Bicriteria Compact Location Problems

Sven O. Krumke [1] *H. Noltemeier* [1] *S. S. Ravi* [2] *Madhav V. Marathe* [3]

December 11, 1995

Abstract

We investigate the complexity and approximability of some location problems when two distance values are specified for each pair of potential sites. These problems involve the selection of a specified number of facilities (i.e. a placement of a specified size) to minimize a function of one distance metric subject to a budget constraint on the other distance metric. Such problems arise in several application areas including statistical clustering, pattern recognition and load–balancing in distributed systems. We show that, in general, obtaining placements that are near-optimal with respect to the first distance metric is $\mathcal{NP}$–hard even when we allow the budget constraint on the second distance metric to be violated by a constant factor. However, when both the distance metrics satisfy the triangle inequality, we present approximation algorithms that produce placements which are near-optimal with respect to the first distance metric while violating the budget constraint only by a small constant factor. We also present polynomial algorithms for these problems when the underlying graph is a tree.

1 Introduction and Motivation

In this paper, we study some location problems with multiple constraints. The problems considered in this paper can be termed as *compact location* problems, since we will typically be interested in finding a "compact" placement of facilities, i.e. a placement minimizing some measure of the distances between the selected nodes. Compact location problems without multiple constraints have been studied extensively in the past (see [RKM$^+$93, AI$^+$91] and the references cited there in).

To illustrate the types of problems considered in this paper, we present the following example. Suppose we are given *two* weight–functions c, d on the edges of the network. Let the first weight function c represent the cost of constructing an edge, and let the second weight function d represent the actual transportation– or communication–cost over an edge (once it has been constructed). Given such a graph, we can define a general bicriteria problem $(\mathcal{A}, \mathcal{B})$ by identifying two minimization objectives of interest from a set of possible objectives. A budget value is specified on the second objective $\mathcal{B}$ and the goal is to find a placement of facilities having minimum possible value for the first objective $\mathcal{A}$ such that this solution obeys the budget constraint on the second objective. For example, consider the *Diameter–Constrained Minimum Diameter Problem* denoted by DC–MDP: Given an undirected complete graph $G = (V, E)$ with two nonnegative integral edge weight functions c (modeling the building cost) and d (modeling the delay or the communication cost), an integer p denoting the number of facilities to be placed, and an integral bound B (on the total delay), find a placement of p facilities with minimum diameter under the c–cost such that the diameter of the placement under the

[1] University of Würzburg, Am Hubland, 97074 Würzburg, Germany. Email: {krumke, noltemei}@informatik.uni-wuerzburg.de

[2] Department of Computer Science, University at Albany - SUNY, Albany, NY 12222. Email: {ravi}@cs.albany.edu.

[3] Los Alamos National Laboratory P.O. Box 1663, MS K990, Los Alamos NM 87545. Email: madhav@c3.lanl.gov.
The work is supported by the Department of Energy under Contract W-7405-ENG-36.

d–costs (the maximum delay between any pair of nodes) is at most B. We term such problems as *bicriteria compact location problems.*
Here, we study the complexity and approximability of bicriteria compact location problems such as the ones mentioned above. Our study of these problems is motivated by practical problems arising in diverse areas such as statistical clustering, pattern recognition, processor allocation and load–balancing (see [HM79, MF90, KN$^+$95a] and the references cited therein).

2 Preliminaries and Problem Formulation

We consider a complete undirected n–vertex graph $G = (V, E)$. Given an integer p, a *placement* P is a subset of V with $|P| = p$. The *set of neighbors* of a vertex v in G, denoted by $N(v, G)$, is defined by $N(v, G) := \{w : (v, w) \in E\}$. The *degree* $\deg(v, G)$ of v in G is the number of vertices in $N(v, G)$. For a subset $V' \subseteq V$ of nodes, we denote by $G[V']$ the subgraph of G induced by V'. Given a graph $G = (V, E)$, the graph $G^2 = (V, E^2)$ is defined by $(u, v) \in E^2$ if and only if there is a path in G between u and v consisting of at most two edges.
If the edge distances are allowed to be zero, then the optimal solution value may be zero. In a such case, obtaining a solution whose value is within some factor of the optimal solution value is trivially equivalent to finding an optimal solution itself. Therefore, we assume that the values of both the distance functions for any edge are strictly *positive*.
With $\delta \in \{c, d\}$ denoting one of the two edge–weight functions, we use $\mathcal{D}_\delta(P)$ to denote the *diameter* and $\mathcal{S}_\delta(P)$ to denote the *sum of the distances* between the nodes in the placement P; that is

$$\mathcal{D}_\delta(P) = \max_{\substack{u,v \in P\\ u \neq v}} \delta(u, v) \qquad \text{and} \qquad \mathcal{S}_\delta(P) = \sum_{\substack{u,v \in P\\ u \neq v}} \delta(u, v).$$

