Scientia Iranica A (2011) 18 (4), 867–877

Sharif University of Technology

Scientia Iranica

Transactions A: Civil Engineering

www.sciencedirect.com

Solving the conditional and unconditional *p*-center problem with modified harmony search: A real case study

A. Kaveh *, **H. Nasr**

Department of Civil Engineering, Iran University of Science and Technology, Tehran, P.O. Box 16846-13114, Iran

Received 22 August 2010; revised 21 April 2011; accepted 23 May 2011

KEYWORDS

Harmony search;
p-center problem;
Conditional;
Unconditional;
Bicycle stations.

Abstract In this paper, we solve the well-known conditional and unconditional *p*-center problem using a modified harmony search algorithm. This music inspired algorithm is a simple meta-heuristic that was proposed recently for solving combinatorial and large-scale engineering and optimization problems. This algorithm is applicable to both discrete and continuous search spaces. We have tested the present algorithm on ORLIB and TSP test problems and compared the results of the classic harmony search approach to those of the modified harmony search method. We also present some results for other meta-heuristic algorithms including the variable neighborhood search, the Tabu search, and the scatter search. Finally, we utilize this location model to locate bicycle stations in the historic city of Isfahan in Iran.

1. Introduction

One of the most famous problems for dealing with locating facilities is the *p*-center problem. In this problem, the maximum distance of customers (demand points) from the facilities is an essential factor. This problem is also known as the minimax location problem. *n* demand points are located in the space and they are in fixed positions in static facility locations. The main objective is to locate *p* facilities among these demand points in such a way that the maximum distance between demand points and their nearest facility becomes minimum. We assume that all the facilities are identical and give the same service to the customers, and there is no limit for the number of customers who can get service from the centers. This kind of location problem is suggested by Hakimi [1,2], and some of its applications are used to locate fire stations, hospital emergency services, data file location, police stations, and so on.

* Corresponding author.
E-mail address: alikaveh@iust.ac.ir (A. Kaveh).

1026-3098 © 2011 Sharif University of Technology. Production and hosting by Elsevier B.V. Open access under CC BY-NC-ND license.
Peer review under responsibility of Sharif University of Technology.

doi:10.1016/j.scient.2011.07.010

Production and hosting by Elsevier

Geem et al., in 2001, proposed a new meta-heuristic algorithm, so-called the Harmony Search (HS) algorithm [3]. Although it is comparatively a new meta-heuristic algorithm, in various applications, it has been proven to be a robust and efficient tool. After being presented by Geem et al., various problems are solved with this algorithm, including function optimization, engineering problems, groundwater modeling, design of water distribution networks, energy-saving dispatch, slope stability analysis, truss design and so on [4]. This fast algorithm has also been combined with other algorithms, such as particle swarm optimization, ant colony optimization [5], and genetic algorithms [6].

Harmony search is a meta-heuristic algorithm, which is inspired by the composition of a piece of music. For composing a piece of music, the aim is to have the maximum Aesthetic, and the musician looks to make perfect harmony. This aim is analogous to finding the optimal solution for an optimization or engineering problem. The harmony search algorithm has been used mostly to solve continuous optimization problems, but here we want to utilize this simple and efficient algorithm for a discrete problem.

2. The *p*-center problem

The *p*-center problems are either continuous or discrete. Many authors have worked on continuous problems in which points can be located anywhere in the space, but another interesting problem is the discrete *p*-center problem in which there are only a few points, such as (x_j, y_j), in the space on which we can position our facilities. In some other problems, customers (a_i, b_i) associate weights, w_i, with each specific

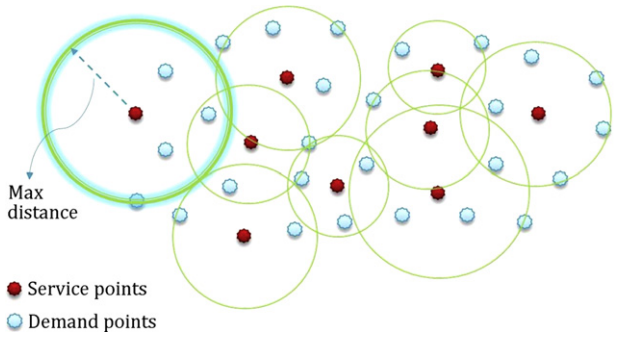

Figure 1: A set of demand points and service points and the maximum distance between them.

point, which means they need a specific amount of service. In some problems, the distances between customers and service points are defined as Euclidean distances. Also, in some other problems, the distances are defined as minimal distances on a graph or network. The p-center problem, with these kinds of distance, was first solved by Minieka [7]. As mentioned before, in a p-center problem, the objective is to locate p facilities (service points) among demand points in such a way that the maximum distance between the demand points and their nearest facility becomes minimum. Figure 1 shows a set of demand points and service points. Red points indicate service points and blue ones are the demand points. In this figure, the maximum distance between a demand point and a service point is specified with a highlighted circle. The aim of the p-center problem is to find the location of the service points, in order to minimize the maximum distance between demand points and their nearest facility.

The formulation of the Euclidean p-center problem with one facility can be expressed as:

$$f(x, y) = \max\left\{\sqrt{(x - a_i)^2 + (y - b_i)^2}\ 1 \le i \le n\right\}, \tag{1}$$

where (a_i, b_i) are n existing customers, and (x, y) is the candidate point for locating a new service point. The objective is to minimize the above function. An equivalent function to the above problem is as follows [8]:

$$\min Z. \tag{2}$$

Subject to:

$$\sqrt{(x - a_i)^2 + (y - b_i)^2} \le Z. \tag{3}$$

The integer programming form of the non-weighted vertex p-center problem is:

$$\min Z. \tag{4}$$

Subject to:

$$\sum_j X_{ij} = 1 \quad \forall i, \tag{5}$$

$$\sum_j Y_j = P, \tag{6}$$

$$X_{ij} \le Y_j \quad \forall i, j, \tag{7}$$

$$Z \ge \sum_j d_{ij} X_{ij} \quad \forall i, \tag{8}$$

$$X_{ij}, Y_j \in \{0, 1\} \quad \forall i, j. \tag{9}$$

Y_j is 1 when the service point is located at node j, and it is 0 otherwise; X_{ij} is 1 if customer i is assigned to the service point that was set up on node j, and 0 otherwise; d_{ij} is the length of the shortest path between demand node i to facility j; Z is the maximum distance between a demand node and the nearest facility; p is the number of facilities that are supposed to be located.

In this study, we consider a discrete p-center problem in which each facility can only be located on the nodes of the graph, thus the solution vector consists of the number of nodes. For instance, suppose we want to find 5 centers for a graph with 50 nodes. A sample solution for this problem can be stated as follows:

$$X = (3, 14, 22, 31, 45).$$

This harmony (solution) means nodes 3, 14, 22, 31 and 45 are the centers of the graph. The discrete p-center problem has been proved to be NP-hard [9]. Some authors have presented exact methods and have solved the p-center problem optimally by mathematical formulation and relaxation algorithms. Ilhan et al. [10] proposed a new exact method for solving a discrete p-center problem. Elloumi et al. presented an integer linear programming formulation for solving a discrete p-center problem and improved the Daskin algorithm [11]. Handler and Mirchandani also used relaxation approaches to solve this problem [12]. Daskin presented an optimal algorithm, which solves the discrete p-center problem by performing a binary search over possible solution values [13]. His algorithm solved maximal covering sub-problems rather than the set-covering sub-problems solved by Minieka [7]. Watson-Gandy proposed a new algorithm to solve problems optimally up to around 50 customers and 3 centers in a reasonable time [14]. Chen and Chen presented new relaxation based algorithms for the solution of continuous and discrete p-center problems [15]. Chen proposed a method that enabled both the solution of the continuous p-center and p-median problems, by utilizing a differentiable approximation to the objective function, and solving it with nonlinear programming [16]. Drezner suggested exact algorithms for the p-center problem in the plane [17, 18]. His heuristic algorithm solved the problems up to 2000 customers and 10 service points, and the optimal method solved problems up to 30 customers and 5 service points or 40 customers and 4 service points. The p-center problem on the graph was first solved by Minieka [7] and also Toregas et al. [19]. Hwang et al. suggested a slab-dividing approach, which efficiently solved the Euclidean p-center problem [20]. Suzuki and Drezner presented heuristic methods and upper bounds on the optimal solution, where the customer points were distributed on a square [21]. Caruso et al. presented an algorithm for solving discrete p-center problems [22]. For the first time, Mladenovic et al. [23] proposed meta-heuristic algorithms, two Tabu Search algorithms and a Variable Neighborhood Search algorithm for solving the p-center problem. Pacheco and Casado developed a metaheuristic procedure based on the scatter search approach for solving the p-center problem and the maximum set covering problem. Due to a limited budget, these authors solved both problems with less than 10 facilities ($p \le 10$) [24].

3. Harmony search algorithm

The harmony search algorithm is a meta-heuristic algorithm for optimizing mathematical functions and engineering problems, which is inspired by the art of music [3]. Analogous to the way musical instruments are played, with certain discrete musical notes, based on musician experience or randomness in

an improvisation process, the design variables can also be assigned with certain discrete values, based on computational intelligence or randomness in the optimization process. Similar to the way a musician improves his skill, based on an aesthetic standard, design variables in a computer memory can be improved based on objective function.

Steps of the harmony search algorithm are summarized as follows:

Step 1. Define the objective function of the problem and the algorithm parameters (HMCR, PAR, bw);
Step 2. Construct the Harmony Memory (HM);
Step 3. Improvise a new harmony;
Step 4. Update the harmony memory;
Step 5. Check the stopping criterion.

These steps are explained in the following sections.

Defining the objective function and algorithm parameters: In this step, the optimization problem is described as follows:

Minimize$f(X)$.

Subject to:

$$L_i \le x_i \le U_i \quad i = 1, 2, \ldots, N, \tag{10}$$

where $f(X)$ is the objective function; X is the set of each decision variable, and N is the number of decision variables. Each decision variable may be restricted to certain upper and lower bounds, $L_i \le x_i \le U_i$, where L_i and U_i are the lower and upper bounds for each decision variable.

In this step, the parameters of the harmony search algorithm are also supposed to be defined, including the Harmony Memory Size (HMS), the number of harmonies (solution vectors) in the harmony memory, the Harmony Memory Considering Rate (HMCR), pitch adjusting rate (PAR), Bandwidth (bw), and the number of iterations we want the algorithm to execute (ITR). These are the basic parameters of the algorithm [4].

The harmony memory is a matrix (memory location) where best harmonies (solution vectors) are saved. Harmony memory is analogous to the genetic pool in the genetic algorithm. HMCR, PAR and bw are parameters which are used to control the diversification and intensification of the harmony search algorithm and to improve the harmonies in the harmony memory [4].

Construct the harmony memory: As mentioned before, in this step, we initialize the harmony memory. In order to do this, the algorithm generates random solution vectors (harmonies) HMS times and puts them in the HM matrix:

$$\mathrm{HM} = \begin{bmatrix} x_1^1 & \cdots & x_N^1 \\ \vdots & \ddots & \vdots \\ x_1^{\mathrm{HMS}} & \cdots & x_N^{\mathrm{HMS}} \end{bmatrix}. \tag{11}$$

At the beginning of the algorithm, each row of the harmony memory matrix represents a random solution vector.

Improvise a new harmony: Once the harmony memory matrix is initialized, the algorithm starts the first iteration by improvising a new harmony. $X = (x_1, x_2, \ldots, x_n)$ is a new solution vector that is constructed based on the following three rules:

1. Memory consideration with probability HMCR;
2. Pitch adjusting with probability PAR;
3. Random selection with probability (1-HMCR).

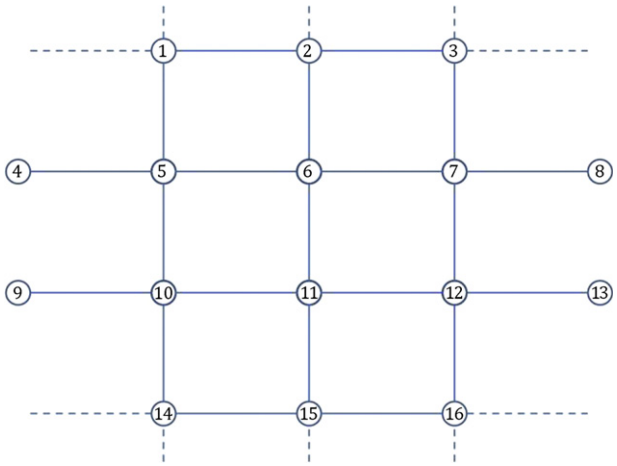

Figure 2: A sample graph.

x_1 is the value of the first decision variable for the new harmony and is selected from the values that are in the first column of the harmony memory (x_1^1 to x_1^{HMS}), with probability HMCR. HMCR is a parameter between 0 and 1. It is the probability of selecting one value from the historical values stored in the HM, while (1-HMCR) is the probability of randomly selecting one value from the possible range of values [4]. Values of the remaining decision variables $(x_2, \ldots, x_n)$ are selected in the same way. For instance, when HMCR is 0.9, the probability that the HS algorithm chooses the decision variable value from historically stored values in the HM is 0.9, and the probability of selecting from the total space is 0.1. Once a decision variable is selected by memory consideration, it is examined to determine whether it should be pitch-adjusted. The pitch adjusting operation uses the PAR parameter, which is between 0 and 1, and is the probability of pitch adjusting the selected value; (1-PAR) is the probability of doing nothing to the value (not adjusting the selected value).

Pitch adjusting for x_i

$$\leftarrow \begin{cases} \text{Yes} & \text{with.probability} \quad \text{PAR} \\ \text{No} & \text{with.probability} \quad 1 - \text{PAR} \end{cases}. \tag{12}$$

If the pitch adjustment decision for x_i is YES, x_i is replaced as follows:

$$x_i \leftarrow x_i \pm (\mathrm{rand}() \times \mathrm{bw}), \tag{13}$$

where rand() is a random number between 0 and 1 and bw is the bandwidth.

The above formulation is for continuous variables. For discrete variables, like the p-center problem, the pitch adjustment can be stated as:

$$x_i \leftarrow x_i(k \pm m), \tag{14}$$

where k is the index of the selected node and m is a random integer in bw range. Figure 2 shows how the selected node can be substituted with its neighbors. Suppose the selected node is the node number 5, bw is 5, and m is randomly selected as 2 in the bw range. According to Formula (14), node number 5 can be substituted by nodes number 3 or 7. It is obvious that in this approach, some of the neighbors are far from the selected node, and it is very likely that the high quality solution turns to a bad one. A new approach for pitch adjusting is represented in Section 4.1.

