OPSEARCH (2017) 54:67–93
DOI 10.1007/s12597-016-0271-8

APPLICATION ARTICLE

A hybrid artificial bee colony algorithm for the *p*-median problem with positive/negative weights

B. Jayalakshmi[1] · **Alok Singh**[1]

Accepted: 16 June 2016 / Published online: 5 July 2016

Abstract In the classical p-median problem, the objective is to find a set Y of p vertices on an undirected graph $G = (V, E)$ in such a way that $Y \subseteq V$ and the sum of distances from all the vertices to their respective closest vertices in Y is minimized. In this paper, we have considered the weighted case where every vertex in G has either a positive or a negative weight under two different objective functions, viz. the sum of the minimum weighted distances and the sum of the weighted minimum distances. In this paper, we have proposed a hybrid artificial bee colony (ABC) algorithm for the positive/negative weighted p-median problem where each solution generated by ABC algorithm is improved by an interchange based randomized local search. In addition, an interchange based exhaustive local search is applied on some of the best solutions obtained after the execution of ABC algorithm in a bid to further improve their quality. We have compared our approach with the state-of-the-art approaches available in the literature on the standard benchmark instances. Computational results demonstrate the effectiveness of our approach.

Keywords Artificial bee colony algorithm · Swarm intelligence · p-Median problem · p-Median problem with positive/negative weights

1 Introduction

Given an undirected weighted graph $G = (V, E)$ with $|V| = n$, the classical p-median problem seeks on this graph a set $Y \subseteq V$ of p vertices in such a way that the

✉ Alok Singh
alokcs@uohyd.ernet.in

B. Jayalakshmi
bjayalakshmi@uohyd.ac.in

[1] School of Computer and Information Sciences, University of Hyderabad, Hyderabad 500 046, India

sum of distances from all the vertices to their respective closest vertices in Y is minimized. The vertices of the graph can be considered as demand points and the vertices in Y as the location of facilities, the goal is to select the locations of p facilities to serve n demand points, so that the sum of the distances of demand points from their nearest facilities is minimized. We have used the vertices in set Y and facility locations interchangeably throughout this paper. The p-median problem and its variations can be used to model many real world situations, e.g., in locating public facilities, industrial plants and ware-houses. These are only a few examples from the long list of situations where this model can be applied. This model can also be used in applications related to cluster analysis, where points in an m-dimensional space can be regarded as user locations. p-median problem can also be posed in the form of a matrix as follows: Given a matrix D of dimension $n \times n$, select p columns of D in a way such that the sum of minimum coefficients in each row within these p columns is as small as possible.

The classical p-median problem is shown to be $\mathcal{NP}$-hard by Kariv and Hakimi [26]. Because of this, applicability of exact methods is limited to small instances only, and we need heuristics to solve large instances. The first heuristic based on the greedy strategy for the classical p-median problem is proposed by Kuehn and Hamburger [27]. Another early heuristic is the interchange heuristic proposed by Teitz and Bart [40]. Since then numerous heuristic and metaheuristic approaches have been proposed, e.g., fast interchange heuristic [42], global/regional interchange algorithm [13], LEVEL-2 and LEVEL-3 heuristics [12], gamma heuristic [36], tabu search [35], variable neighborhood search [19], simulated annealing [11], genetic algorithms [1, 4, 14, 20], hybrid heuristic methods [33], a swap-based local search procedure [34], a parallel genetic algorithm [31], particle swarm optimization based approaches [29, 37] and an artificial bee colony algorithm [3]. A survey on metaheuristic approaches for the p-median problem can be found in [28].

A location problem with positive and negative weights on the vertices is useful in applications, where some facilities are non-attractive to some clients (facilities are obnoxious). Many obnoxious location problems are discussed and classified in Cappanera [8]. Interested readers can find the survey on obnoxious location problems in Carrizosa and Plastria [9] and Plastria [32]. Burkard and Krarup [7] proposed the first location model with positive and negative weights and also proved that the 1-median problem on a cactus with positive and negative vertex weights can be solved in linear time. Burkard et al. [5] observed that there exist two different models when p-median problem with positive/negative weights in graphs is considered. In the first model, referred to as P1, the sum of the minimum weighted distances is minimized. In the second model, referred to as P2, the sum of the weighted minimum distances is minimized.

Burkard et al. [5] developed an $\mathcal{O}(n^2)$ algorithm for the 2-median problem on a tree. They also developed an $\mathcal{O}(n \log n)$ algorithm for stars and an $\mathcal{O}(n)$ algorithm for paths for first model (P1). They presented an $\mathcal{O}(n^3)$ algorithm for the 2-median problem on a tree for the second model (P2) and showed that the complexity can be reduced to $\mathcal{O}(n^2)$ if the medians are restricted to vertices. Burkard and Fathali [6] presented an algorithm for 3-median problem on a tree for second model (P2). There

exists some heuristic methods also to solve the positive/negative weighted p-median problem on graphs. Fathali and Kakhki [17] developed a modified variable neighborhood search (MVNS). Fathali et al. [16] presented an ant colony optimization algorithm (ACO). A genetic algorithm (GA) for the positive/negative weighted p-median problem is proposed by Fathali [15]. ACO and GA are two best approaches known so far for the positive/negative weighted p-median problem.

In this paper, we have proposed an artificial bee colony (ABC) algorithm based approach for the positive/negative weighted p-median problem. ABC algorithm is a new population based metaheuristic technique based on intelligent foraging behaviour of honey bee swarm, which has been applied successfully to solve numerous combinatorial optimization problems in diverse domains. Though there exists an ABC algorithm for the classical p-median problem [3], there exists no ABC algorithm for the positive/negative weighted p-median problem. Besides, the approach of [3] contains some design flaws (see Sect. 4.7). This has motivated us to develop the approach presented in this paper which is altogether different from the ABC approach presented in [3] for classical p-median problem. We have used two local search procedures in our ABC approach. In a bid to improve each solution generated by ABC algorithm, it is passed through an interchange based randomized local search. In addition, an interchange based exhaustive local search is applied on some of the best solutions obtained after the execution of ABC algorithm in an attempt to further improve them. We have compared our ABC approach with ACO and GA [15, 16] on the standard benchmark instances for the problem. Computational results show the effectiveness of our approach.

The remainder of this paper is structured as follows: Sect. 2 defines the positive/negative weighted p-median problem formally. In Sect. 3, we provide an overview of ABC algorithm. Section 4 presents our ABC approach to solve the positive/negative weighted p-median problem. Section 5 reports the computational results and compares our approach with other approaches available in the literature. Finally, Sect. 6 outlines some concluding remarks.

2 Formal problem definition

The classical p-median problem can be formally stated 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 way that the sum of distances from all vertices to their closest vertices in Y is minimized, i.e., the solution is a subset $Y = \{y_1, y_2, \ldots, y_p\}$ of V that minimizes

$$\sum_{i=1}^{n} \min_{j \in \{1,\ldots,p\}} d(v_i, y_j) \tag{1}$$

In the weighted version of the problem, a weight w_i is associated with each vertex and the objective is to minimize the sum of weighted minimum distances, i.e.,

$$\sum_{i=1}^{n} w_i \min_{j\in\{1,\ldots,p\}} d(v_i, y_j) \tag{2}$$

Burkard et al. [5] noted that two different models exist for p-median problem when weights on the vertices can be both positive and negative. In the first model (P1), the objective is to minimize the sum of the minimum weighted distances.

$$O1(Y) = \sum_{i=1}^{n} \min_{j\in\{1,\ldots,p\}} \left(w_i d\left(v_i, y_j\right)\right) \tag{3}$$

In the second model (P2), the objective is to minimize the sum of weighted minimum distances.

$$O2(Y) = \sum_{i=1}^{n} w_i \min_{j\in\{1,\ldots,p\}} d\left(v_i, y_j\right) \tag{4}$$

Please note that both models are equivalent when we have positive weights only.

3 Overview of ABC algorithm

The artificial bee colony (ABC) algorithm proposed by Karaboga [21] is a population based meta-heuristic algorithm, which is inspired by the intelligent behavior of the foraging honey bees. In a bee colony, there are three types of bees: employed, onlooker and scout. Employed bees are those bees which are currently exploiting a food source. The responsibility of the employee bees is to bring loads of nectar to the hive and share the information about their food sources with other bees waiting in the hive. The waiting bees are known as onlookers. The onlookers then choose a food source with a probability directly proportional to its quality and becomes employed. Scout bees search for new food sources in the vicinity of the hive and they become employed as soon as they find a new food source. An employed bee whose food source becomes empty will abandon that food source and becomes either a scout or an onlooker.

