Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 1683249, 8 pages
https://doi.org/10.1155/2018/1683249

Research Article

A Facility Location Model for Air Pollution Detection

G. Lancia, F. Rinaldi, and P. Serafini

Department of Mathematics and Computer Science, University of Udine, Italy

Correspondence should be addressed to G. Lancia; giuseppe.lancia@uniud.it

Received 30 April 2018; Accepted 3 October 2018; Published 18 October 2018

Academic Editor: Wlodzimierz Ogryczak

We describe mathematical models and practical algorithms for a problem concerned with monitoring the air pollution in a large city. We have worked on this problem within a project for assessing the air quality in the city of Rome by placing a certain number of sensors on some of the city buses. We cast the problem as a facility location model. By reducing the large number of data variables and constraints, we were able to solve to optimality the resulting MILP model within minutes. Furthermore, we designed a genetic algorithm whose solutions were on average very close to the optimal ones. In our computational experiments we studied the placement of sensors on 187 candidate bus routes. We considered the coverage provided by 10 up to 60 sensors.

1. Introduction

A current project is concerned with the quality of air in the city of Rome. Part of this project is devoted to measuring the air pollutants in the city. This is a large scale task which, in principle, could call for many expensive measuring devices to be placed on certain chosen sites or mounted on dedicated vehicles.

In order to reduce the costs and at the same time assure a good significance to the measurements, it has been decided to mount the sensors on the public buses. These sensors measure the pollutants and immediately send the data to a central station, while they are moving within the city according to the bus route.

Clearly only the air pollution in the proximity of a bus route can be measured with a good degree of accuracy. Given that there are only a relatively small number of available sensors, one needs to choose which buses to place the sensors on in order to get the best possible covering of the city area.

Notice that any individual bus on a particular bus route could be used to measure the level of air pollution within the area in proximity of the route itself. Therefore, more than the choice of a set of buses, the problem consists in the choice of a set of bus routes. Once a bus route is chosen in the solution, the sensor can be placed on any of the buses assigned to that route, as long as this bus is not moved from a route to another for a sufficiently long period of time.

The exact number of sensors that will be used in the experiment is not a given parameter. In particular, since sensors are expensive pieces of equipment, one has to face the trade off between the cost of the sensors and the percentage of the urban area that can be covered in the sampling. Hence the goal of our study is to find all Pareto optimal solutions under the objectives of maximizing the sampled area and minimizing the cost of the sensors.

We have been contacted by people involved in the project in order to mathematically model the problem and suggest possible solution procedures. In this paper we report on the outcome of this particular application in which we had to pursue a twofold goal.

On one side we wanted to develop an exact procedure despite the large size of the data set. Having an exact solution is important because it makes it possible to check the quality of the solutions provided by heuristic procedures.

On the other side we wanted also to develop a simple procedure that could be later implemented by persons with a very limited knowledge of operations research tools. This procedure had not to be a black box to be plugged into a larger code, since the people in charge of the final implementation of the project wanted to have full control of each line of the code and they did not want to rely on proprietary software.

As for the exact procedure, we had to overcome the difficulty of dealing with the very large data set which, at first sight, gave no possibility to solve a typical integer linear programming model. However, thanks to the particular form of the objective function, it has been possible to dramatically reduce the number of variables and consequently to solve the exact model.

As for the heuristics, our choice was to use a genetic algorithm due to its implementation simplicity and its potential in solving complex optimization problems. The comparison between the exact solution and the heuristic one has shown a satisfactory quality of the obtained solutions and, in accordance with the people responsible for the final implementation, the genetic heuristic was accepted to generate solutions.

Problems of sensor placement have obtained a considerable attention in the literature. These problems strongly differ from each other depending on the particular application context (for recent surveys we direct the reader to [1, 2]).

In wireless sensor networks [3] sensors transmit data among themselves thus constituting a communication network. Moreover, part of the nodes (both target nodes and sensor nodes) can be mobile, there can be only a partial location awareness, messages can require a routing to reach the central station, energy consumption may affect the activity of the sensors, and obstacles may cause imprecise measurements. Therefore different objective functions are considered in order to take care of all these issues [4–7]. Another research field concerns sensor placement problems in real-time early warning systems, expecially in water distribution networks [2]. In this case the minimization of both the expected (or worst-case) impact of a contamination and the contamination detection time [8–10] is the most relevant task.

Concerning the special application of detecting pollutants, besides placing the sensors in fixed sites [11], in recent years the idea of placing sensors on urban transportation has gained popularity [12–15]. All these contributions focus mainly on the technological aspects of both building the sensors and the network infrastructure, but they do not consider the problem of placing the sensors in an optimal way. In this paper we focus essentially on this aspect.

