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

Nili Guttmann-Beck* Refael Hassin* Samir Khuller† Balaji Raghavachari‡

1 Introduction

Let $G = (V, E)$ be a complete undirected graph with vertex set V, edge set E, and edge weights $l(e)$ 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, 9, 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 a bounded performance ratios for several different variants of this problem. The only previously known results are: a 3.5-approximation for the problem with given starting vertices [2] (and this result extends to the case where no starting or ending vertices are given, with the same approximation bound), a $\frac{3}{2}$-approximation for the problem in which there are only two clusters with unspecified end vertices [10], and a $\frac{5}{3}$-approximation for the problem in which the order in which the tour should visit the clusters 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 vertex of each cluster is 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), the *stacker crane problem* (SCP) and the *rural postman problem*

*Department of Statistics and Operations Research, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: {nili,hassin}@math.tau.ac.il

†Department of Computer Science and UMIACS, University of Maryland, College Park, MD 20742, USA. Research supported by NSF CAREER Award CCR 9501355. Email: samir@cs.umd.edu

‡Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75083-0688, USA. Research supported by NSF Research Initiation Award CCR-9409625. Email: rbk@utdallas.edu

(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 its use in improving the previously known approximation factors for CTSP; 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 vertex in each cluster is 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 $\frac{5}{3}$-approximation algorithm for all of these variants. We also show that obtaining an approximation ratio of α for k clusters ($k \geq 4$) with unspecified end vertices, implies an approximation ratio of α 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 $\frac{5}{3}$ [8]).

2 Preliminaries

Some of our algorithms use a *directed cycle cover* routine in digraphs. The directed cycle cover problem finds a set of directed cycles of minimum total weight that includes all its vertices. 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 $l(e)$ the weight (also known as length) of an edge $e \in E$. For a subset $E' \subseteq E$ we denote $L(E') = \sum_{e \in E'} l(e)$, the total weight (or 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 will use the following notation:

- Let $s_i \in V_i$ be the first vertex of cluster i in a feasible solution (if specified);
- Let $t_i \in V_i$ be the last vertex of cluster i in a feasible solution (if specified).

2.1 The traveling salesman path problem

Hoogeveen [8] considered three variations of TSPP depending on whether the two end vertices are specified, one is specified, or none. 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 the TSPP the optimal solution is a path, and not a tour. The bound of $\frac{1}{2}OPT$ on the weight of a min-weight perfect matching of a subset of the vertices, which holds for TSP, does not hold here.

We obtain a $\frac{5}{3}$ 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:*

$$\begin{aligned} l(S_1) &\leq 2\ MST(G) - l(s,t) \\ &\leq 2\ OPT - l(s,t), \\ l(S_2) &\leq MST(G) + \frac{1}{2}(OPT + l(s,t)) \\ &\leq \frac{3}{2}\ OPT + \frac{1}{2}\ l(s,t). \end{aligned}$$

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 $2MST(G) - l(s,t)$, which is at most $2OPT - l(s,t)$.

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 $OPT + l(s,t)$. 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 $\frac{1}{2}(OPT + l(s,t))$. We use the strategy of Hoogeveen [8], and add to $MST(G)$ a minimum-weight matching of vertices selected based on their degree in $MST(G)$ (odd degree vertices of $V \setminus \{s,t\}$ 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 $l(S_2)$ is at most $MST(G) + \frac{1}{2}(OPT + l(s,t)) \leq \frac{3}{2}OPT + \frac{1}{2}l(s,t)$. ■

Corollary 2.2 *The shorter of the tours S_1 and S_2 is at most $\frac{5}{3}OPT$.*

Proof: Using Theorem 2.1, observe that if $l(s,t) > \frac{1}{3}OPT$, then $l(S_1) \leq \frac{5}{3}OPT$. Otherwise, $l(s,t) \leq \frac{1}{3}OPT$, in which case $l(S_2) \leq \frac{5}{3}OPT$. ■

Remark 2.3 Hoogeveen [8] proves the bound of $\frac{5}{3}OPT$ in a different way. Solution S_2 can be directly bounded as $MST(G) + w(M)$, where $w(M)$ is the weight of a min-weight perfect matching on the subset of odd degree vertices of $V \setminus \{s,t\}$ and even degree vertices of $\{s,t\}$ in $MST(G)$. We bound $w(M)$ by showing that $OPT \cup MST(G)$ can be decomposed into three matchings, and this shows that $w(M) \leq \frac{2}{3}OPT$. Our algorithm for TSPP with given end vertices is different, and this leads to improvements in the analysis of our algorithms for CTSP.

