The Errand Scheduling Problem

Petr Slavík
Department of Mathematics, SUNY
Buffalo, NY 14214, USA
E-mail: `slavik@math.buffalo.edu`

March 14, 1997

Abstract

We consider the following natural *errand scheduling problem* (ESP). We must perform a set of errands at the nodes of an edge-weighted graph, and each errand can only be performed at a subset of the nodes. What is the shortest tour that allows us to complete all the errands? We also consider the closely related *tree cover problem* (TCP), in which we seek a tree of minimum length that "covers" all errands. Both problems generalize a number of well-known problems in combinatorial optimization and have a wide range of applications including job scheduling on multi-state CNC machines and VLSI design.

We focus on the case in which the graph is a weighted tree; this trivially generalizes the famous *set cover problem*. Under the assumption that no errand can be performed in "too many" nodes, we obtain an algorithm that asymptotically matches the best possible approximation ratio for *set cover* and approximates both *errand scheduling* and *tree cover* within $O(\log m)$, where m is the total number of errands. Our algorithm is based on "tree stripping"–a technique specifically designed to round the solution to the LP relaxation of *tree cover*, but is hopefully useful for approximating other related problems.

In the second part of the paper, we (briefly) discuss the *errand scheduling* and *tree cover* problems on a general weighted graph with the restriction that each errand can be performed at at most ρ nodes. We show that in this case, ESP can be approximated to within $3\rho/2$ and TCP to within ρ.

Keywords: Approximation Algorithms, Errand Scheduling, Generalized Steiner Tree, LP Relaxation, Set Cover, Traveling Salesman, Tree Stripping

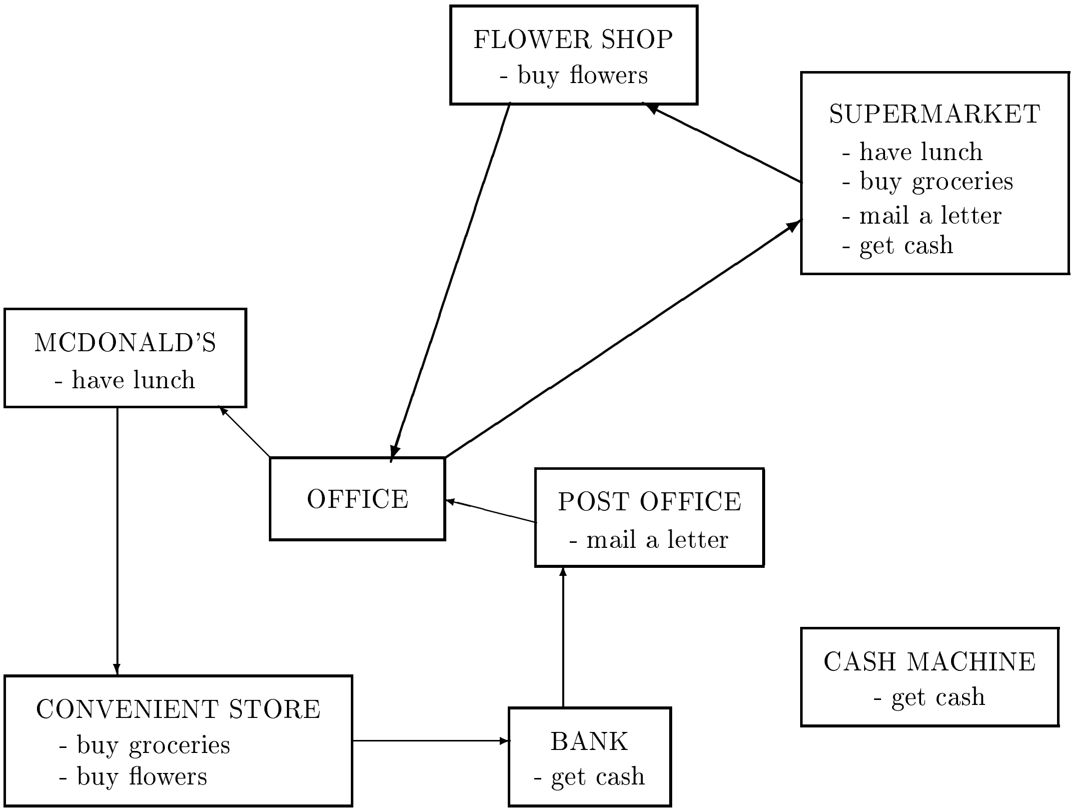

Figure 1: Errands of professor Perfect with two possible solutions.

1 Introduction

Consider the following example (modified from [18] and [5]) – see Figure 1. Professor Peter Perfect decides to run a few errands during his lunch break. He needs to *have lunch, buy some groceries, buy flowers for his fiancee, mail a letter, get some cash*, and then return back to his office. Each errand can be done at several different locations. In particular, he can *have lunch* at *McDonald's* or at the *supermarket*, *buy groceries* at *the local convenient store* or at *the supermarket*, *buy flowers* at *the flower shop* or at *the convenient store*, *mail a letter* at *the supermarket* or at *the post office*, and *get cash* at *the supermarket*, at *a nearby bank*, or at *a cash machine*. Since he always tries to do things in the best possible way, he decides to first create a plan how to run these errands in minimum time. Figure 1 shows two (thick vs. thin arrows) of many possible ways of completing all errands.

Assuming that each errand takes the same time at any given location, and that there is no delay between two errands performed at the same location, professor Perfect "simply" has to choose a partial tour through some of the stores starting and ending at his office, such that all errands can be completed and the total traveling time is minimal.

More formally, let U be a set (of tasks/errands) of size m, and let G be an edge-weighted graph (of locations). Associated with each vertex i in G is a set $S_i \subset U$ (the errands that can be performed at location i).

The ERRAND SCHEDULING problem is to find a shortest partial tour through G such that each element of U is contained in at least one "visited" subset (that is each task can be completed). We allow repetitions of vertices in the partial tour.

The TREE COVER problem is to find a subtree of G of minimum length such that each element of U is contained in at least one "visited" subset.

For simplicity we assume that the starting vertex of the tour and the root of the subtree (in our example the office) is also given. This assumption can be easily removed by running the approximation algorithm n-times with all possible choices for the starting vertex.

1.1 Our results

We give the first approximation algorithms for the above problems with non-trivial worst-case performance bounds in two special cases. Both significantly improve the results of Ihler in [21, 23].

In Section 2 we consider the case where the graph of locations is a tree; here ESP and TCP are equivalent. Even though this case is quite restricted, it trivially includes the *set cover* problem–just put the subsets of errands on the leaves of a star with the distance from the center being the cost of the subset. Under quite general restrictions on the structure of the tree, we give an approximation algorithm with performance ratio $\Theta(\ln m)$–the best one can expect. Our approach is based on solving the LP relaxation of the corresponding integer programming problem and then using our "tree stripping" technique (see Section 2 for details) to transform the whole problem into finding a minimum cost set cover. We believe that the technique of tree stripping, though specifically designed for solving TCP on trees, can be used for approximating other similar problems.

In Section 3 we discuss ESP and TCP on general graphs with the restriction that each element of U appears in at most ρ covering sets (that is each errand can be completed at at most ρ locations). Using again the optimal solution to the LP relaxation of the general ERRAND SCHEDULING problem (or TCP) plus the techniques developed for solving the PRIZE-COLLECTING TSP (see for example [6] or [31]), we show that the ESP problem can be approximated within $3\rho/2$ and the TCP problem within ρ. These contain as special cases the well-known result of Christofides [9] for approximating TSP and the result of Hochbaum [20] where she proved that the SET COVER problem can be approximated to within ρ if each element is covered at most ρ times.

1.2 Related results

Even though ERRAND SCHEDULING is a very natural problem to consider, there have not been any worst-case performance bounds established for this problem, most likely because of the inherent difficulties in providing any reasonable approximations. In fact, any algorithm successful in approximating "similar" problems cannot provide a better performance ratio than $\Theta(m)$ for both ESP and TCP, even when the graph of locations is a tree. The *errand scheduling* problem is a special case of the TRAVELING PURCHASER problem discussed by Ong in [27]. Ong proposed several natural heuristics for approximating this problem and discussed their performance in terms of experimental computations on selected instances. His algorithms are combinations of greedy strategies and local improvements and do not have a good worst-case performance. As before, one can construct instances where his heuristics perform as badly as $\Theta(m)$.

