1. Introduction

Conditional location problem consists of finding $p$ new facilities to serve a set of demand points, with $q$ facilities already being available. When $q = 0$, the problem becomes unconditional. In the conditional $p$-median problems, once the new $p$ locations are located, a demand can be served either by one of the existing facility or by one of the new facilities which ever is the nearest facility to the demand [1].

Every application to the $p$-median problems becomes a conditional model when already there exist some facilities in the area under study. As an example, if one wants to locate $p$ warehouses in an area, it is an unconditional $p$-median problem. However, when $q$ warehouses already exist in the area and we need to add $p$ new warehouses, it becomes a conditional $p$-median problem.

The first studies on conditional location problems are due to [2,3]. In those references, the studied conditional problem was a conditional 1-center problem ($p=1$, and $q \geq 1$). The conditional location problem with $p \geq 1$ new facilities has been studied in two papers by Chen [4,5]. The former is about conditional $p$-median and $p$-center problem and the latter is on conditional $p$-center problems. Drezner solves the conditional $p$-center problem by an algorithm that requires the solution of $O(\log n)$ unconditional $p$-center problems ($n$ being the number of demand nodes) [6]. A heuristic for the conditional $p$-median problem requiring solution of several unconditional $p$-median problems is discussed in [7]. The method proposed by Drezner is applicable to both planar and network configurations. Berman and Simchi-Levi suggested to solve the 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 [8]. Chen and Handler presented a relaxation algorithm for solution of the conditional $p$-center problem [5]. Chen and Chen presented an algorithm [9] for solving the conditional $p$-center problem, which relies on Drezner's observations [1], as well as a relaxation-based algorithm developed by Chen and Chen [10]. This new relaxation algorithm is applicable to both the continuous Euclidean and the discrete conditional $p$-center problems.

Berman and Drezner suggested a simple algorithm which solves both the conditional $p$-median and $p$-center problems on a network [1]. The algorithm requires one-time solution of an unconditional $p$-median or $p$-center problem using a shortest distance matrix. The conditional $p$-median problems were solved by CPLEX program providing superior results compared to those obtained by the method presented by Berman and Simchi-Levi.

In this article the conditional location problem is studied. Conditional location problems have a wide range of applications in location science. A new meta-heuristic algorithm for solving conditional $p$-median problems is proposed and results are compared to those of the previous studies. This algorithm produces much better results than the previous formulations.

2. Formulation of the problem

Let $G = (N, L)$ be a network with $N$ being the set of nodes, $|N|=n$ and $L$ being the set of links. Consider a non-negative number $w_i$ that represents the demand weight at node $i \in N$. Let
\(d_{xy}\) be the shortest distance between any two nodes \(x, y \in G\). Suppose that there is a set \(Q (|Q| = q)\) of existing facilities. Let \(X = (X_1, \ldots, X_p)\) and \(Y = (Y_1, \ldots, Y_q)\) be vectors of size \(q\) and \(p\), respectively, where \(Y_i\) is the location of the existing facility \(i\) and \(X_j\) is the location of new facility \(j\). The conditional \(p\)-median location problem can now be expressed as minimizing

\[
f(x) = \sum_{i=1}^{m} w_i \min[d(X,i),d(Y,i)]
\]  

(1)

Where \(d(X,i)\) (or \(d(Y,i)\)) is the shortest distance from the closest facility in \(X\) (or \(Y\)) to node \(i\) [1].

3. The algorithm of Berman and Simchi-Levi

The idea is to produce a new potential location representing all the existing facilities. If a demand point is utilizing the services of an existing facility, it will use the services of the closest existing facility. Thus, the distance between a demand point and the new location is the minimum distance among all the existing facilities. In order to force the formation of a facility at the new location, a new demand point is considered with a distance of zero from the new potential location and a large distance from all the other potential locations. The new distance matrix, denoted by \(D\), is constructed by adding a new location \(a_0\) (a new column) to \(D\) which represents the \(Q\) existing locations and a new demand point \(v_0\) with an arbitrary positive weight. For each regular demand point (node) \(i\), we have \(d(i,a_0) = \min_{k \in Q} \{d_{ik}\}\) and \(d(v_0,a_0) = 0\). For each regular potential location node \(j\), \(d(v_0,j) = M\), where \(M\) is a large number. Again the nodes in \(Q\) and in the potential locations \(Q\) are removed. The program is tackled by solving the unconditional \(p + 1\) median (or center) problem using \(D\) [8].

