Approximations with Minimum Packing Constraint Violation

Jyh-Han Lin and Jeffrey Scott Vitter

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-92-29
June 1992

ϵ-Approximations with Minimum Packing Constraint Violation [1]

Jyh-Han Lin [2]
Motorola Inc.
Paging and Telepoint Systems Lab
1500 N.W. 22nd Avenue
Boynton Beach, FL 33426–8292
lin@pts.mot.com

Jeffrey Scott Vitter [3]
Dept. of Computer Science
Brown University
Providence, R. I. 02912–1910
jsv@cs.brown.edu

[1] A preliminary version of this paper appears in *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, Victoria, BC, Canada, May 1992, 771–782.

[2] This research was conducted while the author was at the Department of Computer Science, Brown University. Support was provided in part by NSF research grant CCR–9007851.

[3] Support was provided in part by an National Science Foundation Presidential Young Investigator Award CCR–9047466 with matching funds from IBM.

Abstract

We present efficient new randomized and deterministic methods for transforming optimal solutions for a type of relaxed integer linear program into provably good solutions for the corresponding $\mathcal{NP}$-hard discrete optimization problem. Without any constraint violation, the ϵ-approximation problem for many problems of this type is itself $\mathcal{NP}$-hard. Our methods provide polynomial-time ϵ-approximations while attempting to minimize the packing constraint violation. Our methods lead to the first known approximation algorithms with provable performance guarantees for the *s-median problem*, the *tree pruning problem*, and the *generalized assignment problem*. These important problems have numerous applications to data compression, vector quantization, memory-based learning, computer graphics, image processing, clustering, regression, network location, scheduling, protocol testing, and communication. We provide evidence via reductions that our approximation algorithms are nearly optimal in terms of the packing constraint violation. We also discuss some recent applications of our techniques to scheduling problems.

1 Introduction

We consider the following type of 0-1 integer programs with mixed covering and packing constraints:

$$\begin{aligned} &\text{minimize} \quad cx && (1)\\ &\text{subject to} && (2)\\ Ax &= 1, && (3)\\ Bx &\leq b, && (4)\\ x &\in P, && (5)\\ x_i &\in \{0,1\}, \quad i \in \mathcal{I}, && (6) \end{aligned}$$

where $\mathcal{I}$ is the set of indices for the vector x, c and b are nonnegative rational vectors, A is a 0-1 matrix, B is a nonnegative rational matrix, and P is a convex set that corresponds to other linear constraints. The linear program relaxation of the above program is to lessen restriction (6) and allow the x_i's to take rational values between 0 and 1.

Three important problems we consider of the above type are the *s-median problem*, the *tree pruning problem*, and the *generalized assignment problem*. These problems arise directly in data compression, vector quantization, memory-based learning, computer graphics, image processing, clustering, regression, network location, scheduling, protocol testing, and communication.

The ϵ-approximation problem [GaJ, SaG] for a $\mathcal{NP}$-hard minimization problem is, for any $\epsilon > 0$, to approximate the optimal solution within a relative factor of ϵ. The ϵ-approximation problem for many problems of the above type is $\mathcal{NP}$-hard [SaG]. That is, without any constraint violation, the ϵ-approximation problem is as hard as finding the optimal solution. In this paper, we develop the first known algorithms for finding ϵ-approximate solutions with minimum packing constraint violation.

Raghavan and Thompson [Rag, RaT] introduce techniques for approximating a 0-1 integer program by first solving its linear program relaxation and then rounding the resulting solution. They successfully apply their rounding techniques to several discrete optimization problems such as routing problems in VLSI, undirected multicommodity flow problems, and the k-matching problem on hypergraphs. However, their techniques depend on Chernoff-type bounds on the tail of the distribution of the weighted sum of Bernoulli trials and apply only to optimization problems with a very special structure. Furthermore, they require problem parameters to be within a certain range for their methods to be effective. Their techniques do not appear to apply to our problems, thus motivating our new approach.

The main results of this paper are new rounding methods that provide polynomial-time ϵ-approximations for the above type of problems described in (1)–(6) while trying to minimize the packing constraint violation.

We now give an outline of the techniques. Let Π be a 0-1 program of the given type and let Π_{L} be its linear program relaxation. The basic algorithm consists of the following three phases:

Phase 1 Solve Π_{L} by linear programming techniques [Kar, Kha]; denote the optimal fractional solution by $\widehat{x}$.

Phase 2 [Filter.] Given $\epsilon > 0$ and the fractional solution $\widehat{x}$ from the first phase, transform Π into a 0-1 integer program $\overline{\Pi}(\epsilon, \widehat{x})$ of minimizing L, subject to

$$\begin{aligned} Ax &= 1, && (7)\\ Bx &\leq Lb, && (8)\\ x &\in P, && (9)\\ x_i &= 0, \quad i \in \mathcal{Z} && (10)\\ x_i &\in \{0,1\}, \quad i \in \mathcal{I} - \mathcal{Z} && (11) \end{aligned}$$

where $\mathcal{Z} \subset \mathcal{I}$ depends on ϵ and $\widehat{x}$. Besides setting a subset of variables to 0, another effect of $\mathcal{Z}$ is that, for each $i \in \mathcal{Z}$, column i of A and B can be considered to be zeroed out. This transformation is said to be valid if any feasible solution $\overline{x}$ for $\overline{\Pi}(\epsilon, \widehat{x})$ satisfies

$$c\overline{x} \leq (1+\epsilon)c\widehat{x} \leq (1+\epsilon)cx^*,$$

where x^* is the optimal (integral) solution for Π. Let $\overline{\Pi}_{\mathrm{L}}(\epsilon, \widehat{x})$ be the linear program relaxation of $\overline{\Pi}(\epsilon, \widehat{x})$.

Phase 3 [Round.] Solve $\overline{\Pi}(\epsilon, \widehat{x})$, that is, attempt to minimize the packing constraint violation, which is represented by the variable L. We solve $\overline{\Pi}(\epsilon, \widehat{x})$ by first converting $\widehat{x}$ into a fractional solution $\widetilde{x}$ for $\overline{\Pi}_{\mathrm{L}}(\epsilon, \widehat{x})$. Then we transform $\widetilde{x}$ into a provably good 0-1 solution for $\overline{\Pi}(\epsilon, \widehat{x})$ by various techniques.

We call the transformation in the second phase *filtering*, since this transformation filters out a subset of variables, namely $\{x_i\}_{i \in \mathcal{Z}}$, by setting them to 0. We call a program *decomposable* if for any two distinct rows A_{r_1} and A_{r_2} of the matrix A, we have $A_{r_1} \cdot A_{r_2} = 0$. As we shall see later, there is a filtering process that works for all decomposable problems. On the other hand, if the program is not decomposable, the filtering process depends heavily on the problem structure. We will refer to $\overline{\Pi}(\epsilon, \widehat{x})$ as the *filtered program* of Π with respect to ϵ and $\widehat{x}$. Phase 2 can be shortcut in practice and combined with Phase 3. The reason we make Phase 2 explicit is to expose the structure of the problem.

The main idea of Phase 3 is that we can derive from $\widehat{x}$ an upper bound for the optimal solution of the relaxed filtered program. Therefore, any provably good rounding algorithms for transforming a solution for $\overline{\Pi}_{\mathrm{L}}(\epsilon, \widehat{x})$ into a 0-1 solution for $\overline{\Pi}(\epsilon, \widehat{x})$ will also provide performance guarantees for the packing constraint violation. We do not need to solve $\overline{\Pi}(\epsilon, \widehat{x})$ from scratch.

In the next three sections, we provide some direct applications of our methods by presenting the first known approximation algorithms with provable performance guarantees for the s-median problem, the tree pruning problem, and the generalized assignment problem. In Section 2, we discuss the well-known s-median problem and demonstrate the basic principles of our techniques. In Section 3, we apply the method to the tree pruning problem, which is not decomposable. Section 4 deals with the generalized assignment problem and the problem of scheduling unrelated parallel machines. For each problem, we provide evidence via reductions that our approximation algorithms are nearly optimal in terms of the packing constraint violation. Section 5 concludes with remarks on possible extensions.

2 The s-Median Problem

In this section we illustrate the basic principles of our methods by means of the (*discrete*) *s-median problem*, which arises directly in applications such as memory-based learning, network location, protocol testing, clustering, and data compression.

For example, consider a company that wishes to locate a number of warehouses for their products on a network of roads. The company estimates the transportation costs between cities and wants to select warehouse locations such that the total cost of transportation is minimized. For another example, suppose that an information service company has a network of computer centers. It wants to make available a new major software package at each center. It can do this either by installing the package at each center, thereby incurring a fixed annual charge, or by installing the package at some of the centers, thereby incurring a smaller fixed annual charge but additional communication costs. The problem here is to determine at which centers the package must be installed so as to minimize the total annual cost. This variant of the problem is known as the *fixed-cost median problem*.

