Responses to the Comments of Reviewers
Manuscript #: IJSI-151627
Title: Two swarm intelligence based approaches for the *p*-center problem
Authors: -

First of all, we would like to express our gratitude to the two reviewers for their valuable comments and suggestions which helped in improving the quality of this manuscript.

The whole manuscript has been revised as per the suggestions of the reviewers. For the sake of clarity, each specific comment is presented in a box and the corresponding response is reported below the box.

Responses to the Comments of Reviewer A

Paper presents the solution of the p-centre problem using ABC and invasive weed optimization. Paper is well written. However, since an available problem is solved using two popular and well-established methods. Therefore, I find a lack of novelty in the work for a research paper. I suggest the authors to propose an improvement in ABC and/or IWO and then present the results of the problem.

Novelty of our work lies in adaptation of ABC and IWO algorithms to the p-center problem and none of our two approaches can be regarded as straightforward implementations of corresponding basic algorithms. We have incorporated suitable problem specific knowledge in neighboring solution generation method of ABC algorithm and seed production method of IWO algorithm. This problem specific knowledge is critical for the success of these two approaches. Further, there are two methods for neighboring solution generation/seed production in ABC/IWO algorithm which are used in a mutually exclusive manner to maintain a balance between diversity and quality.

Responses to the Comments of Reviewer B

This paper is suitable for publication.

No response needed.

Two swarm intelligence based approaches for the p-center problem

Abstract: The p-center problem is an important facility location problem. In this problem, the objective is to find a set Y of p vertices on an undirected weighted graph $G = (V, E)$ in such a way that $Y \subseteq V$ and the maximum distance over all the distances from vertices to their closest vertices in Y is minimized. The vertices in set Y are called centers. In this paper, we have proposed two swarm intelligence based approaches for the p-center problem. The first approach is based on artificial bee colony (ABC) algorithm, whereas the latter approach is based on invasive weed optimization (IWO) algorithm. The ABC algorithm and IWO algorithm are relatively new meta-heuristic techniques inspired respectively from collective intelligent behavior shown by honey bees while foraging and the sturdy process of weed colonization & dispersion in an ecosystem. Computational results on the well-known benchmark instances of p-center problem show the effectiveness of our approaches in finding high quality solutions.

Keywords: Artificial bee colony algorithm; facility location problem; invasive weed optimization algorithm; p-center problem; swarm Intelligence.

1 Introduction

Given an undirected weighted graph $G = (V, E)$ where V denotes the set of vertices and E denotes the set of edges and $|V| = n$, the p-center problem seeks on this graph a set $Y \subseteq V$ of p vertices so that the maximum distance over any vertex in V and its nearest vertex in Y is minimized. The vertices in set Y are called centers. The vertices of the graph can be considered as demand points and the vertices in Y as the location of facilities, the goal of p-center problem is to choose the locations of p facilities to serve n demand points, so that the maximum distance of any demand point from its nearest facility becomes as small as possible. We have used the vertices in set Y, centers and facility locations interchangeably throughout this paper. The p-center problem can be used in a variety of real life applications such as locating fire stations, police stations, or other emergency facilities etc. The p-center problem has attracted increasing attention in recent years.

The p-center problem is proven to be $\mathcal{NP}$-hard in the literature by Kariv and Hakimi [1]. Chen and Chen [2] solved the p-center problem using a new relaxation algorithm. Exact algorithms have been proposed for some special cases of p-center problem in [3], [4] and [5]. Mladenović et al. [6] presented two tabu search heuristics and a variable neighborhood search heuristic for the p-center problem. Caruso et al. [7] developed another heuristic approach called Dominant which solves a series of set-covering problems according to a pre-defined maximum distance to get a solution to the p-center problem. Pacheco and Casado [8] proposed a new approach based on scatter search. Davidović et al.[9] proposed an approach called

BCOi based on bee colony optimization (BCO) with improvement concept. Till date BCOi is the best approach for the p-center problem. Several other heuristics and metaheuristiscs have also been proposed in the literature, e.g. [10],[3],[11] and [12].

In this paper, we have proposed two new swarm intelligence based approaches for the p-center problem. Our first approach is based on artificial bee colony (ABC) algorithm, whereas the latter approach is based on invasive weed optimization (IWO) algorithm. ABC and IWO algorithms are new population based metaheuristic techniques. ABC algorithm proposed by Karaboga [13] is based on intelligent foraging behaviour of honey bee swarm, whereas IWO algorithm proposed by Mehrabian and Lucas [14] is based on colonizing behavior and surviving ability of weeds. We have compared our approaches with BCOi the best approach available in the literature on the standard benchmark instances for the problem. Computational results show the effectiveness of proposed approaches in finding high quality solutions.

The remaining part of this paper is structured as follows: A formal definition of the p-center problem is presented in Section 2. Section 3 provides an overview of the artificial bee colony algorithm. Section 4 describes our ABC approach to the p-center problem. Section 5 provides an introduction to the invasive weed optimization algorithm, whereas Section 6 describes our IWO approach to the p-center problem. Computational results and their analysis are presented in Section 7. Finally, Section 8 outlines some concluding remarks and directions for future research.

2 Formal problem definition

The p-center problem can be defined formally as follows: Let $G = (V, E)$ be an undirected graph with vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and edge set E. The length of shortest path or distance from vertex v_i to vertex v_j is denoted as $d(v_i, v_j)$. The problem is to choose a set Y containing p vertices of G, in such a way that the maximum distance of a vertex in V to its closest vertex in Y is minimized, i.e., the solution is a subset $Y = \{y_1, y_2, \ldots, y_p\}$ of V that minimizes

$$\max_{i\in\{1,\ldots,n\}} \left\{ \min_{j\in\{1,\ldots,p\}} d(v_i, y_j) \right\} \tag{1}$$

Figure 1 illustrates a typical solution of p-center problem with $p = 3$ facilities and $n = 15$ demand points. In this figure, each demand point is shown by circles (black or white). The demand points which are also the centres or the location of facilities are shown by white circles, whereas the remaining demand points are shown by black circle. The demand points are connected to their closest centers and this connection is shown by straight lines. The objective function value for this solution corresponds to distance between node 14 and center 13 which is the maximum distance over all the distances between a demand point and its closest center, i.e., $f = d(14, 13)$.