4. The algorithm of Drezner

Drezner solved the conditional problems by defining a modified shortest distance matrix \(D'\) as

\[
D'_{ij} = \min\{d_{ij}, \min_{a_0 \in Q} \{d_{ia_0}\}\} \quad \forall i, j, p, n
\]  

(2)

It should be noted that \(D'\) is not symmetric even when \(D\) is symmetric. The unconditional \(p\)-median or \(p\)-center problem using the appropriate \(D'\) solves the conditional \(p\)-median or \(p\)-center problem. This is true since if the shortest distance from node \(i\) to the new \(p\) facilities is larger than \(\min_{k \in Q} \{d_{ik}\}\) then the shortest distance to the existing \(q\) facilities is utilized [1].

5. Hybrid harmony search algorithm

In this paper we use a new meta-heuristic algorithm which is inspired by music, harmony search algorithm [11]. Kaveh and Nasr combined the modified harmony search algorithm with the greedy heuristic and obtained a robust algorithm called Greedy Harmony Search (GHS) [12]. It is discussed there that with a constant Harmony Memory Considering Rate (HMCR) there is a problem with diversification if the number of nodes in Harmony Memory (HM) is a small fraction of total nodes.

That means if we choose HMCR around 0.95 to 0.99, the search space will totally be limited to the nodes of the HM matrix and the algorithm gets trapped in local optimum or at least the convergence rate decreases. On the other hand, if we choose the HMCR parameter around 0.4 to 0.7, this will be a good strategy at the beginning because the diversification of the search remains high at this stage and the algorithm searches the entire search space. However, as the algorithm gets through, the diversification should be decreased and the low HMCR makes the algorithm not to converge fast and damages the intensification. In order to conquer this problem, variable HMCR is introduced (similarly, Mahdavi et al., defined variable Pitch Adjusting Rate (PAR) and Bandwidth (bw) [13]). To combine harmony search with the greedy heuristic, we generate several random numbers in interval [0 1] for a pre-defined number of times (t) and compare those random numbers with variable HMCR (VarHMCR, will be defined in the next section) t 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). Finally we select the best of them according to their fitness values. Here we use this algorithm to solve the conditional \(p\)-median problem. Figure 1 illustrates how the greedy heuristic finds the best value from the first column for the first variable.

\[
f(x) = \sum_{i=1}^{m} w_i d(x,P_i)
\]  

(3)

\[
d(x,P) = \min(d(x,j),P)) \quad : \quad x_j \in X
\]  

(4)

A. Kaveh, H. Nasr
Where \( P_i \) is the node \( i \) of the graph and \( X \) is the set of selected medians and \( x \) represents the array of the set \( X \). Formula 3 finds the nearest facility or service point "\( x \)" among the set of the medians, \( x_j \) (set \( X \)) to the demand point \( p_i \) with Dijkstra's algorithm. Once the demand point specifies its nearest facility, a connection will be produced through the shortest path we found by Dijkstra's algorithm. Formula 4 calculates the sum of distances to their nearest service point \( D(W_j) \) for all demand points (the number of demand points is \( m \) and the number of medians is \( p \)).

The parameters are defined variable as

\[
\begin{align*}
\text{VarHMCR} &= \text{MinHMCR} + (\text{MaxHMCR} - \text{MinHMCR}) \times \frac{j}{\text{MaxImp}} \\
\text{VarPAR} &= \text{MinPAR} + (\text{MaxPAR} - \text{MinPAR}) \times \frac{j}{\text{MaxImp}} \\
\text{Varbw} &= \text{Maxbw} + (\text{Minbw} - \text{Maxbw}) \times \frac{j}{\text{MaxImp}}
\end{align*}
\]

where \( \text{MaxImp} \) denotes the number of iterations, and \( j \) is the number of current iteration. In Figure 2, only neighbors that are in the pre-defined coverage distance (bandwidth) are considered. Here, the maxbw is taken as 9 and the minbw is 5. 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 the considered nodes with numbers 2, 8 and 15 can only be substituted by node number 1. If the number of the nodes is high and the number of medians we want to locate is low, MaxHMCR, MinHMCR, MaxPAR and MinPAR can be defined around 0.85, 0.7, 0.4 and 0.2 respectively, to control the intensification and diversification essentially. Maxbw and Minbw is defined considering the distances between the nodes.

