Information Processing Letters 75 (2000) 181–186

Information Processing Letters

www.elsevier.com/locate/ipl

Better approximations for max TSP

Refael Hassin [*,1], Shlomi Rubinstein

Department of Statistics and Operations Research, School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel

Received 3 December 1998
Communicated by S. Zaks

Abstract

We combine two known polynomial time approximation algorithms for the maximum traveling salesman problem to obtain a randomized algorithm which outputs a solution with expected value of at least r times the optimal one for any given $r < 25/33$.
© 2000 Elsevier Science B.V. All rights reserved.

Keywords: Analysis of algorithms; Maximum traveling salesman; Maximum latency TSP

1. Introduction

Let $G = (V, E)$ be a complete (undirected) graph with vertex set V and edge set E. For $e \in E$ let $w(e) \geqslant 0$ be its weight. For $E' \subseteq E$ we denote $w(E') = \sum_{e \in E'} w(e)$. For a random subset $E' \subseteq E$, $w(E')$ denotes the expected value. The *maximum traveling salesman problem* (Max TSP) is to compute a Hamiltonian circuit (a *tour*) with maximum total edge weight. The problem is max-SNP-hard [1] and therefore there exists some constant $\beta < 1$ such that obtaining a solution with performance guarantee better than β is NP-hard.

We denote the weight of an optimal tour by *opt*. In [3] we described a polynomial algorithm that guarantees for any $r < 5/7$ a solution of weight at least $r\,opt$. We were then informed by Alexander Ageev that a paper by Anatoly Serdyukov [5] already contains an approximation algorithm with a better performance guarantee of $\frac{3}{4}\,opt$ (and another paper [4] with even better bounds for the metric case).

We first describe Serdyukov's algorithm. The algorithm is very simple and elegant and it is given in the next section. We then combine ideas from [5] and [3] to form a randomized polynomial algorithm that computes a tour of expected weight at least $r\,opt$ for any given $r < 25/33$. While the improvement is small, it at least demonstrates that the bound of $3/4$ can be improved and that further research in this direction is encouraged. This algorithm is described in Section 3. Finally, in Section 4 we apply these results to obtain new approximation results for the *maximum latency TSP*.

2. Serdyukov's algorithm

A *cycle cover*, or *binary* 2-*matching*, is a subgraph in which each vertex in V has a degree of exactly 2. A *subtour* is a set of edges that can be completed to a tour (i.e., contains no non-Hamiltonian cycles and no vertex of degree greater than 2). A *maximum*

[*] Corresponding author.
E-mail addresses: hassin@math.tau.ac.il (R. Hassin), shlom@math.tau.ac.il (S. Rubinstein).
[1] This paper was written while R. Hassin visited GSIA, Carnegie Mellon University.

PII: S0020-0190(00)00097-1

```
Serdyukov's Algorithm
  input
  1. A complete undirected graph $G = (V, E)$ with weights $w_e$ $e \in E$.
  returns A tour.
  begin
  Compute a maximum cycle cover $\mathcal{C} = \{C_1, \ldots, C_r\}$.
  Compute a maximum matching $W$.
  for $i = 1, \ldots, r$:
      Transfer from $C_i$ to $W$ an edge so that $W$ remains a subtour.
      end for
  Complete $\mathcal{C}$ into a tour $T_1$.
  Complete $W$ into a tour $T_2$.
  return the tour with maximum weight between $T_1$ and $T_2$.
  end Serdyukov's Algorithm
```

Fig. 1. Serdyukov's algorithm.

cycle cover is one with maximum total edge weight. A *maximum matching* is a set of vertex-disjoint edges of maximum total weight. Serdyukov's algorithm for the case in which $|V|$ is even is given in Fig. 1.

Note that it is always possible to transfer an edge from C_i to W as required. The performance guarantee follows easily using the assumption that $|V|$ is even. The weight of the cycle cover is an upper bound on *opt* while that of the matching is at least $\frac{1}{2}\,opt$ if $|V|$ is even. Thus, $w(T_1) + w(T_2) \geqslant \frac{3}{2}\,opt$ and $\max\{w(T_1), w(T_2)\} \geqslant \frac{3}{4}\,opt$. Serdyukov also shows how to modify the algorithm so that the bound holds when $|V|$ is odd but this part is more involved and we are interested here only in asymptotic bounds so that the parity of $|V|$ is not important.

3. A new algorithm

Algorithm *Max_TSP* is given in Fig. 2. It constructs three tours and selects the one with greater weight.

The first tour is constructed, as in [3], by Algorithm $A1$ (see Fig. 3). It uses a parameter $\varepsilon > 0$. It treats differently *short cycles*, such that $|C_i| \leqslant \varepsilon^{-1}$, and *long cycles*. For each short cycle it computes a maximum Hamiltonian path on its vertices. For each long cycle it deletes an edge of minimum length. The resulting path cover is extended to a tour T_1.

