Applied Soft Computing Journal 78 (2019) 481–495

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem

Pandiri Venkatesh, Alok Singh [*]

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

HIGHLIGHTS

- This paper deals with generalized covering traveling salesman problem.
- This problem is a recent variant of traveling salesman problem.
- This problem has applications in humanitarian relief transportation.
- An artificial bee colony algorithm is proposed for this problem.
- Computational results show the effectiveness of the proposed approach.

ARTICLE INFO

Article history:
Received 7 November 2018
Received in revised form 28 January 2019
Accepted 1 March 2019
Available online 7 March 2019

Keywords:
Generalized covering traveling salesman problem
Covering salesman problem
Traveling salesman problem
Artificial bee colony algorithm

ABSTRACT

The generalized covering traveling salesman problem is a recently introduced variant of the traveling salesman problem. Given a set of cities that includes depot, facilities and customer cities, starting at the depot, the salesman has to visit a subset of facilities in order to cover some of the customers so that a constraint function based on customer coverage is satisfied. The goal is to minimize the total distance traveled by the salesman. A customer is covered if it is within a coverage radius r_i of a facility i visited by the salesman. This problem finds important applications in humanitarian relief transportation and telecommunication networks. In this paper, we have proposed an artificial bee colony algorithm with variable degree of perturbation for this problem where the degree to which a solution is perturbed for generating its neighboring solution is reduced over iterations. Computational results on a wide range of benchmark instances and their analysis show the effectiveness of our proposed approach in comparison to other state-of-the-art approaches.

1. Introduction

In the field of combinatorial optimization, the traveling salesman problem (TSP) enjoys the status of one among the most studied problems and one among the problems having most applications. The TSP is to find a minimum length Hamiltonian cycle of given cities and each city has to be visited exactly once. Due to the limitations on resources or efficiency, it may not be possible to visit each and every city. In these cases, it is advantageous to use the concept of coverage, i.e., satisfying the demands of several cities by visiting a single city. The generalized covering traveling salesman problem (GCTSP) is formulated for one such case and is a variant of TSP. Shaelaie et al. [1] formulated the GCTSP by introducing the potential applications of it in humanitarian relief transportation and telecommunication networks. Given a set of n cities which includes depot, facilities and customer cities. A coverage radius r_i is associated with each facility i, and a demand d_j is associated with each customer j. The objective is to construct a minimum length tour over a subset of facilities so that the sumtotal of demands of customers covered by this subset of facilities is at least D. A customer is said to be covered by a subset of facilities if it is within the coverage radius of one or more facilities. To solve this problem, one has to find a subset of facilities that can satisfy the demand D, and also has to arrange them in an order that minimize the total distance traveled by the salesman.

While introducing GCTSP, Shaelaie et al. [1] proposed two mathematical models (Flow-based and Node-based formulations) and two metaheuristic approaches (memetic algorithm(MA) and variable neighborhood search(VNS)) to solve this problem. Both the metaheuristic approaches (MA and VNS) make use of two local searches viz. the standard 2-opt local search for TSP and drop-and-add local search. The latter local search removes a facility from the tour and in its place tries to add one or two

* Corresponding author.
E-mail addresses: venkatesh78.p@gmail.com (V. Pandiri), alokcs@uohyd.ernet.in (A. Singh).

https://doi.org/10.1016/j.asoc.2019.03.001
1568-4946/© 2019 Elsevier B.V. All rights reserved.

facilities from among its nearest N facilities so that the objective function can be improved. The 2-opt local search is applied first on a solution followed by drop-and-add local search. Each local search is applied in a random order till there is no improvement in tour length. Both MA and VNS outperformed the two mathematical formulations on the benchmark instances considered. As far as comparison between MA and VNS is concerned, for small and medium size instances, both approaches have obtained same number of best solutions, whereas for the large size instances, MA outperformed VNS by providing more number of best solutions than VNS. These are the only approaches available in the literature for this problem. There are many variations of standard TSP in the literature. However, the following problems served as motivation to Shaelaie et al. [1] for introducing the GCTSP.

The quota traveling salesman problem (QTSP) is a generalization of the TSP in which the salesman has a set of n cities with associated demands and has to visit a given quota of cities while minimizing the distance traveled [2,3]. The k-TSP is a special case of the QTSP, where each city has a one unit of demand and the goal is to satisfy the demand k by visiting k cities [4,5].

Another variant of TSP is the prize collecting traveling salesman problem (PCTSP), which is introduced by Balas [6]. Given a set of n cities, each associated with a certain prize value, the objective of the PCTSP is to collect a given amount of prize, while minimizing the traveling distance and penalty costs. The penalty cost is the cost incurred by the salesman for not visiting a city.

The covering salesman problem (CSP) is also a variant of TSP, where each city must be either visited or covered [7]. Basically, a city is covered if it is with in a predetermined distance from the visited city. The objective of the CSP is to construct a minimum length cycle over a subset of cities which cover remaining unvisited cities. The potential applications of this problems are in the fields of emergency management and disaster planning [8–11].

The GCTSP is a generalization of the TSP, CSP, k-TSP, QTSP and the PCTSP. All the applications of these problems could be easily extended to the GCTSP. In this paper, we have proposed an artificial bee colony (ABC) algorithm based approach for the GCTSP. Unlike the MA and the VNS approaches of [1] which utilize two local searches explicitly to improve the solutions obtained through them, our ABC approach does not make use of any local search explicitly. However, it is equipped with a procedure to remove redundant facilities from a newly generated solution. A redundant facility is one whose removal does not effect the feasibility of a GCTSP solution. Computational results on the same set of benchmark instances as used in [1] show the effectiveness of our ABC approach.

The remainder of the paper is organized as follows. Section 2 formally defines the GCTSP. Our proposed ABC approach for GCTSP is described in Section 3. Section 4 presents the computational results on benchmark instances and analyses them. Finally, Section 5 outlines some concluding remarks and directions for future research.

2. Problem definition

Given a graph $G = (V, E)$, where V consists of three sets of vertices, viz. the depot, $\{0\}$, the set of facilities F, and the set of customers C. Hence, $V = \{0\} \cup F \cup C$. Further, $E = \{(i,j) \mid i,j \in F \cup \{0\}\}$. A cost or distance t_{ij} is associated with each edge $(i,j) \in E$, and every customer i has a demand, d_i, that will be satisfied by a tour when it is within the coverage distance of at least one facility on that tour. The facilities that are part of the tour are called visited facilities. The GCTSP seeks a tour of minimum length over $\{0\} \cup F$ that begins and ends at the depot, $\{0\}$ so that the sumtotal of the satisfied demands of the customers is at least D. Note that, only a subset F' of the set of facilities F ($F' \subseteq F$) that can satisfy the total demand of at least D needs to be part of a tour. Assume $|V| = n$, and $|F'| = m$. By introducing binary variables, x_{ij} to indicate whether edge (i,j) is part of the salesman tour (i.e., $x_{ij} = 1$) or not (i.e., $x_{ij} = 0$), y_{ij} to indicate whether the demand of customer $i \in C$ is satisfied by the facility $j \in F$ (i.e., $y_{ij} = 1$) or not (i.e., $y_{ij} = 0$), and another binary variable z_i to indicate whether a facility $i \in F$ is visited by the salesman tour (i.e., $z_i = 1$) or not (i.e., $z_i = 0$), the mathematical model of the GCTSP can be formulated as follows:

$$\text{Minimize} \quad \sum_{(i,j)\in E} t_{ij} x_{ij} \tag{1}$$

subject to:

$$\sum_{i\in C} \sum_{j\in F'} d_i y_{ij} \geq D, \tag{2}$$

$$\sum_{j\in F} y_{ij} \leq 1 \quad \forall i \in C, \tag{3}$$

$$\sum_{j\in F} x_{0j} = 1 = \sum_{j\in F} x_{j0} \tag{4}$$

$$\sum_{(i,j)\in E} x_{ij} + \sum_{(j,i)\in E} x_{ji} = 2z_i \quad i \in F \tag{5}$$

$$\sum_{i\in S} \sum_{j\in S} x_{ij} \leq |S| - 1, \quad \forall S \cup \{0\} \subset F' \cup \{0\} \subset F \cup \{0\}, \\ 2 \leq |S| \leq m - 2 \tag{6}$$

$$x_{ij}, y_{ij}, z_i \in \{0, 1\} \quad \left[\forall i,j \in F \cup \{0\}\right], \left[\forall i \in C, \quad j \in F\right], \\ \left[\forall i \in F \cup \{0\}\right] \tag{7}$$

The objective function (1) minimizes the total distance traveled by the salesman. Eq. (2) satisfies the given demand D by a subset of facilities F' that are part of the tour. Eq. (3) represents the constraint that each customer must be allocated to at most one facility for satisfying that customer's demand. (4) assures that the tour must begin and end at the depot, $\{0\}$. Eq. (5) satisfies the constraints of indegree and outdegree of the visited facilities. Eq. (6) represents the sub tour elimination constraints. Finally, Eq. (7) implies the binary nature of decision variables x_{ij}, y_{ij} and z_i. Fig. 1 illustrates GCTSP with the help of an example. Fig. 1(a) depicts a GCTSP instance where the depot, facilities and customer vertices are shown as black triangle, red squares and blue circles respectively. There are 36 customers, 14 facilities and one depot leading to a total of 51 vertices. A feasible solution to this GCTSP instance is shown in Fig. 1(b) where 4 facilities are visited by the salesman out of 14 facilities and the demands of 24 customers which are within the coverage radius of one or more of these 4 visited facilities are met.

3. Artificial bee colony algorithm for the GCTSP

We have developed an artificial bee colony algorithm based approach for the GCTSP. The artificial bee colony (ABC) algorithm is a population based metaheuristic approach that was proposed by Karaboga in 2005 [12] on getting inspiration from collective intelligent behavior exhibited by natural honey bee swarm while foraging. Now, a number of different variants of the basic ABC algorithm exists in the literature, e.g. [13–23]. In this section, a brief overview of the basic ABC algorithm is presented and then the details about the proposed ABC approach for the GCTSP.

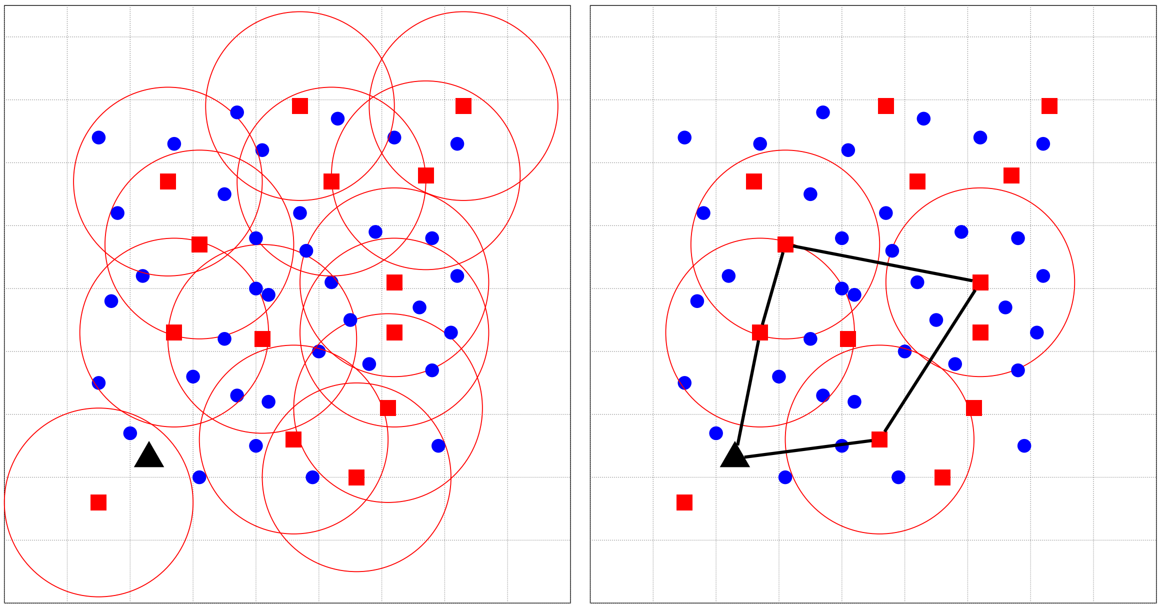

(a) A GCTSP instance with 51 cities (b) A feasible solution

Fig. 1. Illustration of a feasible solution for the GCTSP.

