Approximation Algorithms with Bounded Performance Guarantees for the Clustered Traveling Salesman Problem

N. Guttmann-Beck, R. Hassin, S. Khuller, and B. Raghavachari

Abstract. Let $G = (V, E)$ be a complete undirected graph with vertex set $V$, edge set $E$, and edge weights $w(e)$ satisfying triangle inequality. The vertex set $V$ is partitioned into clusters $V_1, \ldots, V_k$. The clustered traveling salesman problem is to compute a shortest Hamiltonian cycle (tour) that visits all the vertices, and in which the vertices of each cluster are visited consecutively. Since the problem is a generalization of the traveling salesman problem, it is NP-hard. In this paper we consider several variants of this basic problem and provide polynomial time approximation algorithms for them.

Key Words. Traveling salesman problem, Approximation algorithms, Clustered traveling salesman.

1. Introduction. Let $G = (V, E)$ be a complete undirected graph with vertex set $V$, edge set $E$, and edge weights $w(e)$ satisfying triangle inequality. The vertex set $V$ is partitioned into clusters $V_1, \ldots, V_k$. The clustered traveling salesman problem (CTSP) is to compute a shortest Hamiltonian cycle (tour) that visits all the vertices, and in which the vertices of each cluster are visited consecutively. Applications and other related work may be found in [3], [8], and [13] and an exact branch and bound algorithm is described in [15]. The traveling salesman problem (TSP) can be viewed as a special case of CTSP in which there is only one cluster $V_1 = V$ (alternatively, each $V_i$ is a singleton). We deal with several variants of the problem depending on whether or not the starting and ending vertices of a cluster have been specified. Since all the variants are generalizations of TSP, they are all NP-hard.

In this paper we focus on the design of approximation algorithms with guaranteed performance ratios. These are algorithms that run in polynomial time, and produce suboptimal solutions. We measure the worst case ratio of the cost of the solution generated by the algorithm to the optimal cost. We present approximation algorithms with bounded performance ratios for several different variants of this problem. The previously known related results are a 3.5-approximation for the problem with given starting vertices [2].

1 The research of S. Khuller was supported by NSF CAREER Award CCR-950135 and by B. Raghavachari was supported by NSF Research Initiative Award CCR-8804625.
2 Department of Statistics and Operations Research, Tel-Aviv University, Tel-Aviv 69978, Israel. morh@tau.ac.il.
3 Department of Computer Science and UMIACS, University of Maryland, College Park, MD 20742, USA. samir@cs.umd.edu.
4 Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75083-0688, USA. rbh@udallas.edu.

Online publication June 26, 1999.
(and this result extends to the case where no starting or ending vertices are given, with the same approximation bound), a \( \frac{1}{2} \)-approximation for the problem in which there are only two clusters with unspecified end vertices and a prespecified starting vertex [9], and a \( \frac{1}{2} \)-approximation for the problem in which the order of visiting the clusters in the tour is specified as part of the problem [1].

In this paper we describe a 1.9091-approximation algorithm for the problem in which the starting and ending vertices of each cluster are specified. We give a 1.8-approximation algorithm if for each cluster the two end vertices are given, but we are free to choose any one as the starting vertex and the other one as the ending vertex. We give a 2.75-approximation algorithm for the case when we are only given clusters with no specific starting and ending vertices, and a 2.643-approximation algorithm if we are only given the starting vertex in each cluster.

Our solutions use known approximation algorithms to three closely related problems: the traveling salesman path problem (TSPP), theacker crane problem (SCP), and the rural postman problem (RPP). These problems are discussed in Section 2. In fact one of the contributions of our paper is to design new algorithms for TSPP, and to show their use in improving the previously known approximation factors for TSPP, along with careful use of the algorithms for SCP and RPP.

Outline of the paper. In Section 2 we discuss some basic notation that is used in the paper. In addition, we review algorithms for approximating TSPP, SCP, and RPP. In Section 3 we address the case in which the starting and ending vertices in each cluster are specified. In Section 4 we address the case in which the end vertices are given, but either one could be chosen as the entry or exit vertex for the cluster. In Section 5 we address the case in which only the starting vertex in each cluster is specified. In Section 6 we address the case in which no entry or exit vertex is specified for any cluster. Finally, in Section 7 we study the problem when the number of clusters is a constant, and describe a \( \frac{1}{2} \)-approximation algorithm for all variants. We also show that obtaining an approximation ratio of \( \sigma \) for \( k \) clusters (\( k \geq 4 \)) with unspecified end vertices, implies an approximation ratio of \( \frac{\sigma}{k} \) for TSPP with specified end vertices. Thus, improving the approximation ratio for a constant number of clusters is at least as hard as improving the approximation ratio for TSPP for which the best approximation ratio is currently \( \frac{2}{3} \) [10].