2.2 The stacker crane problem

Let $G = (V, E)$ be an arbitrary graph that satisfies the triangle inequality. Let $D = \{(s_i, t_i) : i = 1, \ldots, k\}$ be a given set of *special directed arcs*, each with length ℓ_i. The arc (s_i, t_i) denotes an object that is at vertex s_i and needs to be moved to vertex t_i using a vehicle (called the stacker crane). The *stacker crane problem* (SCP) is to compute a shortest walk that traverses each directed arc (s_i, t_i) at least once in the specified direction (from s_i to t_i). Let $D = \sum_i \ell_i$ and $A = OPT - D$.

The stacker crane problem is a generalization of TSP since TSP can be viewed as an instance of SCP in which each vertex is replaced by an arc of zero-length. It is possible to derive a 2-approximation algorithm for the problem using standard techniques such as "twice around the tree" heuristic. *Algorithm StackerCrane* by Frederickson, Hecht and Kim [7] is a 1.8-approximation algorithm for SCP. It applies two different algorithms and then selects the best of the two solutions generated by them. We briefly review the basic ideas behind the two algorithms (for details, see [7, 12]):

- *Algorithm ShortArcs*: Shrink the directed arcs and reduce the problem to an instance of TSP. Use an approximation algorithm for the TSP instance, and then recover a solution for the original problem (the algorithm itself is somewhat subtle and the reader is referred to the original paper). This algorithm works well when D is small compared to OPT.
- *Algorithm LongArcs*: Complete the set of directed arcs into a directed cycle cover. Then find a set of edges of minimum total weight to connect the cycles together. Add two copies of each such edge, and orient the copies in opposite directions to each other. The resulting graph is Eulerian, and the algorithm outputs an Euler walk of this solution. The algorithm performs well when D is large.

The following theorem can be derived from [7].

Theorem 2.4 *The walk returned by Algorithm ShortArcs has length at most $\frac{3}{2}A + 2D$. The walk returned by Algorithm LongArcs has length at most $3A + D$.*

2.3 The rural postman problem

Let $E' \subseteq E$ be a specified subset of *special edges*. The *rural postman problem* (RPP) is to compute a shortest walk that visits all the edges in E'. The Chinese postman problem is a special case of RPP in which $E' = E$, i.e., 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 for 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 to be a straightforward generalization of Christofides algorithm for TSP. As indicated by Frederickson [6], it produces a $\frac{3}{2}$-approximation algorithm for the problem (see the survey [5] for more details and [11] 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. This in fact guarantees that the walks are actually tours since each vertex is visited once. These instances are themselves CTSP instances with $|V_i| = 2$ $(i = 1, ..., k)$ and given end vertices (starting and ending vertices, respectively). However, we note that if for some constant M, $|V_i| \leq M (1 \leq i \leq k)$ 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 the CTSP in which the starting vertex s_i and ending vertex t_i 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 s_i to the end vertex t_i 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 s_i to t_i. We get an instance of the stacker crane problem: Find a tour that includes all the special directed arcs. From this solution to SCP, we replace each arc (s_i, t_i) by the path $path_i$ computed within that cluster. See Figure 2 for a sample execution of the algorithm for three clusters with given starting and ending vertices.

Figure 1 describes the algorithm in detail.