The paper is organized as follows. In Section 2 we describe the problem and the way we have formally modeled it. In Section 3 we present an exact integer linear programming model and describe how we can reduce its size. In Section 4 we describe the genetic algorithm that we use as a heuristic. In Section 5 we describe the computational tests we have carried out both on the exact model and on the heuristic model for our specific application. Finally, in Section 6 we present the computational results.

2. Problem Modeling

Let N be the number of all bus routes in the urban area under study (in our case the city of Rome). Among these N routes we must select a subset of n routes which cover the area in the best possible way. In order to assess the dependence of the coverage on the number of used sensors, the value of n will be varied within a given range.

In order to solve the problem, it is convenient to first discretize the urban area so it becomes a set of m equally spaced points of a grid (which we will simply call *points* hereafter). The higher the refinement of this discretization is, the larger m becomes, and this can affect significantly the computing time needed to obtain a solution. In this study we have opted for a discretization of the city into a regular grid of squares of size 100m× 100m. Due to the shape of Rome the grid is inscribed in an ellipsis (see Figure 1) and the resulting total number m of points within the ellipsis is 21615.

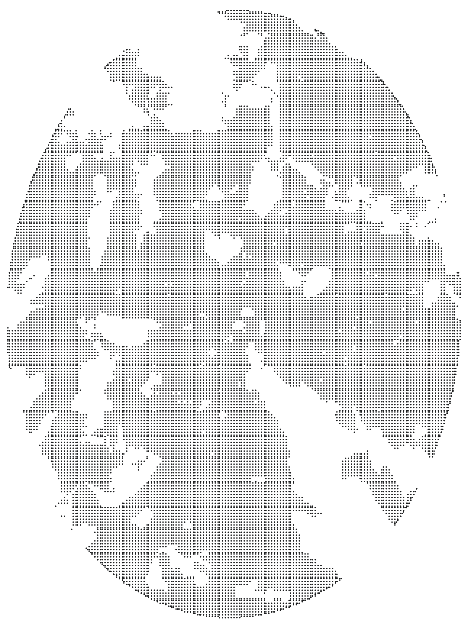
Figure 1: Discretization grid.

Each bus route can be approximated by a sequence of consecutive segments, where each segment is identified by a pair of points (its endpoints). In Rome there are 187 bus routes and the number of segments in each route is variable (it obviously depends on the road being relatively straight or having a lot of turns).

Given the city map data, we have computed an $m \times N$ matrix $D = (\overline{d}_{ij})$ storing the distance $\overline{d}_{ij}$ of each point i from each route j. Even if obtaining the distance matrix is computationally quite demanding (there are 4,042,005 distances to be determined and each distance requires computing a distance for each of the segments of a bus route), the matrix can be computed once and for all in a preprocessing phase.

The computation is based on the following straightforward procedure. The distance of a point $p = (p_1, p_2)$ from a segment defined by its endpoints $a = (a_1, a_2)$ and $b = (b_1, b_2)$ is achieved by a point $q = (q_1, q_2)$ that can be an internal point of the segment or one of the endpoints, according to the following cases:

(1) $(p-a)^T(b-a) \le 0 \ \land \ (p-b)^T(a-b) \ge 0 \Longrightarrow q = a$,

(2) $(p-a)^T(b-a) \ge 0 \ \land \ (p-b)^T(a-b) \le 0 \Longrightarrow q = b$,

(3) $(p-a)^T(b-a) > 0 \ \land \ (p-b)^T(a-b) > 0 \Longrightarrow q$ internal point.

For cases (1) and (2) the respective distances are

$$\sqrt{\left(p_1 - a_1\right)^2 + \left(p_2 - a_2\right)^2},$$
$$\sqrt{\left(p_1 - b_1\right)^2 + \left(p_2 - b_2\right)^2}. \tag{1}$$

In case (3), the well-known formula for computing the distance of a point from a line yields the value

$$\frac{\left|\left(b_2 - a_2\right) p_1 + \left(a_1 - b_1\right) p_2 - a_1 b_2 + b_1 a_2\right|}{\sqrt{\left(a_1 - b_1\right)^2 + \left(a_2 - b_2\right)^2}}. \tag{2}$$

In order to compute the distance of a point $p = (p_1, p_2)$ from a broken line defined as an ordered set of points $a^k = (a_1^k, a_2^k)$, we can simply repeat the above computations for each line segment and then return the smallest of the distances thus computed.