The TREE COVER problem generalizes the GROUP STEINER TREE problem considered by Ihler, and also by Garg and Ravi in [14]. In fact, it can be readily shown that these two problems are equivalent. Ihler in [21, 23] analyses a heuristic whose performance ratio is no worse than $m-1$ where m is the number of tasks to be completed, and discusses a special case of the *group Steiner tree* problem which can be solved in polynomial time. In [22] he also shows that the *group Steiner tree* problem is at least as hard to approximate as the *set cover* problem, which is a rather trivial observation in our formulation of the problem as TCP. Other, quite restricted cases of the *group Steiner tree* problem are discussed in [24].

A problem somewhat similar to ESP and TCP is discussed by Arkin and Hassin in [1]. Given the same set-up as for ESP, their goal was to find a subgraph G' of G of *minimum diameter* such that the ground set is covered by subsets corresponding to vertices in G'. Unfortunately, their algorithm does not help us to obtain any reasonable approximations to either TCP or ESP.

In [5], Bhatia, Khuller, and J. Naor consider a problem somewhat related to ESP. Their LOADING TIME SCHEDULING problem is also a generalization of the SET COVER problem, but in a different direction. In their problem, there are precedence constraints limiting which location can be chosen at any given time but, on the other hand, instead of considering distances *between* locations they assign a weight to *each* location. Because of these principal differences, their approach to approximating the *loading time scheduling* problem cannot be used for our problems.

Recently a class of optimization problems including k-TSP, k-MST, PRIZE-COLLECTING TSP, and other related problems acquired much attention among theory researchers and this led to significant improvements in both performance guarantees and complexity of approximation algorithms for these problems. See for example [2], [3], [6], [7], [8], [13], [16], [29], [30], etc. The main difference between these problems and ESP(TCP) is the fact that the optimal solutions to k-TSP, k-MST, or *prize-collecting* TSP have to satisfy some *quantitative* requirements–finding the shortest partial tour through k vertices, finding a minimum weight spanning tree on k nodes, finding a partial tour through some vertices of a graph such that the length of the tour plus the penalties from unvisited vertices are minimized; whereas the condition on the optimal solution to ESP or TCP is of a *qualitative* nature–the elements of U have to be covered by the sets associated with the visited vertices. Thus, in any of the above mentioned problems, the more vertices we choose, the closer we are to satisfy the constraints; on the other hand, in the ESP(TCP) problem, it might happen that we select many "seemingly good" vertices but still remain far from completing all the tasks. Therefore these problems are also quite different from ESP and TCP.

The *errand scheduling* and *tree cover* problems generalize and contain as special cases several well-known optimization problems, among them the SET COVER problem and the TRAVELING SALESMAN problem. Recent hardness results of Lund and Yannakakis [26], Bellare, Goldwasser, Lund, and Russell [4], and Feige [11] show that no polynomial-time algorithm can approximate the SET COVER problem better than $\Omega(\ln m)$ where m is the size of the ground set, unless some unlikely collapses in the polynomial hierarchy occur. Thus, when approximating ESP, we cannot hope for a better approximation ratio than $\Theta(\ln m)$. On the other hand, the structure of *tree cover* and *errand scheduling* is more complicated than that of SET COVER hence performance ratios worse than $\Theta(\ln m)$ for approximating general TCP and ESP should be expected.

Both TCP and ESP have a host of practical applications, among them job scheduling on multi-state CNC machines (see [18] for details) and VLSI design (see [24]).

1.3 Preliminaries

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{0, 1, \dots, n\}$ (of locations). Let $U = \{t_1, \dots, t_m\}$ be the "ground set" (of errands/tasks). Associated with each vertex $i \in V$ is a set $S_i \subset U$ (the set of tasks which can be performed at the i^{th} location). We assume that $S_0 = \emptyset$ and $\bigcup S_i = U$. For each edge $e = (i, j) \in E$ let $c_e = c_{(i,j)}$ be the distance between vertices i and j. We assume that the distance function is symmetric and satisfies the triangle inequality. A tour through the vertices of some set $V' \subset V$ is called a *covering tour* if $0 \in V'$ and $\bigcup_{i \in V'} S_i = U$. The ERRAND SCHEDULING problem (ESP) is to find a covering tour of minimum length.

Closely related to ESP is the TREE COVER problem (TCP). Let $T = (V', E')$ be a subgraph of G which is a tree. We call T a *covering tree* if $0 \in V'$ and $\bigcup_{i \in V'} S_i = U$. The TREE COVER problem is to find a covering tree of minimum length.

We denote by $A = (a_{ki})$ the m by n incidence matrix of the sets S_i, $i = 1, \dots, n$, that is

$$a_{ki} = \begin{cases} 1 & \text{if } t_k \in S_i \\ 0 & \text{otherwise.} \end{cases}$$

We will denote by OPT the cost of an optimal solution to a problem under discussion and by APP the cost of the approximate solution.

2 ESP and TCP on trees

Let us now consider the case when the graph $G = (V, E)$ is a tree. Any partial tour in G is clearly at least twice as long as the subtree T determined by the vertices visited (the tour passes through each edge of T at least twice). On the other hand, given a subtree T of G, we can use depth-first search to construct a tour through vertices of T twice as long as T. Therefore $length$(minimum covering tour) $=$ $2\,length$(minimum covering tree) and solving the ERRAND SCHEDULING problem is equivalent to solving the TREE COVER problem. Hence, in what follows, we will seek an approximation to the optimal solution of TCP only, under the restriction that $G = (V, E)$ is a tree.

Even though this problem is simpler than the general TCP, it of course remains NP hard. In fact, any instance of the (weighted) SET COVER problem can be reduced to an instance of the TCP on trees. To see this, given any collection $\{S_1, S_2, \ldots, S_n\}$ of subsets of U with costs $\{c_1, c_2, \ldots, c_n\}$, simply set G to be a star with center 0 and leaves $1, 2, \ldots, n$. Then set $S_o = \emptyset$, and for all $i = 1, \ldots, n$ associate each leaf i with a subset S_i and define the cost of an edge from 0 to i to be c_i–see Figure 2. Thus even in the simple case when G is a tree, we cannot in general approximate the length of a minimum covering tree better than $\Omega(\log m)$.

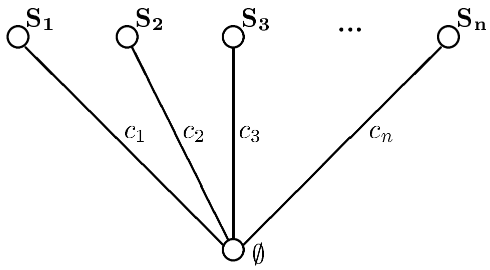

Figure 2: SET COVER is a special case of TCP on trees

We will regard G as a rooted tree with root 0. Let $\pi(i)$ be the parent of node $i \in V$, $\pi(0) = \text{NIL}$. Since each edge joins some node i and its parent, we can set $e_i = (i, \pi(i))$ and $c_i = c_{e_i} = c_{(i,\pi(i))}$ for all $i = 1, 2, \ldots, n$. For simplicity, set $c_o = 0$.

2.1 Polynomially solvable cases of TCP on trees

Consider first the case when graph G is a "line-like" tree, that is all nodes have either degree 1 or 2–see Figure 3.

Node 0 splits G into two parts. Without loss of generality, we can enumerate the nodes in one part as $\{1, 2, \ldots, n_R\}$ and in the other part as $\{-1, -2, \ldots, -n_L\}$ where $n_R + n_L = n$.

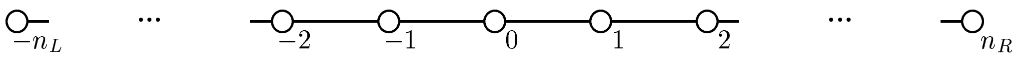

Figure 3: TCP on line

Any subtree of G containing 0 is uniquely determined by its two end-points, n_1 and n_2, where $n_1 \leq 0 \leq n_2$, and contains all nodes from the interval $[n_1, n_2]$. We have only $(n_L + 1)(n_R + 1) = O(n^2)$ such intervals, hence we can check all of them and determine which one corresponds to the covering tree of minimum cost. Using a simple algorithm, this can be done in $O(nm)$ time.

The case of a line-like tree can be generalized in a straightforward way to the case when tree G consists of K "line segments" with 0 as one end-point. A little more thought might be necessary to convince oneself that even in the case of a general tree having constant number K of leaves, one can enumerate all $O(n^K)$ possible subtrees containing 0, check each one of them (with possible shortcuts) to determine whether all elements of U are covered, and then choose a covering tree of minimum length. Using a simple algorithm, this can be done in $O(n^K m)$ time.

