Appl Intell (2018) 48:2247–2267
https://doi.org/10.1007/s10489-017-1075-0

Two new heuristics for the dominating tree problem

Kavita Singh[1] · **Shyam Sundar**[1]

Published online: 4 November 2017

Abstract Dominating Tree Problem (DTP) aims to find a dominating tree ($dTree$) of minimum cost on a given connected, undirected and weighted graph in such a way that a vertex in the graph is either in $dTree$ or adjacent to a vertex in $dTree$. A solution ($dTree$) to this problem can be used as routing backbone in wireless sensor network. Being a $\mathcal{NP}$-Hard problem, several problem-specific heuristics and metaheuristic techniques have been proposed. This paper presents two new heuristics for the DTP. First one is a new problem-specific heuristic that exploits the problem structure effectively, whereas the other is an artificial bee colony (ABC) algorithm. The proposed ABC for the DTP is different from the existing ABC algorithm for the DTP in the literature on its two main components: initial solution generation, and determining a neighboring solution. Computational results show on a set of standard benchmark instances that the proposed problem-specific heuristic and ABC algorithm for the DTP demonstrate the superiority over all existing problem-specific heuristics and metaheuristic techniques respectively in the literature.

Keywords Dominating tree · Wireless sensor networks · Problem-specific heuristic · Artificial bee colony algorithm · Swarm intelligence

✉ Shyam Sundar
ssundar.mca@nitrr.ac.in

Kavita Singh
ksingh.phd2015.mca@nitrr.ac.in

[1] Department of Computer Applications, National Institute of Technology Raipur, Raipur 492010, India

1 Introduction

The dominating tree problem (DTP) [11] is one of recently encountered combinatorial optimization problems in the field of wireless sensor networks (WSNs) due to its practical relevance in routing. The DTP is defined as follows: Let $G = (V, E, w)$ be an undirected, connected, and weighted graph, where V is the set of vertices (nodes), E is the set of edges, and for each edge $(u, v) \in E$, there exists a non-negative weight. The DTP deals with finding a dominating tree ($dTree$) of minimum cost on G in such a way that each vertex of G is either in $dTree$ or adjacent to a vertex in $dTree$. Vertices that are in $dTree$ are called dominating nodes, whereas vertices that are not in $dTree$ are called non-dominating nodes. Hereafter, vertex and node are used interchangeably in this paper.

Figure 1a presents a connected, undirected and edge-weighted graph G with 9 vertices and 13 edges, whereas Fig. 1b presents a $dTree$ of G whose dominating nodes ($<$ 2, 5, 6, 7 $>$) are shown in *dark grey* color, non-dominating vertices ($<$ 0, 1, 3, 4, 8 $>$) are shown in *light grey* color. Thick grey edges in Fig. 1b are part of $dTree$. The total edge-cost of this $dTree$ is 6.

The practical relevance of DTP lies in network routing as its solution (dominating tree) can be used as a routing backbone. Dominating nodes of a dominating tree of minimum cost ($dTree$ or solution) that consist of a subset of nodes of WSN (graph) can be used for storing routing information, as each non-dominating node is adjacent to at least one of the dominating nodes of $dTree$. Under this setup, the edge weight can be considered as energy consumption in sending a message along with that edge. In the process of message forwarding from *source* to *destination*, the message needs to be first forwarded to the nearest dominating node of sender, then this message is further routed to the

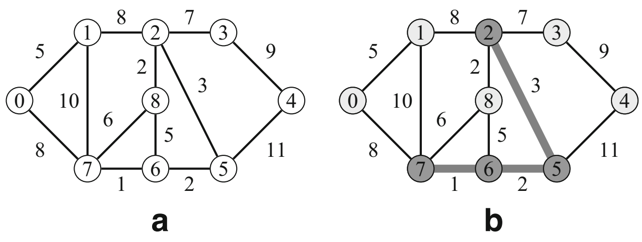

Fig. 1 A graph (G) and $dTree$ of (G)

nearest dominating node of the receiver with the help of $dTree$, and finally it is forwarded to the receiver. Each non-dominating node requires to only memorize the information of its nearest dominating node. The major advantage of this scheme [20] is that dominating nodes of $dTree$ (virtual backbone) that are usually smaller in number in comparison to the total number of nodes in WSNs can store such routing information, resulting overall a significant reduction in the size of routing table. Moreover, no recalculation of such routing table is required in case of occurrence of topological changes in the network, if such occurrence does not cause change in the set of dominating nodes of the network [20].

2 Literature survey

On the concept of connected dominating set, many approaches [6, 9, 17–19] for constructing a routing backbone with the objective of minimizing energy consumption in WSNs have been reported in the literature. However, all these approaches consider only *node-weight* rather than *edge-weight* for minimizing the energy consumption. All these papers discussed only the number of nodes obtained. In practice, the energy consumed at each edge directly effects the energy consumed in routing. This observation led to the introduction of DTP [11, 21]. They proved $\mathcal{NP}$-hardness of this problem together with the inapproximability and proposed an approximation algorithm with quasi-polynomial time complexity ($|V|^{O(lg|V|)}$) for the DTP. Further, they also proposed two polynomial time problem-specific heuristics (respectively referred to as Heu_DT1 and Heu_DT2 in this paper) for the DTP. The performance of their proposed heuristics were compared with a method based on minimum spanning tree without leaf edges, as the resultant tree obtained after this method is also a dominating tree. In addition, there are some work related to the tree cover problem [1, 4, 5] in the literature. However, the tree cover problem is defined as a connected edge dominating set with the total minimum edge weights, whereas the DTP is defined as a node dominating set.

Later, many approaches have been developed for the DTP. Sundar and Singh [15] developed one problem-specific heuristic (referred to as H_DT in the paper) and two swarm intelligence techniques – artificial bee colony algorithm (referred to as O_ABC^{DT} (O in O_ABC^{DT} stands for existing ABC algorithm for the DTP) in this paper) and ant colony optimization algorithm (referred to as ACO_DT in the paper) – for the DTP. Chaurasia and Singh [2] presented one problem-specific heuristic (referred to as M_DT in the paper) and an evolutionary algorithm with guided mutation (referred to as EA/G in the paper) for the DTP. Zorica et al. [3] presented a variable neighborhood search algorithm (referred to as VNS in the paper) for the DTP. Meanwhile, Sundar [13] presented a steady-state genetic algorithm (referred to SSGA in the paper) as for the DTP.

