ASIAN JOURNAL OF CIVIL ENGINEERING (BUILDING AND HOUSING) VOL. 12, NO. 6 (2011)
PAGES 703-718

CHARGED SYSTEM SEARCH ALGORITHM FOR MINIMAX AND MINISUM FACILITY LAYOUT PROBLEMS

A. Kaveh[a*] and P. Sharafi[b]
[a]Centre of Excellence for Fundamental Studies in Structural Engineering, Iran University of Science and Technology, Narmak, Tehran-16, Iran
[2]School of Civil, Mining and Environmental Engineering, University of Wollongong, Wollongong, NSW 2522, Australia

Received: 5 December 2010, **Accepted:** 7 March 2011

ABSTRACT

In this paper, the recently developed meta-heuristic optimization algorithm, known as the charged system search (CSS), is utilized for the problems of facility locations on a network. The CSS is an optimization algorithm that acts on the basis of the interaction between charged particles (CPs) considering the governing laws of Coulomb and Gauss from electrical physics and laws of motion from Newtonian mechanics.

Keywords: Combinatorial optimization; charged system search; graph theory; medians; centers; weighted graphs; facility locations

1. INTRODUCTION

The problem of finding optimal facilities on a network or a directed graph as a discrete optimization problem has attracted the attention of many researchers. The problems of this kind, which have combinatorial nature, are often investigated in the following two forms; First, minimax problems in which the maximum cost of serving clients is minimized known as center finding, and second the minisum problems in which the sum of the costs of serving clients is minimized. The latter is known as the median finding problem. Since the nature of these two problems are the same and the only difference between them are on the way costs are calculated, many evolutionary algorithms for solving one type is capable of dealing with the second type. In this paper, the main focus will be on the second type known as the median finding problem. By a small modification in the algorithm, a solution for the center problems is easily obtained.

Layout optimization of facilities such as airports, reservoirs and stores, universities and educational centers, and shopping centers are some of the applications of the median finding

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

problem. The layout optimization of emergency services is the main application of the center finding problem. For optimum location, finding many parameters such as physical, economical, social, environmental and political factors are considered. Each of these factors can be incorporated in the solution, by assigning weights to the nodes and/or edges of the corresponding graph. In fact, the problem can be formulated as finding some points on a network (construction graph). In this way, some facilities or services should be selected such that the average cost (or the maximum cost in the center problem) of the trips or the average time for reaching these facilities become minimum [1]. Facility layout problem is an NP-hard combinatorial optimization problem [2], and as it will be discussed, the exact solution of the problem is complex and highly time consuming for graphs with a large number of nodes. Therefore, many approximate algorithms are developed for finding the location of such facilities [3-11]. For selecting k nodes as medians out of N nodes of a graph, the number of cases that must be considered, is equal to the number of combinations of N nodes taken k at a time. Due to the factorial nature of this selection, the dimension of the problem is high and it requires a non-polynomial algorithm.

Recently, meta-heuristics such as Genetic algorithms and Ant Colony optimization are being employed to deal with this problem. An Ant based algorithm was proposed by Kaveh and Sharafi [12] for finding k-median of the weighted graphs.

Meta-heuristic algorithms mostly tend to perform properly for the optimization problems. This is because such methods avoid simplifying or making shortening assumptions about the original form. Evidence of this is their successful applications to a vast variety of fields, such as engineering, art, biology, economics, marketing, genetics, operations research, robotics, social sciences, physics, politics and chemistry. As a newly developed type of meta-heuristic algorithm, the charged system search (CSS) was introduced for design of structural problems by Kaveh and Talatahari [13-15].

This method utilizes the governing Coulomb and Gauss laws from electrostatics and the Newtonian law of mechanics. Inspired by these laws, a model is created to formulate the facility layout optimization problem. CSS contains a number of agents which are called charged particle (CP). Each CP is considered as a charged sphere which exerts an electric force on other CPs according to the Coulomb and Gauss laws. The resultant forces and the motion laws determine the new location of the CPs. Using these laws provides a good balance between the exploration and the exploitation of the algorithm. As a result, CSS can easily be utilized for both discrete and continues optimization problems [13]. In this paper, the CSS is employed for facility location finding as a discrete optimization problem.

2. BACKGROUND

2.1 Electrical laws

In physics, the space surrounding an electric charge has a property known as the electric field of that charge. This field exerts a force on other electrically charged objects. The electric field surrounding a point charge is given by Coulomb's law. Coulomb has confirmed that the electric force between two small charged spheres is proportional to the inverse square of their separation distance r_{ij}. Therefore, it provides the magnitude of the

electric force (Coulomb force) between two charged points (CPs). This force on a CP q_j at position r_i, experiencing a field due to the presence of another CP q_j at position r_i, can be expressed as

$$F_{ij} = k_e \frac{q_i q_j}{r_{ij}^{\ 2}} \frac{r_i - r_j}{\left\| r_i - r_j \right\|} \tag{1}$$

where k_e is a constant known as the Coulomb constant; r_{ij} is the separation of the two CPs, Halliday et. al. [16].