2. Preliminaries. Some of our algorithms use a directed cycle cover routine in digraphs. The directed cycle cover problem is to find a set of directed cycles of minimum total weight that includes all the vertices in a given digraph. This problem is equivalent to weighted bipartite matching, which is also called the assignment problem [14]. We also use an undirected cycle cover algorithm that finds in an undirected graph a set of undirected cycles of minimum total weight that includes all its vertices. This problem can be solved by applying a weighted matching algorithm [16].

For a graph \( G = (V, E) \) we denote by \( d(e) \) the weight (also known as length) of an edge \( e \in E \). For a subset \( E' \subseteq E \) we denote \( L(E') = \sum_{e \in E'} d(e) \), the total weight (length) of the edges. Let \( OPT \) denote both an optimal solution of the problem under consideration and its total length. Similarly, let \( MST(G) \) denote both a minimum-weight
spanning tree of $G$ and its weight. We also assume that the edge weights obey the triangle inequality.

2.1. The Traveling Salesman Path Problem. Hoogeveen [10] considered three variations of the traveling salesman path problem (TSPP), in which as part of the input instance the following constraints are placed on the end vertices of the resulting Hamiltonian path:

(i) both end vertices are specified, (ii) only one of the end vertices is specified, and (iii) no end vertices are specified. For the latter two cases, it was shown that a straightforward adaptation of Christofides' algorithm yields an algorithm with a performance ratio of $\frac{3}{2}$. The case with two specified ends is more difficult as we now elaborate.

Let $s$ and $t$ be two specified vertices in $G$. We consider the problem of finding a path with $s$ and $t$ as its two ends, that visits all vertices of $G$. One can solve this problem in a manner similar to Christofides' algorithm for TSP [4], by starting with MST($G$), adding a suitable matching, and then finding an Eulerian walk of the resulting graph. We do not get a $\frac{3}{2}$-approximation ratio since in TSPP the optimal solution is a path, and not a tour. The bound of $\frac{3}{2}$ OPT on the weight of a minimum-weight perfect matching of a subset of the vertices, which holds for TSP (tour), does not hold here.

We obtain a $\frac{3}{2}$-approximation ratio as follows.

**Theorem 2.1.** There exists a polynomial-time algorithm for TSPP between given end vertices $s$ and $t$, that finds solutions $S_1$ and $S_2$ which satisfy the following equations:

\[
I(S_1) \leq 2 \text{MST}(G) - I(s, t)
\]

\[
\leq 2 \text{OPT} - I(s, t),
\]

\[
I(S_2) \leq \text{MST}(G) + \frac{1}{2} (\text{OPT} + I(s, t))
\]

\[
\leq \frac{3}{2} \text{OPT} + \frac{1}{2} I(s, t).
\]

**Proof.** We "double" the edges of MST($G$) except for those on the unique $s$-$t$ path on it. The result is a connected multigraph whose vertex degrees are all even except for those of $s$ and $t$. We now find an Eulerian walk between $s$ and $t$ on this multigraph and turn it into a Hamiltonian path between $s$ and $t$ without increasing the weight by shortcutting and applying the triangle inequality. We call it $S_1$. The length of $S_1$ is at most $2 \text{MST}(G) - I(s, t)$, which is at most $2 \text{OPT} - I(s, t)$.

To obtain $S_2$, we adopt the following strategy. Consider adding the edge $(s, t)$ to OPT, and making it a cycle. The length of this cycle is OPT + I(s, t). The cycle can be decomposed into two matchings between any even-size subset of vertices, and the length of the smaller matching is at most $\frac{3}{2} (\text{OPT} + I(s, t))$. We use the strategy of Hoogeveen [10], and add to MST($G$) a minimum-weight matching of vertices selected based on their degree in MST($G$) (odd degree vertices of $V - \{s, t\}$) and even degree vertices of $(s, t)$ in MST($G$), and output an Eulerian walk $S_2$ from $s$ to $t$ in the resulting graph. This Eulerian walk can be converted into a Hamiltonian path by shortcutting. Using the triangle inequality, we obtain that $I(S_2)$ is at most $\text{MST}(G) + \frac{1}{2} (\text{OPT} + I(s, t)) \leq \frac{3}{2} \text{OPT} + \frac{1}{2} I(s, t)$.

