DISPERSION WITH CONNECTIVITY IN WIRELESS MESH NETWORKS

BIROL YÜCEOĞLU *
R & D Center, Information Technologies Department, Migros T.A.Ş.,
34758, İstanbul, Turkey

Ş. İLKER BIRBİL AND ÖZGÜR GÜRBUZ
Faculty of Engineering and Natural Sciences, Sabancı University,
34956, İstanbul, Turkey

(Communicated by Chao Xu)

Abstract. We study a multi-objective access point dispersion problem, where the conflicting objectives of maximizing the distance and maximizing the connectivity between the agents are considered with explicit coverage (or Quality of Service) constraints. We model the problem first as a multi-objective model, and then, we consider the constrained single objective alternatives, which we propose to solve using three approaches: The first approach is an optimal tree search algorithm, where bounds are used to prune the search tree. The second approach is a beam search heuristic, which is also used to provide lower bound for the first approach. The third approach is a straightforward integer programming approach. We present an illustrative application of our solution approaches in a real wireless mesh network deployment problem.

1. Introduction. In this paper, we study a multi-objective access point dispersion problem, which can be used to model various applications. Here, the dispersion of the agents is not sufficient as the sole performance measure and there is a need for some interaction between the agents in form of mutual visibility or connectivity. In particular, we consider the deployment of a Wireless Mesh Network (WMN) with Quality of Service (QoS) constraints, as an example multi-objective dispersion problem that takes into account both the dispersion and the connectivity of the access points.

Wireless Mesh Networking has emerged as a key wireless technology for numerous applications, such as: broadband home networking, community and neighborhood networks, enterprise networking, building automation [1, 3]. WMNs are designed with self-organization and self-configuration capabilities so that wireless mesh nodes can automatically establish links and dynamically maintain connectivity and also adapt to changes due to node/link failures or introduction of new nodes to the network. The mesh nodes are network access points, which can act both as hosts that transmit their own data to some destination nodes and as routers that forward the data received from their neighbor nodes. In addition, mesh nodes are designed

2010 Mathematics Subject Classification. Primary: 90B80; Secondary: 90B18, 90C90.
Key words and phrases. Wireless mesh networks, wireless network deployment, dispersion problem, tree search, goal programming.
* Corresponding author: byuceoglu@migros.com.tr.
with functions to discover their neighbors and perform optimal routing across the network, while providing similar QoS as that of wired networks. QoS provisioning has been a critical requirement for wireless networks since the third generation cellular systems [11].

Successful operation of WMNs relies significantly on how well the network is connected, how reliable the links are, and how efficiently the resources are utilized. All of these can be addressed by how well the network is initially deployed. A good WMN deployment may bring great benefits, such as; better network management, better performance and cost savings. In addition, covering an area with the least number of mesh nodes minimizes the overall network cost and the knowledge of an optimal mesh node placement can offer improved performance as well as guidelines for extensions to the network especially when it is not possible to re-deploy the entire network. The deployment of a WMN involves placing a number of mesh nodes; i.e., access points, in a given area, such that the minimum distance between any two access points and the total connectivity between the access points are maximized at the same time. In this problem, the dispersion of the access points is desirable as it lowers the equipment cost for providing an acceptable coverage for a given region, while the connectivity of the mesh structure must ensure that each pair of nodes is connected with maximum possible data transmission rate (or throughput). Additionally, due to heterogeneous traffic demands, it may be necessary to provide stronger coverage for certain service points, where the connected users expect to receive higher access rates as their QoS requirement.

Maximizing both objective functions at the same time is the main challenge for designing current networks, and it will become even more critical for the cloud radio access networks (C-RAN) of fifth generation (5G) systems, where all the intelligence and antenna, signal, data and network processing will be concentrated at the cloud base stations [27]. In the future, the terminals will get simpler with only sensors and wireless transceivers, especially for Internet of Things (IoT) applications [32]. Hence, the main cost of the wireless systems will be due to the capital expenditure cost for building the infrastructure, which can be minimized by maximizing the dispersion. Maximizing dispersion also ensures that interference between nodes is controlled. This way, the quality of each link can be specified in terms of Signal-to-Noise Ratio (SNR), which is used to calculate the other metric to be maximized, the connectivity or the throughput as studied in this paper. In applications, such as surveillance, where the captured and relayed data over the WMN represent video or high quality images, connectivity maximization is critical.

The multi-objective formulation presented in this paper allows combining both locating the multiple access points and maximizing the multiple objectives, namely dispersion and connectivity. Our formulation also handles additional coverage constraints for providing desired QoS at certain service points for a WMN. Furthermore, we determine the locations of all access points at once and in case of an extension for an existing network, the location problem can be solved by considering the previously located access points as fixed. Specifically, in our mathematical formulation, the access points are considered for candidate locations, such that they are dispersed to serve a given site while the connectivity among them is maximized. We propose to obtain the Pareto optimal solutions for this multi-objective problem, after which we model two single-objective problems with constraints on distance and connectivity. Then, we present two approaches for solving the resulting problems: The first approach is a tree search with bounding (TSWB) that guarantees optimality
at the expense of computational complexity. As an alternate approach, we propose
a beam search heuristic. Our computational study shows that the first approach is
 guaranteed to find the optimal solution at the expense of high computation time for
very large-scale problems. Our second solution approach, the beam search heuristic,
provides a fast solution, that can be used as an initial feasible solution within the
tree search approach to prune a significant portion of the search tree. The applica-
tion in a real indoor WMN deployment scenario has also shown that this heuristic
can provide a good solution in a very short computation time even in the presence
of QoS constraints. We also compare our results with those obtained by solving an
exact integer programming formulation. Our empirical study shows that TSWB is
significantly faster than the integer programming approach as well.

The rest of the paper is organized as follows: In Section 2, we provide a review
of the related literature. We present our problem in Section 3. The proposed
solution approaches, the TSWB and the beam search algorithms, are introduced in
Section 4. Section 5 is reserved for the illustrative WMN deployment application
along with computational results. In Section 6, we summarize our conclusions and
discuss future research directions.

2. Related work. When viewed as a facility location problem, our problem and
the pure dispersion problems have some common points with the maximum clique
problems; a family of NP-hard problems from graph theory [30]. A clique in a graph
is a subgraph, where all vertices are pairwise adjacent. The maximum clique prob-
lem tries to find the clique with the maximum number of vertices. The maximum
weighted clique problem, on the other hand, aims at finding a clique of a fixed num-
ber of cardinality by maximizing a given weight function, which is usually the sum
of the weights on the edges. In the dispersion context, this corresponds to dispers-
ing the vertices (similar to access points) by maximizing the sum of the distances
among them [10], or solving the maxsum dispersion problem. Macambira intro-
duces a tabu-search heuristic for the maximum weighted subgraph problem [17].
Wood presents a branch-and-bound algorithm for the maximum clique problem
[30]. Similarly, maxmin dispersion problem, where the minimum distance between
the facilities is maximized, can also be seen as a maximum clique problem. A
heuristic for the maxmin dispersion problem based on this idea can be found in [7].
The main difference between these works and our work is that in our problem each
dg edge between two vertices has a pair of weights, which corresponds to the measure
of connectivity and the distance between the vertices. Additionally, in our work,
we consider QoS constraints in order to provide a better coverage of the area.