This paper presents two new heuristics for the DTP: first one is a problem-specific heuristic that exploits the problem structure effectively; and second one is an artificial bee colony (ABC) algorithm. The proposed ABC algorithm is different from existing ABC algorithm (O_ABC^{DT}) for the DTP [15] on two main components: initial solution generation, and determination of a neighboring solution. O_ABC^{DT} [15] which was the first developed metaheuristic technique for the DTP and is not competitive in terms of solution quality and computational time in comparison to other metaheuristic techniques developed later. This motivated us to develop an ABC algorithm for the DTP through main two components: initial solution generation; and determination of a neighboring solution. Both proposed methods i.e., problem-specific heuristic and ABC algorithm for the DTP have been tested on a set of standard benchmark instances and compared respectively with existing problem-specific heuristics (Heu_DT1, Heu_DT2, H_DT and M_DT) and metaheuristic techniques (O_ABC^{DT}, ACO_DT, EA/G, VNS, SSGA) in the literature. Computational results demonstrate the superiority of the proposed problem-specific heuristic and ABC algorithm for the DTP over all existing problem-specific heuristics and metaheuristic techniques respectively in the literature.

The structure of the remaining paper is as follows: Section 3 presents the problem-specific heuristic approach for the DTP. Section 4 presents a brief description of ABC algorithm and describes an ABC algorithm for the DTP. Section 5 reports computational results. Finally, Section 6 contains some concluding remarks.

3 Heuristic for the DTP

In literature, four different problem-specific heuristics (Heu_DT1 [11], Heu_DT2 [21], H_DT[15], M_DT [2]) have been so far presented for the DTP. A brief description of each such heuristic that highlights its properties is as follows:

1. Heu_DT1 [11] constructs a dominating tree based on active and inactive edge concepts. A pruning procedure is applied on the resultant tree.
2. Heu_DT2 [21] first creates a minimum spanning tree (MST) with the help of Kruskal Algorithm, then a sequence of search rule is applied to switch internal edges of MST to leaf edges as many as possible if there is a net gain. All the leaf edges of the resultant tree are pruned, and the remaining tree is a dominating tree.
3. H_DT [15] follows a greedy approach which is based on the concept of Kruskal algorithm and shortest paths between all pairs of vertices in G in order to construct a dominating tree. A pruning procedure is applied on the resultant tree.
4. M_DT [2] which is similar to H_DT [15] except selection of a next edge which is based on a criteria instead of selection of next edge based on least cost.

This section presents a new problem-specific heuristic for the DTP. The idea of developing a new and effective problem-specific heuristic for the DTP came from the observation of the objective of the DTP which lays the basis of two salient features, viz., minimum *edge-weight* set and set of vertices covering the given entire graph (G). This observation motivated us to focus on those nodes that cover the maximum number of non-dominating nodes in G and lead to a dominating tree with minimum cost. Particularly, the second salient feature is the key motivation for the development of a new problem-specific heuristic called Heu_2C_DTP. Heu_2C_DTP consists of two phases that are followed one-by-one.

- *Phase One*: In the beginning, find shortest paths between all pairs of vertices of a given G; label each vertex *unvisited*; compute *degree* of each vertex, where the *degree* of a vertex, say v, is defined as the total number of *unvisited* vert(ex/ices) adjacent to v; and compute *weight* of each vertex, where the *weight* of a vertex, say v, is defined as the total sum of weight of edge(s) incident to v. Initially, both dominating tree ($dTree$) and the set, say S, containing dominating nodes of $dTree$ are empty. Hereafter, select a vertex, say *first vertex* or v_s, with maximum *degree* from V. Note that there are chances of more than one vertex with same maximum *degree*. If such chances exist, a first tie-breaking rule is applied. As per this rule, select a vertex that has minimum *weight* in V. In the course of applying the first tie breaking rule, if more than one vertex with same minimum *weight* are encountered in V, then ties are broken arbitrarily. The selected v_s becomes a dominating node of the partial $dTree$ and is added to S. All *unvisited* vertices that are adjacent to v_s are labeled *visited* along with v_s. Update the *degree* of each vertex in $V \backslash S$.
- *Phase Two*: At each step, select a vertex (say u) with maximum *degree* from $V \backslash S$. Note that u may be *visited* or *unvisited* vertex. To establish a connection between u and a vertex, say v, in S, a shortest path, say SP, is determined. Each vertex except v in SP becomes a dominating node and is added to S. Each edge in SP becomes a part of partial $dTree$. Each *unvisited* vertex in SP along with its *unvisited* adjacent vert(ex/ices) is(are) labeled *visited*. Hereafter, update the *degree* of each vertex in $V \backslash S$. Note that there are chances of more than one vertex with maximum *degree*. If such chances exist, a first tie-breaking rule is applied. As per this rule, select a vertex that has shortest distance from the partial $dTree$. In the course of applying the first tie-breaking rule, if more than one vertex with same shortest distance are encountered, then further a second tie-breaking rule is applied. According to this rule, select a vertex whose shortest distance contains maximum number of *unvisited* vertices. Again, while applying the second tie-breaking rule, if further ties are encountered, then ties are broken arbitrarily. This iterative process continues until all vertices of G are labeled *visited*.

Hence, a $dTree$ is constructed with minimal number of dominating nodes. It is possible that constructed $dTree$ on the set of dominating nodes is not optimal due to some edges which are incorrect, but are part of resultant $dTree$. Since, many dominating trees (spanning trees) are possible on the subgraph of G induced by the set of dominating nodes of resultant $dTree$. Through Prim's algorithm [10], a new dominating tree with optimal cost (minimum spanning tree) can be constructed on the subgraph of G induced by the set of dominating nodes of resultant $dTree$. In doing so, no constraint of DTP is violated. This new dominating tree (minimum spanning tree) replaces the resultant $dTree$ and becomes $dTree$. Further, a pruning method is applied on $dTree$. This method starts with examining each current leaf node, say v_{lf}, of resultant $dTree$ one-by-one. If all non-dominating nodes adjacent to v_{lf} are also adjacent to other dominating nodes in $dTree$, then the edge incident to v_{lf} is eliminated from $dTree$, resulting in further deduction in the total cost of the resultant $dTree$. v_{lf} becomes a non-dominating node and is labeled *unvisited*. This pruning method is applied on only current leaf nodes of $dTree$. The idea of constructing MST and pruning method was first applied in [15].

Algorithm 1 The pseudo-code of Heu_2C_DTP

```
Input  : A connected, weighted, and undirected graph
         $G = (V, E, w)$;
Output: A dominating tree, say $dTree$;

Find shortest paths between all pairs of vertices of $G$;
$dTree \leftarrow \phi$; $S \leftarrow \phi$;
for each vertex u in V do
    $u \leftarrow unvisited$;
for each vertex u in V do
    $degree[u] \leftarrow$ Total number of $unvisited$
    vert(ex/ices) in $Adj[u]$ ; // Adj[u] means
    vert(ex/ices) adjacent to u
    $weight[u] \leftarrow \sum_{i=0}^{i<|Adj[u]|} w(e(Adj[i], u))$;
Select a vertex $v_s$ with maximum degree from $V$ ;
// Apply tie-breaking rule if there
exists ties
$S \leftarrow S \bigcup v_s$; $v_s \leftarrow visited$; $M_v \leftarrow M_v + 1$;
for each vertex i in $Adj[v_s]$ do
    $i \leftarrow visited$; $M_v \leftarrow M_v + 1$;
for each vertex i in $V \setminus S$ do
    Update $degree[i]$;
while $M_v \neq |V|$ do
    Select a vertex (say u) with maximum degree from
    $V \setminus S$ ; // Apply tie-breaking rule if
    there exists ties
    Determine a shortest path (say SP) for connection
    establishment between u and a vertex (say v) in S ;
    // Apply tie-breaking rule if
    there exists ties
    for each vertex i in SP except v do
        $S \leftarrow S \bigcup i$;
        if i is labeled unvisited then
            $i \leftarrow visited$; $M_v \leftarrow M_v + 1$;
        for each vertex $j \in Adj[i]$ do
            if j is labeled unvisited then
                $j \leftarrow visited$; $M_v \leftarrow M_v + 1$;
    for each edge e in SP from u to v do
        $dTree \leftarrow dTree \bigcup e$;
    for each vertex i in $V \setminus S$ do
        Update $degree[i]$;
Construct a new dTree on the subgraph of G induced
by dominating vertices in S using Prim's algorithm;
Apply pruning procedure on current leaf nodes of
newly constructed dTree;
Return dTree;
```