\[
\begin{align*}
\text{Maxbw} &= C_{\text{bw}}(\text{Varbw}^{\text{bw}}, 0) \\
\text{Minbw} &= C_{\text{bw}}(\text{Varbw}^{\text{bw}}, 0)
\end{align*}
\]

\[
\text{MaxHMCR} = \frac{(\text{Maxbw} + \text{Minbw})}{2}
\]

\[
\text{MinHMCR} = \frac{(\text{Maxbw} - \text{Minbw})}{2}
\]

\[
\text{MaxPAR} = \frac{(\text{Maxbw} + \text{Minbw})}{2}
\]

\[
\text{MinPAR} = \frac{(\text{Maxbw} - \text{Minbw})}{2}
\]

Therefore, we define \( \text{MaxHMCR}, \text{MinHMCR}, \text{MaxPAR}, \text{MinPAR}, \text{Maxbw} \) and \( \text{Minbw} \) first.

\[
\text{VarHMCR} = \text{MinHMCR} + (\text{MaxHMCR} - \text{MinHMCR}) \times \frac{j}{\text{MaxImp}}
\]

\[
\text{VarPAR} = \text{MinPAR} + (\text{MaxPAR} - \text{MinPAR}) \times \frac{j}{\text{MaxImp}}
\]

\[
\text{Varbw} = \text{Maxbw} + (\text{Minbw} - \text{Maxbw}) \times \frac{j}{\text{MaxImp}}
\]

\[\text{MaxImp} = \text{Max} - \text{Min} \]

\[\text{VarHMCR} = \text{MinHMCR} + \left( \text{MaxHMCR} - \text{MinHMCR} \right) \times \frac{j}{\text{MaxImp}} \]

\[\text{VarPAR} = \text{MinPAR} + \left( \text{MaxPAR} - \text{MinPAR} \right) \times \frac{j}{\text{MaxImp}} \]

\[\text{Varbw} = \text{Maxbw} + \left( \text{Minbw} - \text{Maxbw} \right) \times \frac{j}{\text{MaxImp}} \]

5.2. Initialization of the harmony memory

In conditional \( p \)-median problems we want to locate \( p \) facilities on the network in which \( q \) facilities have already been located. That means we need a harmony memory that have the same arrays (located facilities) in all rows. If we suppose \( N=\{n_1,...,n_n\} \) is the set of nodes of the graph \((|N|=n)\), \( Q=\{a_1,...,a_q\} \) is the set of nodes in which facilities have already been located \((|Q|=q)\) and \( M=\text{ } N - Q=\{n_1, n_2,...,n_{\text{new}}\} \) is the set of free nodes of graph in which new facilities can be established \((|M|=n-q)\). The solutions in this problem are \( p \)-combinations of the set \( M \) 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 \( HM \) times. For conditional \( p \)-median problems the \( HM \) will have the following general form:

\[
\text{HM} = \left[ \begin{array}{ccc}
\text{a}_1^1 & \text{a}_1^2 & \cdots & \text{a}_1^q \\
\text{a}_2^1 & \text{a}_2^2 & \cdots & \text{a}_2^q \\
\vdots & \vdots & \ddots & \vdots \\
\text{a}_q^1 & \text{a}_q^2 & \cdots & \text{a}_q^q
\end{array} \right]
\]

5.3. Improvising a new harmony

Similar to \( HM \), for improvising a new harmony (\( h\text{new} \)), we should first mention the facilities that have already been located. Thus the \( h\text{new} \) has the following form:

\[h\text{new} = \{a_1, a_2,..., a_q, b_1, b_2,..., b_p\}\]