Anti-covering location problem is another type of dispersion problem, which is
about opening the maximum number of facilities by ensuring a predefined distance
among the facilities. Murray and Church study the anti-covering location prob-
lem via Lagrangian relaxation [22]. Brimkov et al. focus on a geometrical agent
placement problem [4]. This problem is about placing the agents at the intersection
points of line segments (streets of a city, corridors in a building, and so on), and the
objective is to minimize the number of agents required to secure each line segment.
The dispersion literature also involves several work on multi-objective problems,
where the objectives are cost and political opposition to the facility construction
[9, 25]. Several studies focus on similar problems in the form of the semi-obnoxious
facility location problems [25, 20, 21]. For instance, Skriver and Andersen work on
locating a new airport in Denmark [28]. An airport is a semi-obnoxious facility as
it should be close to the cities in order to minimize the transportation costs. At the
same time it should not be too close to the cities because of the potential noise and pollution. The problem is studied for both planar and network cases. In the planar case, the authors divide the plane into smaller squares and by using upper and lower bounds on the conflicting objective functions, they eliminate some of these squares to reduce the search space. In the network case, cities are considered as the nodes of the network and the edges of the network represent the roads. The airport should be constructed on an edge since the disturbing effect would be higher in the cities; meanwhile cities have a certain demand depending on their population. Similar to the method used in the planar case, the authors divide the edges into smaller edges and determine the set of Pareto optimal locations by using lower and upper bounds. This problem is similar to ours except that there is only one facility (the airport) to be located and both objective functions depend on the distances and the population density. Melachrinoudis and Xanthopulos focus on a single facility location problem on a plane, which is also solved by decomposition of the plane to calculate the efficient solutions [20]. In an earlier study, Melachrinoudis describes another single facility location problem, where a new facility that is interacting with the existing ones should be located [21]. The author decomposes a non-convex bi-criteria problem into linear bi-criteria problems to find all the efficient regions in the plane. Another single facility semi-obnoxious location problem is studied in [12]. Having the same objective (minimizing transportation cost and minimizing the obnoxious effects of the facility to the population centers), the problem differs in the metric used to model the propagation of the obnoxious effects of the facility. The authors use an elliptic distance metric to simulate the impact of wind on the distribution of pollution. By using the mathematical properties of the objectives, the authors determine an efficient set on the graph, where the facility can be located. Some applications of semi-obnoxious facility location problems with different objective functions can be found in [23, 24]. Both of these works consider location of a single facility, while in our problem multiple facilities are to be located simultaneously. For a detailed overview of the location problems, we refer the reader to [8].

The cover problem has been studied for deployment of mostly wireless sensor networks. A wireless sensor network management problem is studied in [29], where the sensor nodes in an area are activated in a rolling time horizon. The authors study two multi-objective problems that are similar to our problem, but they also include a planning horizon. The problems are based on the optimization of the performance subject to a cost constraint and the optimization of the cost subject to a performance constraint. The authors give a constrained dynamic programming formulation with a bounding scheme. The bound comes from solving an unconstrained dynamic program obtained by using Lagrangian duality. Furthermore, the authors use a tree search with pruning approach similar to ours, which selects a number of locations with best objective function values as candidate locations, and then these solutions are used as the candidate nodes for the next stage of the search. Khedr and Osamy study a minimum connected cover problem in wireless sensor networks in [15]. In a wireless sensor network, low-cost sensors are placed in a region so that the entire region is covered. In the paper, the authors focus on reducing redundant coverage of sensors to reduce the energy consumption, and providing a mobility-assisted coverage to improve the reliability of the network in case there is a failure. In [26], Rebai et al. study a grid coverage problem with connectivity constraints in a wireless sensor network. They develop an integer linear programming formulation and several heuristics in order to solve the problem. As they are dealing with
sensors, their network contains a sink node that gathers all the information. Their objective is to minimize the number of sensors required to cover an area. However, they do not consider obstacles in their problem, unlike our work. Our work combines dispersion, connectivity and coverage in the WMN deployment problem. We consider QoS constraints with a realistic modeling of all losses, including obstacles such as walls, windows etc. We present novel solution methods to this problem and test the performance of our solutions on a real-life indoor deployment problem.

3. Problem definition. In the access point dispersion problem, we aim to maximize dispersion and connectivity between access points. Maximizing dispersion reduces the number of access points and the capital expenditure cost of the network infrastructure. In addition to that, it limits the interference between mesh nodes. In mesh networks, the nodes farther than two hops are placed so that they do not hear each other. This way, the quality of each link can be specified in terms of Signal-to-Noise Ratio (SNR), which is converted to the connectivity (i.e., throughput), the other metric that we want to maximize in this work.

In the access point dispersion problem, we are given a set of \( m \) candidate locations. The candidate points are fixed locations that are considered for deployment of \( n \) access points such that \( p \) service points are covered with a given QoS. The service points correspond to locations such as offices and classrooms that need to be connected to a wireless network. In addition to covering the service points, our aim is locating the access points on the candidate points such that they are dispersed to cover the site as well as the connectivity among the access points is maximized. To disperse the access points around a site, we maximize the minimum distance between any two access points; a common technique in the dispersion literature that is used especially in the obnoxious facility location problem; see [5] for a survey. The evaluation of the connectivity between the access points is usually problem dependent. We give the general mathematical model with an abstract connectivity function and elaborate on the connectivity function that we choose (which corresponds to connection quality) in Section 5.

We work with three matrices: an \( m \times m \) matrix, \( D = [D_{ij}] \); an \( m \times m \) matrix, \( C = [C_{ij}] \); an \( m \times p \) matrix, \( E = [E_{ik}] \). The matrix \( D \) denotes the distance between the candidate locations, the matrix \( C \) denotes the connectivity between the candidate locations and the matrix \( E \) denotes the connectivity between the candidate locations and the service points. In the rest of this paper, we assume that \( D \) and \( C \) are symmetric. The binary vector \( x \in \{0,1\}^m \) denotes the location of the access point; i.e., if \( x_i = 1 \) then one access point is placed in location \( i \); otherwise, \( x_i = 0 \).

The first term of the objective function is the minimum distance between any two access points

\[
d(x) := \min\{D_{ij}x_ix_j \mid i = 1, \ldots, m - 1; \ j = i + 1, \ldots, m; \ x_i = x_j = 1\}.\]

The second component of the objective function is the total connectivity value between the access points given by

\[
c(x) := \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} C_{ij}x_ix_j.\]

For every service point \( k \), we guarantee coverage with a given QoS threshold. For this purpose, we use the connectivity values between the candidate points and
service points and find the access point that serves the service point with highest connectivity given by

$$e_k(x) := \max\{E_{ik}x_i \mid i = 1, \ldots, m\}.$$  

$e_k(x)$ denotes the maximum connectivity between the located access points and the specific service point $k$. We will require this value to be greater than a QoS threshold value denoted by $e_{min}$ for each service point. Note that, the service points can be located anywhere and they do not have to be restricted to access point locations.