Note that the idea of applying labeling vertex *unvisited* or *visited* is very common as it clearly distinguishes whether a vertex is in $dTree$ or not. Similar ideas can be observed in [2, 15, 20]. The idea of applying *degree* concept for each vertex is rational, as each time selecting a vertex with maximum *degree* during the construction of $dTree$ increases the chances of covering the maximum number of *unvisited* vertices. Also, in case of first-tie breaking rule in *Phase One*, the idea of applying minimum *weight* concept in conjunction with the maximum *degree* only once during the selection of *first vertex* or v_s is intuitively meaningful, as whenever an edge (say $e(u, v_s)$) incident to v_s will get a chance to be a part of partial $dTree$ during establishing a connection between an unselected vertex (u) and v_s through a shortest path concept, its weight (weight on $e(u, v_s)$) will be of possible minimum weight due to minimum *weight* concept on a vertex with maximum *degree*. Our preliminary experiments justify this intuitive idea. In *Phase Two*, applying shortest path concept for connection establishment between an unselected vertex and a vertex in the partial $dTree$ is common-sense or intuitively correct, as such shortest path consisting of edges of minimum weight will help in constructing a dominating tree with minimum cost. Further this phase also uses two tie-breaking rules, i.e., first tie-breaking rule and second tie-breaking rule with the aim of minimizing the cost of $dTree$ and maximizing the number of non-dominating nodes.

The psuedo-code of Heu_2C_DTP is presented in Algorithm 1. One can observe in the pseudo-code of Heu_2C_DTP that the running time of Heu_2C_DTP is mainly dominated by finding shortest paths between all pairs of vertices in G.

Figure 2 illustrates how Heu_2C_DTP works. Figure 2a represents a connected, weighted and undirected graph $G = (V, E, w)$, where $|V| = 9$ and $|E| = 13$. Figure 2b–i depict various stages of execution of Heu_2C_DTP. Initially, $dTree$ and S are two empty sets. Each vertex in V is labeled *unvisited* shown in *white* color. A vertex with maximum *degree* is selected. In Fig. 2a, there exists more than one vertex with same maximum *degree*, i.e., $< 2, 7 >$ in $V \setminus S$. To handle this situation, a first tie-breaking rule (see *Phase One*) is applied. This rule selects vertex 2 as a dominating vertex shown in *dark grey* color, because the *weight* associated with vertex 2 is less than that of vertex 7. Vertex 2 is added to S; labeled visited; and shown in *dark grey* color. All vertices adjacent to vertex 2 are labeled visited and are shown in *grey color.* This is shown in Fig. 2b. Update the *degree* of each vertex in $V \setminus S$. Hereafter, at each iteration, a vertex of maximum *degree* is selected from $V \setminus S$. In the first iteration, there exists more than one vertex with same maximum *degree*, i.e., $< 1, 7, 5, 8 >$ in $V \setminus S$. To handle this, a first tie-breaking rule (see *Phase Two*) is applied. As per this rule, vertex 8 is selected as a dominating vertex shown in *dark grey* color due to having shortest distance from $dTree$. Edge (2, 8) is added to $dTree$, and vertex 8 is added to S. The label of vertex 8 is already visited. All unvisited vertices