Again \( Q=\{a_1,...,a_q\} \) is the set of located facilities. The algorithm tries to find \( p \) medians for \( h\text{new} \) after \( a_q \) (for all \( h\text{news} \{a_1, a_2,...,a_q\}\) comes first). Thus to improvise a new harmony, it is enough to improvise a harmony with \( p \) arrays out of set \( M \), so the length of \( h\text{new} \) is \( p+q \). The medians after \( a_q \) are determined based on the following three rules:

1. Memory consideration (considering \( \text{VarHMCR} \)),
2. Pitch adjustment (considering \( \text{VarPAR} \)), and
3. Random selection (considering \( \text{VarHMCR} \)).

5.4. Updating the harmony memory

If the fitness function value of \( h\text{new} = \{a_1, a_2,...,a_q, b_1, b_2,...,b_p\} \) is better than the worst fitness function value of harmonies in the \( HM \), the new harmony is included in the \( HM \) and the existing worst harmony is excluded from the \( HM \).

5.5. Checking the stopping criterion

If the stopping criterion (maximum number of improvisations or unchanging the best objective function value for a predefined number of iterations) is satisfied, computation is terminated. Otherwise, Steps 3 and 4 are repeated.

6. Computational results

We applied our method to the OR-Library tests proposed by Beasley [15]. Similar to Drezner’s article we assume that the first ten nodes of the given network are existing facilities.
For test numbers 35 to 40, Berman and Drezner could not solve the problem due to the lack of memory. We solved them but there is no guarantee that the obtained values are the optimal ones. Although the best results obtained by the greedy harmony search for tests with n = 800 and p ≤ 10 (test numbers 35, 36, 38, 39) are very likely to be the optimal values. Table 1 shows the computational results. We have taken the CPU time for drezner’s algorithm and Berman and Simchi’s algorithm from Drezner’s paper (in seconds). From this table, it is obvious that when the number of medians is not more than a hundred, harmony search is much better to use and when the number of medians increases over one hundred, other two algorithms are better to employ. Best result in each test is made bold in the table.

### 7. Concluding Remarks

In this paper a brief description is provided for the algorithms by Berman, Drezner and Berman and Simchi, and

