomize Step 5, we replace Steps 4 and 5 in Figure 1 by the four steps in Figure 2.

4$'$. For each $h \in \{1, \ldots, 5\}$, initialize a set N_h to be empty.

5$'$. For $i = 1, 2, \ldots, r$ (in this order), process C_i by performing the following two steps:

(a$'$) Compute five subsets $A_1, \ldots, A_5$ of $E(C_i) - M$ satisfying the following four conditions:

(C1) For each $h \in \{1, \ldots, 5\}$, $A_h \neq \emptyset$.

(C2) For each $h \in \{1, \ldots, 5\}$, $A_h \cap (M \cup N_h) = \emptyset$ and the graph $(V(G), M \cup N_h \cup A_h)$ is a subtour of G.

(C3) Each vertex of C_i is an endpoint of at least one edge in $\bigcup_{1 \le j \le 5} A_j$.

(C4) $w(S_i) \ge w(M'_i)/2$, where M'_i is the set of all edges $(u, v) \in M'$ with $u \in V(C_i)$, and S_i is the set of all edges $(u, v) \in M'_i$ such that for at least one $h \in \{1, \ldots, 5\}$, N_h contains an edge incident to v and A_h contains an edge incident to u.

(b$'$) For each $h \in \{1, \ldots, 5\}$, add the edges in A_h to N_h.

6$'$. For each $h \in \{1, \ldots, 5\}$, let M''_h be the set of all edges $(u, v) \in M'$ such that N_h contains an edge incident to u and another edge incident to v. (Comment: By Condition (C4), $w(\bigcup_{1 \le h \le 5} M''_h) \ge w(M')/2$.)

7$'$. Let N be the N_h with $h \in \{1, \ldots, 5\}$ such that $w(M''_h)$ is the maximum among $w(M''_1), \ldots, w(M''_5)$. (Comment: By the comment on Step 6$'$, $w(M''_h) \ge w(M')/10$.)

Figure 2. Modifying Steps 4 and 5 in Figure 1.

The details of computing $A_1, \ldots, A_5$ will be given in Section 3.3. Since we only add the edges of A_h to N_h while processing C_i in Step 5$'$, we indeed maintain the following invariant during Step 5$'$:

Invariant: At the begining of processing each C_i $(1 \le i \le r)$ in Step 5$'$, we have that for each $h \in \{1, \ldots, 5\}$, $N_h \cap M = \emptyset$, $N_h \subseteq \bigcup_{1 \le j \le i-1} E(C_j)$, $N_h \cap E(C_j) \neq \emptyset$ for each $j \in \{1, \ldots, i-1\}$, and the graph $(V(G), M \cup N_h)$ is a subtour of G.

Because of the above invariant, the following lemma is obvious:

Lemma 3.2 *After Step 5$'$, $M \cap N_h = \emptyset$ and $(V(G), M \cup N_h)$ is a subtour of G for each $h \in \{1, \ldots, 5\}$, and N_h contains at least one edge of C_i for each $h \in \{1, \ldots, 5\}$ and for each cycle C_i in $\mathcal{C}$.*

After Steps 4 and 5 in Figure 1 are replaced by the four steps in Figure 2, the first two assertions in the comment on Step 7 in Figure 1 no longer hold and should be replaced by the assertion that $w(M'') \ge w(M')/10$ (see the comment on Step 7$'$ in Figure 2). This is why our derandomization of Step 5 makes the approximation ratio slightly worse.

Derandomizing Step 8 in Figure 1 is easy; it suffices to replace it by the two steps in Figure 3:

8$'$. Compute two disjoint subsets D_1 and D_2 of M'' such that D_1 contains exactly one edge from each cycle in G'' and so does D_2.

9$'$. If $w(D_1) \le w(D_2)$, then remove the edges in D_1 from G''; otherwise, remove the edges in D_2 from G''.

Figure 3. Modifying Step 8 in Figure 1.

In the next lemma, we analyze the approximation ratio of our deterministic algorithm. Recall T, T_{int}, T_{ext}, and α (they are defined at the beginning of this section).