We note that the average length of an edge in a placement P equals $\frac{2}{p(p-1)}\mathcal{S}_\delta(P)$. Since the average length of an edge in a placement differs from the total length of all the edges in the placement by only the scaling factor $\frac{2}{p(p-1)}$, finding a placement of minimum average length is equivalent to finding a placement of minimum total length. We use this fact throughout this paper.
As usual, we say that $\delta \in \{c, d\}$ satisfies the *triangle inequality* if we have $\delta(v, w) \leq \delta(v, u) + \delta(u, w)$ for all $v, w, u \in V$. Following [HS86], the *bottleneck graph* $\mathrm{bottleneck}(G, \delta, \Delta)$ of $G = (V, E)$ with respect to δ and a bound Δ is defined by

$$\mathrm{bottleneck}(G, \delta, \Delta) := (V, E'), \text{ where } E' := \{e \in E : \delta(e) \leq \Delta\}.$$

We now define the problems studied in this paper.

Definition 2.1 [*Diameter Constrained Minimum Average Placement Problem* (DC–MAP)]
Input: An undirected complete graph $G = (V, E)$ with two positive edge weight functions $c, d : E \to \mathbb{Q}^+$, an integer $2 \leq p \leq n$ and a number $\Omega \in \mathbb{Q}^+$.
Output: A set $P \subseteq V$, with $|P| = p$, minimizing the objective

$$\mathcal{S}_c(P) = \sum_{\substack{v,w \in P\\ v \neq w}} c(v, w)$$

subject to the constraint

$$\mathcal{D}_d(P) = \max_{\substack{v,w \in P\\ v \neq w}} d(v, w) \leq \Omega.$$

Definition 2.2 [*Sum Constrained Minimum Average Placement Problem* (SC–MAP)]
Input: Same as in DC–MAP above.

Output: A set $P \subseteq V$, with $|P| = p$, minimizing the objective

$$\mathcal{S}_c(P) = \sum_{\substack{v,w \in P \\ v \neq w}} c(v, w)$$

subject to the constraint

$$\mathcal{S}_d(P) = \sum_{\substack{v_i, v_j \in P \\ v_i \neq v_j}} d(v_i, v_j) \leq \Omega.$$

The *Sum Constrained Minimum Diameter Placement Problem* (SC–MDP) and the *Diameter Constrained Minimum Diameter Placement Problem* (DC–MDP) can be defined similarly. Given a problem Π, we use TI-Π to denote the problem Π restricted to graphs in which both the edge weight functions satisfy the triangle inequality.

We also investigate the existence of "good" solutions for bicriteria compact location problems when input graphs are restricted to be trees. In such a case, the distance between any two vertices u and v is the length of the path in the tree between u and v. Given a problem Π, we use TREE-Π to denote the problem Π restricted to trees.

Let $\Pi \in$ {SC–MAP, DC-MAP, TI–DC–MAP, TI–SC–MAP}. Define an (α, β)*–approximation algorithm* for Π to be a polynomial–time algorithm, which for any instance I of Π does one of the following:

(a) It produces a solution within α times the optimal value with respect to the first distance function c, violating the constraint with respect to the second distance function d by a factor of at most β.

(b) It returns the information that no feasible placement exists at all.

Notice that if there is no feasible placement but there is a placement violating the constraint by a factor of at most β, an (α, β)–approximation algorithm has the choice of performing either action (a) or (b).

3 Summary of Results

In this paper, we present both $\mathcal{NP}$–hardness results and approximation algorithms with provable performance guarantees for several bicriteria compact location problems. For additional results on these types of problems, we refer the reader to a companion paper [KN$^+$95a]. Our results are based on two basic techniques. The first is an application of a *parametric search technique* discussed in [MR$^+$95] for network design problems. The second is the *power of graphs* approach introduced by Hochbaum and Shmoys [HS86]. Our results for complete graphs are summarized in Table 1. The table contains hardness results and performance ratios for finding compact placements for different pairs of minimization objectives. The horizontal entries denote the objective function. For example the entry in row i, column j denotes the performance guarantee for the problem of minimizing objective j with a budget on the objective i.

→ Object. ↓ Budget	Diameter	Sum
Diameter	approximable within $(2,2)$ [KN+95a] not approximable within $(2-\varepsilon,2)$ or $(2,2-\varepsilon)$	approximable within $(2-\frac{2}{p},2)^*$ not approximable within $(\alpha,2-\varepsilon)^*$
Sum	approximable within $(2,2-\frac{2}{p})$ [KN+95a] not approximable within $(2-\varepsilon,\alpha)$	approximable within $((1+\gamma)(2-\frac{2}{p}),(1+\frac{1}{\gamma})(2-\frac{2}{p}))^*$

Table 1. Performance guarantee results for constrained compact location in a complete graph with edge weights obeying the triangle inequality. Asterisks indicate results obtained in this paper. $\gamma > 0$ is a fixed accuracy parameter. The non-approximability results stated assume that $\mathcal{P} \neq \mathcal{NP}$.