2.2 Integer program and LP relaxation

For any tree $T \subset G$, set $y_i = 1$ if $i \in T$ and 0 otherwise. We can alternately formulate TCP on trees as the following integer program.

Problem $\mathbf{ITCP_1}$:

$$
\begin{aligned}
OPT_1 \;=\; \text{minimize} \quad & \sum_{i \in V \setminus \{0\}} c_i y_i, \\
\text{subject to} \quad & \text{(a)} \quad y_o = 1, \\
& \text{(b)} \quad \sum_{i=1}^{n} a_{ki} y_i \geq 1, \quad \text{for all } k = 1, \ldots, m, \\
& \text{(c)} \quad y_i \leq y_{\pi(i)}, \qquad \text{for all } i \in V \setminus \{0\}, \\
& \text{(d)} \quad y_i \in \{0, 1\}, \qquad \text{for all } i \in V.
\end{aligned}
\tag{1}
$$

Clearly, the original formulation of TCP and the integer program ITCP_1 are equivalent. Condition (1a) guarantees that node 0 is always selected. Condition (1b) assures that the sets corresponding to the selected nodes cover the ground set. Condition (1c) guarantees that the selected nodes form a tree. The structure of (1) is particularly simple because of the fact that graph G is a tree. The general case (see Section 3) is more complicated.

Relaxing condition (1d) yields the following linear program.

Problem $\mathbf{LTCP_1}$:

$$
\begin{aligned}
OPT_1^* \;=\; \text{minimize} \quad & \sum_{i \in V \setminus \{0\}} c_i y_i, \\
\text{subject to} \quad & \text{(a)} \quad y_o = 1, \\
& \text{(b)} \quad \sum_{i=1}^{n} a_{ki} y_i \geq 1, \quad \text{for all } k = 1, \ldots, m, \\
& \text{(c)} \quad y_i \leq y_{\pi(i)}, \qquad \text{for all } i \in V \setminus \{0\}, \\
& \text{(d)} \quad y_i \in [0, 1], \qquad \text{for all } i \in V.
\end{aligned}
\tag{2}
$$

We can solve Problem LTCP_1 using any of the known polynomial time algorithms for solving linear programming problems (ellipsoid method, Karmarkar's method, etc.) and obtain the optimal solution $(\mathbf{y}^*; OPT_1^*) = (y_o^*, y_1^*, \ldots, y_n^*; OPT_1^*)$. But can we use $(\mathbf{y}^*; OPT_1^*)$ to obtain a reasonable approximation of the minimum covering tree? In what follows we will show that this is indeed the case.

Let $\mathbf{y} = (y_o, y_1, \ldots, y_n) \in \mathbf{R}_+^{n+1}$. Define $V_\mathbf{y} = \{i \in V \mid y_i > 0\}$. For any $t_k \in U$ define $cov_\mathbf{y}(t_k)$, the "covering number" of t_k with respect to $\mathbf{y}$, as

$$cov_\mathbf{y}(t_k) = \sum_{i : t_k \in S_i} y_i = \sum_{i=1}^{n} a_{ki} y_i .$$

Intuitively, $cov_\mathbf{y}(t)$ indicates how many times t is covered by fractions y_i of the sets in $V_\mathbf{y}$. If $\mathbf{y}$ is a feasible solution to (2) then conditions (2b) and (2c) imply that $cov_\mathbf{y}(t) \geq 1$ for all $t \in U$ and subgraph $T \subset G$ generated by $V_\mathbf{y}$ is a tree containing 0. In that case, we call the pair $(T, \mathbf{y})$ a *"fractional" covering tree* and define the "length" of $(T, \mathbf{y})$ as $length^*(T, \mathbf{y}) = \sum_{i=1}^{n} c_i y_i$. Thus $(T, \mathbf{y}^*)$ is a fractional covering tree of minimum length.

2.3 Tree sets

Assume without loss of generality that the vertices of tree $G = (V, E)$ are numbered in a breadth-first fashion starting at vertex 0, that is the vertices adjacent to root 0 are $1, 2, \ldots, K$. Removing edges $e_1, e_2, \ldots, e_K$ results in isolating vertex 0 and creating a forest of K subtrees $G_1, G_2, \ldots, G_K$ with roots $1, 2, \ldots, K$.

Let $X \subset U$. We say that X is a "tree set" if $X = \emptyset$ or there is an index $j \in \{1, \ldots, K\}$ and a subtree $Z \subset G_j$ with $j \in Z$ such that $X = \bigcup_{i \in Z} S_i$. We define $C(X)$, the cost of tree set X, to be $C(X) = \sum_{i \in Z} c_i$. Set $C(\emptyset) = 0$.

For any subtree T of G containing 0, set $T_j = T \cap G_j$ and $Y_j = \bigcup_{i \in T_j} S_i$. Then T_j is either empty or is a subtree of G_j with $j \in T_j$; hence Y_j is a tree set, and the cost of tree T_j satisfies $cost(T_j) = \sum_{i \in T_j} c_i = C(Y_j)$ for $j = 1, \ldots, K$.

If T is a covering tree then

$$\bigcup_{j=1}^{K} Y_j = \bigcup_{j=1}^{K} \bigcup_{i \in T_j} S_i = \bigcup_{i \in T} S_i = U,$$

hence $\{Y_1, Y_2, \ldots, Y_K\}$ covers U. Also,

$$cost(\{Y_1, Y_2, \ldots, Y_K\}) = \sum_{j=1}^{K} C(Y_j) = \sum_{j=1}^{K} cost(T_j) = length(T).$$

Hence we have proved the following.

Lemma 1 *Any covering tree $T \subset G$ induces a cover $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_K\}$ of U by tree sets such that*

$$length(T) = cost(\mathbf{Y}).$$

□

We also have the following counterpart of Lemma 1.

Lemma 2 *Any cover $\mathbf{X} = \{X_1, X_2, \ldots, X_R\}$ of U by tree sets induces a covering tree $T \subset G$, such that*

$$length(T) \le cost(\mathbf{X}).$$

Proof: Let $\mathbf{X} = \{X_1, X_2, \ldots, X_R\}$ be a collection of tree sets covering U and let $\{Z_1, \ldots, Z_R\}$ be the corresponding subtrees, that is for each $r = 1, \ldots, R$, there is an index j_r such that $Z_r \subset G_{j_r}$, $j_r \in Z_r$, and $X_r = \bigcup_{i \in Z_r} S_i$. Tree $T = \bigcup Z_r$ can be obtained in a straightforward way using the following algorithm.

Algorithm "Cover_to_Tree"

Input: A cover $\mathbf{X} = \{X_1, \ldots, X_R\}$ of U by tree sets corresponding to subtrees $\{Z_1, \ldots, Z_R\}$.

Output: A covering tree T such that $length(T) \le cost(\mathbf{X})$.

begin
  $T \leftarrow \{0\}$;
  for $r \leftarrow 1$ **to** R **do**
    $T \leftarrow T \cup Z_r$;
  while some leave $i^* \in T$ can be removed without changing the set $\bigcup_{i \in T} S_i$ **do**
    $T \leftarrow T \setminus \{i^*\}$;
  return T;
end

Clearly, the algorithm runs in time polynomial in m, n, and R; the exact time would depend on the implementation. For each $j = 1, \ldots, K$, $T_j = T \cap G_j = \bigcup_{r: j_r = j} Z_r$ is either empty or a subtree of G_j containing j. If $\{X_1, X_2, \ldots, X_R\}$ is a cover of U, then

$$\bigcup_{i \in T} S_i = \bigcup_{j=1}^{K} \bigcup_{i \in T_j} S_i = \bigcup_{j=1}^{K} \bigcup_{r: j_r = j} \bigcup_{i \in Z_r} S_i = \bigcup_{j=1}^{K} \bigcup_{r: j_r = j} X_r = \bigcup_{r=1}^{R} X_r = U,$$

thus T is a covering tree. Moreover

$$cost(T_j) = \sum_{i \in T_j} c_i \leq \sum_{r: j_r = j} \sum_{i \in Z_r} c_i = \sum_{r: j_r = j} C(X_r),$$

hence

$$length(T) = \sum_{j=1}^{K} cost(T_j) \leq \sum_{j=1}^{K} \sum_{r: j_r = j} C(X_r) = \sum_{r=1}^{R} C(X_r) = cost(\{X_1, \ldots, X_R\}).$$

Therefore T is a covering subtree of G containing 0 and

$$length(T) \leq cost(\mathbf{X}).$$

□

As an easy consequence of Lemmas 1 and 2 we have

Proposition 1 *A subtree $T \subset G$ containing vertex 0 is a minimum-length covering tree if and only if the collection $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_K\}$ of tree sets associated with T is a minimum-cost cover of U by tree sets. Moreover the length of a minimum covering tree equals the cost of a minimum cover.* □

2.4 Multi-set cover problem

Proposition 1 transforms the TREE COVER problem into the SET COVER problem. There is a major difficulty though–the number of tree sets is in general exponential! Hence, at this stage, we cannot use any of the well-known algorithms for approximating the SET COVER problem. Fortunately, we can overcome this obstacle.