Lemma 3.3 *Let $\delta w(T)$ be the total weight of edges in N (cf. Step $7'$ in Figure 2). Then, $w(T_2) \geq (0.5 - \frac{1}{2n} + \delta)w(T)$, and either $w(T_3) \geq ((1-\delta) + \frac{1}{40}(1-\alpha))w(T)$ or $w(T_3) \geq (1-\delta+\frac{1}{40}-\frac{1}{40n})w(T)$.*

PROOF. Since T_2 contains the original M (a maximum-weight matching of G) as a subset and also contains the edges in N, it is clear that $w(T_2) \geq (0.5 - \frac{1}{2n} + \delta)w(T)$.

Immediately after Step 7 in Figure 1, $w(G'') \geq (1-\delta)w(T) + \frac{1}{10}w(M')$ because $w(M'') \geq w(M')/10$ (see the comment on Step $7'$ in Figure 2). So, immediately after Step $9'$ in Figure 3, $w(G'') \geq (1-\delta)w(T) + \frac{1}{20}w(M')$. Obviously, if $\alpha > 0$, then $w(M') \geq \frac{1}{2}w(T_{\text{ext}}) = \frac{1}{2}(1-\alpha)w(T)$; otherwise, $w(M') \geq (\frac{1}{2} - \frac{1}{2n})w(T_{\text{ext}}) = (\frac{1}{2} - \frac{1}{2n})w(T)$. Thus, either $w(T_3) \geq ((1-\delta)+\frac{1}{40}(1-\alpha))w(T)$ or $w(T_3) \geq (1-\delta+\frac{1}{40}-\frac{1}{40n})w(T)$. □

Now, combining Fact 3.1 and Lemma 3.3 and noting that the running time of the algorithm is dominated by the $O(n^3)$-time needed for computing a maximum-weight cycle cover and two maximum-weight matchings, we have:

Theorem 3.4 *For any fixed $\epsilon > 0$, there is an $O(n^3)$-time approximation algorithm for Max TSP achieving an approximation ratio of $(61 - \frac{20}{n}) \cdot \frac{1-\epsilon}{81-80\epsilon}$.*

3.3 Computation of $A_1, \ldots, A_5$

We now detail the computation of $A_1, \ldots, A_5$. We need some definitions and facts first. Throughout this section, for each integer $i \in \{1, \ldots, r\}$, the phrase "*at time i*" means the time at which the cycle C_{i-1} has just been processed (in Step $5'$) and the processing of C_i (in Step $5'$) is about to begin.

Fix an $i \in \{1, \ldots, r\}$ and an $h \in \{1, \ldots, 5\}$. An *N_h-dependent vertex at time i* is a vertex v of C_i such that at time i, N_h contains an edge $\{x, y\}$ with $(v, x) \in M'$ or $(v, y) \in M'$. An *N_h-available set at time i* is a subset Z of $E(C_i)$ such that $Z \cap (M \cup N_h) = \emptyset$ and the graph $(V(G), M \cup N_h \cup Z)$ is a subtour of G. A *maximal N_h-available set at time i* is an N_h-available set Z at time i such that for each $e \in E(C_i) - Z$, $Z \cup \{e\}$ is not N_h-available at time i.

Lemma 3.5 *Let Z be an N_h-available set at time i. Suppose that $e_1 = \{u_1, u_2\}$ and $e_2 = \{u_2, u_3\}$ are two adjacent edges in C_i such that no edge in Z is incident to u_1, u_2, or u_3. Then, $Z \cup \{e_1\}$ or $Z \cup \{e_2\}$ is an N_h-available set at time i.*

PROOF. Since Z contains no edge incident to u_1, u_2, or u_3, the degree of each $u_j \in \{u_1, u_2, u_3\}$ in the subtour $(V(G), M \cup N_h \cup Z)$ is at most 1. So, if $Z \cup \{e_1\}$ is not an N_h-available set at time i, then u_1 and u_2 belong to the same connected component (a path) of the subtour $(V(G), M \cup N_h \cup Z)$ and hence u_2 and u_3 belong to different connected components (paths) of the subtour, implying that $Z \cup \{e_2\}$ is an N_h-available set at time i. □

The following corollary is immediate from Lemma 3.5:

Corollary 3.6 *If Z is a maximal N_h-available set at time i, then each connected component of $C_i - Z$ is a path of length at most 3.*

Now, we are ready to describe how to compute $A_1, \ldots, A_5$ when processing C_i. In detail, for each $h \in \{1, \ldots, 5\}$, A_h is computed by performing the three steps in Figure 4 in turn.