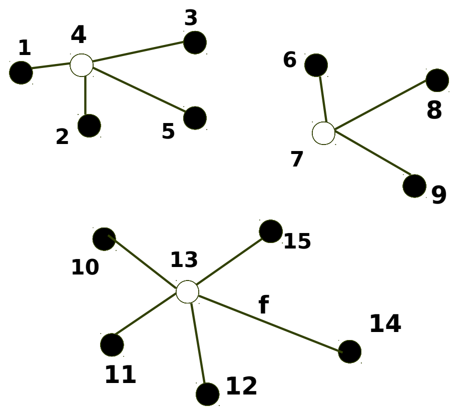

Figure 1 Illustrating p-center

3 Overview of ABC algorithm

The artificial bee colony (ABC) algorithm is a population based metaheuristic developed by Karaboga in 2005 [13]. Its a relatively new metaheuristic technique inspired from the collective intelligent foraging behavior of honey bees. There are three different kinds of bees available in a colony of bees: employed, onlooker and scout. Employed bees are currently tapping the food sources. They collect the nectar from the food sources and bring it to the hive. They also share the information about their food sources with other bees waiting in the hive. Onlookers, the waiting bees in the hive use the information shared by several employee bees, and then choose a food source with a probability directly proportional to its quality to become employed. Scout bees ferret for new food sources in the vicinity of the hive. As soon as they find a new food source, they start exploiting it and becomes employed. An employed bee whose food source becomes empty will become either scout or onlooker by abandoning that food source. Comparing this foraging behavior of bees around the hive with a metaheuristic search process for an optimal solution in a search space, we can say that employed and onlooker bees are doing the job of exploitation, whereas scout bees are responsible for exploration.

This foraging behavior of the real honey bees inspired Karaboga to develop ABC algorithm [13]. This algorithm was originally intended for solving continuous optimization problems. Later, it was extended to solve discrete optimization problems also [15]. Now, numerous variants of ABC algorithm exist in literature, e.g [13, 16, 17, 18, 15, 19, 20, 21, 22, 23, 24]. For a good survey on ABC algorithm and its applications, one may refer to [25].

ABC algorithm also have same three kinds of bees, viz. employed, onlooker and scout with functions analogous to their real counterparts. In this algorithm, a food source represent a solution to the problem under consideration and the quality of a food source represents the fitness of the solution being represented by that food source. There is a one-to-one correspondence between the employed bees and food sources, i.e., there are only as many employed bees as there are food sources. Usually, the number of onlooker bees is also same as the number of employed bees, but it can be different also. The employed bee associated with an exhausted food source always becomes a scout, but never an onlooker. This scout is immediately made employed by generating a new food source for it and associating it with this newly generated food source. We have used solution and food source interchangeably throughout this paper in the context of ABC algorithm. The ABC algorithm starts with initialization, then iterates through the cycles of the employed bee and onlooker bee phases until some stopping criteria is met.

During initialization, each employed bee gets associated with a solution which is usually generated randomly. In the employed bee phase, a new food source is determined by each employed bee in the vicinity of its associated food source and its quality is evaluated. The exact method through which a new food source is determined depends on the problem under consideration. If the quality of the new food source determined by the employed bee is better than the old one then it proceeds to the new food source abandoning the old one. If the quality of the new food source is not as good as the current one then the employed bee remains at the old food source. After all the employed bees have completed this process, the onlooker bee phase starts.

In the onlooker bee phase, onlookers select food sources stochastically based on their quality. The selection policy used by onlookers is such that a good food sources will have more chances of getting selected. The onlookers determine the new food sources in the vicinity of their selected food sources in a manner analogous to the employed bees. The best foodsource among all the new food sources generated by the onlookers associated with foodsource i and the food source i itself is determined. This best food source will be the new location of food source i in the next iteration. Once all food sources are updated, the onlooker bee phase ends.

The algorithm repeats itself through the cycles of the above mentioned two phases, until the termination condition is met. If a food source is not improved for some specified number of iterations, then that food source is discarded by its associated employee bee and it becomes scout. This scout is immediately transformed into employed bee by associating it with a newly generated solution. Such new solutions are generated usually in the manner analogous to the initial employee bee solutions.

In the ABC algorithm, each solution is given a chance to improve itself during employed bee phase. On the other hand, in the onlooker bee phase, good quality solutions will have more chance to improve themselves in comparison to poor quality solutions, as more number of onlookers gets attracted towards good quality solutions because of biased selection policy used in this phase. This bias towards good quality solutions may produce better quality solutions faster, as the chances of finding even better quality solutions in the neighborhood of good quality solutions are higher in comparison to solutions of inferior quality. However, if a solution is locally optimal, then any attempt to improve it is futile as no better quality solution exists in its neighborhood. The concept of scout bees helps in getting rid of this situation. Instead of determining whether a solution is locally optimal or not with respect to the whole neighborhood which can be a computationally expensive process, a solution is deemed to be locally optimal if has not improved over certain number of iterations. This solution is discarded by making its associated employed bee a scout. A new solution is generated for this scout bee to make it employed again. Hence, the solutions which has not improved since long and which can be locally optimal are discarded utilizing the concept of scout bees. For a search process to be robust, the balance between the exploration and exploitation must be maintained. To maintain a proper balance between exploration and exploitation in the ABC algorithm, the number of iterations without improvement in the quality of a solution after which its associated employed bee leaves it and becomes a scout needs to be set appropriately.

4 ABC approach for the p-center problem

This section describes our ABC approach for the p-center problem. The salient features of our ABC approach are discussed below in subsequent subsections.