**Corollary 2.2.** The shortest of the paths $S_1$ and $S_2$ is at most $\frac{3}{2}$ OPT.
Pro...
the walk must include all the edges, but the Chinese postman problem is solvable in polynomial time by reducing it to weighted matching, whereas RPP is NP-hard.

The two algorithms, defined above for SCP, can be modified to solve RPP. We define Algorithm LongArcs2 that is similar to LongArcs, but in this case, $D$ is a set of undirected edges, and we only change the algorithm so that it completes the set of edges into an undirected cycle cover. The second part of Theorem 2.4 holds for this case as well.

Algorithm ShortArcs greatly simplifies when applied to RPP and turns out to be a straightforward generalization of Christofides's algorithm for TSP. As indicated by Frederickson [6], it produces a $\frac{3}{2}$-approximation algorithm for the problem (see the survey by Eisele et al. [5] for more details and a paper by Jensen [16] for a generalization).

We denote by Algorithm RuralPostman the algorithm which executes these two algorithms and returns the shorter solution.

**Remark 2.5.** In some of our algorithms for CTSP, we generate instances of RPP (SCP) and run the above algorithms for them. In the RPP (SCP) instances that we generate the special edges (directed arcs) are vertex disjoint. In this fact guarantees that the walks are actually tours since each vertex is visited once. These instances are themselves CTSP instances with $|V| = 2$ $(1 \leq i \leq k)$ and given end vertices (starting and ending vertices, respectively). However, we note that if, for some constant $c$, $|W| \leq c (1 \leq i \leq k)$, then we can obtain the same approximation ratios as for RPP (SCP) quite easily since within each cluster we can obtain an optimal solution.

3. Given Start and End Vertices. In this section we consider CTSP in which the starting vertex $i_1$ and ending vertex $i_k$ is given for each cluster $V_i$. Our algorithm is based on the following idea. We decompose the problem into two parts. Inside each cluster $V_i$, we find $path_i$, a path from the start vertex $i_i$ to the end vertex $i_k$ that goes through all vertices of that cluster. This is TSPP with given end vertices. In addition, we need to connect the paths by adding edges into a single cycle. We replace each cluster by a special arc from $i_i$ to $i_k$ and get an instance of SCP. Find a tour that includes all the special directed arcs. From this solution to SCP, we replace each arc $(i_i, i_k)$ by the path $path_i$, computed within that cluster.

Figure 1 describes the algorithm in detail. See Figure 2 for a sample execution of the algorithm for three clusters with given starting and ending vertices.

**Theorem 3.1.** Let $T_m$ be the tour returned by Algorithm GivenST in Figure 1. Then,

$$l(T_m) \leq \frac{3}{2}OPT < 1.909OPT.$$  

**Proof.** The algorithm consists of solving two subproblems: TSPP with given end vertices, and SCP. Let $L$ be the sum of the lengths of the paths of OPT through each cluster. Let $A$ be the length of the other edges of OPT that are not in $L$ (edges of OPT that connect the different clusters together). Let $D$ be the total length of the directed arcs $(i_i, i_k), i = 1, \ldots, k$. By Theorem 2.1, the lengths of the two solutions to TSPP with given end vertices are at most $2L - D$ and $\frac{3}{2}L + \frac{3}{2}D$. Using the fact that the minimum
Algorithm GivenST

Input
1. A graph $G = (V, E)$, $|V| = n$, with weights $w(e), e \in E$.
2. A partition of $V$ into clusters $V_1, \ldots, V_k$.
3. Start and end vertices $s$ and $t$, respectively, for each cluster $V_i, i = 1, \ldots, k$.

returns A clustered tour

begin

Step 1
For each $V_i$, compute path $p_i$, a Hamiltonian path with end vertices $s$ and $t$.

Step 2
Apply Algorithm StackerCrane with special directed arcs $(s_i, t_i) \mid i = 1, \ldots, k$ to obtain tour $T$.

Step 3
In $T$, replace the special directed arc $(s_i, t_i)$ by path $p_i, i = 1, \ldots, k$.