Consider an insulating solid sphere of radius which has a uniform volume charge density and carries a total charge of magnitude q_i. The magnitude of the electric force at a point outside the sphere is defined by Eq. (1), while this force can be obtained using Gauss's law at a point inside the sphere as

$$F_{ij} = k_e \frac{q_i q_j}{a^3} r_{ij} \frac{r_i - r_j}{\left\| r_i - r_j \right\|} \tag{2}$$

In order to calculate the electric force on a CP (q_j) at a point (r_j) due to a group of CPs, the principle of superposition can be applied to electric forces as

$$F_j = \sum_{i=1, i \neq j}^{N} F_{ij} \tag{3}$$

where N is the total number of CPs, and F_{ij} is equal to

$$F_{ij} = \begin{cases} k_e \dfrac{q_i}{a^3} r_{ij} \dfrac{r_i - r_j}{\left\| r_i - r_j \right\|} & if\ r_{ij} < a \\ k_e \dfrac{q_i}{r_{ij}^2} \dfrac{r_i - r_j}{\left\| r_i - r_j \right\|} & if\ r_{ij} \geq a \end{cases} \tag{4}$$

Therefore, the resulting electric force can be obtained as

$$F_j = k_e q_j \sum \left(\frac{q_i}{a^3} r_{ij} .i_1 + \frac{q_i}{r_{ij}^2} .i_2 \right) \frac{r_i - r_j}{\left\| r_i - r_j \right\|} \qquad \begin{Bmatrix} i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{Bmatrix} \tag{5}$$

2.2 Newtonian mechanics laws

Newtonian mechanics studies the motion of objects. In the study of motion, the moving object is considered as a particle regardless of its size. In general, a particle is a point-like mass having infinitesimal size. The motion of a particle is completely known if the particle's

position in a space is known at all times. The displacement of a particle is defined as its change in position. As it moves from an initial position r_{old} to a final position r_{new}, its displacement is given by

$$\Delta r = r_{new} - r_{old} \tag{6}$$

The slope of tangent line of the particle position represents the velocity of this particle as

$$v = \frac{r_{new} - r_{old}}{t_{new} - t_{old}} = \frac{r_{new} - r_{old}}{\Delta t} \tag{7}$$

When the velocity of a particle changes with time, the particle is said to be accelerating. The acceleration of the particle is defined as the change in the velocity divided by the time interval Δt during which that change has occurred:

$$\alpha = \frac{v_{new} - v_{old}}{\Delta t} \tag{8}$$

The displacement of any object as a function of time is obtained by using Eq. (8) as

$$r_{new} = \frac{1}{2}\alpha.\Delta t^2 + v_{old}.\Delta\Delta + r_{old} \tag{9}$$

Also according to Newton's second law, we have

$$F = m.\alpha \tag{10}$$

where m is the mass of the object. Substituting Eq. (10) in Eq. (9), we have

$$r_{new} = \frac{1}{2}\frac{F}{m}.\Delta t^2 + v_{old}.\Delta t + r_{old} \tag{11}$$

2.3 The rules of the charged system search

In this section, an optimization algorithm is presented utilizing the aforementioned physics laws, which is called Charged System Search (CSS). In the CSS, each solution candidate X_i containing a number of decision variables (i.e. $X_i = \{ x_{i,j} \}$) is considered as a CP. A CP is affected by the electrical fields of the other CPs. The quantity of the resultant force is determined by using the electrostatics laws as discussed in Sect. 2.1, and the amount of the movement is determined using the Newtonian mechanics laws of section 2.2. It seems that a CP with good results must exert a stronger force than the bad one, so the amount of the charge will be defined considering the objective function value, *fit(i)*. In order to introduce CSS, the following rules are developed:

Rule 1: Many of the natural evolution algorithms maintain a population of solutions which are evolved through random alterations and selection [13-15]. Similarly, CSS considers a number of CPs. Each CP has a magnitude of charge (q_i) and as a result creates an electrical field around its space. The magnitude of the charge is defined considering the quality of its solution as

$$q_i = \frac{fit(i) - fit_{best}}{fit_{best} - fit_{worst}} \quad i=1, 2, \ldots, C \tag{12}$$

where fit_{best} and fit_{worst} are the so far best and the worst fitness of all CPs; *fit(i)* represents the objective function value or the fitness of the *i*th CP, and *N* is the total number of CPs. The separation distance r_{ij} between two CPs is defined as follows:

$$r_{ij} = \frac{\left\| X_i - X_j \right\|}{\left\| (X_i + X_j)/2 - X_{best} \right\| + \varepsilon} \tag{13}$$

where X_i and X_j are the positions of the *i*th and *j*th CPs, X_{best} is the position of the best current CP, and ε is a small positive number to avoid singularities.

Rule 2: The initial positions of CPs are determined randomly in the search space and the initial velocities of charged particles are assumed to be zero.

Rule 3: Electric forces between any two CPs are attractive. Utilizing this rule increases the exploitation ability of the algorithm. It is also possible to define repelling force between CPs. When a search space is a noisy domain, having a complete search before converging to a result is necessary. In such a condition, the addition of the ability of repelling forces to the algorithm may improve its performance.