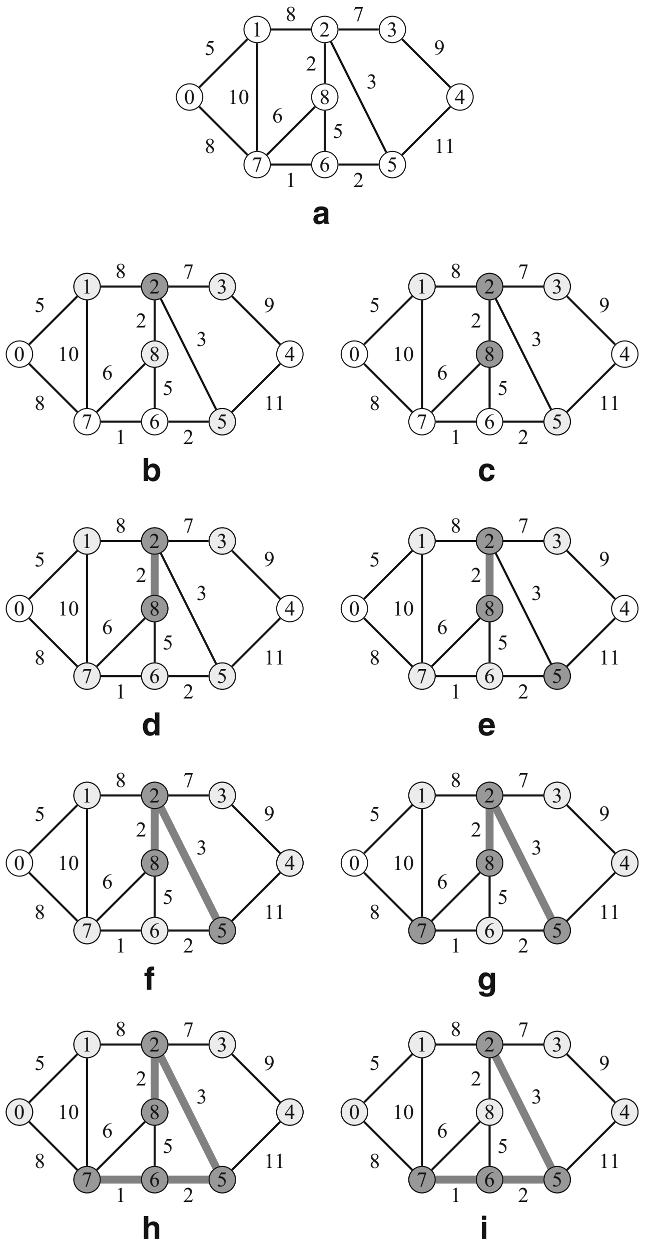

Fig. 2 The various stages in execution of Heu_2C_DTP

adjacent to vertex 8 are labeled visited and are shown in *grey* color. This is shown in Fig. 2c–d. Update the *degree* of each vertex in $V \backslash S$. In the second iteration, there exists more than one vertex with the same maximum degree, i.e., $< 1, 3, 5, 7 >$ in $V \backslash S$. As per the first-tie breaking rule (see *Phase Two*), vertex 5 is selected as a dominating vertex shown in *dark grey* color due to having shortest distance from $dTree$. Edge (2, 5) is added to $dTree$, and vertex 5 is added to S. The label of vertex 5 is already visited. All unvisited vertices adjacent to vertex 5 are labeled visited and are shown in *grey* color. This is shown in Fig. 2e–f. Update the *degree* of each vertex in $V \backslash S$. In the final iteration, there exists more than one vertex with the same maximum degree, i.e., $< 7, 1 >$ in $V \backslash S$. As per the first-tie breaking rule (see *Phase Two*), vertex 7 is selected as a dominating vertex shown in *dark grey* color due to having shortest distance from $dTree$. Edges (7, 6) and (6, 5) are added to $dTree$, and vertices 7 and 6 are added to S. This is shown in Fig. 2g–h. At this stage all the vertices of $dTree$ are visited, so Heu_2C_DTP stops here. Further, Prim's algorithm [10] is applied on the subgraph of G induced by the set of dominating vertices (nodes), i.e., $<$ 2, 5, 6, 7, 8 $>$ in S of resultant $dTree$ (Fig. 2h) in order to construct a new dominating tree with optimal cost (minimum spanning tree). Resultant minimum spanning tree is same as $dTree$ (Fig. 2h) obtained by Heu_2C_DTP. Pruning method is applied on resultant $dTree$ shown in Fig. 2h. Only vertex 8 is the leaf node that can be pruned without violating the constraints of DTP. Vertex 8 is pruned and deleted from S. Pruned vertex 8 is now shown in *grey* color. Figure 2i shows the resultant $dTree$ whose dominating nodes, i.e., $< 2, 5, 6, 7 >$ are in S. Heu_2C_DTP returns this resultant $dTree$ as the final $dTree$.

4 Artificial bee colony algorithm

Artificial bee colony (ABC) algorithm is one among swarm intelligence techniques and inspired by foraging behavior of honey bees in nature [7]. ABC algorithm models the collective behavior of decentralized and self organized systems. Like real bees, ABC algorithm also categorizes artificial bees into three different groups: employed bees; scout bees; and onlooker bees in order to search high quality solutions for the optimization problem under consideration. A food source represents a feasible solution to the problem, and the nectar amount of its food source corresponds to the fitness of its solution. Since each food source is uniquely associated with an employed bee, the number of solutions is same as the number of employed bees. ABC algorithm starts with generating a fixed set of initial solution (food source). Then at each iteration, each group of artificial bee works as follows:

- *Employed bees*: Each employed bee performs the job of determining a new solution in the neighborhood of its currently associated solution. If the new solution, in terms of fitness, is better than that of its currently associated solution, then the current employed bee will move to this new solution discarding the old one, otherwise it will continue with its old one.
- *Scout bees*: If the solution is not improving for some time, controlled by a parameter called *limit*, then its associated employed bee becomes a scout bee by discarding its solution. The job of scout bee is to generate a new random solution. Once the new solution is

generated, the status of scout bee changes to employed bee on this new solution.

- *Onlooker bees*: Once each employed bee completes the job of determining a neighboring solution, the job of each onlooker bee starts. Each onlooker bee uses probability-based selection method to select a solution associated by an employed bee, and then it determines a new solution in the neighborhood of its selected solution that is similar to determining a new neighboring solution by an employed bee. This selection method biases towards selection of high quality solution. Once the job of each onlooker bee in terms of selecting a solution and determining a new neighboring solution is done, then all new solutions – determined in the neighborhood of a particular solution (say X) selected by one or more onlooker bees – and the solution itself X compete against each other for the position of solution X in the next iteration. The best in them will be chosen for the new position of solution X in the next iteration. Once new positions of all solutions are chosen, the next iteration of the ABC algorithm is carried out.

This whole iterative procedure is applied again and again until the termination criteria is met.

Readers can find a detail of ABC algorithm and its applications in [8].

4.1 ABC algorithm for the DTP

This subsection presents an ABC algorithm (ABC_DTP) for the DTP. Hereafter, the proposed ABC algorithm for the DTP will be referred to as ABC_DTP. The description of each component of ABC_DTP is as follows:

4.1.1 Initial solution generation

Instead of generating each initial solution of the population randomly as used in [15], ABC_DTP follows a random version of the proposed heuristic Heu_2C_DTP for generating each initial solution of the population. This random version also contains two phases which are as follows:

- *Phase One*: In the beginning, label each vertex unvisited; calculate *degree* of each vertex similar to *degree* computed in Heu_2C_DTP. Initially, the dominating tree ($dTree$) and the set, say S, containing dominating nodes of $dTree$ are empty. Each vertex in V whose *degree* is greater than zero is kept in a set, say D. Hereafter, select a vertex, say *first vertex* or v_s randomly from D. The selected v_s becomes a dominating node of the partial $dTree$ and is added to S. All unvisited vertices that are adjacent to v_s are labeled visited along with v_s. Update the degree of each vertex in $V \backslash S$. Now D will contain only those vertices in $V \backslash S$ whose degree is greater than zero.
- *Phase Two*: At each step, select a vertex (say u) randomly from D. Note that u may be *visited* or *unvisited* vertex. To establish a connection between u and a vertex, say v, in S, a shortest path, say SP, is determined. Each vertex except v in SP becomes a dominating node and is added to S. Each edge in SP becomes a part of partial $dTree$. Each unvisited vertex in SP as well as its unvisited adjacent vert(ex/ices) are labeled visited. Hereafter, update the degree of each vertex in $V \backslash S$. Now D will contain only those vertices in $V \backslash S$ whose *degree* is greater than zero. Note that if there exists more than one shortest path of same cost, then select a path that consists of maximum number of vertices. This iterative process continues until all vertices of G are labeled visited.

Once $dTree$ is constructed, pruning procedure [15] is applied repeatedly until no leaf node in $dTree$ can be pruned. After that, a minimum spanning tree (MST) is constructed on the sub-graph of G induced by the set (S) of dominating vertices of $dTree$ with the help of Prim's algorithm. The concept of pruning-MST applied here is similar to [15].

We have also tested with generating each initial solution of the population randomly; however, our initial experiments have suggested that this way led to inferior solution quality in comparison to generating each initial solution of the population with the help of a mixed strategy that uses randomness and problem-structure knowledge.

Each employee bee uniquely associates with each initial solution ($dTree$). The fitness of each solution is computed.

4.1.2 Probability of selecting a solution

In *onlooker phase*, each onlooker bee selects a solution (food source), which is one among all solutions associated by all employed bees, with the help of binary tournament selection method. In this selection method, two solutions from all solutions associated with all employed bees are picked randomly. With probability P_{bt}, a solution with better fitness is selected, otherwise worse one is selected.