(i) Compute two nonempty subsets X_h and Y_h of $E(C_i) - M$ such that (1) both X_h and Y_h are N_h-available sets at time i and (2) each vertex of C_i is an endpoint of at least one edge in $X_h \cup Y_h$. (Comment: By Lemma 1 in [5], this step can be done in linear time.)

(ii) Let a_h (respectively, b_h) be the total weight of edges $(u, v) \in M'$ such that u is an N_h-dependent vertex at time i and X_h (respectively, Y_h) contains an edge incident to u. If $a_h \geq b_h$, let $Z_h = X_h$; otherwise, let $Z_h = Y_h$.

(iii) Extend Z_h to a maximal N_h-available set A_h at time i.

Figure 4. Computation of A_h.

A *C_i-settled edge* is an edge $(u, v) \in M'$ such that u is a vertex of C_i (and so v is a vertex of some C_j with $j < i$). The following lemma is immediate from Condition (C3) and the above computation of $A_1, \ldots, A_5$, and ensures Condition (C4).

Lemma 3.7 *For each $h \in \{1, \ldots, 5\}$, we say that a C_i-settled edge $e = (u, v)$ is* A_h-satisfiable at time i *if u is an N_h-dependent vertex at time i and A_h contains an edge of C_i incident to u. We say that a C_i-settled edge e is* satisfiable at time i *if there is at least one $h \in \{1, \ldots, 5\}$ such that e is A_h-satisfiable at time i. Let M_i' be the set of all C_i-settled edges, and let S_i be the set of all satisfiable C_i-settled edges at time i. Then, $w(S_i) \geq w(M_i')/2$.*

Unfortunately, the sets $A_1, \ldots, A_5$ computed as in Figure 4 may not satisfy Condition (C3). In other words, although the connected components of $C_i - (\bigcup_{1 \leq j \leq 5} A_j)$ are paths of length at most 3 by Corollary 3.6, some of them are indeed of length 2 or 3. If this bad case happens, we need to modify $A_1, \ldots, A_5$ so that Conditions (C1) through (C4) and Lemma 3.7 hold. The modification of $A_1, \ldots, A_5$ is done by considering three cases as follows:

Case 1: Some connected component P of $C_i - (\bigcup_{1 \leq j \leq 5} A_j)$ is a path of length 3. Let the edges of P be $e_1 = \{u_1, u_2\}$, $e_2 = \{u_2, u_3\}$, and $e_3 = \{u_3, u_4\}$. Let $e_0 = \{u_0, u_1\}$ be the edge in $E(C_i) - \{e_1\}$ incident to u_1. By Corollary 3.6, $e_0 \in \cap_{1 \leq j \leq 5} A_j$. For each $u_j \in \{u_0, u_1\}$, if there is an $h \in \{1, \ldots, 5\}$ such that u_j is an endpoint of an A_h-satisfiable C_i-settled edge at time i, then let $f(u_j)$ be such an h; otherwise, let $f(u_j)$ be an arbitrary integer in $\{1, \ldots, 5\}$. Possibly, $f(u_0) = f(u_1)$. In any case, let h be an arbitrary integer in $\{1, \ldots, 5\} - \{f(u_0), f(u_1)\}$. Then, by our choice of $f(u_0)$ and $f(u_1)$, the deletion of e_0 from A_h does not cause any originally satisfiable C_i-settled edge at time i to be no longer satisfiable at time i. Moreover, by Lemma 3.5, there is an $e_k \in \{e_1, e_2\}$ such that $(A_h - \{e_0\}) \cup \{e_k\}$ is an N_h-available set at time i. So, we delete e_0 from A_h, and then add e_k and zero or more other edges to A_h so that A_h becomes a maximal N_h-available set at time i. Clearly, after this modification of A_h, $C_i - (\bigcup_{1 \leq j \leq 5} A_j)$ has fewer edges than before. Therefore, we can repeat this kind of modification until no connected component of $C_i - (\bigcup_{1 \leq j \leq 5} A_j)$ is a path of length 3.