3.1. Overview of the artificial bee colony algorithm

Based on the nature of work performed by foraging bees, they can be divided into three categories, viz. employed, onlooker and scout bees. Those bees which are currently exploiting food sources are termed as "employed". Those bees which are waiting in the hive and yet to select their food sources are termed as "onlooker". The last category of "scout" consists of those bees which at present are in search of new food sources in the neighborhood of the hive. In a colony of bees, the process of foraging begins by deputing scout bees to search for new food sources in the neighborhood of the hive. This search is carried out in a purely random manner. As soon as a scout bee discovers a food source, it begins exploiting the discovered food source and its category gets changed from scout to employed. The job of the employed bees is to transport the loads of nectar from the food sources to the hive and then inform onlooker bees about the quality (nectar content) of the food sources currently being exploited by them. Each employed bees convey the information about its food source to onlooker bees by means of a dance which is performed in a common area of the hive. After observing the dances performed by several employed bees, onlooker bees select food sources in a stochastic manner where the probability of selection of a food source is proportional to its nectar content. Hence, a food source having higher nectar content attracts more onlookers than a food source with lower nectar content. After the dance, the employed bee goes back to the food source it is currently exploiting taking along all those onlooker bees which select this food source. All these onlooker bees also begin exploiting this food source and their category gets changed from onlooker to employed. When a food source is exhausted, then it is deserted by all the employed bees which are currently exploiting it. Such employed bees either become scouts or onlookers.

By drawing an analogy between the stochastic search process performed by a metaheuristic while searching for an optimal solution and the intelligent behavior exhibited by bees while foraging, the scout bees can be seen as doing the job of exploration, whereas employed and onlooker bees can be seen as doing the job of exploitation. Motivated by this observation, Karaboga [12] developed the ABC algorithm. In the ABC algorithm, each food source and its nectar content respectively represent a prospective solution to the problem under consideration and its fitness. Throughout this paper, food source and solution are used interchangeably. Like real bees, this algorithm also divides the artificial bees into three categories, viz. scout, employed, and onlooker bees. Each employed bee is linked with a unique food source (solution), i.e., the number of employed bees is equal to the number of food sources. Usually, but not always, the number of onlooker bees are also taken to be equal to the number of employed bees. An employed bee after deserting its linked food source can never become an onlooker. It can only become scout. A new food source (solution) is created for such a scout and it gets linked with this new food source to again become employed. The new food source for the scout can be created either in a purely random fashion or using some specific strategy.

Basically, artificial bee colony algorithm is an iterative algorithm and consists of three phases, viz. employed bee phase, onlooker bee phase and scout bee phase. Starting with a population of initial solutions each of which is linked with an employed bee, these three phases are repeated until the termination condition is satisfied. These phases are described below.

1. **Employed bee phase:** In this phase, each employed bee determines a new solution in the neighborhood of its currently linked solution, and calculates the fitness of the new solution. If the fitness of the new solution is better than the currently linked solution then that employed bee gets linked to new solution by deserting the currently linked solution, otherwise it will retain the current solution. After all employed bees have gone through this process, they share the information of their solutions with onlookers. Basically, in this phase every solution gets a chance for improvement.
2. **Onlooker bee phase:** In this phase, each of the onlooker bees chooses a solution by following a stochastic strategy which basically gives preference to good solutions over bad ones. Once the onlookers have chosen their solutions, each of them determines a new solution in the neighborhood of their respective selected solutions and calculates its fitness.

Algorithm 1: Pseudo code of basic ABC

```
Input: Set of parameters for ABC
Output: Best solution found
Initialization_of_Solutions();
while Termination condition is not satisfied do
    Employed_Bee_Phase();
    Onlooker_Bee_Phase();
    Scout_Bee_Phase();
return best;
```

The best solution among all the neighboring solutions determined by the onlookers linked with a particular solution S and the solution S itself is selected as new solution S for the next iteration. Please note that as part of this phase, some solutions may get multiple chances and some solutions may not at all get a chance to improve.

3. **Scout bee phase:** This phase is useful in cases where a solution has not improved over a certain number of iterations and then that solution is considered as fully explored and employed bee linked with it deserts it to become a scout. How this scout bee is handled is already explained. In essence, in this phase a solution which has not improved since long is replaced with a new solution.

The crux of ABC algorithm is based on a simple fact that the chances of finding better solutions are higher in the neighborhood of good solutions compared to the neighborhood of bad solutions, so the neighborhood of good solutions must be explored thoroughly. This is the reason behind using more onlookers for exploring the neighborhood of good solutions. However, if a solution is locally optimal with respect to the entire neighborhood then it be improved no matter how many chances are given to it. The concept of scout bees is helpful in this situation. Checking whether a solution is locally optimal or not can be a computationally expensive job as one has to explore the entire neighborhood to ascertain this. To avoid this, the ABC algorithm assumes a solution to be locally optimal and discards it, if it has not improved over certain number of iterations say '$limit_{scout}$'. A new solution replaces the discarded solution through the concept of scout bee. The effectiveness of any metaheuristic algorithm lies in maintaining a proper balance between the exploration and exploitation throughout the execution of the algorithm. The parameter $limit_{scout}$ is responsible for this balance in the ABC algorithm, and thus, utmost care needs to be taken while determining its value. A lower value of this parameter favors exploration over exploitation. On the other hand, higher value favors exploitation over exploration. The pseudo code of basic ABC is given in Algorithm 1.

3.2. Proposed artificial bee colony (ABC) algorithm for GCTSP

The main components of our artificial bee colony algorithm for the GCTSP are discussed in following subsections.

3.2.1. Solution encoding

To represent the solution of GCTSP, we have used the encoding which represents a solution as a permutation of visited facilities. This encoding is also used in [1].

Theorem 1. *By recalling the given graph $G = (V, E)$, where $V = \{0\} \cup F \cup C$ is a set of vertices including the depot, $\{0\}$, the set of facilities, F, and the set of customers, C. Here, the number of cities, $|V| = n$, and number of facilities, $|F| = m$. The search space size of the GCTSP according to our encoding is $\sum_{k=1}^{m} \binom{m}{k} \times k!$*

Proof. The GCTSP can possibly have $1 \leq k \leq m$ facilities. These k facilities can be chosen from given m facilities in $\binom{m}{k}$ ways. And the chosen k facilities can be ordered in $k!$ ways. Therefore, the search space size of the GCTSP is $\sum_{k=1}^{m} \binom{m}{k} \times k!$ □

3.2.2. Fitness

The fitness function is same as the objective function given in Eq. (1), i.e., a solution's fitness is the total distance traveled by the salesman. Therefore, a solution with a lower value of the fitness function is considered to be better than a solution with a higher value.

3.2.3. Initialization of solutions

Our initial solution generation procedure follows an iterative process where a facility is selected in each iteration. During each iteration, one of the following three heuristics is chosen uniformly at random to select a facility which is inserted into the best possible position in the salesman's tour. To find this, one has to check all the possible insertion positions and then the insertion is carried out at the position which yields the least increase in the cost. This procedure is repeated until a feasible solution is obtained, i.e., one satisfying the given demand.

- *Least insertion cost*: This heuristic selects a facility randomly from the top three facilities which yields less increase in the cost while inserting them in to the salesman tour.
- *Covering the maximum uncovered*: This heuristic selects a facility randomly from the top three facilities which covers a largest number of uncovered customers.
- *Completely random*: This heuristic selects a facility uniformly at random from the facilities which are able to cover at least one uncovered customer.

The strategy of considering top three facilities instead of only the top is taken from [1] and is motivated by the intention to generate diverse solutions.

Three heuristics were also used in [1] to generate the initial solutions. The first heuristic of [1] is a modified form of nearest neighbor heuristic for TSP, where instead of selecting the nearest unvisited facility at each step, a facility among top three nearest unvisited facilities is selected at random. The second and third heuristics are similar to our second and third heuristics. However, while generating an initial solution, unlike our approach, a facility is always added at the last position in [1] instead of best possible position. Further, in our approach, none of the three heuristics generates an initial solution in-toto. In our approach, each initial solution is generated by combination of these three heuristics where during each iteration one of the three heuristics is chosen at random to select an unvisited facility. On the other hand, in [1], the first heuristic alone is used to generate 50% of initial solutions, whereas the second heuristic alone is used to generate 30% of initial solutions. 15% solutions were generated using a combination of first and second heuristic where during each iteration a facility is selected either by first heuristic or by second heuristic. Remaining 5% solutions were generated using the third heuristic alone.

3.2.4. Probability of choosing a food source

Binary tournament selection method is used for choosing a food source for an onlooker bee. In this method, two different food sources (employed bee solutions) are selected uniformly at random from the population and a comparison is made between their fitnesses. The better one of the two food sources is chosen with probability P_{bts}. Otherwise the worse one is chosen, i.e., the probability of selection of the worst of the two solutions is $(1 - P_{bts})$.

3.2.5. Generating a neighboring food source

An effective neighboring solution generation process should exploit various problem aspects in such a manner so as to guide the search process towards an optimal solution. Solving GCTSP involves two crucial aspects. First, selecting a subset of facilities to be part of the tour and then the ordering among the selected facilities. Our neighboring solution generation process is designed keeping in mind these two aspects. Hence, it makes use of two methods which are used one after the other. These two methods are described below:

1. ***Subset-DRO:*** This method handles the subset selection aspects of facilities in GCTSP. This method is a destroy and repair operator which partially destroys the tour and then repairs it. Each facility in the tour is removed with a probability D_p. This destroy procedure makes the solution infeasible. To make the solution feasible, it is repaired by adding some facilities to the solution under consideration. As part of the repair, following score, which is introduced in [1], is computed for each facility i that is not part of the tour:

$$Score(i) = \frac{AdditionCover(i)}{(AdditionCost(i) + 1)} \tag{8}$$

   Here, $AdditionCover(i)$ is the total number of uncovered customers that will be covered by adding facility 'i' into the tour. $AdditionCost(i)$ is the increase in cost of the tour by adding facility 'i' into its best position in the tour. The facility which has the best score is added at its best possible position in the tour. The procedure of adding facilities into the tour repeats until the solution under consideration becomes feasible. Note that a facility to be added must have at least one uncovered customer. This repair operator is similar to addition procedure used by VNS in [1] with one small difference. In our case, always the facility with the maximum score is added, whereas the addition procedure of [1] choose one facility uniformly at random from among top δ facilities with maximum scores. By following the policy of adding the facility with maximum score has avoided a sort of the facilities available for addition in each iteration, thereby yielding some savings in computational time. The feasible solution obtained after adding the facilities may contain redundant facilities. A redundant facility is the one whose removal does not effect the feasibility of a GCTSP solution, i.e., satisfying the given demand. If there are multiple redundant facilities, then a facility is removed which yields maximum decrease in the cost of the solution. The removal procedure continues till there is no redundant facility left in the solution. If the resulting solution after this method is better than the original input solution then this solution replaces the original input solution, otherwise the original input solution is passed to the next method. It is to be noted that there is no provision in MA and VNS of [1] to deal with redundant facilities and such facilities remain in the solutions thereby adversely affecting their qualities.
2. ***Permut-DRO:*** This method handles the permutation aspect of facilities in GCTSP. This method is also a destroy and repair operator like *Subset-DRO*. In this method, some visited facilities are removed from the solution under consideration and then these deleted facilities are reinserted at their best possible positions only to restore the feasibility. In this method, first, each visited facility is removed with probability D_p. All these removed facilities are added to a set, and then, one-by-one, a facility is chosen at random from this set and inserted at best possible position in the tour.