The formal definition of the s-median problem is as follows: We are given a complete (directed or undirected) graph $G = (V, E)$ on n vertices, with vertex set $V = \{1, \ldots, n\}$, edge set $E \subseteq V \times V$, and nonnegative cost c_{ij} associated with edges. We refer to (c_{ij}) as the *cost matrix*. In the network location context, for example, vertices may represent cities and edge costs may correspond to distances between pairs of cities. The goal is to choose s vertices as medians so that the sum of distances from each vertex to its nearest median is minimized.[1]

The s-median problem can be formulated as an integer linear program of minimizing

$$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{12}$$

[1]Note that the s-median problem is distinctly different from the s-center problem of choosing s centers that minimize the worst-case distance from each vertex to its nearest center. For approximation algorithms for the s-center problem, we refer the readers to [FeG, Gon, HoSb]

subject to

$$\begin{array}{rcll} \sum_{j \in V} x_{ij} & = & 1, & i \in V, \\ \sum_{j \in V} y_j & \leq & s, & \\ x_{ij} & \leq & y_j, & i,j \in V, \\ x_{ij}, y_j & \in & \{0,1\}, & i,j \in V, \end{array}$$

where $y_j = 1$ if and only if vertex j is chosen as a median, and $x_{ij} = 1$ if and only if $y_j = 1$ and vertex i is assigned to vertex j.

The fractional s-median problem, which is the linear program relaxation of the s-median problem, is to allow y_j and x_{ij} to take rational values between 0 and 1. Clearly, the optimal solution for the fractional s-median problem is a lower bound on the solutions of the s-median problem. The following lemma is useful for the rest of this section:

Lemma 1 [ACC] *Given a solution $\widehat{y} = (\widehat{y}_1, \ldots, \widehat{y}_n)$ for the fractional s-median problem, we can determine the optimal fractional values for x_{ij}.*

Proof: Each vertex i is assigned to its nearest fractional medians at vertices $j_1, j_2, \ldots$ until their total "weight" $y_{j_1} + y_{j_2} + \cdots$ reaches 1. More specifically, we sort the values c_{ij}, $j \in V$, so that $c_{ij_1(i)} \leq c_{ij_2(i)} \leq \ldots \leq c_{ij_n(i)}$, and let p be such that $\sum_{\ell=1}^{p-1} \widehat{y}_{j_\ell(i)} \leq 1 \leq \sum_{\ell=1}^{p} \widehat{y}_{j_\ell(i)}$. Then we set $\widehat{x}_{ij} = \widehat{y}_j$ for $j = j_1(i), \ldots, j_{p-1}(i)$, $\widehat{x}_{ij_p(i)} = 1 - \sum_{\ell=1}^{p-1} \widehat{y}_{j_\ell(i)}$, and $\widehat{x}_{ij} = 0$ otherwise. □

The s-median problem is $\mathcal{NP}$-hard, even in Euclidean space [GaJ, KaH, MeS, Pap]. Without any probabilistic assumptions on the distribution of the vertices [ACC, FiH, Pap], no approximation algorithms are known. Even if the cost matrix is symmetric, by similar reasoning as in [KaH, SaG], the approximation problem remains hard:

Lemma 2 *The ϵ-approximation problem for the s-median problem is $\mathcal{NP}$-hard even if the cost matrix is symmetric.*

Proof: We use a reduction from the dominating set problem, which is defined as follows: Given a positive integer $1 < s < n$ and an undirected graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ and $E \subseteq V \times V$, does there exists a subset V_s^* of at most s vertices such that each vertex in G is either in V_s^* or is adjacent to a vertex of V_s^*?

Given a graph G as above and $\epsilon > 0$, we construct another complete undirected graph $G' = (V, E')$ with the same set of vertices. The cost c_{ij} is 1 if vertex i and j are adjacent to each other in G, the cost is 0 for $i = j$, otherwise the cost is $(1+\epsilon)(n-s)+1$.

If there exists a dominating set for G of size at most s, then there also exists a set $U \subset V$ of s-medians such that $\sum_{i \in V} \min_{j \in U}\{c_{ij}\} = n - s$. Furthermore, if there does not exist a dominating set for G of size at most s, then $\sum_{i \in V} \min_{j \in U}\{c_{ij}\} \geq (1 + \epsilon)(n - s) + 1$ for any set $U \subset V$ of s-medians. □

The main results in this section are stated in the following two theorems, whose proofs appear in Section 2.2 and Section 2.3, respectively. The filtered program for the s-median problem is given in Section 2.1.

Theorem 1 *There exists a randomized approximation algorithm that, given any* $\epsilon > 0$ *and* $0 < \delta < 1$, *outputs with probability greater than* $1 - \delta$ *a median set* U *of size at most* $(1 + 1/\epsilon)s \ln(n/\delta)$ *such that*

$$\sum_{i \in V} \min_{j \in U}\{c_{ij}\} \;\leq\; (1+\epsilon) \sum_{i \in V} \min_{j \in V_s^*}\{c_{ij}\}, \tag{13}$$

where $V_s^* \subseteq V$ *is a set of optimal* s*-medians.*

Theorem 2 *A median set* U *of size less than* $(1 + 1/\epsilon)s(\ln n + 1)$ *satisfying (13) can be found deterministically.*

We show in Section 2.4 that our approximation algorithms are nearly optimal, unless there exist approximation algorithms for the dominating set and the set cover problems with better than logarithmic performance guarantees, which seems unlikely.

To the best of our knowledge, the only approximation algorithm for a median problem is due to Hochbaum [Hoc] for the *fixed-cost median problem*, where the objective is to minimize the sum $\sum_{j \in U} f_j + \sum_{i \in V} \min_{j \in U}\{c_{ij}\}$, where U is a median set and f_j is a fixed cost for selecting vertex j, and there is no restriction on the number of medians chosen. Hochbaum's technique does not seem to be adaptable to the s-median problem. Our algorithm, on the other hand, can be easily modified to handle the fixed-cost median problem with performance bound matching, up to a constant factor, the logarithmic performance guarantee of Hochbaum's algorithm. Details are given in Section 2.5.3.

2.1 Filtering

In this section, we present a filtering procedure for the s-median problem, which also works for other decomposable problems. Given $\epsilon > 0$ and a fractional solution $\widehat{x}$, we construct a collection of sets $\{V_i\}_{i \in V}$ as follows: Let $\widehat{C}_i = \sum_{j \in V} c_{ij}\widehat{x}_{ij}$ be the weighted cost of assigning vertex i to its medians in the fractional solution $\widehat{x}$. We define the neighborhood V_i of vertex i to be

$$V_i = \{j \in V \mid c_{ij} \leq (1+\epsilon)\widehat{C}_i\}.$$

This construction is illustrated in Figure 1. We transform the s-median problem into a 0-1 integer program of minimizing L subject to

$$\begin{aligned} \sum_{j \in V_i} x_{ij} &= 1, \qquad\qquad i \in V, \\ \sum_{j \in V} y_j &\leq Ls, \end{aligned}$$

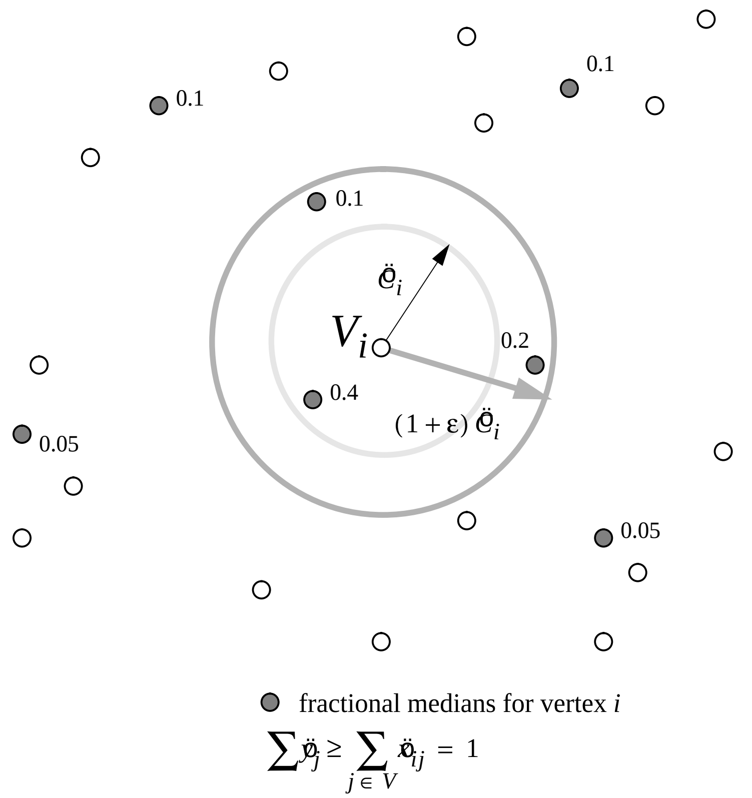

Figure 1: The neighborhood V_i of vertex i is defined to be $V_i = \{j \in V \mid c_{ij} \leq (1+\epsilon)\widehat{C}_i\}$, where $\widehat{C}_i = \sum_{j \in V} c_{ij}\widehat{x}_{ij}$ is the weighted cost of assigning vertex i to its medians in the fractional solution $\widehat{x}$. By Markov's inequality, it follows that $\sum_{j \in V_i} \widehat{y}_j > \frac{\epsilon}{1+\epsilon}$.