4.1 Solution representation and fitness

A solution is represented by the subset of p vertices used for locating the facilities.The demand points are always assigned to the nearest facility present in the subset. For example, the solution in Figure 1 is represented as $Y = \{4, 7, 13\}$. The objective function is used as the fitness function, i.e., the fitness function value is calculated using the equation (1). As p-center is a minimization problem, a more fit solution has a lesser fitness function value.

4.2 Food source selection for onlooker bees

In order to select a food source for an onlooker bee, we have made use of the binary tournament selection method. In this selection method, two food sources are selected uniformly at random and a comparison is made between their fitnesses. The better of the two food sources in terms of their fitness is selected with the probability p_{onl}, otherwise the worse of the two food sources is selected. The reason for preferring this method over commonly used roulette wheel selection method is that fitness values of different solutions are closed to one another. Besides, binary tournament selection method is less expensive computationally.

4.3 Initial solution

The initial solutions for our ABC algorithm are generated in a purely random fashion. One location for facility is selected at a time randomly from the not yet selected vertices of V and this process is repeated until p facilities are located.

4.4 Pre-processing

Like the BCOi approach of [9], the input data is pre-processed before starting our ABC approach with the intention of making some computations faster. This pre-processing involves two phases and we have followed the notational conventions used in [9] to describe them. These two phases are described below:

- In the first phase, all pair shortest distance matrix D is computed using Floyd's algorithm. Then from this distance matrix D, two new matrices are computed. Every row of D together with corresponding indices are sorted in increasing order, thereby creating two new matrices D_{sort} and D_{circle}. The matrix D_{sort} contains each row of D in sorted order, whereas each row of the matrix $D_{circles}$ provides the information about the first, second, . . . ,n^{th} closest node to the node corresponding to the row. For the sake of illustration, consider $n = 4$ and

$$D = \begin{bmatrix} 0\,3\,4\,1 \\ 3\,0\,6\,8 \\ 4\,6\,0\,2 \\ 1\,8\,2\,0 \end{bmatrix}$$

The matrices D_{sort} and $D_{circles}$ corresponding to matrix D above are

$$D_{sort} = \begin{bmatrix} 0\,1\,3\,4 \\ 0\,3\,6\,8 \\ 0\,2\,4\,6 \\ 0\,1\,2\,8 \end{bmatrix}$$

$$D_{circles} = \begin{bmatrix} 1\,4\,2\,3 \\ 2\,1\,3\,4 \\ 3\,4\,1\,2 \\ 4\,1\,3\,2 \end{bmatrix}$$

- In the second phase, a matrix called D_{radius} is computed. All the entries along the main diagonal of this matrix are set to 1. Each element (i, j) with $i \neq j$ of this matrix provides a count of those nodes whose distance from node i is less than the distance of node j from node i. D_{radius} for the example provided above is

$$D_{radius} = \begin{bmatrix} 1\,2\,3\,1 \\ 1\,1\,2\,3 \\ 2\,3\,1\,1 \\ 1\,3\,2\,1 \end{bmatrix}$$

These matrices find their use in generating the neighboring solution efficiently as explained in the next subsection.

4.5 Neighboring solution generation

We have employed two methods to generate neighboring solutions to ensure such solutions are the proper mix of diversity and quality. These two methods are based on the concept of critical distance introduced in [9]. It is defined as the largest distance among all distances between nodes and their nearest center. The node which is at the farthest distance from its nearest center is termed as the critical node. In the first method, to generate a neighboring solution Z' for a solution Z, first we copy the solution Z into Z'. Then we delete Q centers from Z'. These Q centers are deleted one-by-one in an iterative manner, where during each iteration, the center whose deletion has a minimum adverse effect on objective function is deleted from Z' (ties are broken arbitrarily). Deleting Q centers makes the Z' infeasible, and, hence, to restore the feasibility of the Z', Q centers need to be added. These Q centers are added one-by-one in the following manner: First, current critical distance is determined and then the node i and center j corresponding to this critical distance is identified. A node is randomly chosen to be a center from among non-center nodes which can reduce the current critical distance and added to Z', i.e., a new center is chosen using the following equation

$$new\_center = D_{circles}(i, rand(D_{radius}(i, j)))$$

This process is repeated till Q centers have been added. The value of Q is determined as follows

$$Q = \begin{cases} round(\frac{p}{2}) & \text{if } 5 \times p < n \\ round(\frac{p}{5}) & \text{otherwise} \end{cases} \tag{2}$$

The second method is a modified version of the method used by [9]. In this method, to generate a neighboring solution Z' for a solution Z, first we copy the solution Z into Z'. We choose one solution Z_1 (different from Z) from the population randomly. Then we add Q more centers to Z' one-by-one. Each center is added in the following manner: Like the first method, node i and center j corresponding to current critical distance is identified. Obviously, the solution quality can be improved only by reducing the current critical distance. To reduce the current critical distance, we find the set of candidate centers K which falls within the radius of the critical distance. These candidate centers K are at a distance less than the current critical distance from the critical node i.

The set K is determined from the following equation

$$K = \{D_{circles}(i, k), \text{ where } k = 1, \ldots, D_{radius}(i, j)\}$$

We compute the intersection of K and Z_1. If this intersection is nonempty, then we randomly select one node from set $K \cap Z_1$ and add it to Z'. If this intersection is empty, then we randomly select one node from set K and add it to Z'. This process is repeated until we have added Q centers. Davidović et al. [9] has always selected a node from K randomly and has not made use of another solution. The use of another solution is based on the fact that if a facility is located at a particular demand point in one good solution, then it is highly likely that a facility is located at the same demand point in many good solutions. Hence, the use of another solution helps in identifying potential nodes. Also, note that K is never empty as it always contains critical node i which is responsible for current critical distance.