Inspired by the foraging bees' behavior described above, Karaboga developed ABC algorithm. This algorithm was originally developed for solving optimization problems in continuous domain only, later, it has been extended to solve discrete optimization problems also. Since the development of first ABC algorithm [21], numerous variants of basic algorithm have been proposed, e.g. [2, 10, 23–25, 30, 38, 39, 41]. For a recent survey on ABC algorithm and its applications, one may refer to Karaboga et al. [22].

In ABC algorithm also there are three types of bees, viz. employed, onlooker and scout with functions similar to their real counterparts. In ABC algorithm, the food sources represent the possible solutions to the problem under consideration and the quality of a food source represents the fitness of the represented solution. The employed bees are associated with food sources. Always, there is a one-to-one correspondence between food sources and employed bees, which means, the number

of food sources is equal to the number of employee bees. Usually, but not always, the number of onlooker bees is also taken to be equal to number of employed bees. The ABC algorithm follows an iterative search process, which starts with associating the employee bees with randomly generated food sources (solutions), then it repeats through the cycles of the employed bee and onlooker bee phases.

In the employed bee phase, each employed bee generates a food source in the proximity of its associated food source and evaluates its quality. The method of determining a new food source in the proximity of a particular food source depends on the problem under consideration. If the quality of the new food source is better than the current one then the employed bee moves to the new food source leaving the old one. Otherwise, it remains at the old food source. When all the employed bees finish this process, then employed bee phase ends and onlooker bee phase begins.

Onlooker bee phase starts with sharing of information by employed bees about their food sources with the onlookers. Onlookers select the food sources according to their quality, i.e., higher the value of the fitness of the solution represented by a food source, higher will be the chances of its selection. As a result of such a selection, good quality food sources will get more chance for selection by the onlookers. After all onlookers select the food sources, they determine the food sources in the proximity of their selected food sources in a manner similar to the employed bees and evaluate their fitness. Among all the neighboring food sources generated by the onlookers who chose food source i and the food source i itself, the best quality food source is determined. This best food source will be updated as food source i for the next iteration. The onlooker bee phase ends once all food sources are updated, and then the next iteration of the ABC algorithm begins. The algorithm is repeated until the termination condition is satisfied.

If a solution associated with any employed bee does not improve over some specific number of iterations, then that food source is considered as exhausted and it is discarded by its associated employee bee and that employee bee becomes scout. Such scouts are converted back into employed bees by associating them with newly generated solutions. Usually, these new solutions are generated randomly in the same manner as initial employed bee solutions or by perturbing an existing solution.

Clearly, every solution is given a fair chance to improve itself in the employed bee phase. On the other hand, in the onlooker bee phase, because of the selection policy used by the onlookers as mentioned above good quality solutions get more chance to improve themselves in comparison to poor quality solutions. This inclination towards selecting good quality solutions may produce better quality solutions faster, as there are higher chances of finding even better solutions within the proximity of good solutions in comparison to poor ones. However, if a solution is locally optimal, then no better solution exists in its proximity and any attempt to improve it will be futile. The concept of scout bees helps in 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 concept of scout bees helps in getting rid of

solutions which has not improved since long and which can be locally optimal. For a search process to be robust, the balance between the exploration and exploitation must be maintained. In the ABC algorithm, employed bees and onlooker bees carry out the exploitation, whereas scout bees are responsible for exploration. The number of iterations without improvement in the ABC algorithm after which an employed bee leaves a solution and becomes a scout needs to be set appropriately so as to maintain a proper balance between exploration and exploitation.

4 ABC approach for the *p*-median problem with positive/negative weights

In this section, we present our ABC approach for the *p*-median problem with positive/negative weights. Subsequent subsections describe the salient features of our ABC approach.

4.1 Solution representation and fitness

We have represented a solution directly by the subset of vertices used for locating the facilities and used the objective function as the fitness function. So for model P1, fitness is determined using Eq. 3, whereas for model P2, fitness is determined using Eq. 4. Please note the less value of the fitness function means a more fit solution. The two models for the proposed *p*-median problem will differ in the assignment of demand points to the facilities. In model P1, vertices with positive weights are assigned to the nearest facility and vertices with negative weights are assigned to the farthest facility. On the other hand, in model P2 vertices with both positive and negative weights are assigned to the closest facility.

4.2 Food source selection for onlooker bees

For selecting a food source for an onlooker bee, we have employed the binary tournament selection method. In the binary tournament selection method, two food sources are selected randomly and their fitness is compared. The better of the two food sources as per their fitness is selected with the probability p_{onl}. Otherwise, the worse of the two food sources is selected, i.e., the probability of selection of worse solution is $1 - p_{onl}$. The Pseudo-code for the binary tournament selection method is as follows:

Algorithm 1: Pseudo code for binary tournament selection method

```
select two solutions e1 and e2 form the employed bee solutions randomly;
generate a random number r between 0 and 1;
if r ≤ p_onl then
    return the best solution between e1 and e2;
else
    return the worst solution between e1 and e2;
```

4.3 Initial solution

The initial solution for our ABC algorithm is generated using the random method. One location for facility is selected at a time, randomly from the vertex set V and this process is repeated until p facilities are located.

4.4 Neighboring solution generation

Our neighboring solution generation process is inspired by the genetic operators used in [15]. To generate a solution Y' in the neighborhood of solution Y, we choose another solution Y_1 randomly from the population and copy those locations of facilities which are common in both solutions Y and Y_1 into Y'. Then a fraction F_r (with $F_r > 0.5$) of the remaining locations for facilities are added from solution Y, and the rest are added from solution Y_1. Here, F_r is a parameter to be determined empirically. To add a new location to the solution Y', we always add the location which yields the smallest objective function value (assuming only that many facilities need to be opened). If the two solutions Y and Y_1 are identical, i.e., all the facility locations in two solutions are same then there is no point in copying all the location to Y' as doing so will produce another solution identical to Y and Y_1. This situation is known as collision in ABC algorithm jargon [38]. If a collision occurs while generating a neighboring solution for an employee bee then original solution is abandoned and the concerned employee bee becomes a scout. Then a new solution is generated randomly in a fashion similar to an initial solution for this scout bee and its status is again changed back to employed by associating it with this new 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. If again collision occurs then again a solution is chosen randomly. This process continues till a solution different from original solution is found. The reason behind handling the collision for an onlooker bee in a manner different from an employed bee lies in the fact that it is worthless to generate a solution randomly for an onlooker bee. This is so because an onlooker bee solution can survive only when it is better than the original solution and solutions of all other onlooker bees which are associated with this original solution. Obviously, it is highly improbable that a randomly generated solution is better than all these solutions.

In a bid to further improve the neighboring solution obtained through the aforementioned method, we have used 1-interchange heuristic. In this heuristic method, we replace one vertex in the solution Y' by a vertex which is not present in it and which results in maximum reduction in objective function value. We randomly select one vertex in Y' and find a vertex in $\{V - Y'\}$, which results in maximum reduction in objective function value. This method is computationally expensive, because of the large number of fitness calculations performed each time it is applied. However, it aids more often than not in improving the quality of a solution. To balance the computational cost and degree of improvement, we have applied 1-interchange heuristic K times on every solution, where K is a parameter to be determined empirically.

4.5 Other features

If there is no improvement in the quality of a food source over a specified number of iterations say *limit*, then the employed bee associated with that food source leaves it and becomes a scout. This scout is associated with a newly generated food source so that it can become employed again. This food source is generated randomly in the same manner as an initial solution. After generating a food source, this scout again becomes employed. The *limit* is an important control parameter of ABC algorithm. An employed bee can also become a scout, as mentioned in Sect. 4.4 through collision. So the number of scouts in a particular iteration depends on these two conditions and there is no lower and upper limits on the number of scouts in an iteration.

Algorithm 2: Pseudo-Code of our Hybrid ABC Algorithm

```
randomly generate n_e employed bee solutions e_1, e_2, ..., e_{n_e};
for i := 1 to L do
    best_sol_i := i^{th} best solutions among e_1, e_2, ..., e_{n_e};
while termination condition is not satisfied do
    for i := 1 to n_e do
        e' := generate_neighbor(e_i);
        for j = 1 to K do
            e' := 1-interchange(e')
        if e' is better than e_i then
            e_i := e';

        for j = 1 to L do
            if e' is better than best_sol_j then
                best_sol_j := e';
                break;
    for i := 1 to n_o do
        k_i := binary_tournament(e_1, e_2, ..., e_{n_e});
        onl_i := generate_neighbor(e_{k_i});
        for j = 1 to K do
            onl_i := 1-interchange(onl_i)
        for j := 1 to L do
            if onl_i is better than best_sol_j then
                best_sol_j := onl_i;
                break;
    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 last limit iterations then
            replace e_i with a random solution;
for i := 1 to L do
    best_sol_i := local_search(best_sol_i);
best := best solution among best_sol_1, best_sol_2, ..., best_sol_L;
return best;
```

4.6 Local search

Once the ABC algorithm finishes execution, a local search is applied on the L best solutions found by the ABC algorithm in a bid to further improve their solution quality. In this local search, each vertex y in a solution Y is tried to be exchanged one-by-one with a vertex not in Y so that the value of the objective function is reduced by the largest amount. This process is repeated till the objective function can not be improved further. The 1-interchange heuristic, where only K ($<p$) randomly chosen vertices instead of all are tried for exchange, can be considered as a lighter variant of this local search.