The second algorithm (see Fig. 4) is a modified version of Serdyukov's algorithm. It transfers edges from $\mathcal{C}$ to W using a randomized selection step, and generates two subtours. The one formed from W with

```
Max_TSP
  input
  1. A complete undirected graph $G = (V, E)$ with weights $w_e$ $e \in E$.
  2. A constant $\varepsilon > 0$.
  returns A tour $T$.
  begin
  Compute a maximum cycle cover $\mathcal{C} = \{C_1, \ldots, C_r\}$.
  $T_1 := A1(G, \mathcal{C}, \varepsilon)$.
  $(T_2, T_3) := A2(G, \mathcal{C})$.
  return the tour with the maximum weight among $T_1, T_2$ and $T_3$.
  end Max_TSP
```

Fig. 2. Algorithm *Max_TSP*.

```
A1
  input
  1. A complete undirected graph $G = (V, E)$ with weights $w_e$ $e \in E$.
  2. A cycle cover $\mathcal{C}$.
  3. A constant $\varepsilon > 0$.
  returns A tour $T_1$.
  begin
  for $i = 1, \ldots, r$:
     if $|C_i| \leqslant \varepsilon^{-1}$
       then
           Compute a maximum Hamiltonian path $H_i$ in the
           subgraph induced by the vertices of $C_i$.
       else
           Let $e_i$ be a minimum weight edge of $C_i$.
           $H_i := C_i \setminus \{e_i\}$.
       end if
     end for
  Connect $H_1, \ldots, H_r$ in some arbitrary order to form a tour $T_1$.
  return $T_1$.
  end A1
```

Fig. 3. Algorithm $A1$.

the transferred edges is augmented arbitrarily to a tour T_2. The other one, consisting of the remaining edges of $\mathcal{C}$, is first augmented by new edges whose two ends belong to different cycles of $\mathcal{C}$. Then it is arbitrarily augmented to a tour T_3.

Lemma 1. *When Algorithm $A2$ treats C_i, it is possible to construct the desired matchings M_i and M_i' such that both matchings are nonempty, $M_i \cup W$ and $M_i' \cup W$ are subtours, and each vertex of C_i is an end vertex of at least one edge from $M_i \cup M_i'$.*

Proof. Denote the edges of C_i by $e_1, \ldots, e_k$ in cyclic order, starting from an arbitrary edge.

Follow C_i starting from e_1. Alternately insert edges of C_i to M_i and M_i'. If such an insertion (say of e_j to M_i) would create a cycle in $M_i \cup W$ (in particular, if this edge is already in W) then skip e_j and assign instead the next edge, e_{j+1} to M_i. We observe that the latter assignment is always possible, and we never skip two successive edges.

Care must be taken with respect to the last assignment. First, there may be a conflict if we assigned both e_1 and e_k to M_i. We resolve this conflict as follows: If e_2 was assigned to M_i' then we simply skip e_1. Else, if we skipped e_2 because it was not possible to assign it to M_i' then it is possible to assign e_1 to M_i'. Thus, in this case we assign e_1 to M_i' rather than to M_i.

A second conflict may occur if both e_1 and e_k were skipped. Thus we couldn't assign e_1 to M_i and we couldn't assign e_k to M_i'. In this case we will assign e_1 to M_i'.

In all of the above, the property that each vertex of C_i is an end of at least one edge in $M_i \cup M_i'$ is maintained. It is also easy to see that M_i and M_i' contain at least one edge. □

We note that the property that both M_i and M_i' are nonempty is important to assure that after the transfer of any of these matchings to W at least one edge from each cycle was transferred and the remaining edges form a subtour. The following two lemmas now follow:

Lemma 2. *For each vertex of C_i, the probability that one of the edges incident to it in C_i will be transfered to W by Algorithm $A2$ is at least* $1/2$.

```
A2
  input
  1. A complete undirected graph $G = (V, E)$ with weights $w_e$ $e \in E$.
  returns A tour $T$.
  begin
  Compute a maximum cycle cover $\mathcal{C} = \{C_1, \ldots, C_r\}$.
  Let $E'$ be the edges of $G$ with two ends in different cycles of $\mathcal{C}$.
  Compute a maximum weight matching $M' \subseteq E'$.
  Compute a maximum matching $W$ in $G$.
  for $i = 1, \ldots, r$:
      Construct disjoint nonempty matchings, $M_i$ and $M'_i$ from edges of $C_i$ so that $M_i \cup W$ and
      $M'_i \cup W$ are subtours and each vertex of $C_i$ is an end of at least one edge from $M_i \cup M'_i$.
      Transfer either $M_i$ or $M'_i$ from $C_i$ to $W$, each with probability $\frac{1}{2}$.
      end for
  Complete $W$ into a tour $T_2$.
  Let $\mathcal{P}$ be the set of paths that were formed from $C_1, \ldots, C_r$ after the transfer of edges.
  $M := \{(i, j) \in M'$: $i$ and $j$ have degree 1 in $\mathcal{P}\}$.
  % $M \cup \mathcal{P}$ consists of paths $P_1^*, \ldots, P_s^*$ and cycles $C_1^*, \ldots, C_t^*$ such that
  each cycle contains at least two edges from $M$.%
  $\mathcal{P}^* := \{P_1^*, \ldots, P_s^*\}$.
  begin deletion step:
    for $i = 1, \ldots, t$:
        Randomly select an edge $e \in C_i^* \cap M$.
        $\mathcal{P}^* := \mathcal{P}^* \cup (C_i^* \setminus e)$.
        end for
   end deletion step
  Complete $\mathcal{P}^*$ to a tour $T_3$ by arbitrary addition of edges.
  return $T_2, T_3$.
  end A2
```

Fig. 4. Algorithm $A2$.

Lemma 3. *For every edge $e \in M'$, the probability that it is in M (i.e., both of its end vertices have degree 1 in $\mathcal{P}$) is at least 1/4.*

By the fact that each cycle in $C_1^*, \ldots, C_t^*$ contains at least two edges from M, we obtain:

Lemma 4. *For every edge $e \in M$, the probability that it will be deleted by the deletion step of Algorithm $A2$ is at most 1/2.*

Theorem 5.

$$\max\{w(T_1), w(T_2), w(T_3)\} \geqslant \frac{25(1-\varepsilon)}{33-32\varepsilon}\, opt.$$

Proof. Let T be an optimal tour. Define T_{int} (T_{ext}) to be the edges of T whose end vertices are in the same (in different) connectivity components of $\mathcal{C}$. Suppose $w(T_{int}) = \alpha w(T) = \alpha\, opt$. Consider the tour T_1. For each short cycle of $\mathcal{C}$ Algorithm $A1$ computed a maximum weight Hamiltonian path and therefore its contribution to the weight of T_1 is at least the weight of T_{int} in the graph induced by its vertices. Since $\mathcal{C}$ is a maximum cycle cover, $w(C_i)$ is at least the weight of T_{int} in the subgraph induced by the vertices of C_i. In each long cycle we deleted a minimum weight edge, thus subtracting from its weight at most a factor of ε. Therefore, $w(T_1) \geqslant (1-\varepsilon) w(T_{int}) \geqslant (1-\varepsilon)\alpha\, opt$.

Now consider T_2 and T_3. Let $\delta\, opt$ be the total weight of the edges transferred from $\mathcal{C}$ to W. Since the original weight of W is at least $\frac{1}{2}\, opt$, then $w(T_2) \geqslant (\frac{1}{2} + \delta)\, opt$.

The weight of $\mathcal{P}$, the set of paths formed from $\mathcal{C}$ after the transfer of edges, is at least $(1-\delta)\,opt$. To this we added edges as follows: We first computed a maximum matching M' over G'. $w(M') \geqslant \frac{1}{2}w(T_{ext})$ since T_{ext} can be covered by two disjoint matchings in G'. We then obtained M by deleting all of the edges of M' except those whose two ends have degree 1 in $\mathcal{P}$. By Lemma 3, each edge in G' has with probability $1/4$ two ends that have degree 1 in $\mathcal{P}$. Therefore,

$$w(M) \geqslant \frac{1}{4}w(M') \geqslant \frac{1}{8}w(T_{ext}) = \frac{1}{8}(1-\alpha)\,opt.$$

At this stage we considered the edges of M on cycles of $M \cup \mathcal{P}$ and deleted each $e \in M$ with probability at most $1/2$. The expected weight of the remaining edges is at least $\frac{1}{2}w(M) \geqslant \frac{1}{16}(1-\alpha)\,opt$. Finally, we obtained T_3 by connecting the remaining edges to $\mathcal{P}$. This step may only increase the weight of the solution. Thus $w(T_3) \geqslant ((1-\delta) + \frac{1}{16}(1-\alpha))\,opt$.