   The main difference between these two methods is that, in the first method, the facilities constituting the tour and their order in the tour may change, whereas in the second method there is no change in the constituent facilities. However, their order in the tour can change.

Another crucial factor for the success of an ABC approach lies in correctly estimating, how strong the perturbations need to be while generating a neighboring solution. In the initial iterations, higher degree of perturbation helps in improving the solution quality quickly. However, during final iterations, higher degree of perturbation may fail to improve a solution as during final iterations, a solution is very near to an optimal/locally optimal solution and perturbing it too much may move this solution away from the optimal/locally optimal solution. Hence, we have utilized a variable degree of perturbation in our ABC approach. The degree of perturbation is directly controlled by parameter D_p. The main idea of this strategy is that the degree of perturbation needs to be high in the initial iterations, and it has to be reduced as the algorithm progresses in order to get good solutions. The parameter D_p varies over iterations from max_{dp} to min_{dp}. The value of D_p in an iteration $iter$ is calculated as follows:

$$D_p := \left(\frac{max_{dp} - min_{dp}}{iter_{max}}\right)(iter_{max} - iter) + min_{dp} \tag{9}$$

Here $iter_{max}$ is maximum number of iterations up to which D_p can be varied. The value of D_p be decreased beyond a point, otherwise there will be no perturbation and generated neighboring solution will be identical to the original solution. Hereafter, our approach will be referred to as ABC(VDP)

We have not utilized any explicit local search like 2-opt and drop-and-add local searches of [1] to improve the solution obtained through neighboring solution generation process. In fact, if neighboring solution generation process is designed properly keeping in mind various crucial aspects of a problem, explicit local searches will be redundant in most cases. Our computational results prove this point. The pseudo-code for generating a neighboring solution can be seen in Algorithm 2.

3.2.6. Other features

We have used different number of employed bees and onlooker bees which is different from the practice of using the same number of employed and onlooker bees as suggested with original ABC algorithm. Actually, the aim of onlooker bee phase is to provide more chances to the better solutions to improve themselves. It is based on the simple fact that chances of finding even better solutions in the vicinity of good solutions is more in comparison to poor solutions. Now, how many more chances good solutions need in comparison to poor ones depend on the fitness landscape. Usually, for a permutation problems like GCTSP, there is a huge difference in fitness between the good solutions and bad solutions and neighboring solution generation procedure more often than not bridge this gap. Hence, for such problems good solutions need to be given much more chances than what they usually get by taking equal number of employed and onlooker bees. This reason has motivated us to use more number of onlooker bees than employed bees. As described in next section, the empirically determined values for number of employed bees and number of onlooker bees through Taguchi's method also proves our decision correct.

If an employed bee solution does not improve over a fixed number of trials $limit_{scout}$, then the corresponding employed bee abandons that food source and becomes a scout. By number of trials for a solution, we mean the number of times that solution is used for neighboring solution generation. A solution can get more than one trial in an iteration depending on the number of onlookers selecting that solution. Therefore, the number of scout bees in an iteration is equal to the number of food sources

Algorithm 2: Neighboring solution generation

```
Input: A solution $S$
Output: A neighboring solution $S'$
begin
    $S' \leftarrow S$;                                                        ⎫
    foreach visited facility $f$ in $S'$ do                                   ⎪
        Generate a random number $r$ such that $0 \leq r \leq 1$;             ⎪
        if $r < D_p$ then                                                     ⎪
            Remove $f$ from $S'$;                                             ⎪
        else                                                                  ⎬ Subset-DRO method.
            Keep $f$ in $S'$;                                                 ⎪
    while $S'$ is infeasible do                                               ⎪
        Add an unvisited facility by following procedure described in $Subset-DRO$ method;
    $S \leftarrow S'$;                                                        ⎭

    $S' \leftarrow \emptyset$;                                                ⎫
    $U \leftarrow \emptyset$;                                                 ⎪
    foreach visited facility $f$ in $S$ as per their order in $S$ do          ⎪
        Generate a random number $r$ such that $0 \leq r \leq 1$;             ⎪
        if $r < D_p$ then                                                     ⎪
            Add $f$ to $U$;                                                   ⎬ Permut-DRO method.
        else                                                                  ⎪
            Copy $f$ into $S'$;                                               ⎪
    foreach facility $f$ in $U$ in some random order do                       ⎪
        Insert $f$ into $S'$ using the $Permut-DRO$ method;                   ⎪
    return $S'$;                                                              ⎭
```

that has not improved over last $limit_{scout}$ trials. This number can be zero or one or more than one. Please note that some ABC implementations limit the maximum number of scouts to one in an iteration. In our ABC approach, there is no such restriction.

Since we are reducing the degree of perturbation over iterations, assigning a random solution to this scout bee will not be a good move as degree of perturbation at the time of replacement will not be suitable for quick improvement of this randomly generated solution. So, we assign neighboring solution of just abandoned solution, even if it is worse than the just abandoned solution, to the scout bee to make it employed again. The reason for assigning worst solutions is not to waste the time in unworthy locally optimum solutions.

The pseudo-code of our ABC approach is given in Algorithm 3, where n_{eb} and n_{ob} are, respectively, the number of employed and the number of onlooker bees. *Initialize()* is a function that generates and returns an initial solution as per the procedure described in Section 3.2.3. *New_Solution(S)* function returns a new solution in the neighborhood of the solution S as per the procedure described in Section 3.2.5. *Select_Solution*$(S_1, S_2, \ldots, S_{n_{eb}})$ function selects a solution for an onlooker bee from employed bee solutions $S_1, S_2, \ldots, S_{n_{eb}}$ using binary tournament selection method (Section 3.2.4) and returns the index of the solution selected.

3.2.7. Complexity analysis of ABC(VDP)

For neighboring solution generation, ABC(VDP) makes use of two methods, viz. Subset-DRO and Permut-DRO one after the other as described in Section 3.2.5. Recalling the notational convention that number of cities, number of facilities, and number of customers are $|V|$, $|F|$ and $|C|$ respectively, and demand is D. At maximum, a solution(tour) of GCTSP can have all the $|F|$ facilities. In the first method Subset-DRO, at the maximum all the $|F|$ facilities may be removed as part of the destroy procedure so its complexity is $\mathcal{O}(|F|)$. In order to make the tour feasible, addition procedure is followed which works in an iterative manner by adding one facility at a time based on maximum score. For each available facility, the numerator of Eq. (8) can be computed in $\mathcal{O}(|C|)$ and denominator in $\mathcal{O}(|F|)$. There are $\mathcal{O}(|F|)$ iterations of addition procedure and $\mathcal{O}(|F|)$ are available in each iteration. Hence, complexity of addition procedure is $\mathcal{O}(|F|^2(|C| + |F|))$. The complexity of redundancy removal is $\mathcal{O}(|F|^2|C|)$ as there are $\mathcal{O}(|F|)$ facilities each of which can be checked for redundancy in $\mathcal{O}(|C|)$ and it can iterate for $\mathcal{O}(|F|)$ iterations. Therefore, the Subset-DRO method has the complexity of $\mathcal{O}(|F|^2(|C|+|F|))$. In the Permut-DRO method only the removed facilities are added back to the tour at their best possible position. Therefore, the Permut-DRO method has the complexity of $\mathcal{O}(|F|^2)$. Hence, the complexity of neighboring solution generation in ABC(VDP) is $\mathcal{O}(|F|^2(|C| + |F|))$. Scout bee phase in ABC(VDP) also utilize neighboring solution only, and hence, complexity of processing each scout bee is also $\mathcal{O}(|F|^2(|C| + |F|))$. As can be clearly seen from Algorithm 3, maximum number of scout bees in an iteration is n_{eb}. Employed bee phase and onlooker bee phase call the neighboring solution generation procedure n_{eb} and n_{ob} times respectively. Since each iteration of ABC(VDP) calls neighboring solution generation procedure finite number of times (bounded by $2n_{eb} + n_{ob}$) which is independent of the values of $|F|$ and $|C|$, complexity of each iteration of ABC(VDP) algorithm is $\mathcal{O}(|F|^2(|C| + |F|))$.

Arguing in a manner similar to Subset-DRO, and, observing that the complexities of first, second and third heuristic of initial solution generation procedure are $\mathcal{O}(|F|^2)$, $\mathcal{O}(|F||C|)$, and $\mathcal{O}(|F|)$ respectively, and, only one of them is called in each iteration of initial solution generation, the complexity of initial solution generation procedure is $\mathcal{O}(|F|^2 max(|C|, |F|))$ which is same as that of neighboring solution generation (Since $\mathcal{O}(max(\alpha, \beta)) = \mathcal{O}(\alpha + \beta)$).

4. Computational results

For testing the performance of ABC(VDP), we have used the same test instances as used in [1]. Shaelaie et al. [1] used 115 instances to evaluate the performance of their approaches. The number of vertices in these instances range from 51 to 1000. Instances having number of vertices up to 76, between 100 & 200 and between 535 & 1000 are categorized as small, medium, and large size instances respectively. All the 115 instances are Euclidean and contain the co-ordinates of their constituent vertices. The first vertex in these instances is depot, whereas the

Algorithm 3: Pseudo code of our ABC approach

```
Input: Set of parameters for the ABC Algorithm and a GCTSP instance
Output: Best solution found
for $i \leftarrow 1$ to $n_{eb}$ do                                          } Initialization phase
    $S_i \leftarrow$ Initialize();
$best \leftarrow$ best solution among $S_1, S_2, \ldots, S_{n_{eb}}$;
while Termination condition not satisfied do
    for $i \leftarrow 1$ to $n_{eb}$ do                                      } Employed bee phase
        $S' \leftarrow$ New_Solution($S_i$);
        if $S'$ is better than $S_i$ then
            $S_i \leftarrow S'$;
    for $i \leftarrow 1$ to $n_{ob}$ do                                      } Onlooker bee phase
        $p \leftarrow$ Select_Solution($S_1, S_2, \ldots, S_{n_{eb}}$);
        $S' \leftarrow$ New_Solution($S_p$);
        if $S'$ is better than $S_p$ then
            $S_p \leftarrow S'$;
    for $i \leftarrow 1$ to $n_{eb}$ do                                      } Memorizing best solution
        if $S_i$ is better than $best$ then                                    &
            $best \leftarrow S_i$;                                             dealing with scouts
        else if $S_i$ has not improved over last $limit_{scout}$ trials then
            $S_i \leftarrow$ New_Solution($S_i$);
return $best$;
```

Table 1
Parameter settings of ABC.

Parameter	Description	Test set	Best value
n_{eb}	Number of employed bees	$\{75, 100, 125\}$	100
n_{ob}	Number of onlooker bees	$\{125, 150, 175\}$	150
$limit_{scout}$	Number of iterations without improvement after which a solution can be discarded	$\{25, 50, 75\}$	50
P_{bts}	Probability of selecting better of the two solutions in binary tournament selection	$\{0.7, 0.8, 0.9\}$	0.8
min_{dp}	Minimum probability to destroy the salesman tour in both methods of neighboring solution generation	$\{0, 0.1, 0.2\}$	0.1
max_{dp}	Maximum probability to destroy the salesman tour in both methods of neighboring solution generation	$\{0.8, 0.9, 1.0\}$	1.0

remaining vertices are partitioned into facilities and customers. Each customer is supposed to have unit demand, and the value of D is chosen to be largest integer ℓ less than or equal to either 50% or 75% or 100% of number of vertices to get at least that ℓ customers covered. In these instances, the set of customers covered by a facility is explicitly specified instead of specifying a coverage radius. There are 32 small instances, 69 medium instances and 14 large instances.