Algorithm 2 provides the pseudo-code of our hybrid ABC approach where *generate_neighbor*(Y), 1-*interchange*(Y) and *local_search*(Y) are three functions that take as input a solution Y and return respectively a solution in the neighborhood of Y (first paragraph in Sect. 4.4), a solution obtained after applying 1-interchange heuristic on Y (second paragraph in Sect. 4.4), a solution obtained after applying local search on solution Y (Sect. 4.6). *binary_tournament*($e_1, e_2, \ldots, e_{n_e}$) is another function that selects a solution among employed bee solutions $e_1, e_2, \ldots, e_{n_e}$ using binary tournament selection method (Sect. 4.2) and returns the index of the solution selected.

4.7 Key points of difference with a related work

Basti and Sevkli [3] have proposed an artificial bee colony algorithm for the classical p-median problem. This subsection highlights the key differences between their approach and our proposed approach:

- Basti and Sevkli [3] have used a real vector of length n to encode a solution in their ABC algorithm. To decode a solution from this real vector, indices corresponding to smallest p values in this vector are found and demand points corresponding to these indices are assumed to be the location of facilities. On the other hand, in our ABC algorithm, we have represented a solution directly by the subset of vertices used for locating the facilities, hence the length of a solution is equal to p ($p<n$). The encoding scheme of [3] suffers from problem of redundancy, i.e., the same solution can be encoded in many different ways. In fact, in the encoding scheme of [3], each solution can be represented in infinitely many ways. As ABC algorithm works in the space of encoded solutions, in the presence of redundancy, it has to search a larger space which can severely impair its performance. On the other hand, encoding scheme used by us does not suffer from the problem of redundancy as each solution is represented uniquely. The size of search space in [3] is infinite, whereas in our case it is $\binom{n}{p}$. Besides, the length of an encoded solution has an adverse impact on the efficiency of several operators associated with ABC algorithm. With respect to this aspect also, our encoding is better as $p<n$.
- Real vector encoding scheme of [3] incurs a decoding overhead to get the actual solution from its encoded version. In our case, no decoding overhead is incurred

as each solution is represented directly by the subset of vertices used for locating the facilities

- As Basti and Sevkli [3] used a real vector to encode a solution, they followed the original neighboring solution generation method proposed by Karaboga [21], i.e., a neighboring solution is generated by changing the value of one randomly chosen parameter of the original solution. On the other hand, our neighboring solution generation method is based on the assumption that if a vertex is present in one good solution then there are chances that the same vertex may appear in many good solutions. Hence, we have given maximum attention to those vertices which are common in original solution and randomly selected solution, followed by those vertices which are in one of these solutions.
- Basti and Sevkli used roulette wheel selection method for selecting a solution for an onlooker bee. We have used binary tournament selection method for selecting a solution for an onlooker bee. Its an established fact that binary tournament selection method performs better than roulette wheel selection method and at the same time its computationally less expensive. In fact, it has roughly the same performance as rank selection method [18].
- In their work, a greedy local search algorithm is applied on every solution generated by the ABC algorithm. While applying this local search on a solution, facility locations are considered one-by-one. The facility location in consideration is tried for replacement with all other non facility locations. The location that yields the least objective function value is retained and then the next facility location is considered. Our 1-interchange heuristic is similar. However, instead of trying all facilities one-by-one, only $K < p$ facilities are tried for replacement to cut the computational cost. Facilities to be tried for replacement are selected randomly. In addition, in our work, another local search is applied on L best solutions obtained after the execution of ABC algorithm. This local search is also similar to the local search of Basti and Sevkli except for the fact that we keep applying this local search repeatedly as long as there is improvement in solution quality, i.e., our local search stops when a complete pass through the existing facility locations fails to improve the solution quality.

5 Computational results

Our hybrid ABC approach has been implemented in C and executed on a Linux based Intel Core i5 2400 system with 4 GB RAM running at 3.10 GHz. In all our computational experiments, the number of employed bees (n_e) is taken to be 25, the number of onlooker bees (n_o) is taken to be 50, p_{onl} is set to 0.85, *limit* is set to 50, F_r is set to $\frac{2}{3}$, K is set to 2 and L is set to 5. Our hybrid ABC approach terminates after 100 iterations. All these parameter values were chosen empirically after a large number of trials.

We have compared our hybrid ABC approach with two best approaches, viz. GA [15] and ACO [16] approaches. For this comparison, we have used the same 40 test

instances as used in Fathali [15] and Fathali et al. [16]. These instances are slightly modified version of the uncapacitated p-median problem instances available for download from OR-Library[1]. Slight modification is done in these instances to accommodate negative weights. Vertices weights in these instances is restricted to ± 1 only. Further, the negative weight of -1 is assigned to only the first 2 or first 5 or first 10 vertices and to all odd numbered vertices. The case where first 5 vertices have negative weights was considered in Fathali [15] only, whereas the case where first 2 vertices have negative weights was considered in Fathali et al. [16] only. All other cases are considered by both the papers. Hence, the results for GA and ACO are not available for instances with 2 negative weights and 5 negative weights respectively. Like GA and ACO approaches, we have executed our hybrid ABC approach 5 times on each instance and reported the average results.

Tables 1, 3, 4, 5, 6, 7, 8, 9 and 10 present the results of ABC algorithm on various types of instances and compare them with those of genetic algorithm (GA) and ant colony algorithm (ACO) methods. Table 1 reports the results of various approaches on instances with positive weights. This table also reports the average total CPU times in seconds of GA, ACO and ABC approaches on each instance with positive weights. However, for models P1 and P2, we have reported the total CPU time that is averaged over all the instances that are derived from the same instance of the uncapacitated p-median problem. This is done to ensure conformity with Fathali [15] and Fathali et al. [16]. Table 2 reports these times and next paragraph further explains how these times have been computed. Tables 3, 4, 5 and 6 report the results of various approaches for model P1 on instances with 2 negative weights, 5 negative weights, 10 negative weights and half negative weights respectively, whereas Tables 7, 8, 9 and 10 does the same for model P2. As mentioned already, performances of GA and ACO were not evaluated on instances with 2 negative weights and 5 negative weights respectively, and hence, Tables 3 and 7 report the results of ABC and ACO only, whereas Tables 4 and 8 report the results of ABC and GA only. Results for GA and ACO are taken from their respective papers. The columns under the common heading *Objective function value* report the objective function value averaged over 5 runs for various approaches, whereas columns under the common heading *% Error* report the relative error of various approaches on each instance. The relative error is defined as follows:

$$\frac{f - f_{O/B}}{|f_{O/B}|} \times 100$$

where f is the objective function value obtained by the algorithm and $f_{O/B}$ is the optimal or the best known value so far obtained. Optimal values are known only for the instances with positive weights. For other types of instances, proven optimal values are not known. Moreover, for some instances, our ABC approach has found a value better than the best known value. In such cases, we have replaced the best known value with new best known value found by our ABC algorithm. Such cases are reported in bold font in these tables.