4.1.3 Determination of a neighboring solution

Determining a new neighboring solution of high quality relies heavily on how problem structure of a combinatorial optimization problem is unravelled. In this direction, we propose two methods applied in a mutually exclusive way for determining a solution (say X^c) in the neighborhood of current solution (X). First method is $\mathcal{C}$*NAS-Method* that is based on copy a set of dominating nodes from another solution of the population to current solution, whereas second method is $\mathcal{M}$*EDI-Method* that is based on performing random multiple *edge-deletion-insertion* on current solution. Initially, a copy (say X^c) of X is created. With probability P_{nbr}, $\mathcal{C}$*NAS-Method* is applied, otherwise $\mathcal{M}$*EDI-Method* is applied.

1. $\mathcal{C}$*NAS-Method:* Initially, a solution, say Y, (different from X) is picked from the population with the help of binary tournament selection method. Then, $\mathcal{C}$*NAS-Method* picks at most y_dn dominating nodes of Y different from dominating nodes of X and assigns them to a set, say S_y. y_dn is equal to *mdn%* of total number of dominating nodes of Y. *mdn* is a parameter to be determined empirically. Hereafter, at each step, a vertex, say u, in the order is picked from S_y. An edge connecting u and a vertex in current X^c is searched, as soon as an edge connecting u and a vertex in current X^c is found, it is immediately added to current $dTree$ of X^c. Vertex u is added to X^c. This procedure is repeated again and again until all nodes in S_y are added to X^c.

   Note that if $\mathcal{C}$*NAS-Method* fails to pick a dominating node of Y different from dominating nodes of X, then it shows that X and Y are same, which in turn also shows that employed bee solutions are suffering from a lack of diversity. This situation is coined as *collision* [12, 14]. In such a situtation, $\mathcal{M}$*EDI-Method* is applied on X instead of abandoning this solution [16]. Abandoning this solution means employed bee associated with this solution abandons it to become scout so that the diversity in the population can be improved. However, initial experiments have suggested that this way led to inferior solution quality overall in comparison to applying $\mathcal{M}$*EDI-Method* which is perturbation strategy.
2. $\mathcal{M}$*EDI Method:* This method follows a certain number of *edge-deletion-insertion* procedure which is applied S_e times, where S_e is equal to *edi%* of total number of edges of X. *edi* is a parameter to be determined empirically. Initially, this method picks a certain number of edges, where degree of atleast one end vertex (i or j) of each picked edge (say *e(i, j)*) must be greater than one in G. All picked edges are assigned to a set, say S_e. Note that S_e includes only edges of X (not of X^c), as X^c becomes different from X after applying first successful *edge-deletion-insertion* procedure. Hereafter, at each step, this method picks an edge (say *e(i, j)*) randomly from S_e and deletes it from X^c, resulting a partition of X^c into two disjoint sets, say C_1 and C_2. To connect C_1 and C_2, a *second connectivity rule* is applied. This rule searches a *particular* shortest path from all candidate shortest paths connecting C_1 and C_2. This *particular* shortest path does not contain those edges that are already part of current $dTree$ of X^c and that are not added from any previous *edge-deletion-insertion* procedure if it has occurred already for the current $dTree$ of X^c as *edge-deletion-insertion* procedure performs multiple times for a given solution. Once this *particular* shortest path is found, all new edges in this *particular* shortest path are inserted to connect C_1 and C_2 of X^c. However, if such *particular* shortest path is not possible, then deleted edge (*e(i, j)*) is restored to X^c.

   Note that if $\mathcal{M}$*EDI-Method* fails to perform a single *edge-deletion-insertion* procedure, then in case of employed bee phase, current solution X is replaced with a new initial solution and in case of onlooker bee phase, a very large fitness value is assigned to fitness of onlooker bee associated with X.

Once the neighboring solution X^c is determined, a pruning procedure [15] is applied repeatedly on X^c until no leaf node in $dTree$ of X^c except all new nodes added through either $\mathcal{C}$*NAS-Method* or $\mathcal{M}$*EDI-Method* can be pruned. After that, a minimum spanning tree (or dominating tree) is constructed on the sub-graph of G induced by the set (S) of dominating vertices of $dTree$ with the help of Prim's algorithm [10]. This new dominating tree (minimum spanning tree) replaces the resultant $dTree$ of X^c and becomes $dTree$ of X^c. Again, the process of pruning is applied repeatedly on resultant $dTree$ of X^c until no leaf node (including all new nodes added through either $\mathcal{C}$*NAS-Method* or $\mathcal{M}$*EDI-Method*) in $dTree$ of X^c can be pruned.

Note that determining a neighboring solution which is an important component of ABC_DTP is different from that of previous ABC algorithm for the DTP, i.e., O_ABC^{DT} [15] except pruning-MST-pruning methods. O_ABC^{DT} [15] also applies two methods that are based on single *edge-deletion-insertion* procedure randomly from the current solution and adding a non-dominating vertex randomly to the current solution in a mutually exclusive way. However, ABC_DTP uses $\mathcal{C}$*NAS-Method* and $\mathcal{M}$*EDI-Method* in a mutually exclusive way. $\mathcal{C}$*NAS-Method* is based on this concept that if a node as a dominating node is present in a good solution then the same node will be present in many good solutions. $\mathcal{M}$*EDI-Method* based on multiple *edge-deletion-insertion* procedure leads to better exploration in the search space in comparison to single *edge-deletion-insertion* procedure [15]. Experimental results justify that these two methods for determining a neighboring solution coordinated with other components of ABC_DTP make ABC_DTP more effective and robust for searching high quality solutions.

Algorithm 2 The pseudo-code of ABC_DTP

```
Generate a set of $\mathcal{NE}$ solutions, i.e., $\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_{\mathcal{NE}}$;
$Best \leftarrow$ Best solution in $\mathcal{NE}$ solutions;
while Termination criteria is not met do
    for $i \leftarrow 1$ to $\mathcal{NE}$ do
        $\mathcal{E}' \leftarrow \mathcal{DN}bring\_Sol(\mathcal{E}_i)$;
        if $\mathcal{E}'$ is better than $\mathcal{E}_i$ then
            $\mathcal{E}_i \leftarrow \mathcal{E}'$;
        else if $\mathcal{E}_i$ is not improving last $limit$ iterations then
            Scout bee;
        if $\mathcal{E}_i$ is better than $best$ then
            $best \leftarrow \mathcal{E}_i$;
    for $i \leftarrow 1$ to $\mathcal{NO}$ do
        $s_i \leftarrow \mathcal{B}inary\_TSM(\mathcal{E}1, \mathcal{E}2, ..., \mathcal{E}_{\mathcal{NE}})$;
        $\mathcal{O}_i \leftarrow \mathcal{DN}bring\_Sol(\mathcal{E}_{s_i})$;
    for $i \leftarrow 1$ to $\mathcal{NO}$ do
        if $\mathcal{O}_i$ is better than $\mathcal{E}_{s_i}$ then
            $\mathcal{E}_{s_i} \leftarrow \mathcal{O}_i$;
        if $\mathcal{O}_i$ is better than $best$ then
            $best \leftarrow \mathcal{O}_i$;
```

4.1.4 Other features

A solution that is not improving in terms of fitness for some iterations (controlled by a parameter called *limit*) is abandoned by its employed bee. The employed bee becomes a scout. This scout favors in generating a new random solution. Once a new solution is generated (similar to initial solution generation (see Section 4.1.1)), the status of scout bee changes to employed bee on this new solution.