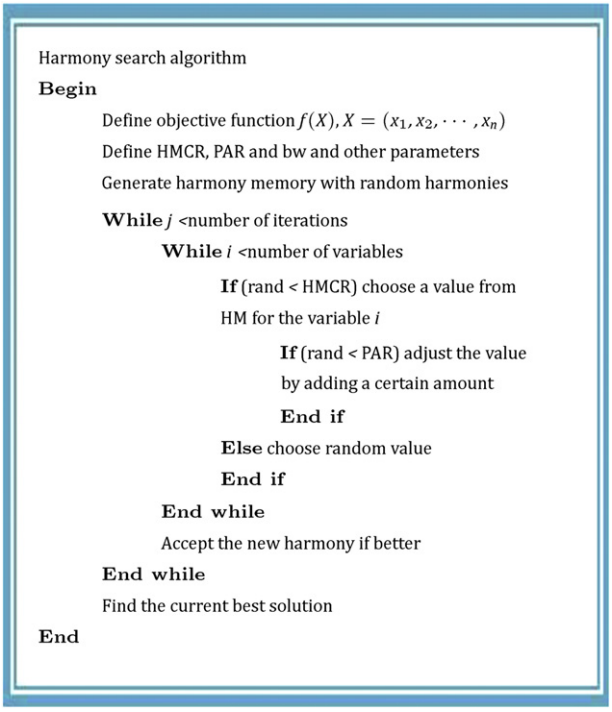

Figure 3: The pseudo code for the classic harmony search algorithm.

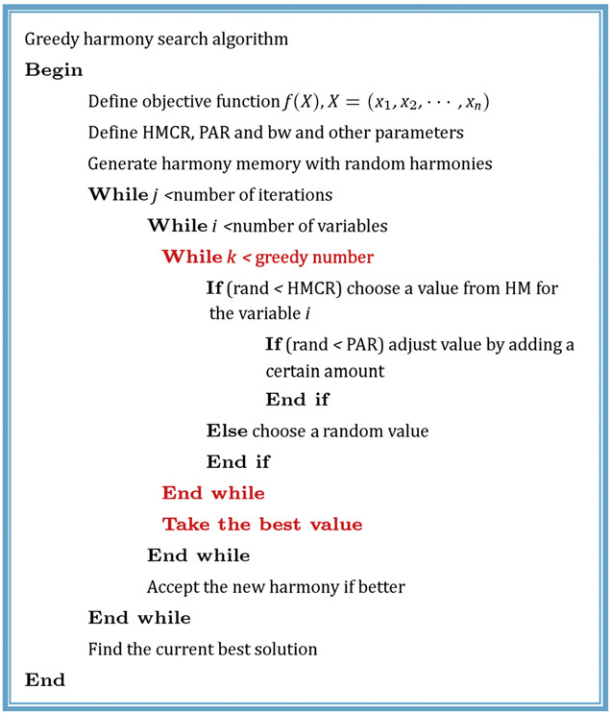

Figure 4: The pseudo code for the greedy harmony search algorithm.

The original harmony search algorithm consists of three operations for considering computational intelligence or randomness, as follows:

$$x_i^{\text{new}} \leftarrow \begin{cases} x_i(k) \in \{x_i(1), x_i(2), \ldots, x_i(k_i)\} & \text{with.probability.} \\ x_i(k) \in \{x_i^1, x_i^2, \ldots, x_i^{\text{HMS}}\} & \text{with.probability.} \\ x_i(k \pm m) \text{ or } x_i \pm (\text{rand}() \times \text{bw}) & \text{with.probability.} \end{cases}$$

$$\left.\begin{array}{c} p_{\text{random}} = 1 - \text{HMCR} \\ p_{\text{memory}} = \text{HMCR} \times (1 - \text{PAR}) \\ p_{\text{pitch}} = \text{HMCR} \times \text{PAR} \end{array}\right\}.$$

Update harmony memory: Once the new harmony (new solution vector) $X = (x_1, x_2, \ldots, x_n)$ is improvised, it is compared to the worst harmony in the HM. If the new harmony is better than the worst harmony in the HM, based on the objective function value, the new harmony is included in the harmony memory and the existing worst harmony is excluded from the harmony memory.

Check the stopping criterion: If the stopping criterion (maximum number of iterations, obtaining the optimum value for the problem and etc.) is satisfied, computation is terminated. Otherwise, Steps 3 and 4 are repeated (see Figure 3) [4].

4. Discrete greedy harmony search algorithm for the *p*-center problem

In this paper, we use a new meta-heuristic algorithm inspired by music, called the harmony search algorithm [3]. Kaveh and Nasr combined the modified harmony search algorithm with the greedy heuristic and obtained a robust algorithm, named the Greedy Harmony Search (GHS) [25]. It is discussed that with a constant HMCR, there is a problem with diversification if the number of nodes in HM is a small fraction of the total nodes. This means that if we choose HMCR to be around 0.95–0.99, the search space will be totally limited to the nodes of the HM matrix and the algorithm gets trapped in a local optimum or at least the algorithm converges slowly. On the other hand, if we choose the HMCR parameter to be around 0.4–0.7, this will be a good strategy at the beginning of the algorithm, because the diversification of the search remains high at this stage and the algorithm searches the entire search space. However, as the algorithm progresses, the diversification should be decreased. The low HMCR makes the algorithm to converge less quickly and damages the intensification. In order to conquer this problem, variable HMCR (VarHMCR) is introduced (similar to Mahdavi et al. [26], defined variables, PAR (VarPAR) and bw (Varbw) are as given in Formulas (17)–(19)). To combine the harmony search with the greedy heuristic, we generate several random numbers in interval [0, 1] for a predefined number of times (greedy number) and compare those random numbers with VarHMCR, "greedy number" times. In this way, we select several arrays of the column of the HM and several random nodes of the graph (depending on the VarHMCR), and select the best of them, according to their fitness values. Choosing the best node among some selected nodes increases the efficiency and robustness of the algorithm. Here, we use this algorithm to solve the conditional and unconditional *p*-center problem. Figures 3 and 4 show the pseudo code of the classic harmony search and the greedy harmony search. As is obvious, one loop is added to the pseudo code of the basic harmony search algorithm and it is repeated for a specified number of times (greedy number). For example, if the greedy number is equal to 5, the algorithm repeats the new loop 5 times, and generates five different values for the variable. Among all generated values for a specific variable, the best value is taken. Figure 5 illustrates how the greedy heuristic finds the best value from the first column for the first variable.

4.1. Defining the objective function and the parameters

As mentioned before, the purpose is to locate some facilities on some nodes of the graph, in such a way that the maximum

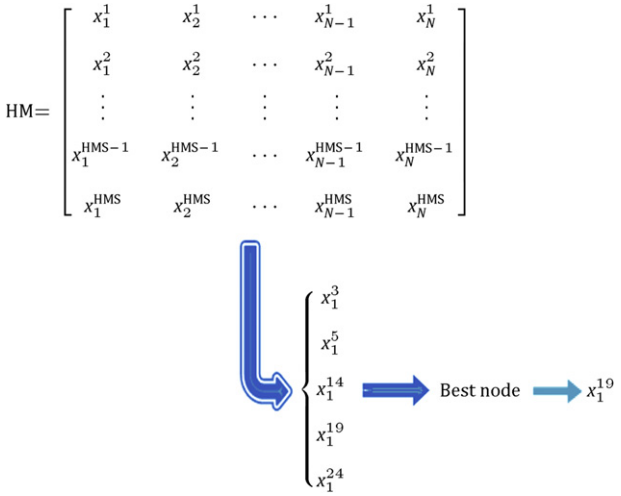

Figure 5: Greedy heuristic finds the best node in five randomly selected nodes.