It is reasonable to assume that the route which best covers a given point i is the closest one among those that have been equipped with a sensor. Hence, if J is a subset of routes on which sensors have been placed (i.e., routes that have been "activated"), we can define the distance of i from J as

$$\bar{d}_i(J) := \min_{j \in J} \bar{d}_{ij}. \tag{3}$$

A crucial constraint of the air-pollution monitoring process is that, in order to cover a point, the sensor, i.e., the bus route, needs to pass sufficiently close to the point. If the distance from a route to a point exceeds some given threshold (and it does not matter if it is much larger or just a little) that point is not covered by the route. Given the distance threshold, we can redefine each distance as

$$d_{ij} = \begin{cases} 0 & \text{if } \bar{d}_{ij} \le \hat{d} \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

Alternatively, we can use a continuous distance measure, such as

$$d_{ij} = \begin{cases} 0 & \text{if } \bar{d}_{ij} \le \dfrac{\hat{d}}{2}, \\ 2\dfrac{\bar{d}_{ij}}{\hat{d}} - 1 & \text{if } \dfrac{\hat{d}}{2} \le \bar{d}_{ij} \le \hat{d}, \\ 1 & \text{if } \bar{d}_{ij} \ge \hat{d}. \end{cases} \tag{5}$$

Either way, the important aspect is that each distance exceeding the threshold is a constant; namely, its value is 1. Let us define

$$d_i(J) := \min_{j \in J} d_{ij}. \tag{6}$$

Then, the total coverage of the solution J can be measured as

$$\sum_i d_i(J). \tag{7}$$

If we use d as defined in (4), the objective is the minimization of the uncovered points, which can be seen as representative of the uncovered urban area. If we adopt d as defined in (5) the objective, as before, assigns a penalty to the uncovered points, but it also penalizes the remaining points with a penalty proportional to their distance from the solution.

In accordance with the people responsible for the project we have adopted the definition (5) with $\hat{d} = 400$. For this threshold value there are 2941 points out of 21615 which are further away than from each existing route and which therefore cannot be possibly covered by any solution.

By modeling the problem in the way we just described, we obtain a problem that falls in the class of "Facility Location" problems and is generally referred to in the literature as the "Simple Plant Location Problem" (SPL). See [16, 17] among the many bibliographical sources. In SPL there is a set of candidate sites for the construction of a certain number of facilities each of which offers the same service. Each client needs to be served by a facility, so that he will choose, among the activated facilities, the most convenient one (depending on the distance or some other cost). Clearly, the activation of a facility is expensive. The SPL problem requires to decide how many facilities need to be activated and which are the sites for their construction, so as to minimize the construction cost plus the cost of servicing the customers.

If we interpret each sensor as a facility, each bus route as a site for a facility, and each point as a customer, it is clear that our problem is similar to SPL. The only difference is that while in SPL the costs of activation and service are combined into a unique objective, in our case these two aspects (on one side the cost of the sensors and on the other the quality of the monitoring) are not combined, rather they are considered within a multiobjective approach. In particular, as we already mentioned, the mathematical problem requires determining a good coverage for a fixed value of n and the problem is then solved several times for various values of n. The problem that has to be solved for each n is known as the p-median problem [18]. This problem has been studied in the literature, with different heuristic approaches proposed for its solution [19–21] (although, in general, for smaller instances than the ones we consider). We were told that a possible number of sensors could be around forty, but the decision was not taken yet and indeed our study was supposed to provide concrete suggestions. Therefore we decided to let n vary in the range $\{10, 11, \ldots, 59, 60\}$.

3. A Mixed Integer Linear Programming Model

We can readily express the above problem by a Mixed Integer Linear Programming (MILP) formulation as follows. There are binary variables y_j equal to one if the route j is activated and binary variables x_{ij} equal to one if the route j is the closest, among the activated ones, to the point i. The model is

$$\begin{aligned} v_1 = \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{N} d_{ij}x_{ij} \\ & \sum_{j=1}^{N} x_{ij} = 1 \quad i = 1,\ldots,m \\ & x_{ij} \le y_j \quad i = 1,\ldots,m,\ j = 1,\ldots,N \\ & \sum_{j=1}^{N} y_j = n \\ & x_{ij} \ge 0 \quad i = 1,\ldots,m,\ j = 1,\ldots,N \\ & y_j \in \{0,1\} \quad j = 1,\ldots,N. \end{aligned} \tag{8}$$

The x_{ij} variables need not be constrained to be binary since they are guaranteed to be binary in all optimal solutions. The main issue with (8) is its huge number of variables and constraints. In particular, there are 4,042,005 x_{ij} variables and even more constraints. This makes the resolution of (8) prohibitive.

By exploiting the fact that all distances are set to 1 when the actual distance exceeds the threshold, we can simplify model (8). Indeed, in this case we can remove from (8) all the pairs (i, j) for which $d_{ij} = 1$, since we already know that when any of the x_{ij} variables takes the value 1, it will contribute exactly 1 to the objective function.