Case 2: Some connected component P of $C_i - (\bigcup_{1 \leq j \leq 5} A_j)$ is a path of length 2 and $|E(C_i)| = 3$. Let the edges of P be $e_1 = \{u_1, u_2\}$ and $e_2 = \{u_2, u_3\}$. Let $e_0 = \{u_0, u_1\}$ be the edge in $E(C_i) - \{e_1\}$ incident to u_1. Then, as in Case 1, we can find an integer $h \in \{1, \ldots, 5\}$ such that the deletion of e_0 from A_h does not cause any originally satisfiable C_i-settled edge at time i to be no longer satisfiable at time i. Moreover, by Lemma 3.5, there is an $e_k \in \{e_1, e_2\}$ such that $(A_h - \{e_0\}) \cup \{e_k\}$ is an N_h-available set at time i. So, we can modify A_h as in Case 1.

Case 3: Some connected component P of $C_i - (\bigcup_{1 \leq j \leq 5} A_j)$ is a path of length 2 and $|E(C_i)| \geq 4$. This case is the bottleneck case. Let the edges of P be $e_1 = \{u_1, u_2\}$ and $e_2 = \{u_2, u_3\}$. Let the two edges in $E(C_i) - E(P)$ incident to an endpoint of P be $e_0 = \{u_0, u_1\}$ and $e_3 = \{u_3, u_4\}$. Note that if $|E(C_i)| = 4$, then $u_0 = u_4$. By Corollary 3.6, $\{e_0, e_3\} \cap A_h \neq \emptyset$ for each $h \in \{1, \ldots, 5\}$. For each $u_j \in \{u_0, u_1, u_3, u_4\}$, if there is an $h \in \{1, \ldots, 5\}$ such that u_j is an endpoint of an A_h-satisfiable C_i-settled edge at time i, then let $f(u_j)$ be such an h; otherwise, let $f(u_j)$ be an arbitrary integer in $\{1, \ldots, 5\}$. Let h be an arbitrary integer in $\{1, \ldots, 5\} - \{f(u_0), f(u_1), f(u_3), f(u_4)\}$. Obviously, even if we delete both e_0 and e_3 from A_h, every satisfiable C_i-settled edge at time i remains to be satisfiable at time i. Moreover, by Lemma 3.5, there is an $e_k \in \{e_1, e_2\}$ such that $(A_h - \{e_0, e_3\}) \cup \{e_k\}$ is an N_h-available set at time i. So, we delete both e_0 and e_3 from A_h, and then add e_k and zero or more other edges to A_h so that A_h becomes a maximal N_h-available set at time i. Clearly, after this modification of A_h, $C_i - (\bigcup_{1 \le j \le 5} A_j)$ has fewer edges than before.

So, as long as some connected component of $C_i - (\bigcup_{1 \le j \le 5} A_j)$ is a path of length 3, we keep applying the modification in Case 1 to $A_1, \ldots, A_5$. Once no connected component of $C_i - (\bigcup_{1 \le j \le 5} A_j)$ is a path of length 3, we keep applying the modification in Case 2 or 3 to $A_1, \ldots, A_5$ until no connected component of $C_i - (\bigcup_{1 \le j \le 5} A_j)$ is a path of length 2. After this, each connected component of $C_i - (\bigcup_{1 \le j \le 5} A_j)$ is a path of length 0 or 1, and hence $A_1, \ldots, A_5$ satisfy Conditions (C1) through (C4) and Lemma 3.7 hold.

4 Algorithm for Metric Max TSP

This section is divided into three subsections. In Section 4.1, we sketch H&R2-algorithm. In Section 4.2, we describe our derandomization of H&R2-algoirthm and analyze its approximation ratio. In Section 4.3, we give the details that are omitted in Section 4.2.

Let (G, w) be as in Section 2. Here, w satisfies the following triangle inequality: For every three vertices x, y, z of G, $w(x, y) \le w(x, z) + w(z, y)$. Then, we have the following useful fact:

Fact 4.1 *Suppose that $P_1, \ldots, P_t$ are vertex-disjoint paths in G each containing at least one edge. For each $1 \le i \le t$, let u_i and v_i be the endpoints of P_i. Then, we can use some edges of G to connect $P_1, \ldots, P_t$ into a single cycle C in linear time such that $w(C) \ge \sum_{i=1}^{t} w(P_i) + \frac{1}{2}\sum_{i=1}^{t} w(\{u_i, v_i\})$.*