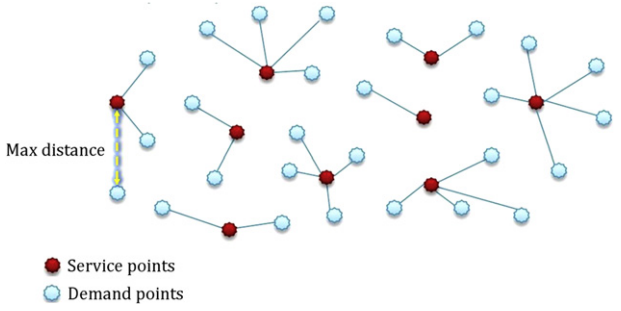

Figure 6: Sample demand and service points and the maximum distance.

distance between the customers and their nearest facilities becomes minimum. This is the discrete p-center problem, and the minimum distance between each two nodes is calculated by the Dijkstra algorithm. This algorithm was first suggested by Edsger Dijkstra in 1959, and is a graph search algorithm that finds the shortest path between each two nodes [27]. The objective function for the p-center problem is:

$$d(x, P_i) = \min(d(x_j, P_i)) : \quad x_j \in X, \tag{15}$$

$$f(x) = \max\{d(x, P_i)\} \quad 1 \le i \le m, \tag{16}$$

where P_i is node i of the graph, X is the set of selected centers and x represents the array of set X. Formula (15) finds the nearest facility or service point "x" among the set of centers x_j (set X) to demand point P_i, with Dijkstra's algorithm. Once the demand point specifies its nearest facility, it connects to it through the shortest path found by Dijkstra's algorithm. Among all the connections between demand points to service points, we try to make the longest connection minimum, by replacing the service points. Formula (16) calculates the maximum distance among all the connections (the number of demand points is m and the number of centers is p). Figure 6 shows the connections between all the demand points to their nearest facility and the maximum distance. Our purpose is to minimize $f(x)$ by replacing the centers of the graph.

The parameters are defined as:

$$\text{VarHMCR} = \text{MinHMCR} + (\text{MaxHMCR} - \text{MinHMCR}) \times \frac{j}{\text{MaxImp}}, \tag{17}$$

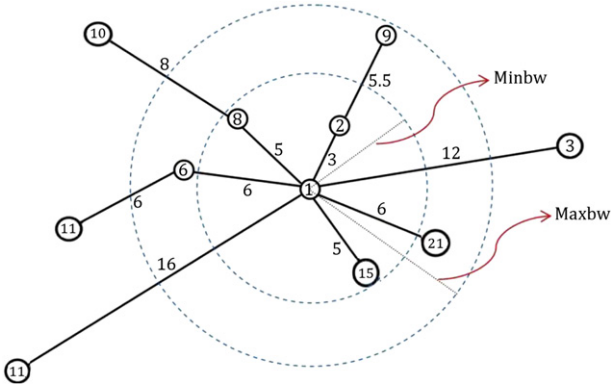

Figure 7: A sample node and the maximum and minimum bw.

$$\text{VarPAR} = \text{MinPAR} + (\text{MaxPAR} - \text{MinPAR}) \times \frac{j}{\text{MaxImp}}, \tag{18}$$

$$\text{Varbw} = \text{Maxbw} + (\text{Minbw} - \text{Maxbw}) \times \frac{j}{\text{MaxImp}}. \tag{19}$$

As mentioned before, the drawback of using Formula (14) for pitch adjusting the selected node to other nodes is that some nodes are far from the selected node and, if we move to the far node, it is most likely to change from the high quality solution to a bad one. For example, in Figure 2, if the selected node is node number 10, and $m = 2$, it can be substituted by node number 8 or 12, and it is obvious that node number 8 is far from node number 10. In order to conquer this problem, we substitute the node only with the neighbors that are at the predefined distance, bw (bandwidth). In other words, we changed the concept of bw in discrete problems. In Formula (14), the pitch adjusting procedure changes the number of nodes, while we have not considered the numbers of nodes and have defined a new concept of bw for discrete problems. This bw is the radius of the imaginary circle, the center of which is the selected node. In Figure 7, only neighbors that are in the pre-defined coverage distance (bw) are considered. Here, the maxbw is taken as 9 and the minbw is 5. As the algorithm progresses, the bw decreases to control the intensification more efficiently. Thus, at the beginning of the algorithm, the Varbw is 9 and the considered nodes are 2, 6, 8, 9, 15 and 21. At the end of the algorithm, only nodes with numbers 2, 8 and 15 can be substituted by node number 1. Therefore, we define MaxHMCR, MinHMCR, MaxPAR, MinPAR, Maxbw and Minbw, according to the type of problem. For example, if the number of the nodes is high and the number of centers that we wish to locate is low, MaxHMCR, MinHMCR, MaxPAR and MinPAR can be defined around 0.85, 0.7, 0.4 and 0.2, respectively, in order to control intensification and diversification. Maxbw and Minbw are defined considering the distances between the nodes.

4.2. Initializing harmony memory

The solutions in this problem are p-combinations of the set of n nodes, because the order of locations is not relevant and is denoted by $C(n, p)$. Consequently, to build the HM, we only need to produce a random p-combination of the nodes of the

Table 1: Results of solving unconditional p-center problem on OR LIB tests using CHS, GHS, VNS, TS and SS algorithms.

Test no.	n	p	Edges	Opt.	Best of CHS	CPU time for CHS	Best of GHS	CPU time for GHS	Best of VNS	CPU time for VNS	Best of TS	CPU time for TS	Best of SS	CPU time for SS
1	100	5	200	127	127	0.031	127	0.000	127	0.48	127	0.08	127	0.11
2	100	10	200	98	100	2.891	98	0.090	98	197.40	98	1.48	98	1.26
3	100	10	200	93	95	1.891	93	0.220	93	8.72	94	27.28	93	0.49
6	200	5	800	84	84	0.156	84	0.031	84	2.92	84	0.12	84	0.06
7	200	10	800	64	64	4.328	64	0.141	65	5.16	64	45.80	64	16.31
11	300	5	1 800	59	59	1.141	59	0.068	59	2.36	59	0.76	59	7.96
12	300	10	1 800	51	52	8.828	51	1.047	51	114.40	51	28.28	51	4.29
16	400	5	3 200	47	47	0.516	47	0.016	47	0.72	47	0.76	47	2.42
17	400	10	3 200	39	41	12.781	39	1.578	39	203.08	40	6.44	39	10.49
21	500	5	5 000	40	40	4.656	40	0.047	40	1.56	40	34.68	40	2.91
22	500	10	5 000	38	44	4.172	39	0.484	39	94.76	39	9.24	38	81.12
26	600	5	7 200	38	39	3.859	38	0.373	38	3.88	38	6.04	38	4.50
27	600	10	7 200	32	35	5.922	32	17.453	33	102.48	33	81.32	32	78.60
31	700	5	9 800	30	30	0.984	30	0.118	30	63.32	30	1.80	30	2.55
32	700	10	9 800	29	31	7.828	30	1.656	29	67.04	29	374.2	29	21.42
35	800	5	12 800	30	30	4.734	30	0.453	30	3.0	30	387.6	30	4.88
36	800	10	12 800	27	30	9.864	28	4.750	28	3.08	28	30.84	27	26.48
38	900	5	16 200	29	29	1.953	29	0.118	29	7.40	29	13.24	29	11.32
39	900	10	16 200	23	25	11.187	24	7.469	23	142.64	24	38.68	23	35.54

graph HMS times. The HM is produced as:

$$\mathrm{HM} = \begin{bmatrix} C^1(n,p) \\ C^2(n,p) \\ \vdots \\ C^{\mathrm{HMS}}(n,p) \end{bmatrix}. \tag{20}$$

Suppose there is a graph with ten nodes. We want to locate three centers on the nodes. A sample HM can be:

$$\mathrm{HM} = \begin{bmatrix} 3 & 4 & 6 \\ 9 & 1 & 7 \\ 4 & 2 & 8 \\ 10 & 3 & 5 \\ 7 & 6 & 3 \end{bmatrix}.$$

4.3. Improvising a new harmony

According to the three rules mentioned in Section 3, the new harmony is improvised. The MaxHMCR, MinHMCR, MaxPAR, MinPAR, Maxbw and Minbw are defined according to the type and conditions of the problem.