Rule 4: Good CPs can attract the other agents and bad CPs repel the others, proportional to their rank, that is

$$c_{ij} \propto rank(CP_j) \wedge \begin{cases} 0 < c_{ij} \le +1 & \text{if the CP is above average} \\ -1 \le c_{ij} < 0 & \text{if the CP is below average} \end{cases} \tag{14}$$

where c_{ij} is a coefficient which determines the type and the degree of influence of each CP on the other CPs, considering their fitness apart from their charges. This means that good agents are awarded the capability of attraction and bad ones are given the repelling feature, which will improve the exploration and exploitation abilities of the algorithm. In one hand, when a good agent attracts a bad one, the exploitation ability is provided for the algorithm. On the other hand, when a bad agent repels a good CP, the exploration is provided.

Rule 5: The value of the resultant electrical force affecting a CP is determined using Eq. (5) as

$$F_j = q_j \sum_{i,i\neq j} \left(\frac{q_i}{a^3} r_{ij}.i_1 + \frac{q_i}{r_{ij}^2}.i_2 \right).c_{ij}.(X_i - X_j) \qquad \begin{cases} j = 1,2,...,C \\ i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{cases} \tag{15}$$

where F_j is the resultant force acting on the *j*th CP, as illustrated in Figure 1. In this algorithm, each CP is considered as a charged sphere with radius "*a*" having a uniform volume charge density. In this paper, *a* is set to unity.

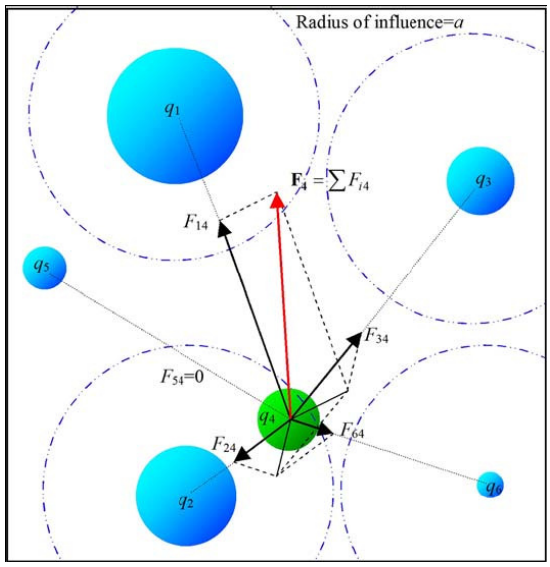

Figure 1. Determining the resultant electrical force acting on a CP [14]

Rule 6: The new position and velocity of each CP is determined considering Eq. (7) and Eq. (11) as follows:

$$x_{j,new} = rand_{j1}.k_a.\frac{F_j}{m_j}.\Delta t^2 + rand_{j2}.k_v.v_{j,old}.\Delta t + x_{j,old} \tag{16}$$

$$v_{j,new} = \frac{x_{j,new} - x_{j,old}}{\Delta t} \tag{17}$$

where, k_a is the acceleration coefficient; k_v is the velocity coefficient to control the influence of the previous velocity; and $rand_{j1}$ and $rand_{j2}$ are two random numbers uniformly distributed in the range of (0,1). m_j is the mass of the *j*th CP which is equal to q_j in this paper. Δt is the time step and is set to one. Figure 2 illustrates the movement of a CP to its new position using this rule.

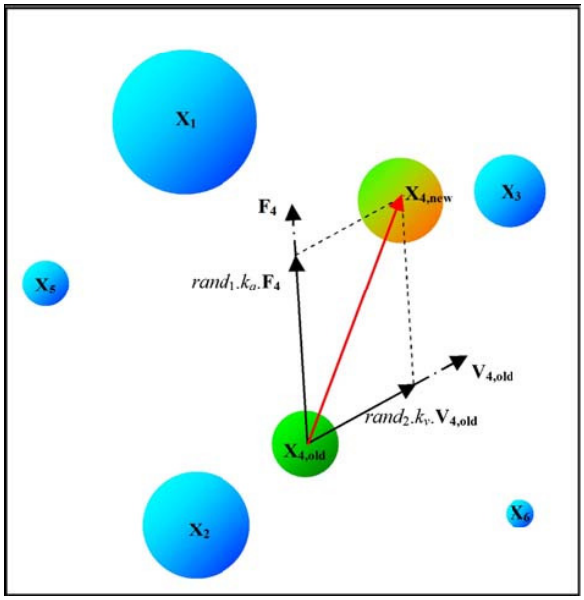

Figure 2. Movement of a CP to its new position [14]

The effect of the pervious velocity and the resultant force acting on a CP can be decreased or increased based on the values of the k_v and k_a, respectively. Excessive search in the early iterations may improve the exploration ability; however, it must be deceased gradually, as described before. Since k_a is the parameter related to the attracting forces, selecting a large value for this parameter may cause a fast convergence and choosing a small value can increase the computational time. In fact k_a is a control parameter of the exploitation. Therefore, choosing an incremental function can improve the performance of the algorithm. Also, the direction of the pervious velocity of a CP is not necessarily the same as the resultant force. Thus, it can be concluded that the velocity coefficient k_v controls the exploration process and therefore a decreasing function can be selected. Thus k_v and k_a are defined as

$$k_v = 0.5(1 - iter/iter_{max}), \quad k_a = 0.5(1 + iter/iter_{max}) \tag{18}$$

Rule 7: Charged memory (CM) is utilized to save a number of the best so far solutions. Also, the vectors stored in the CM can have influence on the CPs. This may increase the computational cost, and therefore it is assumed that the same number of the worst particles cannot attract others.