Step 4
return the resulting tour $T_n$.

end GivenST

Fig. 1. Algorithm GivenST.

of a set of quantities is at most any convex combination,

$$I(TSPP) \leq \min(2L - D, \frac{1}{2}L + \frac{1}{2}D)$$

$$\leq \frac{3}{4}(2L - D) + \frac{1}{4}(\frac{1}{2}L + \frac{1}{2}D)$$

$$= \frac{3}{4}L - \frac{1}{4}D.$$

There exists a solution to SCP of length at most $A + D$. Hence, by Theorem 2.4, the lengths of the two solutions to SCP are at most $\frac{3}{4}A + 2D$ and $\frac{3}{4}A + D$. Therefore the solution returned by the SCP algorithm is at most

$$I(SCP) \leq \min(\frac{3}{4}A + 2D, 3A + D)$$

$$\leq \frac{3}{4}(\frac{3}{4}A + 2D) + \frac{1}{4}(3A + D)$$

$$= \frac{3}{4}A + \frac{3}{4}D.$$

In Step 3 of Algorithm GivenST, the two solutions are combined by replacing arcs of length $D$ in the TSPP solution by the SCP solution. We obtain an upper bound on the length of the solution $T_n$ thus obtained by combining the above equations.

$$I(T_n) = I(TSPP) - D + I(SCP)$$

$$\leq \left(\frac{3}{4}L - \frac{1}{4}D\right) - D + \left(\frac{3}{4}A + \frac{3}{4}D\right)$$

$$= \frac{3}{4}(L + A) = \frac{3}{4}OPT.$
4. Given End Vertices. In this section we consider CTSP in which for each \( i, i = 1, \ldots, k \), we are given the two end vertices, \( (x_i^1, x_i^2) \), of cluster \( V_i \). The vertices of each cluster must be contiguous on the tour, and the end vertices of cluster \( V_i \) must be \( x_i^1 \) and \( x_i^2 \), but we are free to select any one of them as the start vertex and the other vertex as the end vertex. We modify Algorithm GivenST by executing Algorithm RuralPrimman rather than Algorithm StackerCrane, since \( p_i \) can be oriented in any direction. The solution obtained consists of the special edges \( (x_i^1, x_i^2), i = 1, \ldots, k \), and other undirected edges between the end vertices. We replace the special edges by the corresponding paths between \( x_i^1 \) to \( x_i^2 \) computed in Step 1. Figure 3 describes the algorithm in detail.

**Theorem 4.1.** Let \( T_n \) be the tour returned by Algorithm GivenEnds in Figure 3. Then

\[
I(T_n) \leq \frac{3}{2} \text{OPT}.
\]

**Proof.** The proof is similar to that of Theorem 3.1, and we provide only a sketch. The length of the paths computed in Step 1 is at most

\[
I(T_{\text{SPP}}) \leq \min \left( 2L - D, \frac{3}{2}L + \frac{1}{2}D \right)
\]

\[
\leq \frac{1}{2}(2L - D) + \frac{3}{2}L + \frac{1}{2}D
\]

\[
= \frac{3}{2}L - \frac{1}{2}D.
\]
Algorithm GivenEnds

Input
1. A graph \( G = (V, E) \), \( |V| = n \), with weights \( l(e), e \in E \).
2. A partition of \( V \) into clusters \( V_1, \ldots, V_k \).
3. In each cluster \( i \), \( i = 1, \ldots, k \), two vertices \( s_i^1 \) and \( s_i^2 \).

return Returns a clustered tour.

begin

Step 1
For each \( i \), compute \( path_i \), a Hamiltonian path with end vertices \( s_i^1 \) and \( s_i^2 \).

Step 2
Apply Algorithm RuralPostman with the special edges \( \{s_i^1, s_i^2\} \mid i = 1, \ldots, k \) to obtain tour \( T \).

Step 3
In \( T \), replace the special edge \( \{s_i^1, s_i^2\} \) by path\(_i\), \( i = 1, \ldots, k \).

Step 4
return the resulting tour \( T_m \).

end

Fig. 3. Algorithm GivenEnds.

There is a solution to RPP of length at most \( A + D \). The lengths of the two solutions we can find for RPP are \( \frac{3}{2}(A + D) \) and \( 2A + D \). Therefore the solution returned by the RPP algorithm is at most

\[
I(SCP) \leq \min \left\{ \frac{3}{2}(A + D), 2A + D \right\} \\
\leq \frac{3}{2}(A + D) + \frac{1}{2}(A + D) \\
= \frac{3}{2}A + \frac{3}{2}D.
\]

In Step 3 of Algorithm GivenEnds, the two solutions are combined by replacing edges of length \( D \) in the TSP solution by the RPP solution. We obtain an upper bound on the length of the solution \( T_m \) thus obtained by combining the above equations:

\[
I(T_m) = I(TSP) - D + I(SCP) \\
\leq \left( \frac{5}{3}(L - \frac{3}{2}D) - D + \frac{1}{2}A + \frac{1}{2}D \right) \\
= \frac{5}{3}(L + A) = \frac{5}{3}OPT. \quad \square
\]