The simplification of (8) proceeds as follows. Let

$$Q := \left\{(i,j) : d_{ij} < 1\right\} \tag{9}$$

We only define variables x_{ij} for $(i, j) \in Q$. If $x_{ij} = 1$ (i.e., j is the closest route to point i), we associate a 'profit' $(1 - d_{ij})$ to the variable x_{ij}, and we try to maximize the total profit. The new model is then

$$\begin{aligned} v_2 = \max \quad & \sum_{(i,j)\in Q} \left(1 - d_{ij}\right) x_{ij} \\ & \sum_{j:(i,j)\in Q} x_{ij} \le 1 \quad i = 1,\ldots,m \\ & x_{ij} \le y_j \quad (i,j) \in Q \\ & \sum_{j=1}^{N} y_j \le n \\ & x_{ij} \ge 0 \quad (i,j) \in Q \\ & y_j \in \{0,1\} \quad j = 1,\ldots,N. \end{aligned} \tag{10}$$

In our case with $\widehat{d} = 400$, the cardinality of Q is 108,387, which is a significant improvement over 4,042,005. Thanks to this reduction it has become possible to solve (10) in a reasonable time by using CPLEX as the MILP solver, as shown in Section 5. The values v_1 and v_2 are clearly correlated by $v_1 + v_2 = m$.

We may further reduce the size of the model by the following considerations. If there is a set I_k of points such that $i \in I_k$ is close to a line k ($d_{ik} < 1$, for all $i \in I_k$) but far away from all other lines ($d_{ij} = 1$ for all $j \neq k$ and all $i \in I_k$), then the variables x_{ik}, $i \in I_k$ will be all equal to zero if line k is not activated and all equal to one if line k is activated. We may therefore consider just one of these variables as representative of all others and measure its profit as

$$\sum_{i\in I_k} \left(1 - d_{ij}\right) \tag{11}$$

By exploiting this idea in our case we may reduce the number of variables by 2320. However, we have judged that the improvement with respect to 108,387 is not worth the extra effort of a new implementation with an ad hoc data structure. Hence we have used model (10) in our computations.

We could also decide to reduce the number of constraints by replacing the $|Q|$ constraints $x_{ij} \le y_j$ with N constraints

$$\sum_{i:(i,j)\in Q} x_{ij} \le m y_j \quad j = 1,\ldots,N. \tag{12}$$

In this case, however, we would obtain a much weaker model than the original one. We have not pursued this idea.

4. The Genetic Algorithm

Due to the problem size, we have considered the use of heuristic algorithms. In particular, genetic algorithms (GAs) have proved to be a powerful approach when faced with hard optimization problems [22], and they seem appropriate in our case for two main reasons. First, in genetic algorithms it is very easy to implement changes in the objective function (such as introducing penalties, even nonlinear, or having costs expressed in the form of a big "if-then-else" switch). This feature is quite appealing in our case, since our preliminary model can then be adapted to more complex objective functions that could arise in the future. Second, each solution has a natural representation as a binary vector of N components (one for each bus route) in which the bits set to one correspond to the activated routes. Consequently, each binary vector that has exactly n ones represents a potential solution to the problem. It is well-known that binary vectors of fixed size are the most common representation for chromosomes in a GA.

In the general paradigm for genetic algorithms, at each iteration there is a population of P individuals (i.e., solutions). In a *selection* phase, a subset of p individuals are randomly chosen from the current population (with a probability biased toward the most fit individuals), and then they breed a new generation. In particular, the pairs of selected solutions are used to randomly generate new solutions (their "offspring") which will form the population for the next iteration. The new individuals also undergo a process of random mutations, in which they may acquire features that are not shared by any of their parents. The whole procedure is then repeated until there is no improvement for a certain number (decided by the user) of consecutive iterations.

The main characteristic of a genetic algorithm is probably its *cross-over* procedure, i.e., the way in which a new individual is generated starting from its parents. In the genetic algorithm that we have designed the cross-over works as follows. After generating a random number k, with $1 < k < n$, a set Q of k routes is randomly chosen among the activated routes of the first parent. These routes are inherited by the offspring. Then, a set of $n - k$ routes is randomly chosen among the routes activated in the second parent and not belonging to Q (it is easy to see that this choice is always possible). These routes are then added to the offspring.

The remaining aspects of a GA are the selection procedure and the mutation phase. For selection, we have used the standard approach in which individuals with better fitness (i.e., objective function value) are more likely to be selected than others. As for the mutations, we proceed as follows. Let F be the set of routes activated in a newly generated individual. Then, R routes are eliminated randomly from F and replaced with R random routes which do not belong to F. We have noticed that $R = 2$ yields the best results for the values of n in the range of interest for our instance.

5. Computational Results

We expected the computational effort required to run a full set of tests on the real data to be substantial. In particular, in such tests we would have varied the number of sensors and some parameters of the genetic algorithm (most notably, the population size). Furthermore, we would have also tried to compute the optimal solutions so as to assess the effectiveness of the heuristics. Therefore, we decided to run a set of preliminary tests in which we tried to get an estimate of the running time for the MILP and the genetic algorithm. For these preliminary tests we have used a relatively low threshold, i.e., $\hat{d} = 300$ m, so as to have a smaller number of variables and constraints in the MILP model. Furthermore, we set the number of sensors to $n = 40$, i.e., the possible value suggested by the people responsible for the project. All the computational experiments were run on an `Intel Core I3` machine with 4GB RAM. The algorithm was coded in `C` and compiled under `Linux` with `gcc 4.7`.