Rule 8: Those CPs violating the limits of the variables are regenerated using the harmony search-based handling approach as described by Kaveh and Talatahari [13,14].

Rule 9: Maximum number of iterations is considered as the terminating criterion.

The general flowchart of the CSS algorithm is illustrated in Figure 3.

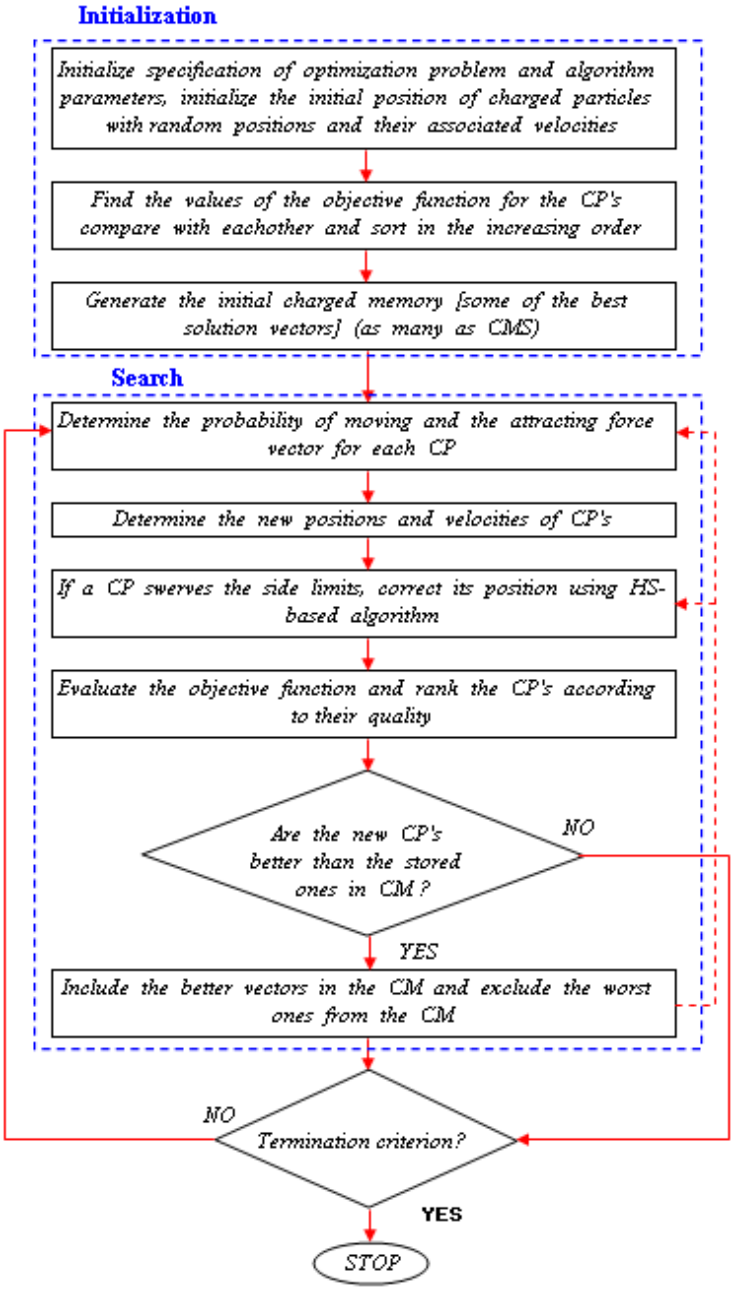

Figure 3. The general flowchart of the CSS algorithm [14]

3. MATHEMATICAL FORMULATION OF THE MEDIAN PROBLEM

3.1 Medians problem (Minisum problems)

Consider a weighted and directed graph $G = (N, E)$ with N nodes and E edges. If $d(i,j)$

denotes the shortest distance from node i to node j (in general $d(i,j) \neq d(j,i)$) and v_i shows the weight of the node i, then

$$\begin{cases} \sigma_0(i) = \sum_{j \in N} v_j d(i,j) \\ \sigma_t(i) = \sum_{j \in N} v_j d(j,i) \end{cases} \tag{19}$$

