Iranian Journal of Numerical Analysis and Optimization
Vol. 7, No. 1, (2017), pp 65-82
DOI: 10.22067/ijnao.v7i1.52961

Efficient methods for goal square Weber location problem

J. Fathali* and A. Jamalian

Abstract

In this paper, we consider a special case of Weber location problem which we call goal location problem. The Weber location problem asks to find location of a point in the plane such that the sum of weighted distances between this point and n existing points is minimized. In the goal location problem each existing point P_i has a relevant radius r_i and it's ideal for us to locate a new facility on the distance r_i from P_i for $i = 1, ..., n$. Since in the most instances there does not exist the location of a new facility such that its distance to each point P_i be exactly equal to r_i. So we try to minimize the sum of the weighted square errors. We consider the case that the distances in the plane are measured by the Euclidean norm. We propose a Weiszfeld like algorithm for solving the problem and also we use two modifications of particle swarm optimization method for solving this problem. Finally the results of these algorithms are compared with results of BSSS algorithm.

Keywords: Location theory; Weiszfeld method; Particle swarm optimization.

1 Introduction

Location theory is one of the useful and interesting fields of operations research and has many applications in real life. Single facility location problems are fundamental problems in location theory in which n existing demand points are located at known distinct points $P_1, \cdots, P_n$, and a new facility should be located at a point X such that the sum of weighted distances from

*Corresponding author
Received 13 January 2016; revised 18 June 2016; accepted 14 September 2016
J. Fathali
Department of Mathematics, Shahrood University of Technology, University Blvd., Shahrood, Iran. email: fathali@shahroodut.ac.ir

A. Jamalian
Department of Computer Science, Faculty of Mathematical Sciences, University of Guilan, Rasht, Iran. e-mail: a.jamalian.math@gmail.com

it to demand points is minimized. Let $d(X, P_i)$ represent the distance traveled per trip between points X and P_i, the weight w_i represents the product of cost per unit distance traveled and number of trips made per year between the new facility and existing demand point i and some times shows value of demand in this point. The total cost is given by:

$$\min F(X) = \sum_{i=1}^{n} w_i d(X, P_i). \tag{1}$$

The Euclidean problem is obviously referred to as the Steiner-Weber problem or the general Fermat problem, and has an extraordinary longevity (see [8]). In fact, a version of the problem for the case $n = 3$ and $w_i = 1$ for $i = 1, 2, 3$, was posed purely as a problem in geometry, by Fermat early in the seventeenth century, and was solved by Toricelli prior to 1640 (see [16]). The problem was studied by Steiner, a Swiss mathematician, in the nineteenth century, and by Weber, a German economist, early in the twentieth century. Single facility location with Euclidean distance is also called Weber problem and an iterative procedure was proposed for solving this problem by Weiszfeld [24] in 1937 and rediscovered by Kuhn [17] in 1962. This algorithm is widely used because of its simplicity and effectiveness. The algorithm can be generalized to other location problems where the cost is a function of the Euclidean distance rather than just being proportional to the distance (see e.g. [10]). Some other cases of Weber location problem can be find in [4,5,23]).

Fathali et al. [12], considered a special case of Weber problem in which a specified radius for every demand point is considered and the distance between the new facility and the demand point P_i is equal to the corresponding radius, r_i. However since in the real instances rarely exist the location of a new facility such that its distance to each point p_i is exactly r_i. So they tried to minimize the sum of the weighted square errors. We call this problem as Goal Square Weber Location Problem (GSWLP). Jamalian and Fathali [14] presented a linear model for the location problem with minimum absolute error on the block norms.

In what follows we explain the GSWLP in Section 2. Model properties and results are stated in Section 3. In this section main results and a proposed algorithm for solving problem are given. Because the objective function may have many local optima, we also use particle swarm optimization method for solving this problem. In Section 4, the particle swarm optimization method is explained. In Section 5, the computational results of comparing four algorithms WLA, PSO, PSOC and BSSS for this problem are given.

2 Goal Square Weber location problem

In single facility location problem, we want to find location of a new facility such that the sum of weighted distances from the facility to all demand points is minimized. In Fathali et al. [12] a radius, r_i, and value of demand, w_i correspond to every point P_i is considered. We want to find the location of a new facility such that the distances between it and the demand points $P_i,\ i = 1, \ldots, n$ is equal to r_i. Since this situation occurs in real applications rarely we try to minimize sum of weighted squared error.

In fact, in this problem for every point a goal distance is considered and we want to minimize the error of satisfaction location of facility in these goals. It is desirable for us that new facility be located on circle of correspond radius and center of demand point. For an example of this problem, if $r_i = r$, $i = 1, 2, \cdots, n$ and demand points be located on a circle of radius r, the optimal point is center of circle. In the case $r_i = r$, $i = 1, 2, \cdots, n$ the problem converted to the square Weber problem, so we call this problem as Goal Square Weber Problem.

The problem studied in this paper is different from the covering location problem (CLP). Two important covering models are the maximal and minimal covering location problems which are suitable for siting desirable and undesirable facilities, respectively (see e.g., [3,6]). In both of these problems a facility covers a demand point if the demand point lies within a pre-specified coverage radius. The objective is to locate a number of facilities so as to minimize or maximize the total coverage. There are some differences between CLP and GSWLP. Firstly, CLP is a multifacility location problem, while in the GSWLP we want to find the location of a single facility. Secondly, in CLP a radius is associated with facilities, in contrast to the GSWLP, where a radius r_i is associated with demand point P_i. Thirdly, in the maximal CLP, it is desirable if the distance between the demand points and facilities be less than or equal to a pre-determined radius and in the minimal CLP this distance should be more than or equal to a given radius, while in GSWLP it is desirable if the new facility exactly be in the given distances of the demand points.