$$\begin{aligned} x_{ij} &\leq y_j, & i, j \in V, \\ x_{ij} &= 0, & i \in V, j \in V - V_i, \\ x_{ij}, y_j &\in \{0,1\}, & i \in V, j \in V_i. \end{aligned}$$

This restriction requires that each vertex i must be assigned to a median in its neighborhood V_i. It follows from the definition of V_i that this filtered program is valid:

Theorem 3 *Let $\epsilon > 0$ and let $\widehat{x}$ be a solution for the fractional s-median problem. If $\overline{x}$ is a feasible 0-1 solution for the filtered program defined above, then*

$$\sum_{i \in V} \sum_{j \in V} \overline{x}_{ij} c_{ij} \leq (1+\epsilon) \sum_{i \in V} \sum_{j \in V} \widehat{x}_{ij} c_{ij}.$$

2.2 Rounding by Sampling

In this section, we prove Theorem 1 by using the following random sampling technique, which is of independent interest. The outline of the algorithm is as follows:

1. Solve the linear program relaxation of the s-median problem by linear programming techniques; denote the fractional solution by $\widehat{y}, \widehat{x}$.

2. Given $\epsilon > 0$ and $0 < \delta < 1$, we select $(1 + 1/\epsilon)s \ln(n/\delta)$ vertices randomly, where vertex j has relative weight $\widehat{y}_j/s$. Let U be the set of vertices that the sampling algorithm selects. Then the solution for the filtered program is obtained by setting $y_j = 1$ for each $j \in U$ and $y_j = 0$ for each $j \in V - U$. The values for x_{ij} are determined as in Lemma 1.

The next lemma shows that neighborhood V_i of vertex i contains fractional medians of weight more than $\epsilon/(1+\epsilon)$.

Lemma 3 *For each $i \in V$ and $\epsilon > 0$, we have*

$$\sum_{j \in V_i} \widehat{y}_j \geq \sum_{j \in V_i} \widehat{x}_{ij} > \frac{\epsilon}{1+\epsilon},$$

where V_i is defined as in Section 2.1.

Proof: Suppose $\sum_{j \in V_i} \widehat{x}_{ij} \leq \epsilon/(1+\epsilon)$. Then

$$\begin{aligned}
\widehat{C}_i &= \sum_{j \in V} c_{ij}\widehat{x}_{ij} \\
&\geq \sum_{j \notin V_i} c_{ij}\widehat{x}_{ij} \\
&> (1+\epsilon)\widehat{C}_i \sum_{j \notin V_i} \widehat{x}_{ij} \\
&\geq (1+\epsilon)\widehat{C}_i \left(1 - \frac{\epsilon}{1+\epsilon}\right) \\
&= \widehat{C}_i,
\end{aligned}$$

which is a contradiction. □

We are now ready for the proof of Theorem 1. We say that vertex i is "covered" if a vertex $j \in V_i$ is selected to be a median by the sampling algorithm. By Lemma 3, the probability that vertex i is covered by a randomly selected vertex is more than $\epsilon/s(1+\epsilon)$. Since the sampling algorithm selects $(1 + 1/\epsilon)s \ln(n/\delta)$ vertices, the probability that vertex i is not covered is less than

$$\left(1 - \frac{\epsilon}{s(1+\epsilon)}\right)^{(1+\frac{1}{\epsilon})s \ln \frac{n}{\delta}} < \frac{\delta}{n}.$$

Since there are only n vertices, then the probability that at least one of the n vertices will not be covered is at most $n(\delta/n) = \delta$. The rest of the proof of Theorem 1 follows from Theorem 3.

2.3 Deterministic Greedy Rounding

In this section, we prove Theorem 2 by showing that the well-known greedy set cover method [Chv, Joh, Lov] can be adapted to solve the filtered program for the s-median problem given in Section 2.1. The algorithm is outlined as follows:

1. Solve the linear program relaxation of the s-median problem by linear programming techniques; denote the fractional solution by $\widehat{y}, \widehat{x}$.

2. For each i, compute $\widehat{C}_i = \sum_{j \in V} c_{ij}\widehat{x}_{ij}$.

3. Given $\epsilon > 0$, we form the following set cover problem: for each vertex j such that $\widehat{y}_j > 0$, we construct a set S_j. Vertex i is in S_j if and only if $c_{ij} \leq (1+\epsilon)\widehat{C}_i$ (or equivalently if and only if $j \in V_i$) (see Figure 2).

4. Apply the greedy set cover algorithm [Chv, Joh, Lov]: At each iteration, we choose the set S_j which covers the most uncovered vertices. We repeat this process until all vertices are covered. Let U be the set of vertices that the greedy algorithm selects. Then the solution for the filtered program is obtained by setting $y_j = 1$ for each $j \in U$ and $y_j = 0$ for each $j \in V - U$. The values for x_{ij} are given by Lemma 1.

Now let us prove Theorem 2. By Lemma 1, we can further simplify the filtered program as a 0-1 integer program of minimizing $\sum_{j \in V} y_j$, subject to

$$\begin{aligned} \sum_{j \in V_i} y_j &\geq 1, & i \in V, \\ y_j &\in \{0,1\}, & j \in V, \end{aligned}$$

where V_i is defined as in Section 2.1.

The linear program relaxation of the simplified filtered program is an instance of the fractional set cover problem [Chv, Lov] with the collection of sets $\{S_j\}_{j \in V}$ defined as above. We now show that the size of the optimal fractional cover is less than $(1+1/\epsilon)s$ by converting $\widehat{y}$ to a fractional cover $\widetilde{y}$ of size less than $(1+1/\epsilon)s$. We set $\widetilde{y}_j = \min\{\widehat{y}_j/\widehat{Y}, 1\}$, where $\widehat{Y} = \min_{i \in V}\{\sum_{j \in V_i} \widehat{y}_j\} > \epsilon/(1+\epsilon)$. By Lemma 3, we have $\sum_{j \in V_i} \widetilde{y}_j \geq 1$. Hence, $\widetilde{y}$ is a valid fractional cover and its size is

$$\begin{aligned} \sum_{j \in V} \widetilde{y}_j &= \sum_{j \in V} \min\{\widehat{y}_j/\widehat{Y}, 1\} \\ &< (1+1/\epsilon)\sum_{j \in V} \widehat{y}_j \\ &\leq (1+1/\epsilon)s. \end{aligned}$$

By the results in [Chv, Lov], the number of sets (medians) selected by the greedy algorithm is less than $(1+1/\epsilon)s(\ln n + 1)$. The proof then follows from Theorem 3. Some experimental results for our greedy algorithm are presented in the appendix.

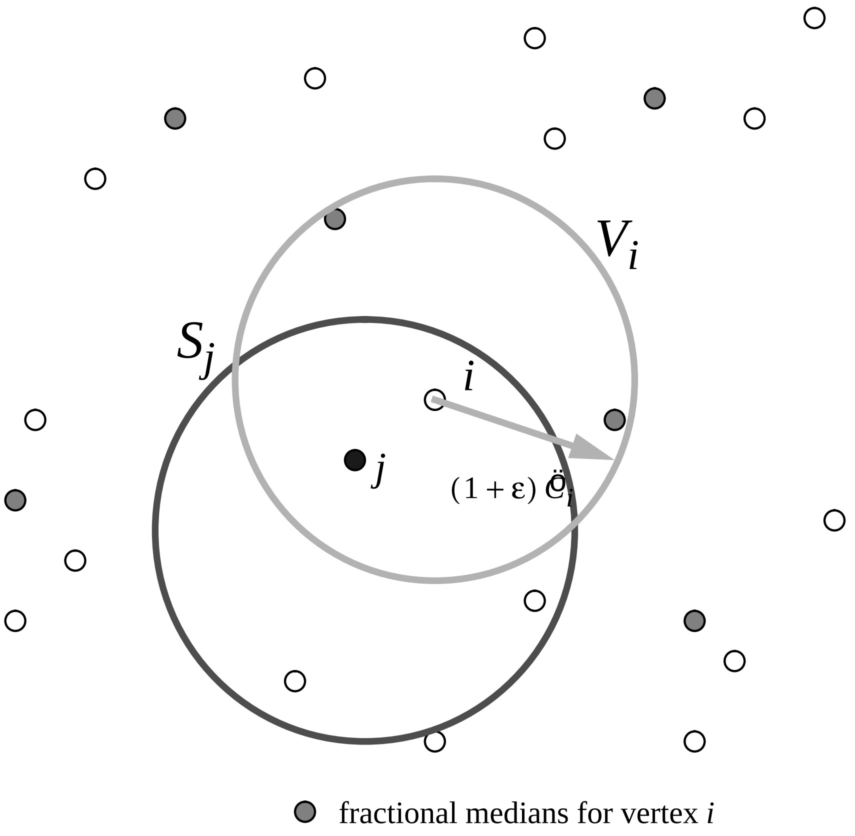

Figure 2: Vertex i is in S_j if and only if $c_{ij} \leq (1+\epsilon)\widehat{C}_i$ (or equivalently if and only if $j \in V_i$).