$\sigma_0(i)$ and $\sigma_t(i)$ are called the *out-transmission* and *in-transmission* of the node i, respectively. The number $\sigma_0(i)$ is the sum of the entries of row i of a matrix obtained by multiplying every column j of the distance matrix $[d_{ij}]$ by v_j, and $\sigma_t(i)$ is the sum of the entries of column i of a matrix obtained by multiplying every row j of the distance matrix by v_j. The node $\bar{i}_o \in N$ for which $\sigma_o(\bar{i}_o)$ becomes minimum is the *out-median* of the graph and $\bar{i}_t \in N$ which corresponds to minimum $\sigma_t(\bar{i}_t)$ is called the *in-median* of the graph. Now let N_k be a subset of N with k nodes, then each N_k will have its own in- and out-transmissions, as follows:

$$\begin{cases} \sigma_0(N_k) = \sum_{j \in N} v_j d(N_k, j) \\ \sigma_t(N_k) = \sum_{j \in N} v_j d(j, N_k) \end{cases} \tag{20}$$

where d is defined as follows:

$$\begin{cases} d(N_k, j) = min[d(i', j)] : (i' \in N_k) \\ d(j, N_k) = min[d(j, i')] : (i' \in N_k) \end{cases} \tag{21}$$

Let i' be the node of N_K which produces the minimum in Eq. (21), then we say the node j is allocated to i'. The concept of allocating one node to a node of a subset N_K is the basis of the approximate algorithms for finding the medians. Now we find the subset with k elements, $\overline{N}_{ko} \in N$, such that $\sigma_o(\overline{N}_{ko})$ becomes minimum. Then $\overline{N}_{ko}$ is a set containing the k-out medians. Similarly, the subset $\overline{N}_{kt} \in N$, which makes $\sigma_t(\overline{N}_{kt})$ a minimum, is known as the k-in medians. Due to the similarity of the process for finding in- and out-medians, from now on the word *medians* will refer to the out-medians of a graph. Using the algorithms of the next section, with small alterations, the in-medians can also be found. For symmetric problems and undirected graphs, where the costs or the distances between the nodes are not dependent on the directions, these two medians become identical.

Therefore, the problem of finding k medians of a graph becomes the same as the minimization of the function $\sigma(\overline{N}_k)$ during which a set of k elements is found which minimizes the above function. Now we define allocation matrix $[\xi_{ij}]$ such that an entry ξ_{ij}

is 1 if the node j is allocated to i, and 0 otherwise, i.e. $\xi_{ij}=1$ shows that the node i is a median and $\xi_{ij}=0$ otherwise.

The problem of k-median can be expressed in an integer programming form as follows:

$$Minimize: \sigma=\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}\xi_{ij} \tag{22}$$

$$Subjected\ to: \sum_{i=1}^{n}\xi_{ij}=1 \quad for\ j=1,2,...,n \tag{23}$$

$$\sum_{i=1}^{n}\xi_{ii}=k \tag{24}$$

$$\xi_{ij}\le\xi_{ii} \quad for\ all \quad i,j=1,2,...,n \tag{25}$$

$$\xi_{ij}=0\ or\ 1 \tag{26}$$

In the above relationships, $[d_{ij}]$ is the weighted distance matrix, i.e. it is the distance matrix with every column j multiplied by v_j. Eq. (23) ensures that any given node j is allocated to one and only one median i. Eq. (24) ensures that there are only k medians, and Eq. (25) guaranties that allocations are made only to the median nodes.

Another type of k-median is formulated as the minimization of $\sigma_0(N_k)=\sum_{j\in N} v_j D(N_k,j)$, where $D(N_k,j)$ is the distance matrix (not the shortest distance) between the nodes. Although the nature of this optimization problem seems to be different with that of Eq. (20), however, algorithms with small changes in the definitions are applicable to this problem as well.

3.2 Centers problem (Minimax problems)

For the above-mentioned graph, if N_p is a subset of N with p nodes each N_p will have its own *in-* and *out-separation***,** associated with *in-* and *out-centers*, as follows:

$$\begin{cases} s_o(i)=Max[v_j d(N_p,j)] & j\in N \\ s_t(i)=Max[v_j d(j,N_p)] & j\in N \end{cases} \tag{27}$$

When each of the following criteria is satisfied for a set of p, it is called *p-out centers* or *p-in centers*, respectively.

$$\begin{cases} s_0(N^*_{po}) = Min[s_o(N_p)] & N_p \in N \\ s_0(N^*_{pt}) = Min[s_t(N_p)] & N_p \in N \end{cases} \quad (28)$$

4. THE CSS ALGORITHM FOR FACILITY LAYOUT PROBLEM

This algorithm attempts to find the medians or centers of a weighted graph using the CSS method. As mentioned before, due to the similarities between the median and center finding problems, the basis of these algorithms are identical. The only difference is in defining the objective functions. That is, the main procedure of the CSS algorithm for finding the facility locations is the same but the objective function for centers finding problem comes from Eq. (21) while for medians finding problem it comes from Eq. (28). As a result, in the following algorithm the word "location" can be referred to both medians and centers locations. Imposing a small modification in the objective function, k-center of the graph can easily be found through the k-median algorithm and vice versa.

The aim is to find k optimum nodes on a weighted graph $G = (N, E)$ with N nodes and E edges in order to minimize the objective function of the problem. Each solution candidate X_i, containing a number of decision variables $x_{i,j}$, is considered as a CP and each $x_{i,j}$, showing the position of the CP, indicates whether or not the node j is capable of being a location. That is, each solution candidate, as an n-dimensional array, contains n elements $x_{i,j}$ $(j=1,2,...,n)$ and each $x_{i,j}$ shows the status of a node. The higher amount of each $x_{i,j}$ indicates the higher probability of belonging to the locations set for the node j. In fact, for a solution candidate X_i the indices of k biggest $x_{i,j}$'s are considered as the k desirable locations of the solution.