Let $\{Z_\alpha\}_{\alpha \in A}$ be some enumeration of all nonempty subtrees of trees G_j with $j \in Z_\alpha$, for $j = 1, \ldots, K$. Notice that we have, in general, exponentially many subtrees. Let $\{X_\alpha\}_{\alpha \in A}$ be the corresponding tree sets (there might be possible repetitions of identical sets obtained from different subtrees) and $\{C_\alpha\}_{\alpha \in A}$ be the costs of the tree sets. Hence $X_\alpha = \bigcup_{i \in Z_\alpha} S_i$ and $C_\alpha = \sum_{i \in Z_\alpha} c_i$.

Define $b_{k\alpha}$, the multiplicity of t_k in X_α, as

$$b_{k\alpha} = \sum_{i \in Z_\alpha} a_{ki}.$$

Thus $b_{k\alpha}$ indicates how many times an element t_k is covered by the sets S_i corresponding to vertices of Z_α.

Consider the following integer program that we call the *Multi-Set Cover* problem.

Problem ITCP$_2$:

$$\begin{array}{rcll} OPT_2 & = \text{minimize} & \displaystyle\sum_{\alpha \in A} C_\alpha x_\alpha, & \\ & \text{subject to} \quad \text{(a)} & \displaystyle\sum_{\alpha \in A} b_{k\alpha} x_\alpha \geq 1, & \text{for all } k = 1, \ldots, m, \\ & \text{(b)} & x_\alpha \in \{0, 1\}, & \text{for all } \alpha \in A. \end{array} \tag{3}$$

In view of Lemmas 1 and 2 and Proposition 1, ITCP$_2$ is clearly a different formulation of the TCP problem and can be considered a (non-standard!) formulation of the SET COVER problem in terms of tree sets. The following proposition is an immediate consequence of Proposition 1.

Proposition 2 *The values OPT_1 and OPT_2 of optimal solutions to ITCP$_1$ and ITCP$_2$ are the same.* □

The LP relaxations of problem ITCP_2 is the following.

Problem $\mathbf{LTCP_2}$:

$$\begin{aligned} OPT_2^* \;=\; \text{minimize} \quad & \sum_{\alpha\in A} C_\alpha x_\alpha, \\ \text{subject to} \quad \text{(a)} \quad & \sum_{\alpha\in A} b_{k\alpha} x_\alpha \geq 1, \quad \text{for all } k = 1,\dots,m, \\ \text{(b)} \quad & x_\alpha \in [0,1], \qquad \text{for all } \alpha \in A. \end{aligned} \tag{4}$$

Let $\{x_\alpha\}_{\alpha\in A} \in \mathbf{R}_+^{|A|}$. Define $\mathbf{X} = \{X_\alpha \mid x_\alpha > 0\} = (X_1, X_2, \dots, X_L)$ and $\mathbf{x} = \{x_\alpha \mid x_\alpha > 0\} = (x_1, x_2, \dots, x_L)$ for some $L \in \mathbf{N}$. Here, we wrote X_l instead of X_{α_l} and x_l instead of x_{α_l} to simplify the notation. We will use this shortcut for the rest of the section.

For any $t_k \in U$, define $cov_{\mathbf{x}}^m(t_k)$, the "multi-covering number" of t_k with respect to $\mathbf{x}$, as

$$cov_{\mathbf{x}}^m(t_k) = \sum_{l=1}^{L} b_{kl} x_l = \sum_{\alpha\in A} b_{k\alpha} x_\alpha.$$

Thus $cov_{\mathbf{x}}^m(t)$ indicates how many times t is covered by fractions x_l of tree sets $X_1, \dots, X_L$, *including* multiplicities.

If $\{x_\alpha\}_{\alpha\in A}$ is a feasible solution to (4) then condition (4a) implies that $cov_{\mathbf{x}}^m(t_k) \geq 1$ for all $t \in U$. We call the pair $(\mathbf{X}, \mathbf{x}) = ((X_1, \dots, X_L), (x_1, \dots, x_L))$ a *"fractional multi-cover"* of U and define the "cost" of $(\mathbf{X}, \mathbf{x})$ as $cost^*(\mathbf{X}, \mathbf{x}) = \sum_{l=1}^{L} C(X_l) x_l$.

Note: Since $b_{k\alpha}$ can, in general, be greater than 1, problem LTCP_2 is quite different from the standard LP relaxation of the *set cover* problem. This has no effect on the integer solutions but it can decrease the value of the optimal fractional solution, thus increasing the "integrality gap". This makes our analysis more complicated and requires the introduction of additional restrictions on the magnitude of $b_{k\alpha}$.

2.5 Tree Stripping

Consider the following algorithm.

Algorithm "Tree_Stripping"

Input: A fractional covering tree $(T, \mathbf{y})$.

Output: A collection of tree sets $\mathbf{X} = \{X_1, \dots, X_L\}$ with costs $\mathbf{C} = \{C_1, \dots, C_L\}$ and a vector $\mathbf{x} = (x_1, \dots, x_L)$ of positive rational numbers such that $length^*(T, \mathbf{y}) = \sum C_l x_l$.

```
begin
    l ← 1;
    for j ← 1 to K do
        if T_j ≠ ∅ then
            Z_l ← T_j;
            while (y_j > 0) do
                X_l ← ⋃_{i:i∈Z_l} S_i;
                C_l ← length(Z_l);
                x_l ← the smallest value y_i among the nodes of Z_l;
                for all i ∈ Z_l do y_i ← y_i − x_l;
                l ← l + 1;
                Z_l ← subtree generated by those nodes i ∈ Z_{l−1} for which y_i > 0;
    L ← l − 1;
    return (X, C, x);
end
```

For each j, the algorithm splits tree T_j with some rational "weights" associated with its nodes into a "sum" of subtrees Z_l such that the weights of the nodes in each of these trees are the same (they all are equal to x_l)–see Figure 4. At each iteration l, the algorithm finds the maximum possible weight which can be subtracted from all nodes of the current tree Z_l without making any weight negative and "subtracts" a tree of this weight from Z_l. Since at each step, $y_i \le y_{\pi(i)}$, this results either in a smaller tree (containing root j) having at least one node less, or in an empty set. Therefore the algorithm Tree_Stripping terminates after at most $|T| - 1 \le n$ iterations, and as such, runs in time polynomial in m and n and outputs $L \le n$ tree sets. Since each Z_l is a subtree of some G_j and $j \in Z_l$, each of the sets X_l is a tree set. Clearly, $C_l = C(X_l)$.

