Title
Improved Deterministic Approximation Algorithms for Max TSP (Theoretical Computer Science and its Applications)

Author(s)
Chen, Zhi-Zhong; Okamoto, Yuusuke; Wang, Lusheng

Citation
数理解析研究所講究録 (2005), 1426: 1-6

Issue Date
2005-04

URL
http://hdl.handle.net/2433/47256

Type
Departmental Bulletin Paper

Textversion
publisher
Kyoto University
Improved Deterministic Approximation Algorithms for Max TSP

Zhi-Zhong Chen (陳致中)  
Department of Mathematical Sciences  
Tokyo Denki University  
Hatoyama, Saitama 350-0394, Japan.  
(東京電機大學理工學部數理科學科)

Yuusuke Okamoto (岡本裕介)  
Department of Mathematical Sciences  
Tokyo Denki University  
Hatoyama, Saitama 350-0394, Japan.  
(東京電機大學理工學部數理科學科)

Lusheng Wang (王魯生)  
Department of Computer Science  
City University of Hong Kong  
Tat Chee Avenue, Kowloon, Hong Kong.  
(香港城市大學計算機系)

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 $\frac{4}{5}$. 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.

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 $\frac{3}{2}$. Their algorithm is very simple and elegant. Tempted by improving the ratio $\frac{3}{2}$, 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 $\frac{17}{10}$. 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 $\frac{17}{10}$, an improvement over the previous best ratio (namely, $\frac{3}{2}$). 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 $X$, $E[X]$ denotes the expected value of $X$.

3 Sketch of H&R-algorithm

3.1 Sketch of H&R-algorithm

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

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 $C$ 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** $w(T_1) \geq (1 - \epsilon)w(T_{\text{int}}) = (1 - \epsilon)\omega(T)$.

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 follows:

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 $C - N$ by adding the edges in $M''$. (Comment: For each edge $\{u, v\} \in M'$, $\Pr[\{u, v\} \in M''] \geq \frac{1}{4}$. So, $E[w(M'')] \geq w(M'')/4$. 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 $E[w(M' \cap E(G''))] \geq w(M')/8$.)
9. Complete $G''$ to a tour $T_3$ of $G$ by adding some edges of $G$.

### 3.2 Derandomization of H&R-algorithm

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 $i > j$. To derandomize Step 5 in H&R-algorithm, we replace Steps 4 and 5 in H&R-algorithm by the following four steps:

- **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 $\bigcup_{1 \leq j \leq 5} A_j$.
     - (C4) $w(S_i) \geq w(M'_i)/2$, where $M'_i$ is the set of all edges $\{u, v\} \in M'$ such that $u \in V(C_i)$ and $v \in \bigcup_{1 \leq j \leq i-1} V(C_j)$, 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), $w(\bigcup_{1\leq h\leq 5} M''_h) \geq w(M')/2$.)

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', $w(M''_h) \geq w(M')/10$.)

The details of computing $A_1, \ldots, A_5$ are complicated and are omitted here for lack of space.

**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 $C$.

To derandomize Step 8 in H&R-algorithm, it suffices to replace it by the following two steps:

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

**Lemma 3.3** Let $\delta w(T)$ be the total weight of edges in $N$. 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 $w(T_3) \geq (1 - \delta + \frac{1}{40} - \frac{1}{40\alpha})w(T)$.

**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 $(61 - \frac{20}{n}) \cdot \frac{1}{81 - 80\epsilon}$.

## 4 Algorithm for Metric Max TSP

Let $(G, w)$ be as in Section 2. Here, $w$ satisfies the following triangle inequality: For every three vertices $x, y, z$ of $G$, $w(x, y) \leq w(x, z) + w(z, y)$. 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 $1 \leq i \leq t$, 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 $w(C) \geq \sum_{i=1}^t w(P_i) + \frac{1}{2} \sum_{i=1}^t w(\{u_i, v_i\})$.

### 4.1 Sketch of H&R2-algorithm

H&R2-algorithm assumes that $n$ is even. It starts by computing a maximum-weight cycle cover $C$. If $C$ is a tour of $G$, then we are done. Throughout the rest of this section, we assume that $C$ 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 as follows:

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 $C$.
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 $C - N$ 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 $(V(G), M \cup A)$ 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, 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 $G$.

### 4.2 Derandomization of H&R2-algorithm

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 H&R2-algorithm 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 $C$.

To derandomize H&R2-algorithm, we replace Steps 4 through 10 in H&R2-algorithm by the following eight steps:

- **4’.** Compute two disjoint subsets $A_1$ and $A_2$ of $E(C) - M$ 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 $i \in \{1, \ldots, r\}$, $E(C_i) \cap A_1 \neq \emptyset$ and $E(C_i) \cap A_2 \neq \emptyset$.
  - $(C7) e_z \in A_1$ 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’.** Compute the graph $C - N$ to a tour $T_1$ of $G$ as described in Fact 4.1. (Comment: Immediately before this step, each connected component of $C - N$ is a path of length at least 1 because of Conditions $(C6)$ and $(C7)$.)

- **7’.** Same as Step 6 in H&R2-algorithm.

- **8’.** Compute a maximum-weight matching $M_S$ in the graph $(S, F)$, where $F$ is as in Step 7 in H&R2-algorithm. (Comment: $M_S$ is perfect.)

- **9’.** Same as Step 8 in H&R2-algorithm. (Comment: The first assertion in the comment on Step 8 in H&R2-algorithm 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.

The details of computing $A_1$ and $A_2$ is omitted here for lack of space.

**Theorem 4.2** There is an $O(n^3)$-time approximation algorithm for metric Max TSP achieving an approximation ratio of $\frac{17}{10} - \frac{1}{5n}$.
References