The algorithm for facility layout follows the nine above mentioned general rules of the CSS algorithm. This algorithm consists of nine steps as follows:

Step 1: The number of CPs, i.e. candidate agents, is determined. For the algorithm of finding k locations, this number is set to $C=[fix(n/k)+1]$. This is because we need to give each node a chance of being chosen as a facility location. Using larger number of CPs may result in more accurate results, but it significantly increases the computational time. On the other hand, using smaller number of them may lead to undesirable results. The considered number of CPs is capable to keep the balance in a moderate level.

Step 2: The CPs are defined and settled in their initial positions. As mentioned, the amount of each position element $x_{i,j}$ of solution candidates is proportional to the probability of node j to be chosen as a median in that solution. For this purpose, the positions of CPs are defined in a way that all the n elements of each CP are set to zero but k of them, which are set to unity provided the CPs cover the entire nodes. That is, for example for defining the position of CP_1, k random elements are set to unity. Then for defining the position of CP_2, the next k nodes, which have not been chosen yet, are randomly set to unity and so on until the last CP is defined. Obviously, for defining the last CP some elements might be set to unity that, have been chosen in last CPs. Therefore, for the agent X_i with n elements

$x^{(0)}_{i,j} = 1$ indicates that j is considered as an initial median for the solution i. In this way, every node is given a chance for being chosen as a median.

Now, the initial candidate solutions X_i and as a result, their positions $\{x^{(0)}_{i,j}\}$ are randomly nominated. In other words, in this phase, C candidate solution X_i $(i=1,2,\ldots,C)$ which are located in their positions presented by $x^{(0)}_{i,j}$ are defined. $(j=1,2,\ldots,n)$. The initial velocity for all the CPs is considered to be zero. $(v^{(0)}_{i,j} = 0 \quad \forall\, i,j)$

Step 3: The magnitude of charge for each CP is calculated using Rule 1. For this purpose, the objective functions for each agent must be calculated. As mentioned before, this is the only phase which is different for the center and median finding algorithms. The objective function for median finding problem is obtained from Eq. (21) while for center finding this is calculated from Eq. (23). In this step when the objective functions are calculated, they should be put in order and the best and the worst ones should be saved. This will help the algorithm to judge better in subsequent steps. Then the magnitude of charge for each CP, q_i, is obtained through the Eq. (12).

Step 4: The separation distance between CPs are calculated. In the previous step, the position of each CP was defined by a coordinate of n elements. Having the X_i for all the CPs, the separation distance between them are calculated using the Eq. (13). It should be mentioned that in problems where X_i is an n-dimensional array, the intention of calculating distance between every two CPs is to find how far the two assumed nodes are in the n-dimensional space. In fact, the calculations of distance, velocity and acceleration are all performed in a multi-dimensional space.

Step 5: The type and the degree of influence of each CP on the other agents are determined. For this purpose, using the rank of the CPs obtained in step 3, a number between +1 and −1 is assigned to each agent proportional to its rank. That is, the number +1 is assigned to the best agent and −1 to the worst one, and so on. Such an assignment leads to improvement in simultaneous exploration and exploitation abilities.

Step 6: The value of the resultant electrical force affecting a CP is determined using the Eq. (15). Each F_j is an n-dimensional array and shows the tendency of agent j towards other CPs.

Step 7: New position and velocity of each CP is determined considering Eqs. (16) and (17). As mentioned before, each new position determined by an n-dimensional array shows the new probability of each node to be selected as a median for each CP.

Step 8: The agents violating the limits of the variables are regenerated using the harmony search-based handling approach. Then the best so far solutions are saved.

Step 9: Maximum number of iterations is considered as the terminating criterion.

5. NUMERICAL EXAMPLES

In this section three examples are presented and the results are compared to those of the ACO algorithm with local search [17] in Table 1. Then the comparisons of convergence rates for the two algorithms for each instance are made.

The topological properties of the finite element models are transferred to the connectivity properties of graphs, by the clique graphs [18]. This graph has the same nodes as those of the corresponding finite element model, and the nodes of each element are cliqued, avoiding the multiple edges for the entire graph.

Example 1: Consider a circular finite element mesh (FEM) as shown in Figure 4. The element clique garph of this model contain 1248 nodes and 4704 edges. The performance of the CSS and ACO are tested on this model, and the results are presented in Table 1. Quality of the results is provided in this table. The number of medians in this example are considered to be k={5,10}.

