Theoretical Computer Science 181 (1997) 379–404

Theoretical Computer Science

Compact location problems

S.O. Krumke[a], M.V. Marathe[b,*], H. Noltemeier[a], V. Radhakrishnan[c,1], S.S. Ravi[d], D.J. Rosenkrantz[d]

[a] *University of Würzburg, Am Hubland, 97074 Würzburg, Germany*
[b] *Los Alamos National Laboratory P.O. Box 1663, MS P990, Los Alamos NM 87545, USA*
[c] *Mail stop 47LA, Hewlett-Packard Company, 19447 Pruneridge Avenue, Cupertino, CA 95014-9913, USA*
[d] *Department of Computer Science, University at Albany-SUNY, Albany, NY 12222, USA*

Résumé

We consider the problem of placing a specified number (p) of facilities on the nodes of a network so as to minimize some measure of the distances between facilities. This type of problem models a number of problems arising in facility location, statistical clustering, pattern recognition, and processor allocation problems in multiprocessor systems. We consider the problem under three different objectives, namely minimizing the diameter, minimizing the average distance, and minimizing the variance. We observe that, in general, the problem is NP-hard under any of the objectives. Further, even obtaining a constant factor approximation for any of the objectives is NP-hard.

We present a general framework for obtaining near-optimal solutions to the compact location problems for the above measures, when the distances satisfy the triangle inequality. We show that this framework can be extended to the case when there are also node weights. Further, we investigate the complexity and approximability of more general versions of these problems, where two distance values are specified for each pair of potential sites. In these cases, the goal is to a select a specified number of facilities to minimize a function of one distance metric subject to a budget constraint on the other distance metric. We present algorithms that provide solutions which are within a small constant factor of the objective value while violating the budget constraint by only a small constant factor.

1. Introduction and motivation

Several fundamental problems in location theory [18, 28] involve finding a placement of facilities obeying certain "covering" constraints. Generally, the goal of such a location problem is to find a placement of minimum cost that satisfies all the specified constraints. In general, finding a placement of sufficient generality minimizing a

* Corresponding author. E-mail: madhav@c3.lanl.gov. Supported by the Department of Energy under Contract W-7405-Eng-36.
[1] Part of the work was done when the author was at the University at Albany.

cost measure is often NP-hard [15]. Here, we consider several problems dealing with the placement of a specified number of facilities on the nodes of a given network so as to minimize some function of the distances between the facilities. Because of the nature of the placement desired, we refer to these problems as *compact location problems*. For an example of such a problem from a different application domain, consider the following processor allocation problem which arises in the context of multiprocessor systems. We are given a computational task consisting of a number of communicating subtasks. At a given time, some of the processors may already be allocated and the remaining processors are available. The problem is to select a subset of processors from the currently available set of processors, one processor per subtask, such that the cost of communication among the processors executing the subtasks is minimized. In this application, the processors must be allocated quickly, and this may conflict with the goal of minimum communication cost among the selected processors.

Such location problems are commonly modeled as problems on undirected graphs. The nodes of the graph represent the available sites. A cost is associated with each pair of sites, and it is specified as the weight of the edge joining the corresponding pair of nodes. Depending on the problem, this cost represents one of several parameters such as the cost of transporting components between the pair of sites, the cost of setting up a communication link between the pair of sites, the time required to communicate between the pair of sites, etc. In some problems, there is also a weight associated with each node. This node weight may reflect the cost of setting up a facility at the corresponding site.

Under this graph-theoretic setting, a placement is a subset of nodes of a given cardinality. The cost of a placement is a problem-specific function of the weights of the nodes and edges in the subgraph induced by the placement. Examples of such cost functions are the sum of the weights of all the edges in the placement (which may reflect the total cost of setting up communication links between each pair of chosen sites), the maximum weight of an edge in the placement (i.e., the *bottleneck cost*, which may reflect the maximum time needed to communicate between any pair of chosen sites), etc. The goal of a compact location problem is to find a placement of minimum cost.

In practice, it is often the case that a minimum cost placement must be chosen subject to budget constraints on other cost measures. We also consider problems where the goal is to find a placement that minimizes one cost measure subject to a budget constraint on another cost measure. Since these problems involve two cost functions, we refer to them as *bicriteria compact location problems*. As an example, consider once again the scenario presented above in the context of multiprocessor systems. If the processors need to communicate with each other frequently to exchange data, then these data communication delays increase the time needed to complete a task. Further, the computation may require the processors to be synchronized often, thus adding to the time needed to complete the task. Therefore, it is desirable to select a subset of processors so that the total communication cost among the processors is minimized and the maximum delay due to synchronization during the computation does not exceed a given bound.

Such compact location problems also arise in a number of other applications such as allocation of manufacturing sites for the components of a system so as to minimize the cost of transporting components, distributing the activities of a project among geographically dispersed offices so as to minimize the transportation or communication cost among the offices, statistical clustering, pattern recognition, load-balancing in distributed systems, etc. (see [3, 2, 18, 28, 22, 23] and the references cited therein).

In graph-theoretic terms, bicriteria compact location problems can be formalized as follows. Suppose we are given *two* weight functions c and d on the edges of the network. (For example, the first weight function c may represent the cost of constructing an edge, and the second weight function d may represent the actual transportation – or communication – cost over an edge once it has been constructed.) Given such a graph, a positive number B and a positive integer p, we define a general bicriteria compact location problem $(\mathscr{A}, \mathscr{B})$ by identifying two minimization objectives of interest from a set of possible objectives. The parameter B represents the budget on the second objective $\mathscr{B}$ and the goal is to find a placement of p facilities having the minimum possible value for the first objective $\mathscr{A}$ such that this solution obeys the budget constraint on the second objective. For example, consider the *Diameter-Constrained Minimum Diameter Problem* denoted by (MIN-DIA, DIA): 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 delay), find a placement of p facilities with minimum diameter under the c-cost such that the diameter of the placement under the d-costs (the maximum delay between any pair of nodes) is at most B.

The remainder of the paper is organized as follows. Section 2 contains preliminaries and the formal definitions of the problems considered in this paper. In Section 3 we summarize the results obtained. In Section 4 we discuss the related research done in this area. Section 5 contains nonapproximability results for graphs with arbitrary weights. In Section 6 we present basic approximation algorithms for the unicriterion compact location problems. Section 7 contains our approximation algorithms for the diameter-constrained bicriteria problems. In Section 8 we outline the approximation algorithms for the sum-constrained problems. Section 9 discusses some extensions of our approximation algorithms. Finally, Section 10 contains concluding remarks and directions for future research.

2. Preliminaries and problem definitions

We consider a complete undirected n-vertex graph $G=(V,E)$ with one or two distance functions specified on the edges. Given an integer p, a *placement* P is a subset of V with $|P| = p$. Let δ denote a distance function on the edges of G. We use $\mathscr{D}_\delta(P) := \max_{\substack{e=(v,w)\\ v,w\in P}} \delta(v,w)$ to denote the *diameter*, $\mathscr{S}_\delta(P) := \sum_{\substack{e=(v,w)\\ v,w\in P}} \delta(v,w)$ to denote the *sum of the distances* and $\mathscr{Q}_\delta(P) := \sum_{\substack{e=(v,w)\\ v,w\in P}} \delta^2(v,w)$ to denote the *sum of the squares of the distances* between the nodes in the placement P.

Note that the average length and the variance[2] [1] of an edge in a placement P are equal to $(2/p(p-1))\mathscr{S}_\delta(P)$ and $(2/p(p-1))\mathscr{Q}_\delta(P)$, respectively. Since these differ from the total length and the sum of the squared distances only by the respective scaling factors, finding a placement of minimum average length or minimum variance is equivalent to finding a placement of minimum total length and minimum sum of the squared distances, respectively. We use this fact throughout this paper.

As is standard in the literature, we say that a nonnegative edge-weight function δ satisfies the *triangle inequality*, if we have $\delta(v,w) \leqslant \delta(v,u) + \delta(u,w)$ for all $v,w,u \in V$. We now define the problems studied in this paper beginning with unicriterion problems.

Definition 2.1. An instance of the *minimum diameter placement problem* (MIN-DIA) is given by a complete graph $G=(V,E)$, a nonnegative edge-weight function c and an integer $2 \leqslant p \leqslant |V|$. The problem is to find a placement P (i.e., a vertex subset of size p), that minimizes $\mathscr{D}_c(P)$.

The *minimum average distance placement* (MIN-SUM), and *minimum variance placement* problem (MIN-VAR), are defined analogously. We now extend the above definition to bicriteria compact location problems.

Definition 2.2. An instance of a *diameter constrained minimum diameter placement problem* (MIN-DIA, DIA) is given by a complete graph $G=(V,E)$, *two* nonnegative edge-weight functions c and d, an integer $2 \leqslant p \leqslant |V|$ and a positive number B. The goal is to find a placement P which minimizes $\mathscr{D}_c(P)$ subject to the constraint $\mathscr{D}_d(P) \leqslant B$.

The versions for the various combinations of objectives $\mathscr{D}_\delta$, $\mathscr{S}_\delta$ and $\mathscr{Q}_\delta$ are 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 will see that the triangle inequality plays an important role in determining the approximability of the compact location problems defined above.