Adding Q centers will reduce the critical distance, but makes the solution infeasible, hence we need to delete Q centers to restore its feasibility. The Q centers are deleted one-by-one in the following manner. Among the $p + Q$ centers, we find the impact of each center on the solution quality and delete that center which has the least impact on the solution quality (ties are broken arbitrarily). This process is repeated until only p centers remain in the solution. The value of Q is determined using the same equation as in first method, i.e., equation 2.

On the other hand, the value of Q in [9] is randomly chosen in the interval $[0, p]$, if $5 \times p$ is less than n and in the interval $[0, \frac{n-2.5p}{2.5}]$ otherwise, each time a new solution needs to be produced. So the value of Q varies in the method of [9]. If the solution Z, the original solution, and, the solution Z_1, the randomly chosen solution, are identical, i.e., these two solutions contain same locations for all the facilities. In this case copying the location to Z' will produce another solution identical to Z and Z_1, hence resulting into one more copy of Z (and Z_1). This situation is known as collision in ABC algorithm jargon [15]. If this situation arises while generating a neighboring solution for an employed bee then the corresponding employed bee becomes a scout by abandoning that solution. The scout bee is reinitialized as employed bee by associating it with a new solution generated randomly in a fashion analogous to an initial solution. This is done to get rid of one duplicate solution. If collision occurs while generating neighboring solution for an onlooker bee then another solution is chosen randomly. This process of selecting a random solution is repeated until a solution different from original solution is found for an onlooker bee. Instead of tackling the collision like employed bee phase, the reason for following a different course of action is that it is futile to generate a solution randomly for an onlooker bee as such a solution seldom survives. An onlooker bee solution can survive only when it is better than the original solution with which the onlooker in consideration is associated and solutions of all other onlooker bees which are also associated with the same original solution. It is highly unlikely that a randomly generated solution is better than all these solutions.

These two methods are used in a mutually exclusive way. The first method is utilized for generating a neighboring solution with probability p_{sel}, otherwise the second method is applied.

4.6 Other features

An employed bee abandons its associated solution when this solution has not improved over a specified number of iterations called *limit*, and becomes a scout. For ABC algorithm, *limit* is an important control parameter and should be appropriately set as it is responsible for maintaining a balance between exploration and exploitation. A lesser value of *limit* favours exploration over exploitation. On the other hand, a higher value of *limit* gives preference to exploitation undermining exploration. An employed bee can also become a scout, as mentioned in Section 4.4 through a collision. A newly designated scout bee is immediately associated with a newly generated food source and it becomes employed again. This food source is generated randomly in a manner analogous to an initial solution. In our algorithm, no explicit upper and lower limits are imposed on the number of scouts in a single iteration. The number of scouts in a particular iteration is governed by the two conditions mentioned above.

Algorithm 1 provides the pseudo-code of our ABC approach where *Generate_Neighbor*(Z) is a function that take as input a solution Z and returns a solution in the neighborhood of Z (Section 4.4). *binary_tournament*$(e_1, e_2, \ldots, e_{n_e})$ is another function that selects a solution among employed bee solutions $e_1, e_2, \ldots, e_{n_e}$ with the help of binary tournament selection method (Section 4.2) and returns the index of the solution selected. In this algorithm, n_e and n_o denote respectively the number of employed and onlooker bees.

Algorithm 1: Pseudo-Code of our Hybrid ABC Algorithm

```
Randomly generate $n_e$ solutions $e_1, e_2, \ldots, e_{n_e}$;
best_sol:= best solution among $e_1, e_2, \ldots, e_{n_e}$;
while termination condition is not satisfied do
    for $i = 1$ to $n_e$ do
        $e'$ := Generate_Neighbor($e_i$);
        if $e'$ is better than $e_i$ then
            $e_i$:=$e'$
        if $e'$ is better than best_sol then
            best_sol:=$e'$
    for $i = 1$ to $n_o$ do
        $k_i$ := Binary_Tournament($e_1, e_2, \ldots, e_n$);
        $onl_i$ := Generate_Neighbor($e_{k_i}$);
        if $onl_i$ is better than best_sol then
            best_sol:=$onl_i$
    for $i = 1$ to $n_o$ do
        if $onl_i$ is better than $e_{k_i}$ then
            $e_{k_i}$:=$onl_i$
    for $i = 1$ to $n_e$ do
        if $e_i$ has not improved over limit iterations then
            replace $e_i$ with a random solution;
return best_sol ;
```

5 The invasive weed optimization algorithm

Invasive weed optimization (IWO) is a new meta-heuristic technique proposed by Mehrabian and Lucas [14] in 2006. It is inspired by the phenomenon of aggressive colonization of weeds in nature. It is proved in the literature that a powerful optimization algorithm can be obtained by mimicking the characteristics of weeds. Weeds have the tendency to grow by means of dispersal and occupy empty spaces between the useful plants. Weeds have great surviving ability because of their adaptive nature. Several different variants of the original IWO algorithm have been proposed in the literature, e.g. see [26], [27], [28], [29], [30], [31], [32], [33] etc.

In the IWO algorithm, each weed corresponds to a possible solution to the problem under consideration and the fitness of a weed is determined by the fitness of the solution it represents. IWO algorithm follows an iterative process which begins by generating a population of solutions randomly. Then an iterative process ensues where during each iteration, each weed produces certain number of seeds (new solutions) according to its fitness. The number of seeds a weed can produce are mapped linearly from s_{min} to s_{max} according to its fitness. The seeds are produced in such a manner so that they are distributed in the search space according to a normally distribution with mean equal to the location of the producing weed and varying standard deviations. Following expression is used to determine the standard deviation (S.D.) σ at each iteration.

$$\sigma_{iter} := \left(\frac{iter_{max} - iter}{iter_{max}}\right)^{nmi} (\sigma_i - \sigma_f) + \sigma_f$$

Here $iter_{max}$ is maximum number of iterations, σ_{iter} is standard deviation at present iteration, and nmi is nonlinear modulation index, σ_i is the pre-defined initial value, σ_f is the pre-defined final value. Please note that the value of the standard deviation is decreased from σ_i to σ_f over iterations of the algorithm so that search space is explored thoroughly during initial iterations and it is reduced gradually over the iterations in order to focus the search in the neighborhood of good solutions only during final iterations. Once all weeds have finished producing their designated number of seeds, the newly produced seeds compete with weeds already existing in the colony for survival through a process known as competitive exclusion. If the number of weeds in the colony is less than the maximum number of weeds allowed in the colony n_{max} then all the newly produced seeds are included in the colony of weeds as new weeds. However, once the number of weeds in the colony reaches n_{max}, only the fittest n_{max} weeds, among the existing ones and newly produced ones, are retained in the colony. After this another iteration of IWO algorithm begins. This process is repeated as long as termination condition is not satisfied.