Once we initialize the HM (in previous section), we start to choose the first node for the new harmony. A random number is generated between 0 and 1. If this number is more than VarHMCR, one node of the graph is selected randomly, otherwise the algorithm selects one node from the first column of HM randomly. The first column is (3, 9, 4, 10, 7). This procedure is repeated for a specified number of times (greedy number) and the best of the selected nodes from the first column, or randomly selected nodes from all nodes (if the random number between 0 and 1 is more than VarHMCR), is selected as the first center in the graph. Updating the harmony memory and stopping criterion stay unchanged.

5. Computational results

5.1. Unconditional p-center problem

We tested our algorithm on two different test problems:

1. OR Library: Originally, Beasley proposed this set in 1985 for p-median problems; there are 40 tests with known optimal value. The range of the problems is from 100 to 900 nodes and the number of centers is from 5 to 200 [28]. In the case study, it is supposed that each year less than 10 stations can be established. Here, we solved tests with 5 or 10 numbers of centers with the modified harmony search. Table 1 shows the results of the Classic Harmony Search (CHS), the Greedy Harmony Search (GHS), the Variable Neighborhood Search (VNS), the Tabu Search (TS) and the Scatter Search based algorithm (SS) for solving the p-center problem on tests with $p \leq 10$ number of centers. In this table, comparison between CHS and GHS shows the improvement of the harmony search, when combined with the greedy heuristic. As discussed in Section 2, variable neighborhood and Tabu search algorithms were proposed by Mladenovic et al. [23] for solving the p-center problem for the first time, and the scatter search based approach was presented by Pacheco and Casad. The time and the values of the VNS, TS and SS are extracted from Pacheco and Casad's paper in which they coded these three algorithms in Pascal with a Delphi 5.0 compiler, ran them on a Pentium III and 600 MHz CPU, compared the results and claimed that their algorithm (SS) yielded better results.
2. TSP Library: It includes a wide range of tests. Originally used for Traveling Salesman Problems (TSP), they are also used for testing median and center problems [29]. Here, we tested our algorithm to find 5, 10, 15 and 20 centers on four tests with large numbers of nodes and Euclidean distances: u724, vm1748, Pcb3038, fnl4461, with 724, 1748, 3038 and 4461 numbers of nodes, respectively. Table 2 shows the results of these test problems.

The modified harmony search for the p-center problem is coded in Microsoft Visual C++ and run on an AMD II X2 250 processor, with 2.99 GHz CPU and 2 GB of RAM. In 15 out of 19 tests of OR LIB problems, the modified harmony search finds the optimal solution. In the 4 other tests, the gap of the obtained solution is very small. Harmony search is a fast algorithm and, in many tests, leads to the optimal solution in less than half a second. The CPU time is in seconds.

5.2. Conditional p-center problem

In a conditional p-center problem, we want to locate p facilities on a network in which q facilities have already been

Table 2: Results of solving unconditional p-center problem on TSP test problems with Euclidean distances using Greedy Harmony Search (GHS).

Test name	n	p	GHS value	CPU time (s) for GHS
u724.tsp	724	5	707	2.110
		10	470	5.825
		15	377	18.984
		20	338	23.125
vm1748.tsp	1748	5	4878	4.172
		10	3323	11.824
		15	2765	24.234
		20	2371	42.110
Pcb3038.tsp	3038	5	1098	6.541
		10	786	25.906
		15	609	42.862
		20	558	70.284
fnl4461.tsp	4461	5	1116	7.437
		10	753	18.565
		15	622	34.532
		20	546	57.875

located. That means we need a harmony memory that has the same arrays (located facilities) in all rows for the first $|Q| = q$ columns. We suppose $N = \{n_1, \ldots, n_n\}$ is the set of nodes of the graph ($|N| = n$). $Q = \{a_1, \ldots, a_q\}$ is the set of nodes in which facilities have already been located on ($|Q| = q$), and $M = N - Q = \{m_1, m_2, \ldots, m_{m=n-q}\}$ is the set of free nodes of the graph in which new facilities can be established on ($|M| = n - q$). The first studies on conditional location problems are due to [30,31]. Berman and Simchi-Levi suggested to solve conditional p-median and p-center problems on a network by an algorithm that requires one time solution of an unconditional $(p + 1)$-median or $(p + 1)$-center problem [32]. Berman and Drezner suggested a simple algorithm that solves conditional location problems on a graph [33]. Chen and Chen presented an algorithm [34] for solving the conditional p-center problem that relies on Drezner's observations [33], as well as a relaxation-based algorithm [15]. The solutions in this problem are p-combinations of the set of M with $n-q$ nodes (free nodes), because the order of locations is not relevant and is denoted by $C(M, p)$. Consequently, to build the HM, we only need to produce a random p-combination of the nodes of the graph HMS times. For conditional p-median and p-center problems, the HM has the following general form:

$$\mathrm{HM} = \begin{bmatrix} a_1^1 & a_2^1 & \cdots & a_q^1 & C^1(M,p) \\ a_1^2 & a_2^2 & \cdots & a_q^2 & C^2(M,p) \\ & & & \vdots & \\ a_1^{\mathrm{HMS}} & a_2^{\mathrm{HMS}} & \cdots & a_q^{\mathrm{HMS}} & C^{\mathrm{HMS}}(M,p) \end{bmatrix}. \tag{21}$$

$C^i(M, p)$ is the p-combination of set M, which means selecting p random nodes out of m nodes of set M for row number i. Since q facilities have already been located, we have to figure out the p-combination of the free nodes (M).

For solving this problem, we assume that the first ten nodes of the network are the existing facilities ($q = 10$), p new facilities need to be located and HMS is assumed to be 10. With this assumption, the HM is calculated as follows:

$$\mathrm{HM} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & C^1(M,p) \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & C^2(M,p) \\ & & & & & & \vdots & & & & \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & C^{10}(M,p) \end{bmatrix}. \tag{22}$$

Table 3: Results of solving conditional p-center problem on OR LIB tests using Greedy Harmony Search (GHS).

Test no.	n	p	Edges	Fitness value of GHS	CPU time (s) for GHS
1	100	5	200	111	0.000
2	100	10	200	88	0.078
3	100	10	200	92	0.200
6	200	5	800	73	0.031
7	200	10	800	58	0.156
11	300	5	1 800	53	0.140
12	300	10	1 800	49	0.692
16	400	5	3 200	47	0.044
17	400	10	3 200	39	0.197
21	500	5	5 000	36	0.110
22	500	10	5 000	35	2.265
26	600	5	7 200	37	0.140
27	600	10	7 200	31	2.875
31	700	5	9 800	29	0.282
32	700	10	9 800	29	3.795
35	800	5	12 800	29	2.263
36	800	10	12 800	27	4.125
38	900	5	16 200	28	0.656
39	900	10	16 200	24	0.703

Table 3 shows the results of solving a conditional p-center problem on OR-Library test problems when $p \leq 10$.

The metaheuristic algorithm developed in this paper can also be applied to p-median problems [35] and used for other purposes [36].