In these tests we used both the binary distance (4) and the continuous distance (5). The running time of GA was around one minute, while the time needed for MILP was about 15 minutes.

We were then able to evaluate the relative error of the genetic algorithm, which turned out to be less than 0.6% in both cases. The error was quite low, so that we took the decision to use GA for the real computations. Some further tests were run to decide the best parameters for GA, like the population size, the number of iterations, and the rate of mutations.

After the above preliminary tests, we first made some computations to assess the quality of the solutions provided by the genetic algorithm by using the distance function that was considered adequate with respect to the project goal, namely the distance (5) with $\hat{d} = 400$, i.e.,

$$d_{ij} = \begin{cases} 0 & \text{if } \overline{d}_{ij} \le 200 \\ \dfrac{\overline{d}_{ij}}{200} - 1 & \text{if } 200 \le \overline{d}_{ij} \le 400 \\ 1 & \text{if } \overline{d}_{ij} \ge 400. \end{cases} \tag{13}$$

The genetic algorithm and the exact MILP formulation were run for $n \in \{10, 20, 30, 40, 50, 60\}$. The genetic algorithm was tested with three values of the population size, namely, $P = 1000$, $P = 2000$, and $P = 5000$ and we used selection values of $p = 50$, $p = 60$, and $p = 100$, respectively. As a stopping criterion, we fixed the maximum number of iterations to 200 and the maximum number of iterations without improvements in the incumbent value to 100. For each choice of n and P we carried out 10 computations. In Table 1 we report the average, minimum, and maximum values of v_1 out of the 10 runs and in Table 2 the corresponding relative errors with respect to the optimal values multiplied by 10^4.

As expected, the errors are increasing with n and decreasing with P. We note that for the case $P = 5000$ the GA almost always produces the optimal solution for $n = 10$ and $n = 20$. The average errors are low for $n = 30$ and $n = 40$ (less than 1%). The average error is also acceptable (1.3%) and the minimum error is low (0.55%) for $n = 50$. Only for $n = 60$ there is a sharp increase in the errors; namely, we have 3.4%, 3.0%, and 3.8% for the average, minimum, and maximum errors. Moreover, the range between the minimum and the maximum value is quite narrow. Hence we concluded that the genetic algorithm applied to our specific case exhibits a good performance around the values of n which are more likely to be chosen.

Since the final goal was to provide a tool to the decision makers to analyze the situation and decide about the number of sensors to buy, we had to make a final computation for each value of n in the range $\{10, 11, \dots, 59, 60\}$. In spite of the fact that the running time of the MILP solution was high (about one hour) for the larger values of n, we let the MILP solver go and compute the optimal solutions for all values of n, so as to be able to further evaluate the performance of the genetic algorithm by comparison.

We ran GA, only once for each value of n, with value $P = 5000$. Although the optimal values are monotonically decreasing with respect to n, we have observed that this is not necessarily the case for the solutions provided by GA, clearly because it is a heuristic. To avoid such nonsense outcomes we have devised a reoptimization procedure that can exploit the information contained in the various runs. The reoptimization procedure works in a greedy way for increasing values of n. For any given $n > 10$, we take the solution obtained for $n - 1$ and then add the n-th sensor to it by choosing the best route among those not already selected. This way, we are able to obtain a monotonic behavior of the suboptimal values.

In Figure 2 we report the optimal values obtained by the MILP (in black) and the values obtained by the genetic algorithm with the reoptimization phase (in red). The same data are displayed in Table 3 (the values for $n = 10$ are not

Table 1: Average, minimum, and maximum values over 10 iterations.

n	10 (opt = 15045.3)			20 (opt = 11168.7)			30 (opt = 8799.4)		
	avr	min	max	avr	min	max	avr	min	max
$P = 1000$	15045.3	15045.3	15045.3	11175.3	11168.7	11201.7	8867.77	8817.23	8943.52
$P = 2000$	15045.3	15045.3	15045.3	11172.6	11168.7	11207.9	8850.17	8809.70	8907.81
$P = 5000$	15045.3	15045.3	15045.3	11168.7	11168.7	11168.7	8819.73	8803.65	8856.41
n	40 (opt = 7287.6)			50 (opt = 6267.5)			60 (opt = 5570.)		
	avr	min	max	avr	min	max	avr	min	max
$P = 1000$	7385.03	7336.87	7470.71	6413.64	6364.73	6514.23	5818.69	5766.51	5848.62
$P = 2000$	7380.44	7353.98	7440.74	6402.17	6345.66	6468.70	5822.14	5771.60	5885.14
$P = 5000$	7352.48	7337.17	7377.81	6349.29	6302.23	6416.40	5760.82	5735.26	5782.13