Proof. The fact is obvious if $t \le 2$. So, assume $t \ge 3$. Define four disjoint sets of edges $E_1, \ldots, E_4$ as follows. E_1 consists of $\{u_1, v_t\}$ and all $\{v_i, u_{i+1}\}$ with $1 \le i \le t-1$. E_2 consists of $\{v_1, u_t\}$ and all $\{u_i, v_{i+1}\}$ with $1 \le i \le t-1$. If t is even, E_3 consists of $\{v_1, v_t\}$, all $\{u_{2i-1}, u_{2i}\}$ with $1 \le i \le t/2$, and all $\{v_{2i}, v_{2i+1}\}$ with $1 \le i \le (t-2)/2$; otherwise, E_3 consists of $\{v_1, u_t\}$, all $\{u_{2i-1}, u_{2i}\}$ with $1 \le i \le (t-1)/2$, and all $\{v_{2i}, v_{2i+1}\}$ with $1 \le i \le (t-1)/2$. If t is even, E_4 consists of $\{u_1, u_t\}$, all $\{v_{2i-1}, v_{2i}\}$ with $1 \le i \le t/2$, and all $\{u_{2i}, u_{2i+1}\}$ with $1 \le i \le (t-2)/2$; otherwise, E_4 consists of $\{u_1, v_t\}$, all $\{v_{2i-1}, v_{2i}\}$ with $1 \le i \le (t-1)/2$, and all $\{u_{2i}, u_{2i+1}\}$ with $1 \le i \le (t-1)/2$. Clearly, for each $1 \le h \le 4$, the edges in E_h can be used to connect $P_1, \ldots, P_t$ into a single cycle. Moreover, $\sum_{h=1}^{4} w(E_h) = \sum_{i=1}^{t}(w(u_i, u_{i+1}) + w(v_i, v_{i+1}) + w(u_i, v_{i+1}) + w(v_i, u_{i+1}))$, where $u_{t+1} = u_0$ and $v_{t+1} = v_0$. So, by the triangle inequality, $\sum_{h=1}^{4} w(E_h) \ge 2\sum_{i=1}^{t} w(u_i, v_i)$. Now, consider the E_h such that $w(E_h)$ is the maximum among $w(E_1), \ldots, w(E_4)$. Let C be the cycle obtained by using the edges in E_h to connect $P_1, \ldots, P_t$ together. Then, $w(C) \ge \sum_{i=1}^{t} w(P_i) + \frac{1}{2}\sum_{i=1}^{t} w(\{u_i, v_i\})$. $\square$

4.1 Sketch of H&R2-algorithm

H&R2-algorithm assumes that n is even. It starts by computing a maximum-weight cycle cover $\mathcal{C}$. If $\mathcal{C}$ is a tour of G, then we are done. Throughout the rest of this section, we assume that $\mathcal{C}$ is not a tour of G. H&R2-algorithm then computes two tours T_1, T_2 of G and outputs the heavier one between them.

1. Compute a maximum-weight matching M in G. (Comment: Since n is even, M is perfect.)
2. Let $C_1, \ldots, C_r$ be an arbitrary ordering of the cycles in $\mathcal{C}$.
3. Initialize a set N to be empty.
4. For $i = 1, 2, \ldots, r$ (in this order), perform the following two steps:
   (a) Compute two distinct edges e_1 and e_2 in C_i such that both graphs $(V(G), M \cup N \cup \{e_1\})$ and $(V(G), M \cup N \cup \{e_2\})$ are subtours of G.
   (b) Select $h \in \{1, 2\}$ uniformly at random, and add edge e_h to N.
5. Complete the graph $\mathcal{C} - N$ to a tour T_1 of G by *suitably* choosing and adding some edges of G. (Comment: Randomness is needed in this step.)
6. Let S be the set of vertices v in G such that the degree of v in the graph $(V(G), M \cup N)$ is 1. (Comment: $|S|$ is even because each connected component in the graph $(V(G), M \cup N)$ is a path of length at least 1.)
7. Compute a random perfect matching M_S in the subgraph (S, F) of G, where F consists of all edges $\{u, v\}$ of G with $\{u, v\} \subseteq S$.
8. Let G' be the multigraph obtained from the graph $(V(G), M \cup N)$ by adding every edge $e \in M_S$ even if $e \in M \cup N$. (Comment: Each connected component of G' is either a path of length 1 or more, or a cycle of length 2 or more. The crucial point is that for each edge e in G', the probability that the connected component of G' containing e is a cycle of size smaller than $\sqrt{n}$ is at most $O(1/\sqrt{n})$.)
9. For each cycle C in G', select one edge in C uniformly at random and delete it from G'.
10. Complete G' to a tour T_2 of G by adding some edges of G.