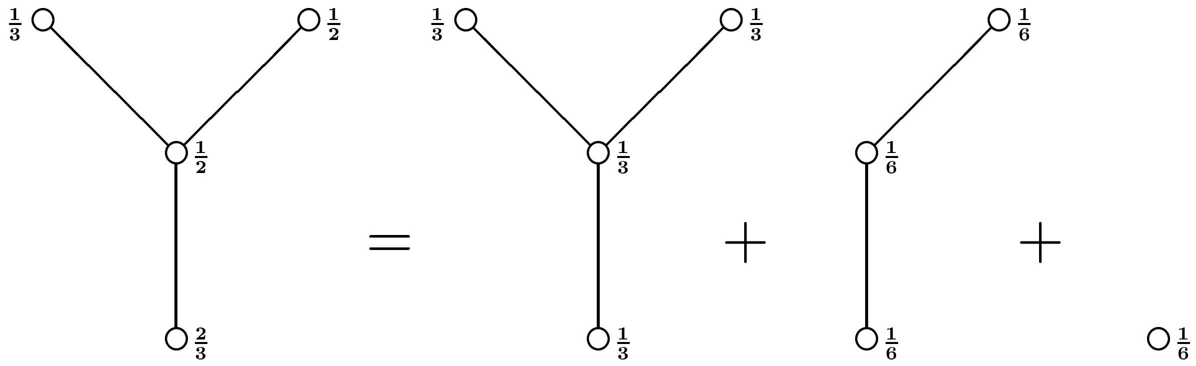

Figure 4: Tree stripping

For any $t_k \in U$, we have

$$cov^m_{\mathbf{x}}(t_k) = \sum_{l=1}^{L} b_{kl}x_l = \sum_{l=1}^{L}\sum_{i\in Z_l} a_{ki}x_l = \sum_{i\in T} a_{ki} \sum_{l:i\in Z_l} x_l = \sum_{i\in T} a_{ki}y_i = \sum_{i=1}^{n} a_{ki}y_i = cov_{\mathbf{y}}(t_k).$$

Also,

$$cost^*(\mathbf{X},\mathbf{x}) = \sum_{l=1}^{L} C(X_l)x_l = \sum_{l=1}^{L}\sum_{i\in Z_l} c_i x_l = \sum_{i\in T} c_i \sum_{l:i\in Z_l} x_l = \sum_{i\in T} c_i y_i = \sum_{i=1}^{n} c_i y_i = length^*(T,\mathbf{y}).$$

Thus we have proved the following theorem, which plays a pivotal role in developing the results of this section.

Theorem 1 *Given a fractional covering tree $(T, \mathbf{y})$, algorithm "Tree_Stripping" outputs a fractional multi-cover $(\mathbf{X}, \mathbf{x})$ of U of size $L \le n$ by tree sets such that*

$$cost^*(\mathbf{X},\mathbf{x}) = length^*(T,\mathbf{y}).$$

Moreover, for each $t \in U$,

$$cov^m_{\mathbf{x}}(t) = cov_{\mathbf{y}}(t).$$

□

Any fractional multi-cover $(\mathbf{X}, \mathbf{x})$ corresponds to a feasible solution of LTCP$_2$ with the same total cost–just set $x_\alpha = x_l$ if $\alpha = \alpha_l$ for some $l = 1, \ldots, L$ and $x_\alpha = 0$ otherwise. Thus for any feasible solution $\mathbf{y}$ of LTCP$_1$ there is a corresponding feasible solution $\{x_\alpha\}_{\alpha\in A}$ of LTCP$_2$ of the same value, that is $\sum C_\alpha x_\alpha = \sum c_i y_i$.

By reversing the procedure of tree stripping, that is simply "adding" trees Z_l corresponding to tree sets with nonzero value of x_l, one can easily show that for any feasible solution $\{x_\alpha\}_{\alpha\in A}$ of LTCP$_2$, that is for any fractional multi-cover $(\mathbf{X}, \mathbf{x})$, there is a corresponding tree cover $(T, \mathbf{y})$, that is a feasible solution $\mathbf{y}$ of LTCP$_1$ such that $\sum c_i y_i = \sum C_\alpha x_\alpha$ and $cov_{\mathbf{y}}(t) = cov^m_{\mathbf{x}}(t)$ for any $t \in U$.

Thus we have the following.

Proposition 3 $(T, \mathbf{y})$ *is a minimum-length fractional covering tree if and only if the corresponding fractional multi-cover* $(\mathbf{X}, \mathbf{x})$ *is a minimum-cost fractional multi-cover.* □

Note: The idea of tree-stripping is somewhat similar to path-stripping introduced by Raghavan and Thompson in [28].

2.6 Approximation algorithm and performance bounds

Example 1 Let $U = \{t\}$. Consider the tree in Figure 5 where $S_i = U$ for each vertex $i \in V \setminus \{0\}$. Let $\epsilon > 0$ be a very small number. We have n nonempty tree sets $X_{-1} = X_1 = \cdots = X_{n-1} = U$ determined by subtrees $T_{-1}, T_1, \ldots, T_{n-1}$ where $T_i = [1, i]$ for $i = 1, \ldots, n-1$ and $T_{-1} = \{-1\}$. The costs of tree sets are $C_i = n + (i-1)\varepsilon$ for $i = 1, \ldots, n-1$ and $C_{-1} = \sqrt{n}$. Also, $b_{1i} = b_i = i$ for $i = 1, \ldots, n-1$ and $b_{-1} = 1$.

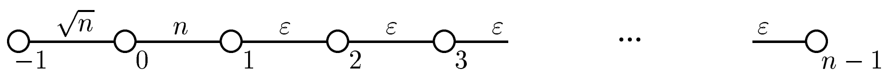

Figure 5: Tree from Example 1

The optimal solution to LTCP$_1$ is clearly $y_1^* = y_2^* = \cdots = y_{n-1}^* = \frac{1}{n-1}$, $y_{-1} = 0$ with value $OPT_1^* = \frac{n+(n-1)\varepsilon}{n-1} \approx 1$. Similarly, the optimal solution to LTCP$_2$ is $x_{n-1}^* = \frac{1}{n-1}$ and $x_i^* = 0$ otherwise, and its value is $OPT_2^* = \frac{n+(n-1)\varepsilon}{n-1} \approx 1$.

On the other hand, the optimal solution to ITCP$_1$ is $\hat{y}_{-1} = 1$ and $\hat{y}_i = 0$ otherwise with value $OPT_1 = \sqrt{n}$. This corresponds to the optimal solution to ITCP$_2$ for which $\hat{x}_{-1} = 1$ and $\hat{x}_i = 0$ otherwise with value $OPT_2 = OPT_1 = \sqrt{n}$.

Thus the values of optimal solutions to (2) and (4) are the same (which was already guaranteed by Proposition 3), but the value of the optimal integer solution differs significantly from the value of the fractional solution. In fact, we can make the integrality gap as bad as $\Omega(n)$ by simply removing the node -1. Therefore the performance guarantee of any algorithm for approximating the solutions to ITCP$_1$ and ITCP$_2$ based on linear programming (randomized rounding, the primal-dual method, etc.) is, in general, poor.

And there is yet another difficulty. Assume that we have some algorithm (such as randomized rounding) which uses an optimal fractional solution to LTCP$_1$ or LTCP$_2$ to construct an approximation to the optimal integer solution of ITCP$_1$ or ITCP$_2$. This algorithm would necessarily output an approximation having cost $APP_1 \approx APP_2 \approx n$, since it would completely disregard the node -1 (the set X_{-1}). Hence, in general, not only the fractional and integer optimal solutions to TCP$_1$ or TCP$_2$ differ significantly, but also the optimal integer solution and the approximation based on fractional optimal solution can be quite different. In fact, changing the cost c_{-1} to say 2 makes the performance ratio $\Omega(n)$. □

In the remainder of this section we will need the following notation. Set $Y_j = \bigcup_{i \in G_j} S_i$. Define $B = \max_{k,\alpha} b_{k\alpha} = \max\{\text{multiplicity of } t_k \text{ in } Y_j \mid k, j\}$ and $d = \max_\alpha |X_\alpha| = \max_j |Y_j|$. Thus B is the maximum number of locations in any subtree G_j at which any fixed errand can be completed and d is an upper bound on the size of any tree set.

The previous example shows that the optimal solutions to ITCP and LTCP can have quite different values. In order for us to get reasonable worst-case performance bounds on any algorithm based on LP relaxation of the original TCP, B has to be bounded by a constant, or be a slowly growing function of n and m. In what follows, we will show that with these restrictions on B we can approximate TCP quite well.

Consider the following algorithm for approximating the TCP problem on trees.

Algorithm "Approximate_TCP"

Input: A tree $G = (V, E)$ of "locations" with nonnegative cost on each edge and a collection $\{S_1, \ldots, S_n\}$ of subsets of U.

Output: A subtree $\hat{T} \subset G$ approximating an optimal solution to TCP on G.

begin
$(T, \mathbf{y}^*) \leftarrow$ Solve LTCP_1 on $(G, \mathbf{c})$;
$(\mathbf{X}, \mathbf{C}, \mathbf{x}^*) \leftarrow$ Tree_Stripping $(T, \mathbf{y}^*)$;
$\hat{\mathbf{X}} \leftarrow$ Greedy_Subcover $(\mathbf{X}, \mathbf{C})$;
$\hat{T} \leftarrow$ Cover_to_Tree $(\hat{\mathbf{X}})$;
return $\hat{T}$;
end

Here, the algorithm Greedy_Subcover is the usual greedy algorithm for approximating the weighted set cover problem (see [10]). Let us present it here for the sake of completeness.