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

$$l(T_m) \leq \frac{21}{11}\ OPT < 1.9091 OPT.$$

Proof: The algorithm consists of solving two subproblems: TSPP with given end vertices, and SCP. Since there are two solutions for each of these subproblems, there are four possible solutions to our problem, and the algorithm selects a solution of minimum length. 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 $(s_i, t_i), i = 1, \ldots, k$. By Theorem 2.1, the lengths of the two solutions computed in Step 1 for the TSPP with given end vertices are at most $2L - D$ and $\frac{3}{2}L + \frac{1}{2}D$. By Theorem 2.4, the solution T_s to the SCP that is computed by *Algorithm ShortArcs* is at most $\frac{3}{2}A + 2D$. T'_s is obtained by replacing arcs with a total length of D by the shorter of the paths obtained in Step 1. Therefore,

$$l(T'_s) \leq \min\Big(\frac{3}{2}A + 2L, \frac{3}{2}A + \frac{3}{2}L + \frac{3}{2}D\Big).$$

Algorithm GivenST

Input

1. A graph $G = (V, E)$, $|V| = n$, with weights $l(e)$ $e \in E$.

2. A partition of V into clusters $V_1, ..., V_k$.

3. Start and end vertices s_i and t_i for each cluster V_i, $i = 1, \ldots, k$.

returns *A clustered tour.*

begin

Step 1

For each V_i, compute $path_i$, a Hamiltonian path with end vertices s_i and t_i.

Step 2

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

Step 3

In T, replace the special directed arc (s_i, t_i) by $path_i$, $i = 1, ..., k$.

Step 4

return *the resulting tour T_m.*

end *GivenST*

Figure 1: Procedure *GivenST*

Similarly, the length of T_l, the SCP tour obtained by *Algorithm LongArcs* is at most $3A + D$, and using similar techniques as in the above case we obtain,

$$l(T'_l) \leq \min\Big(3A + 2L - D, 3A + \frac{3}{2}L + \frac{1}{2}D\Big).$$

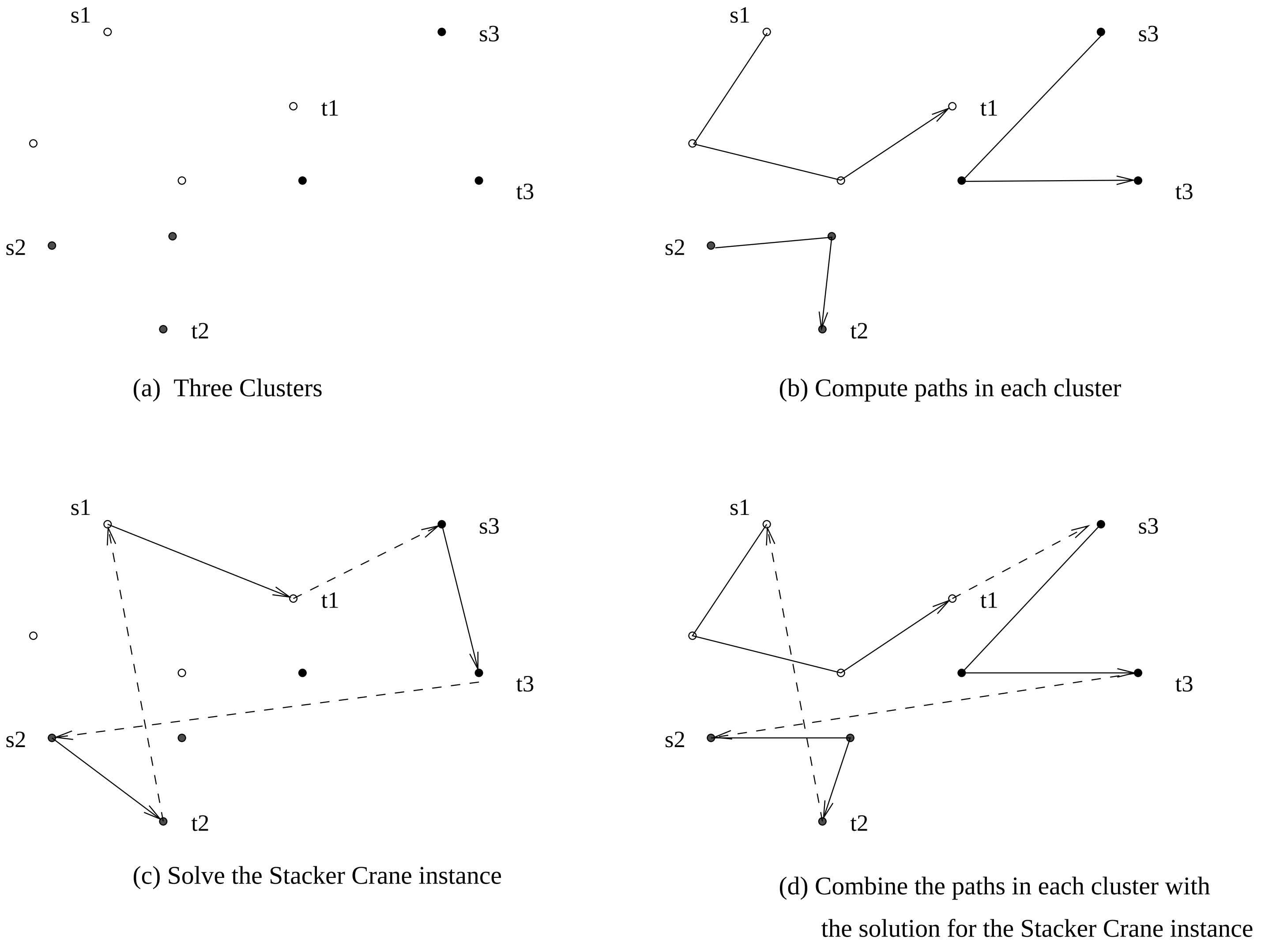
The algorithm returns the shorter tour among T'_s and T'_l. Using the fact that the minimum of a set of quantities is at most any convex combination, we get,