One of the goals of our work is to find good approximation algorithms for several compact location problems introduced here. A *relative* approximation algorithm guarantees a solution which is within a *multiplicative* constant K of the optimal value

[2] In [1], the scaling factor used to define variance is $1/p$. We use $2/p(p-1)$ for reasons of uniformity.

for every instance of the problem. The multiplicative constant K is referred to as the *performance guarantee* provided by the algorithm. This paper is concerned with the study of relative approximation algorithms for the placement problems defined above.

As shown in Section 5, unless $\mathsf{P} = \mathsf{NP}$, for several bicriteria problems considered here, it is not possible to find placements efficiently that strictly satisfy the constraint on the d-distances. This motivates the definition of a slightly relaxed version of the performance of an approximation algorithm for bicriteria problems. Formally, let Π be a bicriteria compact location problem. An (α, β)-*approximation algorithm* for Π (or an algorithm with a performance of (α, β)) is a polynomial-time algorithm, which for any instance 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).

We close this section with some basic definitions and an important observation. 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. Following [19], the *bottleneck graph* bottleneck(G, δ, M) of $G = (V, E)$ with respect to δ and a bound M is defined by

$$\text{bottleneck}(G, \delta, M) := (V, E') \qquad \text{where } E' := \{e \in E : \delta(e) \leqslant M\}.$$

Observation 2.3. *Let H be a subgraph of the complete graph $G = (V, E)$ with edge-weights $c(e)$ $(e \in E)$ satisfying the triangle inequality. Then the weight of the heaviest edge in H^2 is at most twice the weight of the heaviest edge in H.*

3. Summary of results

Here, we study the complexity and approximability of a number of unicriterion and bicriteria compact location problems. One contribution to this paper is a general framework that leads to efficient approximation algorithms with provable performance guarantees for several compact location problems.

In Section 5, we show that the compact location problems studied in this paper are NP-hard. We establish hardness results for the unicriterion compact location problems (see Definition 2.1) which also extend, with appropriate modifications, to bicriteria versions. Next, we prove that, in general, obtaining placements that are near-optimal is NP-hard. For the bicriteria versions these negative results continue to hold, even when we allow the budget constraint on the second cost function to be violated by a constant factor and one of the cost functions satisfies the triangle inequality.

Given these hardness results, we focus our attention on graphs in which *both* the weight functions satisfy the triangle inequality. In the past, a substantial amount of work has been carried out in investigating the approximability of problems arising in network design and location theory when edge weights satisfy the triangle inequality (for example, see [33, 19, 31]). We refer the reader to the paper by Bern and Eppstein [7] for a comprehensive survey of other geometric location problems. We obtain the following approximability results.

1. We provide an efficient generic method for approximating the unicriterion compact location problems when the distance function obeys triangle inequality. The procedure runs $O(n^2)$ in time. For the (Min-Dia) problem, the algorithm provides a performance guarantee of 2. We also observe that no polynomial time heuristic for the (Min-Dia) problem can provide a better performance guarantee unless $P = NP$. For the (Min-Sum) and (Min-Var) problems, our heuristics provide performance guarantees of $2-2/p$, and $4-6/p$, respectively.
2. We provide a polynomial-time approximation algorithm for TI-(Min-Dia, Dia) with performance guarantee $(2,2)$. Furthermore, we show that unless $P = NP$, no polynomial-time approximation algorithm can provide a performance guarantee of $(2-\varepsilon, 2)$ or $(2, 2-\varepsilon)$, for any $\varepsilon > 0$.
3. We give a polynomial-time approximation algorithm for TI-(Min-Dia, Dia) with performance guarantee $(2-2/p, 2)$. We also show that, unless $P = NP$, no polynomial-time approximation algorithm can provide a performance guarantee of $(\alpha, 2-\varepsilon)$, for any $\alpha \geqslant 1$, $\varepsilon > 0$.
4. We provide a polynomial-time approximation algorithm for TI-(Min-Dia, Sum) with performance guarantee $(2, 2-2/p)$. Furthermore, we show that unless $P = NP$, no polynomial-time approximation algorithm can provide a performance guarantee of $(2-\varepsilon, \alpha)$, for any $\alpha \geqslant 1$, $\varepsilon > 0$.
5. For all $\gamma > 0$, we give a polynomial-time approximation algorithm for TI-(Min-Sum, Sum) with performance guarantee $((1+\gamma)(2-2/p), (1+1/\gamma)(2-2/p))$. We also show that this algorithm can be implemented to run in time $O(n^2 \log^2 n)$ using an elegant technique of Megiddo [27].

All the above approximation results can be extended to the case when there weights are on vertices. We discuss these extensions in Section 9.

Our results are based on two basic techniques. The first is a combination of two ideas, namely the *power of graphs* approach of Hochbaum and Shmoys [19] and the local search approach for approximating single criteria compact location problems introduced in [22]. The second is an application of a *parametric search technique* similar to the one discussed in [26] for network design problems.

4. Related work

In contrast to the NP-hardness results which hold for general distance matrices, geometric versions of (Min-Dia) and (Min-Var) were shown to be solvable in polynomial time in [1]. In the geometric versions of these problems, the nodes are points in space and the distance between a pair of nodes is their Euclidean distance. For points in the plane, [1] contains an $O(p^{2.5}n\log p + n\log n)$ algorithm for the (Min-Dia) problem and an $O(p^2 n\log n)$ algorithm for the (Min-Var) problem, and it is observed that these algorithms extend to higher dimensions. These algorithms are based on the construction of pth-order Voronoi diagrams [24, 29].

Problems involving the placement of p facilities so as to minimize suitable cost measures have been studied extensively in the literature (see [7, 1, 10, 20, 30, 12] and the references cited therein). 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 cost (distance) from the unselected sites to the selected sites. Problems that can be cast in this framework include the p-center problem [19, 11], the p-cluster problem [19, 14, 17] and the p-median problem [25, 28]. 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 [31, 34, 13], and the k-minimum spanning tree problem [32, 16, 4, 8, 35].

In contrast, not much work has been done in finding optimal location of facilities when there is more than one objective. A notable work in this direction is by Bar-Ilan, Kortsarz and Peleg [5] 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 [26, 22, 23] for a survey of the work done in the area of algorithms for bicriteria network design and location theory problems.

5. Hardness results

In this section, we prove our nonapproximability results. The hardness results are first established for the unicriterion versions of compact location problems. Then we show how these hardness results imply the hardness of bicriteria versions of compact location problems. Several of our hardness results use recent hardness results concerning approximability of the MAX-CLIQUE problem obtained by Bellare and Sudan [6] (see [15] for the definition of the problem).

Theorem 5.1. *Unless* $P = NP$, *for all* $\varepsilon > 0$ *the problem* MAX-CLIQUE *does not have a polynomial-time approximation algorithm with performance guarantee* $|V|^{1/6-\varepsilon}$.

Proposition 5.2. *If the distances are not required to satisfy the triangle inequality, then unless* $P = NP$, *for any polynomial-time computable function* f *there is no* *polynomial time approximation algorithm for any of the problems* (Min-Dia), (Min-Sum), (Min-Var), *with a performance of* $f(|V|)$. *For any* $\varepsilon > 0$, TI-(Min-Dia) *is* NP-*hard to approximate within a factor of* $2 - \varepsilon$.

Proof. We first sketch the NP-hardness proofs for the (Min-Dia) and (Min-Sum) problems. The proof for (Min-Var) is similar. We use a reduction from CLIQUE. Given an instance of the CLIQUE problem given by a graph $G = (V, E)$ and an integer J, construct an instance of (Min-Dia) consisting of the complete graph on the node set V. For nodes x and y in V, let $c(x,y) = 1$ if $\{x,y\} \in E$ and let $c(x,y) = 2$ otherwise. Clearly, the distances satisfy the triangle inequality. It is straightforward to verify that G has a clique of size J if and only if we can place J facilities such that the diameter of the placement is equal to 1.

The same construction works for (Min-Sum). If G has a clique of size J, then the nodes which form this clique constitute an optimal solution for the (Min-Sum) instance with objective function value $J(J-1)/2$. If G does not have a clique of size J, then any placement of J nodes has cost at least $J(J-1)/2 + 1$.

The same proof also shows that the existence of a polynomial-time algorithm for TI-(Min-Dia). with a performance of $2 - \varepsilon$ for some $\varepsilon > 0$ implies that $\mathsf{P} = \mathsf{NP}$. When distances are not required to satisfy triangle inequality, the distance 2 in the above construction can be replaced by $f(|V|)$ for any polynomial-time computable function f. □

The NP-hardness of the unicriterion compact location problems naturally implies the hardness of the bicriteria versions. In fact, one can establish even stronger hardness results, as shown below.