→ Object. ↓ Budget	Diameter	Sum
Diameter	polynomial time solvable	polynomial time solvable
Sum	polynomial time solvable	$\mathcal{NP}$–hard approximable within $(1+\gamma,1+\frac{1}{\gamma})$

Table 2. Results for constrained compact location in tree networks.

4 Related Work

As mentioned earlier, problems involving the placement of p facilities so as to minimize suitable cost measures have been studied extensively in the literature. These problems can roughly be divided into two main categories. The first category of problems involves selecting a set of p facilities so as to minimize (or maximize) the distance (cost) from the unselected sites to the selected sites. Problems that can be cast in this framework include the p–center problem [HS86, DF85], the p–cluster problem [HS86, FG88, Go85] and the p–median problem [LV92, MF90]. The second category consists of problems where the goal is to select p facilities so as to optimize a certain cost measure defined on the set of selected facilities. Problems that can be cast in this framework include the p–dispersion problem [RRT91, EN89], the p–minimum spanning tree problem [RR+94, GH94, AA+94, BCV95] and the p–compact location problem [RKM+93, AI+91, KN+95a].

In contrast, not much work has been done in finding optimal location of facilities when there is more than one constraint. A notable work in this direction is by Bar-Ilan and Peleg [BP91] who considered the problem of assigning network centers, with a bound imposed on the number of nodes that any center can service. We refer the reader to [MR+95, RMR+93] for a survey of the work done in the area of algorithms for bicriteria network design and location theory problems. In [KN+95a], we studied the minimum diameter problems under sum and diameter constraints. There we gave efficient approximation algorithms with constant performance guarantees for these problems when both the edge weight functions obey the triangle inequality.

Due to lack of space, the rest of the paper consists of statements of results and selected proof sketches.

5 Problems for General Graphs

5.1 Diameter Constrained Problems

We begin with a non-approximability result for DC–MAP and TI–DC–MAP. The proof this result uses a reduction from the Clique problem [GJ79].

Theorem 5.1 If the distance functions c, d are not required to satisfy the triangle inequality, there can be no polynomial time (α, β)–approximation algorithm for DC–MAP for any fixed $\alpha, \beta \geq 1$, unless $\mathcal{P} = \mathcal{NP}$. Moreover, if there is a polynomial time $(\alpha, 2-\varepsilon)$–approximation algorithm for TI–DC–MAP for any fixed $\alpha \geq 1$ and $\varepsilon > 0$, then $\mathcal{P} = \mathcal{NP}$.

Proof Sketch: We first consider the DC–MAP problem. Suppose there is a polynomial approximation algorithm $\mathcal{A}$ with a performance guarantee of (α, β) for some $\alpha, \beta \geq 1$. We will show that $\mathcal{A}$ can be used to solve an arbitrary instance of the Clique problem in polynomial time, contradicting the assumption that $\mathcal{P} \neq \mathcal{NP}$.
Let the graph $G = (V, E)$ and the integer J form an arbitrary instance of Clique. Construct the following instance I of DC–MAP. The vertex set for I is V itself. For all $u, v \in V$ $(u \neq v)$, let $c(u, v) = 1$; also, let $d(u, v) = 1$ if $(u, v) \in E$ and $d(u, v) = \beta + 1$ otherwise. Finally set $p = J$ to complete the construction. In the remainder of this proof sketch, we will refer to any edge in the instance I with d value equal to $\beta + 1$ as a *long* edge; other edges are referred to as *short* edges.
If G has a clique of size J, then the nodes which form this clique constitute an optimal solution to the DC–MAP instance I with sum (under c–distance) equal to $J(J-1)/2$ and diameter (under d–distance) equal to 1. Since $\mathcal{A}$ provides a performance guarantee of (α, β), the solution returned by $\mathcal{A}$ cannot include any long edges. If G does not have a clique of size J, then every subset of J nodes must include at least one long edge. Therefore, by merely examining the solution produced by $\mathcal{A}$, we can solve the Clique problem.
We use the same construction for TI–DC–MAP except that for every long edge, the d value is chosen as 2. This ensures that both the distance functions satisfy the triangle inequality. ■

Using recent hardness results from [BS94] about the non-approximability of Max Clique, we obtain the following non-approximability result.

Theorem 5.2 Let $\varepsilon > 0$ and $\varepsilon' > 0$ be arbitrary. Suppose that A is a polynomial time algorithm that, given any instance of TI–DC–MAP, either returns a subset $S \subseteq V$ of at least $\frac{2p}{|V|^{1/6-\varepsilon'}}$ nodes satisfying $\mathcal{D}_d(S) \leq (2-\varepsilon)\Omega$, or provides the information that no placement of p nodes having d–diameter of at most Ω does exist. Then $\mathcal{P} = \mathcal{NP}$. ■

We now consider the TI–DC–MAP problem where the distance functions satisfy the triangle inequality. For this problem, we present an approximation algorithm that provides a performance guarantee of $(2 - 2/p, 2)$. The algorithm is shown in Figure 1. The performance guarantee is established below.