$$\begin{aligned} l(T_m) &\leq \min\Big(\frac{3}{2}A + 2L, \frac{3}{2}A + \frac{3}{2}L + \frac{3}{2}D, 3A + 2L - D, 3A + \frac{3}{2}L + \frac{1}{2}D\Big) \\ &\leq \frac{7}{11}\Big(\frac{3}{2}A + 2L\Big) + \frac{1}{11}\Big(\frac{3}{2}A + \frac{3}{2}L + \frac{3}{2}D\Big) + \frac{2}{11}\Big(3A + 2L - D\Big) + \frac{1}{11}\Big(3A + \frac{3}{2}L + \frac{1}{2}D\Big) \\ &= \frac{21}{11}\Big(A + L\Big) = \frac{21}{11}\ OPT. \end{aligned}$$

■

4 Given end vertices

In this section, we consider the CTSP in which for each i, $i = 1, \ldots, k$, we are given the two end vertices, $\{s_i^1, s_i^2\}$, 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 s_i^1 and s_i^2, but we are free to select any one of them



(a) Three Clusters

(b) Compute paths in each cluster

(c) Solve the Stacker Crane instance

(d) Combine the paths in each cluster with the solution for the Stacker Crane instance

Figure 2: Sample execution of Algorithm GivenST.

as the start vertex and the other vertex as the end vertex. We modify *Algorithm GivenST* by executing *Algorithm RuralPostman* rather than *Algorithm StackerCrane*, since $path_i$ can be oriented in any direction. The solution obtained consists of the special edges $(s_i^1, s_i^2), i = 1, \ldots, k$ and other undirected edges between the end vertices. We replace the special edges by the corresponding paths between s_i^1 to s_i^2 computed in Step 1. Figure 3 describes the algorithm in detail.

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 $V_1, ..., V_k$.

3. In each cluster i, $i = 1, ..., k$, two vertices s_i^1 and s_i^2.

returns *A clustered tour.*

begin

Step 1

For each V_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 $\{(s_i^1, s_i^2) \mid i = 1, ..., k\}$ to obtain tour T.

Step 3

In T, replace the special edge (s_i^1, s_i^2) by $path_i$, $i = 1, ..., k$.