Lemma 5.3. *Unless* P = NP, *for any* $\varepsilon, \varepsilon' > 0$, *there can be no polynomial-time algorithm* A *which given an arbitrary instance of* TI-(Min-Dia, Dia),

- *either returns a subset* $S \subseteq V$ *of at least* $\frac{2p}{|V|^{1/6-\varepsilon'}}$ *nodes satisfying* $\mathscr{D}_d(S) \leqslant (2-\varepsilon)B$,
- *or provides the information that a placement of p nodes having d-diameter of at most B does not exist.*

Proof. Let I' be an arbitrary instance of MAX-CLIQUE, given by a graph $G' = (V', E')$. Without loss of generality, we can assume that $E' \neq \emptyset$. We will give a many-one Turing reduction to the problem TI-(Min-Dia, Dia).

For each $2 \leqslant k \leqslant |V|$, we construct an instance $I^{(k)}$ of TI-(Min-Dia, Dia) as follows: Let $G^{(k)} = (V', E)$ $(E = \{(u,v) : u, v \in V,\ u \neq v\})$ and define $c^{(k)}, d^{(k)} : E \to \mathbb{N}$ by $c^{(k)}(e) := 1$ for all $e \in E$ and

$$d^{(k)}(e) := \begin{cases} 1 & \text{if } e \in E', \\ 2 & \text{otherwise.} \end{cases}$$

It is trivial to see that both weight functions obey the triangle inequality. We let $B^{(k)} := 1$ and $p^{(k)} := k$. Notice that the size of an instance $I^{(k)}$ is still polynomial in the size of I', and that we have constructed only polynomially many (namely, $O(|V|)$) instances. Now consider an instance $I^{(k)}$. Note that any placement P of $p^{(k)} = k$ nodes that has diameter $\mathscr{D}_{d^{(k)}}(P) \leqslant (2-\varepsilon)B = (2-\varepsilon)$ must have diameter 1 and, thus, must form a clique in the original graph G'.

Assume that the original graph G' has a clique C of size $p^{(k)} = k$. Then this clique will satisfy $\mathscr{D}_{c^{(k)}}(C) = \mathscr{D}_{d^{(k)}}(C) = 1 = B^{(k)}$. By our assumption, the algorithm A must return a set S of at least $2p/|V|^{1/6-\varepsilon'}$ nodes with d-diameter at most $(2-\varepsilon)B = (2-\varepsilon) < 2$. Thus, as noted above, the algorithm must find a placement of diameter 1, and this set will form a clique in the original graph G'.

If there is no clique of size $p^{(k)} = k$ in G', any placement P of $p^{(k)} = k$ nodes in G' must include at least one edge e of length $d^{(k)}(e) = 2 > (2-\varepsilon)$. Now, according to our assumptions about A, the algorithm has the choice of either returning a set of size at least $2p/|V|^{1/6-\varepsilon'}$ that will form a clique in the original graph or providing the information that there is no placement P of diameter at most $B = 1$.

Thus, the output of the algorithm A can be used to either obtain the information that G' does not contain a clique of size $p^{(k)} = k$ or that G' does have a clique of size at least $2p/|V|^{1/6-\varepsilon'}$.

Now, we run A for all the instances $I^{(k)}$ $(2 \leqslant k \leqslant |V|)$. Since the size of each instance $I^{(k)}$ is polynomial in the size of I' and we only have $O(|V|)$ instances, this will result in an overall polynomial-time algorithm, according to our assumptions about A. Let $m := \max\{k : \text{A returns a set } S \text{ of diameter } 1\}$. Then, by our observations from above, we can conclude that G' has a clique of size at least $2m/|V|^{1/6-\varepsilon'}$ and that there is no clique of size $m+1$ in G'. Hence, we can approximate the maximum clique number of G' by a factor of at most $((m+1)/2m) \cdot |V|^{1/6-\varepsilon'} \leqslant |V|^{1/6-\varepsilon'}$. By the results in [6] (Theorem 5.1), this would imply that $\mathsf{P} = \mathsf{NP}$. $\square$

Again, replacing the factor 2 by a suitable polynomial-time computable function f (e.g. $f = \Theta(2^{\text{poly}(|V|)})$, which given an input length of $\Omega(|V|)$ is polynomial-time computable, it can be seen that, if the triangle inequality is not required to hold, there can be no polynomial-time approximation with performance ratio $f(|V|)$ for either the optimal function value or the constraint, unless $\mathsf{P} = \mathsf{NP}$). These results are summarized in Table 1 of Section 10.

6. Unicriterion compact location problems

In this section, we present our approximation algorithms for the unicriterion versions of compact location problems. We begin (Section 6.1) by discussing our approximation algorithm for the minimum variance problem and then discuss (Section 6.2) similar heuristics for the minimum sum and minimum diameter problems.

```
procedure Gen-Alg($G = (V, E), \mathcal{M}_c$)
Comment: $G = (V, E)$ denotes the graph with edge and/or node weights. $\mathcal{M}_c$ is a function
that returns the cost of a given placement.

1. if $|V| < p$ then return "certificate of failure"
2. else
     begin
        (a) Solution $\leftarrow \emptyset$ and Value $\leftarrow \infty$.
        (b) for each node $v \in V$ do
              begin
                 i. Find $N(v) = \{v_1, \ldots, v_{p-1}\} \subseteq V - \{v\}$ such that $N(v)$ contains
                    $p-1$ nodes in $V - \{v\}$ closest to $v$.
                 ii. $P(v) = N(v) \cup \{v\}$.
                 iii. $t \leftarrow \mathcal{M}_c(v, N(v))$.
                 iv. if $t <$ Value then Solution $\leftarrow P(v)$ and Value $\leftarrow t$.
              end
     end
3. output Solution and Value.
```

Fig. 1. Generic algorithm.

6.1. Approximating minimum variance

Recall that the goal of (Min-Var) problem is to find a placement P for which $\mathcal{Q}_c(P)$, the sum of the squares of the distances between the nodes in the placement, is minimized.

Our heuristic for the (Min-Var) problem, denoted by Heur-(Min-Var), consists merely of a call to the generic procedure of Fig. 1 with G as the graph corresponding to the problem instance and $\mathcal{M}_c := \hat{\mathcal{Q}}_c$, where for a set $N \subseteq V \setminus \{v\}$ we have

$$\hat{\mathcal{Q}}_c(v, N) := \sum_{w \in N} [c(v,w)]^2. \tag{1}$$

We now establish the performance of the algorithm using the following lemma whose proof is immediate.

Lemma 6.1. *If the distances in a network satisfy the triangle inequality, then for all nodes x, y, z, $[c(x,z)]^2 \leqslant 2([c(x,y)]^2 + [c(y,z)]^2)$.*

Lemma 6.2. *Let $v \in V$ be an arbitrary node and let $N(v) = \{w_1, \ldots, w_{p-1}\}$ be the set of nearest neighbors of v in $V - \{v\}$ with respect to c. Define*

$$Q_v := \hat{\mathcal{Q}}_c(v, N(v)) = \sum_{w \in N(v)} [c(v,w)]^2. \tag{2}$$

Then $\mathcal{Q}_c(\{v\} \cup N(v)) \leqslant (2p-3)Q_v$.

Proof.

$$\mathcal{Q}_c(\{v\} \cup N(v)) = \sum_{w \in N(v) \cup \{v\}} [c(v,w)]^2 + \sum_{\substack{u, w \in N(v) \cup \{v\} \\ u \neq w}} [c(u,w)]^2$$

$$\leqslant Q_v + \sum_{\substack{u,w \in N(v)\cup\{v\} \\ u \neq w}} 2([c(u,v)]^2 + [c(v,w)]^2) \quad \text{(by Lemma 6.1)}$$

$$= Q_v + \sum_{w \in N(v)\cup\{v\}} \sum_{u \in N(v)\cup\{v,w\}} 2\,[c(v,w)]^2$$

$$= Q_v + 2(p-2) \sum_{w \in N(v)\cup\{v\}} [c(v,w)]^2$$

$$= Q_v + 2(p-2)Q_v$$

$$= (2p-3)Q_v. \qquad \square$$

Lemma 6.3. *Let I be an instance of* (Min-Var). *Let $P^* \subseteq V$ be an optimal placement and let $P \subseteq V$ be the placement produced by* Heur-(Min-Var), *respectively for the instance I. Then $\mathcal{Q}_c(P)/\mathcal{Q}_c(P^*) \leqslant 4 - 6/p$.*

Proof. For each node $w \in P^*$, let

$$R_w = \sum_{v \in P^*} [c(v,w)]^2. \tag{3}$$

We have

$$\mathcal{Q}_c(P^*) = \frac{1}{2} \sum_{w \in P^*} R_w. \tag{4}$$

Choose $v \in P^*$ such that

$$R_v = \min_{w \in P^*} R_w. \tag{5}$$

Then

$$\mathcal{Q}_c(P^*) = \frac{1}{2} \sum_{w \in P^*} R_w \geqslant \frac{1}{2} \sum_{w \in P^*} R_v = \frac{p}{2} R_v. \tag{6}$$