5. Given Starting Vertices. In this section we consider the version of CTSP in which, for each cluster \( i \), we are given only its starting vertex \( s_i \), and we are free to select its ending vertex. We give an algorithm with an approximation ratio of \( 2.643 \). We propose two different heuristics, and select the shorter of the tours generated. In the first heuristic, we combine a tour of the starting vertices with tours of the individual clusters to generate
Algorithm GivenStart

Input
1. A graph $G = (V, E)$, $|V| = n$, with weights $w(e), e \in E$.
2. A partition of $V$ into clusters $V_1, \ldots, V_k$.
3. In each cluster $i$, starting vertex $s_i$.

returns A clustered tour

begin

Step 1
Compute a tour $T_i$ that visits all vertices of $V_i$, for each $i \in \{1, 2, \ldots, k\}$.
Compute a tour $T$ of just the starting vertices $\{s_1, s_2, \ldots, s_k\}$.
Let $T_1$ be a tour obtained by shortcutting $S \cup (\bigcup_{i=1}^{k} T_i)$.

Step 2
For each cluster $V_i$, choose an end vertex $t_i$ that is farthest from $s_i$.
Let Algorithm GivenST return tour $T_i$ with start vertices $s_i$ and end vertices $t_i$.

Step 3
return the shorter of the tours $T_1$ and $T_k$.

end GivenStart

Fig. 4. Algorithm GivenStart.

a clustered tour. In the second heuristic, for each cluster, we select the end vertex $t_i$ to be the farthest vertex from $s_i$ within the cluster. We then find a solution using Algorithm GivenST (Section 5) that finds a clustered tour when the start and end vertices of each cluster are given. Figure 4 describes the algorithm.

In Step 1 we compute $k + 1$ tours, one for each cluster $V_i$, and a tour that visits just the start vertices. All these tours can be computed using the TSP algorithm of Christofides [4]. The intuition behind this heuristic is that it does well when the sum of the distances between the start and end vertices of each cluster in the optimal solution is small relative to the total cost. In Step 2 we select an end vertex $t_i \in V_i$ that maximizes $l(s_i, t_i)$. With that selection of end vertices, we can now apply Algorithm GivenST to find a clustered tour.

We introduce some notation to analyze the algorithm. Let $A$ be the total cost of those edges of OPT that connect vertices of two different clusters (there are exactly $k$ such edges). Let $l$ be the sum of the lengths of the paths of OPT through the clusters. By definition $OPT = A + l$. Let $d$ be the sum of the distances between the start and end vertices of each cluster of OPT; note that we are summing the lengths of direct edges that connect start vertices to the end vertices chosen by OPT. Let $D$ be the sum of distances between $s_i$ and $t_i$, i.e., $D = \sum_{i=1}^{k} l(s_i, t_i)$. Since we chose $t_i$ in $V_i$ as the vertex that maximizes $l(s_i, t_i)$, $D < D$.

**Lemma 5.1.** Let $T_i$ be the tour computed in Step 1 of Algorithm GivenStart. Then

$$l(T_i) \leq \frac{1}{2} OPT + 2d.$$
**PROOF.** The Hamiltonian paths through each cluster can be converted to cycles by adding an edge that connects an end vertex with the end vertex in that cluster of OPT (see Figure 5). Hence there exists a collection of $k$ tours, one through each cluster $V_i$, whose total cost is $L + d$. We now follow the analysis of Christofides (4). The sum of costs of minimum spanning trees through each cluster is at most $L$. The cost of the matchings connecting odd-degree vertices of each cluster is at most $\frac{1}{2}(L + d)$. Therefore the sum of the $k$ tours computed in Step 1 is at most $\frac{1}{2}L + \frac{1}{2}d$. There exists a tour of the just the start vertices of length $A + d$, which is obtained from OPT by replacing the paths through each cluster of length $L$ by a direct edge connecting the end vertices and deleting the intermediate vertices. Hence the cost of the tour $T$ computed in Step 1 is at most $\frac{1}{2}(A + d)$. Tour $T'$ is obtained by combining all the above tours and its length is at most $\frac{1}{2}(L + \lambda) + 2d = \frac{1}{2}OPT + 2d$.