6 The IWO approach to the p-center problem

Several components of our IWO approach have been borrowed from our ABC approach (Section 4). Our IWO approach employs same solution encoding and same fitness function as our ABC approach. Subsequent subsections describe various features of our IWO algorithm for the p-center problem.

6.1 Initial solution

Initial weed solutions are produced randomly in the same manner as described in Section 4.3. The number of initial solutions n_i is a parameter of the algorithm.

6.2 Determination of number of seeds, a weed can produce

Each weed solution, based on its own and the colony's lowest and highest fitness, produces certain number of seed solutions. The number of such seed solutions lies in the interval $[X_{min}, X_{max}]$ where X_{min} is the minimum possible number of seeds and X_{max} is the maximum possible number of seeds. However, unlike the traditional IWO algorithm where there is a linear mapping between a weed solution and the number of seed solutions it can produce, we have followed the strategy used in [24] where the total number of solutions present in the colony at a particular moment (N_c) are partitioned into $(X_{max} - X_{min}) + 1$ disjoint groups according to their fitness and the solutions belonging to the same group will produce the same number of seed solutions which lies in the interval $[X_{min}, X_{max}]$. The pseudo code for determining number of seeds is as follows.

Algorithm 2: Method for determining number of seeds

```
function No_of_Seeds(i)
Input: index $i$ of a solution in the sorted solution list
Output: number of seeds that the solution corresponding to index $i$ can
        produce
begin
    if $\left(i \leq \frac{N_c}{(X_{max}-X_{min})+1}\right)$ then
        return $X_{max}$;
    else if $\left(i \leq \frac{2*N_c}{(X_{max}-X_{min})+1}\right)$ then
        return $X_{max} - 1$;
    ⋮
    ⋮
    else if $\left(i \leq \frac{(X_{max}-X_{min})*N_c}{(X_{max}-X_{min})+1}\right)$ then
        return $X_{min} + 1$;
    else
        return $X_{min}$;
```

6.3 *Production of seeds by a weed*

We have utilized the neighboring solution generation method of ABC algorithm (Section 4.5) for producing a seed solution corresponding to a weed solution.

6.4 *Competitive exclusion*

After some number of iterations, number of solutions in the colony reaches the maximum allowable value n_{max} as new seed solutions are continuously produced and added to the colony. When such a situation arises, the competitive exclusion procedure begins discarding the solutions with poor fitness. The competitive exclusion procedure works as follows: Once all solutions in the colony have generated their designated number of seed solutions, all the solutions including newly generated seed solutions are sorted into non-increasing order as per their fitness and best n_{max} solutions are retained in the colony and remaining solutions are discarded. In this way, solutions having higher fitness only survive and reproduce.

The pseudo code of our IWO approach is given in Algorithm 3, where n_c and n_i denote the number of weeds in the colony at any instance of time and the number of weeds in the colony at the beginning respectively. Sorting of weeds into non-increasing order based on their fitness is done by the function Sort_The_Weeds(). The number of seeds a weed W_i can produce as per the method described in Section 6.2 is computed by the function No_of_Seeds(). Generate_Seed_Solution(W) is another function which returns a seed solution around the solution W (Section 6.3).

Algorithm 3: Pseudo-code of IWO Approach

```
Generate $n_i$ initial weed solutions $W_1, W_2, \ldots, W_{n_i}$ randomly;
$best :=$ best solution among $W_1, W_2, \ldots, W_{n_i}$;
$n_c := n_i$;
Sort_The_Weeds($n_c$);
if $n_c > n_{max}$ then
    $n_c := n_{max}$;
while Termination condition not satisfied do
    $n_{colony}$=$n_c$;
    for $i := 1$ to $n_{colony}$ do
        $seeds :=$ No_of_Seeds($i$);
        for $j := 1$ to $seeds$ do
            $n_c := n_c + 1$;
            $W_{n_c} :=$ Generate_Seed_Solution($W_i$);
            if $W_{n_c}$ is better than best then
                $best := W_{n_c}$;
    Sort_The_Weeds($n_c$);
    if ($n_c > n_{max}$) then
        $n_c := n_{max}$;
return $best$;
```

7 Comptational Results

Our ABC & IWO approaches have been implemented in C and executed on a Linux based Intel Core i5 2400 system with 4 GB memory running at 3.10 GHz. We have used the following parameter values for ABC algorithm in all our computational experiments : the number of employed bees (n_e) is 50, the number of onlooker bees (n_o) is 100, p_{onl} is set to 0.65, p_{sel} is set to 0.3, $limit$ is set to 50, Our ABC approach terminates after 100 iterations. The parameters of IWO are as follows: number of initial solutions (n_i) is 50, maximum number of solutions allowed in colony n_{max} is 200, p_{sel} is set to 0.35, X_{max} is set to 5, X_{min} is set to 1, maximum number of iterations is set to 50. All these parameter values have been chosen empirically based on large number of trials. These parameter values yield good results for most instances. However, they are in no way optimal parameter values for all instances.