Algorithm 2 presents the pseudo-code of ABC_DTP. In this pseudo-code, binary tournament selection method is called by $\mathcal{B}inary\_TSM(\mathcal{E}1, \mathcal{E}2, ..., \mathcal{E}_{\mathcal{NE}})$ function which returns a selected solution; and the method for determining a solution in the neighborhood of a solution, say $\mathcal{X}$, is called by $\mathcal{DN}bring\_Sol(\mathcal{X})$ function which returns a neighboring solution, say $\mathcal{X}'$.

5 Computational results

Both proposed approaches – Heu_2C_DTP and ABC_DTP – were implemented in C and tested on three different benchmark instance sets for evaluation. All experiments were executed on a Linux based 1.6 GHz Core 2 Duo system with 1 GB RAM which is different from that of previous existing metaheuristic techniques for the DTP, such as O_ABC^{DT} [15], ACO_DT [15], EA/G [2], SSGA [13], and VNS [3]. O_ABC^{DT}, ACO_DT, EA/G and SSGA used Intel Core 2 Duo processor 3.0 GHz with 2 GB RAM under Fedora 12, whereas VNS used Intel Core I7-4702MQ 2.2 GHz with 4 GB RAM under Windows XP. In such circumstances, it is difficult to exactly compare the speed of these metaheuristic techniques with ABC_DTP; however, a rough comparison can always be made. One can observe SSGA in terms of computational time and solution quality (see Table 6) that overall SSGA is superior to O_ABC^{DT}, ACO_DT, EA/G and VNS. This observation gives an idea for rough comparison, i.e., the number of solutions generated by ABC_DTP is approximately similar to that of SSGA [13] in order to test the effectiveness of ABC_DTP. Since SSGA [13] generates X_Sol solutions to find a high quality solution for each instance, where X_Sol is equal to the sum of size of initial population and $|V| \times 500$. Therefore, ABC_DTP is also allowed to generate approximately Y_Sol solutions to find a high quality solution for each instance, where Y_Sol is equal to the sum of size of initial population and $(|V| \times 500)/(\mathcal{NE} + \mathcal{NO})$. In addition, ABC_DTP, similar to all previous existing metaheuristic techniques for the DTP, was also executed 20 independent times on each instance in order to test its robustness. Heu_2C_DTP, similar to all previous existing problem-specific heuristics – such as Heu_DT1 [11], Heu_DT2 [21], H_DT [15] and M_DT [2] –, Heu_2C_DTP was also executed once.

All instance sets used for the DTP are available on http://dcis.uohyd.ac.in/~alokcs/dtp.zip. Each instance [11, 21] is considered as a disk graph $G = (V, E)$, where disk of a node denotes the transmission range of that node. For every pair of nodes, there will be an edge iff two nodes connecting this edge will be in the common area of their corresponding disks. For each edge $e_{u,v} \in E$, there exists a non-negative weight $w(e_{u,v})$ which is defined as $C_j \times d^2_{u,v}$, where $d^2_{u,v}$ is the Euclidean distance between u and v; and C_j is a random constant whose value is set to 1. Since instances used in [11, 21] were not available, Sundar and Singh [15], similar to [11, 21], generated a set of instances with different size varying in total number of nodes. Nodes are randomly distributed over an area of $500m \times 500m$, and transmission range of each node is 100m. For each size, i.e., |V| = {50, 100, 200, 300, 400, 500}, three different instances were generated, resulting a total of 18 instances. Later, Chaurasia and Singh [2] generated two different instance sets exactly similar to instance set [15], but the transmission range of each node in two different instance sets is 125m and 150m, resulting in generation of additional 36 (18+18) different instances.

In the next three subsections, parameter settings for ABC_DTP, comparison of the results obtained by Heu_2C_DTP with that of existing problem-specific heuristics in the literature, and comparison of the results obtained by ABC_DTP with that of existing metaheuristic approach in the literature are reported respectively.

5.1 Parameter tuning for ABC_DTP

Being stochastic in nature, proper tuning of parameters for ABC_DTP become significant. For this, various possible values of each candidate parameter (reported in Table 1) were chosen based on preliminary experimentations and available literature. To investigate parameter sensitiveness, two different instances from each instance set were taken into account. Then, the performance of ABC_DTP on different combination due to possible values of different parameters were carefully tested on these instances. One can observe in Table 2, where column *Parameter* denotes various parameters ($\mathcal{NE}$, $\mathcal{NO}$, P_{bt}, P_{nbr}, *mdn*, *edi*, *limit*) used for ABC_DTP; *val_para* denotes possible values of each parameter; *Best* and *Avg* respectively denote the best value and the average value over 20 runs obtained on such instances from corresponding value of the parameter. This investigation led to the best combination of parameter settings, i.e., $\mathcal{NE} = 50$, $\mathcal{NO} = 100$, $P_{bt} = 0.85$, $P_{nbr} = 0.7$, *mdn* = 20, *edi* = 10, and *limit* = 50 that approximately produces high quality solutions (in terms of *Best* and *Avg* in Table 2) highlighted in bold on most of these instances taken into account. Note that this testing uses every time one possible value of a parameter, while keeping values of other remaining parameters fixed (see *val_para* in bold of each parameter).

5.2 Comparison of Heu_2C_DTP with Heu_DT1, Heu_DT2, H_DT and M_DT

In this subsection, we present an overview of effectiveness of the proposed problem-specific heuristic, i.e., Heu_2C_DTP in comparison to other existing problem-specific heuristics [2, 11, 15, 21], which will be referred to as Heu_DT1 [11], Heu_DT2 [21], H_DT [15] and M_DT [2] respectively. Tables 3, 4 and 5 report the results of Heu_2C_DTP along with Heu_DT1, Heu_DT2, H_DT and M_DT on instances with transmission range 100m, 125m and 150m respectively. In Tables 3–5, column *Instance* denotes the name of each instance, and for each heuristic, column *Value* denote the value obtained on each instance. In addition, similar to [2, 11, 15, 21], column *NDN* that denotes the number of nodes in dominating tree obtained on each instance is added, as the total number of dominating nodes has a significant role in the performance of any routing protocols based on virtual backbone structure. In Table 3, results of Heu_DT1, Heu_DT2, H_DT are taken from [15], whereas results of M_DT are taken from [2]. In Tables 4 and 5, results of Heu_DT1, Heu_DT2, H_DT and M_DT on instances with transmission range 125m and 150m respectively are taken from [2]. For each instance, the best value *Value* among Heu_DT1, Heu_DT2, H_DT, M_DT and Heu_2C_DTP is highlighted in bold.