The overall mathematical programming model for the multi-objective (vector-valued) problem then becomes

$$\begin{align*}
\text{maximise} & \quad [d(x), c(x)]^T, \\
\text{subject to} & \quad e_k(x) \geq e_{min}, \quad \forall k \in \{1, \ldots, p\}, \\
& \quad \sum_{i=1}^{m} x_i = n, \\
& \quad x_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, m\}. \\
\end{align*}$$

(1)

In this model, we maximize the (multi-)objective function, by locating $n$ agents in the area and by satisfying QoS constraints for the service points. In our application, the connectivity values correspond to the probability of correct transmission and depend on several parameters including communication technique, link quality, and so on (see Section 5).

Clearly, the multi-objective model 1 is apt to the concept of Pareto optimality [14]. Two feasible solutions $\hat{x}$ and $\bar{x}$ are Pareto optimal, if one does not dominate the other. As our objective is maximizing the two components of the objective function, we say that $\hat{x}$ dominates $\bar{x}$, if $d(\hat{x}) \geq d(\bar{x})$ and $c(\hat{x}) \geq c(\bar{x})$. When non-dominated solutions are found in a bi-objective model, it is common to place the solutions on a two-dimensional curve, called the Pareto curve. On this curve, each axis gives the value of one of the objective functions (see Figure 3 for an example). However, selecting a configuration using the Pareto curve requires human interaction. Therefore, it is also common to move one of the objectives into the set of constraints and ensure that the value obtained with that objective function is bounded by a user-defined value. This is the well-known goal programming approach. Following the same steps, we first propose to focus directly on problem 1 to obtain all the Pareto optimal solutions. Then, we model two single-objective problems with bound constraints as in the goal programming approach. Formally, we obtain

$$\begin{align*}
\text{maximise} & \quad d(x), \\
\text{subject to} & \quad e_k(x) \geq e_{min}, \quad \forall k \in \{1, \ldots, p\}, \\
& \quad c(x) \geq c_{min}, \\
& \quad \sum_{i=1}^{m} x_i = n, \\
& \quad x_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, m\}. \\
\end{align*}$$

(2)
maximize \( c(x) \),
subject to \( e_k(x) \geq e_{\text{min}}, \quad \forall k \in \{1, \ldots, p\} \),
\( d(x) \geq d_{\text{min}} \),
\[ \sum_{i=1}^{m} x_i = n, \]
\( x_i \in \{0,1\}, \quad \forall i \in \{1, \ldots, m\} \).

where \( c_{\text{min}}, \) and \( d_{\text{min}} \) are the user-defined parameters that correspond to the minimum allowed total connectivity and the minimum allowed distance between any two access points. Note that, in problem 2, the minimum distance between two access points is maximized by satisfying a minimum total connectivity level and the QoS constraints. In problem 3, the total connectivity between the agents is maximized by satisfying a minimum separation (dispersion) between them and respecting QoS constraints.

4. Solution approaches. We next propose two solution approaches based on tree search. The first approach is TSWB that can be used to solve optimally the multi-objective problem 1 as well as the constrained single-objective problems. In the worst case, however, tree search is an exhaustive approach. Therefore, we also propose a beam search algorithm as a fast heuristic alternative for large-scale problems. Moreover, the results obtained with the beam search or the multi-objective problem can be used to provide the initial solutions for the tree search approach. However, the performance of the heuristic approach may deteriorate when we add constraints to the model, such as the QoS constraints for service points.

One of the pioneering works in combining dynamic programming with branch-and-bound is given in [19]. The authors combine these two methods to solve discrete mathematical programs. Their hybrid algorithm uses elimination by feasibility, domination and optimality. We also make use of these elimination rules. In our application, we use lower and upper bounds to prune the search tree. Carraway and Schmidt use a dynamic programming approach with a branch-and-bound strategy for the capital allocation problem with interdependent projects, which requires a memory feature in the states [6]. The authors develop upper and lower bounds for improving computation time and also memorization of the computed bounds. Our problem also requires the memory of past stages, which makes the problem harder as two incomplete solutions cannot dominate each other.

An idea similar to the beam search here is given in [18], where the authors apply this idea to the time dependent traveling salesman problem. In this application, at each stage a certain number of solutions (user defined parameter) are carried as candidate solutions to the next stage. This approach does not guarantee optimality but is more space- and time-efficient compared to an unrestricted dynamic programming method. Furthermore, this approach gives more flexibility to incorporate precedence relations or time dependence than the linear or nonlinear programming formulations and heuristics.

4.1. Tree search with bounding. Our first approach is TSWB. This approach is illustrated in Figure 1. The root node of the tree starts with the empty set, and the \( k^{th} \) level of the tree consists of cardinality-\( k \) subsets of \( \{1, \ldots, m\} \). The leaf nodes will be at the \( n^{th} \) level of the tree. Since we generate all cardinality \( k \) subsets of \( \{1, \ldots, m\} \) at each level, there will be \( \sum_{k=0}^{n} m!/(k!(n-k)!) \) nodes in the full tree.
Although the tree search approach can be exhaustive, the computational time can be significantly improved by using a bounding scheme. In Figure 1, $\beta^d_U$ represents the upper bound on the distance function $d(\cdot)$ and $\beta^c_U$ is the upper bound on the connectivity function $c(\cdot)$ at that node. The pair $(\beta^d_I, \beta^c_I)$ corresponds to one of the non-dominated solutions and this pair plays the role of the incumbent bound as in the well-known branch-and-bound scheme, [16]. The parameter $\beta^e_U$, defined as the upper bound vector for the function $e(\cdot)$, is also used to fathom nodes, if a service point cannot receive the necessary service level but is omitted in Figure 1 for a clear exposition. A more detailed analysis of the usage of these bounds is given later.

To traverse the tree, we apply depth-first search strategy using simple stacks. Using depth-first search helps us reach new lower bounds faster, and hence, makes the bounding mechanism more efficient. The steps of the tree search method with bounding is given in Algorithm 1. At the beginning, a set of Pareto optimal solutions are used to form the initial solution set $N$ (line 2). These solutions can come from any heuristic, including the beam search heuristic discussed in Section 4.2. The solutions in $N$ are used to determine the incumbent bounds denoted by $\beta^d_I$ and $\beta^c_I$ in Figure 1, with each solution in $N$ providing a different values for $\beta^d_I$ and $\beta^c_I$ and satisfying the QoS constraint. With the Pareto optimal solutions set $N$ and the current upper bounds $\beta^d_U(S)$ and $\beta^c_U(S)$ of solution $S$ at hand, we check for the domination (lines 5 and 11). We also check for feasibility (lines 5 and 11) to guarantee that a solution satisfies QoS constraints. Each non-dominated and potentially feasible partial solution (line 11) is pushed into the stack (line 13) until the leaf node is reached (line 15). If we are at the leaf node, then the upper bounds for this node simply becomes its objective function values. If this pair of solutions constitutes a Pareto optimal solution, then $N$ is updated by adding the new solution to the set and removing the solutions that are dominated by the new solution (line 16). The iterations continue in this fashion until all the branches (children) of a node is processed. Then the node is removed from the stack (line 17).