Step 4

return *the resulting tour T_m.*

end *GivenEnds*

Figure 3: Procedure *GivenEnds*

Theorem 4.1 *Let T_m be the tour returned by Algorithm GivenEnds in Figure 3. Then*

$$l(T_m) \leq \frac{9}{5} \; OPT.$$

Proof: The proof is similar to that of Theorem 3.1, and we provide only a sketch. The length of all the paths computed in Step 1 is at most $\min(2L - D, \frac{3}{2}L + \frac{1}{2}D)$. The length of the rural postman tour T_r is at most $\frac{3}{2}OPT_{RP} = \frac{3}{2}(A + D)$. It is also bounded by the length of the tour T_l computed by *Algorithm LongArcs2* which is at most $3A + D$. As before we replace in the above tours the special edges of total length D by the set of paths $path_i$ $i = 1, ..., k$ and get solutions T_r' and T_l' satisfying

$$\begin{aligned} l(T_r') &\leq \min\Big(\frac{3}{2}A + 2L - \frac{1}{2}D, \frac{3}{2}A + \frac{3}{2}L + D\Big), \\ l(T_l') &\leq \min\Big(3A + 2L - D, 3A + \frac{3}{2}L + \frac{1}{2}D\Big). \end{aligned}$$

$$
\begin{aligned}
l(T_m) &= \min\Big(l(T_r'), l(T_l')\Big) \\
&\leq \min\Big(\frac{3}{2}A + 2L - \frac{1}{2}D, \frac{3}{2}A + \frac{3}{2}L + D, 3A + 2L - D, 3A + \frac{3}{2}L + \frac{1}{2}D\Big) \\
&\leq \frac{3}{5}\Big(\frac{3}{2}A + 2L - \frac{1}{2}D\Big) + \frac{1}{5}\Big(\frac{3}{2}A + \frac{3}{2}L + D\Big) + \frac{1}{5}\Big(3A + \frac{3}{2}L + \frac{1}{2}D\Big) \\
&= \frac{9}{5}\ \Big(A + L\Big) = \frac{9}{5}\ OPT.
\end{aligned}
$$

■

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. This was the problem considered in [2] and a 3.5 approximation was given for it. 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 a clustered tour. In the second heuristic, we select end vertices t_i for each cluster based on certain criteria, and then find a solution using *Algorithm GivenST* (Section 3) that finds a clustered tour when the start and end vertices of each cluster are given. Figure 4 describes the algorithm.

Algorithm GivenStart

Input

1. A graph $G = (V, E)$, $|V| = n$, with weights $l(e)$ $e \in E$.

2. A partition of V into clusters $V_1, ..., V_k$.

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 S of just the starting vertices $\{s_1, s_2, \ldots, s_k\}$.
Let T_s be a tour obtained by shortcutting $S \cup (\cup_{i=1}^{k} T_i)$.

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_l with start vertices s_i and end vertices t_i.

Step 3

return *the shorter of the tours T_s and T_l.*

end *GivenStart*

Figure 4: Procedure *GivenStart*

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

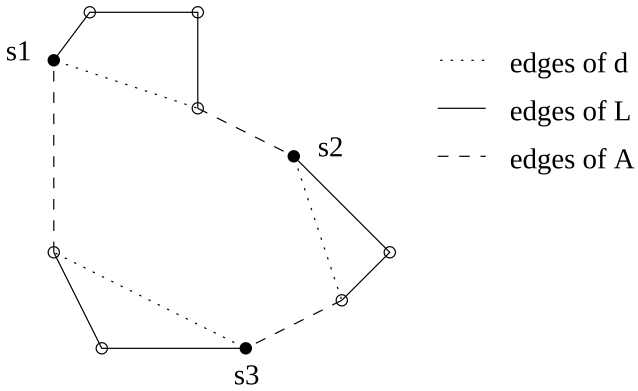


Figure 5: An optimal solution with three clusters is illustrated.

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 $l(s_i, t_i)$, $d \leq D$.