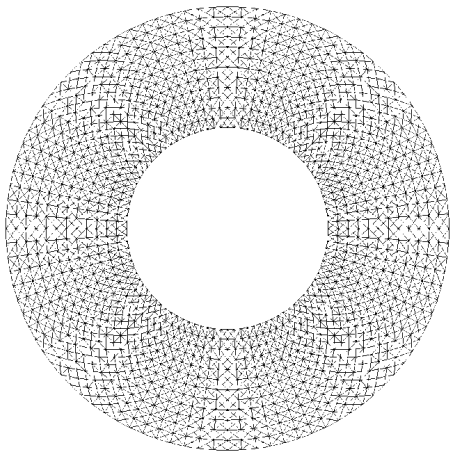

Figure 4. A graph with 1248 nodes and 4704 elements with the weights of all the edges, and the demands of all nodes being taken as unity

Example 2: A graph with 1575 nodes and 2925 elements is considered as shown in Figure 5. Similar to the previous example, the performance of the CSS and ACO are provided in Table 1. The numbers of the medians in this example are considered to be k={5,10}.

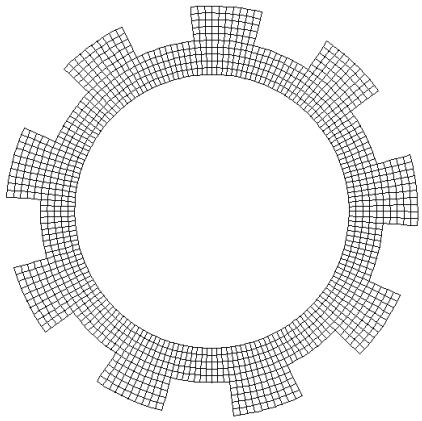

Figure 5. A fan-shaped finite element model with 1575 nodes and 2925 elements with the weights of all the edges, and the demands of all nodes being taken as unity

Example 3: An H-shape finite element mesh (FEM) is considered as shown in Figure 6**.** The element clique garph of this model contain 4949 nodes and 9688 beam element (edges). The performance of the CSS and ACO are tested on this model, and the results are presented in Table 1. The numbers of medians in this example are considered to be k={5,10}. Figures 7 and 8 illustrate the convergence history of this example.

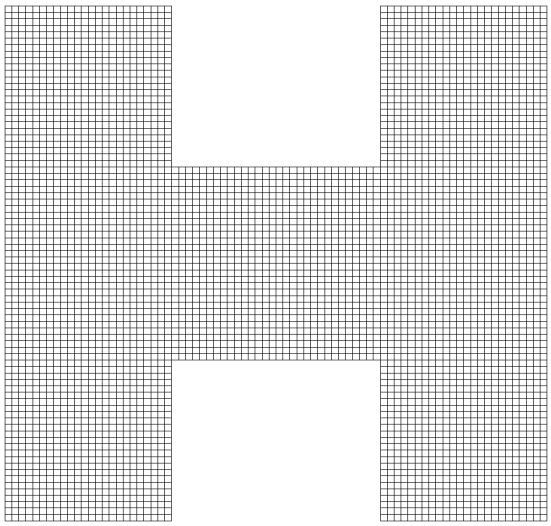

Figure 6. An H-Shaped graph with 4949 nodes and 9688 elements with the weights of all the edges, and the demands of all nodes being taken as unity

Table 1: Results of the CSS and ACO algorithms for the examples

Instance	**Number of Facilities**	**Medians**		**Centers**	
		CSS Algorithm	**ACO Algorithm**	**CSS Algorithm**	**ACO Algorithm**
Example 1	k=5	Cost=17449	Cost=17471	Cost=50	Cost=53
	k=10	Cost=9872	Cost=9888	Cost=31	Cost=31
Example 2	k=5	Cost=18195	Cost=18140	Cost=44	Cost=44
	k=10	Cost=10257	Cost=10266	Cost=28	Cost=27
Example 3	k=5	Cost=120339	Cost=121004	Cost=119	Cost=127
	k=10	Cost=79663	Cost=79812	Cost=69	Cost=74

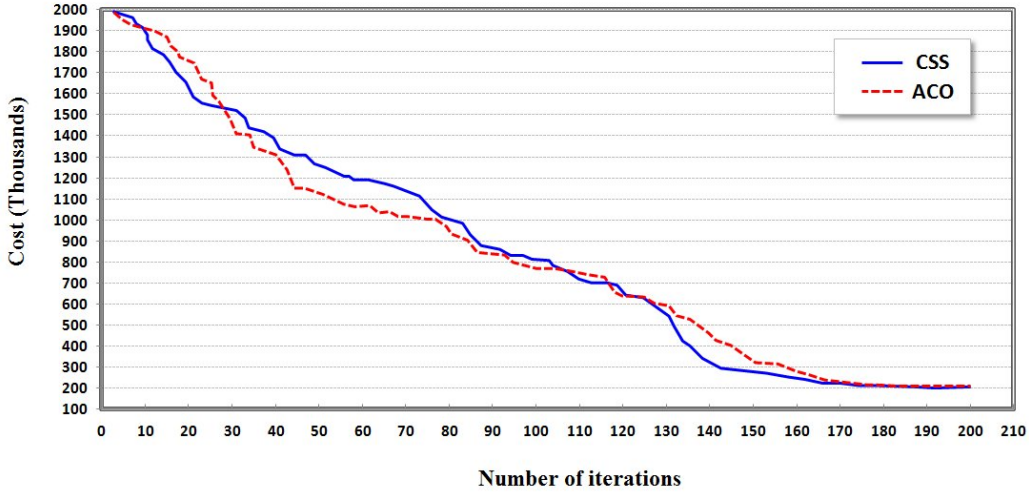

Figure 7. The convergence history of example 3 for the CSS compared to the ACO with local search for finding medians