We have tested our approaches on 40 test problems from the OR-Library[1], where these instances are listed as instances of uncapacitated p-median problem. The number of demand points in these instances vary from 100 to 900 and the number of centers from 5 to 200. We have compared the results obtained by ABC & IWO approaches on each instance with best values known so far, and, the best method known so far viz. BCOi [9]. Like [9], we have executed our approach once on each instance so as to allow fair comparison with BCOi and other approaches. In table 1, we have reported these results. The first column contains test problem number. The total number of nodes and the total number of centers are given in the next two columns. The best known value obtained so far is reported under the column heading Best. These best values are due to single run of at least one of the methods among M-I, VNS, TS-1 and TS-2 [6], SS approach [8] and BCOi [9]. Next three columns presents the values obtained by BCOi, ABC and IWO during their solitary run. Columns 7-10 report the running time to reach the best value. Column heading Best(in lit) report the running time from the literature to reach the best value. The last three columns report the time needed by BCOi, ABC and IWO to reach the best objective function value. Data for M-I, VNS, TS-1 and TS-2, SS and BCOi approaches are taken from [9] where results were obtained on a Linux based Core 2 Duo system with 8GB RAM running at 2.66GHz. As this system is different from the system used to execute ABC & IWO approaches, therefore execution times can not be compared precisely. However, a rough comparison can always be made.

Table 1 clearly show the effectiveness of our approaches in terms of solution quality. ABC algorithm obtained best known solution values in 35 out of 40 instances and IWO reached the best know solution values for 36 instances out of 40 in contrast to 33 for BCOi. Even, IWO was also able to improve the best known solution value for one instance (viz. pmed25), which is reported in bold face. From this table, it can also be seen that for those problem instances where our approaches fail to reach the best known values, they reached the second best value. However, our approaches are slower than other approaches in terms of time to reach the best solutions with IWO the slowest among the lot. In this table, we have compared the time to reach the best solution due to past precedences only. However, total execution time is a more standard and reliable measure of time taken by various approaches as best solution is returned only when the algorithm finishes execution and terminates no matter how fast the best solution is found. BCOi was executed for fixed amount of time on each instance that range from 1 second to 70 seconds depending on the instance. Table 2 report the execution times of ABC & IWO approaches along with BCOi for each instance. Here the execution times of ABC & IWO approaches are average of 10 independent runs. From this table, we can see that our approaches, particularly ABC are faster than BCOi on a number of instances.

To test the robustness of ABC & IWO approaches, we have executed each of them 10 independent times. The results are reported in Table 3. The first column contains the problem instance number, second column represents the best known value. Next 3 columns report the best value, average value and standard deviation value obtained by the ABC algorithm. Column 6 report the average time till best and column 7 report the average execution time of ABC algorithm. Similarly, we report the best value, average value , standard deviation, average time till best and average execution time for IWO algorithm in columns 8-13. From the table, it is clear that ABC algorithm is able to improve the best known values for 3 instances and IWO algorithm obtained improved best known values for 5 instances, which

Table 1 The results of executing various approaches once on each of the 40 test problems

			Objective function value				Time(in seconds)			
Sno	N	P	Best	BCOi	ABC	IWO	Best(in lit)	BCOi	ABC	IWO
1	100	5	127	127	127	127	0	0	0.01	0.01
2	100	10	98	98	98	98	0	0	0.01	0.21
3	100	10	93	93	93	93	0	0	0.03	0.11
4	100	20	74	74	74	74	0	0	0.04	0.2
5	100	33	48	48	48	48	0	0	0.02	0.22
6	200	5	84	84	84	84	0	0	0.01	0.47
7	200	10	64	64	64	64	0	0	0.04	0.12
8	200	20	55	55	55	55	0.01	0.01	0.28	0.67
9	200	40	37	37	37	37	0.02	0	0.14	1.05
10	200	67	20	20	20	20	0.08	0.03	0.22	1.6
11	200	5	59	59	59	59	0.01	0	0.08	0.03
12	300	10	51	51	51	51	0.01	0.02	0.07	0.79
13	300	30	36	37	36	36	0.01	0.01	0.55	4.13
14	300	60	26	27	26	26	0.32	0.14	0.38	5.8
15	300	100	18	18	18	18	0.05	0.04	0.43	3.55
16	400	5	47	47	47	47	0	0	0.01	0.07
17	400	10	39	39	39	39	0	0.01	0.12	0.68
18	400	40	28	29	29	29	0.16	0.15	1.19	4.8
19	400	80	19	19	19	19	1.01	0.07	0.99	7.46
20	400	133	14	14	14	14	0.44	0.09	1.29	10.85
21	500	5	40	40	40	40	0	0	0.02	0.05
22	500	10	38	39	39	39	0.3	0.19	0.15	0.63
23	500	50	23	23	23	23	0.07	0.23	2.96	2.58
24	500	100	16	16	16	16	0.23	0.08	1.26	11.54
25	500	167	12	12	12	**11**	1.46	0.38	1.81	78
26	600	5	38	38	38	38	0	0	0.1	0.24
27	600	10	32	32	32	32	0	0.01	0.2	0.85
28	600	60	19	19	19	19	0.21	0.05	0.68	4.04
29	600	120	13	14	14	14	1.26	1.14	1.68	17.19
30	600	200	10	10	10	10	0.71	0.35	3.08	500.11
31	700	5	30	30	30	30	0	0	0.03	0.11
32	700	10	29	29	29	29	0.23	0.06	1.4	5.85
33	700	70	16	16	16	16	0.8	0.72	17.01	63.93
34	700	140	12	12	12	12	0.67	0.39	2.93	34.39
35	800	5	30	30	30	30	0.02	0.01	0.15	0.56
36	800	10	27	28	28	27	0.42	0.08	0.08	3.85
37	800	80	16	16	16	16	0.83	0.12	1.53	11.53
38	900	5	29	29	29	29	0	0	0.03	0.25
39	900	10	23	24	24	24	0.04	0.01	0.28	0.87
40	900	90	14	14	14	14	0.4	0.19	1.5	26.88

are reported in bold face in this table. For ABC & IWO both, on around 66% of instances, average solution quality is same as best solution quality which indicates ABC & IWO approaches are able to obtain best known values in every run. Even for remaining instances, average solution quality does not differ much from best solution quality.