Figure 5. Computation of tours T_1 and T_2 in H&R2-algorithm.

4.2 Derandomization of H&R2-algorithm

Only Steps 4, 5, 7, and 9 in Figure 5 need randomness. Step 5 can be derandomized using Fact 4.1. Step 9 is similar to Step 8 in Figure 1 and can be derandomized similarly. However, Steps 4 and 7 are hard to derandomize. Our main contribution is a derandomization of Step 4 (without making the approximation ratio worse). However, we do not know how to derandomize Step 7 without making the approximation ratio worse; so we simply let M_S be a maximum-weight matching in the subgraph (S, F) of G. This makes the approximation ratio slightly worse.

Unlike H&R2-algorithm, we assume that n is odd (the case where n is even is simpler). Then, the matching M computed in Step 1 in Figure 5 is not perfect. Let z be the vertex in G to which no edge in M is incident. Let e_z and e'_z be the two edges incident to z in $\mathcal{C}$.

In detail, to derandomize H&R2-algorithm, we replace Steps 4 through 10 in Figure 5 by the eight steps in Figure 6.

4′. Compute two disjoint subsets A_1 and A_2 of $E(\mathcal{C}) - M$ satisfying the following three conditions:

(C5) Both graphs $(V(G), M \cup A_1)$ and $(V(G), M \cup A_2)$ are subtours of G.

(C6) For each $i \in \{1, \ldots, r\}$, $|E(C_i) \cap A_1| = 1$ and $|E(C_i) \cap A_2| = 1$.

(C7) $e_z \in A_1$ and $e'_z \in A_2$.

5′. Choose N from A_1 and A_2 uniformly at random. (Comment: This step needs one random bit. For a technical reason, we allow our algorithm to use only one random bit; so we can easily derandomize it, although we omit the details.)

6′. Complete the graph $\mathcal{C} - N$ to a tour T_1 of G as described in Fact 4.1. (Comment: Immediately before this step, each connected component of $\mathcal{C} - N$ is a path of length at least 1 because of Condition (C6).)

7′. Same as Step 6 in Figure 5. (Comment: The assertion in the comment on Step 6 in Figure 5 still holds here too because of Condition (C7).)

8′. Compute a maximum-weight matching M_S in the graph (S, F), where F is as in Step 7 in Figure 5. (Comment: M_S is perfect.)

9′. Same as Step 8 in Figure 5. (Comment: The first assertion in the comment on Step 8 in Figure 5 holds here too, but the second assertion does not hold here.)

10′. Compute a subset M'_S of M_S such that each cycle in G' contains exactly one edge of M'_S.

11′. Complete the graph $G' - M'_S$ to a tour T_2 of G as described in Fact 4.1.

Figure 6. Modifying Steps 4 through 10 in Figure 5.

The details of computing A_1 and A_2 will be given in Section 4.3. In the next theorem, we analyze the approximation ratio of our new algorithm.

Theorem 4.2 *There is an $O(n^3)$-time approximation algorithm for metric Max TSP achieving an approximation ratio of $\frac{17}{20} - \frac{1}{5n}$.*

PROOF. It suffices to prove that $\max\{\mathcal{E}[w(T_1)], \mathcal{E}[w(T_2)]\} \geq (\frac{17}{20} - \frac{1}{5n})opt$, where *opt* is the weight of an optimal tour in G. Let $\alpha = w(A_1 \cup A_2)/w(\mathcal{C})$. By Fact 4.1, $\mathcal{E}[w(T_1)] \geq (1 - \frac{\alpha}{2} + \frac{\alpha}{4})w(\mathcal{C}) \geq (1 - \frac{\alpha}{4})opt$.