Algorithm "Greedy_Subcover"

Input: A cover $\mathbf{X} = \{X_1, \ldots, X_L\}$ of U with costs $\mathbf{C} = \{C_1, \ldots, C_L\}$ of sets in $\mathbf{X}$.

Output: A subcover $\hat{\mathbf{X}} = \{\hat{X}_1, \ldots, \hat{X}_R\} = \{X_{l_1}, \ldots, X_{l_R}\} \subset \mathbf{X}$ of U.

begin
$r \leftarrow 0$;
while $U \neq \emptyset$ **do**
$min \leftarrow \infty$;
$r \leftarrow r + 1$;
for $l \leftarrow 1$ **to** L **do**
if $X_l \neq \emptyset$ **and** $\frac{C_l}{|X_l|} < min$ **then** $\{min \leftarrow \frac{C_l}{|X_l|}; \ l^* \leftarrow l;\}$
$U \leftarrow U \setminus X_{l^*}$;
$\hat{X}_r \leftarrow$ the original set X_{l^*};
for $l \leftarrow 1$ **to** L **do** $X_l \leftarrow X_l \setminus X_{l^*}$;
$R \leftarrow r$;
return $\hat{\mathbf{X}}$;
end

Before we state the main theorem, let us prove the following lemma.

Lemma 3 *Let $((X_1, \ldots, X_L), (x_1^*, \ldots, x_L^*), (C_1, \ldots, C_L))$ be a minimum-cost fractional multi-cover of U by tree sets of cost OPT_2^*. The cover $\hat{\mathbf{X}}$ output by the greedy algorithm with input $((X_1, \ldots, X_L), (C_1, \ldots, C_l))$ satisfies*

$$cost(\hat{\mathbf{X}}) \leq OPT_2^* B H(d),$$

where $H(d) = 1 + 1/2 + \cdots + 1/d = \log d + O(1)$.

Proof: A special case of this lemma for $B = 1$ was proved by Chvátal in [10]. In particular he showed the following. Let $(\bar{a}_{kl})$ be the $m \times L$ incidence matrix of the set cover $(X_1, \ldots, X_L)$, that is $\bar{a}_{kl} = 1$ if $k \in X_L$, and $\bar{a}_{kl} = 0$ otherwise. Let $\{l_1, l_2, \ldots, l_R\}$ be the indices of the sets in the subcover output by the greedy algorithm. Then for any L-tuple $(z_1, \ldots, z_L)$ of nonnegative numbers satisfying $\sum_{l=1}^{L} \bar{a}_{kl} z_l \geq 1$ for all $k = 1, \ldots, m$, it must be the case that

$$\sum_{l=1}^{L} H(\sum_{k=1}^{m} \bar{a}_{kl}) C_l z_l \geq \sum_{r=1}^{R} C_{l_r} = cost(\hat{\mathbf{X}}).$$

Since $\sum_k \bar{a}_{kl} = |X_l| \leq d$, we have

$$H(d)\sum_{l=1}^{L} C_l z_l \geq cost(\hat{\mathbf{X}}).$$

We can use Chvátal's result to prove the lemma. In our case, $\bar{a}_{kl} = 1$ if $b_{kl} \geq 1$ and $\bar{a}_{kl} = 0$ if $b_{kl} = 0$. Hence $B\bar{a}_{kl} \geq b_{kl}$. Set $z_l = Bx_l^*$. Then for any $k = 1, \ldots, m$

$$\sum_{l=1}^{L} \bar{a}_{kl} z_l = \sum_{l=1}^{L} \bar{a}_{kl} B x_l^* \geq \sum_{l=1}^{L} b_{kl} x_l^* \geq 1$$

since $(x_1^*, \ldots, x_L^*)$ is a feasible solution to LTCP_2. Thus Chvátal's result implies that

$$H(d)\sum_{l=1}^{L} C_l z_l \geq cost(\hat{\mathbf{X}})$$

that is

$$BH(d)\sum_{l=1}^{L} C_l x_l^* \geq cost(\hat{\mathbf{X}}).$$

Therefore

$$BH(d)OPT_2^* \geq cost(\hat{\mathbf{X}}).$$

□

Theorem 2 *The algorithm "Approximate_TCP" runs in time polynomial in m and n, and its performance ratio is no worse than $BH(d)$. Therefore, for any instance of the TCP problem on trees, the length APP of the approximate covering tree output by the algorithm satisfies*

$$\frac{APP}{OPT} \leq BH(d).$$

Proof: The problem LTCP_1 can be solved in time polynomial in m and n using any of the well-known polynomial-time algorithms (ellipsoid method, Karmarkar's method, etc.). Since $R \leq L \leq n$, the algorithms Greedy_Subcover and Cover_to_Tree also run in time polynomial in m and n. Thus the total running time of Approximate_TCP is bounded by a polynomial in m and n.

Combining the above results, we easily obtain

$$APP = length(\hat{T}) \leq cost(\hat{\mathbf{X}}) \leq OPT_2^* BH(d) = OPT_1^* BH(d) \leq OPT\,BH(d).$$

□

We now give performance bounds for some specific cases of TCP.

Corollary 1 *Let tree G be such that the sets S_i in each of the subtrees G_j are disjoint. Then the performance ratio of the algorithm Approximate_TCP satisfies*

$$\frac{APP}{OPT} \leq H(m).$$

Proof: The fact that for each $j = 1, \ldots, K$ the sets $\{S_i \mid i \in G_j\}$ are disjoint implies that $B = 1$. □

Corollary 2 *Let $B = O(1)$. Then we can approximate TCP within $O(\log m)$.* $\square$

Corollary 3 *Let G be a tree such that the number of nodes in each subtree G_j is bounded by some β (possibly a function of m and n). Then the performance ratio of algorithm Approximate_TCP is no worse than $\beta H(d) \leq \beta H(m)$. In particular, when $\beta = 1$, we have*

$$\frac{APP}{OPT} \leq H(d).$$

Thus, in this special case of TCP corresponding to weighted set cover, our algorithm has the same worst-case performance guarantee as the usual greedy algorithm for weighted set cover. $\square$

Note: One can further improve the performance of our algorithm in the following way. Choose a subtree $Z \subset G$ with $0 \in Z$ having $O(\log n)$ nodes. The graph $G \setminus Z$ is a forest of trees G'_j, $j = 1 \ldots K'$. Now, for each of $O(n)$ possible subtrees $Z_s \subset Z$, $0 \in Z_s$, do the following. Create a "super-root" R_s consisting of all the nodes of Z_s. Then consider a new graph G' obtained as a union of the trees G'_i adjacent to Z_s and the super-root R_s. This graph is again a tree. Set $U' = U \setminus \bigcup_{i \in Z_s} S_i$ and $S'_i = S_i \cap U'$ for all $i \in G'$. Thus we have a simpler instance of the TCP problem and can use our algorithm Approximate_TCP to find a good approximation. We have a total of $O(n)$ subproblems, hence the total running time will be $O(n)$-times the time for Approximate_TCP. This method is particularly useful for trees with a small number of nodes directly adjacent to root 0.

3 ESP and TCP on graphs with bounded $\sum_i a_{ki}$

Define ρ to be the maximum row sum of matrix A, that is $\rho = \max_k\{\sum_i a_{ki}\}$. Hence each task can be completed at at most ρ locations. Since it is more complicated to approximate ESP than TCP, we will consider only the ESP problem. We comment on approximating the TCP problem at the end of the section.

Let G' be a complete graph on V with a shortest-path metric inherited from G. Then for any partial tour with repeated vertices (sometimes called a "closed partial walk") through some $V' \subset V$ in G, there is a "simple" partial tour (without repeated vertices) through V' in G' at most that long. On the other hand, given a simple partial tour through some $V' \subset V$ in G', one can easily construct a partial tour (possibly with repeated vertices) in G of the same length visiting all V' (and possibly some more vertices). Hence, without loss of generality, we will assume that $G = (V, E)$ is a complete undirected graph and the distance function c is symmetric and satisfies the triangle inequality. Also, all the tours considered are simple tours.