Determination of the location of a company in the vicinities of some cities such that the setting up costs and the transportation costs and minimized can be an application of this problem as mentioned in [12] . Suppose that the cost of establishing a facility in the regions that are farther than a given distance r_i from city r_i is very low. On the other hand a move away from a city causes the transportation costs to increase. Therefore a tradeoff between the setting up costs and the transportation costs seems to be reasonable. GSWLP also has some other applications for locating facilities that are either desirable or undesirable. In these cases we want the facility not be close than a specified distance to facility centers, because of its undesirability, on the other hand, if the facility be so far from the facility centers, cost of providing security, human forces, transportation installation, and other costs will increase.

3 Model Properties

Let $P_i = (a_i, b_i)$ for $i = 1, \cdots, n$ be given points in the plane. Formulation of GSWLP is as follows:

$$\min F(X) = \sum_{i=1}^{n} w_i \left(d(X, P_i) - r_i\right)^2 \tag{2}$$

where $X = (x, y)$, $d(X, P_i) = \sqrt{(x - a_i)^2 + (y - b_i)^2}$, and r_i and w_i are corresponding radius and weight of demand point P_i, respectively. Fathali et al. [12] show that the objective function of model (2), is non convex and optimal solution is in extended rectangular hull of demand points. Extended rectangular hull of demand points is the smallest rectangle which contains all the demand points with their circles. We present a new proof by Farkas lemma in the Theorem 1. The advantage for our new proof is that in this proof we show there is an improvement direction from any point which is out of rectangular hull.

Farkas lemma is one of the theorems of alternatives a good discussion of which is given by Mangasarian [19]. It can also be found in Dantzig and Thapa [9] and Bazaraa et al. [2], among others.

Lemma 1. *[**Farkas Lemma (1902)**] For a given matrix A and a vector C, either $AX \geq 0$ and $CX < 0$ has a solution, or there exists a vector like W such that $WA = C$ and $W \geq 0$.*

Theorem 1. *The optimal solution of Equation* (2) *is in the extended rectangular hull of demand points.*

Proof. First we define four points as follow:

$$\begin{aligned} RH_1 &= \min\{a_i - r_i | i = 1, 2, \cdots, n\} \\ RH_2 &= \max\{b_i + r_i | i = 1, 2, \cdots, n\} \\ RH_3 &= \max\{a_i + r_i | i = 1, 2, \cdots, n\} \\ RH_4 &= \min\{b_i - r_i | i = 1, 2, \cdots, n\}. \end{aligned}$$

We show that the optimal point is in the rectangular hull of these four points. By contradiction, we suppose that $X_b = (x_b, y_b)^T$ is the optimal point of model (2) and is out of rectangle. Therefor it is out of cone generated either by $(RH_1 - RH_3, 0)$ and $(0, RH_4 - RH_2)$ or $(RH_3 - RH_1, 0)$ and $(0, RH_2 - RH_4)$. **Case 1**: suppose that the optimal point is out of cone generated by $(RH_1 - RH_3, 0)$ and $(0, RH_4 - RH_2)$. So the vector $X_b - (RH_3, RH_2)^T$ is not in the cone, i.e. following system has not solution:

$$\begin{cases} v_1(RH_1 - RH_3, 0) + v_2(0, RH_4 - RH_2) = (x_b - RH_3, y_b - RH_2) \\ v_1, v_2 \geq 0. \end{cases}$$

By Farkas' lemma, the following system have solution:

$$\begin{cases} \begin{bmatrix} RH_1 - RH_3 & 0 \\ 0 & RH_4 - RH_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \geq 0 \\ \\ [x_b - RH_3, y_b - RH_2] \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} < 0. \end{cases} \tag{3}$$

So by the first inequality of (3) we have:

$$\begin{cases} (RH_1 - RH_3)u_1 \geq 0 \\ (RH_4 - RH_2)u_2 \geq 0 \end{cases}$$

since $RH_1 \leq RH_3$ and $RH_4 \leq RH_2$ therefore $u_1 \leq 0$ and $u_2 \leq 0$. And by the second inequality of (3) we obtain

$$(x_b - RH_3)u_1 + (y_b - RH_2)u_2 < 0.$$

Moreover $a_i \leq RH_3$ and $b_i \leq RH_2$ for $i = 1, 2, \cdots, n$ so
$(x_b - a_i)u_1 + (y_b - b_i)u_2 \leq (x_b - RH_3)u_1 + (y_b - RH_2)u_2 < 0.$

Now, if $x_b - RH_3 < 0$ and $y_b - RH_2 > 0$ then let $U = \begin{pmatrix} 0 \\ u_2 \end{pmatrix}$ and if $x_b - RH_3 > 0$ and $y_b - RH_2 < 0$ then let $U = \begin{pmatrix} u_1 \\ 0 \end{pmatrix}$ and if $x_b - RH_3 > 0$ and $y_b - RH_2 > 0$ then let $U = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$. So in each case

$$\nabla d(X_b, P_i).U = \frac{1}{d(X_b, P_i)}(x_b - a_i, y_b - b_i).U < 0.$$

Therefore

$$\begin{aligned} \nabla F(X_b).U &= 2\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X_b, P_i)}\right)(x_b - a_i)u_1 \\ &\quad + 2\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X_b, P_i)}\right)(y_b - b_i)u_2 \\ &= 2\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X_b, P_i)}\right)[(x_b - a_i)u_1 + (y_b - b_i)u_2] < 0. \end{aligned}$$

This means that U is a descent direction and X_b can not be optimal point. Note that in the last equality statement $1 - \frac{r_i}{d(X_b,P_i)}$ is positive for all i, because X_b is out of extended rectangular hull and $d(X_b, P_i) \geq r_i$ so $\frac{r_i}{d(X_b,P_i)} \leq 1$.