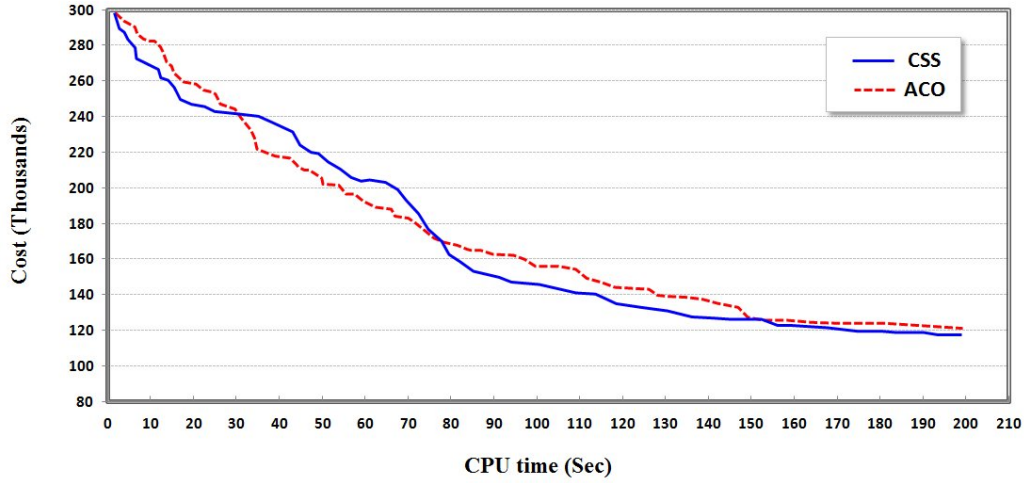

Figure 8. The convergence history of example 3 for the CSS compared to the ACO with local search for finding centers

6. CONCLUSIONS

The main objective of this paper was to show the applicability and robustness of the CSS for facility layout problem as a combinatorial optimization problem. From Table 1, it can be observed that the results are quite satisfactory, comparing to those of the Ant Colony Optimization algorithm.

Figures 7 and 8 show that the convergence histories for the CSS and ACO are to some extent similar, and move towards their optimum in a relatively identical manner. However, the outcomes suggest that the CSS can achieve better results in shorter time compared to the

ACO. In fact, in many instances for which the factor of time is more important, one may achieve relatively better results by the CSS algorithm.

Acknowledgement: The first author is grateful to the Iranian Academy of Sciences for the support.

REFERENCES

1. Christofides N. *Graph Theory: An Algorithmic Approach*. Academic Press, New York, 1975.
2. Kariv O, Hakimi SL. An algorithmic approach to network location problems, *SIAM Journal on Applied Mathematics*, No. 3, **37**(1979) 539-60.
3. Hakimi SL. Optimum locations of switching centres and the absolute centres and medians of a graph, *Operations Research*, No. 3, **12**(1964) 450-9.
4. Christofides N, Beasley JE. A tree search algorithm for the p-median problem. *European Journal of Operational Research*, No. 2, 10(1982)196-204.
5. Megiddo N, Supowi K. On the complexity of some common geometric location problems, *Society for Industrial and Applied Mathematics*, (1984).
6. Galvao RD. A dual-bounded algorithm for the p-median problem, *Operations Research,* 28(1980)1112-21.
7. Daskin MS. *Network and Discrete Location*, Wiley, 1995.
8. Erlenkotter D. A dual-based procedure for uncapacitated facility location, *Operations Research,* No. 6, **26**(1987)992-1009.
9. Correa ES, Steiner MTA, Freitas AA, Carnieri C. A genetic algorithm for the pmedian problem, *Genetic and Evolutionary Computation Conference*, 2001, pp. 1268-1275.
10. Alba E, Domínguez E. Comparative analysis of modern optimization tools for the p-median problem, *Statistics and Computing*, **16**(2006) 251-60.
11. Alp O, Erkut E, Drezner Z. An efficient genetic algorithm for the p-median problem, *Annals of Operations Research*, **122**(2003)21-42.
12. Kaveh A, Sharafi P. Ant colony optimization for finding medians of weighted graphs. *Engineering Computations*, No. 2, **25**(2008)102-14.
13. Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search, *Acta Mechanica*, Nos. (3-4), **213**(2010)267-86.
14. Kaveh A, Talatahari S. Optimal design of truss structures via the charged system search algorithm, *Structural Multidisplinary Optimization*, No. 6, **37**(2010)893-911.
15. Kaveh A, Talatahari S. Charged system search for optimum grillage systems design using the LRFD-AISC code, *Journal of Constructional Steel Research*, **66**(2010)767-71.
16. Halliday D, Resnick R, Walker J. *Fundamentals of Physics*, 8th edition, Wiley, New York, 2008.
17. Kaveh A, Talatahari S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures, *Computers and Structures*, **87**(2009) 267-83.
18. Kaveh A. *Optimal Structural Analysis*, 2nd edition, Wiley, Chichister, UK, 2006.