**Determination of Upper Bounds.** As shown in Algorithm 1, at each node the upper bounds for both objectives need to be calculated. The bounding works as follows: Notice that in the subsequent iterations the minimum distance value cannot increase, since adding other access points may only decrease the minimum distance. So the minimum distance value at node $S$ gives the upper bound $\beta^d_U(S)$ for the nodes branching from this node. On the other hand, obtaining the total connectivity bound $\beta^c_U(S)$ is more involved. We can divide the calculation of this bound into summation of two parts. The first part consists of the total connectivity values between the located access points and the new access points, and the second part is the total connectivity between the new access points that are yet to be located. Suppose that we have already located $q$ access points and the last access point is located at the candidate location $r$. For the first part, we must find $n - q$ access points that give the maximum connectivity value with the already located access points. To avoid the symmetries we keep the solutions ordered so those $n - q$ access points must be from the set \{\(r + 1, \ldots, m\)\}. For the other component of the connectivity, we select the maximum $\frac{(n-q)\times(n-q-1)}{2}$ connectivity values between the access points in the set \{\(r + 1, \ldots, m\)\}. An illustrative example about finding the connectivity upper bound follows.
Example 1. Consider a problem with six access points and ten candidate locations. Suppose that three access points are already located in candidate locations \{1, 3, 4\}, so we proceed to place the remaining access points starting from location 5. At this stage, the connectivity matrix $C$ is given by

\[
\begin{array}{ccccccccccc}
0 & 0.42 & 0.33 & 0.25 & 0.86 & 0.48 & 0.48 & 0.37 & 0.81 & 0.29 \\
0.42 & 0 & 0.64 & 0.74 & 0.10 & 0.54 & 0.92 & 0.61 & 0.68 & 0.45 \\
0.33 & 0.64 & 0 & 0.23 & 0.41 & 0.25 & 0.51 & 0.72 & 0.40 & 0.55 \\
0.25 & 0.74 & 0.23 & 0 & 0.42 & 0.48 & 0.75 & 0.55 & 0.54 & 0.44 \\
0.86 & 0.10 & 0.41 & 0.42 & 0 & 0.77 & 0.52 & 0.40 & 0.24 & 0.24 \\
0.48 & 0.54 & 0.25 & 0.48 & 0.77 & 0 & 0.33 & 0.78 & 0.63 & 0.92 \\
0.48 & 0.92 & 0.51 & 0.75 & 0.52 & 0.33 & 0 & 0.42 & 0.31 & 0.41 \\
0.37 & 0.61 & 0.72 & 0.55 & 0.40 & 0.78 & 0.42 & 0 & 0.47 & 0.55 \\
0.81 & 0.68 & 0.40 & 0.54 & 0.24 & 0.63 & 0.31 & 0.47 & 0 & 0.40 \\
0.29 & 0.45 & 0.55 & 0.44 & 0.24 & 0.92 & 0.41 & 0.55 & 0.40 & 0 \\
\end{array}
\]

where the numbers in bold are used next to calculate the upper bound value. The upper bound is calculated in two parts. In the first part, the upper right section of the matrix is used. We calculate the sum $C_{1i} + C_{3i} + C_{4i}$ for $i = 5, \ldots, 10$. The values are respectively 1.69, 1.21, 1.74, 1.64, 1.75, 1.28. Then we select three largest values, which correspond to candidate locations 5, 7, and 9, respectively. Those values represent the upper bound on the connectivity values among the access points that are located and the distances among the access points we are yet to locate. In the second part, we consider the connectivity values between the access points that are not located. Those values are in the lower right section of the matrix, and we select the largest three values in that part of the matrix. These are 0.77, 0.78 and 0.92 (if we discard the symmetric entries).

For the QoS constraints, it is clear that the maximum connectivity between an access point and a service point cannot decrease, if we add more access points. It
Algorithm 1: TSWB.

1: Input: $n, m, D, C, E$
2: $N \leftarrow \text{INITIAL SOLUTION}$
3: for $t = m - n + 1$ to 1 do
4: \hspace{1em} $S \leftarrow \{t\}$
5: \hspace{2em} if dominated($N, \beta_d^c(S), \beta_c^u(S)$) = false and feasible($\beta_e^u(S)$) then
6: \hspace{3em} stack.push($S$)
7: \hspace{1em} while stack.empty() = false do
8: \hspace{2em} $S \leftarrow \text{stack.top()}$ \hspace{1em} $\text{Does not remove } S \text{ from the stack.}$
9: \hspace{2em} while $S$ has children that are not processed do
10: \hspace{3em} $S \leftarrow S$.nextchild()
11: \hspace{2em} if dominated($N, \beta_d^c(S), \beta_c^u(S)$) = false and feasible($\beta_e^u(S)$) then
12: \hspace{3em} if $|S| < n$ then
13: \hspace{4em} stack.push($S$) \hspace{1em} $\text{Node cannot be eliminated; add to the stack.}$
14: \hspace{4em} Go to line 8
15: \hspace{3em} else if $|S| = n$ then
16: \hspace{4em} $N \leftarrow \text{update}(N, S)$ \hspace{1em} $\text{Update } N \text{ as we reached a leaf node.}$
17: \hspace{2em} stack.pop()
18: Output: $N$

can however increase depending on the connectivity values. To obtain an upper bound we simply choose the best connectivity value for a service point that can be obtained by placing an agent to a location that is yet to be considered. An example to calculate $\beta_e^u(S)$ is given below.

Example 2. Consider a problem with four access points, ten candidate locations, and eight service points. Suppose that three access points are already located in candidate locations $\{1, 3, 4\}$, so we proceed to place the remaining access points starting from location 5. At this stage, the connectivity matrix $E$ is given by

\[
\begin{array}{ccccccccccc}
0.06 & 0.40 & 0.27 & 0.31 & 0.83 & 0.73 & 0.89 & 0.04 \\
0.51 & 0.18 & 0.93 & 0.89 & 0.99 & 0.87 & 0.94 & 0.15 \\
0.99 & 0.66 & 0.98 & 0.58 & 0.40 & 0.99 & 0.79 & 0.77 \\
0.68 & 0.31 & 0.66 & 0.30 & 0.76 & 0.91 & 0.88 & 0.52 \\
0.32 & 0.26 & 0.35 & 0.63 & 0.06 & 0.48 & 0.54 & 0.89 \\
0.53 & 0.73 & 0.77 & 0.68 & 0.43 & 0.10 & 0.34 & 0.03 \\
0.34 & 0.91 & 0.42 & 0.01 & 0.10 & 0.81 & 0.13 & 0.40 \\
0.72 & 0.52 & 0.47 & 0.97 & 0.18 & 0.75 & 0.48 & 0.95 \\
0.44 & 0.68 & 0.70 & 0.06 & 0.89 & 0.82 & 0.33 & 0.73 \\
0.82 & 0.75 & 0.52 & 0.88 & 0.21 & 0.62 & 0.74 & 0.67 \\
0.99 & 0.66 & 0.98 & 0.58 & 0.83 & 0.99 & 0.89 & 0.77 \\
0.99 & 0.91 & 0.98 & 0.97 & 0.89 & 0.99 & 0.89 & 0.95 \\
\end{array}
\]

where the numbers in bold are used to calculate the upper bound value and the last two rows are the current maximum service level (or the lower bound) and the upper bound on the maximum service level for a service point. To calculate the upper bound we take the maximum connectivity value among the access points
already placed (1, 2, and 4), which also gives a lower bound for the connectivity values of the service points. We, then, find the maximum connectivity value among the access points that are not located (from 5 to 10) and we select the largest value as the upper bound on the connectivity value of the service points. For the first and third service points, there is not a change in the connectivity value whereas for the second service point, the value increases from 0.66 to 0.91. If the quality threshold value $c_{min} = 0.8$, then the partial solution is potentially feasible and we explore the children of that partial solution. However, if $c_{min} = 0.9$ the service points 5 and 7 cannot be served adequately and we eliminate the solution using the feasibility constraint.