This yields

$$R_v \leqslant \frac{2}{p} \mathcal{Q}(P^*). \tag{7}$$

Consider the iteration of the for loop (Step 2(b) of (Fig. 1), where the node v chosen above is considered by the heuristic. Let $N(v) = \{w_1, \ldots, w_{p-1}\}$ be the set of nearest neighbors of v in $V - \{v\}$ chosen in that iteration. By definition of R_v and Q_v we have

$$Q_v = \sum_{w \in N(v)} [c(v,w)]^2 \leqslant \sum_{w \in P^*} [c(v,w)]^2 = R_v, \tag{8}$$

because $N(v)$ is chosen as the set of $p-1$ nearest neighbors of v. Then

$$\begin{aligned}(2p-3)Q_v &\leqslant (2p-3)R_v \quad \text{(by Eq. (8))} &(9)\\ &\leqslant 2\left[\frac{2p-3}{p}\right]\mathcal{Q}_c(P^*) \quad \text{(by Eq. (7))} &(10)\\ &= \left[4-\frac{6}{p}\right]\mathcal{Q}_c(P^*). &(11)\end{aligned}$$

By construction, our algorithm will choose a node $\tilde{v}$ with minimum value $Q_{\tilde{v}} = \hat{\mathcal{Q}}_c(\tilde{v}, \bigcup N(\tilde{v}))$. Thus, $Q_{\tilde{v}} \leqslant Q_v$ and we get

$$\begin{aligned}\mathcal{Q}(N(\tilde{v}) \cup \{\tilde{v}\}) &\leqslant (2p-3)Q_{\tilde{v}} \quad \text{(by Lemma 6.2)}\\ &\leqslant (2p-3)Q_v\\ &\leqslant \left[4-\frac{6}{p}\right]\mathcal{Q}_c(P^*) \quad \text{(by Eq. (11))}.\end{aligned}$$

This completes the proof. □

We now address the running time of our algorithm. Let $n := |V|$ be the number of nodes in the graph G. Then the number m of edges is $m = \frac{1}{2}n(n-1) = \Theta(n^2)$. The main effort of the algorithm is in the loop in Step 2(b). For each node v we must determine the $(p-1)$ nearest neighbors with respect to the c distance function. This can be done in $O(n+p)$ time as follows. We first use a linear time selection algorithm (see e.g. [9]) to find the node w with the $(p-1)$st smallest distance from v. Then, by performing one linear pass over the $O(n)$ neighbors of v and comparing their distance to $c(v,w)$, we can then extract the $p-1$ nearest neighbors of v in $O(n+p)$ time.

Then, the algorithm computes the function $\hat{\mathcal{Q}}_c(v, N(v))$ for each of the n placements $P(v) = \{v\} \cup N(v)$. Evaluating $\hat{\mathcal{Q}}_c$ for one placement needs $O(p)$ time. Choosing the best placement $P(v)$ (with respect to $\hat{\mathcal{Q}}_c$) can then be accomplished in $O(n)$ time. This results in an overall time of $O(n^2+np) = O(n^2)$.

6.2. *Approximating* TI-(Min-Sum) *and* TI-(Min-Dia)

We are going to establish two lemmas which enable us to use measures that can be computed faster than $\mathcal{S}_c$ and $\mathcal{D}_c$.

Lemma 6.4. *Let $v \in V$ be an arbitrary node and let $N \subset V \setminus \{v\}$ be a set of $p-1$ nodes. Define*

$$\hat{\mathcal{S}}_c(v, \bigcup N) := \sum_{w \in N} c(v,w). \tag{12}$$

and

$$\hat{\mathcal{D}}_c(v, N) := \max_{w \in N} c(v,w). \tag{13}$$

Then $\mathcal{S}_c(\{v\} \bigcup N) \leqslant (p-1) \cdot \hat{\mathcal{S}}_c(v,N)$ and $\mathcal{D}_c(\{v\} \bigcup N) \leqslant 2\hat{\mathcal{D}}_c(v,N)$.

Proof. We first prove the bound with respect to $\mathscr{S}_c$. Let $w \in N$ be arbitrary. Then

$$\begin{aligned}
\sum_{u \in N \cup \{v\} \setminus \{w\}} c(w,u) &= c(w,v) + \sum_{u \in N \setminus \{w\}} c(w,u) \\
&\leqslant c(w,v) + \sum_{u \in N \setminus \{w\}} (c(w,v) + c(v,u)) \\
&= (p-1)c(w,v) + \sum_{u \in N \setminus \{w\}} c(v,u) \\
&= (p-2)c(v,w) + \sum_{u \in N} c(v,u) \\
&= (p-2)c(v,w) + \hat{\mathscr{S}}_c(v,N). \qquad (14)
\end{aligned}$$

Now using (14) we obtain

$$\begin{aligned}
\mathscr{S}_c(N \cup \{v\}) &= \frac{1}{2}\left(\sum_{u \in N} c(v,u) + \sum_{w \in N} \sum_{u \in N \cup \{v\} \setminus \{w\}} c(w,u)\right) \\
&= \frac{1}{2}\hat{\mathscr{S}}_c(v,N) + \frac{1}{2}\sum_{w \in N} \sum_{u \in N \cup \{v\} \setminus \{w\}} c(w,u) \\
&\overset{(14)}{\leqslant} \frac{1}{2}\hat{\mathscr{S}}_c(v,N) + \frac{1}{2}\sum_{w \in N} ((p-2)c(v,w) + \hat{\mathscr{S}}_c(v,N)) \\
&= \frac{1}{2}\hat{\mathscr{S}}_c(v,N) + \frac{p-2}{2}\hat{\mathscr{S}}_c(v,N) + \frac{p-1}{2}\hat{\mathscr{S}}_c(v,N) \\
&= (p-1)\hat{\mathscr{S}}_c(v,N).
\end{aligned}$$

This proves the first part of the lemma. For the second part, observe that $\{v\} \bigcup N$ will form a clique in the square of the bottleneck graph bottleneck$(G, c, \hat{\mathscr{D}}_c(v,N))$. The claim now follows immediately from Observation 2. □

Using the above lemma and following a proof outline similar to those of Lemmas 6.4 and 6.3, we obtain:

Theorem 6.5. *Denote by Heur-*(MIN-SUM) *and Heur-*(MIN-DIA) *the algorithms resulting by setting $\mathscr{M}_c := \hat{\mathscr{S}}_c$ and $\mathscr{M}_c := \hat{\mathscr{D}}_c$, respectively, in the generic procedure shown in Fig. 1.*

**Then Heur-*(MIN-SUM) *as applied to TI-*(MIN-SUM) *has a performance of $2 - 2/p$. Heur*(MIN-DIA) *is an approximation algorithm for TI-*(MIN-DIA) *for with a performance of 2. Both algorithms have a running time of* $O(n^2)$.

We will prove more general versions of Theorem 6.5 in Sections 7 and 8.

We have presented heuristics for unicriterion compact location problems under the three objectives, namely minimum diameter, minimum average distance and minimum variance. In [21], we present problem instances to show that a solution which is optimal with respect to one objective can be arbitrarily poor with respect to another objective, even when triangle inequality holds. These examples demonstrate that the objectives are in some sense orthogonal.

7. Heuristics for diameter constrained problems

In this section, we discuss our approximation algorithms for bicriteria compact location problems when the budget constraint is on the diameter. We begin with a brief discussion of the basic technique (Section 7.1). This is followed by our approximation algorithms for TI-(MIN-SUM, DIA) (Section 7.2) and TI-(MIN-DIA, DIA) (Section 7.3).

7.1. *Basic technique*

Hochbaum and Shmoys [19] introduced a general approach for approximating a number of bottleneck problems when the costs satisfy triangle inequality. We combine this approach with the local search heuristic for compact location problem developed in [22] to obtain our approximation algorithms for diameter-constrained compact location problems.

To illustrate our ideas, consider the problem TI-(MIN-SUM, DIA). Let B be the bound on the diameter of the placement with respect to the d-cost. It is clear that in the subgraph induced by an optimal placement, no edge has d-cost more than B. Thus, we can prune all the edges in the graph with d-costs more than B. Call the resulting graph G'. Since finding an optimal placement that minimizes the sum of distances with respect to c-cost is "hard" even in the modified graph, we resort to finding a near optimal placement. We find a near optimal placement in G' with respect to the c-costs. This placement becomes a clique in the square graph $(G')^2$. Because the edge weights satisfy triangle inequality, it follows that the placement does not violate the constraint by a factor of more than 2.

7.2. *Approximating* TI-(MIN-SUM, DIA)

We present an approximation algorithm for the problem TI-(MIN-SUM, DIA) that provides a performance guarantee of $(2-2/p, 2)$. The algorithm is shown in Fig. 2, where the cost measure $\mathcal{M}_c$ corresponds to $\hat{\mathcal{S}}_c$, defined in (12).

7.2.1. *Performance guarantee*

We will show that a placement returned by the algorithm will be almost feasible in the sense that it violates the diameter constraint by a factor of at most 2.

Lemma 7.1. *Any placement considered by the algorithm in Step 7 has d-diameter at most* $2B$.