[1] http://www.brunel.ac.uk/~mastjjb/jeb/info.html

Table 1 The results for the test problems with positive weights

S. no.	N	P	Optimal	Objective function value			% Error			Time (s)		
				GA	ACO	ABC	GA	ACO	ABC	GA	ACO	ABC
1	100	5	5819	5819	5819	5819	0.00	0.00	0.00	0.350	1.000	0.856
2	100	10	4093	4093	4093	4093	0.00	0.00	0.00	0.650	0.495	0.856
3	100	10	4250	4250	4250	4250	0.00	0.00	0.00	0.700	0.792	0.898
4	100	20	3034	3034	3034	3034	0.00	0.00	0.00	0.850	1.008	0.892
5	100	33	1355	1355	1355	1355	0.00	0.00	0.00	0.950	0.891	0.932
6	200	5	7824	7824	7824	7824	0.00	0.00	0.00	0.800	2.000	3.938
7	200	10	5631	5631	5631	5631	0.00	0.00	0.00	1.500	2.000	3.482
8	200	20	4445	4445	4445	4445	0.00	0.00	0.00	3.330	2.140	3.494
9	200	40	2734	2739	2734	2734	0.18	0.00	0.00	5.000	2.574	3.632
10	200	67	1255	1260	1255	1255	0.40	0.00	0.00	6.330	2.772	3.644
11	300	5	7696	7696	7696	7696	0.00	0.00	0.00	6.330	8.000	7.916
12	300	10	6634	6634	6634	6634	0.00	0.00	0.00	9.630	8.400	7.666
13	300	30	4374	4374	4374	4374	0.00	0.00	0.00	17.330	8.400	7.944
14	300	60	2968	2969	2968	2969	0.03	0.00	0.03	30.330	11.655	8.972
15	300	100	1729	1736	1730	1731	0.40	0.06	0.12	36.000	8.168	9.914
16	400	5	8162	8162	8162	8162	0.00	0.00	0.00	19.660	13.090	15.160
17	400	10	6999	6999	6999	6999	0.00	0.00	0.00	32.330	19.802	13.776
18	400	40	4809	4811	4809	4812	0.04	0.00	0.06	58.330	24.210	14.072
19	400	80	2845	2849	2846	2850	0.14	0.04	0.18	104.000	27.850	19.360
20	400	133	1789	1789	1789	1793	0.00	0.00	0.22	148.330	17.028	24.302
21	500	5	9138	9138	9138	9138	0.00	0.00	0.00	41.330	37.900	23.916
22	500	10	8579	8579	8579	8579	0.00	0.00	0.00	70.330	55.815	21.974
23	500	50	4619	4619	4619	4619	0.00	0.00	0.00	141.000	33.662	23.120

Table 1 continued

S. no.	N	P	Optimal	Objective function value			% Error			Time (s)		
				GA	ACO	ABC	GA	ACO	ABC	GA	ACO	ABC
24	500	100	2961	2961	2961	2967	0.00	0.00	0.20	253.330	42.849	28.954
25	500	167	1828	1831	1828	1830	0.16	0.00	0.11	391.660	56.534	34.462
26	600	5	9917	9917	9917	9917	0.00	0.00	0.00	89.330	57.424	35.710
27	600	10	8307	8307	8307	8307	0.00	0.00	0.00	138.660	70.296	31.594
28	600	60	4498	4498	4502	4499	0.00	0.09	0.02	311.660	93.245	34.902
29	600	120	3033	3034	3034	3035	0.03	0.03	0.07	568.000	90.214	43.062
30	600	200	1989	1989	1992	1996	0.00	0.15	0.35	957.660	87.505	60.132
31	700	5	10,086	10,086	10,086	10,086	0.00	0.00	0.00	152.000	80.596	48.752
32	700	10	9297	9297	9297	9297	0.00	0.00	0.00	188.000	87.004	43.336
33	700	70	4700	4703	4700	4702	0.06	0.00	0.04	543.660	89.670	47.054
34	700	140	3013	3013	3014	3014	0.00	0.03	0.03	1120.000	138.480	65.372
35	800	5	10,400	10,400	10,400	10,400	0.00	0.00	0.00	216.660	122.760	74.066
36	800	10	9934	9934	9934	9934	0.00	0.00	0.00	302.330	103.960	68.440
37	800	80	5057	5058	5057	5059	0.02	0.00	0.04	1111.330	107.028	77.530
38	900	5	11,060	11,060	11,060	11,060	0.00	0.00	0.00	316.330	132.667	99.110
39	900	10	9423	9423	9423	9423	0.00	0.00	0.00	487.660	133.660	89.850
40	900	90	5128	5129	5134	5130	0.02	0.12	0.04	2120.000	220.958	105.140

Table 2 The average total CPU times (in seconds) for problems P1 and P2

S. no.	N	P	P1				P2			
			GA	ABC	ACO	ABC	GA	ABC	ACO	ABC
1	100	5	1.000	0.896	1.280	0.892	1.000	0.831	1.000	0.827
2	100	10	1.550	0.871	1.520	0.872	1.500	0.807	1.400	0.809
3	100	10	1.333	0.873	1.400	0.873	1.500	0.817	1.200	0.815
4	100	20	1.777	0.847	1.160	0.859	1.777	0.821	0.880	0.818
5	100	33	1.777	0.894	0.960	0.910	1.555	0.858	0.640	0.856
6	200	5	2.666	3.697	3.000	3.745	2.444	3.736	2.600	3.734
7	200	10	3.000	3.180	3.200	3.233	2.888	3.363	2.600	3.365
8	200	20	4.777	3.194	2.840	3.243	4.111	3.278	2.640	3.279
9	200	40	7.222	3.377	2.760	3.444	5.777	3.455	2.440	3.455
10	200	67	8.222	3.792	2.360	3.703	6.777	3.742	1.720	3.859
11	300	5	9.777	8.085	10.480	7.975	7.777	8.124	10.192	8.064
12	300	10	11.888	7.804	10.640	7.782	11.666	7.876	9.238	7.802
13	300	30	21.777	7.809	10.730	7.827	18.444	7.605	11.572	7.535
14	300	60	38.625	8.979	11.530	8.845	36.125	9.070	10.411	8.503
15	300	100	35.777	9.537	11.819	9.607	32.444	9.069	9.525	9.027
16	400	5	25.333	15.307	27.787	15.421	24.777	15.447	25.238	15.330
17	400	10	41.500	13.844	26.512	13.823	37.000	13.857	21.400	13.701
18	400	40	63.888	14.278	26.231	14.153	58.111	13.548	20.000	13.295
19	400	80	105.777	17.209	26.852	16.649	86.333	15.789	28.079	15.979
20	400	133	151.555	20.514	32.183	19.595	117.888	18.000	25.911	18.048
21	500	5	49.777	25.129	46.960	25.157	43.375	23.656	35.400	23.577
22	500	10	80.777	22.983	48.957	22.827	75.666	22.769	45.755	22.777
23	500	50	143.333	24.257	50.299	23.981	127.111	22.172	39.292	22.039

Table 2 continued

S. no.	N	P	P1				P2			
			GA	ABC	ACO	ABC	GA	ABC	ACO	ABC
24	500	100	242.000	28.925	53.628	29.082	218.222	25.758	43.935	25.735
25	500	167	363.222	35.381	63.856	35.769	279.444	30.781	60.158	31.098
26	600	5	102.000	37.861	74.440	38.055	102.222	39.522	69.460	39.227
27	600	10	165.666	34.483	72.760	34.330	141.666	36.340	61.400	36.351
28	600	60	336.777	37.419	87.052	36.618	284.888	37.198	68.514	36.889
29	600	120	540.444	49.204	100.343	48.915	531.888	47.151	77.294	47.486
30	600	200	817.111	55.105	112.216	56.410	734.000	57.405	83.340	57.224
31	700	5	162.777	55.871	116.082	55.856	141.429	69.401	87.480	68.134
32	700	10	246.333	51.196	128.334	51.373	201.111	63.564	92.130	63.109
33	700	70	607.222	59.753	152.155	57.710	558.555	70.251	158.425	71.898
34	700	140	1083.555	73.677	226.733	73.923	1038.444	91.304	170.329	92.356
35	800	5	241.888	72.291	134.640	72.823	217.222	95.668	109.880	96.042
36	800	10	395.444	68.875	138.566	68.522	379.777	95.995	115.993	95.907
37	800	80	1125.666	81.188	190.364	79.345	985.111	115.559	208.808	117.084
38	900	5	354.222	96.293	203.823	97.181	309.333	126.913	144.920	126.650
39	900	10	474.777	88.159	174.992	87.775	428.333	121.855	176.755	122.011
40	900	90	2051.555	118.983	244.079	116.361	1612.888	149.598	200.122	149.907

Table 3 The results of various approaches on instances with 2 negative weights under model P1