**Lemma 5.2.** Let $T_i$ be the tour computed in Step 2 of Algorithm GivenStart. Then

$$l(T_i) \leq \frac{1}{2}OPT + 2L + \frac{1}{2}d.$$  

**PROOF.** The tour $T_i$ is obtained by running Algorithm GivenStart after choosing an end vertex $x_i$ for each cluster. The algorithm computes a solution to SCP (Step 2), which is computed in turn by computing two solutions and taking the shorter of the two (see Section 2.2). We prove that if one just takes the SCP solution computed by Algorithm Shortest, we get the desired result. We observe that OPT can be a shortcut to obtain a feasible solution to the corresponding SCP and therefore if Algorithm Shortest is applied to this problem, we get a tour whose length is at most $\frac{1}{2}OPT + \frac{1}{2}D$. In this tour we replace each arc $(x_i, t_i)$ by a path from $x_i$ to $t_i$. By Theorem 2.1 the length of such paths is at most $\sum_{i=1}^{m} MST(G_i) - D \leq 2L - D$, where $G_i$ is the subgraph induced by $V_i$. Hence the total length of the tour obtained is

$$l(T_i) \leq \frac{1}{2}OPT + \frac{1}{2}D - D + (2L - D)$$

$$= \frac{1}{2}OPT + 2L - \frac{1}{2}D$$

$$\leq \frac{1}{2}OPT + 2L - \frac{1}{2}d,$$

since $d \leq D$.  


THEOREM 5.3. Let $T_n$ be the clustered tour returned by Algorithm GivenStart in Figure 4. Then

$$l(T_n) \leq \frac{5}{4} \text{OPT} < 2.643 \text{OPT}.$$  

PROOF. If $d \leq \frac{1}{4} L$, then, by Lemma 5.1 and the obvious inequality $L \leq \text{OPT}$,

$$l(T_n) \leq \frac{5}{4} \text{OPT} + \frac{1}{4} L \leq \frac{5}{4} \text{OPT}.$$  

If $d > \frac{1}{4} L$, then the same inequality holds for $l(T_1)$ by Lemma 5.2. Since the algorithm chooses the shortest of the tours, the theorem follows.

6. Unspecified End Vertices. In this case we are only given the clusters and we are free to choose the start and end vertices in the clusters. We give a 2.75 approximation for an arbitrary number of clusters. We first apply a TSPP algorithm with unspecified ends within each cluster. We use the direct edges between the ends of these paths as special edges for an RPP instance. We compute an approximate tour of this instance and finally replace each special edge by the corresponding path to produce our first tour. To obtain our second tour, in each cluster, select two vertices $v_i$ and $v_j$, such that $l(v_i, v_j)$ is maximized, to be the end vertices in the cluster, and then apply Algorithm GivenEnds to obtain a tour. Finally, we select the shorter tour. Figure 6 describes the algorithm.

As in the previous section, $L$ denotes the sum of the lengths of the Hamiltonian paths within the clusters in OPT, and $A$ denotes the sum of the lengths of the remaining edges of OPT. Let $D = \sum_{i=1}^n l(s_i, b_i)$. The first algorithm works well when $D$ is small, and the second works well when $D$ is large.

LEMMA 6.1. Let $T_n$ be the tour computed in Step 1 of Algorithm UnspecifiedEnds. Then

$$l(T_n) \leq \frac{5}{4} \text{OPT} + \frac{1}{4} L + A.$$  

PROOF. Consider an optimal solution OPT. Give it an arbitrary orientation. Let $v_i$ and $v_j$ be the first and last vertices of $V_i$ in OPT. Suppose, without loss of generality, that we name $a_i$ and $b_i$ so that OPT visits $v_i, a_i, b_i, v_j$ in this order. It follows that

$$L \geq \sum_{i=1}^n (l(v_i, a_i) + l(a_i, b_i) + l(b_i, v_j)).$$  

There exists a Rural Postman tour with special edges $(a_i, b_i)$ of length at most $A + \sum_{i=1}^n (l(v_i, a_i) + l(a_i, b_i) + l(b_i, v_j))$ so that the length of the tour returned by Algorithm RuralPostman is at most

$$\frac{5}{4} \left( A + \sum_{i=1}^n (l(v_i, a_i) + l(a_i, b_i) + l(b_i, v_j)) \right).$$
Algorithm UnspecifiedEnds

Input
1. A graph $G = (V, E)$, $|V| = n$, with weights $l(e), e \in E$.
2. A partition of $V$ into clusters $V_1, \ldots, V_k$.

returns A clustered tour

begin

Step 1

Apply a TSFP algorithm with unspecified end vertices to each $V_i$, $i \in \{1, 2, \ldots, k\}$.

Let $path_i$ be the resulting path on $V_i$, and denote its end vertices by $a_i$ and $b_i$.

Apply Algorithm RuralPostman with special edges $(a_i, b_i), i = 1, \ldots, k$, and let $T_i$ be the tour obtained by replacing the special edge $(a_i, b_i)$ by path$_i, i = 1, \ldots, k$.

Step 2

In each cluster find vertices $u_i$ and $v_i$ that maximize $l(u_i, v_i)$.