Since ABC algorithm is a stochastic algorithm, therefore, choosing proper parameter values is crucial for the success of ABC(VDP). We have used the Taguchi's method [24,25] to determine various parameter values. There are six parameters in our approaches. We have identified three candidate parameter values for each of these six parameters. These six parameters along with their description and candidate values are listed in Table 1. These candidate parameter values are arrived at by considering literature on ABC algorithm, our own previous experience with ABC algorithm, and some preliminary experimentations. In order to determine the value of six parameters each having three candidate values as per Taguchi's method, 18 experiments involving various combinations of candidate parameter values were performed. The combination of parameter values for each of these experiments were determined as per the composition of the rows of $L18$ orthogonal array [24]. Based on these experiments, we have used a population of 100 employed bees ($n_{eb} = 100$), 150 onlooker bees ($n_{ob} = 150$), $P_{bts} = 0.8$, $limit_{scout} = 50$ and the value of D_p ranges from 1.0 to 0.1, i.e. $max_{dp} = 1.0$ and $min_{dp} = 0.10$ in subsequent experiments. These values are also reported in the last column of Table 1.

To show the advantage of variable degree of perturbation, we have also implemented a version of ABC algorithm with fixed degree of perturbation where we have taken $D_p = 0.5$ throughout. This version with fixed degree of perturbation will be referred to as ABC(FDP). Except for the value of D_p, this version is same as ABC(VDP) in all other aspects.

We have implemented ABC(VDP) and ABC(FDP) in C and executed them on a PC with a 3.10 GHz Intel Core-i5 processor with 4 GB RAM. For benchmarking, we have compared our approaches with previously proposed approaches in [1], viz. Flow-based formulation, Node-Based formulation, memetic algorithm and variable neighborhood search, which are represented as Flow-based formulation, Node-based formulation, MA and VNS in the following tables. Our approaches are executed on each test instance

Table 2
Comparison of various approaches on small size instances.

Name	$\lvert V\rvert$	$\lvert F\rvert$	$\lvert C\rvert$	D	*Flow-based formulation*			*Node-based formulation*			*MA*		*VNS*		*ABC(FDP)*		*ABC(VDP)*	
					Obj.	Gap(%)	Time	Obj.	Gap(%)	Time	Average obj.	Time	Average obj.	Time	Average obj.	Time	Average obj.	Time
S1	51	15	35	17	**71.33**	0.00	3.00	**71.33**	0.00	0.75	**71.33**	0.13	**71.33**	0.05	**71.33**	0.16(0.22)	**71.33**	0.15(0.20)
S2	51	15	35	26	**105.98**	0.00	3.25	**105.98**	0.00	1.08	**105.98**	0.12	**105.98**	0.06	**105.98**	0.20(0.27)	**105.98**	0.20(0.27)
S3	51	15	35	35	**172.36**	0.00	3.25	**172.36**	0.00	0.08	**172.36**	0.09	**172.36**	0.07	**172.36**	0.23(0.31)	**172.36**	0.23(0.31)
S4	51	20	30	15	**75.29**	0.00	11.78	**75.29**	0.00	3.55	**75.29**	0.14	**75.29**	0.06	**75.29**	0.18(0.25)	**75.29**	0.17(0.23)
S5	51	20	30	22	**102.59**	0.00	37.83	**102.59**	0.00	9.47	**102.59**	0.17	**102.59**	0.07	**102.59**	0.24(0.33)	**102.59**	0.20(0.27)
S6	51	25	25	12	**38.49**	0.00	1.25	**38.49**	0.00	0.24	**38.49**	0.12	**38.49**	0.05	**38.49**	0.15(0.20)	**38.49**	0.15(0.20)
S7	51	25	25	18	**82.38**	0.00	131.23	**82.38**	0.00	25.28	**82.38**	0.17	**82.38**	0.07	**82.38**	0.20(0.27)	**82.38**	0.20(0.27)
S8	51	25	25	25	**140.62**	0.00	108.05	**140.62**	0.00	4.05	**140.62**	0.16	**140.62**	0.10	**140.62**	0.21(0.29)	**140.62**	0.21(0.29)
S9	52	16	35	17	**1378.45**	0.00	1.83	**1378.45**	0.00	0.67	**1378.45**	0.12	**1378.45**	0.05	**1378.45**	0.16(0.22)	**1378.45**	0.16(0.22)
S10	52	16	35	26	**2198.91**	0.00	4.05	**2198.91**	0.00	0.59	**2198.91**	0.11	**2198.91**	0.05	**2198.91**	0.20(0.27)	**2198.91**	0.20(0.27)
S11	52	21	30	15	**669.76**	0.00	0.56	**669.76**	0.00	0.19	**669.76**	0.13	**669.76**	0.06	**669.76**	0.15(0.20)	**669.76**	0.15(0.20)
S12	52	21	30	22	**1554.50**	0.00	60.91	**1554.50**	0.00	4.08	**1554.50**	0.15	**1554.50**	0.06	**1554.50**	0.17(0.23)	**1554.50**	0.17(0.23)
S13	52	21	30	30	**3910.04**	0.00	173.47	**3910.04**	0.00	13.94	**3910.04**	0.14	**3910.04**	0.10	**3910.04**	0.18(0.25)	**3910.04**	0.18(0.25)
S14	52	26	25	12	**572.13**	0.00	0.47	**572.13**	0.00	0.97	**572.13**	0.15	**572.13**	0.06	**572.13**	0.15(0.20)	**572.13**	0.15(0.20)
S15	52	26	25	18	**1240.67**	0.00	249.61	**1240.67**	0.00	53.17	**1240.67**	0.14	**1240.67**	0.07	**1240.67**	0.18(0.25)	**1240.67**	0.18(0.25)
S16	52	26	25	25	**2958.78**	0.00	1179.13	**2958.78**	0.00	746.22	**2958.78**	0.15	**2958.78**	0.12	**2958.78**	0.21(0.29)	**2958.78**	0.21(0.29)
S17	52	16	35	17	**1340.29**	0.00	2.19	**1340.29**	0.00	0.81	**1340.29**	0.15	**1340.29**	0.06	**1340.29**	0.15(0.20)	**1340.29**	0.14(0.19)
S18	52	16	35	26	**2291.58**	0.00	10.77	**2291.58**	0.00	2.11	**2291.58**	0.13	**2291.58**	0.06	**2291.58**	0.20(0.27)	**2291.58**	0.20(0.27)
S19	52	21	30	15	**878.30**	0.00	3.36	**878.30**	0.00	0.38	**878.30**	0.15	**878.30**	0.05	**878.30**	0.14(0.19)	**878.30**	0.14(0.19)
S20	52	21	30	22	**1521.64**	0.00	45.94	**1521.64**	0.00	6.88	**1521.64**	0.14	**1521.64**	0.06	**1521.64**	0.20(0.27)	**1521.64**	0.20(0.27)
S21	52	21	30	30	**3590.08**	0.00	38.06	**3590.08**	0.00	5.17	**3590.08**	0.13	**3590.08**	0.10	**3590.08**	0.18(0.25)	**3590.08**	0.18(0.25)
S22	52	26	25	12	**479.76**	0.00	0.42	**479.76**	0.00	0.81	**479.76**	0.13	**479.76**	0.05	**479.76**	0.15(0.20)	**479.76**	0.15(0.20)
S23	52	26	25	18	**1334.81**	0.00	432.09	**1334.81**	0.00	44.74	**1334.81**	0.15	**1334.81**	0.07	**1334.81**	0.18(0.25)	**1334.81**	0.18(0.25)
S24	52	26	25	25	**2894.96**	0.00	3484.84	**2894.96**	0.00	578.73	**2894.96**	0.19	**2894.96**	0.09	**2894.96**	0.19(0.26)	**2894.96**	0.19(0.26)
S25	76	23	52	26	**98.94**	0.00	130.92	**98.94**	0.00	30.03	**98.94**	0.16	**98.94**	0.07	**98.94**	0.21(0.29)	**98.94**	0.21(0.29)
S26	76	23	52	39	**147.89**	0.00	358.36	**147.89**	0.00	45.19	**147.89**	0.15	**147.89**	0.09	**147.89**	0.35(0.48)	**147.89**	0.29(0.40)
S27	76	30	45	22	**86.16**	0.00	78.69	**86.16**	0.00	50.33	**86.16**	0.17	**86.16**	0.08	**86.16**	0.23(0.31)	**86.16**	0.21(0.29)
S28	76	30	45	33	**119.54**	0.00	141.55	**119.54**	0.00	38.92	**119.54**	0.19	**119.54**	0.09	**119.54**	0.27(0.37)	**119.54**	0.26(0.36)
S29	76	38	37	18	**57.79**	0.00	57.94	**57.79**	0.00	34.25	**57.79**	0.14	**57.79**	0.08	**57.79**	0.22(0.30)	**57.79**	0.22(0.30)
S30	76	38	37	27	**106.39**	15.31	3600.00	**106.39**	16.07	3600.00	**106.39**	0.24	**106.39**	0.10	**106.39**	0.25(0.34)	**106.39**	0.25(0.34)
S31	76	38	37	37	**173.59**	16.03	3600.00	**173.59**	0.00	1846.03	**173.59**	0.27	**173.59**	0.21	**173.59**	0.41(0.56)	**173.59**	0.28(0.38)
S32	76	38	37	18	**20 279.16**	0.00	2653.58	21 638.57	33.99	3600.00	**20 279.16**	0.15	**20 279.16**	0.09	**20 279.16**	0.18(0.25)	**20 279.16**	0.17(0.23)
No. Best					**32**			31			**32**		**32**		**32**		**32**	
Average					**1583.54**	0.98	519.01	1626.02	1.56	335.90	**1583.54**	0.15	**1583.54**	0.08	**1583.54**	0.20(0.27)	**1583.54**	0.19(0.26)

Table 3
Comparison of various approaches on medium size instances.