---

**Table 1. Computational results**

<table>
<thead>
<tr>
<th>Test No.</th>
<th>N</th>
<th>P</th>
<th>Optimum</th>
<th>Berman, Simchi’s algorithm value</th>
<th>CPU time for Berman Simchi’s algorithm</th>
<th>CPU time for drezner’s algorithm</th>
<th>GHS value</th>
<th>Error % of GHS</th>
<th>CPU time for GHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>5</td>
<td>4369</td>
<td>4369</td>
<td>1.12</td>
<td>1.64</td>
<td>4369</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>10</td>
<td>3087</td>
<td>3087</td>
<td>1.12</td>
<td>1.73</td>
<td>3087</td>
<td>0</td>
<td>0.015</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>10</td>
<td>3417</td>
<td>3417</td>
<td>1.51</td>
<td>2.31</td>
<td>3417</td>
<td>0</td>
<td>0.031</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>20</td>
<td>2392</td>
<td>2392</td>
<td>0.87</td>
<td>1.28</td>
<td>2392</td>
<td>0</td>
<td>0.047</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>33</td>
<td>916</td>
<td>916</td>
<td>0.90</td>
<td>1.26</td>
<td>916</td>
<td>0</td>
<td>0.218</td>
</tr>
<tr>
<td>6</td>
<td>200</td>
<td>5</td>
<td>6153</td>
<td>6153</td>
<td>71.18</td>
<td>6153</td>
<td>6153</td>
<td>0</td>
<td>0.015</td>
</tr>
<tr>
<td>7</td>
<td>200</td>
<td>10</td>
<td>4778</td>
<td>4778</td>
<td>5.42</td>
<td>4778</td>
<td>4778</td>
<td>0</td>
<td>0.078</td>
</tr>
<tr>
<td>8</td>
<td>200</td>
<td>20</td>
<td>3877</td>
<td>3877</td>
<td>4.50</td>
<td>3877</td>
<td>3877</td>
<td>0</td>
<td>0.250</td>
</tr>
<tr>
<td>9</td>
<td>200</td>
<td>67</td>
<td>1086</td>
<td>1086</td>
<td>4.02</td>
<td>1086</td>
<td>1086</td>
<td>0</td>
<td>3.525</td>
</tr>
<tr>
<td>10</td>
<td>300</td>
<td>5</td>
<td>6682</td>
<td>6682</td>
<td>51.05</td>
<td>6682</td>
<td>6682</td>
<td>0</td>
<td>0.031</td>
</tr>
<tr>
<td>11</td>
<td>300</td>
<td>10</td>
<td>5661</td>
<td>5661</td>
<td>82.70</td>
<td>5661</td>
<td>5661</td>
<td>0</td>
<td>0.110</td>
</tr>
<tr>
<td>12</td>
<td>300</td>
<td>30</td>
<td>3967</td>
<td>3967</td>
<td>12.75</td>
<td>3967</td>
<td>3967</td>
<td>0</td>
<td>1.718</td>
</tr>
<tr>
<td>13</td>
<td>300</td>
<td>60</td>
<td>2716</td>
<td>2716</td>
<td>14.61</td>
<td>2716</td>
<td>2716</td>
<td>0</td>
<td>9.594</td>
</tr>
<tr>
<td>14</td>
<td>150</td>
<td>100</td>
<td>1569</td>
<td>1569</td>
<td>11.84</td>
<td>1569</td>
<td>1569</td>
<td>0</td>
<td>31.750</td>
</tr>
<tr>
<td>15</td>
<td>400</td>
<td>5</td>
<td>7443</td>
<td>7443</td>
<td>1221.44</td>
<td>7443</td>
<td>7443</td>
<td>0</td>
<td>0.094</td>
</tr>
<tr>
<td>16</td>
<td>400</td>
<td>10</td>
<td>6136</td>
<td>6136</td>
<td>742.42</td>
<td>6136</td>
<td>6136</td>
<td>0</td>
<td>0.281</td>
</tr>
<tr>
<td>17</td>
<td>400</td>
<td>40</td>
<td>4449</td>
<td>4449</td>
<td>41.44</td>
<td>4449</td>
<td>4449</td>
<td>0</td>
<td>2.891</td>
</tr>
<tr>
<td>18</td>
<td>400</td>
<td>80</td>
<td>2668</td>
<td>2668</td>
<td>27.22</td>
<td>2668</td>
<td>2668</td>
<td>0</td>
<td>29.156</td>
</tr>
<tr>
<td>19</td>
<td>400</td>
<td>133</td>
<td>1648</td>
<td>1648</td>
<td>26.74</td>
<td>1648</td>
<td>1648</td>
<td>0</td>
<td>85.954</td>
</tr>
<tr>
<td>20</td>
<td>500</td>
<td>5</td>
<td>7693</td>
<td>7693</td>
<td>60.06</td>
<td>7693</td>
<td>7693</td>
<td>0</td>
<td>0.093</td>
</tr>
<tr>
<td>21</td>
<td>500</td>
<td>10</td>
<td>7662</td>
<td>7662</td>
<td>1113.70</td>
<td>7662</td>
<td>7662</td>
<td>0</td>
<td>0.390</td>
</tr>
<tr>
<td>22</td>
<td>500</td>
<td>50</td>
<td>4331</td>
<td>4331</td>
<td>58.50</td>
<td>4331</td>
<td>4331</td>
<td>0</td>
<td>11.438</td>
</tr>
<tr>
<td>23</td>
<td>500</td>
<td>100</td>
<td>2799</td>
<td>2799</td>
<td>50.70</td>
<td>2799</td>
<td>2799</td>