Instance	N	P	Objective function value			% Error	
			Best	ACO	ABC	ACO	ABC
1	100	5	5300	5300	5300	0.00	0.00
2	100	10	3724	3724	3724	0.00	0.00
3	100	10	3541	3541	3541	0.00	0.00
4	100	20	2450	2450	2450	0.00	0.00
5	100	33	878	878	878	0.00	0.00
6	200	5	7485	7485	7485	0.00	0.00
7	200	10	**5311**	5327	5311	0.30	**0.00**
8	200	20	4050	4050	4050	0.00	0.00
9	200	40	**2458**	2471	2458	0.53	**0.00**
10	200	67	986	986	986	0.00	0.00
11	300	5	7522	7522	7522	0.00	0.00
12	300	10	6494	6494	6494	0.00	0.00
13	300	30	4122	4122	4122	0.00	0.00
14	300	60	2679	2679	2679	0.00	0.00
15	300	100	1534	1534	1534	0.00	0.00
16	400	5	7994	7994	7994	0.00	0.00
17	400	10	6899	6899	6899	0.00	0.00
18	400	40	4569	4569	4569	0.00	0.00
19	400	80	2691	2691	2695	0.00	0.15
20	400	133	1615	1615	1616	0.00	0.06
21	500	5	9044	9044	9044	0.00	0.00
22	500	10	8444	8444	8444	0.00	0.00
23	500	50	4475	4475	4475	0.00	0.00
24	500	100	2805	2805	2810	0.00	0.18
25	500	167	1633	1633	1635	0.00	0.12
26	600	5	9803	9803	9803	0.00	0.00
27	600	10	8190	8190	8190	0.00	0.00
28	600	60	**4339**	4343	4339	0.09	**0.00**
29	600	120	2913	2913	2913	0.00	0.00
30	600	200	1840	1840	1842	0.00	0.11
31	700	5	10,015	10,015	10,015	0.00	0.00
32	700	10	9211	9211	9211	0.00	0.00
33	700	70	4575	4575	4576	0.00	0.02
34	700	140	2830	2830	2830	0.00	0.00
35	800	5	10,319	10,319	10,319	0.00	0.00
36	800	10	9862	9862	9862	0.00	0.00
37	800	80	4921	4921	4921	0.00	0.00
38	900	5	10,993	10,993	10,993	0.00	0.00
39	900	10	9347	9347	9347	0.00	0.00
40	900	90	**5029**	5031	5029	0.04	**0.00**

Table 4 The results of various approaches on instances with 5 negative weights under model P1

Instance	N	P	Objective function value			% Error	
			Best	GA	ABC	GA	ABC
1	100	5	4681	4730	4730	1.05	1.05
2	100	10	2915	2915	2916	0.00	0.03
3	100	10	2529	2529	2532	0.00	0.12
4	100	20	1432	1432	1464	0.00	2.23
5	100	33	61	61	61	0.00	0.00
6	200	5	7061	7061	7064	0.00	0.04
7	200	10	4812	4846	4858	0.71	0.96
8	200	20	3399	3399	3433	0.00	1.00
9	200	40	**1918**	1919	1918	0.05	**0.00**
10	200	67	514	514	514	0.00	0.00
11	300	5	7290	7290	7290	0.00	0.00
12	300	10	6201	6201	6201	0.00	0.00
13	300	30	3798	3798	3808	0.00	0.26
14	300	60	**2263**	2269	2263	0.27	**0.00**
15	300	100	**1151**	1153	1151	0.17	**0.00**
16	400	5	7787	7787	7787	0.00	0.00
17	400	10	6723	6735	6727	0.18	0.06
18	400	40	4219	4219	4221	0.00	0.05
19	400	80	2420	2420	2421	0.00	0.04
20	400	133	1346	1346	1349	0.00	0.22
21	500	5	8888	8888	8897	0.00	0.10
22	500	10	8230	8230	8230	0.00	0.00
23	500	50	4256	4256	4288	0.00	0.75
24	500	100	2538	2538	2542	0.00	0.16
25	500	167	1394	1394	1400	0.00	0.43
26	600	5	9655	9655	9655	0.00	0.00
27	600	10	8038	8038	8038	0.00	0.00
28	600	60	4043	4043	4055	0.00	0.30
29	600	120	2698	2698	2727	0.00	1.07
30	600	200	**1603**	1604	1603	0.06	**0.00**
31	700	5	9909	9909	9909	0.00	0.00
32	700	10	8935	8955	8955	0.22	0.22
33	700	70	4411	4411	4424	0.00	0.29
34	700	140	2605	2605	2628	0.00	0.88
35	800	5	10,230	10,230	10,230	0.00	0.00
36	800	10	9735	9735	9742	0.00	0.07
37	800	80	4723	4723	4742	0.00	0.40
38	900	5	10,860	10,860	10,860	0.00	0.00
39	900	10	9070	9070	9070	0.00	0.00
40	900	90	4862	4862	4881	0.00	0.39

Table 5 The results of various approaches on instances with 10 negative weights under model P1

Instance	N	P	Best	Objective function value			% Error		
				GA	ACO	ABC	GA	ACO	ABC
1	100	5	3611	3611	3611	3611	0.00	0.00	0.00
2	100	10	1247	1247	1247	1247	0.00	0.00	0.00
3	100	10	1029	1029	1029	1029	0.00	0.00	0.00
4	100	20	−52	−52	−52	−52	0.00	0.00	0.00
5	100	33	−1143	−1143	−1143	−1143	0.00	0.00	0.00
6	200	5	6374	6374	6374	6374	0.00	0.00	0.00
7	200	10	4095	4095	4095	4095	0.00	0.00	0.00
8	200	20	2575	2575	2575	2575	0.00	0.00	0.00
9	200	40	**1085**	1088	1091	1085	0.28	0.55	**0.00**
10	200	67	−204	−204	−204	−204	0.00	0.00	0.00
11	300	5	6756	6756	6756	6756	0.00	0.00	0.00
12	300	10	5610	5610	5610	5610	0.00	0.00	0.00
13	300	30	3193	3193	3193	3193	0.00	0.00	0.00
14	300	60	1480	1480	1489	1480	0.00	0.61	0.00
15	300	100	632	635	632	632	0.47	0.00	0.00
16	400	5	7426	7426	7426	7426	0.00	0.00	0.00
17	400	10	6292	6292	6292	6292	0.00	0.00	0.00
18	400	40	3693	3693	3693	3693	0.00	0.00	0.00
19	400	80	2012	2013	2012	2013	0.05	0.00	0.05
20	400	133	910	910	910	911	0.00	0.00	0.11
21	500	5	8630	8630	8630	8630	0.00	0.00	0.00
22	500	10	7765	7765	7845	7765	0.00	1.03	0.00
23	500	50	3795	3795	3795	3795	0.00	0.00	0.00
24	500	100	2151	2151	2151	2153	0.00	0.00	0.09
25	500	167	990	990	990	994	0.00	0.00	0.40
26	600	5	9400	9400	9400	9400	0.00	0.00	0.00
27	600	10	7651	7651	7651	7651	0.00	0.00	0.00
28	600	60	3576	3576	3578	3577	0.00	0.06	0.03
29	600	120	2358	2359	2358	2360	0.04	0.00	0.08
30	600	200	1196	1199	1196	1196	0.25	0.00	0.00
31	700	5	9519	9688	9688	9688	1.78	1.78	1.78
32	700	10	8362	8418	8418	8418	0.67	0.67	0.67
33	700	70	**4142**	4144	4147	4142	0.05	0.12	**0.00**
34	700	140	2219	2219	2219	2220	0.00	0.00	0.05
35	800	5	10,039	10,039	10,039	10,039	0.00	0.00	0.00
36	800	10	9415	9415	9415	9415	0.00	0.00	0.00
37	800	80	4384	4384	4384	4385	0.00	0.00	0.02
38	900	5	10,696	10,696	10,696	10,696	0.00	0.00	0.00
39	900	10	8535	8535	8535	8535	0.00	0.00	0.00
40	900	90	4595	4595	4599	4596	0.00	0.09	0.02

Table 6 The results of various approaches on instances with half negative weights under model P1