4.2. Beam search. The beam search approach is basically a fast heuristic that can be applied to unconstrained problems. The main idea of this approach is based on storing, at every stage, at most a fixed number ($W$) of feasible solutions that yield the $W$ best objective function values. Therefore, a truncated search is conducted, since the width of the tree is $W$ throughout the search. Notice that when $W = 1$, we obtain a greedy approach. This myopic approach does not necessarily yield the optimal solution. However, it is extremely fast and it can also be used to provide the initial solution in TSWB algorithm (Algorithm 1, line 1).

As obtaining a good feasible solution helps with bounding, we modified the algorithm to suit constrained problems as well. To obtain a solution for our constrained problems we can keep the algorithm unaltered and check the feasibility of the solutions we obtain at the very end. In our experiments this approach failed to yield feasible solutions for most of the problems. As a result of this failure, we have decided to integrate constraints to every step of the beam search. Independent of the problem we solve, the idea is to keep $W$ best solutions that satisfy a certain constraint depending on the problem we solve. Algorithm 2 shows the details of the beam search heuristic. Here, the generic function $v(\cdot)$ is used to evaluate the objective function value of a solution. Similarly we use a generic $u(\cdot)$ to check the feasibility of the constraints. The set of $W$ best feasible solutions at stage $j$ is denoted by $S_j$. The algorithm starts with the main loop (line 2) and places the first access point in one of the $m$ locations and continues with the truncated tree search. Thus, the search is repeated by placing the first access point in each one of the $m$ locations. If a feasible solution has a better objective function value than a solution in $S_j$, then it is added to the set and the set is updated (lines 7 and 8). The algorithm ends with returning the solution with the maximum objective function value (line 9). Overall, the computational complexity of the beam search algorithm is of the order of $O(Wm^2n)$.

In our models we have three types of limiting constraints: minimum distance between agents, total connectivity between the agents, QoS for service points. For the constraint on minimum distance, we require that the partial solution satisfies the minimum distance requirement or, $d(s \cup \{k\}) \geq d_{min}$. For the constraint on total connectivity, we make sure that a partial solution has the average connectivity required. For a set of cardinality $n'$ the constraint becomes: $c(s \cup \{k\}) \geq c_{min} \frac{2}{n(n-1)} \frac{n'(n'-1)}{2}$. For the QoS constraints on the service points, we do not exercise feasibility check on every step of the algorithm but we check the solutions at the end to see if they respect the service constraints.

Algorithm 2 can be used within Algorithm 1 in two ways: In the first way, the solutions obtained with the beam search algorithm are fed to Algorithm 1 as the initial solution (or as the initial solution set for the multi-objective problem). To
Algorithm 2: Beam Search.

1: **Input:** \( n, m, \omega, D, C, E \)

2: \textbf{for} \( i = 1 \) \textbf{to} \( m \) \textbf{do}

3: \( S_1 \leftarrow \{i\} \)

4: \textbf{for} \( j = 2 \) \textbf{to} \( n \) \textbf{do}

5: \textbf{for} \( s \in S_{j-1} \) \textbf{do}

6: \textbf{for} \( k = 1 \) \textbf{to} \( m \) \textbf{do}

7: \textbf{if} \( (v(s \cup \{k\}) > \min\{v(t) \mid t \in S_j\}) \text{ and } u(s \cup \{k\}) \) \textbf{then}

8: \textbf{update} \( (S_j, s \cup \{k\}, \omega) \)

9: **Output:** \( \arg \max \{v(t) \mid t \in S_j\} \)

obtain this set, we run Algorithm 2 for varying \( W \) values and solve the two extreme cases; (i) neglect the distance values and maximize the total connectivity, (ii) neglect connectivity values and maximize the minimum distance. In the second way, we may integrate the beam search into the search tree of Algorithm 1 to obtain a better lower bound to fathom more branches. During our computational study we have implemented this idea for \( W = 1 \). Even though we were able to find better bounds, the solution time for a problem has increased because of excessive number of calls added by the beam search algorithm. Another important point is that increasing \( W \) may deteriorate the quality of the solutions, as the heuristic is myopic and increasing \( W \) may prevent selecting nodes at a stage that have promising branches in the subsequent iterations. Since it is very fast, in our applications we run the heuristic with different \( W \) values and take the best solution as the lower bound.

5. WMN deployment. We next study the application of the proposed models and the solution approaches in a real WMN node placement problem. In this problem, the connectivity corresponds to the accessibility between two wireless nodes. The connectivity metric is obtained by normalizing the throughput between the nodes, which depends on the parameters of the environment and on the properties of the network nodes that are to be located. In addition to dispersing the access points around the coverage area, we also ensure that the access points have a certain connectivity value among themselves and between the service points, where the demand should be satisfied with a given quality.

To obtain the connectivity matrix, we use the model given in [13], where the entries in the matrix represent connectivity values, calculated as the normalized throughput between the network access points. The resulting connectivity matrix is symmetric with values in the range \([0, 1]\). In this setting, we assume that after locating the nodes, the network we obtain becomes connected, and hence, further feasibility check is not necessary. Considering the quality of modern wireless equipment, this is a realistic assumption. Furthermore, we assume that transmissions from neighbor nodes (nodes that are one hop away from each other) share the channel via a contention based protocol (such as IEEE 802.11 WLAN protocol) for transmissions, and the nodes that are away from each other by two or more hops do not hear each other. This way, inter-node interference is not experienced during
a transmission and the quality of each link can be specified in terms of Signal-to-Noise Ratio (SNR), which is converted to the connectivity metric, throughput, as explained next.

The connectivity between nodes is assessed by the SNR between them, which denotes the quality of the channel. For this purpose, we first calculate $SNR_0$, or the reference $SNR$ level, measured at a reference distance $d_0$ typically taken 1m. Parameter $\alpha$ is the path loss factor (exponent), specific to the propagation environment. Transmit power, the antenna gain of the transmitter and the antenna gain of the receiver are denoted by $P_t$, $G_t$, and $G_r$, respectively. These parameters depend on the properties of the wireless equipment. The power of the received signal at $d_0$ is given by

$$P_0 = \frac{P_t \cdot G_t \cdot G_r \cdot \lambda^2 \cdot d_0^{-\alpha}}{(4\pi)^2},$$

where $\lambda$ is the wavelength of the transmitted signal. The calculation of $SNR_0$ depends on two other environment factors; the bandwidth, $B$ and the noise level, $N_0$. Formally, it is given by

$$SNR_0 = \frac{P_0}{B \cdot N_0}.$$

If there is no obstacle (or barrier) between points $i$ and $j$, then the $SNR$ value between the access points becomes

$$SNR_{ij} = SNR_0 \cdot D_{ij}^{-\alpha},$$

where $D_{ij}$ is the distance between points $i$ and $j$ in meters. Typically, power or $SNR$ values are expressed in logarithmic scale as $10\log(SNR_{ij})$, measured in decibels (dBs). When there is an obstacle between two access points, the attenuation due to that obstacle is incorporated to the model [2]. Consider that the points $i$ and $j$ have $K$ barriers between them with attenuation levels $t_1, t_2, \cdots, t_K$ (dBs), then the total attenuation level between these two access points becomes