Proof. Observe that any placement $P(v)$ which is considered by the algorithm consists of a vertex v and $p-1$ vertices $w_1, \ldots, w_{p-1}$ that are adjacent to v in the bottleneck

```
Procedure HEUR-FOR-DIA-CONSTRAINT
1  $G' :=$ bottleneck$(G, d, B)$
2  $V_{cand} := \{v \in G' : \deg(v, G') \geq p - 1\}$
3  if $V_{cand} = \emptyset$ then return "certificate of failure"
4  for each $v \in V_{cand}$ do
5      Let $N(v, G') = \{w_1, \dots, w_r\}$ with $c(v, w_1) \leq \cdots \leq c(v, w_r)$
6      Let $N(v) := \{w_1, \dots, w_{p-1}\}$
7      Let $P(v) := \{v\} \cup N(v)$
8  output the placement $P(v) = \{v\} \cup N(v)$ with the smallest value $\mathcal{M}_c(v, N(v))$
```

Fig. 2. Details of the heuristic TI-(MIN-SUM, DIA).

graph $G' :=$ bottleneck(G,d,B), which does not contain edges of d-weight greater than B. The placement will form a clique a clique in $(G')^2$. By Observation 2 all the edges between the vertices in this clique have d-weight at most $2B$. In other words, the d-diameter of $P(v)$ is at most $2B$. □

As an immediate consequence of Lemma 7.1 we obtain the following corollary.

Corollary 7.2. *If the algorithm returns a placement P (i.e., the algorithm does not report that that no feasible solution exists), then the d-diameter of P is at most $2B$.*

We are now ready to establish the performance of the algorithm in Fig. 2.

Lemma 7.3. *Algorithm HEUR-FOR-DIA-CONSTRAINT called with $\mathcal{M}_c := \hat{\mathcal{S}}_c$ has a performance of $(2 - 2/p, 2)$.*

Proof. By Corollary 7.2 we know that any solution output by the algorithm will violate the constraint on the d-diameter by a factor of at most 2. Suppose the algorithm reports the infeasibility of an instance in Step 3. Then, indeed, no feasible solution to the instance can exist, since any feasible solution with d-diameter at most B will form a clique in the bottleneck graph G'. In particular, there will be at least p nodes of degree p or greater in G' and V_{cand} cannot be empty.

It remains to show that for any instance of TI-(MIN-SUM, DIA) with a nonempty set of feasible solutions, the algorithm will find a good placement with respect to the objective $\mathcal{S}_c$.

Consider an optimal solution P^* such that $\mathcal{D}_d(P^*) \leqslant B$ and let $OPT := \mathcal{S}_c(P^*)$ be the optimal objective function value. By definition, this placement forms a clique of size p in $G' :=$ bottleneck(G,d,B). Hence, for any node $v \in P^*$ we have $|N(v, G')| \geqslant p$ and $P^* \subseteq V_{\text{cand}}$. For each node $v \in P^*$, let $R_v := \sum_{w \in P^*, w \neq v} c(v, w)$. Then we have $\mathcal{S}_c(P^*) = \frac{1}{2} \sum_{v \in P^*} R_v$. Now let $v \in P^*$ be so that R_v is a minimum among all nodes in P^*. Then clearly

$$OPT = \mathcal{S}_c(P^*) \geqslant \frac{p}{2} R_v. \tag{15}$$

As mentioned earlier, $v \in V_{\text{cand}}$. Consider the step of the algorithm 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

$$\hat{\mathscr{S}}_c(v,N(v)) = \sum_{w\in N(v)} c(v,w) \leqslant R_v \tag{16}$$

by definition of $N(v)$ as the set of nearest neighbors. By construction, our algorithm chooses a placement $P(\tilde{v}) = N(\tilde{v}) \cup \{\tilde{v}\}$ such that $\hat{\mathscr{S}}_c(\tilde{v},N(\tilde{v}))$ is minimized. Consequently, we have

$$\hat{\mathscr{S}}_c(\tilde{v},N(\tilde{v})) \leqslant \hat{\mathscr{S}}_c(v,N(v)) \overset{(16)}{\leqslant} R_v \overset{(15)}{\leqslant} \frac{2}{p} OPT. \tag{17}$$

By Lemma 6.4 we have

$$\begin{aligned} \mathscr{S}_c(P(\tilde{v})) &\leqslant (p-1)\hat{\mathscr{S}}_c(\tilde{v},N(\tilde{v})) \\ &\overset{(17)}{\leqslant} (2-2/p)\,OPT. \end{aligned}$$

This establishes the claimed performance guarantee. □

Let $n := |V|$ be the number of nodes in the graph G. Then the number m of edges is $m = \frac{1}{2}n(n-1) = \Theta(n^2)$. Thus, the bottleneck graph G' in Step 1 of the algorithm can be computed in time $O(m)$ by simply inspecting the weight $d(e)$ for each edge $e \in E$. The set of candidate nodes V_{cand} can then be determined in $O(n)$ time. Now, using the same analysis as in Section 6 for the generic algorithm, it now follows that the total time complexity of Algorithm HEUR-FOR-DIA-CONSTRAINT is $O(n^2+np) = O(n^2)$.

7.2.2. *Lower-bound example*

We provide an example of an instance I of TI-(MIN-SUM, DIA), where the $(2-2/p,2)$-performance of our heuristic is asymptotically tight. In this example, the node set V consists of $p+1$ subsets $B_0,\dots,B_p$. The set $B_0 = \{v_1,\dots,v_p\}$ forms an optimal placement of p nodes and the distances between the nodes in B_0 are $c(v_i,v_j) = 1$ and $d(v_i,v_j) = 1$. All other sets B_i $(i > 1)$ consist of $p-1$ nodes and we let $c(u,w) = 2-2\varepsilon$ and $d(u,w) = 2$, for all $u,w \in B_i$.

Let v_i be any node in the optimum placement B_0. We let $c(v_i,w) = 1-\varepsilon$ and $d(v_i,w) = 1$ for all $w \in B_i$, while we let $c(v_i,w') = 2-\varepsilon$ and $d(v_i,w) = 2$ for all $w' \in B_j$, $j \neq i$.

Finally, for each edge (u,v) such that u and v belong to distinct sets B_i, we let $c(u,v) = 3-2\varepsilon$ and $d(u,v) = 2$. It is easily verified that the distance functions c and d obey the triangle inequality. The instance I just described is illustrated in Fig. 3.

For the bound $B := 1$, the set B_0 forms an optimum placement with objective function value $\mathscr{S}_c(B_0) = \frac{1}{2}p(p-1)$ and diameter $\mathscr{D}_d(B_0) = 1$. In the first step, the algorithm calculates the bottleneck graph $G' = \text{bottleneck}(G,d,1)$. This graph now only contains the edges between the nodes in B_0 and the edges of the form (v_i,w) with $w \in B_i$. It is now easy to see that $V_{\text{cand}} = B_0$. For each node $v_i \in B_0$ the set of $p-1$ nearest neighbors consists of the set B_i and $\mathscr{D}_d(\{v_i\} \cup B_i) = 2 = 2B$. Moreover, $\mathscr{S}_c(\{v_i\} \cup B_i) = (p-1)(1-\varepsilon) + \frac{1}{2}((p-1)(p-2))(2-2\varepsilon)$ and $\lim_{\varepsilon\to 0} \mathscr{S}_c(\{v_i\} \cup B_i)/\mathscr{S}_c(B_0) = 2-2/p$.

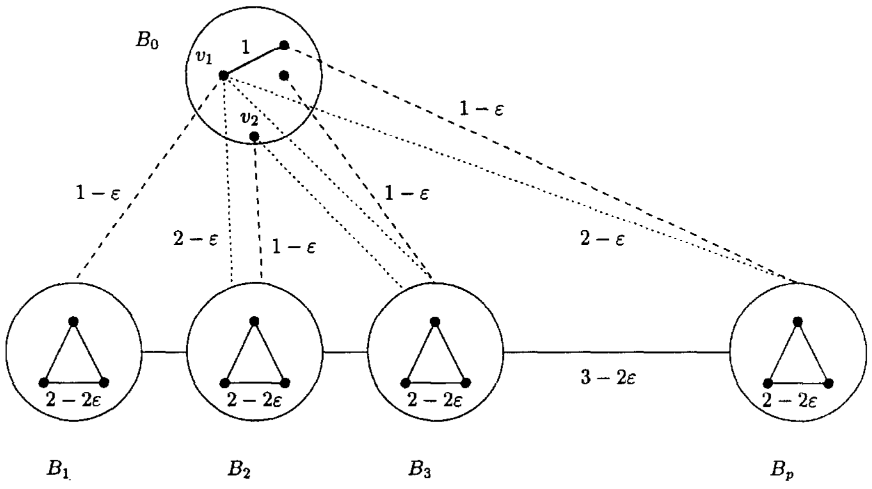

Fig. 3. Lower-bound example for the heuristic for TI-(MIN-SUM, DIA).

7.3. *Approximating* TI-(MIN-DIA, DIA)

Using the results from Section 6 in conjunction with the results in [26] we can devise an approximation algorithm with a performance guarantee (4, 4) for TI-(MIN-DIA, DIA). Here, we present an improved heuristic for this problem. This heuristic provides a performance guarantee of (2, 2). In view of the negative result discussed earlier, this is the best approximation we can expect to obtain in polynomial time.