Instance	N	P	Best	Objective function value			% Error		
				GA	ACO	ABC	GA	ACO	ABC
1	100	5	−7651	−7651	−7651	−7651	0.00	0.00	0.00
2	100	10	−9445	−9445	−9445	−9445	0.00	0.00	0.00
3	100	10	−12,398	−12,398	−12,398	−12,398	0.00	0.00	0.00
4	100	20	−11,507	−11,507	−11,507	−11,507	0.00	0.00	0.00
5	100	33	**−10,930**	−10,811	−10,811	−10,930	1.09	1.09	**0.00**
6	200	5	−9971	−9971	−9971	−9971	0.00	0.00	0.00
7	200	10	−10,403	−10,403	−10,403	−10,403	0.00	0.00	0.00
8	200	20	−13,912	−13,912	−13,901	−13,912	0.00	0.08	0.00
9	200	40	−13,997	−13,997	−13,997	−13,997	0.00	0.00	0.00
10	200	67	−12,437	−12,437	−12,437	−12,437	0.00	0.00	0.00
11	300	5	−10,271	−10,271	−10,271	−10,271	0.00	0.00	0.00
12	300	10	−14,850	−14,850	−14,850	−14,850	0.00	0.00	0.00
13	300	30	−13,557	−13,557	−13,557	−13,557	0.00	0.00	0.00
14	300	60	−17,676	−17,676	−17,676	−17,676	0.00	0.00	0.00
15	300	100	−14,437	−14,437	−14,437	−14,437	0.00	0.00	0.00
16	400	5	−10,792	−10,792	−10,792	−10,792	0.00	0.00	0.00
17	400	10	−11,583	−11,583	−11,583	−11,583	0.00	0.00	0.00
18	400	40	−16,286	−16,286	−16,286	−16,286	0.00	0.00	0.00
19	400	80	−14,200	−14,200	−14,200	−14,199	0.00	0.00	0.01
20	400	133	−16,362	−16,362	−16,361	−16,362	0.00	0.01	0.00
21	500	5	−11,296	−11,296	−11,296	−11,296	0.00	0.00	0.00
22	500	10	−16,588	−16,588	−16,588	−16,588	0.00	0.00	0.00
23	500	50	−15,272	−15,272	−15,272	−15,271	0.00	0.00	0.01
24	500	100	**−17,427**	−17,221	−17,221	−17,427	1.18	1.18	**0.00**
25	500	167	−17,924	−17,924	−17,922	−17,923	0.00	0.01	0.01
26	600	5	−13,060	−13,060	−13,060	−13,060	0.00	0.00	0.00
27	600	10	−16,204	−16,204	−16,179	−16,204	0.00	0.15	0.00
28	600	60	−22,970	−22,970	−22,970	−22,970	0.00	0.00	0.00
29	600	120	−17,796	−17,796	−17,796	−17,796	0.00	0.00	0.00
30	600	200	−21,333	−21,333	−21,333	−21,332	0.00	0.00	0.00
31	700	5	−11,466	−11,396	−11,396	−11,396	0.61	0.61	0.61
32	700	10	−30,465	−30,465	−30,456	−30,465	0.00	0.03	0.00
33	700	70	−16,917	−16,914	−16,917	−16,917	0.02	0.00	0.00
34	700	140	**−24,017**	−23,803	−23,805	−24,017	0.89	0.88	**0.00**
35	800	5	−14,709	−14,709	−14,709	−14,709	0.00	0.00	0.00
36	800	10	−21,934	−21,934	−21,934	−21,934	0.00	0.00	0.00
37	800	80	−21,038	−21,038	−21,036	−21,036	0.00	0.01	0.01
38	900	5	−21,059	−21,059	−21,059	−21,059	0.00	0.00	0.00
39	900	10	−38,980	−38,980	−38,980	−38,980	0.00	0.00	0.00
40	900	90	−19,350	−19,350	−19,350	−19,347	0.00	0.00	0.02

Table 7 The results of various approaches on instances with 2 negative weights under model P2

Instance	N	P	Objective function value			% Error	
			Best	ACO	ABC	ACO	ABC
1	100	5	5499	5499	5499	0.00	0.00
2	100	10	**4009**	4029	4009	0.50	**0.00**
3	100	10	3920	3920	3920	0.00	0.00
4	100	20	2845	2845	2845	0.00	0.00
5	100	33	1292	1292	1292	0.00	0.00
6	200	5	7590	7590	7590	0.00	0.00
7	200	10	**5457**	5471	5457	0.26	**0.00**
8	200	20	4281	4281	4281	0.00	0.00
9	200	40	**2702**	2713	2702	0.41	**0.00**
10	200	67	1213	1213	1214	0.00	0.08
11	300	5	7574	7574	7574	0.00	0.00
12	300	10	6584	6584	6584	0.00	0.00
13	300	30	4259	4259	4259	0.00	0.00
14	300	60	**2888**	2897	2888	0.31	**0.00**
15	300	100	1706	1706	1706	0.00	0.00
16	400	5	8034	8034	8034	0.00	0.00
17	400	10	**6943**	6945	6943	0.03	**0.00**
18	400	40	4713	4713	4713	0.00	0.00
19	400	80	2815	2815	2817	0.00	0.07
20	400	133	1747	1747	1749	0.00	0.11
21	500	5	9100	9100	9100	0.00	0.00
22	500	10	8487	8487	8487	0.00	0.00
23	500	50	4577	4577	4577	0.00	0.00
24	500	100	2923	2923	2927	0.00	0.14
25	500	167	1777	1777	1779	0.00	0.11
26	600	5	9827	9827	9827	0.00	0.00
27	600	10	8217	8217	8217	0.00	0.00
28	600	60	**4453**	4457	4453	0.09	**0.00**
29	600	120	3016	3016	3019	0.00	0.10
30	600	200	1973	1973	1976	0.00	0.15
31	700	5	10,038	10,038	10,038	0.00	0.00
32	700	10	9251	9251	9251	0.00	0.00
33	700	70	4654	4654	4656	0.00	0.04
34	700	140	2956	2956	2956	0.00	0.00
35	800	5	10,336	10,336	10,336	0.00	0.00
36	800	10	**9897**	9931	9897	0.34	**0.00**
37	800	80	5015	5015	5017	0.00	0.04
38	900	5	11,014	11,014	11,014	0.00	0.00
39	900	10	9350	9377	9377	0.29	0.29
40	900	90	**5111**	5114	5111	0.06	**0.00**

Table 8 The results of various approaches on instances with 5 negative weights under model P2

Instance	N	P	Objective function value			% Error	
			Best	GA	ABC	GA	ABC
1	100	5	5324	5324	5324	0.00	0.00
2	100	10	3696	3696	3696	0.00	0.00
3	100	10	3592	3592	3592	0.00	0.00
4	100	20	2460	2460	2460	0.00	0.00
5	100	33	994	994	994	0.00	0.00
6	200	5	7266	7266	7266	0.00	0.00
7	200	10	5224	5224	5224	0.00	0.00
8	200	20	4034	4034	4034	0.00	0.00
9	200	40	**2539**	2540	2539	0.04	**0.00**
10	200	67	**1074**	1077	1074	0.28	**0.00**
11	300	5	7440	7440	7440	0.00	0.00
12	300	10	6511	6511	6511	0.00	0.00
13	300	30	4187	4187	4187	0.00	0.00
14	300	60	**2773**	2785	2773	0.43	**0.00**
15	300	100	**1576**	1581	1576	0.32	**0.00**
16	400	5	7870	7870	7870	0.00	0.00
17	400	10	6849	6849	6849	0.00	0.00
18	400	40	4589	4589	4589	0.00	0.00
19	400	80	**2737**	2741	2737	0.15	**0.00**
20	400	133	1703	1703	1706	0.00	0.18
21	500	5	8980	8980	8980	0.00	0.00
22	500	10	8403	8403	8403	0.00	0.00
23	500	50	4520	4520	4520	0.00	0.00
24	500	100	2853	2853	2853	0.00	0.00
25	500	167	1735	1735	1738	0.00	0.17
26	600	5	9719	9719	9719	0.00	0.00
27	600	10	8137	8137	8137	0.00	0.00
28	600	60	**4384**	4385	4384	0.02	**0.00**
29	600	120	2965	2965	2965	0.00	0.00
30	600	200	1939	1939	1941	0.00	0.10
31	700	5	9915	9968	9968	0.53	0.53
32	700	10	9179	9179	9179	0.00	0.00
33	700	70	**4619**	4624	4619	0.11	**0.00**
34	700	140	**2910**	2913	2910	0.10	**0.00**
35	800	5	10,286	10,286	10,286	0.00	0.00
36	800	10	9803	9803	9803	0.00	0.00
37	800	80	4967	4967	4970	0.00	0.06
38	900	5	10,914	10,914	10,914	0.00	0.00
39	900	10	9305	9305	9305	0.00	0.00
40	900	90	5075	5075	5078	0.00	0.06

Table 9 The results of various approaches on instances with 10 negative weights under model P2