Table 2: Relative errors multiplied by 10^4.

n		10			20			30	
	avr	min	max	avr	min	max	avr	min	max
$P = 1000$	0	0	0	6	0	30	78	20	164
$P = 2000$	0	0	0	4	0	35	58	12	123
$P = 5000$	0	0	0	0	0	0	23	5	65
n		40			50			60	
	avr	min	max	avr	min	max	avr	min	max
$P = 1000$	134	68	251	233	155	394	446	353	500
$P = 2000$	127	91	210	215	125	321	453	362	566
$P = 5000$	89	68	124	131	55	238	343	297	381

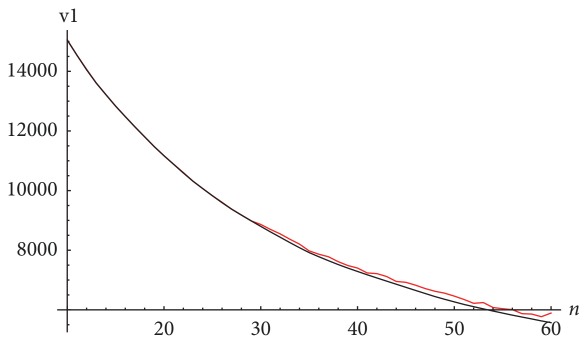

Figure 2: Solution values as a function of n.

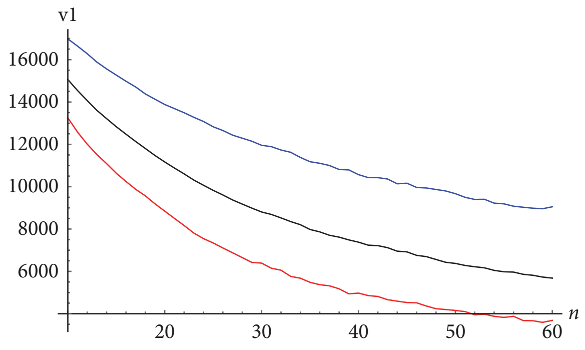

Figure 3: Covered, partially covered, and uncovered points.

shown) where in addition the relative errors with respect to the optimal values multiplied by 10^4 are also shown.

We also provide a particular plot (see Figure 3) in which we may better understand how the points are covered by the solution. The red line (bottom) reports the number of uncovered points, i.e., the points with distance exceeding 400 m from each activated route. The blue line (top) shows the number of uncovered points plus the number of partially covered points, i.e., with distance between 200 m and 400 m. The black line shows the values of the objective function v_1. Since the black line is almost halfway between the other two lines, this shows that the number of partially covered points is distributed in a quite uniform way in the range 200-400 m.

In Figure 4 we show in a third different way the solution values (obtained by GA) for 10, 20, 30, 40, 50, and 60 sensors. The black points are the 2941 points that cannot be possibly covered by any solution. The red points are the uncovered points, i.e., the points exceeding 400 m from each activated route (these points could be covered by some of the nonactivated routes). The purple points are the partially covered points (i.e., with distance between 200 m and 400 m) while the blue points are the points whose distance from the activated routes is smaller than 200 m.

6. Conclusions

In this paper we have investigated a particular problem of sensor placement originated by a real application. The application is related to a project aimed at assessing and

Table 3: Optimal values, GA values, and errors for all n from 11 to 60.

n	11	12	13	14	15	16	17	18	19	20
opt MILP	14542.5	14051.8	13606.0	13216.4	12832.9	12481.3	12135.4	11803.5	11472.5	11168.7
GA + reopt	14542.5	14051.8	13606.0	13216.4	12832.9	12481.3	12135.4	11803.5	11472.5	11168.7
error x 10^4	0	0	0	0	0	0	0	0	0	0
n	21	22	23	24	25	26	27	28	29	30
opt MILP	10881.5	10589.9	10314.0	10068.2	9827.6	9593.8	9374.9	9179.0	8986.2	8799.5
GA + reopt	10881.5	10589.9	10314.0	10068.2	9827.6	9593.8	9374.9	9179.0	8986.2	8803.6
error x 10^4	0	0	0	0	0	0	0	0	0	5
n	31	32	33	34	35	36	37	38	39	40
opt MILP	8614.6	8437.0	8257.5	8084.3	7920.1	7781.5	7648.0	7519.0	7400.3	7287.7
GA + reopt	8668.9	8448.8	8290.8	8168.6	7977.5	7813.6	7699.8	7577.9	7468.9	7349.6
error x 10^4	63	14	40	104	72	41	68	78	93	85
n	41	42	43	44	45	46	47	48	49	50
opt MILP	7176.4	7069.6	6964.3	6859.5 6	6754.9	6650.8.	6548.0	6445.3	6354.2	6267.6
GA + reopt	7226.2	7106.0	7001.9	6890.2	6824.1	6760.0	6583.4	6481.1	6414.2	6365.9
error x 10^4	69	51	54	45	102	164	54	56	94	157
n	51	52	53	54	55	56	57	58	59	60
opt MILP	6181.7	6106.1	6032.3	5960.2	5890.8	5821.9	5758.2	5694.8	5629.0	5570.1
GA + reopt	6263.4	6219.1	6144.4	6085.6	6019.2	5927.3	5850.4	5800.1	5714.7	5684.2
error x 10^4	132	185	186	210	218	181	160	185	152	205