Table 2 Execution times (in seconds) of ABC, IWO and BCOi approaches

Sno	BCOi	ABC	IWO
1	1.00	0.14	0.34
2	2.00	0.51	1.24
3	2.00	0.52	1.35
4	2.00	0.64	1.51
5	1.00	1.29	3.18
6	2.00	0.29	3.18
7	2.00	1.02	2.56
8	2.00	3.09	7.96
9	2.00	3.87	10.00
10	2.00	9.04	23.28
11	2.00	0.40	1.07
12	2.00	1.59	3.99
13	2.00	9.31	23.52
14	3.00	11.78	30.29
15	6.00	28.48	73.00
16	1.00	0.54	1.36
17	2.00	2.15	5.46
18	70.00	20.61	52.82
19	4.00	26.12	66.56
20	4.00	62.33	160.02
21	2.00	0.74	1.84
22	2.00	2.74	6.91
23	4.00	38.75	97.78
24	5.00	49.64	123.31
25	4.00	122.25	296.14
26	2.00	1.10	2.05
27.	2.00	4.24	7.68
28	4.00	98.52	153.08
29	10.00	138.79	197.88
30	8.00	37.21	49.79
31	2.00	1.42	2.48
32	2.00	5.75	9.27
33	10.00	190.70	232.92
34	15.00	290.17	306.16
35	2.00	1.91	2.82
36	2.00	7.63	10.50
37	7.00	333.82	339.31
38	2.00	1.96	3.17
39	20.00	8.04	11.75
40	20.00	461.01	472.65

8 Conclusions

In this paper, we have proposed two metaheuristic approaches, viz. artificial bee colony algorithm and invasive weed optimization algorithm for the p-

Table 3 Results of ABC & IWO approaches over 10 runs

		ABC					IWO				
Sno	Best_Known	Best	Avrg	Std_Dev	ABT	ATT	Best	Avrg	Std_Dev	ABT	ATT
1	127	127	127	0.00	0.005	0.138	127	127	0.00	0.012	0.339
2	98	98	98	0.00	0.083	0.510	98	98	0.00	0.096	1.242
3	93	93	93.7	0.46	0.142	0.524	93	93.5	0.49	0.110	1.351
4	74	74	74	0.00	0.072	0.635	74	74.2	0.60	0.168	1.505
5	48	48	48	0.00	0.029	1.298	48	48	0.00	0.196	3.179
6	84	84	84.2	0.40	0.097	0.285	84	84	0.00	0.385	0.713
7	64	64	64	0.00	0.088	1.017	64	64	0.00	0.193	2.558
8	55	55	55	0.00	0.320	3.097	55	55	0.00	0.728	7.961
9	37	37	37	0.00	0.169	3.869	37	37	0.00	0.989	10.000
10	20	20	20	0.00	0.235	9.043	20	20	0.00	1.685	23.279
11	59	59	59.2	0.40	0.069	0.400	59	59.1	0.30	0.119	1.079
12	51	51	51.2	0.40	0.280	1.585	51	51	0.00	0.636	3.994
13	36	36	36.7	0.46	1.671	9.312	36	36.1	0.30	3.881	23.520
14	26	26	26.2	0.40	2.970	11.784	26	26	0.00	4.908	30.285
15	18	18	18	0.00	0.414	28.480	18	18	0.00	3.811	72.996
16	47	47	47	0.00	0.011	0.538	47	47	0.00	0.073	1.360
17	39	39	39	0.00	0.299	2.151	39	39	0.00	0.521	5.458
18	28	29	29	0.00	2.323	20.613	29	29	0.00	4.450	52.824
19	19	19	19	0.00	1.687	26.122	19	19	0.00	7.006	66.560
20	14	14	14	0.00	1.048	62.329	14	14	0.00	10.591	160.022
21	40	40	40	0.00	0.046	0.735	40	40	0.00	0.146	1.843
22	38	39	39	0.00	0.147	2.742	39	39	0.00	0.476	6.911
23	23	23	23	0.00	2.478	38.749	23	23	0.00	6.838	97.781
24	16	16	16	0.00	1.558	49.637	**15**	15.9	0.30	15.154	123.315
25	12	**11**	11.5	0.50	24.822	122.248	**11**	11.4	0.49	38.269	296.135
26	38	38	38.1	0.30	0.232	1.096	38	38	0.00	0.208	2.054
27.	32	32	32	0.00	0.383	4.237	32	32	0.00	0.627	7.677
28	19	19	19	0.00	1.032	98.517	**18**	18.9	0.30	17.259	153.081
29	13	13	13.7	0.46	24.829	138.792	13	13.5	0.50	34.875	197.882
30	10	10	10	0.00	5.172	37.205	10	10	0.00	494.505	49.789
31	30	30	30	0.00	0.064	1.421	30	30	0.00	0.175	2.481
32	29	29	29.7	0.46	0.831	5.754	29	29.3	0.46	2.503	9.268
33	16	16	16.1	0.30	73.94	190.696	16	16	0.00	25.828	232.917
34	12	**11**	11.6	0.49	69.203	290.169	**11**	11.6	0.49	53.221	306.157
35	30	30	30.1	0.30	0.336	1.905	30	30.1	0.30	0.381	2.824
36	27	27	27.8	0.40	1.160	7.626	27	27.8	0.40	1.450	10.493
37	16	16	16	0.00	4.329	333.820	16	16	0.00	13.445	339.307
38	29	29	29	0.00	0.056	1.956	29	29	0.00	0.209	3.168
39	23	24	24	0.00	0.343	8.042	24	24	0.00	0.656	11.747
40	14	14	14	0.00	5.134	461.011	14	14	0.00	15.796	472.646

center problem. We have evaluated the performance of our proposed approaches against the state-of-the-art approach, viz. BCOi available in the literature on the standard benchmark instances for this problem. Computational results show the effectiveness of our approaches in finding high quality solutions.