Instance	N	P	Best	Objective function value			% Error		
				GA	ACO	ABC	GA	ACO	ABC
1	100	5	4826	4826	4826	4826	0.00	0.00	0.00
2	100	10	2976	2976	2976	2976	0.00	0.00	0.00
3	100	10	3341	3341	3341	3341	0.00	0.00	0.00
4	100	20	1854	1854	1854	1854	0.00	0.00	0.00
5	100	33	533	533	533	533	0.00	0.00	0.00
6	200	5	6787	6787	6787	6787	0.00	0.00	0.00
7	200	10	4949	4949	4949	4949	0.00	0.00	0.00
8	200	20	3812	3812	3812	3812	0.00	0.00	0.00
9	200	40	**2389**	2391	2392	2389	0.08	0.13	**0.00**
10	200	67	903	904	903	903	0.11	0.00	0.00
11	300	5	7136	7136	7136	7136	0.00	0.00	0.00
12	300	10	6395	6395	6395	6395	0.00	0.00	0.00
13	300	30	4011	4011	4011	4011	0.00	0.00	0.00
14	300	60	2564	2564	2564	2564	0.00	0.00	0.00
15	300	100	1457	1462	1457	1461	0.34	0.00	0.27
16	400	5	7624	7624	7624	7624	0.00	0.00	0.00
17	400	10	6668	6668	6668	6668	0.00	0.00	0.00
18	400	40	4437	4437	4437	4437	0.00	0.00	0.00
19	400	80	2629	2633	2629	2632	0.15	0.00	0.11
20	400	133	1621	1623	1621	1623	0.12	0.00	0.12
21	500	5	8800	8800	8800	8800	0.00	0.00	0.00
22	500	10	8291	8291	8291	8291	0.00	0.00	0.00
23	500	50	4337	4337	4337	4337	0.00	0.00	0.00
24	500	100	2748	2779	2779	2784	1.13	1.13	1.31
25	500	167	1636	1636	1636	1638	0.00	0.00	0.12
26	600	5	9528	9528	9528	9528	0.00	0.00	0.00
27	600	10	8004	8004	8004	8004	0.00	0.00	0.00
28	600	60	4296	4296	4296	4296	0.00	0.00	0.00
29	600	120	2896	2897	2896	2898	0.03	0.00	0.07
30	600	200	1850	1850	1851	1855	0.00	0.05	0.27
31	700	5	9820	9820	9820	9820	0.00	0.00	0.00
32	700	10	9053	9053	9053	9053	0.00	0.00	0.00
33	700	70	4566	4572	4566	4568	0.13	0.00	0.04
34	700	140	2833	2835	2833	2835	0.07	0.00	0.07
35	800	5	10,142	10,142	10,142	10,142	0.00	0.00	0.00
36	800	10	9637	9637	9637	9637	0.00	0.00	0.00
37	800	80	4875	4875	4878	4877	0.00	0.06	0.04
38	900	5	10,800	10,800	10,839	10,800	0.00	0.36	0.00
39	900	10	9201	9201	9201	9201	0.00	0.00	0.00
40	900	90	5026	5026	5028	5026	0.00	0.04	0.00

Table 10 The results of various approaches on instances with half negative weights under model P2

Instance	N	P	Best	Objective function value			% Error		
				GA	ACO	ABC	GA	ACO	ABC
1	100	5	−635	−635	−635	−635	0.00	0.00	0.00
2	100	10	−1245	−1245	−1245	−1245	0.00	0.00	0.00
3	100	10	−1131	−1131	−1131	−1131	0.00	0.00	0.00
4	100	20	−1477	−1477	−1477	−1477	0.00	0.00	0.00
5	100	33	−1687	−1687	−1687	−1687	0.00	0.00	0.00
6	200	5	−1163	−1163	−1163	−1163	0.00	0.00	0.00
7	200	10	−1360	−1360	−1360	−1360	0.00	0.00	0.00
8	200	20	−1765	−1765	−1764	−1765	0.00	0.06	0.00
9	200	40	−2212	−2212	−2212	−2212	0.00	0.00	0.00
10	200	67	−1815	−1815	−1815	−1815	0.00	0.00	0.00
11	300	5	−797	−797	−797	−797	0.00	0.00	0.00
12	300	10	−1290	−1290	−1290	−1290	0.00	0.00	0.00
13	300	30	−1709	−1709	−1697	−1708	0.00	0.70	0.06
14	300	60	−2224	−2224	−2219	−2223	0.00	0.22	0.04
15	300	100	**−2154**	−2152	−2151	−2154	0.09	0.14	**0.00**
16	400	5	−932	−932	−932	−932	0.00	0.00	0.00
17	400	10	−1318	−1254	−1254	−1233	4.86	4.86	6.45
18	400	40	−2096	−2096	−2096	−2089	0.00	0.00	0.33
19	400	80	−2119	−2119	−2119	−2119	0.00	0.00	0.00
20	400	133	**−2295**	−2291	−2290	−2295	0.17	0.22	**0.00**
21	500	5	−687	−687	−622	−687	0.00	9.46	0.00
22	500	10	−1111	−1111	−1098	−1098	0.00	1.17	1.17
23	500	50	−1933	−1933	−1915	−1933	0.00	0.93	0.00
24	500	100	−2221	−2216	−2221	−2221	0.23	0.00	0.00
25	500	167	**−2379**	−2376	−2368	−2379	0.13	0.46	**0.00**
26	600	5	−820	−820	−820	−820	0.00	0.00	0.00
27	600	10	−1053	−1053	−1053	−1053	0.00	0.00	0.00
28	600	60	−2119	−2119	−2107	−2117	0.00	0.57	0.09
29	600	120	−2198	−2198	−2197	−2197	0.00	0.05	0.05
30	600	200	**−2321**	−2320	−2320	−2321	0.04	0.04	**0.00**
31	700	5	−748	−748	−748	−748	0.00	0.00	0.00
32	700	10	−1030	−1030	−975	−1030	0.00	5.34	0.00
33	700	70	−2009	−2009	−2009	−2005	0.00	0.00	0.20
34	700	140	**−2437**	−2436	−2427	−2437	0.04	0.41	**0.00**
35	800	5	−855	−855	−855	−855	0.00	0.00	0.00
36	800	10	−985	−985	−985	−985	0.00	0.00	0.00
37	800	80	−2278	−2278	−2263	−2276	0.00	0.66	0.09
38	900	5	−607	−607	−522	−607	0.00	14.0	0.00
39	900	10	−901	−901	−901	−901	0.00	0.00	0.00
40	900	90	−2347	−2347	−2343	−2347	0.00	0.17	0.00

Table 11 Summary table

Model	Weights	GA		ACO		ABC		BKV-I
		W	TE	W	TE	W	TE	
–	Positive	11	1.48	7	0.53	14	1.51	–
P1	2-neg	–	–	4	0.96	6	0.64	4
P1	5-neg	8	2.71	–	–	24	11.12	4
P1	10-neg	8	3.59	8	4.91	11	3.30	2
P1	Half-neg	5	3.79	10	4.05	6	0.67	3
P2	2-neg	–	–	9	2.29	10	1.13	8
P2	5-neg	9	1.98	–	–	6	1.10	8
P2	10-neg	9	2.16	6	1.77	10	2.42	1
P2	Half-neg	7	5.56	18	39.46	9	8.48	5

As mentioned in the previous paragraph, Table 2 reports the average total CPU times in seconds of GA, ACO and ABC approaches for each of the two models (P1 and P2). The first three columns represents the problem number of the original uncapacitated *p*-median problem, number of nodes and number of centres. columns 4–7 report the average time for model P1 and columns 8–11 report the average time for the model P2. As explained earlier, performances of GA and ACO were not evaluated on instances with 2 negative weights and 5 negative weights respectively. Hence, to ensure fair comparison, we have computed average total CPU times for our approach in two ways. For comparison with the GA, the averages are computed using instances with 5 negative weights, 10 negative weights and half negative weights, whereas for comparison with ACO, the averages are computed using instances with 2 negative weights, 10 negative weights and half negative weights. Hence, we have two columns labelled ABC for each of the two models. Columns 5 and 9 report the time of ABC approach for comparison with GA, whereas columns 7 and 11 report the time of ABC approach for comparison with ACO. Again, the data for GA and ACO are taken from their respective papers.

Table 11 summarizes the results. This table reports for each approach on each instance group of 40, the number of instances for which the approach in question found result inferior to best known value (column *W*) and sumtotal of relative error (column *TE*). This table also report the number of instances in each instance group where our ABC approach found the new best known value (column *BKV-I*). Please note that for instances with positive weights both models are equivalent and optimal solutions are known. That is why a '−' is placed for these instances under *Model* and *BKV-I* columns.

From these tables, some interesting observation can be made. Results of different approaches vary according to the types of instances. ABC approach improves the best known values for more than 10 % of the instances (35 out of 320). Most of these instances are those with relatively large value of *p*. Barring few exception, there is not much difference in the performance of various approaches on the instances with small values of *p*. Only when the value of *p* is large, the performance of different approaches tend to differ significantly. However, none of the approaches

can be considered as clearly superior to others on all types of instances with large value of p. The difficulty of various approaches on instances with large value of p can be explained theoretically also on the basis of the search space size. Actually, there are $\binom{n}{p}$ solutions and for the fixed n, the number of solutions increase with increase in p till $p = \lfloor \frac{n}{2} \rfloor$. All the values of p in the instances considered here is less than $\lfloor \frac{n}{2} \rfloor$, and hence, the search space size increases with the increase in p for fixed n. As the approaches have to search a larger search space, they find these instances relatively difficult.