6. Results of bicycle stations in the city of Isfahan

In this section, we collected the real data for locating bicycle stations in Isfahan city. In 2009, the government planned to establish around ten bicycle stations each year. Now, in 2010, they succeeded in locating eight stations in the city. Here, first we want to find the optimum locations for establishing the first eight stations, and since the purpose is to minimize the coverage distance as much as possible, we face a center problem. Since at the first stage no stations have been located, it is an unconditional p-center problem. At the second stage we want to find ten other locations for ten other stations, and we have a conditional p-center problem. Figure 8 shows a map of Isfahan city in which we want to locate the stations.

We obtained this map from Google, and we have identified the central part with a red line. The bicycle stations are set in the central part of the city and we assumed there is a graph in which the intersections are the nodes of the graph, and the streets, lanes and alleys are the edges of the graph. This graph has 163 nodes and the unit of distance between the nodes is 10 m. Thus, when we say the distance between two nodes is 50, it means the original distance is 500 m. In this way, the distance matrix and then the shortest distance between each pair of nodes are calculated by the Dijkstra algorithm. As mentioned, at the first stage, we want to locate eight stations in the first year. To test our model, we located two, four and eight stations. Figure 9 illustrates the location of two stations. The objective function value for two stations is 369. In this figure, the maximum distance between the customers and stations is 3690 m.

Figure 10 illustrates the location of four stations obtained by the model. The objective function value for four stations is 243. In Figure 10, the maximum distance between the customers and stations is 2430 m. Figure 11 shows the location of eight stations obtained by the model. The objective function value for eight

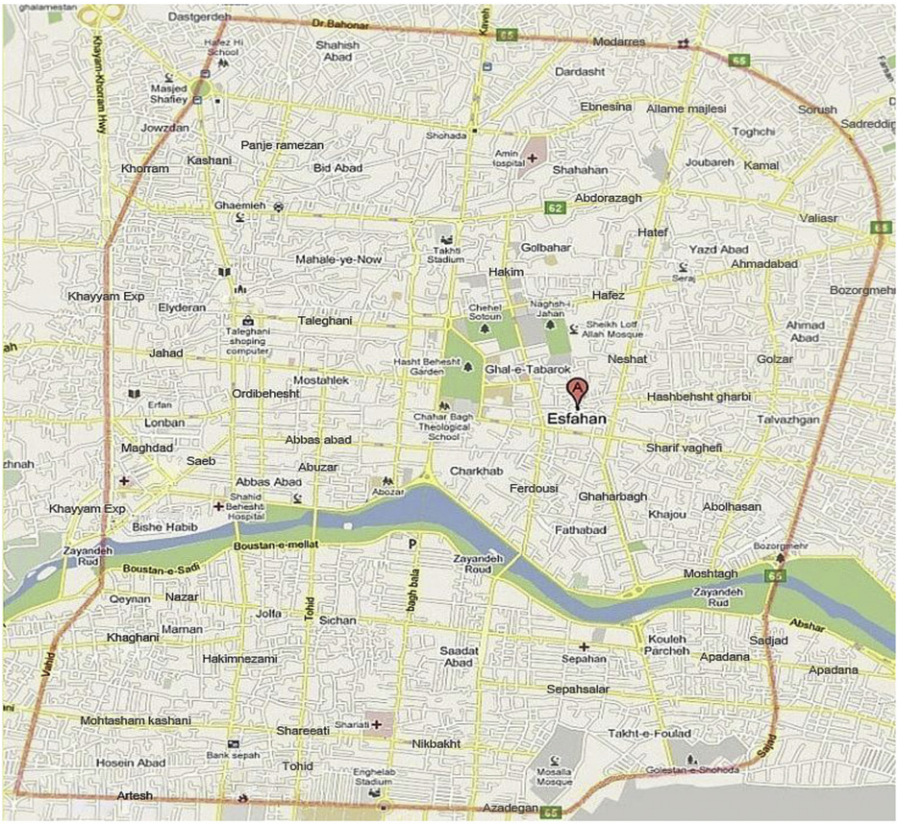

Figure 8: Map of Isfahan city.

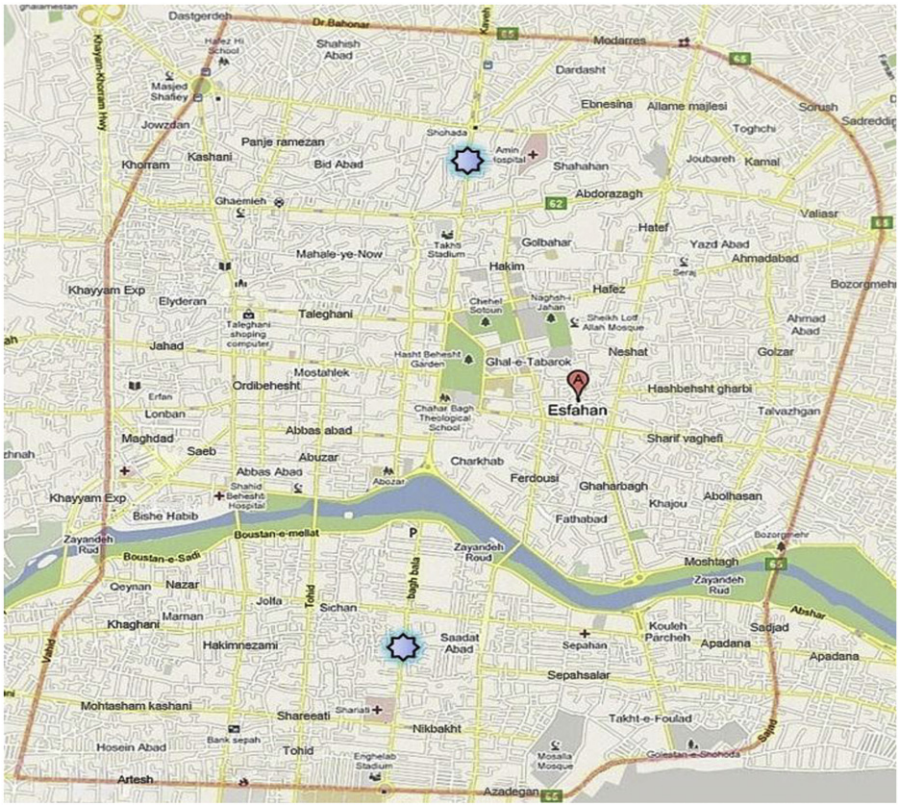

Figure 9: Locating two stations (unconditional problem).