Apply Algorithm GivenEnds with end vertices $(u_i, v_i)$, and let $T_i$ be the tour that it returns.

Step 3

return the shorter of the tours $T_i$ and $T_i$.

Fig. 6. Algorithm UnspecifiedEnds.

We now replace (for each $i$) the special edge $(a_i, b_i)$ by path$_i$. Since $\sum_{i=1}^{k} l(path_i) \leq \frac{3}{2}L$, we obtain

$$l(T_i) \leq \frac{3}{2} \left( A + \sum_{i=1}^{k} l((a_i, a_i) + l(a_i, b_i) + l(b_i, v_i)) \right) - \sum_{i=1}^{k} l((a_i, b_i) + \frac{3}{2}L,$$

$$\leq \frac{3}{2}OPT + \frac{3}{2} \sum_{i=1}^{k} l((a_i, a_i) + l(a_i, b_i) + l(b_i, v_i)) + \frac{3}{2} \sum_{i=1}^{k} l((a_i, a_i) + l(b_i, v_i)) + 2D,$$

$$\leq \frac{3}{2}OPT + \frac{1}{2} L + 2D,$$

where the last inequality follows from (1).

□

LEMMA 6.2. Let $T_i$ be the tour computed in Step 2 of Algorithm UnspecifiedEnds. Then

$$l(T_i) \leq \frac{3}{2}OPT + 2L - 2D.$$

PROOF: The proof is similar to that of Lemma 5.2, except that we apply Algorithm RuralPostman instead of Algorithm StackerCrane, since we can choose either of $(u_i, v_i)$ as
the start vertex. We observe that $OPT$ can be a shortcut to obtain a feasible solution to the corresponding RPP. Therefore, the RPP solution computed is at most $\frac{1}{2} OPT$. From this solution, we replace each special edge $(x_i, x_j)$ by a path connecting $x_i$ and $b$, that includes all vertices in $V_i$. The length of these paths is at most $\sum_{i=1}^{k} MST(V_i) - D = 2L - D$ (Theorem 2.1). Hence the length of the tour is at most $\frac{1}{2} OPT - D + (2L - D) = \frac{1}{2} OPT + 2L - 2D$.

\[ \mathcal{I}(T_a) \leq \frac{1}{2} OPT. \]

**Proof.** If $2D \leq \frac{1}{2} L$, then, by Lemma 6.1 and the inequality $L \leq OPT$, \[ \mathcal{I}(T_a) \leq \frac{1}{2} OPT + \frac{1}{2} L \leq \frac{1}{2} OPT. \]

If $2D > \frac{1}{2} L$, then the same inequality holds for $\mathcal{I}(T)$ by Lemma 6.2. Since the algorithm chooses the shorter of the tours $T$ and $T_a$, the theorem follows.

\[ \mathcal{I}(T_a) \leq \frac{1}{2} OPT. \]

7. **Constant Number of Clusters.** In this section we consider CTSP where the number of clusters, $k$, is a constant. The case $k = 1$ is TSPP. We show that CTSP with given end vertices is equivalent to TSPP with given end vertices. Hence we can obtain the obvious $\frac{1}{2}$-approximation for this case by using the $\frac{1}{2}$-approximation for TSPP, but any further improvement in the approximation ratio is possible only if the approximation algorithm for TSPP with given end vertices is improved.

We also show that TSPP with given end vertices is equivalent to CTSP with unspecified end vertices for four or more clusters. This shows that an $\alpha$-approximation algorithm for this problem would imply an $\alpha$-approximation algorithm for TSPP with given end vertices.

**Theorem 7.1.** If there exists an $\alpha$-approximation algorithm to TSPP with given end vertices, then there exists an $\alpha$-approximation algorithm for CTSP for a constant number of clusters (for all the variants we consider in this paper).

**Proof.** Let $k$ be the number of the clusters. For unspecified end vertices, we repeat the following algorithm for each possible choice of a starting vertex in each cluster. For a given set of start vertices in each cluster, we construct an auxiliary directed graph with $k$ vertices as follows. Each cluster is represented by a vertex. The length of arc $(i, j)$ is equal to the approximate length of a Hamiltonian path that starts at $x_i$, traverses all the other vertices of $V_i$, and ends at $x_j$. To compute these arc lengths, we use the $\alpha$-approximation algorithm for TSPP with given end vertices. We find an optimal TSPP tour in the auxiliary graph by complete enumeration of all possible orderings of clusters. Since there are at most $O(n^k)$ possible sets of starting vertices, we can repeat the above procedure for each set and select the best tour among them. It is easy to see that when we try the set corresponding to the same start vertices as an optimal solution, the cost
of the tour that we compute is no more than $\alpha$ times OPT. If the start vertices are given we use the same algorithm but only for the given set of start vertices. For given start and end vertices we use the TSPP algorithm to compute a tour between each pair of given vertices, and complete the tour optimally by complete enumeration of all possible ordering of clusters. For fixed end vertices (when the order is not given) we enumerate over all $2^2$ possible orderings.