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

$$l(T_s) \leq \frac{3}{2} OPT + 2d.$$

Proof: The Hamiltonian paths through each cluster can be converted to cycles by adding an edge that connects s_i 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 $\frac{3}{2}L + \frac{1}{2}d$. There exists a tour of 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 S computed in Step 1 is at most $\frac{3}{2}(A + d)$. Tour T_s is obtained by combining all the above tours and its length is at most $\frac{3}{2}(L + A) + 2d = \frac{3}{2}OPT + 2d$. ∎

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

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

Proof: The tour T_l is obtained by running *Algorithm GivenST* after choosing an end vertex t_i for each cluster. This algorithm returns the best of two tours that it computes. We prove that if one just took the tour from Step 2 of *Algorithm GivenST* (see Figure 1), we get the bound stated in the lemma. We observe that OPT can be shortcut to obtain a feasible solution to the corresponding stacker crane problem and therefore if *Algorithm ShortArcs* is applied to this problem, we get a tour whose length is at most $\frac{3}{2}OPT + \frac{1}{2}D$. In this tour, we replace one copy of each arc (s_i, t_i) by a path from s_i to t_i. We showed in Theorem 2.1 that the lengths of such paths is at most $2\sum_{i=1}^{k} MST(G_i) - D \leq 2L - D$, where G_i is the subgraph induced by V_i. Hence the total length of the tour obtained is $(\frac{3}{2}OPT + \frac{1}{2}D) - D + (2L - D) = \frac{3}{2}OPT + 2L - \frac{3}{2}D \leq \frac{3}{2}OPT + 2L - \frac{3}{2}d$ (since $d \leq D$). ■

Theorem 5.3 *Let T_m be the clustered tour returned by Algorithm GivenStart in Figure 4. Then,*

$$l(T_m) \leq 2\frac{9}{14}\ OPT < 2.643\ OPT.$$

Proof: If $d \leq \frac{4}{7}L$, then by Lemma 5.1 and the obvious inequality $L \leq OPT$,

$$l(T_s) \leq \frac{3}{2}OPT + \frac{8}{7}L \leq 2\frac{9}{14}\ OPT.$$

If $d > \frac{4}{7}L$, then the same inequality holds by Lemma 5.2. Since the algorithm chooses the shorter 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. For this version a 1.5 approximation was given for 2 clusters [10]. 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 s_i and t_i such that $l(s_i, t_i)$ 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 $D = \sum_{i=1}^{k} l(s_i, t_i)$.

Lemma 6.1 *Let T_s be the tour computed in Step 1 of Algorithm UnspecifiedEnds. Then,*

$$l(T_s) \leq \frac{3}{2}OPT + \frac{1}{2}L + 2D.$$

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 $V_1, ..., V_k$.

returns *A clustered tour*

begin

Step 1

Apply a TSPP 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) $i = 1, ..., k$ and
let T_s be the tour obtained by replacing the special edge (a_i, b_i) by $path_i$ $i = 1, ..., k$.

Step 2

In each cluster find vertices s_i and t_i that maximize $l(s_i, t_i)$.
Let Algorithm GivenEnds return tour T_l with end vertices $\{s_i, t_i\}$.

Step 3

return *the shorter of the tours T_s and T_l.*

end *UnspecifiedEnds*

Figure 6: Procedure *UnspecifiedEnds*

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

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

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

$$\frac{3}{2}\{A + \sum_{i=1}^{k}(l(u_i, a_i) + l(a_i, b_i) + l(b_i, v_i))\}.$$

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

$$\begin{aligned}
l(T_s) &\leq \frac{3}{2}\Big(A + \sum_{i=1}^{k}(l(u_i, a_i) + l(a_i, b_i) + l(b_i, v_i))\Big) - \sum_{i=1}^{k} l(a_i, b_i) + \frac{3}{2}L \\
&\leq \frac{3}{2}OPT + \frac{1}{2}\sum_{i=1}^{k}(l(u_i, a_i) + l(a_i, b_i) + l(b_i, v_i)) + \sum_{i=1}^{k}(l(u_i, a_i) + l(b_i, v_i)) \\
&\leq \frac{3}{2}OPT + \frac{1}{2}\sum_{i=1}^{k}(l(u_i, a_i) + l(a_i, b_i) + l(b_i, v_i)) + 2D
\end{aligned}$$