The heuristic for approximating TI-(MIN-DIA, DIA) is the same as **HEUR-FOR-DIA-CONSTRAINT** given in Fig. 2, except that the measure $\mathcal{M}_c := \hat{\mathcal{D}}_c$, where $\hat{\mathcal{D}}_c$ is defined in Eq. (13).

Theorem 7.4. *Algorithm HEUR-FOR-DIA-CONSTRAINT called with* $\mathcal{M}_c := \hat{\mathcal{D}}_c$ *(where* $\hat{\mathcal{D}}_c$ *is defined in* (13)) *is an approximation algorithm for TI-*(MIN-SUM, DIA) *with a performance of* (2, 2).

Proof. Consider an optimal solution P^* such that $\mathcal{D}_d(P^*) \leqslant B$. It can be seen that $P^* \subseteq V_{\text{cand}}$, and thus V_{cand} is nonempty. Consequently, the heuristic will not output a "certificate of failure" in Step 3. Any placement considered by the algorithm will turn into a clique in $(G')^2$, where $G' = \text{bottleneck}(G, d, B)$. Thus, as in the proof of Lemma 7.3, we can conclude that any placement considered by the heuristic will violate the diameter constraint by a factor of at most 2.

Now consider an arbitrary node $v \in P^*$. Clearly, $v \in V_{\text{cand}}$ and consequently it will be considered in the loop. By definition of $N(v)$, for any $w \in N(v)$ we have $c(v, w) \leqslant OPT$, hence $\hat{\mathcal{D}}_c(v, N(v)) \leqslant OPT$. Thus, by the triangle inequality

$$\mathcal{D}_c(N(v) \cup \{v\}) \leqslant 2\hat{\mathcal{D}}_c(v, N(v)) \leqslant 2OPT. \qquad \square$$

The same running time analysis as in the case of TI-(MIN-SUM, DIA) shows that the algorithm runs in time $O(n^2)$.

Lower-bound example: The lower-bound example in Fig. 3 can be used to show that the approximation ratio indicated in Theorem 7.4 is tight for the heuristic. Again, an optimum placement consists of the nodes in B_0 for which both c- and d-costs are equal to 1. The placement returned by the heuristic is one of the sets $\{v_i\} \cup B_i$, which has c-cost equal to $2 - 2\varepsilon$ and d-cost equal to 2.

8. Heuristics for sum-constrained problems

In this section, we study approximation algorithms for bicriteria compact location problems where the objective is to minimize either the diameter $\mathscr{D}_c$ or the sum of the distances $\mathscr{S}_c$ subject to constraints of sum type.

First note that an (α, β)-approximation algorithm for TI-(MIN-DIA, SUM) can be constructed with the help of a (β, α)-approximation algorithm A for TI-(MIN-DIA, DIA) as follows: Using binary search, we find the minimum $M \in \{c(e) : e \in E\}$ such that A as applied to the instance I' of TI-(MIN-SUM, DIA), with the bound on the c-diameter set to M, finds a set of nodes where the total sum of the distances between the nodes is at most βB.

Thus, taking into account our results from Section 7.2 we obtain an approximation algorithm for TI-(MIN-SUM, SUM) with a performance of $(2, 2 - 2/p)$ and a running time $O(n^2 \log n)$.

As a lower bound example, consider again the instance depicted in Fig. 3 but interchange the c- and d-costs and set the bound B to be $\frac{1}{2}p(p-1)$. It is easy to see that by choosing $\varepsilon > 0$ small enough, the performance of the heuristic for this instance is arbitrarily close to $(2, 2 - 2/p)$.

We proceed to present a heuristic for TI-(MIN-SUM, SUM). The basic idea behind the approximation algorithm is to use a *parametric search* technique to reduce the problem to that of solving the minimum sum problem for a modified weight function. Then, by appropriate scaling and rounding techniques, this solution can be transformed back into a near optimal solution for the original bicriteria problem.

The presentation of our approximation algorithm is organized as follows. We first present our heuristic and show that it provides a performance guarantee of $((2 - 2/p)(1 + 1/\gamma), (2 - 2/p)(1 + \gamma))$ for any fixed $\gamma > 0$. We then show how to improve the running time of the heuristic using an elegant technique due to Megiddo [27].

8.1. A slow heuristic

The main procedure shown in Fig. 4 uses the test procedure from Fig. 5. Step 3 of the main procedure, that is, computing a $(2 - 2/p)$-approximation for the constructed instance of (MIN-SUM) (i.e. minimizing $\mathscr{S}_h$), can be done by using the unicriterion

```
Procedure HEUR-FOR-(MIN-SUM, SUM)
1  Use a binary search to find the smallest integer
       $M \in \mathcal{I} := \left[0, p^2 \max\{ c(e) : e \in E \}\right]$
   such that Sum-Test($T$)=Yes.
2  if the binary search terminates with the information that there is no such integer
   then output "There is no feasible solution" and stop
3  Let $P$ be placement generated by Sum-Test($T$)
4  if $\mathcal{S}_d(P) > (2-2/p)B$ then output "There is no feasible solution" else output $P$
```

Fig. 4. Main procedure for TI-(MIN-SUM, SUM).

```
Procedure Sum-Test($M$)
1  Let $\mu := \frac{M}{B}$.
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 (MIN-SUM) instance given by the graph
   $G$ the number $p$ and distances $h(e)$, $e \in E$
4  Let $P_M$ be a set of $p$ nodes with $\mathcal{S}_h(P_M) \le (2-2/p) \cdot \min_{P \subseteq V, |P|=p} \mathcal{S}_h(P)$.
5  if $\mathcal{S}_h(P_M) \le (2-2/p)(1+\gamma)M$ then output "Yes" else output "No".
```

Fig. 5. Test procedure used for TI-(MIN-SUM, SUM).

algorithm from Section 6 for (MIN-SUM) (or the algorithm from Section 7.2, where we set $c := h$, $d := 1$ and $B := 1$). We also note that γ is a fixed quantity that specifies the accuracy requirement.

For a given value of M, let OPT_{h_M} denote the sum of the distances of an optimal placement of p nodes with respect to the distance function $h_M(v,w) := c(v,w) + (M/B)d(v,w) = c(v,w) + \mu d(v,w)$; that is

$$OPT_{h_M} = \min_{\substack{P \subseteq V \\ |P|=p}} \mathcal{S}_{h_M}(P).$$

We have the following lemma.

Lemma 8.1. *The function $F(M) = OPT_{h_M}/M$ is monotonically nonincreasing on for $M > 0$.*

Proof. Let M_1 and M_2 be given numbers with $M_1 < M_2$. Let P_1 and P_2 denote optimal placements of p nodes under h_M when $M = M_1$ and $M = M_2$, 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 $F(M_i) = C_i/M_i + D_i/B$ for $i \in \{1,2\}$.

Consider the cost under h of the placement P_1 when $M = M_2$. By the definition of C_1 and D_1, it follows that the cost of P_1 is $C_1 + D_1 \cdot M_2/B$. Thus, the value of $F(M_2)$ is at most this cost divided by M_2 which is $C_1/M_2 + D_1/B$. This in turn is less than $C_1/M_1 + D_1/B$, since $M_1 < M_2$. But $C_1/M_1 + D_1/B$ is exactly $F(M_1)$, and hence $F(M_1) \geqslant F(M_2)$. $\square$

The next corollary proves that the binary search in Algorithm HEUR-FOR-MIN-SUM,SUM works correctly. Before we state and prove the corollary, observe that Step 4 of the algorithm ensures that, if the algorithm outputs a placement, this placement will violate the constraint on the sum of the d-distances by a factor of at most $2-2/p$. Thus, in the sequel, we can restrict ourselves to instances with a nonempty set of feasible solutions. Given such an instance, let $OPT = \mathcal{S}_c(P^*)$ denote the function value of an optimal placement P^* of p nodes. To simplify the analysis, we assume that OPT/γ 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.

Corollary 8.2. *The test procedure Sum-Test returns* "Yes" *for all $M > OPT/\gamma$. Thus, the binary search in Algorithm HEUR-FOR-*(MIN-SUM, SUM) *works correctly and either finds a value $M' \leqslant OPT/\gamma$ or provides the information that Sum-Test returns* "No" *for all values of M.*

Proof. We first show that the procedure will return "Yes" if called with $M^* = OPT/\gamma$. Notice that M^* is an integer by our assumption. We estimate the sum of the h_{M^*}-distances between the nodes in the optimal placement P^*. This sum is then $OPT + (M^*/B)B = OPT + M^* = (1+\gamma)M^*$. Thus, it follows that $OPT_{h_{M^*}} \leqslant (1+\gamma)B^*$ and the $(2-2/p)$-approximation P_T that is computed in Step 3 will satisfy $\mathcal{S}_h(P_{M^*}) \leqslant (2-2/p)OPT_{h_{M^*}} \leqslant (2-2/p)(1+\gamma)M^*$.

Thus, we observe that the procedure will return "Yes". Moreover, since $OPT_{h_{M^*}} \leqslant (1+\gamma)B^*$, it follows that $F(M^*) \leqslant (1+\gamma)$, where F is the function defined in Lemma 8.1. By the results of this lemma we then have $F(M) \leqslant (1+\gamma)$ for all $M \geqslant M^*$.