**Theorem 7.4.** If there exists an $\alpha$-approximation algorithm for TSPP, then there exists an $\alpha$-approximation algorithm for TSPP with given end vertices.

**Proof.** We prove the claim for $k = 2$, and the same idea extends to all $k > 2$. Let $G = (V, E)$ and end vertices $s$ and $t$ be given as an instance of TSPP. We construct an instance of TSPP with two clusters as follows. We make two copies of $G$, namely, $G_1$ and $G_2$. The copy of vertex $v$ in $G_1$ is identified as $v_1$. Distances between vertices of the same copy are the same as the distances in $G$. We also set $l(s_1, s_2) = l(t_1, t_2) = 0$. Distances between vertices in different copies are computed by routing the path through the linking vertices. Thus, for $u_1 \in G_1$ and $v_2 \in G_2$, $l(u_1, v_2) = \min(l(u_1, t_1) + l(t_1, v_2), l(u_1, s_2) + l(s_2, v_2))$.

The vertices of each copy of $G$ form a cluster and the start and end vertices are specified to be $s_1$ and $t_1$, respectively, for $G_1$, $i \in \{1, 2\}$. Any solution to TSPP with given end vertices consists of the union of two Hamiltonian paths between $s$ and $t$. Therefore, we get an $\alpha$-approximation for TSPP with a given end point problem by computing an $\alpha$-approximation of the instance of the CTSP, and selecting the shorter of the two $s$-$t$ paths.

**Theorem 7.5.** If there exists an $\alpha$-approximation algorithm for TSPP with unspecified end vertices and $k$ clusters, for some $k \geq 4$, then there exists an $\alpha$-approximation algorithm for TSPP with given end vertices.

**Proof.** Clearly a TSPP algorithm for $k > 4$ can be used to get the same performance guarantee for $k = 4$. Hence we prove the claim assuming $k = 4$. Let $G = (V, E)$ and end vertices $s$ and $t$ be given as an instance of TSPP. We construct an instance of TSPP with four clusters as follows (see Figure 7). We make two copies of $V = \{s, t\}$, namely, $V_1$ and $V_2$. The copy of vertex $v$ in $V_1$ is identified as $v_1$. Distances between vertices of the
same copy are the same as the distances in $G$. For $x \in \{s, t\}$ and $i \in \{1, 2\}$, the distance between $x$ and $v_i$ is equal to the distance between $x$ and $v_i$ in $G$. Distances between vertices in different copies are computed by routing the path through $s$ or $t$. Thus, for $u_1 \in V_1$ and $v_2 \in V_2$, $d(u_1, v_2) = \min(d(u_1, i) + d(i, v_2))$.

We have four clusters, namely $\{1\}$, $\{2\}$, $\{1, V_1\}$, and $\{V_2\}$.

If there is a solution to TSP of length $L$, then there is a solution to CTSP with length $2L$. If we have an $\alpha$-approximation for CTSP, we are guaranteed to obtain a solution of length at most $2\alpha L$. We show that any solution to CTSP induces two paths connecting $s$ and $t$ that also visit the vertices in $V' = \{s, t\}$, and by taking the shorter path we can obtain a TSP solution from $s$ to $t$ of length $\alpha L$. This is an $\alpha$-approximation for TSP.

There are two cases (the others are essentially isomorphic). Suppose the tour visits the clusters in the order $s, V_1, t, V_2, s$. In this case the tour clearly decomposes into two $s$-$t$ Hamilton paths, one through $V_1$ and the other through $V_2$. The second case is when the tour goes through the clusters in the order $s, t, V_1, V_2, s, t$. When the path goes from $V_1$ to $V_2$, it goes through either $s$ or $t$. Assume that it goes through $s$ (the other case is similar).

The two paths are $i, V_1, s$ and $i, V_2, s, i$.

\textbf{Remark 7.5.} The approximation given in [9] for CTSP with unspecified end vertices with three clusters, where one is a singleton. The main point of the last theorem is that we cannot obtain such a bound even when a single new cluster is added, unless the bound for TSP of $\frac{3}{2}$ is improved.

\textbf{References}