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

where the last inequality follows from (1). ■

Lemma 6.2 *Let T_l be the tour computed in Step 2 of Algorithm UnspecifiedEnds. Then,*

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

Proof: The proof is similar to that of Lemma 5.2, except we apply *Algorithm RuralPostman* instead of *Algorithm StackerCrane*, since we can choose either of $\{s_i, t_i\}$ as the start vertex. We observe that OPT can be shortcut to obtain a feasible solution to the corresponding RPP. Therefore, the RPP solution computed is at most $\frac{3}{2}OPT$. From this solution, we replace each special edge (s_i, t_i) by a path connecting s_i and t_i, that includes all vertices in V_i. The length of these paths is at most $2\sum_{i=1}^{k} MST(V_i) - D \leq 2L - D$ (Theorem 2.1). Hence the length of the tour is at most $\frac{3}{2}OPT - D + (2L - D) = \frac{3}{2}OPT + 2L - 2D$. ■

Theorem 6.3 *Let T_m be the tour returned by Algorithm UnspecifiedEnds in Figure 6. Then,*

$$l(T_m) \leq 2\frac{3}{4}OPT.$$

Proof: If $2D \leq \frac{3}{4}L$, then by Lemma 6.1 and the inequality $L \leq OPT$,

$$l(T_s) \leq \frac{3}{2}OPT + \frac{5}{4}L \leq 2\frac{3}{4}OPT.$$

If $2D > \frac{3}{4}L$, then the same inequality holds by Lemma 6.2. Since the algorithm chooses the shorter of the tours T_s and T_l, the theorem follows. ■

7 Constant number of clusters

In this section, we consider the CTSP where the number of clusters, k, is a constant. The case $k = 1$ is the TSP. Gendreau, Laporte and Hertz [10] considered the case of 2 clusters with unspecified end vertices, and gave a $\frac{3}{2}$-approximation algorithm for it. We show that the CTSP with given end vertices is equivalent to TSPP with given end vertices. Hence we can obtain the obvious $\frac{5}{3}$-approximation for this case by using the $\frac{5}{3}$ 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.

Theorem 7.1 *If there exists an α-approximation algorithm to the TSPP with given end vertices then there exists an α-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 starting vertex in each cluster. For a given set of start vertices in each cluster, we construct an auxiliary directed graph 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 s_i, traverses all the other vertices of V_i, and ends at s_j. To compute these arc lengths, we use the α-approximation algorithm for TSPP with given end vertices. We find an optimal TSP 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 α 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^k possible orderings. ■

Corollary 7.2 *There exists a $\frac{5}{3}$-approximation algorithm to CTSP for constant number of clusters (in all the variants we consider for this paper).*

Remark 7.3 If there exists an α-approximation algorithm for TSPP with given end vertices there exists an α-approximation algorithm for CTSP when the order of the clusters and the start and end vertices are given. This follows by finding a path between each pair of end vertices inside each cluster using the approximation algorithm for the TSPP, and connecting the paths using the given order. If the order of the clusters is specified but the end vertices are not specified, Anily, Bramel and Hertz [1] give a $\frac{5}{3}$-approximation algorithm.

Theorem 7.4 *If there exists an α-approximation algorithm for CTSP with given end vertices and k clusters, for some $k \geq 2$, then there exists an α-approximation algorithm for the TSPP with given end vertices.*

Proof: Clearly a CTSP algorithm for $k > 2$ can be used to get the same performance guarantee for $k = 2$. Hence we will prove the claim assuming $k = 2$. Let $G = (V, E)$ and end vertices s and t be given as an instance of TSPP. We construct an instance of CTSP with two clusters as follows: we make two copies of G, namely G_1 and G_2. The copy of vertex v in G_i is identified as v_i. Distances between vertices of the same copy is the same as the distances in G. We also set $l(t_1, s_2) = l(t_2, s_1) = 0$. Distances between vertices in different copies are computed by routing the path through the linking vertices. Thus, for $u_1 \in G_1$ and $v_2 \in G_2$, $l(u_1, v_2) = \min(l(u_1, t_1) + l(s_2, v_2),\ l(v_2, t_2) + l(s_1, u_1))$.