Theorem 5.3 Let I be any instance of TI–DC–MAP such that an optimal solution P^* of total c–cost $OPT(I) = \mathcal{S}_c(P^*)$ exists. Then the algorithm HEUR–FOR–DIA–CONSTRAINT returns a placement P satisfying $\mathcal{D}_d(P) \leq 2\Omega$ and $\mathcal{S}_c(P)/OPT(I) \leq 2 - 2/p$.

Proof: Consider an optimal solution P^* such that $\mathcal{D}_d(P^*) \leq \Omega$. By definition, this placement forms a clique of size p in $G' := \mathrm{bottleneck}(G, d, \Omega)$. Consequently, for any node $v \in P^*$ the set $N(v, G')$ has size at least p and V_{cand} is non–empty. Thus the heuristic will not output a "certificate of failure".
Moreover, any placement $P(v)$ considered by the heuristic will form a clique in $(G')^2$. By the definition of G' as a bottleneck graph with respect to d, the bound Ω and the assumption

Procedure HEUR–FOR–DIA–CONSTRAINT

$G' := \mathrm{bottleneck}(G, d, \Omega)$
$V_{cand} := \{v \in G' : \deg(v, G') \geq p - 1\}$
if $V_{cand} = \emptyset$ **then return** "certificate of failure"
Let $best := +\infty$
Let $P_{best} := \emptyset$
for each $v \in V_{cand}$ **do**
   Sort the neighbors $N(v, G')$ of v according to their c–distance from v
   Assume now that $N(v, G') = \{w_1, \ldots, w_r\}$ with $c(v, w_1) \leq \cdots \leq c(v, w_r)$
   Let $P(v) := \{v, w_1, \ldots, w_{p-1}\}$
   if $\mathcal{S}_c(P(v)) < best$ **then** $P_{best} := P(v)$
                              $best := \mathcal{S}_c(P(v))$
output P_{best}

Figure 1: Details of the heuristic for TI–DC–MAP.

that the edge weights obey the triangle inequality, it follows that no edge e in $(G')^2$ has d–weight more than 2Ω. Thus, for *every* placement $P(v)$ considered by the heuristic, the value of $\mathcal{D}_d(P(v))$ is no more than 2Ω.

Now we are going to establish the performance guarantee with respect to the objective function value. To this end, define for a node $v \in P^*$: $S_v := \sum_{\substack{w \in P^* \\ w \neq v}} c(v, w)$. Then we have $\mathcal{S}_c(P^*) = \sum_{v \in P^*} S_v$. Now let $v \in P^*$ be so that S_v is a minimum among all nodes in P^*. Then clearly

$$OPT(I) = \mathcal{S}_c(P^*) \geq p S_v. \tag{1}$$

As mentioned earlier, $v \in V_{cand}$. Consider the step of the algorithm HEUR–FOR–DIA–CONSTRAINT in which it examines v. Let $N(v) := P(v) \setminus \{v\}$ denote the set of $p - 1$ nearest neighbors of v in G' with respect to c. Then we have

$$\sum_{\substack{w \in N(v) \\ w \neq v}} c(v, w) \leq S_v, \tag{2}$$

by definition of $N(v)$ as the set of nearest neighbors. Let $w \in N(v)$ be arbitrary. Then

$$\begin{aligned}
\sum_{u \in N(v) \cup \{v\} \setminus \{w\}} c(w, u) &= c(w, v) + \sum_{u \in N(v) \setminus \{w\}} c(w, u) \\
&\leq c(w, v) + \sum_{u \in N(v) \setminus \{w\}} (c(w, v) + c(v, u)) \\
&= (p - 1) c(w, v) + \sum_{u \in N(v) \setminus \{w\}} c(v, u) \\
&= (p - 2) c(v, w) + \sum_{u \in N(v)} c(v, u) \\
&\overset{(2)}{\leq} (p - 2) c(v, w) + S_v.
\end{aligned} \tag{3}$$

Procedure HEUR–FOR–SUM–CONSTRAINT

1 Use a binary search to find the smallest integer $T \in [0, p^2 \max\{ c(e) : e \in E \}]$ such that Sum–Test(T)=Yes.

2 **output** the placement generated by Sum–Test(T).

Figure 2: Main procedure for TI–SC–MAP.