Denote by OPT_{3+} the cost of the optimal solution to ESP when the covering tour consists of at least three vertices, and OPT_2 the minimum cost of the covering "tour" consisting of vertex 0 and one other vertex (provided that such a "tour" exists), that is $OPT_2 = \min\{2c_{(0,i)} \mid S_i = U\}$. Then, clearly, $OPT = \min\{OPT_2, OPT_{3+}\}$. We can find the exact value of OPT_2 in $O(n)$ time. Thus, in what follows, we will concentrate on approximating the value OPT_{3+}.

For a given partial tour, let $y_i = 1$ if vertex i is in the tour and $y_i = 0$ otherwise. Similarly, let $x_e = 1$ if the edge e is in the tour and $x_e = 0$ otherwise. For any $S \subset V$ let $\delta(S)$ be the edges from the cut $(S, V \setminus S)$, that is all the edges with one endpoint in S and the other endpoint in $V \setminus S$.

Then the problem of finding OPT_{3+} can be formulated as the following integer programming problem:

Problem $\mathbf{IP}_1$:

$$
\begin{array}{rclll}
Z_1 & = & \text{minimize} & \displaystyle\sum_{e\in E} c_e x_e, & \\
 & & \text{subject to} & (a) \quad \displaystyle\sum_{e\in\delta(\{i\})} x_e = 2y_i, & \text{for all } i \in V, \\
 & & & (b) \quad \displaystyle\sum_{e\in\delta(S)} x_e \geq 2y_i, & \text{for all } S \subset V,\ 0 \notin S, \text{ and all } i \in S, \\
 & & & (c) \quad \displaystyle\sum_{i=1}^{n} a_{ki} y_i \geq 1, & \text{for all } k = 1,\dots,m, \\
 & & & (d) \quad y_o = 1, & \\
 & & & (e) \quad y_i \in \{0,1\}, & \text{for all } i \in V, \\
 & & & (f) \quad x_e \in \{0,1\}, & \text{for all } e \in E.
\end{array}
$$

Condition (a) guarantees that for every vertex in the tour, there are exactly two edges incident on it. Condition (b) prevents subtours, that is the tour cannot be made of two disjoint subtours. Condition (c) guarantees that the chosen sets indeed cover the whole set U. Condition (d) simply assures that the vertex 0 is in the tour. It is easy to see that the two formulations of ESP are equivalent.

3.1 Approximation algorithm

First, let us formulate problem LP_1, the LP relaxation of problem IP_1. In order to do that, we simply replace conditions (e) and (f) in IP_1 by new conditions (e') and (f'), where

$$
\begin{array}{lll}
(e') & 0 \leq y_i \leq 1, & \text{for all } i \in V, \\
(f') & 0 \leq x_e \leq 1, & \text{for all } e \in E.
\end{array}
$$

The algorithm for approximating the optimal solution to ESP is as follows:

Algorithm "Approximate_ESP"

Input: A complete graph $G = (V, E)$ of "locations" with non-negative symmetric distance function on edges satisfying the triangle inequality, and a collection $\{S_1, \dots, S_n\}$ of subsets of U.

Output: A partial tour through some set $T \subset V$ approximating the optimal solution to ESP.

1. Find $APP_2 = \min\{2c_{0,i} \mid i \in V,\ S_i = U\} (= OPT_2)$.

2. Solve problem LP_1 - the LP relaxation of problem IP_1. Let $(\mathbf{y}, \mathbf{x}, Z_1^*) = ((y_i^*)_{i=1}^n, (x_e^*)_{e\in E}, Z_1^*)$ be the optimal solution.

3. Use the Christofides algorithm to find a tour through all the vertices of the set $T = \{i \in V \mid y_i^* \geq 1/\rho\}$. Let APP_{3+} be the length of such a tour.

4. Output $APP = \min\{APP_2, APP_{3+}\}$, and the shorter tour.

Step 1 can be done in polynomial time. Step 2 can be done in polynomial time using the ellipsoid method. Step 3 can also be done in polynomial time.

3.2 Auxiliary results

In order to establish upper bounds on the performance ratio of the above algorithm, we now present some auxiliary results.

Consider the following problem.

Problem IP$_2$:

$$
\begin{array}{rcll}
Z_2 & = & \text{minimize} & \displaystyle\sum_{e \in E} c_e x_e, \\
 & & \text{subject to} & (a) \quad \displaystyle\sum_{e \in \delta(\{i\})} x_e = 2, \quad \text{for all } i \in V, \\
 & & & (b) \quad \displaystyle\sum_{e \in \delta(S)} x_e \geq 2, \quad \text{for all } S \subset V,\ S \neq \emptyset,\ S \neq V, \\
 & & & (c) \quad x_e \in \{0, 1\}, \quad \text{for all } e \in E.
\end{array}
\tag{5}
$$

Clearly, it is the integer programming formulation of the classical TSP. Let LP$_2$ be the LP relaxation of IP$_2$, that is we simply replace the constraint (c) by a new constraint (c'), where

$$(c') \quad 0 \leq x_e \leq 1, \quad \text{for all } e \in E.$$

Denote by Z_2^* the value of the optimal solution to LP$_2$.

For any subset S of V, denote by $L(S)$ the length of the optimal traveling salesman tour through vertices in S and by $L^C(S)$ the length of the tour passing through all vertices of S obtained by the well-known Christofides algorithm. That is $L(V) = Z_2$. Wolsey in [32] and Shmoys and Williamson in [31] proved the following:

Proposition 4

$$L(V) \leq L^C(V) \leq \frac{3}{2} Z_2^*$$

□

Let $V_2 \subset V$. Define

$$r_i = \begin{cases} 2 & \text{if } i \in V_2, \\ 0 & \text{otherwise.} \end{cases}$$

Consider the following LP problem.

Problem LP$_3$:

$$
\begin{array}{rcll}
Z_3^* & = & \text{minimize} & \displaystyle\sum_{e \in E} c_e x_e, \\
 & & \text{subject to} & (a) \quad \displaystyle\sum_{e \in \delta(\{i\})} x_e = r_i, \quad \text{for all } i \in V, \\
 & & & (b) \quad \displaystyle\sum_{e \in \delta(S)} x_e \geq 2, \quad \text{for all } S \subset V \\
 & & & \qquad\qquad\qquad\qquad \text{such that } V_2 \cap S \neq \emptyset \neq V_2 \setminus S, \\
 & & & (c) \quad 0 \leq x_e \leq 1, \quad \text{for all } e \in E.
\end{array}
\tag{6}
$$

We can easily modify Proposition 4 to obtain

Proposition 5

$$L(V_2) \leq L^C(V_2) \leq \frac{3}{2} Z_3^*$$

□

Let us consider the following modification of problem LP$_3$.

Problem LP$_4$:

$$
\begin{array}{rcll}
Z_4^* & = & \text{minimize} & \sum_{e\in E} c_e x_e, \\
 & & \text{subject to} \quad (b) & \sum_{e\in\delta(S)} x_e \ge 2, \quad \text{for all } S\subset V \\
 & & & \qquad\qquad \text{such that } V_2\cap S \ne \emptyset \ne V_2\setminus S, \\
 & & \qquad\qquad (c) & 0 \le x_e, \qquad\quad \text{for all } e\in E.
\end{array} \tag{7}
$$

Thus we omitted constraint (a) and relaxed constraint (c).

Goemans and Bertsimas in [15] and Bienstock, Goemans, Simchi-Levi, and Williamson in [6] proved the following.

Lemma 4 *The optimal solution values to problems LP_3 and LP_4 are the same, that is*

$$Z_3^* = Z_4^*.$$

□

3.3 Performance bounds

Let $(\mathbf{y}, \mathbf{x}, Z_1^*) = ((y_i^*)_{i=1}^n, (x_e^*)_{e\in E}, Z_1^*)$ be the optimal solution to problem LP$_1$. Define

$$
\begin{aligned}
\hat{x}_e &= \rho x_e^* \\
\hat{y}_i &= \begin{cases} 1 & \text{if } y_i^* \ge \frac{1}{\rho} \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

Then

$$T = \{i \in V \mid y_i^* \ge 1/\rho\} = \{i\in V \mid \hat{y}_i = 1\}.$$

Lemma 5

$$\bigcup_{i\in T} S_i = U.$$

Proof: Let k be arbitrary. Set $I_k = \{i\in V \mid a_{ki} = 1\} = \{i \in V \mid t_k \in S_i\}$. We know that $\sum_{i=1}^k a_{ki} \le \rho$, hence $|I_k| \le \rho$. Also,

$$1 \le \sum_{i=1}^n a_{ki} y_i^* = \sum_{i\in I_k} y_i^*,$$

hence $y_{i_o}^* \ge 1/\rho$ for some index $i_o \in I_k$, thus $\hat{y}_{i_o} = 1$, $i_o \in T$, and $t_k \in \bigcup_{i\in T} S_i$. □

Thus each errand t can be completed at at least one location in T.