Since the graph (S, F) is a complete graph and $|S|$ is even, F can be partitioned into $|S| - 1$ perfect matchings of the graph. So, $w(M_S) \geq \frac{1}{|S|-1} w(F)$. We need to compare $\mathcal{E}[w(F)]$ with $w(\mathcal{C}) - w(A_1 \cup A_2)$. To this end, we use an idea in [6], i.e., we charge the weight of each edge $e = \{u, v\} \in F$ to edges in $\mathcal{C}$ but not in $A_1 \cup A_2$ as follows. We call the edges in $A_1 \cup A_2$ *candidates*. For each $x \in \{u, v\}$, if x is incident to no candidate edge in $\mathcal{C}$, then we charge $w(e)/4$ to each edge incident to x in $\mathcal{C}$; otherwise, exactly one of the two edges incident to x in $\mathcal{C}$ is a candidate (because $x \in S$), and we charge $w(e)/2$ to the non-candidate edge incident to x in $\mathcal{C}$. As observed in [6], the expected total weight charged to a non-candidate edge $\{y_1, y_2\}$ of $\mathcal{C}$ is $\sum_{x \in S - \{y_1\}} w(x, y_1)/4 + \sum_{x \in S - \{y_2\}} w(x, y_2)/4$; so, it is at least $(|S| - 1)w(y_1, y_2)/4$ by the triangle inequality. Thus, $\mathcal{E}[w(F)] \geq (|S| - 1)\sum_e w(e)/4$, where e ranges over all non-candidate edges of $\mathcal{C}$. Therefore, $\mathcal{E}[w(M_S)] \geq \frac{1}{|S|-1}\mathcal{E}[w(F)] \geq (1 - \alpha)w(\mathcal{C})/4$.

By Fact 4.1, $\mathcal{E}[w(T_2)] \geq \mathcal{E}[w(M \cup N)] + \mathcal{E}[w(M_S)] - \mathcal{E}[w(M'_S)] + \frac{1}{2}\mathcal{E}[w(M'_S)]$. Since $w(M'_S) \leq w(M_S)$, we now have $\mathcal{E}[w(T_2)] \geq (\frac{1}{2} - \frac{1}{2n})opt + \frac{\alpha}{2}w(\mathcal{C}) + \frac{1}{2}\mathcal{E}[w(M_S)]$. Hence, by the last inequality in the previous paragraph, $\mathcal{E}[w(T_2)] \geq (\frac{5}{8} + \frac{3}{8}\alpha - \frac{1}{2n})opt$. Combining this with the inequality $\mathcal{E}[w(T_1)] \geq (1 - \frac{\alpha}{4})opt$, we finally have $\max\{\mathcal{E}[w(T_1)], \mathcal{E}[w(T_2)]\} \geq (\frac{17}{20} - \frac{1}{5n})opt$.

The running time of the algorithm is dominated by the $O(n^3)$-time needed for computing a maximum-weight cycle cover and two maximum-weight matchings. □

Since maximum-weight cycle covers, maximum-weight matchings, and maximal path sets can be computed by fast parallel algorithms [7, 9, 2], our algorithm for metric Max TSP is parallelizable. Indeed, using our idea of computing the sets A_1 and A_2, we can even parallelize H&R2-algorithm. We omit the details here.

4.3 Computation of A_1 and A_2

We now detail the computation of A_1 and A_2. We need two definitions first. A *path set* in a graph $H = (V_H, E_H)$ is a subset Q of E_H such that the graph (V_H, Q) is a collection of vertex-disjoint paths. A path set Q in H is *maximal* if for every $e \in E_H - Q$, $Q \cup \{e\}$ is not a path set in H.

Obviously, both $M \cup \{e_z\}$ and $M \cup \{e'_z\}$ are path sets in G. Now, A_1 is computed as in Figure 7.

(i) Compute a maximal path set Q_1 with $M \cup \{e_z\} \subseteq Q_1$ in H_1, where H_1 is the graph obtained from $\mathcal{C}$ by adding the edges in M. (Comment: The degree of each vertex in H_1 is at most 3.)

(ii) Set $A_1 = Q_1 - M$.

(iii) For each cycle C_i in $\mathcal{C}$ such that A_1 contains two or more edges of C_i, keep only one edge of C_i in A_1 (the others are deleted from A_1). (Note: In this step, we keep e_z in A_1.)

Figure 7. Computation of A_1.