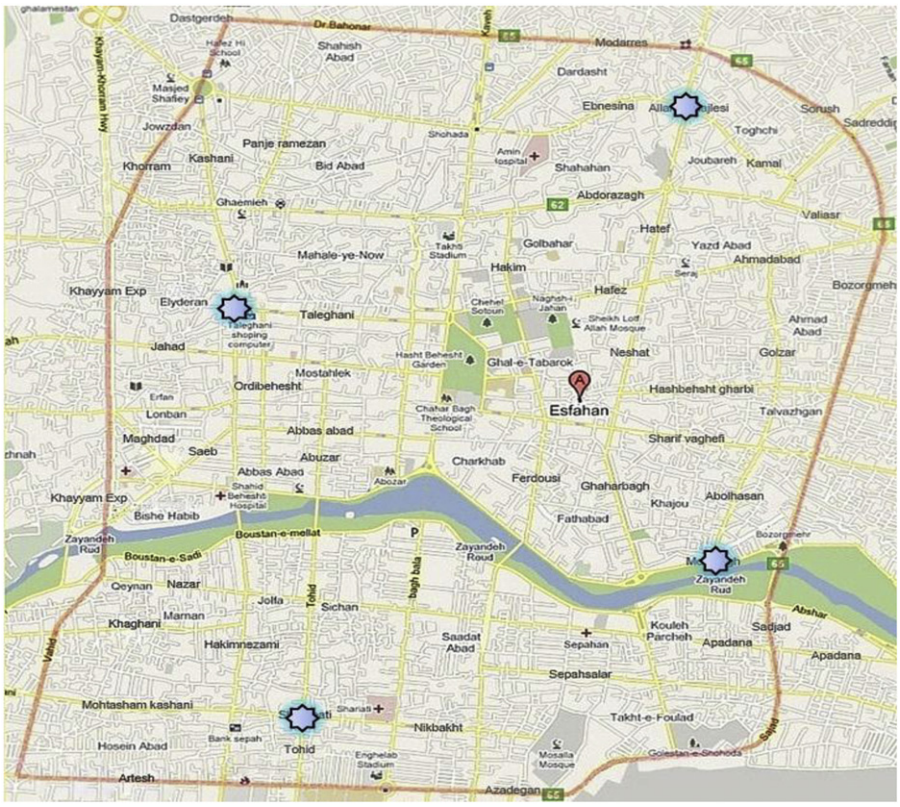

Figure 10: Locating four stations (unconditional problem).

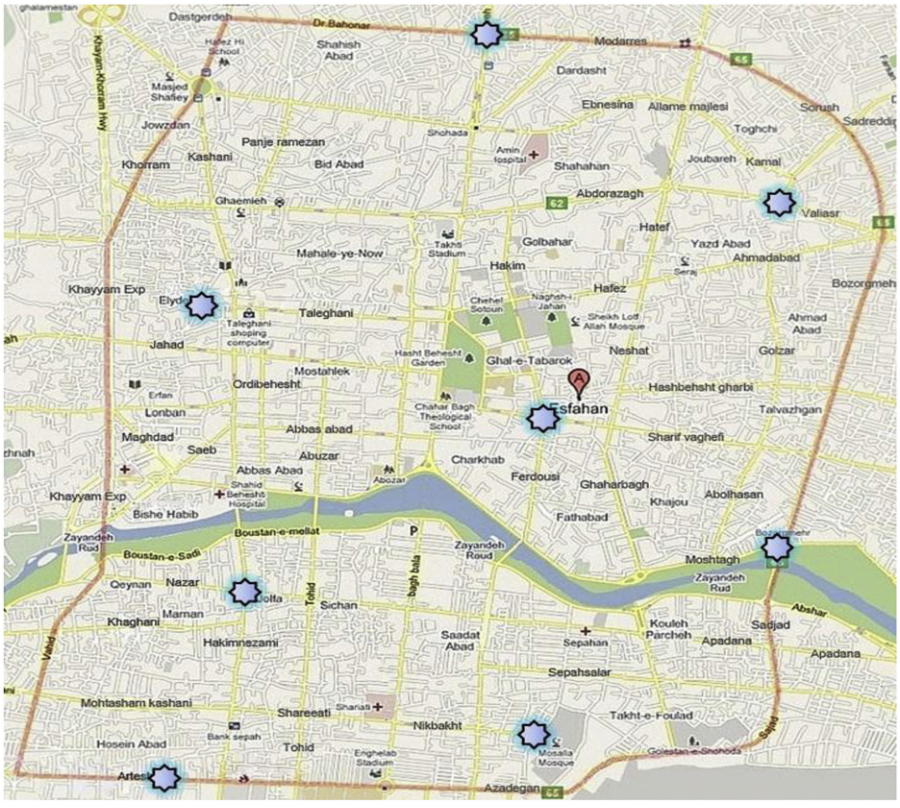

Figure 11: Locating eight stations (unconditional problem).

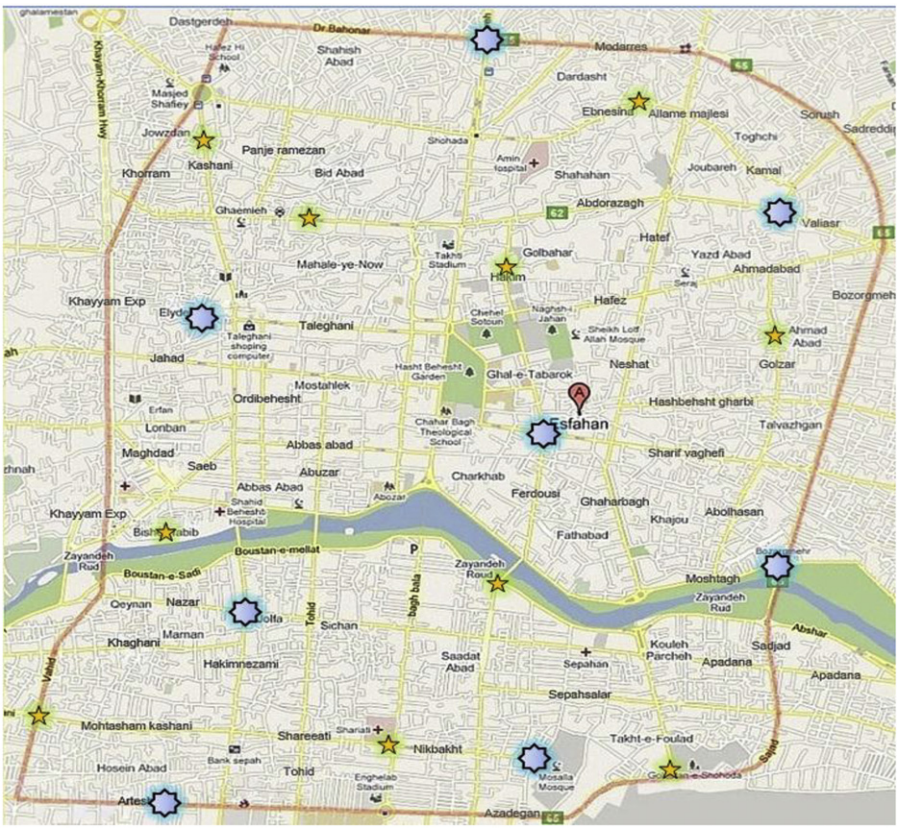

Figure 12: Locating ten other stations for next year (conditional problem).

stations is 171. In Figure 11 the maximum distance between the customers and stations is 1710 m.

After locating eight stations in the city, we need to locate ten other stations for the following year. Consequently, these eight located stations are fixed in the city. Thus, locating ten other stations for the following year is a conditional *p*-center problem. After solving the problem, totally, eighteen stations are located in the city. Figure 12 shows how ten other stations are located for the following year by the model. The objective function value for eighteen stations is 121. In Figure 12 the maximum distance between the customers and stations is 1210 m. The sign, " ⬟ ", indicates those stations that have already been located, and the sign, " ★ ", indicates ten new stations that are supposed to be located in the subsequent year.

7. Conclusion

In this paper, a modified harmony search algorithm is utilized for solving conditional and unconditional *p*-center problems. The pre-defined parameters for a harmony search are altered, and a variable HMCR, variable PAR and variable bandwidth are utilized. We used our algorithm to solve the conditional and unconditional discrete *p*-center problem. This new version of the harmony search is found to be an efficient and robust algorithm. Results of solving the *p*-center problem on OR Library instances, with both classic and greedy harmony searches, showed that the modified version is much faster and more efficient. We also presented the results of other authors who solved the *p*-center problem by using other meta-heuristic algorithms, including VNS, Tabu search and scatter search. Finally, we used this model to locate bicycle stations in the historic city of Isfahan.