Case 2: suppose that optimal point is out of cone generated by $(RH_3 - RH_1, 0)$ and $(0, RH_2 - RH_4)$. Proof of this case is the same as Case 1. So the optimal point is in rectangular hull generated by RH_1, RH_2, RH_3 and RH_4. □

4 Weiszfeld-like algorithm

In this section, we propose an iterative algorithm for solving GSWLP. Our algorithm is based on the Weiszfeld method. The original Weiszfeld algorithm is an iterative method for solving the following Weber problem.

$$MinF(X) = \sum_{i=1}^{n} w_i \sqrt{(x - a_i)^2 + (y - b_i)^2}. \tag{4}$$

From the necessary condition of optimality $\frac{\partial F(X^*)}{\partial X} = 0$ for this problem, the Weiszfeld method starts with an initial solution and try to find optimal solution by the following iterative relations;

$$x_{k+1} = \frac{\sum_{i=1}^{n} \frac{w_i a_i}{\sqrt{(x_k - a_i)^2 + (y_k - b_i)^2}}}{\sum_{i=1}^{n} \frac{w_i}{\sqrt{(x_k - a_i)^2 + (y_k - b_i)^2}}}$$

$$y_{k+1} = \frac{\sum_{i=1}^{n} \frac{w_i b_i}{\sqrt{(x_k - a_i)^2 + (y_k - b_i)^2}}}{\sum_{i=1}^{n} \frac{w_i}{\sqrt{(x_k - a_i)^2 + (y_k - b_i)^2}}}.$$

Now consider the GSWLP. The same as Weiszfeld method from the necessary optimality condition to obtain a candidate for optimal solution we set

$$\frac{\partial F}{\partial X} = 2 \sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X, P_i)}\right) (X - P_i) = 0 \tag{5}$$

therefore

$$X = \frac{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X,P_i)}\right) P_i}{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X,P_i)}\right)}. \tag{6}$$

Like in the Weiszfeld algorithm, there is a possibility that during the iterations, $d(X, P_i)$ is zero for one of the points, and then the ratio is infinite and the process cannot be continued. For practical purposes, the whole issue

of singular points may be avoided by replacing each distance in (6) by its hyperbolic approximation (e.g., see [18]) given by:

$$X = \frac{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X,P_i)+\varepsilon}\right) P_i}{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X,P_i)+\varepsilon}\right)} \tag{7}$$

where $\varepsilon > 0$ is a smoothing constant. Therefor we can use the following iterative method.

Algorithm [WLA].
Initialization:

1. Choose an initial solution, X_0 and set $k = 0$.

Iteration step:
While not stopping criterion do

1.
$$x_{k+1} = \frac{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{\sqrt{(x_k-a_i)^2+(y_k-b_i)^2}+\varepsilon}\right) a_i}{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{\sqrt{(x_k-a_i)^2+(y_k-b_i)^2}+\varepsilon}\right)}$$
$$y_{k+1} = \frac{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{\sqrt{(x_k-a_i)^2+(y_k-b_i)^2}+\varepsilon}\right) b_i}{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{\sqrt{(x_k-a_i)^2+(y_k-b_i)^2}+\varepsilon}\right)}$$

2. k:=k+1

endwhile

The stopping criterion rule is usually one of the following criteria: maximum number of iterations or reaching a tolerance.

Now we discuss the convergency of WLA. From Equation (6) if we consider the right hand side of this equation as X_k and the left hand side of it as X_{k+1}, so we have

$$X_{k+1} = \frac{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X_k,P_i)}\right) P_i}{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X_k,P_i)}\right)}, \tag{8}$$

then by choosing an initial solution, X_0, for $k = 1, 2, \cdots$ we have

$$X_{k+1} = X_k - \frac{1}{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X_k,P_i)}\right)} \left[\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X_k,P_i)}\right)(X_k - P_i)\right]. \tag{9}$$

Let us assume that

$$d_k = -\frac{1}{2}\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X_k,P_i)}\right)(X_k - P_i) \tag{10}$$

it is clear that d_k is a descent direction, i.e. $d_k^T.\nabla F(X_k) < 0$, so the Equation (9) is a gradient method and the sequence $\{F(X_k)\}$ is non increasing. Let

$$h(X) = X - \frac{1}{2\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X,P_i)}\right)} \frac{\partial F(X)}{\partial X}. \tag{11}$$

The function $F(X)$ is two times differentiable and $h(X)$ in $X \neq P_i$, $i = 1, \cdots, n$, is continuous.

Lemma 2. *The sequence generated by Equation* (8) *is in affine hull of the* $P_i, i = 1, 2, \cdots, n$.

Proof. In each iteration, we have $X_{k+1} = \frac{\sum_{i=1}^{n} \alpha_i P_i}{\sum_{i=1}^{n} \alpha_i}$, in which $\alpha_i = \sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X_k,P_i)}\right)$. Let $\gamma_i = \frac{\alpha_i}{\sum_{i=1}^{n} \alpha_i}$, we have $\gamma_i \in [-1, 1]$ and $\sum_{i=1}^{n} \gamma_i = 1$. From Equation (8) the sequence $\{X_k\}$ is a affine combination of demand points. So the sequence $\{X_k\}$ is in affine hull of $P_i, i = 1, 2, \cdots n$. □

Lemma 3. *If* $X_k = X_{k+1}$ *then* $\frac{\partial F(X_k)}{\partial X} = 0$.