Performance of our approaches can be improved further with the help of a suitable local search strategy and we intend to work in this direction. Approaches similar to ours can be developed for other related facility location problems also. Another possible future work is to consider a bi-objective version where one objective is same as p-center problem and second objective is same as p-median problem, i.e., second objective is to minimize the sumtotal of distances of demand points from their nearest center and study the tradeoff between the two objectives.

References

[1] O.Kariv and S.L.Hakimi. An algorithmic approach to network location problems. part ii. the p-median. *SIAM Journal of Applied Mathematics*, 37:539–560, 1969.

[2] D.Chen and R.Chen. New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. *Computers & Operations Research*, 36(5):1646–1655., 2009.

[3] Z.Drezner. The planar two center and two median problems. *Transportation Science*, 18:451–461., 1984.

[4] M.S.Daskin. Network and discrete location: models, algorithms, and application. *John Wiley & Sons*, 1995.

[5] G.Y. Handler. p-center problems. In *Discrete location theory*, pages 305–315. John Wiley & Sons, 1990.

[6] N.Mladenović, M.Labbé, and P.Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42(1):48–64, 2003.

[7] C.Caruso, A.Colorni, and L.Aloi. Dominant, an algorithm for the p-center problem. *European Journal of Operational Research*, 149(1):53–64, 2003.

[8] J.A. Pacheco and S.Casado. Solving two location models with few facilities by using a hybrid heuristic: a real health resources case. *Computers & Operations Research*, 32(12):3075–3091, 2005.

[9] T.Davidović, D. Ramljak, M. Śelmić, and D.Teodorović. Bee colony optimization for the p-center problem. *Computers & Operations Research*, 38(10):1367–1376, 2011.

[10] R.Chandrasekaran and A.Tamir. Polynomial bounded algorithms for locating p-centers on a tree. *Mathematical Programming*, 22:304 –315, 1982.

[11] B.Pelgrin. Heuristic methods for the p-center problem. *RAIRO Recherche Operationelle*, 25:65–72., 1991.

[12] R.Hassin, A.Levin, and D.Morad. Lexicographic local search and the p-center problem. *European Journal of Operational Research*, 151(2):265–279, 2003.

[13] D. Karaboga. An idea based on honey bee swarm for numerical optimization. technical report tr06, 2005. Computer Engineering Department, Erciyes University, Turkey.

[14] A.R.Mehrabian and C. Lucas. A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics*, 1(4):355–366, 2006.

[15] A. Singh. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing*, 9:625–631, 2009.

[16] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numeric function optimization: Artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39:459–471, 2007.

[17] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing*, 8:687–697, 2008.

[18] D. Karaboga and B. Akay. A modified artificial bee colony (abc) algorithm for constrained optimization problems. *Applied Soft Computing*, 11:3021–3031, 2011.

[19] A.Singh and S.Sundar. An artificial bee colony algorithm for the minimum routing cost spanning tree problem. *Soft Computing*, 15(12):2489–2499, 2011.

[20] Q.-K.Pan, M.F.Tasgetiren, P.N. Suganthan, and T.J. Chua. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences*, 181(12):2455–2468, 2011.

[21] T. K. Sharma and M. Pant. Enhancing the food locations in an artificial bee colony algorithm. *Soft Computing*, 17(10):1939–1965, 2013.

[22] B.Jayalakshmi and A.Singh. A hybrid artificial bee colony algorithm for the terminal assignment problem. In *Swarm, Evolutionary, and Memetic Computing*, volume 8947, pages 134–144. Springer International Publishing, 2015.

[23] S.N.Chaurasia and A.Singh. A hybrid swarm intelligence approach to the registration area planning problem. *Information Sciences*, 302:50–69, 2015.

[24] P.Venkatesh and A.Singh. Two metaheuristic approaches for the multiple traveling salesperson problem. *Applied Soft Computing*, 26:74–89, 2015.

[25] D. Karaboga and B. Basturk. Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In *IFSA 2007, Lecture notes in Artificial Intelligence*, volume 4529, pages 789–798. Springer, 2007.

[26] Z.Xuncai, W.Yanfeng, C.Guangzhao, N.Ying, and X.Jin. Application of a novel {IWO} to the design of encoding sequences for {DNA} computing. *Computers & Mathematics with Applications*, 57(11âŁ"12):2001–2008, 2009.

[27] G.G.Roy, P.Chakroborty, Z.Shi-Zheng, S.Das, and P.N.Suganthan. Artificial foraging weeds for global numerical optimization over continuous spaces. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8, 2010.

[28] S.Roy, Sk.M.Islam, S.Das, and S.Ghosh. Multimodal optimization by artificial weed colonies enhanced with localized group search optimizers. *Applied Soft Computing*, 13(1):27–46, 2013.

[29] A.Basak, D.Maity, and S.Das. A differential invasive weed optimization algorithm for improved global numerical optimization. *Applied Mathematics and Computation*, 219(12):6645–6668, 2013.

[30] D.Kundu, K.Suresh, S.Ghosh, S.Das, B.K. Panigrahi, and S.Das. Multi-objective optimization with artificial weed colonies. *Information Sciences*, 181(12):2441–2454, 2011.

[31] A.Dastranj, H.Abiri, and A.Mallahzadeh. Design of a broadband cosecant squared pattern reflector antenna using IWO algorithm. *IEEE Transactions on Antennas and Propagation*, 61(7):3895–3900, 2013.

[32] D.I. Abu-Al-Nadi, O.M.K. Alsmadi, Z.S. Abo-Hammour, M.F. Hawa, and J.S. Rahhal. Invasive weed optimization for model order reduction of linear {MIMO} systems. *Applied Mathematical Modelling*, 37(6):4570–4577, 2013.

[33] P.Ramezani, M.Ahangaran, and X.-S.Yang. Constrained optimisation and robust function optimisation with EIWO. *International Journal of Bio-Inspired Computation*, 5(2):84–98, 2013.