This is equivalent to saying that for all $M \geqslant M^*$ the corresponding optimal placement P^*_M minimizing $\mathcal{S}_{h_M}$ satisfies $\mathcal{S}_{h_M}(P^*_M) \leqslant (1+\gamma)M$. Hence, for all these values of M, the approximation P_M computed in Step 3 of the test procedure will satisfy $\mathcal{S}_{h_M}(P_M) \leqslant (2-2/p)(1+\gamma)M$. But this means that Algorithm 5 will return "Yes" for all $M \geqslant M^*$. □

Now we are ready to complete the proof of the performance of our approximation algorithm.

Lemma 8.3. *For any fixed $\gamma > 0$ Algorithm HEUR-FOR-*(MIN-SUM, SUM), *as applied to TI-*(MIN-SUM, SUM), *has a performance of* $((2-2/p)(1+1/\gamma), (2-2/p)(1+\gamma))$.

Proof. By Corollary 8.2, the binary search in Algorithm 4 will successfully end with a value of M satisfying $M \leqslant OPT/\gamma$. Let P_M be the corresponding placement that is returned by Sum-Test. Then we have

$$\mathcal{S}_c(P_M) \leqslant \mathcal{S}_h(P_M) \leqslant \left(2-\frac{2}{p}\right)\left(OPT + \frac{M}{B}B\right) \leqslant \left(2-\frac{2}{p}\right)\left(1+\frac{1}{\gamma}\right) OPT.$$

Moreover, we see that

$$\frac{M}{B}\mathscr{S}_d(P_M) \leqslant \mathscr{S}_h(P_M) \leqslant \left(2-\frac{2}{p}\right)(1+\gamma)M.$$

Multiplying the last chain of inequalities by B/M yields

$$\mathscr{S}_d(P_M) \leqslant (2-2/p)(1+\gamma)B.$$

This completes the proof. □

In the above version of the heuristic for TI-MIN-(SUM,SUM) the test procedure **Sum-Test** is called $O(\log(p^2 c_{\max}/\gamma))$ times during the binary search, where $c_{\max} := \max\{c(e) : e \in E\}$. For the rest of this section, let $T_{\text{test}}(n) = O(n^2)$ be the time required for a single call to **Sum-Test**. Then, the total time for the algorithm would be $O(\log p^2 c_{\max}/\gamma \cdot T_{\text{test}}(n))$. We will now show how to improve this running time.

8.2. *Outline of a faster heuristic*

If **Sum-Test** is called with some parameter K, it first computes the compound weights h_K. Then, in Step 3 it computes a $2-2/p$-approximation for the (unicriterion) TI-(MIN-SUM) instance with edge weights given by h_K. This is done with the help of our algorithm from Section 6. Recall that this algorithm generates n placements $P(v)$, one for each vertex and its nearest neighbors with respect to h_K. It then outputs the placement with the best objective function value $\mathscr{S}_{h_K}$.

As before, let $K^* \in \mathscr{I} := [0, p^2 \max\{c(e) : e \in E\}]$ be the minimum value such that **Sum-Test**(K^*) =“**Yes**”. Assume that we already know the ordering of the edges in the graph with respect to h_{K^*}. Then, for each vertex v we can find the $p-1$ nearest neighbors with respect to h_{K^*}. We do *not* need to know the weights; the ordering suffices. Thus, given the knowledge about the ordering with respect to h_{K^*}, we can find a set of n placements containing the placement output by our slow heuristic for TI-(MIN-SUM, SUM) above.

This is what our faster algorithm will do in the first phase. It will find the ordering with respect to h_{K^*} and narrow our search to n placements. This is done *without* actually knowing K^*. In the second phase, we will determine the placement among these n placements which our slow heuristic would output.

Let $m = \frac{1}{2}n(n-1)$ be the number of edges in the graph $G = (V, E)$. Basically we wish to sort the set $S := \{h_{K^*}(e_1), \ldots, h_{K^*}(e_m)\}$ where K^* is not known. However, for any K we can decide whether $K^* \leqslant K$ or $K^* > K$ by one call to our test procedure **Sum-Test**: If **Sum-Test**(K) =“**Yes**”, then we know that $K^* \leqslant K$. Otherwise, we can conclude that $K^* > K$.

Recall that the h_K-weight of an edge e is given by $c(e) + Md(e)/B$. Thus, for each edge e, the compound weight $h_K(e)$, viewed as a function of K, is linear. Given two edges e and e', their ordering with respect to the compound weight h_K changes at most once when K varies, namely at the point where the two linear functions intersect. Clearly, given two edges e and e' this value of K can be computed in constant time.

Using Megiddo's technique [27] we can accomplish the sorting of the set S efficiently by using only $O(\log^2 m)$ calls to Algorithm Sum-Test plus an overhead of $O(m \log m)$ elementary operations. Since each call to Sum-Test takes $T_{\text{test}}(n) = O(n^2)$ time and $m = O(n^2)$, we have the following lemma.

Lemma 8.4. *The improved heuristic computes the ordering of the edges with respect to h_{K^*} in time $O(n^2 \log^2 n)$.*

Let P_{slow} be the placement generated by our slow heuristic for TI-(Min-Sum, Sum). We have already argued that, given the ordering of the edges with respect to h_{K^*} we can find a set $\mathscr{P} = \{P(v_1), \ldots, P(v_n)\}$ of placements such that $P_{\text{slow}} \in \mathscr{P}$. By the construction of our slow algorithm, it follows that P_{slow} is a placement $P(v_j)$ in $\mathscr{P}$ with minimum $\hat{\mathscr{S}}_{h_{K^*}}(v_j, N(v_j))$, where $N(v_j)$ is the set of $p-1$ nearest neighbors to v_j with respect to h_{K^*}. We now show how to find P_{slow} in the set $\mathscr{P}$ efficiently.

For each placement $P(v_i) \in \mathscr{P}$ denote by $C_i := \hat{\mathscr{S}}_c(P(v_i))$ and $D_i := \hat{\mathscr{S}}_d(P(v_i))$ the simplified sum of the c-weights and d-weights respectively (see Eq. (12)). Clearly, all the C_i and D_i can be found in an overall time of $O(np)$.

Observe that $\hat{\mathscr{S}}_{h_K}(P(v_i)) = C_i + KD_i/B$. Our task now becomes that of finding a placement having minimum value when $K = K^*$. Again, we can view $\hat{\mathscr{S}}_{h_K}(P(v_i))$ as a linear function of K. The ordering of two placements changes, when the two functions intersect. Using the same technique as finding the ordering of the *edges* with respect to h_{K^*}, we can find the ordering of the *placements* in $\mathscr{P}$ with respect to $\hat{\mathscr{S}}_{h_{K^*}}$ in time $O(m \log m + T_{\text{test}}(n) \log^2 m)$.

This enables us to find the same placement as the slow heuristic in an overall time of $O(m \log m + T_{\text{test}}(n) \log^2 m)$. Since, again, $T_{\text{test}}(n) = O(n^2)$ and $m = O(n^2)$, we obtain the following theorem.

Theorem 8.5. *For any fixed $\gamma > 0$, the improved heuristic for* TI-(Min-Sum, Sum) *has a performance of* $((2 - 2/p)(1 + 1/\gamma), (2 - 2/p)(1 + \gamma))$ *and a running time of* $O(n^2 \log^2 n)$.

9. Extension to the node-weighted case

We now briefly discuss how to extend our algorithms to apply to the case when we additionally have weights for each node in G, and the minimization objective is a function of both edge weights and node weights. For the sake of brevity, we will illustrate our ideas for such an extension by considering one specific problem, namely TI-(Min-Sum, Dia). The approximation algorithms for the other problems can be extended in a similar fashion.

The input consists again of a complete undirected graph $G = (V, E)$ with edge-weight functions c, d and weights $\omega(v)$, for each $v \in V$. The goal is to find a placement P, with $|P| = p$, such that the objective function

$$\mathcal{S}_c^{wt}(P) = \sum_{\substack{e=(v,w)\\ v,w \in P}} c(v,w) + \sum_{v \in P} \omega(v)$$

is minimized subject to the same constraint as for the (Min-Sum, Dia) problem, namely $\mathcal{D}_d(P) \leqslant B$. We denote this extension of the (Min-Sum, Dia) problem by (Min-Sum, Dia)^{wt}.

To obtain an approximate solution for TI-(Min-Sum, Dia)^{wt}, we transform a given instance I of TI-(Min-Sum, Dia)^{wt} into an instance I' of TI-(Min-Sum, Dia) as follows. Let $G'(V', E')$ denote the graph which is part of the instance I'. The nodes of $G'(V', E')$ are in one-to-one correspondence with the nodes of G. The d'-cost on each edge in G' is the same as the d-cost of the corresponding edge in G. The c'-cost for an edge (v', w') in G' is defined as follows:

$$c'(v', w') := c(v, w) + \frac{1}{2p}(\omega(v) + \omega(w)).$$

It is easy to check that the triangle inequality is satisfied for c'. Next consider a placement P' of p nodes in I'. The set P' can also be interpreted as a placement P for instance I. By a straightforward calculation, it can be seen that for any placement P of p nodes,

$$\sum_{\substack{e'=(v',w')\\ v',w' \in P'}} c'(v', w') = \sum_{v \in P} \omega(v) + \sum_{\substack{e=(v,w)\\ v,w \in P}} c(v, w).$$

Therefore, the above transformation and our heuristic for TI-(Min-Sum, Dia) together provide an approximation algorithm with a performance of $(2 - 2/p, 2)$ for TI-(Min-Sum, Dia)^{wt}.