Proof. From the Equation (9), we have $X_{k+1} = h(X_k) = X_k - \frac{1}{2}\lambda_k \frac{\partial F(X_k)}{\partial X}$ where $\lambda_k = \frac{1}{\sum_{i=1}^{n} w_i \left(1 - \frac{r_i}{d(X_k,P_i)}\right)}$. If $X_{k+1} = X_k$ then $-\frac{1}{2}\lambda_k \frac{\partial F(X_k)}{\partial X} = 0$, i.e. $\frac{\partial F(X_k)}{\partial X} = 0$. □

Lemma 4. *If* $X_k \neq X_{k+1}$ *then* $F(X_{k+1}) < F(X_k)$, *and If* $F(X_k) = F(X_{k+1})$ *then* $X_k = X_{k+1}$.

Proof. From the Equation (9), if $X_k \neq X_{k+1}$, we have $\frac{\partial F(X_k)}{\partial X} \neq 0$. Let $D = -\frac{\partial F(X_k)}{\partial X}$ so $D^T.\nabla F(X_k) < 0$ i.e. $F(X_{k+1}) < F(X_k)$.
To show the second part of lemma if $F(X_k) = F(X_{k+1})$ then since for every k, the direction d_k is non increasing we have $X_k = X_{k+1}$. □

Note that the direction d_k in iteration k, is a descent direction, so the sequence $\{F(X_k)\}$ is non increasing. Since this sequence is bounded from below by zero, the limit $\lim_{k\to\infty} F(X_k)$ exists.

Theorem 2. *The sequence* $\{X_k\}$ *converges to a local minimum or a saddle point of function* $F(X)$.

Proof. By Lemma 2, the sequence $\{X_k\}$ is in affine hull of demand points and since by explanation of Equation (9), d_k^T are descent direction so the sequence $\{X_k\}$ is converged to a point like $\hat{X}$. Since $\lim_{k\to\infty} F(X_k)$ exists, $\lim_{k\to\infty} F(X_{k+1}) - F(X_k) = 0$ and $F(h(\hat{X})) = F(\hat{X})$. By Lemma 4, we have $h(\hat{X}) = \hat{X}$. If $X_{k+1} \neq X_k$ then the inequality $F(X_{k+1}) \leq F(X_k)$ holds strictly. From continuity of function $h(X)$, if X_k tends to $\hat{X}$ then X_{k+1} will tend to $h(\hat{X}) = \hat{X}$ and $\{X_k\}$ converges to $\hat{X}$. According to Lemma 3, $\frac{\partial F(\hat{X})}{\partial X} = 0$ and since $\{F(X_k)\}$ is non increasing and bounded from below by zero, the point $\hat{X}$ con not be local maxima, in other word, it is a local minimum or saddle point of function $F(X)$. □

We presented an analytic method for solving GSWLP that do not guarantee converging to a minimum and may fall in saddle point or local minima which is no minimum. Since the objective function of GSWLP is non convex, the local minimum may not be global (see Example 1). So we should use a global optimization method. In the next section we apply a particle swarm optimization algorithm to find the best solution of problem.

Example 1. Consider the points, their weights and their radiuses that are given in Table 1. This problem is solved in [12] by big square small square method and the value of optimal solution is 182. But if we solve this problem by WLA we will see this method convergence to a point which its value of objective function is 209.

Table 1: Data for 18 points problem.

(a, b, w, r)	(a, b, w, r)	(a, b, w, r)
$(1, 2, 3, 2)$	$(4, 4, 1, 2)$	$(7, 1, 2, 1)$
$(1, 3, 2, 2)$	$(4, 9, 2, 1)$	$(7, 2, 3, 2)$
$(2, 5, 1, 3)$	$(5, 3, 2, 2)$	$(8, 5, 1, 2)$
$(3, 6, 3, 2)$	$(5, 5, 1, 2)$	$(8, 8, 3, 1)$
$(4, 8, 2, 2)$	$(6, 6, 3, 3)$	$(9, 7, 3, 2)$
$(4, 1, 3, 1)$	$(6, 3, 3, 1)$	$(9, 6, 2, 3)$

Note that this is an special example that WLA traps in a local minima. This case rarely happens. We examined many other test problems but we could not find any example with this property. Also as we shall see in the computational experiments section, in non of the examined test problems in this section this case dose not happen.

5 Particle Swarm Optimization

Nature has always been an inspiration for researchers in designing solutions for many problems. Particle Swarm Optimization (PSO) is a meta-heuristic algorithm that is inspired of the behaviors of social models like bird flocking or fish schooling. It is introduced by Eberhart and Kennedy [11], as an optimization method for continuous nonlinear functions. Later, it has been applied to wide range of problems due to its conceptual and implementation simplicity (see e.g. [1, 15]).

Particle swarm optimization is a stochastic optimization technique based on individual improvement, social cooperation and competition in a population. PSO is famous for its fast convergence to a solution close to the optimal, by balancing the local search and global search i.e., exploitation and exploration, respectively. PSO is a population based method in which a swarm includes several individuals called particles. Each particle has a position and velocity vector. Particles are initially positioned randomly. The position of a particle is a candidate solution and it is updated at each iteration by using the particle's current velocity. The velocity of a particle is updated by using the particle's inertia weight as well as the social interaction (its tendency to follow the best direction experienced by the group) and personal experience (the best direction experienced by itself) of the particle. In the course of several iterations, particles make use of this experience and are supposed to move towards the optimum position.