We can observe that our ABC approach performs better than ACO on 5 out of 7 instance groups in terms of total error, whereas reverse is true if we compare two approaches in terms of number of instances where an approach fails to reach the best known value. This shows that whenever ABC approach fails to reach the best known value, its solution is closer to best known value in comparison to the solution of ACO under similar situation. Overall, there are 62 and 66 instances (out of 280) where ACO and ABC fail to reach the best known value. ACO approach perform much worse in comparison to our ABC approach on instances with half negative weights under both the models, whereas it performs the best in comparison to ABC on instances with no negative weight.

On the other hand, GA fares better than ABC on both counts. There are 3 instance groups only where ABC performs better than GA in terms of total error, whereas on remaining 4 groups GA performs better. There is only one instance group where GA fails to reach the best known value on higher number of instances than ABC. However, GA performs much worse in comparison to ABC in terms of total error on instances with half negative weights under model P1.

As far as execution times of various approaches are concerned, GA and ACO approaches were executed on a 1.7 GHz Pentium 4 system which is different from the system used to execute our ABC approach. Therefore, execution times can not be compared precisely. However, a rough comparison can always be made. Even after compensating for differences in processing speed, we can safely say that our approach is faster than GA on most of the instances. However, ACO is faster than our approach.

6 Conclusions

In this paper, we have proposed an ABC algorithm based approach for solving the p-median problem with positive and negative weights. We have compared the results of our approach with two best approaches, viz. GA and ACO on the standard benchmark instances of the problem. Comparison with ACO approach is specially significant as both ABC and ACO are swarm intelligence based approaches. ABC approach is able to improve the best known values for slightly more than 10 % of the instances. Though the relative performance of different approaches vary according to the types of instances, the overall performance of GA is clearly better than ABC approach in terms of solution quality, but ABC approach is faster. On the other hand, ABC approach is much better than ACO approach on instances where

half of the weights are negative under both the models, but at the expense of larger execution times.

As a future work, we would like to extend our ABC approach to capacitated *p*-median problem. Similar approaches can be designed for other related facility location problems also.

References

1. Alp, O., Erkut, E., Drezner, Z.: An efficient genetic algorithm for the p-median problem. Ann. Oper. Res. **122**(1–4), 21–42 (2003)
2. Banda, J., Singh, A.: A hybrid artificial bee colony algorithm for the terminal assignment problem. In: Swarm, Evolutionary, and Memetic Computing, vol 8947 of Lecture Notes in Computer Science, pp. 134–144. Springer (2015)
3. Basti, M., Sevkli, M.: An artificial bee colony algorithm for the *p*-median facility location problem. Int. J. Metaheuristics **4**(1), 91–113 (2015)
4. Bozkaya, B., Zhang, J., Erkut, E.: An efficient genetic algorithm for the p-median problem. In: Facility Locations: Applications and Theory, Chap. 6, pp. 179–205. Springer, New York (2002)
5. Burkard, R.E., Çela, E., Dollani, H.: 2-medians in trees with pos/neg weights. Discrete Appl. Math. **105**(13), 51–71 (2000)
6. Burkard, R., Fathali, J.: A polynomial method for the pos/neg weighted 3-median problem on a tree. Math. Methods Oper. Res. **65**(2), 229–238 (2007)
7. Burkard, R., Krarup, J.: A linear algorithm for the pos/neg-weighted 1-median problem on a cactus. Computing **60**(3), 193–215 (1998)
8. Cappanera, P.: A survey on obnoxious facility location problems, Technical Report TR-99-11, Departmento di Informatica, Universita di Pisa (1999)
9. Carrizosa, E., Plastria, F.: Location of semi-obnoxious facilities. Stud. Locat. Anal. **12**(1999), 1–27 (1999)
10. Chaurasia, S.N., Singh, A.: A hybrid swarm intelligence approach to the registration area planning problem. Inf. Sci. **302**, 50–69 (2015)
11. Chiyoshi, F., Galvao, R.D.: A statistical analysis of simulated annealing applied to the *p*-median problem. Ann. Oper. Res. **96**, 61–74 (2000)
12. Dai, Z., Cheung, T.-Y.: A new heuristic approach for the p-median problem. J. Oper. Res. Soc. **48**(9), 950–960 (1997)
13. Densham, P.J., Rushton, G.: A more efficient heuristic for solving large *p*-median problems. Pap. Reg. Sci. **71**(3), 307–329 (1992)
14. Dibble, C., Densham, P.J.: Generating interestingalternatives in gis and sdss using genetic algorithms. In: Proceedings of 1993 GIS/LIS Symposium. University of Nebraska, Lincoln (1993)
15. Fathali, J.: A genetic algorithm for the *p*-median problem with pos/neg weights. Appl. Math. Comput. **183**(2), 1071–1083 (2006)
16. Fathali, J., Kakhki, H., Burkard, R.: An ant colony algorithm for the pos/neg weighted *p*-median problem. Cent. Eur. J. Oper. Res. **14**(3), 229–246 (2006)
17. Fathali, J., Kakhki, H.T.: Solving the *p*-median problem with pos/neg weights by variable neighborhood search and some results for special cases. Eur. J. Oper. Res. **170**(2), 440–462 (2006)
18. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. In: Foundations of Gentic Algorithms, pp. 69–93. Morgan Kaufmann (1990)
19. Hansen, P., Mladenović, N.: Variable neighborhood search for the *p*-median. Locat. Sci. **5**(4), 207–226 (1997)
20. Hosage, C., Goodchild, M.: Discrete space location-allocation solutions from genetic algorithms. Ann. Oper. Res. **6**(2), 35–46 (1986)
21. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. In: Technical report-TR06. Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey (2005)
22. Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N.: A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif. Intell. Rev. **42**(1), 21–57 (2014)

23. Karaboga, D., Akay, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. **39**(3), 459–471 (2007)
24. Karaboga, D., Akay, B.: On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. **8**(1), 687–697 (2008)
25. Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Appl. Math. Comput. **214**(1), 108–132 (2009)
26. Kariv, O., Hakimi, S.L.: An algorithmic approach to network location problems II: the *p*-medians. SIAM J. Appl. Math. **37**(3), 539–560 (1979)
27. Kuehn, A.A., Hamburger, M.J.: A heuristic program for locating warehouses. Manag. Sci. **9**(4), 643–666 (1963)
28. Mladenović, N., Brimberg, J., Hansen, P., Moreno-Pérez, J.A.: The *p*-median problem: a survey of metaheuristic approaches. Eur. J. Oper. Res. **179**(3), 927–939 (2007)
29. Özçakir, N., Basti, M.: Particle swarm otimization algorithm approach for solving *p*-median facility location selection problem. Istanb. Univ. Fac. Manag. J. **41**, 241–257 (2012)
30. Pan, Q.-K., Tasgetiren, M.F., Suganthan, P., Chua, T.: A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Inf. Sci. **181**(12), 2455–2468 (2011)
31. Perez, J.M., Garcia, J.R., Moreno, M.: A parallel genetic algorithm for the discrete *p*-median problem. Stud. Locat. Anal. **7**, 131–141 (1994)
32. Plastria, F.: Optimal location of undesirable facilities: an overview. Belg. J. Oper. Res. Stat. Comput. Sci. **36**, 109–127 (1996)
33. Resende, M.G., Werneck, R.F.: A hybrid heuristic for the *p*-median problem. J. Heuristics **10**(1), 59–88 (2004)
34. Resende, M.G.C., Werneck, R.F.: A fast swap-based local search procedure for location problems. Ann. Oper. Res. **150**(1), 205–230 (2007)
35. Rolland, E., Schilling, D.A., Current, J.R.: An efficient tabu search procedure for the *p*-median problem. Eur. J. Oper. Res. **96**(2), 329–342 (1997)
36. Rosing, K., ReVelle, C., Schilling, D.: A gamma heuristic for the *p*-median problem. Eur. J. Oper. Res. **117**(3), 522–532 (1999)
37. Sevkli, M., Mamedsaidov, R., Camci, F.: A novel discrete particle swarm optimization for *p*-median problem. J. King Saud Univ. Eng. Sci. **26**, 11–19 (2014)
38. Singh, A.: An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Appl. Soft Comput. **9**(2), 625–631 (2009)
39. Singh, A., Sundar, S.: An artificial bee colony algorithm for the minimum routing cost spanning tree problem. Soft Comput. **15**(12), 2489–2499 (2011)
40. Teitz, M.B., Bart, P.: Heuristic methods for estimating the generalized vertex median of a weighted graph. Oper. Res. **16**, 955–961 (1968)
41. Venkatesh, P., Singh, A.: Two metaheuristic approaches for the multiple traveling salesperson problem. Appl. Soft Comput. **26**, 74–89 (2015)
42. Whitaker, R.: A fast algorithm for the greedy interchange of large-scale clustering and median location prolems. INFOR **21**, 95–108 (1983)