Tables 3, 4 and 5 clearly show that Heu_2C_DTP outperforms all previous existing problem-specific heuristics, i.e., Heu_DT1, Heu_DT2, H_DT and M_DT in terms of solution quality. In terms of *Value*, Heu_2C_DTP is better on all 54 instances in comparison to Heu_DT1, Heu_DT2, H_DT and M_DT. In terms of *NDN*, Heu_2C_DTP is better on all 54 instances except on one instance, i.e., *300_3* in 150m range whose *NDN* value is similar to that of M_DT.

Execution times of Heu_2C_DTP, H_DT and M_DT are dominated by precomputing all pair shortest paths in G. It is mentioned in [15] that Heu_DT1 and Heu_DT2 are faster than H_DT. Hence Heu_2C_DTP is slower than Heu_DT1 and Heu_DT2. Since Heu_2C_DTP outperforms all Heu_DT1, Heu_DT2, H_DT and M_DT in terms of solution quality and the number of dominating vertices for each test instance, such performance can compensate its execution time.

5.3 Comparison of ABC_DTP with state-of-the-art metaheuristic techniques

This subsection presents an overview of effectiveness of ABC_DTP with state-of-the-art metaheuristic techniques such as O_ABC^{DT} [15], ACO_DT [15], EA/G [2], SSGA [13] and VNS [3] for the DTP. Table 6 reports the results of ABC_DTP along with the results of O_ABC^{DT}, ACO_DT,

Table 1 Potential values of each parameter for ABC_DTP

Parameter	Description	Possible values of a parameter
$\mathcal{NE}$	Number of employed bees	{25, 50, 100}
$\mathcal{NO}$	Number of onlooker bees	{50, 100, 150}
P_{bt}	Probability used in Binary Tournament Selection Method	{0.8, 0.85, 0.90}
P_{nbr}	Probability used for selection of neighborhood	{0.6, 0.7, 0.8}
mdn	Percentage value of total number of dominating nodes of Y	{10, 20, 30}
edi	Percentage value of total number of edges in X	{5, 10, 15}
limit	A value in terms of iterations used for abandoning a solution (Scout bee)	{25, 50, 100}

Table 2 Influence of parameter setting on solution quality

Parameter	val_para	Transmission Range 100m				Transmission Range 125m				Transmission Range 150m			
		050_1		500_3		050_1		500_3		050_1		500_3	
		Best	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*
	25	1204.41	1204.85	1232.01	1241.12	802.95	802.95	981.90	987.75	647.75	647.75	808.50	808.52
$\mathcal{NE}$	50	**1204.41**	**1204.41**	**1231.95**	**1236.33**	**802.95**	**802.95**	**981.90**	**986.01**	**647.75**	**647.75**	**808.50**	**808.50**
	100	1204.41	1204.41	1233.26	1240.10	802.95	802.95	982.49	985.19	647.75	647.75	808.50	808.50
	50	1204.41	1204.41	1233.15	1237.31	802.95	802.95	982.49	986.84	647.75	647.75	808.50	808.50
$\mathcal{NO}$	100	**1204.41**	**1204.41**	**1231.95**	**1236.33**	**802.95**	**802.95**	**981.90**	**986.01**	**647.75**	**647.75**	**808.50**	**808.50**
	150	1204.41	1204.41	1231.95	1238.70	802.95	802.95	981.90	986.11	647.75	647.75	808.50	808.50
	0.80	1204.41	1204.41	1232.65	1237.41	802.95	802.95	982.37	986.67	647.75	647.75	808.50	808.50
P_{bt}	0.85	**1204.41**	**1204.41**	**1231.95**	**1236.33**	**802.95**	**802.95**	**981.90**	**986.01**	**647.75**	**647.75**	**808.50**	**808.50**
	0.90	1204.41	1204.41	1232.60	1238.11	802.95	802.95	982.37	986.00	647.75	647.75	808.50	808.50
	0.60	1204.41	1204.41	1232.65	1237.41	802.95	802.95	982.37	987.23	647.75	647.75	808.50	808.52
P_{nbr}	0.70	**1204.41**	**1204.41**	**1231.95**	**1236.33**	**802.95**	**802.95**	**981.90**	**986.01**	**647.75**	**647.75**	**808.50**	**808.50**
	0.80	1204.41	1204.41	1233.15	1239.99	802.95	802.95	982.08	985.43	647.75	647.75	808.37	808.50
	10	1204.41	1204.41	1238.33	1254.50	802.95	802.95	996.16	1001.38	647.75	647.88	808.50	809.36
mdn	20	**1204.41**	**1204.41**	**1231.95**	**1236.33**	**802.95**	**802.95**	**981.90**	**986.01**	**647.75**	**647.75**	**808.50**	**808.50**
	30	1204.41	1205.30	1232.01	1242.27	802.95	802.95	983.78	989.81	647.75	647.75	808.37	808.89
	5	1204.41	1204.41	1235.32	1243.20	802.95	802.95	982.37	986.07	647.75	647.75	808.50	808.50
edi	10	**1204.41**	**1204.41**	**1231.95**	**1236.33**	**802.95**	**802.95**	**981.90**	**986.01**	**647.75**	**647.75**	**808.50**	**808.50**
	15	1204.41	1204.41	1231.95	1238.22	802.95	802.95	980.67	984.96	647.75	647.75	808.50	808.50
	25	1204.41	1204.41	1231.95	1237.94	802.95	802.95	981.90	986.07	647.75	647.75	808.50	808.50
limit	50	**1204.41**	**1204.41**	**1231.95**	**1236.33**	**802.95**	**802.95**	**981.90**	**986.01**	**647.75**	**647.75**	**808.50**	**808.50**
	100	1204.41	1205.74	1231.95	1239.55	802.95	802.95	982.37	986.00	647.75	647.75	808.50	808.56

Table 3 Results of Heu_DT1 [11], Heu_DT2 [21], H_DT [15], M_DT [2] and Heu_2C_DTP on the instances with transmission range 100m