The stopping criterion rule is usually one of the following criteria: maximum number of iterations, the maximum CPU time, the number of successive best objective function values without any improvements or reaching a predetermined tolerance. During the update step, for each particle, the velocity and position vector of the particle at iteration $t+1$ are calculated by using the Equations (12) and (13).

$$v_{i,j}^{t+1} = wv_{i,j}^{t} + c_1 r_1 \left(pbest_{i,j}^{t} - x_{i,j}^{t}\right) + c_2 r_2 \left(gbest_{j}^{t} - x_{i,j}^{t}\right) \tag{12}$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1}. \tag{13}$$

In these equations, $v_{i,j}^{t}$ and $x_{i,j}^{t}$ are the velocity and position of the j^{th} dimension of the i^{th} particle at iteration t, respectively. The parameters c_1 and c_2 are coefficients of learning factors, which are the weights of contributions of personal experience and social interaction. The stochastic behavior of PSO is achieved by r_1 and r_2 which are random numbers, generally in $(0, 1)$. The parameter w is the inertia weight which is the balancing factor between exploration and exploitation. Small inertia weight facilitates more exploitation while large inertia weight enables more exploration.

After the update step, the fitness function value is calculated for each particle based on its position. The local best position *pbest* of each particle and

the global best position *gbest* are updated using these fitness values as follow:

$$pbest_i^{t+1} = \begin{cases} pbest_i^t & F(x_i^{t+1}) \geq F(pbest_i^t) \\ x_i^{t+1} & F(x_i^{t+1}) > F(pbest_i^t) \end{cases} \tag{14}$$

$$gbest^{t+1} = argmin_{i\in\{1,2,\cdots,n\}} F(pbest_i^{t+1}). \tag{15}$$

These ideas lead to the following algorithm.

Algorithm [PSO].

Initialization:

1. Initialize iteration counter.
2. Initialize N random position of particles and store them in S.
3. Initialize N random velocities and store them in V.
4. Initialize N *pbest* and store them in P.
5. Set $gbest^t$ equal the best *pbest* in P.

Iteration step:
While not stopping criterion do

1. For each i^{th} particle:
   (a) Update V: Calculate velocity v_i^{t+1} using (12)
   (b) Update S: Calculate position x_i^{t+1} using (13)
   (c) Update P: Calculate position $pbest_i^{t+1}$ using (14)
2. Update gbest: $gbest^{t+1} = \arg\min_{i\in\{1,2,\cdots,N\}} \left\{f(pbest_i^{t+1})\right\}$
3. t:=t+1

endwhile

Where V is the set of velocities and P is the set of pbests. To apply this algorithm for Goal Square Weber Location Problem we set S the region of optimal solution, i.e. extended rectangular hull of points and each particle as a solution. In this algorithm first we choose N random positions and velocities for each particle. From rules of particle swarm optimization we obtain the pbest and gbest of swarm. The algorithm with updating variables continues until termination conditions reached.

Many modification for PSO algorithm has been proposed to improve the performance of this algorithm. In standard PSO algorithm, the information

of individual best and global best are shared by next generation particles. Shi and Eberhart [21] use dynamic inertia weight that decreases according to iterative generation increasing. The weight w in (12) follows the relation $w_{t+1} = \alpha w_t$ where $0 < \alpha < 1$. A large inertia weight facilitates global exploration and a smaller inertia weight tends to facilitate local exploration to search the local neighborhoods in the current search area more effectively.

Clerc and Kennedy [7] present another good modification of PSO. They consider the form of a constriction coefficient K which controls all the three components in velocity update rule. This has an effect of reducing the velocity as the search progress. In this modification, the velocity update is given as:

$$v_{i,j}^{t+1} = K\left(v_{i,j}^{t} + c_1 r_1\left(pbest_{i,j}^{t} - x_{i,j}^{t}\right) + c_2 r_2\left(gbest_{j}^{t} - x_{i,j}^{t}\right)\right) \qquad (16)$$

where

$$K = \frac{2}{\mid 2 - \phi - \sqrt{\phi^2 - 4\phi} \mid} \qquad (17)$$

where $\phi = c_1 + c_2 > 4$. This version is referred to as PSO with constriction or PSOC. We apply the original PSO and PSOC for solving GSWLP.

In the next section we compare the WLA with PSO, PSO with constriction (PSOC) and big square small square (BSSS) method. Big Square Small Square (BSSS) method is a geometrical branch and bound algorithm originally suggested by Hansen et al. [13] for solving an obnoxious facility location problem. The idea is to divide the plane into regions (squares), over each of which a lower (upper) bound for the problem is found. If the objective over a given square is worse than an existing upper (lower) bound, then that square is fathomed. The procedure continues until a prescribed tolerance is achieved. This method also applied by some authors such as Plastria [20], Skriver and Andersen [22], Zaferanieh et al. [25] and Fathali et al. [12] for solving location problems.

6 Computational results

The proposed algorithms were tested on forty test problems with 100 to 3000 points that are generated randomly in MATLAB. In these problems demand point's coordinates, weights and radiuses, all are positive numbers and generated randomly in $[1, 60]$, $[1, 3]$ and $[1, 10]$, respectively.

We compare the Weiszfeld like algorithm (WLA), PSOC and PSO with BSSS method of Fathali et al. [12]. Algorithms are coded in MATLAB software and run on a Laptop with Core(TM)2 CPU with 2.00 GHz processor and 1 GB of RAM. In the PSO and PSOC algorithms we should generate randomly N initial position of particles and velocities. Therefore in different executions of these methods we may find different solutions, so we run 5

times this two methods for all problems and report the average results. In all methods the iteration step of the algorithm is repeated until the tolerance is achieved. The tolerance for all examples was taken to be 0.01.

In our experiments with different values of c_1 and c_2 ($c_1, c_2 \in [1, 3]$) we found the best values to be $c_1 = c_2 = 2.1$. The inertia weight is started with $w = 1.5$ and decrease in each iteration such that reach 0.2 at the end of algorithm. The population size is set $N = 50$ for problems with $n = 100, \cdots, 500$ and $N = 100$ for remaining problems. We also examined larger numbers of population size. Our tests showed that larger population size slow down the algorithm but did not lead to better solutions.