2.4 Hardness Results

In this section, we show the equivalence (up to constant factors) between the approximabilities of the s-median problem, dominating set, and set cover problems.

Theorem 4 *If there exists an approximation algorithm for the s-median problem that, given any $\epsilon > 0$, outputs a median set U of size at most $(1+1/\epsilon)f(n)s$ satisfying*

$$\sum_{i \in V} \min_{j \in U}\{c_{ij}\} \leq (1+\epsilon) \sum_{i \in V} \min_{j \in V_s^*}\{c_{ij}\},$$

where V_s^* is a set of optimal s-medians, then there exist approximation algorithms for both the dominating set problem and the set cover problem with an $O(f(n))$ relative performance guarantee. This is true even if the cost matrix is required to be symmetric.

Proof: Let ϵ be an arbitrary fixed positive constant. The reduction from the dominating set problem to the s-median problem in Lemma 2 preserves approximability, and the cost matrix is symmetric. It is well-known that the dominating set problem and the set cover problem can be reduced to each other with the approximability preserved. □

By the result of Section 2.3, we immediately have the following corollary:

Corollary 1 *The approximabilities of the s-median problem, the dominating set problem, and the set cover problem are equivalent up to constant factors.*

If the cost matrix must satisfy the triangle inequality, then Theorem 4 does not hold. In this case, however, we have a slightly weaker hardness result. We show that the number of medians cannot be approximated within better than logarithmic factors without violating the bound on total distance, unless the dominating set and set cover problems can be approximated within better than logarithmic factors.

Theorem 5 *Let $f(n) = o(\log n)$. If there exists an approximation algorithm for the s-median problem in metric spaces that outputs a median set U of size at most $f(n)s$ satisfying*

$$\sum_{i \in V} \min_{j \in U} \{c_{ij}\} \le \sum_{i \in V} \min_{j \in V_s^*} \{c_{ij}\},$$

where V_s^* is a set of optimal s-medians, then both the dominating set problem and the set cover problem can be approximated within a relative factor of $O(f(n))$.

Proof: We use a reduction from the dominating set problem defined in the proof of Lemma 2. Given graph G, we construct another complete undirected graph $G' = (V, E')$ with the same set of vertices. The distance c_{ij} is 1 if vertex i and j are adjacent to each other in G; otherwise c_{ij} is the length of the shortest path between i and j in the original graph G. This distance assignment clearly satisfies metric properties.

If there exists a dominating set for G of size at most s, then there also exists a set $U \subset V$ of s-medians such that $\sum_{i \in V} \min_{j \in U}\{c_{ij}\} = n-s$. Furthermore, if there does not exist a dominating set for G of size at most s, then $\sum_{i \in V} \min_{j \in U}\{c_{ij}\} \ge n - s + 1$ for any set $U \subset V$ of s-medians.

Let s be the size of an optimal dominating set for the dominating set problem. By the performance guarantee, the approximation algorithm outputs a median set U of at most $f(n)s$ medians with $\sum_{i \in V} \min_{j \in U}\{c_{ij}\} \le n - s$. This implies that the number of non-dominated vertices is at most $(f(n) - 1)s$. Otherwise, the total cost will be at least $2((f(n) - 1)s + 1) + (n - (f(n) - 1)s - 1) = (f(n) - 1)s + n - s + 1 > n - s$, which is a contradiction. Thus we can construct a dominating set of size at most $f(n)s + (f(n) - 1)s = (2f(n) - 1)s$. □

2.5 Generalized Median Problems

2.5.1 Weighted Cost

We can allow the cost to be weighted, that is, we can replace the objective function (12) by

$$\sum_{i \in V} p_i \sum_{j \in V} c_{ij} x_{ij},$$

where $\{p_1, \ldots, p_n\}$ is a set of nonnegative weights representing the "importance" of vertices. Since our randomized sampling algorithm and deterministic greedy algorithm only require the cost matrix to be nonnegative, we can solve the weighted-cost case simply by using the cost matrix (c'_{ij}), where $c'_{ij} = p_i c_{ij}$.

2.5.2 The Weighted Median Problem

The *weighted median problem* is similar to the s-median problem except that we replace constraint (13) by

$$\sum_{j \in V} w_j y_j \leq W,$$

where $\{w_1, \ldots, w_n\}$ is a set of positive weights and $W > 0$ is a bound on the total weight. This problem is clearly $\mathcal{NP}$-hard. Our greedy algorithm can be easily adapted for the weighted median problem by using the weighted set cover algorithm in the fourth step of the deterministic greedy algorithm:

Theorem 6 *There exist a deterministic approximation algorithm that, given any $\epsilon > 0$, outputs a median set U of weight less than $(1 + 1/\epsilon)W(\ln n + 1)$ such that*

$$\sum_{i \in V} \min_{j \in U} \{c_{ij}\} \leq (1 + \epsilon) \sum_{i \in V} \min_{j \in V_W^*} \{c_{ij}\},$$

where $V_W^* \subseteq V$ is an optimal median set of weight at most W.

2.5.3 The Fixed-Cost Median Problem

The integer linear program of the *fixed-cost median problem* is to minimize

$$\sum_{j \in V} f_j y_j + \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

subject to

$$\begin{aligned}
\sum_{j \in V} x_{ij} &= 1, & i &\in V, \\
x_{ij} &\leq y_j, & i, j &\in V, \\
x_{ij}, y_j &\in \{0, 1\}, & i, j &\in V,
\end{aligned}$$

where $y_j = 1$ if and only if vertex j is chosen as a median, and $x_{ij} = 1$ if and only if $y_j = 1$ and vertex i is assigned to vertex j.

The weighted greedy algorithm for the weighted median problem can be easily adapted for the fixed-cost median problem. The only modification needed is to use the fixed cost f_j as the weight for the set S_j. This gives us the following result:

Theorem 7 *There exists a deterministic approximation algorithm for the fixed-cost median problem that, given any $\epsilon > 0$, outputs a median set U such that*

$$\sum_{j\in U} f_j + \sum_{i\in V} \min_{j\in U}\{c_{ij}\} \quad < \quad (1+1/\epsilon)(\ln n + 1) \sum_{j\in V^*} f_j + (1+\epsilon)\sum_{i\in V} \min_{j\in V^*}\{c_{ij}\},$$

where V^* is an optimal median set.

Theorem 7 matches the logarithmic performance bound of [Hoc] up to a constant factor. However, when $\sum_{j\in V^*} f_j$ is dominated by $\sum_{i\in V} \min_{j\in V^*}\{c_{ij}\}$, our performance bound is better than that of [Hoc].

3 Approximate Tree Pruning

A tree-structured vector quantizer partitions a signal space into a hierarchy of regions, each of which is represented by a representative vector. An input signal vector is quantized by traversing a root-to-leaf path in the tree; the vector is then encoded as the representative vector of the leaf. The central problem for tree-structured vector quantization is how to find a particular tree subject to some cost constraint (such as a bound on the average path length) that minimizes the average distortion. The most popular approach for this problem, introduced by Breiman, Freidman, Olshen, and Stone [BFO] in the context of classification and regression trees, is first to build a large initial tree-structured vector quantizer and then to prune back the tree to satisfy the cost requirement. Besides tree-structured vector quantization, tree pruning algorithms have many applications such as memory-based learning, regression trees, decision trees, and computer graphics.

In computer graphics, for example, let us consider an image consisting of 512×512 pixels. The image can be partitioned into four 256×256 subimages, each of which can be partitioned into four 128×128 subimages, and so forth, until each subimage consists of only one pixel. This gives depth-9 quadtree representation of images. We can use tree pruning algorithm to prune the complete quadtree back and partition the image into regions of similar characteristics. For more applications of the tree pruning problem, we refer the readers to [BFO, CLG].

In this section we present a provably good approximate tree pruning algorithm. We also introduce the notion of probability search trees, which have the potential of outperforming the optimal pruned tree when the cost of trees is the average path length. Experimental results on lossy image compression are given in [LiVc].