$$A_{ij} = \sum_{k=1}^{K} t_k.$$

The attenuation values depend on the material the obstacle or barrier is made of [2]. Using the total attenuation level, we obtain the $SNR$ value of the channel between network access points $i$ and $j$ as

$$SNR_{ij} = SNR_0 \cdot D_{ij}^{-\alpha} \cdot 10^{-A_{ij}}.$$

Let $q$ be the number of encoded bits (number of data bits to be transmitted in unit time) and let the size of the codeword be 32, then the code rate is obtained as $\frac{q}{32}$ and the success rate becomes

$$C_{ij}(q) = \sum_{j=0}^{\left\lfloor \frac{32-q}{2} \right\rfloor} \binom{32}{j} (1 - (1 - \varepsilon)^5)^j \left(1 - \varepsilon^{5(32-j)}\right),$$

where $\varepsilon = \frac{1}{2 + SNR_{ij}}$ denotes the bit-error probability, obtained as a function of $SNR_{ij}$ [2]. Finally, the connectivity value between two access points is found by testing the value for different code rate values and choosing the one that maximizes the normalized throughput per unit time. That is,

$$C_{ij} = \max_{q \in \{2, 4, \cdots, 32\}} \left\{ \frac{q}{32} C_{ij}(q) \right\}, \quad (4)$$
As it can be inferred from 4, a network node can use different code rates for communicating with different nodes. In other words, we need to fix a code rate value for the node’s communication with the possible nodes in each one of the candidate locations.

5.1. An indoor WMN deployment scenario and results. As an example scenario, we aim to deploy the wireless network nodes in a floorplan based on the first floor of the Faculty of Engineering and Natural Sciences (FENS) building at Sabancı University. In this plan, there are 70 candidate locations (see Figure 2 for the floor plan and the candidate locations, marked with +) and 44 possible services points (marked with • in Figure 2). We use Algorithm 2 to provide an initial solution for Algorithm 1. Our computer programs are implemented in C++ on a machine with Intel i7-4600 2.1 GHz processor and 8 GB RAM running Windows 8 operating system.

We have used the parameters in Table 1 for constructing the connectivity matrix. Notice that when we construct the connectivity matrix, we need to find the attenuation levels. This can be accomplished by checking whether the line between any two candidate locations crosses any barrier (e.g. walls, windows) in the site. In our calculations, we assumed that there is no attenuation if nodes and barriers lie on the same line. Considering that the relays have some volume this is not a very restrictive assumption. In the subsequent discussion, we use figures to demonstrate our results. In all of the floor plan figures, a solid circle shows a location selected by the TSWB algorithm and the distances in the figures are given in meters. Furthermore, we employed the greedy heuristic to have an initial feasible solution. The candidate locations are shown in Figure 2. In the computational study, we present the results we obtain with $e_{\text{min}} = 0.8$.

| Parameter | Value |
|-----------|-------|
| $d_0$     | 1 meter |
| $\alpha$  | 2     |
| $P_t$     | 4 Watt |
| $G_t$     | 1     |
| $\lambda$ | $(3 \times 10^8)/(2.4 \times 10^9)$ |
| $B$       | 20 GHz |
| $N_0$     | $3.98 \times 10^{-17}$ |
| $t_{\text{wall}}$ | 5.4 dB |
| $t_{\text{glass}}$ | 7.7 dB |

**Table 1.** The problem parameters.

Figure 3 shows the Pareto curve obtained for the multi-objective problem 1 using TSWB for $n = 5$. The Pareto curve is important because it helps us determine connectivity and distance constraints in single-objective problems. According to Figure 3(a), in the unconstrained problem, we can see that it is possible to increase the minimum distance to more than 10$m$ without a significant decrease in the connectivity. On the other hand, increasing the minimum distance to 25$m$ significantly decreases the connectivity. For the constrained problem with $e_{\text{min}} = 0.8$, similar observations can be made with respect to the Pareto curve in Figure 3(b).

Figure 4 illustrates the solution obtained for problem 2 with $c_{\text{min}} = 40$ for $n = 10$. In Figure 4, the part shown with the dashed line is a barrier formed of meshed glass. It has less attenuation than the walls of the building and therefore it is preferable that wireless nodes are located around that region. The model places six nodes around the meshed glass, three nodes on the same corridor to increase connectivity and a node on a different corridor to satisfy the demands of the service points on that corridor.
Figure 2. The candidate locations on the floor plan. The dashed lines in the middle of the building is a meshed glass and solid lines around the rooms (colored areas) are walls.

Figure 5 shows solutions that we obtain by solving problem 2 for $n = 5$ with two different $c_{\text{min}}$ values. In Figure 5(a), the nodes are well dispersed around the building. Notice that to increase the total connectivity, three nodes are placed around the meshed glass. In Figure 5(b), the agents are not well distributed in the building as four of them are in the same corridor to satisfy the connectivity constraint as well as the service constraint.

Figure 6 shows different configurations that we obtain by solving problem 3 for $n = 5$ with two different $d_{\text{min}}$ values. We can clearly see that the configurations in Figure 6 are quite similar and they differ significantly from the configuration in Figure 5(b). In Figure 5(b) the access points are clustered around the same corridor of the building, whereas in Figure 6 the dispersion becomes the primary concern due to higher $d_{\text{min}}$ value.

Figure 7 shows our solutions again for problem 3 with two $d_{\text{min}}$ values, but this time $n = 10$. As the lower bound on the distance increases in Figure 7, we observe that the access points are forced to disperse along the outer corridors at the expense of decreasing total connectivity.

Table 2 summarizes the results of the solutions considered in this section. We compare the time required to solve the integer programming formulation of our problem (see Appendix A) by using CPLEX, [31], with our approaches under ‘Time’ columns, where Time is reported in seconds. We also report the quality of the solutions obtained by beam search heuristic under ‘Gap’ column. In three instances, the beam search heuristic reaches the optimal solution. In the other three instances, the gap is still quite small. However in two problems, the heuristic provides a solution with 50% deviation. The heuristic is suitable, especially, for finding an initial solution to the problem. The time spent by the heuristic is negligible and is not reported. We also conclude that, except for one instance, TSWB is an order of magnitude faster than CPLEX.

The advantage of using TSWB over exhaustive search is the possibility of fathoming important number of branches in the solution tree. Figures 8, 9 and 10 show the number of solutions that we search and accept at each stage of TSWB. Figure 8
Figure 3. The Pareto curve with $n = 5$ and with different values of $e_{\min}$.

shows the number of possible and selected nodes at each level of the search tree for the multi-objective problem. Figure 8(a) shows the number of possible solutions, for which the objective functions are evaluated. Figure 8(b) shows the number of accepted solutions or the solutions that are not dominated by the Pareto optimal solutions. The problem is computationally intense as we are dealing with a multi-objective problem and elimination is harder. On the other hand, when we compare the results in Figure 8 with the total number of solutions, $70 \choose 10 \approx 3.967 \times 10^{11}$, we can conclude that the search space is significantly reduced.