Now using (3) and again (2), we obtain

$$\begin{aligned}
\mathcal{S}_c(P(v)) &= \mathcal{S}_c(N(v) \cup \{v\}) \\
&= \sum_{u \in N(v)} c(v,u) + \sum_{w \in N(v)} \sum_{u \in N(v) \cup \{v\} \setminus \{w\}} c(w,u) \\
&\overset{(2)}{\leq} S_v + \sum_{w \in N(v)} \sum_{u \in N(v) \cup \{v\} \setminus \{w\}} c(w,u) \\
&\overset{(3)}{\leq} S_v + \sum_{w \in N(v)} ((p-2)c(v,w) + S_v) \\
&= S_v + (p-2)S_v + (p-1)S_v \\
&= (2p-2)S_v \\
&\overset{(1)}{\leq} (2 - 2/p)OPT(I).
\end{aligned}$$

As the algorithm chooses the placement P_{best} with the least $\mathcal{S}_c$, the claimed performance guarantee follows. ■

5.2 Sum Constrained Problems

Next, we study bicriteria compact location problems where the objective is to minimize the sum of the distances $\mathcal{S}_c$ subject to a budget–constraint on $\mathcal{S}_d$.
Again, it is not an easy task to find a placement P satisfying the budget–constraint or to determine that no such placement exists. Using a reduction from Clique [GJ79] similar to that used in the proof of Theorem 5.1, we get the following result.

Proposition 5.4 If the distance functions c, d are not required to satisfy the triangle inequality, there can be no polynomial time (α, β)–approximation algorithm for SC–MAP for any fixed $\alpha, \beta \geq 1$, unless $\mathcal{P} = \mathcal{NP}$. ■

We proceed to present a heuristic for TI–SC–MAP. The main procedure shown in Figure 2 uses the test procedure from Figure 3. We note that γ is a fixed quantity that specifies the accuracy requirement.
For a value of T let $OPT_h(T)$ denote the sum of the distances of an optimal placement of p nodes with respect to the distance function $h(v,w) := c(v,w) + \frac{T}{\Omega} d(v,w) = c(v,w) + \mu d(v,w)$; i.e.,

$$OPT_h(T) = \min_{\substack{P \subseteq V \\ |P|=p}} \mathcal{S}_h(P).$$

Then we have the following lemma:

Lemma 5.5 The function $\mathcal{R}(T) = \frac{OPT_h(T)}{T}$ is monotonically nonincreasing on $\mathbb{Q} \setminus \{0\}$. ■

Procedure Sum–Test(T)
1 Let $\mu := \frac{T}{\Omega}$.
2 **for** each pair (v, w) of nodes define the distance function $h(v, w)$ by $h(v, w) := c(v, w) + \mu d(v, w)$.
3 Compute a $(2-2/p)$–approximation for the problem of finding a set of p nodes minimizing $\mathcal{S}_h$.
4 Let P_T be a set of p nodes with $\mathcal{S}_h(P_T) \leq (2-2/p) \cdot \min_{\substack{P \subseteq V \\ |P|=p}} \mathcal{S}_h(P)$.
5 **if** $\mathcal{S}_h(P_T) \leq (2-2/p)(1+\gamma)T$ **then output** Yes **else output** No.

Figure 3: Test procedure used for TI–SC–MAP.

Proof: Suppose for the sake of contradiction that for two values of T, say T_1 and T_2 with $T_1 < T_2$, we have that $\mathcal{R}(T_1) < \mathcal{R}(T_2)$. Let P_1 and P_2 denote optimal placements of p nodes under h when $T = T_1$ and $T = T_2$ respectively. For $i \in \{1, 2\}$, let C_i and D_i denote the costs of placement P_i under c and d respectively. Thus, we have that $\mathcal{R}(T_i) = \frac{C_i}{T_i} + \frac{D_i}{\Omega}$ for $i \in \{1, 2\}$. Consider the cost under h of the placement P_1 when $T = T_2$. By the definition of C_1 and D_1, it follows that the cost of P_1 is $C_1 + \frac{D_1 \cdot T_2}{\Omega}$. Thus the value of $\mathcal{R}(T_2)$ is at most this cost divided by T_2 which is $\frac{C_1}{T_2} + \frac{D_1}{\Omega}$. This in turn is less than $\frac{C_1}{T_1} + \frac{D_1}{\Omega}$, since $T_1 < T_2$. But $\frac{C_1}{T_1} + \frac{D_1}{\Omega}$ is exactly $\mathcal{R}(T_1)$, and this contradicts the assumption that $\mathcal{R}(T_1) < \mathcal{R}(T_2)$. ■

Now we can establish the result about the performance guarantee of the heuristic. Let $OPT(I) = \mathcal{S}_c(P^*)$ denote the function value of an optimal placement P^* of p nodes. To simplify the analysis, we assume that $OPT(I)/\gamma$ is an integer. This can be enforced by first scaling the cost function c so that all values are integers and then scaling again by γ.

Theorem 5.6 Let I denote any instance of TI–SC–MAP and assume that there is an optimal placement P^* with $OPT(I) = \mathcal{S}_c(P^*)$. Then HEUR–FOR–SUM–CONSTRAINT with the test procedure Sum–Test returns a placement P with $\mathcal{S}_d(P) \leq (1+\gamma)(2-2/p)\Omega$ and $\mathcal{S}_c(I)/OPT(I) \leq (2-2/p)(1+1/\gamma)$.