3.1 Definitions

We use the notation of [CLG]: A tree T is a finite set of nodes $t_0, t_1, \ldots, t_n$, with a unique root node t_0. The set of leaves of a tree T is denoted by $\widetilde{T}$. A subtree S of tree T is a tree rooted at some node $root(S) \in T$ such that the following condition holds: For each internal node t of T, if any of the children of t is in S, then all the children of t must be in S as well. The leaves $\widetilde{S}$ of a subtree S are not necessarily a subset of $\widetilde{T}$; some leaves of S may be internal nodes of T. If $\widetilde{S} \subseteq \widetilde{T}$, then S is called a branch of T and is denoted by $T_{root(S)}$. We call a subtree S of T a *pruned subtree* and write $S \preceq T$ if the root of S is t_0.

For $t \neq t_0$, we denote the parent node of t as $parent(t)$. For $t \in T - \widetilde{T}$, let $children(t)$ be the set of children of node t. We define $path(t)$ as the set of nodes, including t, from t_0 leading to t.

Definition 1 Let $u(t) \geq 0$ be an arbitrary function on the nodes of T. A *linear tree functional* u on subtrees is defined by

$$u(S) = \sum_{\tilde{t} \in \widetilde{S}} u(\tilde{t}).$$

Let $\Delta u(S) = u(S) - u(root(S))$. A tree functional u is monotonic nondecreasing if and only if for any subtree S of T we have $\Delta u(S) \geq 0$. Similarly, u is monotonic nonincreasing if and only if $\Delta u(S) \leq 0$ for any subtree S of T.

Let $\mathcal{C}$ be a monotonic nondecreasing tree functional and $\mathcal{D}$ be a monotonic nonincreasing tree functional. We call $\mathcal{C}$ the *cost functional* and $\mathcal{D}$ the *distortion functional*. For example, in vector quantization, we have the following setting: Let P be a probability function such that $P(t_0) = 1$ and for all $t \in T - \widetilde{T}$ we have $P(t) = \sum_{t' \in children(t)} P(t')$. Let d be a distortion function on nodes satisfying $P(t)d(t) \geq \sum_{t' \in children(t)} P(t')d(t')$. Usually we let $\mathcal{D}(S)$ be the average distortion of subtrees. Possible definitions for $\mathcal{C}(S)$ include the average path length, the leaf entropy, or the number of leaves.

Definition 2 Given a tree T and a bound C on the cost, the *tree pruning problem* is to find a pruned subtree S of T such that $\mathcal{C}(S) \leq C$ and $\mathcal{D}(S)$ is minimized.

Chou, Lookabaugh, and Gray [CLG] proposed a tree pruning heuristic based on the BFOS algorithm [BFO] in which a given initial tree is pruned back according to certain optimization criterion. Their heuristic traces the lower convex hull of the distortion-cost function and the final pruned subtrees are optimal for their costs. However, if there is no point (pruned subtree) on the lower convex hull at the desired cost, it requires time-sharing between two neighboring points (pruned subtrees). Lin, Storer, and Cohn [LSC] show that the tree pruning problem is $\mathcal{NP}$-hard in general.[2]

[2]On the other hand, Lin, Storer, and Cohn also show that the tree pruning problem can be solved in polynomial time if the trees are binary and the cost constraint is the number of leaves. Our approximate tree pruning algorithm works for general trees and applies to other cost constraints, such as the average path length and the leaf entropy.

Our main result is the following theorem:

Theorem 8 *Given any $\epsilon > 0$, there exist an approximate tree pruning algorithm that outputs a pruned subtree S satisfying*

$$\mathcal{C}(S) \quad < \quad (1 + 1/\epsilon)C \tag{14}$$

and

$$\mathcal{D}(S) \leq (1+\epsilon)\widehat{D} \leq (1+\epsilon)D, \tag{15}$$

where $\widehat{D}$ is the distortion of the fractional solution for the tree pruning problem and D is the optimal distortion of pruned subtrees with cost at most C.

The algorithm mentioned in Theorem 8 is given in Section 3.4 and the proof appears in Sections 3.6 and 3.7.

3.2 The Integer Linear Program

We may formulate the tree pruning problem as the following integer linear program: For each node $t \in T$, we let x_t be a decision variable such that $x_t = 1$ if and only if node t is a leaf in the final pruned subtree, and $x_t = 0$ otherwise. The integer linear program for the optimal tree pruning problem is to minimize the cost

$$\sum_{t \in T} x_t \mathcal{D}(t)$$

subject to

$$\begin{aligned} \sum_{t \in path(\tilde{t})} x_t &= 1, & \tilde{t} \in \tilde{T}, \\ \sum_{t \in T} x_t \mathcal{C}(t) &\leq C, & \\ x_t &\in \{0,1\}, & t \in T. \end{aligned}$$

3.3 The Filtered Program

Since the program is not decomposable, we need a different filtering procedure: Given any $\epsilon > 0$ and a fractional solution $\widehat{x}$, we define for each leaf $\tilde{t}$ the set $I_{\tilde{t}} = \{t \mid t \in path(\tilde{t})$ and $\sum_{t' \in path(t)} \widehat{x}_t \leq 1/(1+\epsilon)\}$. The filtered program is to minimize L, subject to

$$\begin{aligned} \sum_{t \in I_{\tilde{t}}} x_t &= 1, & \tilde{t} \in \tilde{T}, \\ \sum_{t \in T} x_t \mathcal{C}(t) &\leq L \cdot C, & \\ x_t &= 0, & t \in \bigcup_{\tilde{t} \in \tilde{T}} \left(path(\tilde{t}) - I_{\tilde{t}} \right), \\ x_t &\in \{0,1\}, & t \in T. \end{aligned}$$

The following theorem shows that the filtered program is valid. The proof is given in Section 3.6.

Theorem 9 *Let $\overline{x}$ be a feasible 0-1 solution for the filtered program defined as above. Then we have*

$$\sum_{t \in T} \overline{x}_t \mathcal{D}(t) \leq (1+\epsilon) \sum_{t \in T} \widehat{x}_t \mathcal{D}(t).$$

3.4 Deterministic Pruning

The following is an outline of the approximate tree pruning algorithm:

1. Solve the linear program relaxation of the tree pruning problem by linear programming techniques; denote the fractional solution by $\widehat{x}$.

2. Given $\epsilon > 0$, in a top-down and breadth-first fashion, we prune the tree at any node t where $\sum_{t' \in path(t)} \widehat{x}_{t'} \geq 1/(1+\epsilon)$.

The performance guarantee of the pruning algorithm will be proven in Sections 3.6 and 3.7.

3.5 Probability Search Trees

Probability search trees are an interesting interpretation of the fractional solution of the integer linear program for the tree pruning problem.

Definition 3 A *probability search tree* $\widehat{T} = (T, q)$ is a tree T with augmented probability function q on tree nodes, which satisfies $\sum_{t \in path(\widetilde{t})} q(t) = 1$, for all leaves $\widetilde{t} \in \widetilde{T}$. Let us define

$$Q(t) = 1 - \sum_{t' \in path(t)} q(t').$$

A search along a path through node t will continue at node t, assuming the search reaches node t, with probability $Q(t)/Q(parent(t))$.

We may use $Q(t)$ as the probability that the search passes through node t and $q(t)$ as the probability that the search stops at node t.

We can extend tree functionals to probability search trees in the following way:

Definition 4 Let $\widehat{T} = (T, q)$ be a probability search tree. Given a linear tree functional u, we define the *probability tree functional* u^* on subtrees as

$$u^*(S) = \sum_{t \in S - \widetilde{S}} q(t)u(t) + \sum_{\widetilde{t} \in \widetilde{S}} Q(parent(\widetilde{t}))u(\widetilde{t}),$$

and we define $\Delta u^*(S) = u^*(S) - u^*(root(S))$. A probability tree functional u^* is monotonic nondecreasing if and only if for any subtree S of T, we have $\Delta u(S) \geq 0$. Similarly, u^* is monotonic nonincreasing if and only if $\Delta u(S) \leq 0$ for any subtree S of T.