Since LP$_1$ is the LP relaxation of IP$_1$, we have

$$Z_1^* \le Z_1.$$

Set $V_2 = T$ and $r_i = 2\hat{y}_i$ in problems LP$_3$ and LP$_4$, and use Proposition 5 and Lemma 4 to obtain

$$L^C(T) \le \frac{3}{2} Z_3^* = \frac{3}{2} Z_4^*. \tag{8}$$

Now let us show that $(\hat{x}_e)_{e\in E}$ is a feasible solution to LP$_4$. Indeed, $\hat{x}_e \ge 0$ for all $e \in E$, hence condition (c) is satisfied. Let $S \subset V$ be such that $T\cap S \ne \emptyset \ne (T\setminus S)$. Without loss of generality assume that $0 \in T\setminus S$ and choose some $i\in T\cap S$. Hence $\hat{y}_i = 1$ and $y_i^* \ge \frac{1}{\rho}$. Then we have

$$\sum_{e\in\delta(S)} \hat{x}_e = \rho \sum_{e\in\delta(S)} x_e^* \ge \rho 2 y_i^* \ge \rho 2 \frac{1}{\rho} = 2$$

by the definition of $\hat{x}_e$ and the fact that the x_e^* are a solution to LP_1. Hence the $\hat{x}_e$ satisfy constraint (b) in LP_4.

Therefore,

$$APP_{3+} = L^C(T) \le \frac{3}{2}Z_4^* \le \frac{3}{2}\sum_{e\in E} c_e\hat{x}_e = \frac{3}{2}\rho\sum_{e\in E} c_e x_e^* = \frac{3}{2}\rho Z_1^* \le \frac{3}{2}\rho Z_1 = \frac{3}{2}\rho OPT_{3+}.$$

Thus we have proved the following:

Theorem 3 *The performance ratio of the above algorithm for approximating the optimum solution to the ERRAND SCHEDULING problem satisfies*

$$\frac{APP}{OPT} \le \frac{3}{2}\rho.$$

□

Note: If the sets S_i are disjoint, that is $\rho = 1$, we get the classical TSP performance bound.

Note: One can easily generalize the algorithm and its analysis to the case when, in addition to distances between vertices, there is a cost associated with each vertex. In fact, we can show that, in this case,

$$APP = L_{APP} + C_{APP} \le \rho(3/2L_{OPT} + C_{OPT}) \le (3\rho/2)(L_{OPT} + C_{OPT}) \le (3\rho/2)OPT.$$

Note: In the case of TCP, we can first solve the LP relaxation of the corresponding IP problem and then find a minimum spanning tree through the vertices for which ${y_i}^* \ge \frac{1}{\rho}$. It is easy to show that the performance ratio of this algorithm is bounded by ρ.

Acknowledgments

I would like to thank R. Ravi for useful discussions about the *group Steiner tree* problem.

References

[1] E.M. Arkin and R. Hassin. Minimum Diameter Covering Problems. Submitted.

[2] B. Awerbuch, Y. Azar, A. Blum, S. Vempala. Improved Approximation Guarantees for Minimum-Weight k-Trees and Prize-Collecting Salesmen. In *Proc. ACM Symposium on Theory of Computing* (1995), pp. 277-283.

[3] E. Balas. The Prize Collecting Traveling Salesman. *Networks* **19**(1989), pp. 621-636.

[4] M. Bellare, S. Goldwasser, C. Lund, A. Russel. Efficient Probabilisticly Checkable Proofs and Applications to Approximation. In *Proc. ACM Symposium on Theory of Computing* (1993), pp. 294-304.

[5] R. Bhatia, S. Khuller, and J. Naor. The Loading Time Scheduling Problem. *Proc. Symposium on Foundations of Computer Science* (1995), pp. 72-81.

[6] D. Bienstock, M.X. Goemans, D. Simchi-Levi, D. Williamson. A Note on the Prize Collecting Traveling Salesman Problem. *Mathematical Programming* **59**(1993), pp. 413-420.

[7] A. Blum, P. Chalasani, S. Vempala. A Constant-Factor Approximation for the k-MST Problem in the Plane. In *Proc. ACM Symposium on Theory of Computing* (1995), pp. 163-171.

[8] A. Blum, R. Ravi, S. Vempala. A Constant-Factor Approximation for the k-MST Problem. In *Proc. ACM Symposium on Theory of Computing* (1996).

[9] N. Christofides. Worst Case Analysis of a New Heuristic for the Traveling Salesman Problem. Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[10] V. Chvátal. A Greedy Heuristic for the Set-covering Problem. *Mathematics of Operations Research* **4**(1979), pp. 233-235.

[11] U. Feige. A Threshold of $\ln n$ for Approximating Set Cover. In *Proc. ACM Symposium on Theory of Computing* (1996).

[12] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness* W.H. Freeman, New York, NY (1979).

[13] N. Garg. A 3-approximation for the Minimum Tree Spanning k Vertices. *Proc. Symposium on Foundations of Computer Science* (1996), pp. 302-309.

[14] N. Garg and R. Ravi. Personal Communication (1996).

[15] M.X. Goemans and D.J. Bertsimas. On the Parsimonious Property of Connectivity Problems. *Proc. of the 1$^{\text{st}}$ ACM-SIAM Symposium on Discrete Algorithms* (1990).

[16] M.X. Goemans and D. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM J. Computing* **24**(1995), pp. 296-317.

[17] M. Grotschel, L. Lovasz, A. Schrijver. *Geometric Algorithms and Combinatorial Optimization* Springer-Verlag, Berlin, 1988.

[18] C.C. Hayes. A Model of Planning for Plan Efficiency: Taking Advantage of Operator Overlap. *Proc. of the 11$^{\text{th}}$ International Joint Conference on Artificial Intelligence* (1989), pp. 949-953.

[19] M. Held and R.M. Karp. The Traveling Salesman Problem and Minimum Spanning Trees. *Operations Research* **18**(1970), pp. 1138-1162.

[20] D. Hochbaum. Approximation Algorithms for the Set Covering and Vertex Cover Problem. *SIAM J. Comput.* **11**(1982), pp. 555-556.

[21] E. Ihler. Bounds on the quality of approximate solutions to the group Steiner problem. In *Graph-Theoretic Concepts in Computer Science, WG90, Lecture Notes in Computer Science* **484**, Springer, 1991, pp. 109-118.

[22] E. Ihler. The complexity of approximating the class Steiner tree problem. In *Graph-Theoretic Concepts in Computer Science, WG91, Lecture Notes in Computer Science* **570**, Springer, 1992, pages 85-96.

[23] E. Ihler. The rectilinear class Steiner tree problem for intervals on two parallel lines. *Mathematical Programming, Series B*, **63**(1994), pp. 281-296.

[24] E. Ihler, G. Reich, P. Widmayer. Class Steiner Trees and VLSI-design. Draft paper, `http://www.gmd.de/People/Edmund.Ihler/` July 31, 1995.

[25] 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, New York, 1985.

[26] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. *Journal of the ACM* **41**(1994), pp. 960-982.

[27] H.L. Ong. Approximate Algorithms for the Traveling Purchaser Problem. *Operations Research Letters* **1**(1982), pp. 201-205.

[28] P. Rahgavan and C.D. Thompson. Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs. *Combinatorica* **7**(1987), pp. 365-374.

[29] S. Rajagopalan and V. Vazirani. Logarithmic Approximation of Minimum Weight k-trees. Unpublished manuscript, August 1995.

[30] R. Ravi, R. Sundaram, M.V. Marathe, D.J. Rosenkrantz, and S.S. Ravi. Spanning trees short and small. In *Proc. of the 5$^{\text{th}}$ Annual ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[31] D.B. Shmoys and D.P. Williamson. Analyzing the Held-Karp TSP bound: A Monotonicity Property with Application. *Information Processing Letters* **35**(1990), pp. 281-285.

[32] L. Wolsey. Heuristic Analysis, Linear Programming and Branch and Bound. *Mathematical Programming Study* **13**(1980), pp. 121-134.