The vertices of each copy of G form a cluster. Any solution to the CTSP with given end vertices consists of the union of two Hamiltonian paths between s and t. Therefore, we get an α-approximation for the TSPP with given end point problem by computing an α-approximation of the instance of the CTSP, and selecting the shorter of the two $s - t$ paths. ■

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

Proof: Clearly a CTSP algorithm for $k > 4$ can be used to get the same performance guarantee for $k = 4$. Hence we will 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 CTSP with four clusters as follows: we make two copies of $V - \{s, t\}$, namely V_1 and V_2. The copy of vertex v in V_i is identified as v_i. Distances between vertices of the same copy is the same as the distances in G. Distance from $s(t)$ to v_i is the same as the distance from $s(t)$ to v. Distances between vertices in different copies are computed by routing the path through s or t. Thus, for $u_1 \in V_1$ and $v_2 \in V_2$, $l(u_1, v_2) = \min(l(u_1, t) + l(t, v_2),\ l(u_1, s) + l(s, v_2))$.

We have four clusters, namely $\{s\}, \{t\}, \{V_1\}, \{V_2\}$.

STILL NEED TO ADD THE PROOF THAT SHOWS THAT A SOLUTION TO TSPP OF LENGTH L TRANSLATES TO A SOLUTION FOR CTSP OF LENGTH 2L AND VICE VERSA. Involves some case analysis...since the CTSP solution could be s,V1,t,V2 or s,t,V1,t,V2,s etc ■

Remark 7.6 The approximation given in [10] is for the CTSP with unspecified end vertices with two clusters. The main point of the last theorem is that we cannot obtain such a bound for four or more clusters, unless the bound for TSPP of $\frac{5}{3}$ is improved.

References

[1] S. Anily, J. Bramel and A. Hertz, "A $\frac{5}{3}$-approximation algorithm for the clustered traveling salesman tour and path problems," Manuscript, December 1997.

[2] E. Arkin, R. Hassin and L. Klein, "Restricted delivery problems on a network," *Networks* **29** (1997), pages 205-216.

[3] J. A. Chisman, "The clustered traveling salesman problem," *Computers & Operations Research* **2** (1975), pages 115-119.

[4] N. Christofides, "Worst-case analysis of a new heuristic for the traveling salesman problem," Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University (1976).

[5] H. A. Eiselt, M. Gendreau and G. Laporte, "Arc routing problems, part II: The rural postman problem," *Operations Research* **43** (1995), pages 399-414.

[6] G. N. Frederickson, "Approximation algorithms for some postman problems," *J. Assoc. Comput. Mach.* **26** (1979), pages 538-554.

[7] G. N. Frederickson, M. S. Hecht and C. E. Kim, "Approximation algorithms for some routing problems," *SIAM J. Comput.* **7** (1978), pages 178-193.

[8] J. A. Hoogeveen, "Analysis of Christofides' heuristic: Some paths are more difficult than cycles," *Operations Research Letters*, **10** (1991), pages 291-295.

[9] M. Gendreau, A. Hertz and G. Laporte, "The traveling salesman problem with backhauls," *Computers and Operations Research*, **23** (1996), pages 501-508.

[10] M. Gendreau, G. Laporte and A. Hertz, "An approximation algorithm for the traveling salesman problem with backhauls," *Operations Research*, **45** (1997), pages 639-641.

[11] K. Jansen, "An approximation algorithm for the general routing problem," *Information Processing Letters*, **41** (1992), pages 333-339.

[12] D. S. Johnson and C. H. Papadimitriou, "Performance guarantees for heuristics," E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* Wiley (1985), pages 145-180.

[13] K. Jongens and T. Volgenant, "The symmetric clustered traveling salesman problem" *European Journal of Operational Research* **19** (1985), pages 68-75.

[14] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids* Holt, Reinehart and Winston (1976).

[15] F. C. J. Lokin, "Procedures for traveling salesman problems with additional constraints," *European Journal of Operational Research* **3** (1978), pages 135-141.

[16] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization* Wiley & Sons (1988).