Lemma 4 *For any subtree S rooted at t of height greater than 0, we have*

$$\Delta\mathcal{C}^*(S) = -Q(t)\mathcal{C}(t) + \sum_{t' \in children(t)} \mathcal{C}^*(S_{t'}).$$

Proof: By manipulating the sum, we have

$$\begin{aligned}
\Delta\mathcal{C}^*(S) &= \mathcal{C}^*(S) - \mathcal{C}^*(t) \\
&= \widehat{x}_t\mathcal{C}(t) - \mathcal{C}^*(t) + \sum_{t' \in children(t)} \mathcal{C}^*(S_{t'}) \\
&= -Q(t)\mathcal{C}(t) + \sum_{t' \in children(t)} \mathcal{C}^*(S_{t'}).
\end{aligned}$$

□

The following theorem shows the monotonic properties of probability tree functionals:

Theorem 10 *If a linear tree functional u is monotonic nondecreasing (nonincreasing), then the probability tree functional u^* is also monotonic nondecreasing (nonincreasing).*

Proof: By induction on the height of S. If the subtree S consists of a single node t, then $\Delta\mathcal{C}^*(S) = \mathcal{C}^*(t) - \mathcal{C}^*(t) = 0$. Suppose that the theorem is true for all subtrees of height no greater than k. Let S be a subtree of height $k+1$, and let $t = root(S)$. By Lemma 4, we may write

$$\Delta\mathcal{C}^*(S) = -Q(t)\mathcal{C}(t) + \sum_{t' \in children(t)} \mathcal{C}^*(S_{t'}).$$

By the induction hypothesis, we have

$$\begin{aligned}
\Delta\mathcal{C}^*(S) &\geq -Q(t)\mathcal{C}(t) + \sum_{t' \in children(t)} \mathcal{C}^*(t') \\
&= Q(t)\Big(-\mathcal{C}(t) + \sum_{t' \in children(t)} \mathcal{C}(t')\Big) \\
&\geq 0.
\end{aligned}$$

The last inequality follows since $\mathcal{C}$ is monotonic nondecreasing. □

We can interpret an optimal fractional solution $\widehat{x}$ as an augmented probability function q by setting $q(t) = \widehat{x}_t$. The resulting probability search tree is a novel extension of the notion of time sharing in tree structured vector quantization and has the potential of outperforming the optimal pruned subtree in terms of the average path length:

Corollary 2 *Let $\widehat{T} = (T, q)$ be a probability search tree with $q(t) = \widehat{x}_t$, where $\widehat{x}$ is an optimal fractional solution for the tree pruning problem, and let $\mathcal{C}(S)$ be the average path length of subtrees. Then we have*

$$\mathcal{C}^*(\widehat{T}) \leq C$$

and

$$\mathcal{D}^*(\widehat{T}) = \widehat{D} \leq D,$$

where $\widehat{D}$ is the distortion of the optimal fractional solution for the tree pruning problem and D is the optimal distortion of pruned subtrees with cost at most C.

3.6 Bounding the Distortion

We prove Theorem 9 in this section. Let S be the final pruned subtree corresponding to $\overline{x}$. We want to show that

$$\sum_{t \in S} \widehat{x}_t \mathcal{D}(t) \geq \frac{1}{1+\epsilon} \mathcal{D}(S).$$

Since $\mathcal{D}$ is monotonic nonincreasing, for any $t \in S$ we have

$$\mathcal{D}(t) \geq \sum_{\widetilde{t} \in \widetilde{S}_t} \mathcal{D}(\widetilde{t}).$$

Therefore, we have

$$\begin{aligned} \sum_{t \in S} \widehat{x}_t \mathcal{D}(t) &\geq \sum_{t \in S} \widehat{x}_t \sum_{\widetilde{t} \in \widetilde{S}_t} \mathcal{D}(\widetilde{t}) \\ &\geq \sum_{\widetilde{t} \in \widetilde{S}} \mathcal{D}(\widetilde{t}) \sum_{t \in path(\widetilde{t})} \widehat{x}_t \\ &\geq \frac{1}{1+\epsilon} \sum_{\widetilde{t} \in \widetilde{S}} \mathcal{D}(\widetilde{t}). \end{aligned}$$

The last inequality follows from the definition of $I_{\widetilde{t}}$ in the filtered program.

3.7 Bounding the Cost

In this section we prove Theorem 8. The bound (15) follows from Theorem 9, so all that remains is to prove (14). Let S be the final pruned subtree. We prove $L < 1 + 1/\epsilon$ by showing that

$$\sum_{t \in \widetilde{S} \cup (T-S)} \widehat{x}_t \mathcal{C}(t) > \frac{\epsilon}{1+\epsilon} \mathcal{C}(S).$$

By expanding the sum, we have

$$\begin{aligned}
\sum_{t\in\widetilde{S}\cup(T-S)} \widehat{x}_t\mathcal{C}(t) &= \sum_{\widetilde{t}\in\widetilde{S}} \widehat{x}_{\widetilde{t}}\,\mathcal{C}(\widetilde{t}) + \sum_{t\in T-S} \widehat{x}_t\mathcal{C}(t) \\
&= \sum_{\widetilde{t}\in\widetilde{S}} \Big(Q(parent(\widetilde{t})) - Q(\widetilde{t})\Big)\,\mathcal{C}(\widetilde{t}) + \sum_{t\in T-S} \widehat{x}_t\mathcal{C}(t) \\
&= \sum_{\widetilde{t}\in\widetilde{S}} Q(parent(\widetilde{t}))\mathcal{C}(\widetilde{t}) - \sum_{\widetilde{t}\in\widetilde{S}} Q(\widetilde{t})\mathcal{C}(\widetilde{t}) + \sum_{t\in T-S} \widehat{x}_t\mathcal{C}(t).
\end{aligned}$$

By Lemma 4, we may write

$$\sum_{t\in\widetilde{S}\cup(T-S)} \widehat{x}_t\mathcal{C}(t) = \sum_{\widetilde{t}\in\widetilde{S}} Q(parent(\widetilde{t}))\mathcal{C}(\widetilde{t}) + \sum_{\widetilde{t}\in\widetilde{S}} \Delta\mathcal{C}^*(T_{\widetilde{t}}).$$

For any $\widetilde{t}\in\widetilde{S}$, by Theorem 10, we have

$$\sum_{\widetilde{t}\in\widetilde{S}} \Delta\mathcal{C}^*(T_{\widetilde{t}}) \geq 0.$$

Therefore, we have

$$\begin{aligned}
\sum_{t\in\widetilde{S}\cup(T-S)} \widehat{x}_t\mathcal{C}(t) &\geq \sum_{\widetilde{t}\in\widetilde{S}} Q(parent(\widetilde{t}))\mathcal{C}(\widetilde{t}) \\
&> \frac{\epsilon}{1+\epsilon}\sum_{\widetilde{t}\in\widetilde{S}} \mathcal{C}(\widetilde{t}),
\end{aligned}$$

which implies (14). The last inequality follows from the way we prune the tree.

4 Other Applications

4.1 The Generalized Assignment Problem

Let $J = \{1,\ldots,K\}$ be a set of agent indices and $I = \{1,\ldots,n\}$ be a set of task indices. We denote the cost of assigning task i to agent j as c_{ij} and the resource available to agent j as $b_j > 0$. Let $0 < r_{ij} \leq b_j$ be the resource required of agent j to perform task i. The *generalized assignment problem* is formulated as minimizing

$$\sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij}$$

subject to

$$\begin{aligned}
\sum_{j\in J} x_{ij} &= 1, & i\in I, \\
\sum_{i\in I} r_{ij}x_{ij} &\leq b_j, & j\in J, \\
x_{ij} &\in \{0,1\}, & i\in I, j\in J,
\end{aligned}$$

where x_{ij} is 1 if and only if task i is assigned to agent j, $x_{ij} = 0$ otherwise.

If $r_{ij} = 1$ for all $i \in I$ and $j \in J$, then the well-known Hungarian method [Kuh] can be adapted for its solution [GLS]. On the other hand, Sahni and Gonzalez [SaG] show that the ϵ-approximation problem is $\mathcal{NP}$-hard for the generalized assignment problem.

For simplicity, we assume for the rest of the section that the generalized assignment problem is feasible. We remark that we can use the notion of relaxed decision procedures [HoSa, LST] for checking feasibility approximately.

The filtering procedure for the generalized assignment problem is similar to that of the s-median problem. Given $\epsilon > 0$ and a fractional solution $\widehat{x}$ for the linear relaxation of the above program, let $\widehat{C}_i = \sum_{j \in J} c_{ij} \widehat{x}_{ij}$. We define

$$J_i = \{ j \in J \mid c_{ij} \leq (1+\epsilon)\widehat{C}_i \}$$

for each $i \in I$, and

$$I_j = \{ i \in I \mid c_{ij} \leq (1+\epsilon)\widehat{C}_i \}$$

for each $j \in J$. The filtered program is to minimize L, subject to

$$\begin{array}{rcll} \sum_{j \in J_i} x_{ij} & = & 1, & i \in I, \\ \sum_{i \in I_j} \frac{r_{ij}}{b_j} x_{ij} & \leq & L, & j \in J, \\ x_{ij} & = & 0, & i \in I, j \in J - J_i, \\ x_{ij} & \in & \{0,1\}, & i \in I, j \in J. \end{array}$$

Any feasible solution to the filtered program provides ϵ-approximations for the generalized assignment problem. By our techniques, we can show the following:

Theorem 11 *There exists a deterministic approximation algorithm for the generalized assignment problem that, given any $\epsilon > 0$, outputs an assignment $\overline{x}$ such that*

$$\sum_{i \in I} \sum_{j \in J} c_{ij} \overline{x}_{ij} \leq (1+\epsilon) \sum_{i \in I} \sum_{j \in J} c_{ij} x^*_{ij},$$

where x^* is an optimal assignment, and

$$\sum_{i \in I} r_{ij} \overline{x}_{ij} < (2 + 1/\epsilon) b_j$$

for all $j \in J$.