Proof: Consider the call to the procedure Sum–Test when $T = T^* = OPT(I)/\gamma$. Notice that T^* is an integer by our assumption. The h–cost of the placement P^* is then $OPT(I) + \frac{T^*}{\Omega}\Omega = OPT(I) + T^* = (1+\gamma)T^*$. Thus we have $OPT_h(T^*) \leq (1+\gamma)T^*$ and the $(2-2/p)$–approximation P_T that is computed in step 3 will satisfy $\mathcal{S}_h(P_T) \leq (2-2/p)OPT_h(T^*) \leq (2-2/p)(1+\gamma)T^*$. Thus, we observe that the procedure will return Yes and that $\mathcal{R}(T^*) \leq 1+\gamma$. Further, the value T found by the binary search in the main procedure satisfies $T \leq T^*$, since T is the minimum value such that Sum–Test(C') returns Yes. Let P_T be the corresponding placement that is returned by Sum–Test. Then we have

$$\mathcal{S}_c(P_T) \leq \mathcal{S}_h(P_T) \leq (2-\frac{2}{p})(OPT(I) + \frac{T}{\Omega}\Omega) \leq (2-\frac{2}{p})(1+\frac{1}{\gamma}) \cdot OPT(I).$$

Moreover, we see that

$$\frac{T}{\Omega}\mathcal{S}_d(P_T) \leq \mathcal{S}_h(P_T) \leq (2-\frac{2}{p})(1+\gamma)T,$$

and multiplying the last chain of inequalities by Ω/T yields

$$\mathcal{S}_d(P_T) \leq (2-2/p)(1+\gamma)\Omega$$

and this completes the proof. ■

6 Problems for Tree Networks

In this section we study the constrained compact location problems for tree networks. In this case the distances between two vertices correspond to the path lengths along the trees.

Definition 6.1 A tree based distance structure τ is a set $V = \{v_1, v_2, \cdots v_n\}$ of n vertices, a spanning tree T on these vertices, and two non–negative lengths $c(e), d(e)$ assigned to each edge of the tree. For each pair v_i, v_j of vertices, the distances $c(v_i, v_j)$ and $d(v_i, v_j)$ implied by τ are the sum of the corresponding edge lengths along the unique path in T connecting v_i and v_j.

Versions of compact location problems can be defined for trees, in the same manner as we defined for arbitrary graphs but the distances are now specified by a tree-based distance structure. We denote these problems by TREE–DC–MAP and TREE–SC–MAP respectively. For instance, for the TREE–DC–MAP problem the input is a tree based distance structure, an integer p and a bound Ω. The requirement is to find a subset consisting of p nodes, such that the sum of the c–distances between the nodes is minimized and the diameter with respect to the d–distance does not exceed the bound Ω.
It has been shown in [RKM+93] that the *unconstrained* problems, TREE–MAP and TREE–MDP, which involve finding a subset of p nodes minimizing the sum of the c–distances and the c–diameter respectively (and ignoring the d–weights on the edges), can be solved in polynomial time.

6.1 The Complexity of TREE-MAP

The following result points out that obtaining an optimal solution to the SC–MAP problem is difficult even for trees.

Proposition 6.2 SC–MAP is $\mathcal{NP}$–hard even when the underlying graph is a tree.

Proof: We use a reduction from Partition: Given a multiset of (not necessarily distinct) positive integers $\{a_1, \ldots, a_n\}$ the question is whether there exists a subset $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$. Partition is known to be $\mathcal{NP}$–complete (cf. [GJ79]).
Given any instance of Partition we construct a star–shaped graph G having $n+1$ nodes $\{x, y_1, \ldots, y_n\}$ and n edges (x, y_i), $i = 1, \ldots, n$. We then define $c(x, y_i) := a_i$ and $d(x, y_i) := D - a_i$, where $D := \sum_{i=1}^{n} a_i$.
We then run the hypothetical polynomial time algorithm A for SC–MAP for the instance I_j consisting of the graph defined as above and the parameters p_j, Ω_j, $j = 1, \ldots, n$ where $p_j := j+1$ and $\Omega_j := 2(j-1)jD - (j-1)D$. Observe that this will still result in an overall polynomial time.
Let $j \in \{1, n\}$ be fixed and assume that P is any placement of $p_j = j+1$ nodes that includes the node x. It then follows that

$$\mathcal{S}_c(P) = 2(j-1) \sum_{x_j \in P} a_j \qquad \text{and} \qquad \mathcal{S}_d(P) = 2(j-1)jD - 2(j-1) \sum_{x_j \in P} a_j. \tag{4}$$