Figure 9 and 10 show the number of solutions that we search and accept at each stage of the tree search when we applied TSWB to problems given in Figure 4 and Figure 7(a). We report the details of these two problems because the numbers we obtain for the objective functions are quite close for these problems. Although our
fathoming method is slow in the early levels of the tree, it becomes more efficient as we proceed even if we cannot improve the lower bound. These numbers indicate a significant improvement compared to the exhaustive search. In TSWB, if we have a good bound, then we can fathom large portions of the search tree. On the other hand, even when the initial bound is not very good, we observe that we can reach new tighter bounds rather quickly. In the worst case, TSWB boils down to the exhaustive search. Therefore, the effect of increasing $n$ is worse than the effect of increasing $m$. The plots in Figure 9 show that the bounding becomes especially efficient after the fifth stage; i.e., locating the fifth access point. In Figure 9(a) there are almost half a million candidate solutions for the fifth stage but only a tenth of them are accepted as candidate solutions for the next stage as shown in Figure 9(b). We also observe that the performance of the proposed approach depends on the tightness of the bound constraint. Tighter constraints help reduce the size of the search tree. It takes less than half a minute to solve this problem. However, if we solve the same problem with $c_{\text{min}} = 35$, then it takes more than 2.5 minutes to solve the resulting problem.

| Instance | Beam Search | TSWB | CPLEX |
|----------|-------------|------|-------|
| Model    | $n$ | Constraint | Gap | Time | Time |
| 2        | 5   | $c(x) \geq 8$ | 0.00% | 1 | 129 |
| 2        | 5   | $c(x) \geq 9$ | 50.00% | 1 | 561 |
| 2        | 10  | $c(x) \geq 35$ | 0.00% | 155 | 1566 |
| 2        | 10  | $c(x) \geq 40$ | 50.00% | 23 | 1695 |
| 3        | 5   | $d(x) \geq 15$ | 0.10% | 1 | 28 |
| 3        | 5   | $d(x) \geq 20$ | 0.00% | 1 | 7 |
| 3        | 10  | $d(x) \geq 10$ | 0.18% | 22 | 154 |
| 3        | 10  | $d(x) \geq 15$ | 3.54% | 79 | 12 |

Table 2. Results of the computational study for the single objective problems.
Figure 5. The solutions of problem 2 for $n = 5$.

Figure 10 shows the number of solutions that we search and accept at each stage of the tree search when Algorithm 1 applied to problem given in Figure 7(a). The cardinality of the set of all solutions is the same as in Figure 9, since the values for $m$ and $n$ do not change. However, the number of searched nodes in Figure 10(a) is much higher than the number in Figure 9(a). On the other hand, when we compare the results in Figure 10(b) with the results in Figure 9(b), we observe that the increase in the number of accepted nodes is not as drastic as the increase in the number of searched nodes. This is because the problem in Figure 7(a) does not benefit from the tighter bound like the problem in Figure 4.

Throughout the computational study in this section, we observe that solving problem 2 with a tight connectivity constraint is easier than solving problem 3. This is due to the fast reduction of the search space. When we use a high connectivity value as a constraint, then there are much less feasible solutions as the bound can be easily tight for a site in a real-world application. On the other hand, if there
is a connectivity constraint with a small $c_{\min}$ value, then the upper bounds on the connectivity value, in general, is not tight enough in the early stages of the tree. In such a case the problem becomes significantly harder. Therefore computation time is more affected by changes in the constraint tightness in the problem 2. Note also that the connectivity function $c(\cdot)$ is additive and the optimality bound resulting from this function improves slowly at each stage. However, the distance function $d(\cdot)$ is evaluated by taking the minimum over the distances between the locations. Therefore, we reach tighter bounds rather quickly. However, we must note that the problem 3 is less sensitive to changes in the constraints. We also noticed that changing the service constraint does not have an important effect on the computation time. The dispersion of the agents is good enough to ensure high service levels.

6. **Conclusion and future research.** In this paper, we set forth a new multi-objective location problem, where the objectives are related to the dispersion as well
the mutual connectivity of the access points. After presenting the multi-objective mathematical model, we also consider two single-objective problems obtained by a goal programming approach. For solving these problems, we propose two solution approaches. The first approach is a TSWB approach, which searches the whole tree and uses pruning to improve the computation time. This approach is guaranteed to find an optimal solution. However, we observe for very large problems that the computational time becomes high. The second approach is a beam search heuristic. This heuristic may not provide an optimal solution but may provide a feasible solution that can be used as a lower bound in the TSWB in order to reduce the computation time drastically. We also present an application in wireless mesh network design accompanied with some numerical experiments and a comparison with integer programming approach. This application shows that the proposed models and the associated solutions approaches are suitable for solving real-life problems. To
(a) The number of searched nodes at different stages in Algorithm 1.

(b) The number of accepted nodes at different stages in Algorithm 1.

Figure 8. The performance of the tree search algorithm for the multi-objective problem \((n = 10, c_{min} = 0.8)\). The cardinality of the set of all solutions is \(\binom{70}{10} \approx 3.967 \times 10^{11}\).

the best of our knowledge, this is the first work that combines coverage and connectivity with QoS constraints (for service points), while considering obstacles in the environment. Our computational study shows that the optimal solution approach we present is suitable even for moderate size instances.

Using a good upper bound is very important for pruning the search tree and we believe that there is room for improvement in this area. Along the same line, we also plan to consider different heuristics to obtain better lower bounds. However,
Figure 9. The performance of the tree search algorithm for the problem in Figure 4. The cardinality of the set of all solutions is \( \binom{70}{10} \approx 3.967 \times 10^{11} \).

In our experiments we observed that the obtained lower bounds are already quite good and hence, the gain in that area could be marginal.

Appendix A. Integer linear programming formulation. In this section, we present an integer linear programming formulation for problems 2 and 3. We first define for \( j > i \), the auxiliary decision variables

\[
y_{ij} = \begin{cases} 
1, & \text{if two agents are located to candidate points } i \text{ and } j \text{ respectively;} \\
0, & \text{otherwise.}
\end{cases}
\]
(a) The number of searched nodes at different stages in Algorithm 1.

(b) The number of accepted nodes at different stages in Algorithm 1.

Figure 10. The performance of the tree search algorithm for the problem in Figure 7(a).

This variable is used to linearize the quadratic terms in the objective functions and the constraints. We also define a nonnegative decision variable $d$ representing the minimum distance between two agents. Then, the integer linear programming formulation for problem 2 becomes

$$\text{maximize } d, \quad (5)$$

$$\text{subject to } \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} C_{ij} y_{ij} \geq c_{\min}, \quad (6)$$

$$d \leq D_{ij} y_{ij} + M(1 - y_{ij}), \quad 1 \leq i < j \leq m, \quad (7)$$
\[
\sum_{i \in E_k} x_i E_{ik} \geq 1, \quad \forall k \in \{1, \ldots, p\}, \quad (8)
\]
\[
y_{ij} \leq x_i, \quad 1 \leq i < j \leq m, \quad (9)
\]
\[
y_{ij} \leq x_j, \quad 1 \leq i < j \leq m, \quad (10)
\]
\[
x_i + x_j \leq 1 + y_{ij}, \quad 1 \leq i < j \leq m, \quad (11)
\]
\[
\sum_{i=1}^{m} x_i = n, \quad (12)
\]
\[
x_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, m\}, \quad (13)
\]
\[
y_{ij} \in \{0, 1\}, \quad 1 \leq i < j \leq m, \quad (14)
\]

where \( E_k = \{i \mid i = 1, \ldots, m; E_{ik} \geq e_{\text{min}}\} \), i.e. \( E_k \) corresponds to the set of candidate points that can serve service point \( k \). In the objective function \( 5 \), we maximize the minimum distance between any two agents. We restrict the total connectivity to be greater than \( e_{\text{min}} \) by constraint \( 6 \). In constraints \( 7 \), \( M \) corresponds to a large number. For any \( i \) and \( j \), if \( y_{ij} = 0 \), then the constraint is not relevant. If \( y_{ij} = 1 \), then the minimum distance cannot be greater than \( D_{ij} \). With constraints \( 8 \), we guarantee that each service point is served with a certain QoS by at least one agent. Constraints \( 9-11 \) make sure that \( y_{ij} = 1 \) if and only if \( x_i = x_j = 1 \). We limit the number of agents that we locate to \( n \) with constraint \( 12 \). Constraints \( 13 \) and \( 14 \) are the domain constraints for the variables.