Table 2 shows the computational results. The column with the heading *%Error* indicates the relative error; i.e.,

$$|\frac{f - f_{best}}{f_{best}}| \cdot 1000$$

where f_{best} is the best solution obtained by the four methods and f is the value of objective function obtained by each method.

Note that the BSSS method continues until the side of the sub-square is less than a given tolerance. So the solutions which obtained by this method is near optimal with the given tolerance. By this method we can also compute a lower bound for the objective for each sub-square. The column with the heading LB in Table 2 indicates the lower bound for the objective for the best sub-square. This sub-square contains optimal solution. Also the column with the heading *gap* shows the relative gap between the lower bound and the best solution, i.e.,

$$|\frac{f_{best} - LB}{f_{best}}| \cdot 1000.$$

We observe the largest relative gap is 5.79×10^{-4}, which happens for the test problem number 23.

Table 3 contains the average total CPU times for test problems. The CPU times of WLA are less than other methods in all cases. However the other methods could find better solution in some test problems.

To compare speed of presented methods one may consider the time complexity of these methods. However since these methods are improvement methods then with more iterations we may obtain better solutions. Therefore in the following we give their time complexity per iteration.

The main step of WLA is computing x and y in each iteration. Since we need $23n$ operations for each of x and y, obviously, the time complexity per iteration is $O(n)$. Where n is number of existing points.

In step 1 of each iteration of PSO, we should calculate velocity v, position of x and *pbest* according to (12), (13) and (14), respectively. They need $O(nN)$ operations. Where N is the population size. Step 2 of Iteration step

Table 2: The objective functions and errors

		objective function					*%Error*				
NO.	n	LB	WLA	BSSS	PSO	PSOC	WLA	BSSS	PSO	PSOC	gap
1	100	59599	59617	59618	59617	59617	0.000	0.017	0.000	0.000	0.302
2	100	68958	68995	68995	68995	68995	0.000	0.000	0.000	0.000	0.536
3	100	82265	82287	82287	82288	82287	0.000	0.000	0.012	0.000	0.267
4	100	78156	78232	78180	78182	78180	0.665	0.000	0.026	0.000	0.435
5	100	76104	76145	76145	76145	76145	0.000	0.000	0.000	0.000	0.539
6	200	121305	121340	121341	121341	121340	0.000	0.008	0.008	0.000	0.288
7	200	166160	166254	166254	166255	166254	0.000	0.000	0.006	0.000	0.565
8	200	159632	159720	159720	159720	159721	0.000	0.000	0.000	0.006	0.551
9	200	162253	162312	162312	162312	162312	0.000	0.000	0.000	0.000	0.363
10	200	160276	160324	160324	160325	160324	0.000	0.000	0.006	0.000	0.299
11	300	233660	233765	233766	233765	233766	0.000	0.004	0.000	0.004	0.449
12	300	234183	234314	234314	234314	234315	0.000	0.000	0.000	0.004	0.559
13	300	245908	246045	246045	246046	246045	0.000	0.000	0.004	0.000	0.557
14	300	241698	241761	241761	241761	241761	0.000	0.000	0.000	0.000	0.267
15	300	235575	235674	235621	235642	235621	0.225	0.000	0.089	0.000	0.195
16	400	309670	309777	309777	309777	309776	0.003	0.003	0.003	0.000	0.342
17	400	337358	337543	337544	337543	337543	0.000	0.003	0.000	0.000	0.548
18	400	319847	320030	320030	320031	320032	0.000	0.000	0.003	0.003	0.493
19	400	326612	326734	326735	326734	326735	0.000	0.003	0.000	0.003	0.373
20	400	318651	318783	318783	318783	318783	0.000	0.000	0.000	0.000	0.389
21	500	394201	394364	394364	394364	394364	0.000	0.000	0.000	0.000	0.418
22	500	391482	391654	391598	391597	391598	0.143	0.003	0.000	0.003	0.294
23	500	407515	407751	407752	407751	407751	0.000	0.003	0.000	0.000	0.579
24	500	412614	412763	412763	412763	412763	0.000	0.000	0.000	0.000	0.361
25	500	402198	402348	402277	402277	402277	0.176	0.000	0.000	0.000	0.196
26	1000	786520	786731	786732	786732	786732	0.000	0.001	0.001	0.001	0.268
27	1000	796696	797148	797149	797148	797148	0.000	0.001	0.000	0.000	0.191
28	1000	812939	813407	813406	813407	813406	0.001	0.000	0.001	0.000	0.575
29	1000	805501	805634	805634	805634	805634	0.000	0.000	0.000	0.000	0.165
30	1000	802280	802431	802432	802431	802432	0.000	0.001	0.000	0.001	0.313
31	2000	1465017	1465731	1465705	1465718	1465705	0.018	0.000	0.009	0.000	0.469
32	2000	1582076	1582976	1582977	1582977	1582977	0.000	0.001	0.001	0.001	0.569
33	2000	1633068	1634000	1634000	1634000	1634000	0.000	0.000	0.000	0.000	0.570
34	2000	1563967	1564236	1564185	1564186	1564185	0.001	0.000	0.001	0.000	0.139
35	2000	1607103	1607640	1607640	1607641	1607640	0.000	0.000	0.001	0.000	0.334
36	3000	2297871	2298872	2298873	2298872	2298873	0.000	0.001	0.000	0.001	0.435
37	3000	2362318	2363663	2363663	2363663	2363663	0.000	0.000	0.000	0.000	0.569
38	3000	2403658	2405041	2405040	2405040	2405040	0.001	0.000	0.000	0.000	0.575
39	3000	2387987	2388745	2388746	2388745	2388753	0.000	0.001	0.000	0.003	0.317
40	3000	2324214	2325395	2325396	2325396	2325395	0.000	0.001	0.001	0.000	0.508
Total							1.233	0.051	0.172	0.030	