Recently, after the acceptance of the conference version of this paper, Shmoys and Tardos [ShTb] have developed an approximation algorithm for the generalized assignment problem that outputs an assignment $\overline{x}$ such that $\sum_{i \in I} \sum_{j \in J} c_{ij} \overline{x}_{ij} \leq \sum_{i \in I} \sum_{j \in J} c_{ij} x^*_{ij}$, where x^* is an optimal assignment, and $\sum_{i \in I} r_{ij} \overline{x}_{ij} \leq 2b_j$ for all $j \in J$.

We now relate the approximability of the generalized assignment problem to that of the *minimum makespan problem on unrelated parallel machines* (*without preemption*). The later problem can be formulated as an integer linear program of minimizing L, subject to

$$\begin{aligned} \sum_{j \in J} x_{ij} &= 1, && i \in I, \\ \sum_{i \in I} p_{ij} x_{ij} &\leq L, && j \in J, \\ x_{ij} &\in \{0,1\}, && i \in I, j \in J, \end{aligned}$$

where $J = \{1, \ldots, K\}$ is a set of machines, $I = \{1, \ldots, n\}$ is a set of jobs, and p_{ij} is a positive integer representing the processing time for job i on machine j.

Theorem 12 *For any $0 < \rho < 1/2$, unless $\mathcal{NP} = \mathcal{P}$, there does not exist an approximation algorithm for the generalized assignment problem that, given any $\epsilon > 0$, outputs an assignment $\overline{x}$ such that*

$$\sum_{i \in I} \sum_{j \in J} c_{ij} \overline{x}_{ij} \leq (1 + \epsilon) \sum_{i \in I} \sum_{j \in J} c_{ij} x^*_{ij},$$

where x^* is an optimal assignment, and

$$\sum_{i \in I} r_{ij} \overline{x}_{ij} \leq (\rho + 1 + 1/\epsilon) b_j,$$

for all $j \in J$.

Proof: Suppose that there exists an approximation algorithm for the generalized assignment problem that, given any $\epsilon > 0$, outputs an assignment $\overline{x}$ such that

$$\sum_{i \in I} \sum_{j \in J} c_{ij} \overline{x}_{ij} \leq (1 + \epsilon) \sum_{i \in I} \sum_{j \in J} c_{ij} x^*_{ij},$$

where x^* is an optimal assignment, and

$$\sum_{i \in I} r_{ij} \overline{x}_{ij} \leq (\rho + 1 + 1/\epsilon) b_j,$$

where $0 < \rho < 1$, for all $j \in J$. In such a case, we can show by reductions that there also exists a ρ-approximation algorithm for the minimum makespan problem on unrelated parallel machines. By Corollary 2 in [LST], we have proven the hardness result. $\square$

4.2 Scheduling of Unrelated Parallel Machines

One basic question in the scheduling of unrelated parallel machines is the relationship between the optimal makespan with preemption and the optimal makespan without preemption. Surprisingly, no nontrivial bound was known for this problem. It is straightforward to use our techniques to prove the following theorem:

Theorem 13 *Let $\widehat{L}$ be the optimal makespan with preemption and let L^* be the optimal makespan without preemption for scheduling unrelated parallel machines. Then we have $L^* < 4\widehat{L}$.*

Proof: Lawler and Labetoulle [LaL] showed the optimal makespan with preemption is the optimal solution to the following linear program of minimizing L subject to

$$\begin{array}{rcll} \sum_{j \in J} x_{ij} & = & 1, & i \in I, \\ \sum_{i \in I} p_{ij} x_{ij} & \leq & L, & j \in J, \\ \sum_{j \in J} p_{ij} x_{ij} & \leq & L, & i \in I, \\ x_{ij} & \geq & 0, & i \in I, j \in J. \end{array}$$

Given the optimal fractional solution $\widehat{x}$ and the optimal makespan with preemption $\widehat{L}$, we want to convert it into a schedule of length less than $4\widehat{L}$.

We first form a new fractional solution $\tilde{x}$ as follows: for each $\widehat{x}_{ij} > 0$ with $p_{ij} > 2\widehat{L}$ we set $\tilde{x}_{ij}$ to 0 (filtering); otherwise, we let $\tilde{x}_{ij} = \widehat{x}_{ij}/\widehat{X}_i$, where $\widehat{X}_i = \sum_{p_{ij} \leq 2\widehat{L}} \widehat{x}_{ij} > 1/2$. It is clear that $\tilde{x}$ is a feasible solution to the following set of linear constraints:

$$\begin{array}{rcll} \sum_{j \in J} x_{ij} & = & 1, & i \in I, \\ \sum_{j \in J} p_{ij} x_{ij} & < & 2\widehat{L}, & i \in I, \\ x_{ij} & \in & \{0, 1\}, & i \in I, j \in J, \\ x_{ij} & = & 0 & \text{if } \; p_{ij} > 2\widehat{L}. \end{array}$$

By the Rounding Theorem of [LST], we can find a non-preemptive schedule of length less than $4\widehat{L}$ in polynomial time. □

Partially based on our techniques, Shmoys and Tardos [ShTa] have developed an $O(\log^2 K)$-approximation algorithm for non-preemptive scheduling of unrelated parallel machines with precedence constraints on the jobs.

5 Conclusions

Our results deal with a class of frequently encountered 0-1 optimization problems whose ϵ-approximation problems are $\mathcal{NP}$-hard. For each of the problems considered, we first solve its relaxed integer program by linear programming techniques. Given the fractional solution and $\epsilon > 0$, we transform the problem by new filtering methods, which we introduce in this paper, into another 0-1 optimization problem, whose feasible solutions insure ϵ-approximation, of minimizing the packing constraint violation. Finally we solve the filtered program by interesting randomized and deterministic techniques.

More specifically, we present the first known polynomial-time approximation algorithms with provable performance guarantees for the s-median problem, the tree pruning problem, and the generalized assignment problem. We also provide evidence that our approximation algorithms are nearly optimal in terms of the packing constraint violation. The results in Section 4.2 indicate that our methods may have many more applications other than finding ϵ-approximate solutions with small packing constraint violation for $\mathcal{NP}$-hard optimization problems.

It is open whether our performance guarantees for the s-median problem can be tightened for the special case of Euclidean space. Initial work shows that the logarithmic blowup in the number of chosen medians in the s-median problem can be avoided at the expense of doubling the objective cost [LiVa]. Practical use of our algorithms for vector quantization, clustering, and memory-based learning can be found in [LiVb, LiVc].

Acknowledgements. The application in Section 4.2 was communicated to us by Shmoys and Tardos [ShTa]. We thank them for their permission to include the result here.

References

[ACC] S. Ahn, A. Cooper, G. Cornuejols, and A. Frieze, "Probabilistic Analysis of a Relaxation for the K-Median Problem," *Mathematics of Operations Research* 13 (February 1988), 1–31.

[BFO] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification Trees and Regression Trees*, Wadsworth, Belmont, CA, 1984.

[CLG] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Optimal Pruning with Applications to Tree-Structured Source Coding and Modeling," *IEEE Transactions on Information Theory* (1989), 299–315.

[Chv] V. Chvátal, "A Greedy Heuristic for the Set-Covering Problem," *Mathematics of Operations Research* 4 (1979), 233–235.

[FeG] T. Feder and D. Greene, "Optimal Algorithms for Approximate Clustering," in *Proceedings of the 20th Annual Symposium on the Theory of Computing*, 1988, 434–444.

[FiH] M. L. Fisher and D. S. Hochbaum, "Probabilistic Analysis of the Planar K-Median Problem," *Mathematics of Operations Research* 5 (February 1980), 27–34.

[GaJ] M. R. Garey and D. S. Johnson, *Computers and intractability: A Guide to the Theory of $\mathcal{NP}$-completeness*, W. H. Freeman and Co., San Francisco, CA, 1979.

[GNR] R. S. Garfinkel, A. W. Neebe, and M. R. Rao, "An Algorithm for the M-Median Plant Location Problem," *Transportation Science* 8 (1974), 217–236.

[Gon] T. F. Gonzalez, "Clustering to Minimize the Maximum Intercluster Distance," *Theoretical Computer Science* 38 (1985), 293–306.

[GLS] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, Heidelberg, New York, 1988.

[Hoc] D. S. Hochbaum, "Heuristics for the Fixed-Cost Median Problem," *Mathematical Programming* 22 (1982), 148–162.

[HoSa] D. S. Hochbaum and D. B. Shmoys, "Using Dual Approximation Algorithms for Scheduling Problems: Practical and Theoretical Results," *Journal of the Association for Computing Machinery* 34 (1987), 144–162.

[HoSb] D. S. Hochbaum and D. B. Shmoys, "A Unified Approach to Approximation Algorithms for Bottleneck Problems," *Journal of the Association for Computing Machinery* 33 (July 1986), 533–550.

[Joh] D. S. Johnson, "Approximation Algorithms for Combinatorial Problems," *Journal of Computer and System Sciences* 9 (1974), 256–278.