Moreover, if P is any *feasible* placement for I_j (i.e., $\mathcal{S}_d(P) \leq \Omega_j$) that includes the node x, then using the feasibility, equation (4) and the definition of Ω_j, we obtain

$$S_d(P) = 2(j-1)jD - 2(j-1) \sum_{x_j \in P} a_j \leq 2(j-1)jD - (j-1)D.$$

Thus for such a placement we get

$$\sum_{x_j \in P} a_j \geq D/2. \tag{5}$$

So far we have considered only placements that include the node x. The striking point now is that any optimal feasible placement for I_j must indeed include x. This follows from the fact that replacing any node x_j in the placement by the node x will decrease *both* S_c and S_d. Hence using (4) and (5) we see that for any optimal placement P for I_j we have

$$\mathcal{S}_c(P) = 2(j-1) \sum_{x_j \in P} a_j \geq (j-1)D. \tag{6}$$

Assume that there is a partition I with $|I| = j$ elements. Then, if we choose the placement $P_j := \{x\} \cup \{y_j : a_j \in I\}$ for the instance I_j, we get

$$\mathcal{S}_d(P_j) = 2(j-1)jD - 2(j-1) \sum_{x_j \in P} a_j = 2(j-1)j - (j-1)D = \Omega_j.$$

Thus the placement is feasible. Moreover,

$$\mathcal{S}_c(P) = 2(j-1) \sum_{x_j \in P} a_j = (j-1)D. \tag{7}$$

Hence, by (6) this placement is optimal and the bound from equation (6) is satisfied as an equality.
Assume conversely that there is an optimal placement for some I_j where the bound from (6) is satisfied as an equality, i.e., equation (7) holds. If we let $I := \{j : x_j \in P\}$, we then have $\sum_{i \in I} a_i = \sum_{x_i \in P} a_i = D/2$.
Thus by running the hypothetical algorithm A on all the instances I_j, $j = 1, \ldots, n$ and inspecting the optimum function value $\mathcal{S}_c$ we can decide whether or not the given instance of Partition has a solution. ■

Given that TREE–SC–MAP is $\mathcal{NP}$–hard we investigate the existence of efficient approximation algorithms for it. By combining the parametric search technique from section 5.2 with the polynomial time algorithm in [RKM+93], for solving TREE–MAP (unconstrained version) optimally, we can obtain approximation algorithm for TREE–SC–MAP with performance guarantee $(1+\gamma, 1+1/\gamma)$. Thus we have the following theorem.

Theorem 6.3 For any fixed $\gamma > 0$ there is a polynomial time algorithm which, given any instance of TREE–SC–MAP such that there exists an optimal solution P^* of total c–cost $OPT(I) = \mathcal{S}_c(P^*)$ exists, finds a placement P of total d–cost $\mathcal{S}_d(P)$ no more than $(1+\gamma)\Omega$ and satisfying $\mathcal{S}_c(P) \leq (1+1/\gamma)OPT(I)$. ■

6.2 Polynomial Time Solvable Subcases

While TREE–SC–MAP is $\mathcal{NP}$–hard, it turns out that the other three constrained compact location problems for trees (namely TREE–DC–MAP, TREE–DC–MDP and TREE–SC–MDP) are polynomial time solvable.
Here, we outline our idea for the TREE–DC–MAP problem. Polynomial time solvability for the other problems follows the same outline and is omitted in this version of the paper.

Theorem 6.4 TREE–DC–MAP can be solved in polynomial time.

Proof Sketch: It is easy to see that if two vertices a and b are in a solution, then each vertex on the unique path between a and b can also be added to the solution without violating the diameter constraint and also without increasing the value of the sum cost. Thus, there always exists an optimal solution which is connected; that is, there is an optimal solution which is a subtree of the original tree.

Procedure TREE–DC–MAP
1 **for** each $v, w \in V$ **do**
2 Let L be the d–distance between u and v.
if $L > \Omega$ (the diameter constraint) **then** go to the next iteration.
Prune the tree I to obtain a new tree $T_{u,v}(V_1, E_1)$ such that
every pair of nodes in $T_{u,v}$ is within a d–distance of L.
if $|V_1| < p$, then start the next iteration of the for loop.
3 Solve the unconstrained compact location problem with distances given by
c
on the tree $T_{u,v}$ *optimally* in polynomial time using the algorithm in
[RKM+93].
Let $P(u, v)$ the placement obtained this way.
4 **output** the best placement $P(u, v)$.

Figure 4: Details of the heuristic for TREE–DC–MAP

Consider an optimal solution T (i.e., a subtree of the original tree with p nodes) for an instance I of TREE–DC–MAP. Let L be the diameter of the tree with respect to distance function d and let a and b be the vertices in T which are at a distance of L from each other. For this proof sketch, let us assume that the cost with respect to the distance function d is integral and also that it is polynomially bounded. (The general case can be handled in a manner similar to the algorithm for the minimum diameter p-spanning tree problem discussed in [RR+94].) Let us subdivide the edge by placing a dummy node r on it in such a way that $d(a, r) = d(b, r) = L/2$. Next, we prune the tree given by the instance I to obtain T_1 as follows. We delete all vertices in I which are at a distance more than $L/2$ from the point r. Then the pruned tree T_1 has the following desirable property. Every pair of vertices in T_1 is within a d–distance of L from each other. Now we solve the TREE–MAP problem on T_1 using the procedure outlined in [RKM+93]. By repeating this procedure for each pair of vertices a and b such that the d–distance between a and b is at most Ω and choosing a placement with the minimum sum cost with respect to the c–distance, we obtain an optimal solution to the TREE–DC–MAP instance I. ■

The algorithm for TREE–DC–MAP resulting from the above discussion is outlined in Figure 4.