<td>0</td>
<td>51.875</td>
</tr>
<tr>
<td>24</td>
<td>500</td>
<td>167</td>
<td>1726</td>
<td>1726</td>
<td>48.95</td>
<td>1726</td>
<td>1726</td>
<td>0</td>
<td>416.243</td>
</tr>
<tr>
<td>25</td>
<td>600</td>
<td>5</td>
<td>8792</td>
<td>8792</td>
<td>5458.86</td>
<td>8792</td>
<td>8792</td>
<td>0</td>
<td>0.110</td>
</tr>
<tr>
<td>26</td>
<td>600</td>
<td>10</td>
<td>7632</td>
<td>7632</td>
<td>*</td>
<td>7632</td>
<td>*</td>
<td>0</td>
<td>0.468</td>
</tr>
<tr>
<td>27</td>
<td>600</td>
<td>60</td>
<td>4316</td>
<td>4316</td>
<td>102.58</td>
<td>4316</td>
<td>4316</td>
<td>0</td>
<td>29.437</td>
</tr>
<tr>
<td>28</td>
<td>600</td>
<td>120</td>
<td>2913</td>
<td>2913</td>
<td>82.49</td>
<td>2913</td>
<td>2913</td>
<td>0.03</td>
<td>501.875</td>
</tr>
<tr>
<td>29</td>
<td>600</td>
<td>200</td>
<td>1896</td>
<td>1896</td>
<td>74.47</td>
<td>1896</td>
<td>1896</td>
<td>0.05</td>
<td>610.328</td>
</tr>
<tr>
<td>30</td>
<td>700</td>
<td>5</td>
<td>9151</td>
<td>9151</td>
<td>9151</td>
<td>2449.34</td>
<td>9151</td>
<td>0</td>
<td>0.156</td>
</tr>
<tr>
<td>31</td>
<td>700</td>
<td>10</td>
<td>8637</td>
<td>10641</td>
<td>10641</td>
<td>*</td>
<td>8637</td>
<td>0</td>
<td>0.500</td>
</tr>
<tr>
<td>32</td>
<td>700</td>
<td>70</td>
<td>4460</td>
<td>4460</td>
<td>154.18</td>
<td>4460</td>
<td>4460</td>
<td>0</td>
<td>133.140</td>
</tr>
<tr>
<td>33</td>
<td>700</td>
<td>140</td>
<td>2904</td>
<td>2904</td>
<td>*</td>
<td>2904</td>
<td>*</td>
<td>0</td>
<td>186.062</td>
</tr>
<tr>
<td>34</td>
<td>800</td>
<td>5</td>
<td>9174**</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>9374</td>
<td>0</td>
<td>0.155</td>
</tr>
<tr>
<td>35</td>
<td>800</td>
<td>10</td>
<td>9155**</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>9155</td>
<td>0</td>
<td>0.609</td>
</tr>
<tr>
<td>36</td>
<td>800</td>
<td>80</td>
<td>4879**</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4879</td>
<td>0</td>
<td>112.355</td>
</tr>
<tr>
<td>37</td>
<td>900</td>
<td>5</td>
<td>10117***</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10117</td>
<td>0</td>
<td>0.375</td>
</tr>
<tr>
<td>38</td>
<td>900</td>
<td>10</td>
<td>8799***</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8799</td>
<td>0</td>
<td>0.812</td>
</tr>
<tr>
<td>39</td>
<td>900</td>
<td>90</td>
<td>5000***</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5000</td>
<td>0</td>
<td>77.481</td>
</tr>
</tbody>
</table>

* Berman and Drezner could not find optimal within two hours.
** Berman and Drezner could not solve the problem due to the lack of memory.
*** Since these tests solved only with GHS, there is no guarantee to get optimal value although it is very likely.

A. Kaveh, H. Nasr
then a new harmony search algorithm is presented for solving the conditional p-median problem. Here we used the combinations of the harmony search algorithm and the greedy heuristic for solving a discrete problem. The discrete space and the conditional problem require the use of modified parameters and harmony memory for the algorithm. Our computational results for the harmony search are compared with the other two algorithms and it is shown that the presented algorithm is quite fast and more efficient. From the computational results table, it can be seen that as the number of medians increases, the execution time for the harmony search algorithm increases, and the previous algorithms run slightly faster for a large number of medians. However, in general the proposed algorithm outperforms the previous exact algorithms, especially when the number of medians is not high.

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

References