[KaH] O. Kariv and S. L. Hakimi, "An Algorithmic Approach to Network Location Problems. II: The p-Medians," *SIAM Journal on Applied Mathematics* (1979), 539–560.

[Kar] N. Karmarkar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica* 4 (1984), 373–395.

[Kha] L. G. Khachiyan, "A Polynomial Algorithm in Linear Programming," *Soviet Math. Doklady* 20 (1979), 191–194.

[Kuh] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly* 2 (1955), 83–97.

[LaL] E. L. Lawler and J. Labetoulle, "On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming," *Journal of ACM* 25 (October 1978), 612–61.

[LST] J. K. Lenstra, D. B. Shmoys, and É. Tardos, "Approximation Algorithms for Scheduling Unrelated Parallel Machines," *Mathematical Programming A* 46 (1990), 259–271.

[LSC] J. Lin, J. A. Storer, and M. Cohn, "On the Complexity of Optimal Tree Pruning for Source Coding," in *Proceedings of the Data Compression Conference*, J. A. Storer and J. H. Reif, eds., Snowbird, Utah, April 1991, 63–72.

[LiVa] J.-H. Lin and J. S. Vitter, "Approximation Algorithms for Geometric Median Problems," Technical Report, Department of Computer Science, Brown University, 1992.

[LiVb] J.-H. Lin and J. S. Vitter, "A Theory for Memory-Based Learning," in *Proceedings of the Fifth ACM Workshop on Computational Learning Theory*, Pittsburgh, Pennsylvania, July 1992.

[LiVc] J.-H. Lin and J. S. Vitter, "Nearly Optimal Vector Quantization via Linear Programming," in *Proceedings of the IEEE Data Compression Conference*, Snowbird, Utah, March 1992, 22–31.

[Lov] L. Lovász, "On the Ratio of Optimal Integral and Fractional Covers," *Discrete Mathematics* 13 (1975), 383–390.

[MeS] N. Megiddo and K. J. Supowit, "On the Complexity of Some Common Geometric Location Problems," *SIAM Journal on Computing* 13 (1984), 182–196.

[Pap] C. H. Papadimitriou, "Worst-case and Probabilistic Analysis of a Geometric Location Problem," *SIAM Journal on Computing* 10 (1981), 542–557.

[Rag] P. Raghavan, "Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs," *Journal of Computer and System Science* 37 (1988), 130–143.

[RaT] P. Raghavan and C. D. Thompson, "Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs," *Combinatorics* 7 (1987), 365–374.

[SaG] S. Sahni and T. F. Gonzalez, "P-Complete Approximation Problems," *Journal of the Association for Computing Machinery* 23 (July, 1976), 555–565.

[ShTa] D. B. Shmoys and É. Tardos, Personal Communication, January 23, 1992.

[ShTb] D. B. Shmoys and É. Tardos, "Scheduling Unrelated Parallel Machines with Costs," Technical Report No. 1001, School of Operations Research and Industrial Engineering, Cornell University, April 1992.

Appendix: Experimental Results

In this appendix, we report some experimental results on the Euclidean s-median problem. (Experimental data on the performance of our tree pruning algorithms for data compression applications are given in [LiVc].) The s-median data set consists of the locations of 134 American cities, as shown in Figure 3. We solve the linear programs for $s = 10, 15, 20, 25, 30, 35, 40, 45, 50$. The number of variables in the linear program is $O(n^2)$ and the number of constraints is also $O(n^2)$. A straightforward implementation of the simplex method for linear programming requires $O(n^4)$ space.

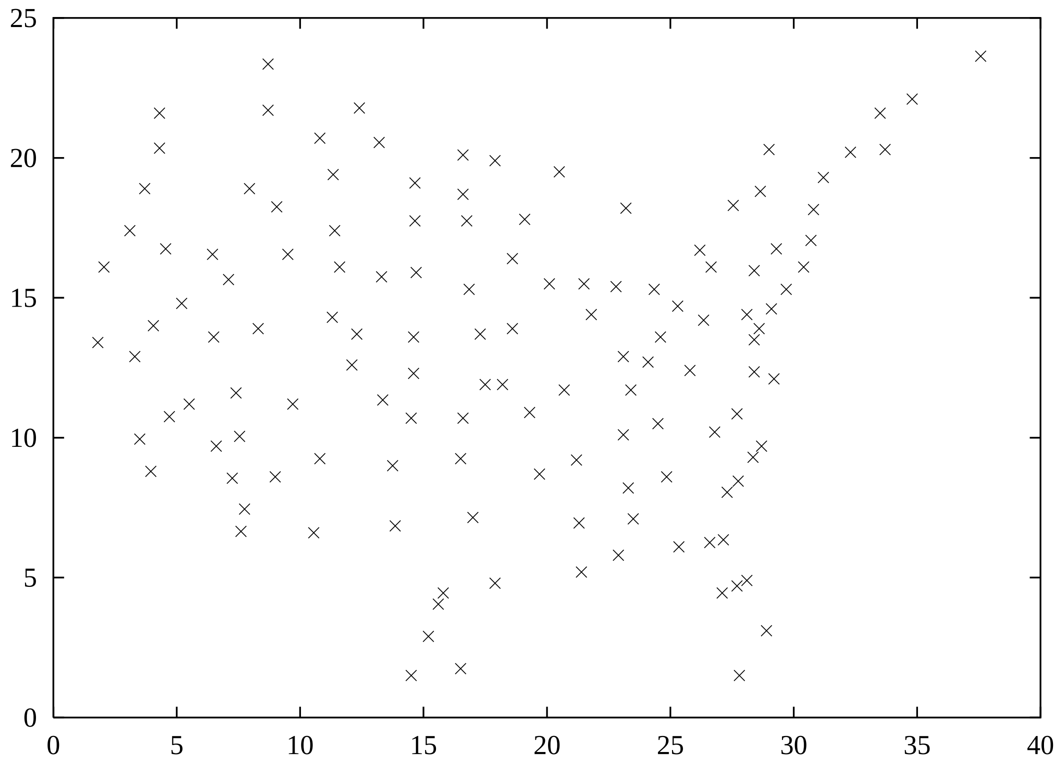

Figure 3: The locations of 134 American cities.

By the decomposition technique in [GNR], the space requirement can be reduced to $O(n^2)$. The optimal fractional solutions are transformed into integral solutions by our deterministic greedy algorithm.

The results are shown in Figure 4, which indicate that the performance of our greedy algorithm is very similar to that of the optimal fractional solution and much better than the performance guarantee suggests.

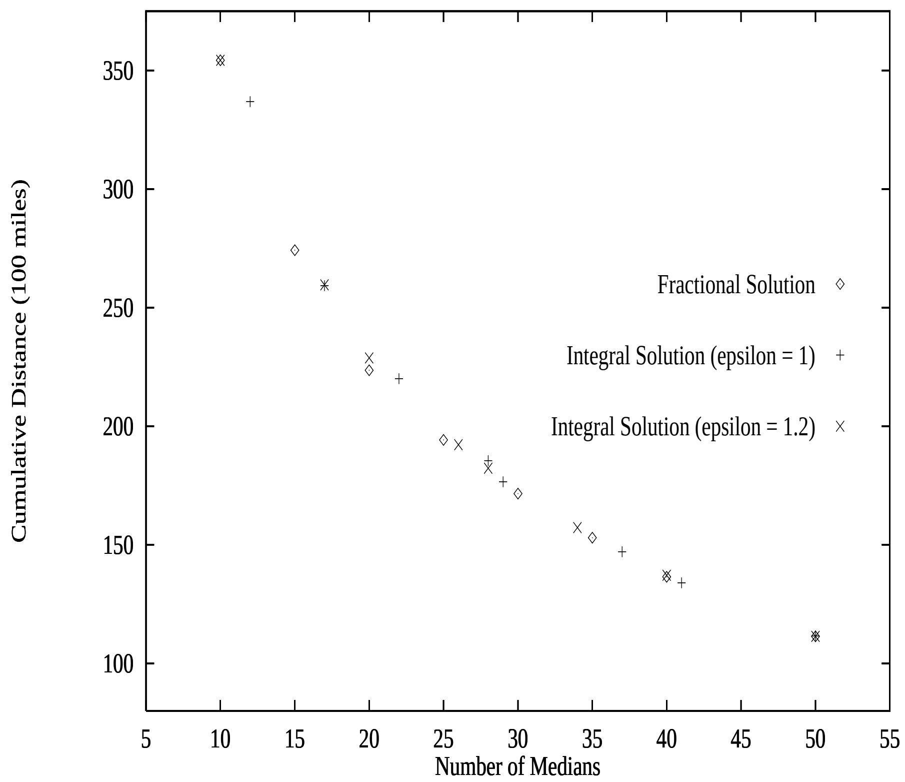

Figure 4: The data set consists of the locations of 134 American cities. The fractional solutions are transformed into integral solutions by the deterministic greedy algorithm.