For problem \( 3 \), the integer linear programming formulation can be written as

\[
\text{maximize} \quad \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} C_{ij} y_{ij}, \quad (15)
\]

subject to \( d \geq d_{\text{min}}, \quad (16) \)

\[
7 - 14.
\]

This time, we maximize the total connectivity, by equation \( 15 \). With constraint \( 16 \), we guarantee that the agents satisfy the minimum distance restriction.

Acknowledgments. We would like to thank the two anonymous reviewers for their suggestions and comments.

REFERENCES

[1] I. F. Akyildiz, X. Wang and W. Wang, Wireless mesh networks: A survey, Computer Networks, 47 (2005), 445–487.
[2] C. R. Anderson and T. S. Rappaport, In-building wideband partition loss measurements at 2.5 and 60 GHz, IEEE Transactions on Wireless Communications, 3 (2004), 922–928.
[3] F. Birlik, Öz. Gürbüz and Ö. Erçetin, Iptv home networking via 802.11 wireless mesh networks: An implementation experience, IEEE Transactions on Consumer Electronics, 55 (2009), 1192–1199.
[4] V. E. Brimkov, A. Leach, J. Wu and M. Mastroianni, Approximation algorithms for a geometric set cover problem, Discrete Applied Mathematics, 160 (2012), 1039–1052.
[5] P. Cappanera, A survey on obnoxious facility location problems, Tech. Rep. TR-99-11, University of Pisa, 1999.
[6] R. L. Carraway and R. L. Schmidt, An improved discrete dynamic programming algorithm for allocating resources among interdependent projects, Management Science, 37 (1991), 1195–1200.
[7] F. D. Croce, A. Grosso and M. Locatelli, A heuristic approach for the max-min diversity problem based on max-clique, Computers & Operations Research, 36 (2009), 2429–2433.

[8] Z. Drezner and H. W. Hamacher, Facility Location. Applications and Theory, Berlin: Springer, 2002.

[9] E. Erkut and S. Neuman, A multiobjective model for locating undesirable facilities, Annals of Operations Research, 40 (1992), 209–227.

[10] S. Fekete and H. Meijer, Maximum dispersion and geometric maximum weight cliques, in APPROX ’00: Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization, (London, UK), Springer-Verlag, (2000), 132–143.

[11] Ö. Gürbüz and H. Owen, Power control based QoS provisioning for multimedia in W-CDMA, ACM Wireless Networks (WINET), 8 (2002), 37–44.

[12] R. Heydari and E. Melachrinoudis, Location of a semi-obnoxious facility with elliptic maximin and network minisum objectives, European Journal of Operational Research, 223 (2012), 452–460.

[13] P. T. Kabamba, S. M. Meerkov and C. Y. Tang, Optimal, suboptimal, and adaptive threshold policies for power efficiency of wireless networks, IEEE Transactions on Information Theory, 51 (2005), 1359–1376.

[14] R. L. Keeney and H. Raiffa, Decisions with Multiple Objectives, New York: John Wiley, 1976.

[15] A. M. Khedr and W. Osamy, Mobility-assisted minimum connected cover in a wireless sensor network, Journal of Parallel and Distributed Computing, 72 (2012), 827–837.

[16] A. H. Land and A. G. Doig, An automatic method of solving discrete programming problems, Econometrica, 28 (1960), 497–520.

[17] E. Macambira, An application of tabu search heuristic for the maximum edge-weighted subgraph problem, Annals of Operations Research, 117 (2003), 173–188.

[18] C. Malandraki and R. B. Dial, A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem, European Journal of Operational Research, 90 (1996), 45–55.

[19] R. E. Marsten and T. L. Morin, A hybrid approach to discrete mathematical programming, Mathematical Programming, 14 (1978), 21–40.

[20] E. Melachrinoudis and Z. Xanthopoulos, Semi-obnoxious single facility location in euclidean space, Computers & Operations Research, 30 (2003), 2191–2209.

[21] E. Melachrinoudis, Bicriteria location of a semi-obnoxious facility, Computers & Industrial Engineering, 37 (1999), 581–593.

[22] A. T. Murray and R. L. Church, Solving the anti-covering location problem using lagrangian relaxation, Computers & Operations Research, 24 (1997), 127–140.

[23] Y. Ohsawa, Bicriteria euclidean location associated with maximin and minimax criteria, Naval Research Logistics (NRL), 47 (2000), 581–592.

[24] Y. Ohsawa and K. Tamura, Efficient location for a semi-obnoxious facility, Annals of Operations Research, 123 (2003), 173–188.

[25] J. Rakas, D. Teodorovic and T. Kim, Multi-objective modeling for determining location of undesirable facilities, Transportation Research Part D: Transport and Environment, 9 (2004), 125–138.

[26] M. Rebai, M. L. berre, H. Snoussi, F. Hnaïen, and L. Khoukhi, Sensor deployment optimization methods to achieve both coverage and connectivity in wireless sensor networks, Computers & Operations Research, 59 (2015), 11–21.

[27] O. Simeone, A. Maeder, M. Peng, O. Sahin and W. Yu, Cloud radio access network: Virtualizing wireless access for dense heterogeneous systems, Journal of Communications and Networks, 18 (2016), 135–149.

[28] A. J. V. Skriver and K. A. Andersen, The bicriterion semi-obnoxious location (bsl) problem solved by an [epsilon]-approximation, European Journal of Operational Research, 146 (2003), 517–528.

[29] J. L. Williams, J. W. Fisher and A. S. Wilksky, Approximate dynamic programming for communication-constrained sensor network management, IEEE Transactions on Signal Processing, 55 (2007), 4300–4311.
[30] D. Wood, An algorithm for finding a maximum clique in a graph, *Operation Research Letters*, 21 (1997), 211–217.

[31] IBM, IBM ILOG CPLEX Optimizer, www.ibm.com/software/commerce/optimization/cplex-optimizer/, Last retrieved April 2015.

[32] The 5G Infrastructure Public Private Partnership, https://5g-ppp.eu/.

Received March 2016; 1st revision October 2016; 2nd revision December 2016.

E-mail address: byuceoglu@migros.com.tr
E-mail address: sibirbil@sabanciuniv.edu
E-mail address: ogurbuz@sabanciuniv.edu