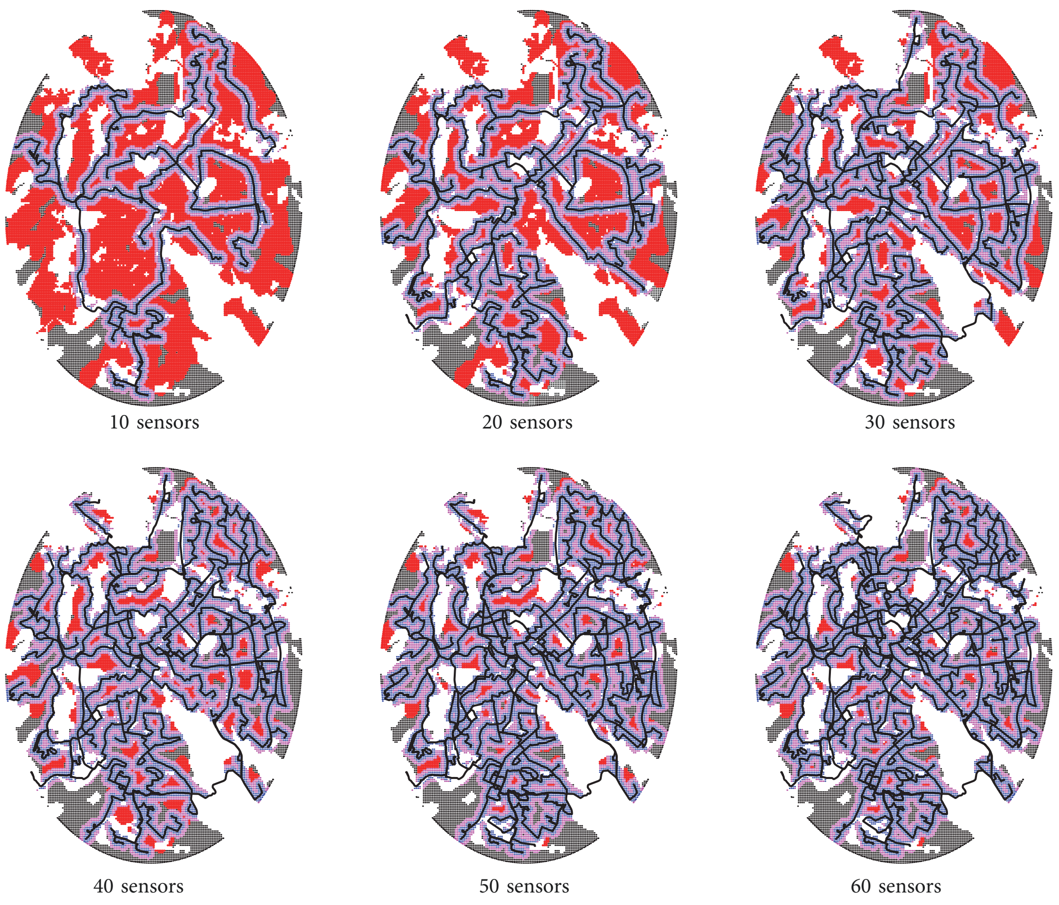

Figure 4: GA coverage for various numbers of sensors.

improving the air quality in the city of Rome. One of the issues of the project is concerned with the determination of the number of sensors in order to have a good coverage of the measured air, taking into account the decision of mounting the measuring devices on the urban buses.

It turned out that a Facility Location model (in particular the p-median problem) best fits the real problem. In spite of the large number of data we have been able to develop a viable integer linear programming model to find exact solutions. Although only a genetic algorithm will be eventually used in the real application, we found that it is mandatory to have available also the exact solutions to check the quality of the heuristic solutions. We have found out that a genetic algorithm meets in a satisfactory way the two goals of having a simple procedure to be implemented and at the same time a reliable solution.

Data Availability

There are no available data due to proprietary reasons of the company involved in the project.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] V. L. Boginski, C. W. Commander, P. M. Pardalos, and Y. Ye, "Sensors: theory, algorithms, and applications," in *Optimization and Its Applications*, vol. 61, Springer, New York, 2012.