Acknowledgment

The first author is grateful to the Iran National Science Foundation for its support.

References

[1] Hakimi, S.L. "Optimum locations of switching centers and the absolute centers and medians of a graph", *Journal of the Operational Research*, 12, pp. 450–459 (1964).
[2] Hakimi, S.L. "Optimum distribution of switching centers in a communications network and some related graph theoretic problems", *Journal of the Operational Research*, 13, pp. 462–475 (1965).
[3] Geem, Z.W., Kim, J.H. and Loganathan, G.V. "A new heuristic optimization algorithm: harmony search", *Simulation*, 76(2), pp. 60–68 (2001).
[4] Geem, Z., *Music-Inspired Harmony Search Algorithm*, Springer (2009).
[5] Kaveh, A. and Talatahari, S. "An efficient hybrid algorithm based on harmony search, particle swarm and ant colony strategies for optimal design of structures", In *Harmony Search Algorithms for Structural Design*, Z.W. Geem, Ed., Springer-Verlag (Chapter 5 of a Book Entitled) (2009).
[6] Nadi, F., Tajudin Khader, A. and Azmi Al-Betar, M. "Adaptive genetic algorithm using harmony search", In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO'10, pp. 819–820, ACM, New York, NY, USA (2010).
[7] Minieka, E. "The *m*-center problem", *SIAM Review*, 12, pp. 139–140 (1970).
[8] Zanjirani Farahani, R. and Hekmatfar, M. "Facility location concepts, models, algorithms and case studies", In *Contributions to Management Science*, In *Physica*, Springer, Heidelberg, London (2009).
[9] Kariv, O. and Hakimi, S.L. "An algorithmic approach to network location problems. Part 1: the *p*-centers problem", *SIAM Journal of Applied Mathematics*, 37, pp. 513–538 (1979).
[10] Ilhan, T., Özsoy, F.A. and Pinar, M.C. "An efficient exact algorithm for the vertex *p*-center problem and computational experiments for different set covering subproblems", Bilkent University, Department of Industrial Engineering, Technical Report, 2002. Available at URL: http://www.optimization-online.org/DB_HTML/2002/12/588.html.
[11] Elloumi, S., Labbe, M. and Pochet, Y. "A new formulation and resolution method for the *p*-center problem", *INFORMS Journal of Computing*, 16, pp. 84–94 (2004).

[12] Handler, G.Y. and Mirchandani, P.B., *Location on Networks: Theory and Algorithms*, MIT Press, MA Cambridge (1979).
[13] Daskin, M.S. "A new approach to solving the vertex *p*-center problem to optimality: algorithm and computational results", *Communications of the Operations Research Society of Japan*, 45(9), pp. 428–436 (2000).
[14] Watson-Gandy, C.D.T. "The multi-facility min–max Weber problem", *European Journal of Operational Research*, 18, pp. 44–50 (1984).
[15] Chen, D. and Chen, R. "New relaxation-based algorithms for the optimal solution of the continuous and discrete *p*-center problems", *Computers & Operations Research*, 36, pp. 1646–1655 (2009).
[16] Chen, R. "Solution of minisum and minimax location–allocation problems with Euclidean distances", *Naval Research Logistics Quarterly*, 30, pp. 449–459 (1983).
[17] Drezner, Z. "The *p*-center problem -heuristic and optimal algorithms", *Journal of the Operational Research*, 8, pp. 741–748 (1984).
[18] Drezner, Z. "The planar two center and two median problems", *Transportation Science*, 18, pp. 351–361 (1984).
[19] Toregas, C., Swain, R., ReVelle, C. and Bergmann, L. "The location of emergency service facilities", *Journal of the Operational Research*, 19, pp. 1363–1373 (1971).
[20] Hwang, R.Z., Lee, R.C.T. and Cheng, R.C. "The slab dividing approach to solve the Euclidean *p*-center problem", *Algorithmica*, 9, pp. 1–22 (1993).
[21] Suzuki, A. and Drezner, Z. "The *p*-center location problem in the area", *Location Science*, 4, pp. 69–82 (1996).
[22] Caruso, C., Colorni, A. and Aloi, L. "Dominant, an algorithm for the *p*-center problem", *European Journal of Operational Research*, 149, pp. 53–64 (2003).
[23] Mladenovic, N., Labbe, M. and Hansen, P. "Solving the *p*-center problem with Tabu search and variable neighborhood search", *Networks*, 42, pp. 48–64 (2003).
[24] Pacheco, J.A. and Casado, S. "Solving two location models with few facilities by using a hybrid heuristic: a real health resources case", *Computers & Operations Research*, 32, pp. 3075–3091 (2005).
[25] Kaveh, A. and Nasr, H. Greedy harmony search for solving *p*-median problems, Applied Soft Computing (2011) (submitted for publication).
[26] Mahdavi, M., Fesanghary, M. and Damangir, E. "An improved harmony search algorithm for solving optimization problems", *Applied Mathematics and Computation*, 188, pp. 1567–1579 (2007).
[27] Dijkstra, E.W. "A note on two problems in connexion with graphs", *Numerische Mathematik*, 1, pp. 269–271 (1959).
[28] Beasley, J.E. "A note on solving large *p*-median problems", *European Journal of Operational Research*, 21, pp. 270–273 (1985).
[29] Reinelt, G. "TSPLIB—a traveling salesman problem library", *ORSA Journal on Computing*, 3(4), pp. 376–384 (1991).
[30] Handler, Y.G. and Mirchandani, P.B., *Location on Networks Theory and Algorithms*, MIT Press, Cambridge, MA (1979).
[31] Lin, C.C. "A note about the new emergency facility insertion in an undirected connected graph", In *The Sixth Annual Pittsburgh Conference on Modelling Simulation*, Penn, Pittsburgh (1975).
[32] Beramn, O. and Simchi-Levi, D. "The conditional location problem on networks", *Transportation Science*, 24, pp. 77–78 (1990).
[33] Beramn, O. and Drezner, Z. "A new formulation for the conditional *p*-median and *p*-center problems", *Operations Research Letters*, 36, pp. 481–483 (2008).
[34] Chen, D. and Chen, R. "A relaxation-based algorithm for solving the conditional *p*-center problem", *Operations Research Letters*, 38, pp. 215–217 (2010).
[35] Kaveh, A., Shahrouzi, M. and Naserifar, Y. "Tuned genetic algorithms for finding *p*-median of a weighted graph", *Scientia Iranica. Transaction A: Civil Engineering*, 17(5), pp. 350–362 (2010).
[36] Kaveh, A. and Shojaee, S. "Optimal domain decomposition via *p*-median methodology using ACO and hybrid ACGA", *Finite Elements in Analysis and Design*, 44(8), pp. 505–512 (2008).

Ali Kaveh was born in 1948 in Tabriz, Iran. After graduation from the Department of Civil Engineering at the University of Tabriz in 1969, he continued his studies on Structures at Imperial College of Science and Technology at London University, and received his M.S., DIC and Ph.D. degrees in 1970 and 1974, respectively. He then joined the Iran University of Science and Technology in Tehran where he is presently Professor of Structural Engineering. Professor Kaveh is author of 260 papers published in international journals and 125 papers presented at international conferences. He has authored 23 books in Farsi and 3 books in English published by Wiley, the American Mechanical Society and Research Studies Press.

Hamed Nasr Esfahani was born in 1983 in Isfahan. He obtained his B.S. degree in the field of Industrial Engineering from Iran University of Science and Technology in 2006, and his M.S. degree in the field of Civil Engineering-Construction Management, from the same university in 2010. At present, he is working with the ISFA corporation.