Name	$\|V\|$	$\|F\|$	$\|C\|$	D	Flow-based formulation			Node-based formulation			MA		VNS		ABC(FDP)		ABC(VDP)	
					Obj.	Gap(%)	Time	Obj.	Gap(%)	Time	Average obj.	Time	Average obj.	Time	Average obj.	Time	Average obj.	Time
M1	100	30	69	34	**6043.13**	15.58	3600.00	6061.44	12.34	3600.00	**6043.13**	0.23	**6043.13**	0.11	**6043.13**	0.38(0.52)	**6043.13**	0.30(0.41)
M2	100	30	69	51	**8471.25**	0.00	2763.55	**8471.25**	0.00	1358.34	**8471.25**	0.27	**8471.25**	0.12	**8471.25**	0.39(0.53)	**8471.25**	0.39(0.53)
M3	100	40	59	29	**3836.03**	0.00	3111.47	**3836.03**	0.00	2589.89	**3836.03**	0.28	**3836.03**	0.10	**3836.03**	0.23(0.31)	**3836.03**	0.22(0.30)
M4	100	40	59	44	6424.29	33.82	3600.00	6424.29	30.47	3600.00	**6288.73**	0.33	**6288.73**	0.13	**6288.73**	0.30(0.41)	**6288.73**	0.28(0.38)
M5	100	40	59	59	**8789.24**	13.04	3600.00	**8789.24**	0.00	1310.11	**8789.24**	0.36	**8789.24**	0.28	**8789.24**	0.35(0.48)	**8789.24**	0.34(0.46)
M6	100	50	49	24	**3148.77**	0.00	3285.84	**3148.77**	10.95	3600.00	**3148.77**	0.21	**3148.77**	0.12	**3148.77**	0.22(0.30)	**3148.77**	0.22(0.30)
M7	100	50	49	36	5783.34	45.20	3600.08	6056.26	46.18	3600.00	**5777.62**	0.38	**5777.62**	0.16	5779.58	0.30(0.41)	5779.58	0.26(0.36)
M8	100	50	49	49	9114.16	35.46	3600.00	**9094.20**	26.20	3600.00	**9094.20**	0.40	**9094.20**	0.43	**9094.20**	0.38(0.52)	**9094.20**	0.34(0.46)
M9	100	30	69	34	**4362.92**	0.00	2479.83	**4362.92**	0.00	251.84	**4362.92**	0.18	**4362.92**	0.10	**4362.92**	0.34(0.46)	**4362.92**	0.31(0.42)
M10	100	30	69	51	**8105.04**	13.70	3600.00	**8105.04**	14.85	3600.00	**8105.04**	0.25	**8105.04**	0.14	**8105.04**	0.49(0.67)	**8105.04**	0.35(0.48)
M11	100	40	59	29	**3592.77**	10.99	3600.00	**3592.77**	0.00	736.28	**3592.77**	0.23	**3592.77**	0.11	**3592.77**	0.27(0.37)	**3592.77**	0.22(0.30)
M12	100	40	59	44	7007.12	38.64	3600.00	7326.59	33.70	3600.00	**6948.96**	0.33	**6948.96**	0.15	**6948.96**	0.35(0.48)	**6948.96**	0.32(0.44)
M13	100	50	49	24	**2929.98**	0.00	2763.84	**2929.98**	0.00	2554.08	**2929.98**	0.30	**2929.98**	0.10	**2929.98**	0.21(0.29)	**2929.98**	0.19(0.26)
M14	100	50	49	36	**5446.84**	38.22	3600.00	**5446.84**	33.41	3600.00	**5446.84**	0.32	**5446.84**	0.14	**5446.84**	0.25(0.34)	**5446.84**	0.26(0.36)
M15	100	30	69	34	5274.42	23.29	3600.00	**5164.91**	0.00	1298.50	**5164.91**	0.29	**5164.91**	0.11	**5164.91**	0.27(0.37)	**5164.91**	0.26(0.36)
M16	100	30	69	51	**7298.66**	12.33	3600.00	**7298.66**	3.59	3600.00	**7298.66**	0.30	**7298.66**	0.14	**7298.66**	0.39(0.53)	**7298.66**	0.33(0.45)
M17	100	40	59	29	4662.59	32.54	3600.00	**4580.82**	19.13	3600.00	**4580.82**	0.25	**4580.82**	0.13	**4580.82**	0.25(0.34)	**4580.82**	0.24(0.33)
M18	100	40	59	44	**6880.86**	35.94	3600.00	6917.87	34.47	3600.00	**6880.86**	0.35	**6880.86**	0.15	**6880.86**	0.36(0.49)	**6880.86**	0.31(0.42)
M19	100	50	49	24	**3749.47**	34.96	3600.00	3753.16	21.93	3600.00	**3749.47**	0.24	**3749.47**	0.11	**3749.47**	0.24(0.33)	**3749.47**	0.25(0.34)
M20	100	50	49	36	6549.37	50.35	3600.00	6170.11	43.45	3600.00	**5936.53**	0.40	**5936.53**	0.17	5950.85	0.28(0.38)	**5936.53**	0.29(0.40)
M21	100	50	49	49	**8506.35**	17.83	3600.68	8643.74	16.03	3600.00	**8506.35**	0.46	**8506.35**	0.44	**8506.35**	0.34(0.46)	**8506.35**	0.34(0.46)
M22	100	30	69	34	5842.87	8.83	3600.00	5842.87	21.07	3600.00	**5704.71**	0.27	**5704.71**	0.10	**5704.71**	0.33(0.45)	**5704.71**	0.29(0.40)
M23	100	30	69	51	**7513.86**	0.00	334.38	**7513.86**	0.00	53.09	**7513.86**	0.26	**7513.86**	0.13	**7513.86**	0.36(0.49)	**7513.86**	0.36(0.49)
M24	100	40	59	29	**5029.98**	33.38	3600.00	**5029.98**	36.85	3600.00	5072.39	0.34	**5029.98**	0.11	**5029.98**	0.30(0.41)	**5029.98**	0.26(0.36)
M25	100	40	59	44	**6456.74**	20.15	3600.00	**6456.74**	4.61	3600.00	**6456.74**	0.34	**6456.74**	0.13	**6456.74**	0.43(0.59)	**6456.74**	0.33(0.45)
M26	100	50	49	24	4420.68	42.92	3600.00	4490.01	56.73	3600.00	**4280.76**	0.25	**4280.76**	0.11	**4280.76**	0.29(0.40)	**4280.76**	0.26(0.36)
M27	100	50	49	36	**5833.35**	41.01	3600.00	5976.48	42.91	3600.00	**5833.35**	0.35	**5833.35**	0.14	**5833.35**	0.34(0.46)	**5833.35**	0.24(0.33)
M28	100	50	49	49	**8143.74**	25.06	3600.00	**8143.74**	13.07	3600.00	**8143.74**	0.46	**8143.74**	0.42	**8143.74**	0.31(0.42)	**8143.74**	0.31(0.42)
M29	100	30	69	34	**4750.53**	0.00	1815.30	**4750.53**	0.00	82.46	**4750.53**	0.21	**4750.53**	0.09	**4750.53**	0.28(0.38)	**4750.53**	0.26(0.36)
M30	100	30	69	51	**7784.01**	14.81	3600.00	**7784.01**	0.00	835.74	**7784.01**	0.26	**7784.01**	0.12	**7784.01**	0.36(0.49)	**7784.01**	0.33(0.45)
M31	100	40	59	29	**3396.99**	6.23	3600.00	**3396.99**	0.00	171.93	**3396.99**	0.29	**3396.99**	0.12	**3396.99**	0.23(0.31)	**3396.99**	0.22(0.30)
M32	100	40	59	44	**6584.91**	33.59	3600.00	**6584.91**	24.36	3600.00	**6584.91**	0.37	**6584.91**	0.17	**6584.91**	0.34(0.46)	**6584.91**	0.31(0.42)
M33	100	40	59	59	**9949.88**	15.22	3600.00	**9949.88**	0.00	899.30	**9949.88**	0.32	**9949.88**	0.34	**9949.88**	0.42(0.57)	**9949.88**	0.38(0.52)
M34	100	50	49	24	**3066.63**	20.66	3600.00	**3066.63**	14.26	3600.00	**3066.63**	0.29	**3066.63**	0.10	**3066.63**	0.20(0.27)	**3066.63**	0.20(0.27)
M35	100	50	49	36	5750.13	44.72	3600.00	5750.13	46.67	3600.00	**5657.02**	0.41	**5657.02**	0.13	**5657.02**	0.25(0.34)	**5657.02**	0.24(0.33)
M36	100	50	49	49	**8009.42**	11.93	3600.00	**8009.42**	11.73	3600.00	**8009.42**	0.47	**8009.42**	0.51	**8009.42**	0.37(0.51)	**8009.42**	0.33(0.45)
M37	100	30	69	34	**1726.09**	0.00	1251.83	**1726.09**	0.00	493.80	**1726.09**	0.22	**1726.09**	0.09	**1726.09**	0.25(0.34)	**1726.09**	0.25(0.34)
M38	100	30	69	51	**2501.29**	0.00	104.81	**2501.29**	0.00	44.00	**2501.29**	0.22	**2501.29**	0.13	**2501.29**	0.34(0.46)	**2501.29**	0.32(0.44)
M39	100	40	59	29	**1518.85**	31.55	3600.00	1540.25	35.57	3600.00	**1518.85**	0.28	**1518.85**	0.11	**1518.85**	0.23(0.31)	**1518.85**	0.22(0.30)
M40	100	40	59	44	**2107.69**	20.41	3600.00	**2107.69**	26.42	3600.00	**2107.69**	0.34	**2107.69**	0.15	**2107.69**	0.31(0.42)	**2107.69**	0.32(0.44)
M41	100	50	49	24	1570.85	41.36	3600.00	1632.52	54.75	3600.00	**1499.04**	0.23	**1499.04**	0.11	**1499.04**	0.24(0.33)	**1499.04**	0.21(0.29)
M42	100	50	49	36	2238.72	41.59	3600.00	2558.96	50.54	3600.00	**2208.52**	0.31	**2208.52**	0.16	**2208.52**	0.32(0.44)	**2208.52**	0.26(0.36)
M43	100	50	49	49	**3228.85**	7.83	3600.00	**3228.85**	3.61	3600.00	**3228.85**	0.30	**3228.85**	0.61	**3228.85**	0.37(0.51)	**3228.85**	0.37(0.51)
M44	150	45	104	52	6800.55	36.43	3600.00	6611.13	30.72	3600.00	**6557.02**	0.33	**6557.02**	0.15	**6557.02**	0.44(0.60)	**6557.02**	0.35(0.48)
M45	150	45	104	78	9558.62	29.37	3600.00	9494.90	23.62	3600.00	**9489.22**	0.43	9503.10	0.24	**9489.22**	0.55(0.75)	**9489.22**	0.47(0.64)
M46	150	60	89	44	5941.15	54.81	3600.00	5983.58	55.35	3600.00	**5393.39**	0.36	**5393.39**	0.17	**5393.39**	0.32(0.44)	**5393.39**	0.30(0.41)
M47	150	60	89	66	8557.71	52.88	3600.00	8171.50	48.29	3600.00	**8149.13**	0.64	**8149.13**	0.26	8171.50	0.44(0.60)	8171.50	0.41(0.56)
M48	150	75	74	37	4950.34	57.27	3600.00	**4583.03**	56.53	3600.00	**4583.03**	0.35	**4583.03**	0.19	**4583.03**	0.33(0.45)	**4583.03**	0.30(0.41)
M49	150	75	74	55	7388.02	59.08	3600.00	8329.72	66.61	3600.00	**6841.86**	0.57	**6841.86**	0.23	6903.24	0.40(0.55)	**6841.86**	0.37(0.51)
M50	150	75	74	74	11 127.67	50.08	3600.00	10 067.26	35.43	3600.00	**10 061.50**	0.70	10 170.60	1.41	**10 061.50**	0.56(0.76)	**10 061.50**	0.52(0.71)

(*continued on next page*)

Table 3 (*continued*).