We conclude that

$$\max\{w(T_1), w(T_2), w(T_3)\} \geqslant \max\left\{(1-\varepsilon)\alpha, \frac{1}{2}+\delta, \frac{17}{16}-\delta-\frac{\alpha}{16}\right\}opt.$$

The minimum value of the right hand side obtains when $\alpha = \frac{25}{33-32\varepsilon}$ and it then equals $\frac{25(1-\varepsilon)}{33-32\varepsilon}\,opt$. □

The two time consuming parts of the algorithm are the computation of a maximum 2-matching and the computation of maximum Hamiltonian paths on the subgraphs induced by the short cycles. The first can be done in $O(n^3)$ time and the latter can be done by applying dynamic programming in time $O(l^2 2^l)$ per subgraph induced by l vertices. Since for short cycles $l \leqslant \varepsilon^{-1}$ this amounts to $O(n^2 2^{1/\varepsilon})$. Thus the overall complexity is $O(n^2(n + 2^{1/\varepsilon}))$. Given any factor $r < 25/33$ we can fix $\varepsilon > 0$ and obtain a solution of value at least $r\,opt$ in $O(n^3)$ time.

4. Maximum latency TSP

Chalasani and Motwani [2] considered the following *maximum latency traveling salesman problem* (Max latency TSP) in relation to their treatment of dynamic delivery problems. Given an undirected graph G with vertices $\{v_0, v_1, \ldots, v_n\}$ and edge weights $w(e)$, find a Hamiltonian path starting from v_0 such that the total *latency* of the vertices is maximized. If in a given path P the length of the ith edge traversed is w_i, then the latency of the jth vertex visited is $L_j = \sum_{i=1}^{j} w_i$ and the total latency $L(P)$ is

$$L(P) = \sum_{j=1}^{n} L_j = \sum_{i=1}^{n}(n-i+1)w_i.$$

Chalasani and Motwani showed that, under the assumption that the edge weights satisfy the triangle inequality, the *farthest neighbor* algorithm, starting from v_0, yields a solution of latency at least half the optimal.

We now point out a relation between Max TSP and Max latency TSP that yields a bounded performance guarantee *without assuming the triangle inequality*. Specifically, we suggest the algorithm given in Fig. 5.

Let P be a maximum latency path. Let $\widehat{T}$ be the Hamiltonian tour obtained by adding to P the edge

```
Max_Latency
  **input**
  1. A complete undirected graph $G = (V, E)$ with weights $w_e$ $e \in E$.
  2. A distinguished vertex $v_0 \in V$.
  **returns**
  A Hamiltonian path starting at $v_0$.
  **begin**
  Compute a tour $T$.
  Let $e_1$ and $e_2$ the two edges of $T$ which are incident with $v_0$.
  Let $P_i = T \setminus \{e_i\}$ $i = 1, 2$.
  **return** the path with maximum latency between $P_1$ and $P_2$.
  **end** Max_Latency
```

Fig. 5. Maximum latency algorithm.

between its first and last vertices. Let P' be the alternative solution obtained by deleting the first edge of P (i.e., the one incident to v_0) from $\widehat{T}$. Thus, P' visits the vertices, other than v_0, in reverse order of P. It is easy to see that each edge of $\widehat{T}$ precedes each of the vertices $v_1, \ldots, v_n$ in exactly one of the two paths P and P'. Therefore, each of these edges contributes n times its weight to the sum $L(P) + L(P')$. It follows that

$$L(P) + L(P') = nw(\widehat{T})$$

and in particular,

$$L(P) \leqslant nw(\widehat{T}).$$

Suppose now that the weight of the tour T computed by *Max_Latency* is guaranteed to be at least α times that of a maximum weight tour in G. In particular $w(T) \geqslant \alpha w(\widehat{T})$. Then, $L(P_1) + L(P_2) = nw(T) \geqslant \alpha nw(\widehat{T})$. Thus,

$$\max\bigl\{L(P_1), L(P_2)\bigr\} \geqslant \frac{\alpha}{2} nw(\widehat{T}) \geqslant \frac{\alpha}{2} L(P).$$

With Serdyukov's algorithm we obtain a $3/8$ algorithm for the maximum latency TSP while with our new algorithm for Max TSP we obtain a randomized algorithm with a 25/66 bound.

References

[1] A.I. Barvinok, D.S. Johnson, G.J. Woeginger, R. Woodroofe, Finding maximum length tours under polyhedral norms, in: Proceedings of IPCO VI, Lecture Notes in Computer Science, Vol. 1412, 1998, pp. 195–201.

[2] P. Chalasani, R. Motwani, Approximating capacitated routing and delivery problems, SIAM J. Comput. 28 (1999) 2133–2149.

[3] R. Hassin, S. Rubinstein, An approximation algorithm for the maximum traveling salesman problem, Inform. Process. Lett. 67 (1998) 125–130.

[4] A.V. Kostochka, A.I. Serdyukov, Polynomial algorithms with the estimates 3/4 and 5/6 for the traveling salesman problem of the maximum, Upravlyaemye Sistemy 26 (1985) 55–59 (in Russian).

[5] A.I. Serdyukov, An algorithm with an estimate for the traveling salesman problem of the maximum, Upravlyaemye Sistemy 25 (1984) 80–86 (in Russian).