The following lemma together with the computation of A_1 in Figure 7 ensures that Conditions (C5) through (C7) in Figure 6 hold for A_1.

Lemma 4.3 *Immediately after Step (ii) in Figure 7, A_1 contains at least one edge of C_i for each cycle C_i in $\mathcal{C}$.*

Proof. Consider an arbitrary cycle C_i in $\mathcal{C}$. Let $P_1, \ldots, P_s$ be the connected components of the graph $(V(G), Q_1)$. Each P_j ($1 \leq j \leq s$) is a path. For a contradiction, assume that A_1 contains no edge of C_i. Then, each vertex of C_i is an endpoint of some P_j ($1 \leq j \leq s$). Let $e_1 = \{u_1, u_2\}$ and $e_2 = \{u_2, u_3\}$ be two adjacent edges in C_i. Because $e_1 \notin A_1$ and Q_1 is maximal, either $e_1 \in M$ or $Q_1 \cup \{e_1\}$ is not a path set. In both cases, u_1 and u_2 must be the endpoints of the same path P_j for some $j \in \{1, \ldots, s\}$. Thus, u_3 is an endpoint of another path P_k with $k \neq j$. So, $e_2 \notin M$ and $Q_1 \cup \{e_2\}$ is a path set, a contradiction against the maximality of Q_1. □

Recall that $M \cup \{e'_z\}$ is a path set in G. Now, A_2 is computed as in Figure 8.

(iv) Compute a maximal path set Q_2 with $M \cup \{e'_z\} \subseteq Q_2$ in H_2, where H_2 is the graph obtained from $\mathcal{C}$ by first deleting the edges in A_1 and then adding the edges in M. (Comment: The degree of each vertex in H_2 is at most 3.)

(v) Set $A_2 = Q_2 - M$.

(vi) For each cycle C_i in $\mathcal{C}$ such that A_2 contains two or more edges of C_i, keep only one edge of C_i in A_2 (the others are deleted from A_2). (Note: In this step, we keep e'_z in A_2.)

Figure 8. Computation of A_2.

The following lemma together with the computation of A_2 in Figure 8 ensures that Conditions (C5) through (C7) in Figure 6 hold for A_2.

Lemma 4.4 *For each $C_i \in \mathcal{C}$, A_2 contains one edge of C_i.*

PROOF. For each cycle C_i in $\mathcal{C}$, since exactly one edge of C_i is contained in A_1, there are two adjacent edges in C_i that are also edges in H_2. So, the proof of Lemma 4.3 is still valid here if we replace each occurence of "Q_1" there by "Q_2" and replace each occurence of "A_1" there by "A_2". $\square$

Acknowledgments

The authors thank the referee for helpful comments.

References

[1] A. I. Barvinok, D. S. Johnson, G. J. Woeginger, and R. Woodroofe. Finding Maximum Length Tours under Polyhedral Norms. *Proceedings of the Sixth International Conference on Integer Programming and Combinatorial Optimization* (IPCO), Lecture Notes in Computer Science, **1412** (1998) 195–201.

[2] Z.-Z. Chen. Parallel Constructions of Maximal Path Sets and Applications to Short Superstrings. *Theoretical Computer Science*, **161** (1996) 1–21.

[3] Z.-Z. Chen and L. Wang. An Improved Randomized Approximation Algorithm for Max TSP. *Submitted*.

[4] R. Hassin and S. Rubinstein. An Approximation Algorithm for the Maximum Traveling Salesman Problem. *Information Processing Letters*, **67** (1998) 125–130.

[5] R. Hassin and S. Rubinstein. Better Approximations for Max TSP. *Information Processing Letters*, **75** (2000) 181–186.

[6] R. Hassin and S. Rubinstein. A 7/8-Approximation Approximations for Metric Max TSP. *Information Processing Letters*, **81** (2002) 247–251.

[7] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a Perfect Matching is in random NC. *Combinatorica*, **6** (1986) 35-48.

[8] 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 Maximum (in Russian). *Upravlyaemye Sistemy*, **26** (1985) 55–59.

[9] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, **7** (1987) 105–113.

[10] A. I. Serdyukov. An Algorithm with an Estimate for the Traveling Salesman Problem of Maximum (in Russian). *Upravlyaemye Sistemy*, **25** (1984) 80–86.