Name	$\lvert V\rvert$	$\lvert F\rvert$	$\lvert C\rvert$	D	Flow-based formulation			Node-based formulation			MA		VNS		ABC(FDP)		ABC(VDP)	
					Obj.	Gap(%)	Time	Obj.	Gap(%)	Time	Average obj.	Time	Average obj.	Time	Average obj.	Time	Average obj.	Time
M51	150	45	104	52	6152.91	35.80	3600.00	6377.95	30.71	3600.00	**6133.27**	0.37	**6133.27**	0.19	6152.91	0.39(0.53)	**6133.27**	0.33(0.45)
M52	150	45	104	78	**8629.85**	25.70	3600.00	8730.54	16.41	3600.00	**8629.85**	0.45	**8629.85**	0.22	**8629.85**	0.50(0.68)	**8629.85**	0.48(0.66)
M53	150	60	89	44	5200.64	56.50	3600.00	5305.46	58.71	3600.00	**5199.33**	0.39	**5199.33**	0.16	**5199.33**	0.32(0.44)	**5199.33**	0.28(0.38)
M54	150	60	89	66	8712.97	55.90	3600.00	7624.55	48.94	3600.00	7471.30	0.64	**7471.14**	0.24	7572.55	0.37(0.51)	**7471.14**	0.38(0.52)
M55	150	75	74	37	5218.44	62.19	3600.00	7246.78	83.33	3600.00	**4898.98**	0.38	**4898.98**	0.18	**4898.98**	0.28(0.38)	**4898.98**	0.27(0.37)
M56	150	75	74	55	7573.78	60.09	3600.00	7959.61	73.71	3600.00	**7169.79**	0.57	**7169.79**	0.27	7245.55	0.50(0.68)	**7169.79**	0.35(0.48)
M57	200	60	139	69	6286.19	47.90	3600.00	6356.29	44.03	3600.00	6281.59	0.61	**6281.11**	0.27	6291.84	0.62(0.85)	**6281.11**	0.49(0.67)
M58	200	60	139	104	**9476.31**	30.88	3600.00	**9476.31**	20.52	3600.00	**9476.31**	0.64	9499.65	0.37	**9476.31**	1.34(1.83)	**9476.31**	0.75(1.02)
M59	200	80	119	59	6162.23	60.83	3600.00	6199.21	71.84	3600.00	**5620.09**	0.66	**5620.09**	0.26	5702.01	0.55(0.75)	**5620.09**	0.39(0.53)
M60	200	80	119	89	9241.57	63.05	3600.00	8827.56	57.22	3600.00	**8464.17**	0.91	**8464.17**	0.41	8547.86	0.74(1.01)	**8464.17**	0.56(0.76)
M61	200	100	99	49	5439.22	64.73	3600.00	5549.83	73.90	3600.00	**4895.78**	0.57	**4895.78**	0.28	4907.93	0.37(0.51)	4907.93	0.35(0.48)
M62	200	100	99	74	8388.30	66.49	3600.00	8346.32	71.59	3600.00	**7366.94**	1.01	7383.64	0.44	7446.27	0.46(0.63)	7379.09	0.43(0.59)
M63	200	60	139	69	6562.64	45.87	3600.00	6814.37	36.87	3600.00	**6257.61**	0.49	6471.74	0.25	**6257.61**	0.69(0.94)	**6257.61**	0.43(0.59)
M64	200	60	139	104	9626.72	34.50	3600.00	9642.84	19.73	3600.00	9588.34	0.80	9579.08	0.39	**9550.47**	0.64(0.87)	**9550.47**	0.63(0.86)
M65	200	80	119	59	6307.84	59.72	3600.00	6005.98	56.01	3600.00	**5442.88**	0.63	**5442.88**	0.24	**5442.88**	0.50(0.68)	**5442.88**	0.36(0.49)
M66	200	80	119	89	8854.88	54.51	3600.00	9175.32	51.43	3600.00	8368.61	0.84	**8359.44**	0.40	8521.95	0.64(0.87)	**8359.44**	0.58(0.79)
M67	200	100	99	49	5226.24	61.70	3600.00	5342.40	67.10	3600.00	4887.60	0.57	**4872.50**	0.23	4988.82	0.37(0.51)	**4872.50**	0.35(0.48)
M68	200	100	99	74	9338.61	64.42	3600.00	8593.37	63.76	3600.00	**7217.00**	0.79	**7217.00**	0.35	7514.69	0.57(0.78)	**7217.00**	0.48(0.66)
M69	200	100	99	99	11 966.08	44.77	3600.00	11 088.01	38.97	3600.00	**10 754.88**	1.03	10 841.55	4.32	**10 754.88**	0.69(0.94)	**10 754.88**	0.62(0.85)
No. Best					33			30			63		62		55		**65**	
Average					6262.20	32.59	3390.05	6262.62	30.31	3053.34	6045.62	0.42	6051.23	0.28	6060.63	0.39(0.53)	**6044.80**	0.34(0.46)

Table 4
Comparison of various approaches on large size instances.

Name	30 s[a]							
	Best Obj.				Average Obj.			
	MA	VNS	ABC(FDP)	ABC(VDP)	MA	VNS	ABC(FDP)	ABC(VDP)
L1	**260.92**	261.32	**260.92**	**260.92**	265.66	262.58	261.24	**261.00**
L2	568.82	569.46	**567.95**	568.01	571.59	592.01	568.90	**568.29**
L3	221.78	**219.01**	219.35	**219.01**	224.43	222.36	221.47	**220.14**
L4	537.71	521.77	521.19	**519.32**	540.90	534.94	523.57	**521.65**
L5	174.33	175.90	173.49	**173.47**	177.27	178.07	173.55	**173.53**
L6	409.69	421.19	**409.60**	410.41	413.59	432.73	**410.98**	411.23
L7	**9659.42**	9860.09	9662.19	9702.00	9783.33	10 262.22	9742.61	**9719.99**
L8	7445.34	7428.56	7429.76	**7425.21**	7475.27	7444.14	7434.23	**7432.41**
L9	3 406 315.00	3 466 420.00	3 319 038.00	**3 310 439.50**	3 451 099.00	3 690 228.25	3 331 750.75	**3 322 650.50**
L10	5 809 963.00	5 956 696.00	**5 748 409.50**	5 763 888.50	5 882 573.50	6 204 090.50	5 815 024.50	**5 771 842.50**
L11	2 715 609.50	**2 698 974.25**	2 706 386.00	2 706 040.00	2 848 941.50	2 830 881.25	2 733 059.25	**2 717 308.25**
L12	**4 852 149.00**	4 969 874.00	4 859 753.00	4 870 647.50	4 966 469.00	5 354 164.50	4 961 075.50	**4 908 488.00**
L13	2 466 742.00	**2 402 721.75**	2 514 447.00	2 467 296.00	2 540 296.75	2 623 601.25	2 537 567.75	**2 491 357.00**
L14	**4 391 044.00**	4 485 000.50	4 401 222.00	4 397 055.00	4 425 225.00	4 729 981.50	4 443 971.00	**4 420 655.50**
No. Best	4	3	4	**6**	0	0	1	**13**
Average	1 690 078.50	1 714 224.62	1 683 464.25	**1 681 046.00**	1 723 861.25	1 818 062.62	1 702 984.62	**1 689 400.62**

Name	60 s[a]							
	Best Obj.				Average Obj.			
	MA	VNS	ABC(FDP)	ABC(VDP)	MA	VNS	ABC(FDP)	ABC(VDP)
L1	**260.92**	261.32	261.32	**260.92**	265.66	262.58	261.50	**261.03**
L2	568.82	569.32	**567.95**	**567.95**	571.59	591.25	568.47	**568.27**
L3	221.78	**219.01**	219.45	**219.01**	224.43	221.86	221.15	**220.11**
L4	521.77	521.62	521.19	**519.36**	536.31	534.36	522.58	**520.62**
L5	174.33	175.90	**173.47**	**173.47**	177.27	178.07	**173.49**	173.52
L6	409.69	416.06	**409.60**	**409.60**	413.59	426.54	409.88	**409.60**
L7	**9659.42**	9860.09	9666.44	9679.05	9775.89	10 236.86	9705.81	**9698.64**
L8	7445.34	7428.56	7428.24	**7423.48**	7475.27	7434.01	7429.89	**7426.71**
L9	3 363 319.00	3 462 163.00	3 310 082.00	**3 304 538.50**	3 433 883.25	3 688 099.50	3 326 213.00	**3 316 275.25**
L10	5 801 249.00	5 944 205.50	**5 685 087.00**	5 722 106.50	5 861 252.00	6 195 724.00	5 750 109.50	**5 742 636.50**
L11	2 715 609.50	**2 698 974.25**	2 699 517.50	**2 698 974.25**	2 841 862.00	2 825 327.00	2 723 300.00	**2 704 331.25**
L12	**4 788 332.00**	4 954 128.00	4 865 624.50	4 818 295.50	4 925 313.50	5 348 881.50	4 964 871.00	**4 837 634.00**
L13	2 466 742.00	**2 402 721.75**	2 481 414.50	2 405 866.50	2 531 566.75	2 592 871.00	2 510 129.25	**2 437 261.00**
L14	**4 361 460.00**	4 481 123.50	4 378 647.00	4 391 208.00	**4 404 383.50**	4 652 509.00	4 407 189.00	4 408 860.50
No. Best	4	3	4	**9**	1	0	1	**12**
Average	1 679 712.38	1 711 626.25	1 674 258.50	**1 668 588.75**	1 715 550.12	1 808 807.00	1 692 936.00	**1 676 162.62**

Name	120 s[a]							
	Best Obj.				Average Obj.			
	MA	VNS	ABC(FDP)	ABC(VDP)	MA	VNS	ABC(FDP)	ABC(VDP)
L1	**260.92**	261.32	**260.92**	**260.92**	265.03	262.58	261.24	**261.03**
L2	568.82	569.32	**567.95**	**567.95**	570.56	585.43	568.92	**568.06**
L3	221.78	**219.01**	**219.01**	**219.01**	224.43	221.86	220.18	**220.13**
L4	521.77	521.62	519.62	**519.33**	536.01	531.29	521.22	**520.73**
L5	174.33	175.90	**173.47**	**173.47**	177.27	178.07	**173.48**	173.54
L6	409.69	416.06	**409.60**	**409.60**	413.59	425.91	410.51	**410.15**
L7	**9659.42**	9860.09	9661.50	**9659.42**	9746.22	10 222.04	9691.38	**9678.74**
L8	7445.34	**7410.18**	7417.76	7423.06	7474.78	7430.33	**7424.07**	7424.93
L9	3 359 436.25	3 420 977.50	3 304 241.50	**3 293 890.50**	3 428 711.25	3 669 406.25	3 316 500.75	**3 301 858.50**
L10	5 797 975.00	5 940 558.50	**5 673 063.50**	5 687 455.00	5 851 822.50	6 169 100.50	5 703 747.50	**5 696 322.50**
L11	2 715 609.50	2 698 974.25	**2 698 278.00**	**2 698 278.00**	2 840 534.25	2 776 202.50	2 700 609.00	**2 699 507.00**
L12	**4 766 605.50**	4 954 128.00	4 818 353.00	4 802 720.00	4 910 558.50	5 330 731.00	4 843 818.00	**4 810 786.50**
L13	2 459 972.75	**2 399 977.75**	2 431 477.50	2 423 088.00	2 525 418.25	2 592 322.25	2 479 552.50	**2 434 717.25**
L14	4 338 690.50	4 479 302.50	**4 327 844.50**	4 339 843.00	4 391 780.00	4 643 592.50	**4 361 635.00**	4 366 767.00
No. Best	3	3	8	**9**	0	0	3	**11**
Average	1 675 539.38	1 708 096.50	1 662 320.50	**1 661 750.50**	1 712 016.62	1 800 086.62	1 673 223.88	**1 666 372.62**

[a]Time limit for MA/VNS. Time limit for ABC was $\frac{2.27}{3.10}$ times the time limit for MA/VNS.

five times independently like MA and VNS approaches of Shaelaie et al. [1]. On small and medium instances, our approaches terminates when best solution fails to improve over 500 iterations. The approaches of Shaelaie et al. [1] were executed on a PC with a 2.27 GHz Intel Core-i5 processor with 4 GB RAM. As these approaches were executed on a system which is different from the system used to execute ABC(VDP) and ABC(FDP) approaches, the running times of these approaches be compared precisely with ABC(VDP) and ABC(FDP). However, a rough comparison is possible. As both systems use Core-i5 processors, execution times of ABC(VDP) and ABC(FDP) can be adjusted by multiplying by a factor of $\frac{3.10}{2.27}$ for a rough comparison with approaches of Shaelaie et al. [1].