[2] S. Rathi and R. Gupta, "Sensor Placement Methods for Contamination Detection in Water Distribution Networks: A Review," *Procedia Engineering*, vol. 89, pp. 181–188, 2014.

[3] A. Nayak and I. Stojmenovic, *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*, John Wiley & Sons, 2010.

[4] E. M. Craparo, "Constrained node placement and assignment in mobile backbone networksSensors: Theory, Algorithms, and Applications," in *Optimization and Its Applications*, V. L. Boginski, C. W. Commander, P. M. Pardalos, and Y. Ye, Eds., vol. 61, Springer, New York, 2012.

[5] F. Guerriero, A. Violi, E. Natalizio, V. Loscri, and C. Costanzo, "Modelling and solving optimal placement problems in wireless sensor networks," *Applied Mathematical Modelling*, vol. 35, no. 1, pp. 230–241, 2011.

[6] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom '04)*, pp. 45–57, October 2004.

[7] C. Vogiatzis, "Sensors in transportation and logistics networks Sensors: Theory, Algorithms, and Applications," in *Optimization and Its Applications*, V. L. Boginski, C. W. Commander, P. M. Pardalos, and Y. Ye, Eds., vol. 61, pp. 145–163, Springer, New York, 2012.

[8] T. Y. Berger-Wolf, W. E. Hart, and J. Saia, "Discrete sensor placement problems in distribution networks," *Mathematical and Computer Modelling*, vol. 42, no. 13, pp. 1385–1396, 2005.

[9] J. W. Berry, W. E. Hart, C. A. Phillips, and J. Watson, "A Facility Location Approach to Sensor Placement Optimization," in *Proceedings of the Eighth Annual Water Distribution Systems Analysis Symposium (WDSA)*, pp. 1–4, Cincinnati, Ohio, United States.

[10] J.-P. Watson, R. Murray, and W. E. Hart, "Formulation and optimization of robust sensor placement problems for drinking water contamination warning systems," *Journal of Infrastructure Systems*, vol. 15, no. 4, pp. 330–339, 2009.

[11] Y. Ma, M. Richards, M. Ghanem, Y. Guo, and J. Hassard, "Air pollution monitoring and mining based on sensor Grid in London," *Sensors*, vol. 8, no. 6, pp. 3601–3623, 2008.

[12] K. de Zoysa, C. Keppitiyagama, G. P. Seneviratne, and W. W. A. T. Shihan, "A public transport system based sensor network for road surface condition monitoring," in *Proceedings of the 2007 Workshop on Networked Systems for Developing Regions*, Kyoto, Japan, August 2007.

[13] S. Devarakonda, P. Sevusu, H. Liu, R. Liu, L. Iftode, and B. Nath, "Real-time air quality monitoring through mobile sensing in metropolitan areas," in *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing (UrbComp '13)*, article 15, ACM, Chicago, Ill, USA, August 2013.

[14] G. Lo Re, D. Peri, and S. D. Vassallo, "Urban Air Quality Monitoring Using Vehicular Sensor Networks," in *Advances onto the Internet of Things*, vol. 260 of *Advances in Intelligent Systems and Computing*, pp. 311–323, Springer International Publishing, Cham, 2014.

[15] M. Pavani and P. T. Rao, "Urban Air Pollution Monitoring Using Wireless Sensor Networks: A Comprehensive Review," *International Journal of Communication Networks and Information Security (IJCNIS)*, Article ID 9439449, pp. 9439-449, 2017.

[16] J. Krarup and P. M. Pruzan, "The simple plant location problem: survey and synthesis," *European Journal of Operational Research*, vol. 12, no. 1, pp. 36–81, 1983.

[17] P. B. Mirchandani and R. L. Francis, *Discrete Location Theory*, John Wiley & Sons, 1990.

[18] M. S. Daskin, K. L. Maass, G. Laporte, S. Nickel, and F. Saldanha-da-Gama, *The p-Median ProblemLocation Science*, Springer, 2015.

[19] N. Mladenović, J. Brimberg, P. Hansen, and J. A. Moreno-Pérez, "The p-median problem: a survey of metaheuristic approaches," *European Journal of Operational Research*, vol. 179, no. 3, pp. 927–939, 2007.

[20] F. Chiyoshi and R. D. Galvão, "A statistical analysis of simulated annealing applied to the p-median problem," *Annals of Operations Research*, vol. 96, no. 1-4, pp. 61–74, 2000.

[21] F. García-López, B. Melián-Batista, J. A. Moreno-Pérez, and J. M. Moreno-Vega, "The parallel variable neighborhood search for the p-median problem," *Journal of Heuristics*, vol. 8, no. 3, pp. 375–388, 2002.

[22] R. Vinter, "Dynamic optimization Arthur E. Bryson; Addison-Wesley Longman, Inc., Reading, MA, 1999,," *Automatica*, vol. 38, no. 10, pp. 1831–1833, 2002.