10. Concluding remarks

We introduced and studied the complexity and approximability of several natural bicriteria compact location problems. Our results demonstrate that when distance functions obey the triangle inequality, the problems are provably easier to approximate.

Tables 1–3 summarize our results. Table 1 shows the hardness results for the various unicriterion problems. Table 2 gives the corresponding approximation results. Table 3 shows our results for bicriteria compact location problems. 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 objective i.

Table 1
Complexity results for compact location problems

Problem	(Min-Dia)	(Min-Sum)	(Min-Var)
General	NP-hard	NP-hard	NP-hard
Triangle inequality	NP-hard	NP-hard	NP-hard
1D version	Efficiently solvable [1]	Efficiently solvable	Efficiently solvable [1]
2D version	Efficiently solvable [1]	Open	Efficiently solvable [1]

Note: The 1D version of (Min-Sum) can be solved efficiently because every optimal solution consists of p contiguous points.

Table 2
Approximability results for NP-hard compact location problems

	(Min-Dia)		(Min-Sum)		(Min-Var)	
Problem	UB	LB	UB	LB	UB	LB
General	–	NGR	–	NGR	–	NGR
Triangle Inequality	2	2	$2 - 1/p$	Open	$4 - 6/p$	Open

Notes: UB denotes the best-known upper bound on the performance guarantee. LB denotes the lower bound on the performance guarantee; that is, the intrinsic limit assuming P$\neq$NP. NGR is an abbreviation for "no guaranteed ratio".

Table 3
Performance guarantee results for bicriteria compact location problems where both the edge weights obey the triangle inequality

$\rightarrow$ Objective $\downarrow$ Budget	Diameter	Sum
Diameter	Approximable within $(2,2)$. Not approximable within $(2-\varepsilon,2)$ or $(2,2-\varepsilon)$.	Approximable within $(2-2/p,2)$. Not approximable within $(\alpha,2-\varepsilon)$.
Sum	Approximable within $(2,2-2/p)$. Not approximable within $(2-\varepsilon,\alpha)$.	Approximable within $((1+\gamma)(2-2/p),(1+1/\gamma)(2-2/p))$.

$\gamma > 0$ is a fixed accuracy parameter. The non-approximability results stated assume that P$\neq$NP. As discussed in Section 9, these results can be extended to handle node weights.

Acknowledgements

We thank Dr. Gyan Bhanot (Thinking Machines Corporation) for bringing to our attention the processor allocation problem mentioned in the introduction. We thank Professor R. Ravi (Carnegie-Mellon University) and Dr. Ravi Sundaram (Delta Trading, Inc.) for valuable discussions which lead to the research described in this paper. We also thank the referee and Professor Ming Li for many helpful suggestions which improved the quality of the paper.

References

[1] A. Aggarwal, H. Imai, N. Katoh and S. Suri, Finding k points with minimum diameter and related problems, *J. Algorithms* **12** (1991) 38–56.

[2] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, *Network Flows: Theory, Algorithms and Applications* (Prentice-Hall, Englewood Cliffs, NJ, 1993).

[3] H.C. Andrews, *Introduction to Mathematical Techniques in Pattern Recognition* (Wiley-Interscience, New York, NY, 1972).

[4] B. Awerbuch, Y. Azar, A. Blum and S. Vempala, Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen, in: *Proc. 27th Ann. ACM Symp. on the Theory of Computing* (STOC '95) (1995) 277–376.

[5] J. Bar-Ilan, G. Kortsarz and D. Peleg, How to allocate network centers, *J. Algorithms* **15** (1993) 385–415.

[6] M. Bellare and M. Sudan, Improved nonapproximability results, in: *Proc. 26th Ann. ACM Symp. on the Theory of Computing* (STOC '94) (1994) 184–193.

[7] M. Bern and D. Eppstein, Approximation algorithms for geometric problems, unpublished manuscript, January 1995.

[8] A. Blum, P. Chalasani and S. Vempala, constant-factor approximation for the k-MST problem in the plane, in: *Proc. 27th Ann. ACM Symp. on the Theory of Computing* (STOC '95) (1995) 294–302.

[9] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms* (MIT Press, New York, 1990).

[10] A. Datta, H.P. Lenhof, C. Schwarz and M. Smid, Static and dynamic algorithms for k-point clustering problems, in: *Algorithms and Data Structures, Third Workshop*, Lecture notes in Computer Science, Vol. 709 (Springer, Berlin, 1993) 265–276.

[11] M.E. Dyer and A.M. Frieze, A simple heuristic for the p-center problem, *Oper. Res. Lett.* **3** (1985) 285–288.

[12] J. Erickson and D. Eppstein, Iterated nearest neighbors and finding minimal polytopes, *Discrete Comput. Geometry* (11) (1994) 321–350.

[13] E. Erkut and S. Neuman, Analytical models for locating undesirable facilities, *European J. Oper. Res.* **40** (1989) 275–291.

[14] T. Feder and D. Greene, Optimal algorithms for approximate clustering, in: *ACM Symp. on Theory of Computing* (STOC '88) (1988) 434–444.

[15] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of* NP-*completeness* (W.H. Freeman, San Francisco, CA, 1979).

[16] N. Garg and D.S. Hochbaum, An $O(\log k)$ approximation algorithm for the k minimum spanning tree problem in the plane, in: *Proc. 26th ACM Symp. on Theory of Computing* (STOC '94) Montreal (1994) 432–438.

[17] T.F. Gonzalez, Clustering to minimize the maximum intercluster distance, *Theoret. Comput. Sci.* **38** (1985) 293–306.

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

[19] D.S. Hochbaum and D.B. Shmoys, A unified approach to approximation algorithms for bottleneck problems, *J. ACM* **33** (1986) 533–550.

[20] G. Kortsarz and D. Peleg, On choosing a dense subgraph, in: *Proc. 34th Ann. IEEE Symp. on the Foundations of Computer Science* (FOCS '93) (1993) 692–703.

[21] S.O. Krumke, M.V. Marathe, H. Noltemeier, V. Radhakrishnan, S.S. Ravi and D.J. Rosenkrantz, Compact location problems, Tech. Report LA-UR 96-0210, October 1996.

[22] S.O. Krumke, H. Noltemeier, S.S. Ravi and M.V. Marathe, Compact location problems with budget and communication constraints, in: *Proc. 1st Internat. Conf. on Computing and Combinatorics*, X'ian, China, Lectore Notes in Computer Science, Vol. 959 (Springer, Berlin, 1995) 510–519.

[23] S.O. Krumke, H. Noltemeier, S.S. Ravi and M.V. Marathe, Bicriteria problems in location theory, in: *21st Workshop in Graph Theoretic Concepts in Computer Science* (WG '95) Aachen, Germany, June 1995.

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

[25] J.H. Lin and J.S. Vitter, ε-approximations with minimum packing constraint violation, in: *ACM Symp. on Theory of Computing* (STOC '92) (1992) 771–781.

[26] M.V. Marathe, R. Ravi, R. Sundaram, S.S. Ravi, D.J. Rosenkrantz and H.B. Hunt III, Bicriteria network design problems, in: *Proc. 22nd Internat. Coll. on Automata Languages and Programming* (ICALP '95) Lecture Nores in Computer Science, Vol. 944 (Springer, Belin, 1995) 487–498.

[27] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM* **30** (1983) 852–865.

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

[29] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction* (Springer, New York, NY, 1985).

[30] V. Radhakrishnan, S.O. Krumke, M.V. Marathe, D.J. Rosenkrantz and S.S. Ravi, Compact location problems, in: *Proc. 13th Conf. on the Foundations of Software Technology and Theoretical Computer Science* (FST&TCS '93) Lecture Notes in Theoretical Computer Science, Vol. 761 (Springer, Berlin, 1993) 238–247.

[31] S.S. Ravi, D.J. Rosenkrantz and G.K. Tayi, Heuristic and special case algorithms for dispersion problems, *Oper. Res.* **42** (1994) 299–310.

[32] R. Ravi, R. Sundaram, M.V. Marathe, D.J. Rosenkrantz and S.S. Ravi, Spanning trees short or small, *SIAM J. Discrete Math.* **9** (1996) 178–200.

[33] D.J. Rosenkrantz, R.E. Stearns and P.M. Lewis II, An analysis of several heuristics for the traveling salesman problem, *SIAM J. Comput.* **6** (1977) 563–581.

[34] A. Tamir, Obnoxious facility location on graphs, *SIAM J. Discrete Math.* **4** (1991) 550–567.

[35] A.A. Zelikowsky and D.D. Lozevanu, Minimal and bounded trees, *Acad. Romano-Americane*, Kishinev (1993) 25–26.