4.1. Comparison of our approaches with approaches of [1]

Tables 2–4 compare the performance of our approaches ABC (FDP) and ABC(VDP) with different approaches proposed in [1]. Tables 2–4 reports the results for small, medium, and large size instances respectively. In these tables, the results of Flow-based formulation, Node-based formulation, MA and VNS have been taken from Shaelaie et al. [1]. For each of these three tables, the best results over all the approaches are shown in bold font for easy identification. In 2 and 3, the first column represents the name of the instance, whereas the next four columns represents the number of vertices ($|V|$), number of facilities ($|F|$), number of customers ($|C|$) and the given demand (D). For both the formulations, viz. Flow-based and Node-based formulations, the columns 'Obj.', 'Gap(%)' and 'Time' reports the objective function value, estimated gap from the optimality (if any) and execution times required by CPLEX to solve the respective formulations. In the columns of MA, VNS, ABC(FDP) and ABC(VDP) approaches, the 'Average Obj.' and 'Time' reports the average objective function value and average execution time over five different runs. The value in parenthesis under column 'Time' for ABC(FDP) and ABC(VDP) reports the adjusted execution times of our approaches as explained in the beginning of this section. The last two rows in both Tables 2 and 3, 'No. Best' reports the number of best solutions achieved by each algorithm, where as 'Average' reports the average values of each column. On small instances, all the approaches except for node based formulation performed the same in terms of solution quality on all the instances. Node based formulation performed slightly worse in terms of solution quality. Both ABC(FDP) and ABC(VDP) approaches got best known results on all 32 instances similar to MA and VNS. In comparison to MA and VNS approaches, both ABC(FDP) and ABC(VDP) are clearly slower on small instances. However, they are faster than the two exact approaches. For medium instances, exact approaches are too slow and they fail to find the proven optimal solution on most instances in maximum allowed execution time of 3600 s. Their solution quality is also quite inferior in comparison to four metaheuristic approaches, viz. MA, VNS, ABC(FDP) and ABC(VDP). As far as comparison among four metaheuristic approaches is concerned, ABC(VDP) performed the best, followed by MA, VNS and ABC(FDP). ABC(VDP) obtained the best known solution values for 65 instances. In comparison, MA and VNS obtained best known solution values for 63 and 62 instances respectively. ABC(FDP) performed worse by obtaining the best known solution values for 55 instances only. ABC(VDP) obtained better overall average solution quality than other approaches. Both ABC(FDP) and ABC(VDP) are slower than MA and VNS on majority of instances. However, on some instances they are clearly faster.

On 14 large instances, Shaelaie et al. [1] reported the results of MA and VNS after executing them for a fixed amount of time. They have reported three sets of results with time limit of 30, 60 and 120 s. For each instance, best and average solution obtained by MA and VNS over five different runs were reported under each of the three time limits. Results of exact approaches were not reported on these large instances presumably because they were too slow to be of any use on these large instances. To compare our approaches with the previous approaches (MA and VNS) on these large instances, we have also used the CPU time as termination criteria like Shaelaie et al. [1]. However, we have adjusted our time limits to $30 \times \frac{2.27}{3.10}$, $60 \times \frac{2.27}{3.10}$ and $120 \times \frac{2.27}{3.10}$ to compensate for difference in processing speeds of the system used to execute our approaches ABC(FDP), ABC(VDP) and the system used to execute MA and VNS approaches. The comparison of the results according to three different termination criteria based on time limits have been reported in Table 4. The columns 'Best Obj.' and 'Average Obj.' represents the best and average objective function values obtained by respective approaches over five different runs. The last two rows in this Table 4, 'No. Best' reports the number of best solutions achieved by each algorithm, whereas 'Average' reports the average value of corresponding column. The results presented in Table 4 clearly demonstrates the superiority of our approaches ABC(FDP) and ABC(VDP) on large size instances. Our approach ABC(VDP) has obtained more number of best known values and better overall average solution quality than MA, VNS & ABC(FDP) on both quality parameters (Best Obj. and Average Obj.).

Tables 5 and 6 presents a summary of comparison of ABC(FDP) and ABC(VDP) respectively with Flow-based formulation, Node-based formulation, MA and VNS approaches in terms of number of instances on which ABC(FDP)/ABC(VDP) obtained better ('$<$'), same ('$=$') or worse solutions ('$>$'). On small and medium instances this comparison is done in terms of average solution quality, and on large instances in terms of both best and average solution quality under each of the three time limits. Actually, Shaelaie et al. [1] reported only average solution quality for small and medium instances, whereas for large instances both best and average solution quality were reported. Also they have not reported the results of Flow-based formulation and Node-based formulation for the large instances. These tables show that relative performance of our approaches ABC(FDP) and ABC(VDP) improves with instance size and with increase in time limit in comparison to MA and VNS approaches. Further, on large instances both ABC(FDP) and ABC(VDP) performed consistently better in terms of average solution quality, which shows the robustness of our approaches. Further, benefits of using variable degree of perturbation can be clearly seen in Tables 2–4 and 6.

Fig. 2 plots the solutions obtained by our approaches on four different instances, viz S2, S3, S7 and S8 where the depot, facilities and customer vertices are shown as black triangle, red squares and blue circles respectively. All these four instances are optimally solved by our approaches. This can be inferred from Table 2 as both ABC(FDP) and ABC(VDP) obtained same solutions as obtained by exact approaches with optimality gap of zero on these four instances. These plots show how the facilities and customers are distributed and how the customers are covered by facilities in these instances. These plots also depict the composition of an optimal solution for each of these instances. Instances S2 and S3 have 15 facilities and 35 customers, whereas instances S7 and S8 have 25 facilities and 25 customers. The instances S2 and S7 demand 75% of customers to be covered, whereas the instances S3 and S8 demand 100% of customers to be covered. From these plots, it can be clearly seen that tour length increases when there is a higher percentage of demand (i.e., more number of customers need to be covered) even though total number of facilities and total number of customers remain the same.

Table 5
Summary of results comparing ABC(FDP) with other approaches.

ABC(FDP) vs.	Flow-based formulation			Node-based formulation			MA			VNS		
Types of instances	$<$	$=$	$>$	$<$	$=$	$>$	$<$	$=$	$>$	$<$	$=$	$>$
Small	0	32	0	1	31	0	0	32	0	0	32	0
Medium	34	34	1	38	31	0	2	52	15	6	48	15
Large (30 s)-Best	–	–	–	–	–	–	9	1	4	10	0	4
Large (30 s)-Avg	–	–	–	–	–	–	13	0	1	14	0	0
Large (60 s)-Best	–	–	–	–	–	–	9	0	5	10	1	3
Large (60 s)-Avg	–	–	–	–	–	–	12	0	2	14	0	0
Large (120 s)-Best	–	–	–	–	–	–	11	1	2	11	1	2
Large (120 s)-Avg	–	–	–	–	–	–	14	0	0	14	0	0
Total	34	66	1	39	62	0	70	86	29	79	82	24

Table 6
Summary of results comparing ABC(VDP) with other approaches.

ABC(VDP) vs.	Flow-based formulation			Node-based formulation			MA			VNS			ABC(FDP)		
Types of instances	$<$	$=$	$>$	$<$	$=$	$>$	$<$	$=$	$>$	$<$	$=$	$>$	$<$	$=$	$>$
Small	0	32	0	1	31	0	0	32	0	0	32	0	0	32	0
Medium	36	33	0	38	31	0	6	59	4	7	59	3	12	57	0
Large (30 s)-Best	–	–	–	–	–	–	8	1	5	11	1	2	8	1	5
Large (30 s)-Avg	–	–	–	–	–	–	14	0	0	14	0	0	13	0	1
Large (60 s)-Best	–	–	–	–	–	–	10	1	3	11	2	1	8	3	3
Large (60 s)-Avg	–	–	–	–	–	–	13	0	1	14	0	0	12	0	2
Large (120 s)-Best	–	–	–	–	–	–	10	2	2	11	1	2	5	6	3
Large (120 s)-Avg	–	–	–	–	–	–	14	0	0	14	0	0	11	0	3
Total	36	65	0	39	62	0	75	95	15	82	95	8	69	99	17

Table 7
Results of Wilcoxon signed rank test between various approaches.

	ABC(FDP) vs.							ABC(VDP) vs.					
	$N/Total$	W^+	W^-	Z	Z_{Cri}	Significant		$N/Total$	W^+	W^-	Z	Z_{Cri}	Significant
FBF	35/101	627	3	−5.110	−1.96	Yes		36/101	666	0	−5.232	−1.96	Yes
NBF	39/101	780	0	−5.442	−1.96	Yes		39/101	780	0	−5.442	−1.96	Yes
MA	59/143	1198	572	−2.363	−1.96	Yes		52/143	1273	105	−5.318	−1.96	Yes
VNS	63/143	1583	433	−3.937	−1.96	Yes		52/143	1332	46	−5.856	−1.96	Yes
							ABC(FDP)	54/143	1379	106	−5.480	−1.96	Yes

4.2. Wilcoxon signed rank test

To rule out the possibility of attributing the better performance of our approaches to random fluctuations, Wilcoxon signed rank test [26] has been used. The calculator available online[1] is used to perform the two tailed Wilcoxon signed rank test with significance criteria set to 5% (i.e. p-value ≤ 0.05). In this test, the difference between the normalized values of *'Average Obj.'* obtained by ABC(FDP)/ABC(VDP) and the compared approach is ranked. Table 7 summarizes the results of this test. In Table 7, the '$N/Total$' denotes the number of instances without tie out of the total number of instances compared. The W^+ denotes the sum of ranks for the instances where ABC(FDP)/ABC(VDP) performs better than its competitor, whereas the W^- denotes the sum of ranks for the instances where ABC(FDP)/ABC(VDP) performed worse than its competitor. As the number of instances are more than thirty ($N > 30$), the test statistic Z is used to compare with the critical value Z_{Cri} according to Wilcoxon signed rank test [26]. If $Z \leq Z_{Cri}$, then there is a significant difference between the performance of the two compared approaches, otherwise the difference is not significant. Table 7 clearly shows that the better results of ABC(FDP) and ABC(VDP) are significant with respect to four other algorithms, viz. Flow-based formulation, Node-based formulation, MA and VNS with $Z \leq Z_{Cri}$ and $W^+ > W^-$. Comparison between our approaches shows that the better results of ABC(VDP) are significant with respect to ABC(FDP).