Table 3: The CPU Time of PSOC, PSO, WLA and BSSS algorithms

n	WLA	BSSS	PSO	PSOC
100	0.002	4.204	0.630	0.778
200	0.004	7.765	1.145	1.318
300	0.005	11.363	1.686	1.852
400	0.007	15.061	2.192	2.920
500	0.008	18.205	2.709	3.470
1000	0.014	83.607	5.348	5.929
2000	0.027	389.972	10.570	12.346
3000	0.040	576.231	15.750	17.230

Table 4: The objective functions and errors in a constant time

			$objective function$			$\%Error$		
test#	n	time	BSSS	PSO	PSOC	BSSS	PSO	PSOC
1	100	0.5	59630	59617	59617	0.218	0.000	0.000
2	100	0.5	68997	68996	68995	0.028	0.014	0.000
3	100	0.5	82290	82289	82287	0.036	0.024	0.000
4	100	0.5	78241	78205	78202	0.780	0.320	0.281
5	100	0.5	76147	76146	76145	0.026	0.013	0.000
6	200	1.0	121355	121345	121343	0.124	0.041	0.025
7	200	1.0	166799	166263	166260	3.278	0.054	0.036
8	200	1.0	160699	159771	159730	6.129	0.319	0.063
9	200	1.0	162330	162318	162314	0.111	0.037	0.012
10	200	1.0	160345	160325	160324	0.131	0.006	0.000
11	300	1.5	233809	233766	233767	0.188	0.004	0.008
12	300	1.5	234316	234316	234316	0.008	0.008	0.008
13	300	1.5	246072	246047	246045	0.110	0.008	0.000
14	300	1.5	241763	241770	241761	0.008	0.037	0.000
15	300	1.5	235704	235664	235651	0.352	0.182	0.127
16	400	2.0	309790	309779	309780	0.039	0.009	0.012
17	400	2.0	337568	337543	337543	0.075	0.000	0.000
18	400	2.0	320116	320090	320076	0.258	0.180	0.138
19	400	2.0	326814	326784	326784	0.248	0.153	0.153
20	400	2.0	318853	318783	318783	0.220	0.000	0.000
21	500	2.5	394384	394366	394365	0.061	0.005	0.003
22	500	2.5	391755	391598	391599	0.403	0.003	0.005
23	500	2.5	407801	407751	407751	0.151	0.000	0.000
24	500	2.5	412833	412772	412763	0.211	0.027	0.000
25	500	2.5	402451	402279	402280	0.433	0.006	0.009
26	1000	5.0	786851	786742	786733	0.153	0.011	0.002
27	1000	5.0	797153	797149	797149	0.004	0.001	0.001
28	1000	5.0	813407	813410	813407	0.001	0.004	0.001
29	1000	5.0	805710	805642	805639	0.094	0.008	0.005
30	1000	5.0	802671	802431	802433	0.299	0.000	0.002
31	2000	10.0	1465978	1465720	1465711	0.186	0.010	0.004
32	2000	10.0	1583288	1582977	1582977	0.197	0.001	0.001
33	2000	10.0	1634121	1634004	1634000	0.121	0.004	0.000
34	2000	10.0	1564480	1564186	1564186	0.189	0.001	0.001
35	2000	10.0	1607871	1607665	1607660	0.144	0.025	0.020
36	3000	15.0	2299001	2298876	2298874	0.056	0.004	0.002
37	3000	15.0	2363820	2363663	2363663	0.066	0.000	0.000
38	3000	15.0	2405131	2405041	2405041	0.037	0.001	0.000
39	3000	15.0	2388876	2388754	2388759	0.055	0.004	0.006
40	3000	15.0	2325480	2325396	2325395	0.037	0.001	0.000
			Total			15.265	1.525	1.04

calculate *gbest* which needs $O(N)$ operations. Therefor per iteration of PSO has $O(nN)$ time complexity.

PSOC is the same as PSO just we need calculate velocity v by (16). It also need $O(nN)$ operations. Therefor the time complexity of PSOC is $O(nN)$ for per iteration.

Due to [12] in each iteration of BSSS we should calculate the objective function for a given point in a square. This take $O(n^2)$ time, therefor each iteration of BSSS has $O(n^2)$ time complexity.

To compare the algorithms on equal terms, we allow the BSSS, PSO and PSOC algorithms run during the same time. Table 4 contains computational results for this case. Since WLA is very fast, its results for the given times in Table 4 is the same as Table 2.

7 Summary and conclusion

In this paper we studied a new version of the single facility location problem in which we want to find location of a new facility such that the sum of weighted squared errors is minimized. In general, this problem is non convex and we

showed that optimal solution of problem is in extended rectangular hull of demand points. We proposed two approaches for solving this problem. An iterative Weiszfeld-like procedure and two modifications of PSO algorithm. We compared the results with those obtained by BSSS algorithm. It was shown that for almost all problems the PSOC outperforms the other three approaches.

References

1. Ali, M.M. and Kaelo, P. *Improved particle swarm algorithm for global optimization,* Applied Mathematics and Computation, 2008, 196: 578-593.

2. Bazaraa, M.S., Sherali, H. and Jarvis, J. *Linear Programming and Network Flows*, Third ed., John Wiley & Sons; 2005.

3. Berman, O. and Huang, R. *The minimum weighted covering location problem with distance constraints,* Computers and Operations Research, 2008, 35: 356-372.