References

[AI+91] A. Aggarwal, H. Imai, N. Katoh, and S. Suri. Finding k points with Minimum Diameter and Related Problems. *J. Algorithms*, 12(1):38–56, March 1991.

[BP91] J. Bar-Ilan and D. Peleg. Approximation Algorithms for Selecting Network Centers. In *Proc. 2nd Workshop on Algorithms and Data Structures (WADS)*, pages 343–354, Ottawa, Canada, August 1991. Springer Verlag, LNCS vol. 519.

[AA+94] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala, Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing* (STOC'95), pp. 277–376.

[BCV95] A. Blum, P. Chalasani and S. Vempala, A Constant-Factor Approximation for the k-MST Problem in the Plane. *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing* (STOC'95), pp. 294–302.

[BS94] M. Bellare and M. Sudan. Improved Non–Approximability Results. in *Proceedings of the 26th annual ACM Symposium on the Theory of Computing (STOC)*, May 1994.

[DF85] M.E. Dyer and A.M. Frieze. A Simple Heuristic for the p–Center Problem. *Operations Research Letters*, 3(6):285–288, Feb. 1985.

[EN89] E. Erkut and S. Neuman. Analytical Models for Locating Undesirable Facilities. *European J. Operations Research*, 40:275–291, 1989.

[FG88] T. Feder and D. Greene. Optimal Algorithms for Approximate Clustering. In *ACM Symposium on Theory of Computing (STOC)*, pages 434–444, 1988.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability*. W.H. Freeman, 1979.

[Go85] T.F. Gonzalez. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science*, 38:293–306, 1985.

[GH94] N. Garg, D. Hochbaum, An $O(\log n)$ Approximation for the k-minimum spanning tree problem in the plane. *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing* (STOC'95), pp. 294–302.

[HM79] G.Y. Handler and P.B. Mirchandani. *Location on Networks: Theory and Algorithms*. MIT Press, Cambridge, MA, 1979.

[HS86] D. S. Hochbaum and D. B. Shmoys. A Unified Approach to Approximation Algorithms for Bottleneck Problems. *Journal of the ACM*, 33(3):533–550, July 1986.

[KN+95a] S.O. Krumke, H. Noltemeier, S.S. Ravi and M.V. Marathe, Compact Location Problems with Budget and Communication Constraints. In *1st International Conference on Computing and Combinatorics,* X'ian, China, August 1995.

[Lee82] D.T. Lee. On k–nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Comput.*, C–31:478–487, 1982.

[LV92] J.H. Lin and J. S. Vitter. ε–Approximations with Minimum Packing Constraint Violation. In *ACM Symposium on Theory of Computing (STOC)*, pages 771–781, May 1992.

[MR+95] M.V. Marathe, R. Ravi, R. Sundaram, S.S. Ravi, D.J. Rosenkrantz, and H.B. Hunt III. Bicriteria Network Design Problems. To appear in *Proceedings of the 22nd International Colloquium on Automata Languages and Programming (ICALP)*, 1995.

[MF90] P.B. Mirchandani and R.L. Francis. *Discrete Location Theory*. Wiley–Interscience, New York, NY, 1990.

[PS85] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer–Verlag Inc., New York, NY, 1985.

[RKM+93] V. Radhakrishnan, S.O. Krumke, M.V. Marathe, D.J. Rosenkrantz, and S.S. Ravi. Compact Location Problems. In *13th Conference on the Foundations of Software Technology and Theoretical Computer Science (FST-TCS)*, volume 761 of *LNCS*, pages 238–247, December 1993.

[RMR+93] R. Ravi, M.V. Marathe, S.S. Ravi, D.J. Rosenkrantz, and H.B. Hunt III. Many birds with one stone: Multi-objective approximation algorithms. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 438–447, 1993.

[RRT91] S.S. Ravi, D.J. Rosenkrantz, and G.K. Tayi. Facility Dispersion Problems: Heuristics and Special Cases. In *Proc. 2nd Workshop on Algorithms and Data Structures (WADS)*, pages 355–366, Ottawa, Canada, August 1991. Springer Verlag, LNCS vol. 519. (Journal version: *Operations Research*, 42(2):299–310, March-April 1994.)

[RR+94] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi, Spanning trees short or small. *Proceedings, Fifth Annual ACM-SIAM Symposium on Discrete Algorithms,* (1994), pp 546–555. (Journal version to appear in *SIAM Journal on Discrete Mathematics*.)

[MR+95] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H.B. Hunt III. Bicriteria Network Design problems, In *Proceedings of the 22nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 438–447, 1993.

[ST93] D. B. Shmoys and E. Tardos. Scheduling unrelated parallel machines with costs. In *Proceedings of the 4th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 438–447, 1993.