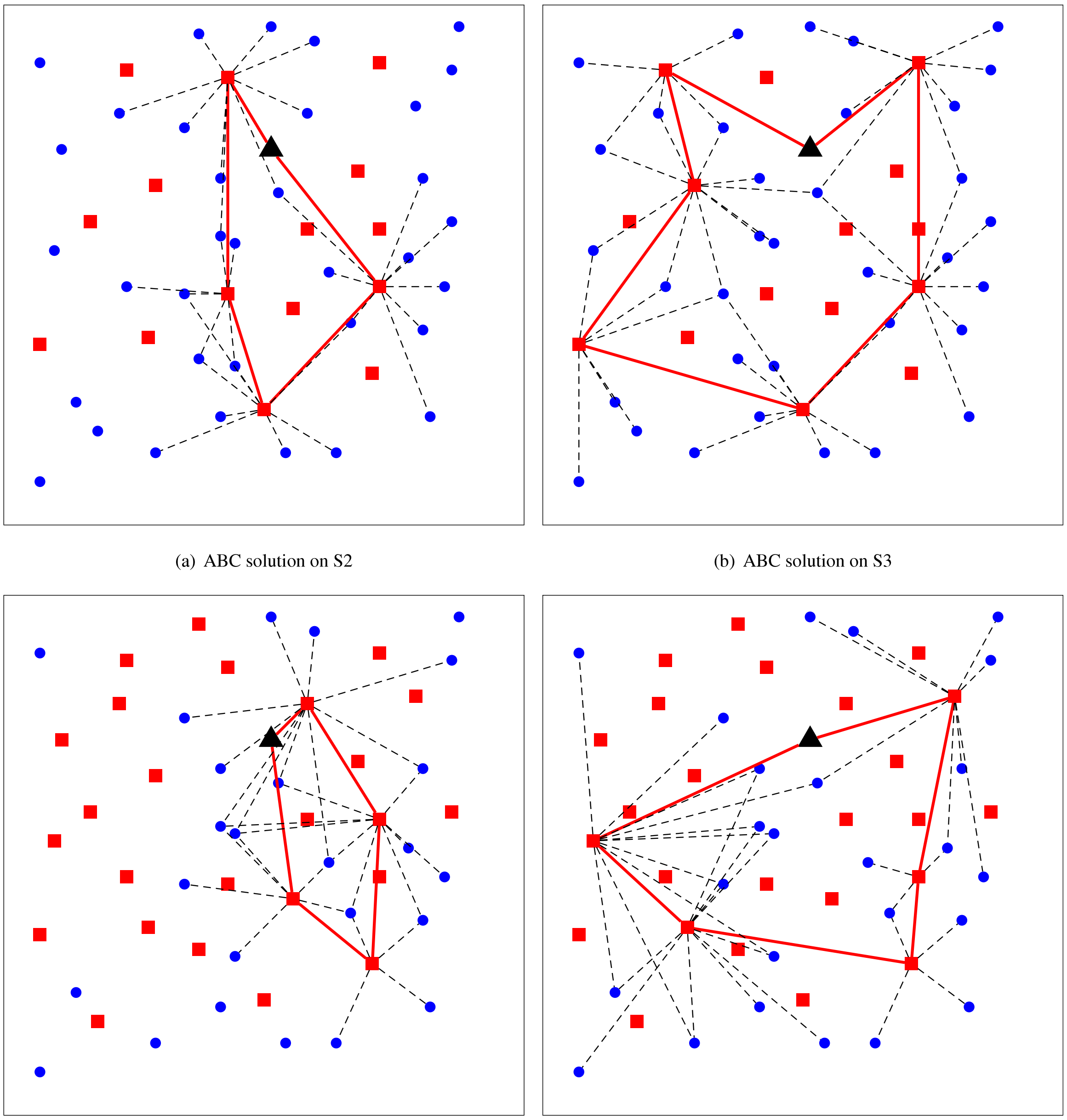
4.3. Convergence behavior of our approaches

To illustrate the convergence behavior of our approaches, four instances are considered namely L3, L4, L12 and L13. Fig. 3 plots the convergence behavior of our approaches ABC(FDP) and ABC(VDP) on these instances. The convergence graphs depicts that the ABC(VDP) convergences faster than ABC(FDP) again establishing the superiority of ABC(VDP) over ABC(FDP).

5. Conclusions

In this paper, we have proposed a swarm intelligence based approach, viz. an artificial bee colony algorithm for the generalized covering traveling salesman problem where we have

1 https://mathcracker.com/wilcoxon-signed-ranks.php.



(a) ABC solution on S2

(b) ABC solution on S3

(c) ABC solution on S7

(d) ABC solution on S8

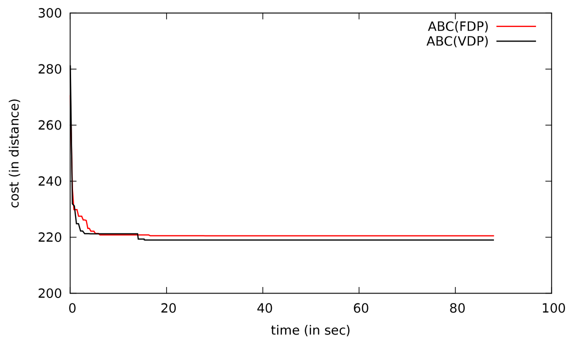
Fig. 2. Solutions obtained by ABC on four different instances.

incorporated a variable degree of perturbation strategy and redundant facility removal in neighboring solution generation. We have evaluated and compared our proposed approach with the state-of-the-art approaches on the benchmark instances available in the literature. Computational results on these benchmark instances show the effectiveness of our approach over all the other state-of-the-art approaches in terms of solution quality.

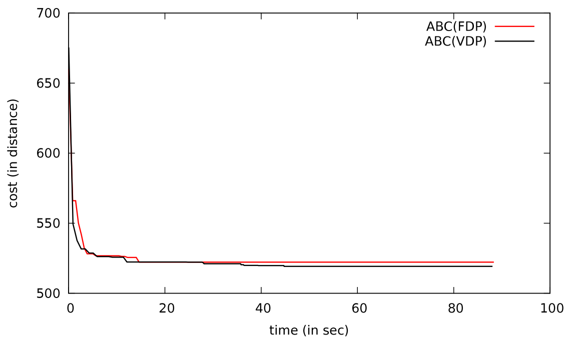
A possible future work is to extend our approach to related variants of traveling salesman problem where the concept of coverage is used. As our approaches are designed keeping in mind the subset selection and permutation characteristics of the GCTSP, similar approaches can be developed for other problems having similar characteristics, e.g., single machine order acceptance and scheduling. The concept of varying the degree of perturbation over the number of iterations can be adopted for any problem where higher degree of perturbation during later iterations is turning out to be quite disruptive, thereby preventing the search process from reaching the optimal solution.

Acknowledgments

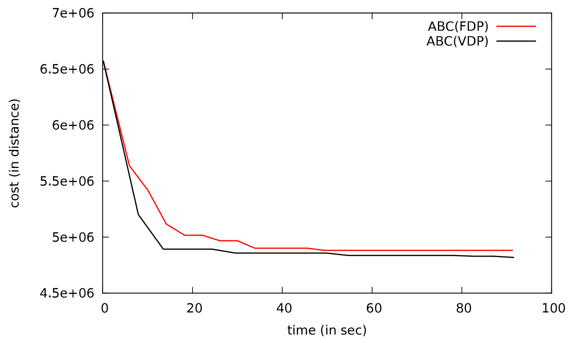
Authors are grateful to Dr. Majid Salari and Mr. Mohammad H. Shaelaie for providing us the benchmark instances and answering our queries regarding their approaches presented in [1]. Authors would like to thank three anonymous reviewers for their valuable comments and suggestions which helped in improving the quality of this manuscript. The first author acknowledges the financial support received from the Council of Scientific & Industrial Research (CSIR), Government of India in the form of a Senior Research Fellowship. The second author acknowledges



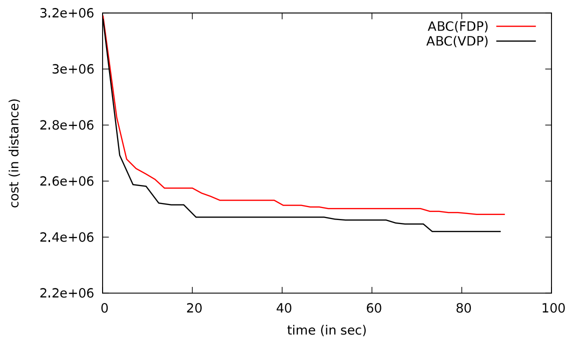
(a) L3 instance



(b) L4 instance



(c) L12 instance



(c) L13 instance

Fig. 3. Convergence behavior of our approaches on different instances.

the support of the research grant no. MTR/2017/000391 received under MATRICS scheme from Science and Engineering Research Board (SERB), Government of India.

References

[1] M.H. Shaelaie, M. Salari, Z. Naji-Azimi, The generalized covering traveling salesman problem, Appl. Soft Comput. 24 (2014) 867–878.
[2] B. Awerbuch, Y. Azar, A. Blum, S. Vempala, New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen, SIAM J. Comput. 28 (1) (1998) 254–262.
[3] A. Blum, R. Ravi, S. Vempala, A constant-factor approximation algorithm for the k mst problem, in: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, ACM, 1996, pp. 442–448.
[4] N. Garg, A 3-approximation for the minimum tree spanning k vertices, in: Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on, IEEE, 1996, pp. 302–309.
[5] V.H. Nguyen, J.-F. Maurras, On the linear description of the k-cycle polytope, Int. Trans. Oper. Res. 8 (6) (2001) 673–692.
[6] E. Balas, The prize collecting traveling salesman problem, Networks 19 (6) (1989) 621–636.
[7] J.R. Current, D.A. Schilling, The covering salesman problem, Transp. Sci. 23 (3) (1989) 208–213.
[8] N. Altay, W.G. Green, OR/MS research in disaster operations management, European J. Oper. Res. 175 (1) (2006) 475–493.
[9] S.R. Shariff, N.H. Moin, M. Omar, Location allocation modeling for healthcare facility planning in malaysia, Comput. Ind. Eng. 62 (4) (2012) 1000–1010.
[10] D. Reina, S.T. Marin, N. Bessis, F. Barrero, E. Asimakopoulou, An evolutionary computation approach for optimizing connectivity in disaster response scenarios, Appl. Soft Comput. 13 (2) (2013) 833–845.
[11] M. Salari, M. Reihaneh, M.S. Sabbagh, Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem, Comput. Ind. Eng. 83 (2015) 244–251.
[12] D. Karaboga, An Idea based on Honey bee Swarm for Numerical Optimization, Technical Report TR06, Computer Engineering Department, Erciyes University, Turkey, 2005.
[13] B. Basturk, D. Karaboga, An artificial bee colony (ABC) algorithm for numeric function optimization, in: Proceedings of the IEEE Swarm Intelligence Symposium, Indianapolis, In, USA, IEEE, 2006, pp. 12–14.
[14] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numeric function optimization: Artificial bee colony (ABC) algorithm, J. Global Optim. 39 (2007) 459–471.
[15] D. Karaboga, B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, in: Lecture Notes in Artificial Intelligence, Vol. 4529, Springer-Verlag, Berlin, 2007, pp. 789–798.
[16] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (2008) 687–697.
[17] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Appl. Soft Comput. 9 (2009) 625–631.
[18] S. Sundar, A. Singh, A swarm intelligence approach to the quadratic minimum spanning tree problem, Inform. Sci. 180 (2010) 3182–3191.
[19] D. Karaboga, B. Akay, A modified artificial bee colony (ABC) algorithm for constrained optimization problems, Appl. Soft Comput. 11 (2011) 3021–3031.
[20] Q.-K. Pan, M. Tasgetiren, P. Suganthan, T. Chua, A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, Inform. Sci. 181 (2011) 2455–2468.
[21] W. Gao, S. Liu, Improved artificial bee colony algorithm for global optimization, Inform. Process. Lett. 111 (2011) 871–882.
[22] V. Pandiri, A. Singh, Swarm intelligence approaches for multidepot salesmen problems with load balancing, Appl. Intell. 44 (4) (2016) 849.
[23] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inform. Sci. 192 (2012) 120–142.
[24] G. Candan, H.R. Yazgan, Genetic algorithm parameter optimisation using taguchi method for a flexible manufacturing system scheduling problem, Int. J. Prod. Res. 53 (3) (2015) 897–915.
[25] J. Li, X. Meng, M. Zhou, X. Dai, A two-stage approach to path planning and collision avoidance of multibridge machining systems, IEEE Trans. Syst. Man Cybern. Syst. (2016) 1–11.
[26] F. Wilcoxon, S. Katti, R.A. Wilcox, Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test, in: Selected Tables in Mathematical Statistics, Vol. 1, 1970, pp. 171–259.