4. Brimberg, J. *The Fermat-Weber location problem revisited,* Mathematical Programming, 1995, 71: 71-76.

5. Chen, R. *Noniterative Solution of Some Fermat-Weber Location Problems,* Advances in Operations Research, Volume 2011 (2011), Article ID 379505, 10 pages.

6. Church, R. and ReVelle, C. *The maximal covering location problem,* Papers of Regional Science Association, 1974, 32: 101-18.

7. Clerc, M. and Kennedy, J. *The particle swarm-explosion, stability, and convergence in a multidimensional complex space,* IEEE Transactions on Evolutionary Computation, 2002, 1: 58-73.

8. Courant, R. and Robbins, H. *What is mathematics?* Oxford: Oxford University Press, 1941.

9. Dantzig, G.B. and Thapa, M.N. *Linear Programming 2: Theory and Extensions*. Springer; 2003.

10. Drezner, Z. *On convergence of the generalized Weiszfeld algorithm,* Ann Oper Res. 2008; 167: 327-336.

11. Eberhart, R.C. and Kennedy, J. *A new optimizer using particle swarm theory,* proc. 6th int. Symp. Micro Machine and Human science, Nagoya, Japan, 1995; 39-43.

12. Fathali, J., Zaferanieh, M. and Nezakati, A. *A BSSS algorithm for the location problem with minimum square error,* Advances in Operations Research, Volume 2009 (2011), Article ID 212040, 10 pages.

13. Hansen, P., Peeters, D. and Thisse, J.F. *On the location of an obnoxious facility,* Sistemi Urbani, 1981; 3: 299-317.

14. Jamalian, A. and Fathali, J. *Linear programming for the location problem with minimum absolute error,* World Applied Sciences Journal 2009, 7: 1423-1427.

15. Jin, Y.X., Cheng, H.Z., Yan, J. and Zhang, L. *New discrete method for particle swarm optimization and its application in transmission network expansion planning,* Electric Power Systems Research, 2007, 77: 227-233.

16. Kuhn, H.W. *On a pair of dual nonlinear programs. In: Nonlinear Programing,* J.Abadie (eds.), North-Holand Publishers Co. Amsterdam, 1967, 37-54.

17. Kuhn, H.W. and Kuenne, R.E. *An efficient algorithm for the Numerical Solution of the Generalized Weber Problem in spatial Economics,* Journal of Regional Science, 1967; 4: 21-33.

18. Love, R.F., Morris, J.G. and Wesolowsky, G.O. *Facility Location :models and methods.*, North-Holland,(1988).

19. Mangasarian, O.L. *Nonlinear Programming.* McGraw-Hill; 1969.

20. Plastria, F. *GBSSS: the generalized big square small square method for planar single-facility location,* European Journal of Operational Research, 1992; 62: 163-174.

21. Shi, Y. and Eberhart, R.C. *Empirical study of particle swarm optimization,* IEEE International Congress on Evolutionary Computation, 1999, 3: 101-106.

22. Skriver, A.J.V. and Andersen, K.A. *The bicriterion semi-obnoxious location (BSL) problem solved by an ϵ-approximation*, European Journal of Operational Research, 2003; 146: 517-528.

23. Trinh, M.H., Lee, B.H. and Ahn, H.S. *The Fermat-Weber location problem in single integrator dynamics using only local bearing angles,* Automatica, 2015; 59: 90-96.

24. Weiszfeld, E. *Sur Le Point Pour Lequel La Somme Des Distances De N Points Donnes Est Minimum,* Tohoku Mathematical Journal, 1937; 60: 355-386.

25. Zaferanieh, M., Kakhki, H.T., Brimberg, J. and Wesolowsky, G.O. *A BSSS algorithm for the single facility location problem in two regions with different norms,* European Journal of Operational Research, 2008; 190: 79-89.

مساله مکانیابی وبر مربعی آرمانی

جعفر فتحعلی [۱] *و علی جمالیان* [۲]

[۱] دانشگاه صنعتی شاهرود، دانشکده علوم ریاضی
[۲] دانشگاه گیلان، دانشکده علوم ریاضی

دریافت مقاله ۲۳ دی ۱۳۹۴، دریافت مقاله اصلاح شده ۲۹ خرداد ۱۳۹۵، پذیرش مقاله ۲۴ شهریور ۱۳۹۵

چکیده : دراین مقاله به حالت خاصی از مساله مکانیابی وبر می پردازیم که آن را مساله مکانیابی آرمانی می نامیم. در مساله مکانیابی وبر هدف پیدا کردن نقطه ای در صفحه به گونه ای است که مجموع وزنی فاصله ها بین این نقطه و n نقطه موجود کمینه شود. در مساله مکانیابی آرمانی هر نقطه موجود مانند P_i یک شعاع متناظر r_i دارد و حالت ایده آل وقتی است که سرویس دهنده جدید در فاصله r_i از نقطه P_i به ازای $i = ۱, ۲, \cdots n$ قرار گیرد. اما واضح است که در اغلب مسائل چنین نقطه ای ممکن است وجود نداشته باشد. لذا ما سعی میکنیم مجموع وزنی خطای مربعی از نقطه ایده آل را کمینه کنیم. در این مقاله فاصله ها با نرم اقلیدسی درنظر گرفته می شوند . الگوریتمی شبیه روش وایزفلد و دو الگوریتم به روش بهینه سازی پرندگان برای مساله ارائه کرده و نتایج حاصل از آنها را با روش مربع بزرگ، مربع کوچک مقایسه می کنیم.

کلمات کلیدی : نظریه مکانیابی؛ الگوریتم وایزفلد؛ بهینه سازی پرندگان .