Instance	Heu_DT1 [11]		Heu_DT2 [21]		H_DT [15]		M_DT [2]		Heu_2C_DTP			
	Value	*NDN*	*Value*	*NDN*	*Value*	*NDN*	*Value*	*NDN*	*Value*	*NDN*	*BKV*	*PRD*
50_1	1608.11	30	1819.91	35	1321.83	20	1288.80	20	**1231.14**	18	1204.41	2.22
50_2	1564.05	33	1795.89	35	1427.65	26	1467.03	26	**1405.95**	21	1340.44	4.89
50_3	1659.70	34	1863.01	35	1494.12	25	1429.76	24	**1372.77**	20	1316.39	4.28
100_1	1836.57	54	2290.28	61	1852.86	28	1482.11	26	**1363.97**	21	1217.47	12.03
100_2	2096.97	62	2265.68	62	1449.25	23	1454.16	25	**1248.75**	17	1128.40	10.67
100_3	2213.15	60	2488.15	59	1732.35	29	1704.19	31	**1443.71**	21	1252.99	15.22
200_1	2530.57	110	3093.01	115	1880.50	30	1766.97	35	**1387.50**	21	1206.79	14.97
200_2	2709.42	113	3437.79	125	1909.86	32	1695.43	34	**1401.00**	21	1216.23	15.19
200_3	2561.99	110	3132.56	112	1587.48	27	1589.81	31	**1387.35**	21	1247.25	11.23
300_1	2932.26	154	3653.64	165	1929.91	34	1695.08	30	**1417.93**	22	1215.48	16.66
300_2	3480.73	178	4136.57	183	1781.00	31	1773.32	35	**1357.11**	22	1170.85	15.91
300_3	3640.35	184	3990.55	170	1815.28	33	1673.31	33	**1517.27**	26	1247.51	21.62
400_1	3776.71	230	4524.29	228	2017.50	32	1587.43	30	**1438.16**	23	1211.72	18.69
400_2	4004.41	243	4744.41	248	1972.89	36	1904.82	37	**1485.52**	25	1199.23	23.87
400_3	4026.04	241	4394.95	218	1907.05	29	1883.79	32	**1430.32**	22	1246.94	14.71
500_1	4276.57	291	4534.93	257	1795.28	27	1771.82	34	**1408.41**	23	1200.06	17.36
500_2	4399.44	296	5251.35	309	1824.03	34	1683.54	29	**1532.62**	24	1220.68	25.55
500_3	4629.12	304	4944.21	269	1903.86	29	1837.40	30	**1436.96**	24	1231.81	16.65

Table 4 Results of Heu_DT1 [11], Heu_DT2 [21], H_DT [15], M_DT [2] and *Heu_2C_DTP* on the instances with transmission range 125m

Instance	Heu_DT1 [11]		Heu_DT2 [21]		H_DT [15]		M_DT [2]		*Heu_2C_DTP*			
	Value	*NDN*	*Value*	*NDN*	*Value*	*NDN*	*Value*	*NDN*	*Value*	*NDN*	*BKS*	*PRD*
50_1	1404.49	26	1679.80	29	982.61	14	1047.25	13	**802.95**	10	802.95	0.00
50_2	1407.53	26	1705.67	31	1165.63	16	1179.19	19	**1155.90**	15	1055.10	9.55
50_3	1488.20	29	1774.12	32	1154.05	14	1201.88	19	**929.48**	12	877.77	5.89
100_1	1737.47	50	2258.10	56	1442.11	18	1331.99	20	**1007.73**	14	943.01	6.86
100_2	1969.81	59	2372.79	60	1511.53	21	1238.10	20	**981.83**	11	917.00	7.07
100_3	2213.15	59	2402.43	59	1445.39	20	1311.60	21	**1055.66**	15	998.18	5.76
200_1	2510.88	109	2990.67	100	1639.11	20	1355.67	24	**1060.79**	14	910.17	16.55
200_2	2733.43	113	3074.32	115	1436.93	23	1367.08	20	**1040.25**	14	921.76	12.85
200_3	2540.70	108	3118.57	109	1345.50	21	1307.22	20	**1165.63**	14	939.58	24.06
300_1	2899.16	153	3537.02	150	1454.30	19	1516.07	26	**1156.68**	17	977.65	18.31
300_2	3500.06	180	4310.97	189	1739.35	23	1387.87	19	**1063.61**	14	913.01	16.49
300_3	3612.75	183	3802.58	160	1541.75	21	1370.75	20	**1129.57**	16	974.78	15.88
400_1	3758.23	228	4211.58	210	1654.87	23	1528.15	24	**1200.75**	17	965.99	24.30
400_2	3901.81	233	4596.69	235	1739.40	25	1539.59	23	**1167.08**	16	941.02	24.02
400_3	3981.63	238	4421.78	213	1630.39	23	1524.44	24	**1122.09**	16	1002.61	11.92
500_1	4354.11	298	4472.74	245	1563.24	23	1551.67	26	**1117.21**	17	963.89	15.91
500_2	4471.75	299	5005.20	298	1638.64	23	1548.49	23	**1134.93**	17	948.57	19.65
500_3	4508.65	297	4715.44	259	1731.32	24	1344.64	23	**1158.13**	17	980.67	18.10

Table 5 Results of Heu_DT1 [11], Heu_DT2 [21], H_DT [15], M_DT [2] and Heu_2C_DTP on the instances with transmission range 150m

Instance	Heu_DT1 [11]		Heu_DT2 [21]		H_DT [15]		M_DT [2]		Heu_2C_DTP			
	Value	*NDN*	*Value*	*NDN*	*Value*	*NDN*	*Value*	*NDN*	*Value*	*NDN*	*BKS*	*PRD*
50_1	1408.01	26	1766.43	31	1058.69	12	784.03	11	**720.95**	9	647.75	11.30
50_2	1309.87	27	1728.53	33	1061.02	12	1047.55	15	**947.30**	10	863.69	9.68
50_3	1389.86	28	1887.77	32	1104.85	14	1010.84	12	**811.98**	9	743.94	9.15
100_1	1737.47	50	2219.63	53	1420.84	14	1278.74	17	**913.06**	10	876.69	4.15
100_2	2014.17	63	2276.44	57	1009.99	13	964.16	13	**824.09**	9	657.35	25.37
100_3	2148.23	58	2331.72	55	1124.55	13	1184.35	15	**791.40**	9	722.87	9.48
200_1	2530.52	109	2911.73	95	1319.86	15	1286.15	16	**892.39**	11	809.90	10.19
200_2	2703.77	114	3327.77	112	1300.02	19	1203.19	15	**864.19**	11	736.23	17.38
200_3	2561.99	109	3112.19	108	1258.29	16	1224.62	17	**870.18**	11	792.71	9.77
300_1	2908.12	153	3344.34	143	1170.23	16	1144.65	15	**952.53**	11	796.15	19.64
300_2	3493.12	180	3740.08	160	1324.59	19	1224.33	13	**869.64**	10	741.02	17.36
300_3	3589.75	183	4016.39	151	1382.82	18	1195.94	13	**1043.64**	13	819.76	27.31
400_1	3790.75	231	3704.20	187	1295.98	15	1166.44	17	**967.20**	12	795.53	21.58
400_2	4017.53	243	4350.94	218	1174.12	13	1171.00	17	**835.66**	11	779.63	7.19
400_3	4006.47	238	4304.85	197	1335.58	17	1272.84	17	**889.97**	10	814.14	9.31
500_1	4253.35	288	4540.70	249	1252.10	15	1089.90	18	**879.57**	11	792.21	11.03
500_2	4379.35	296	5269.14	303	1286.67	15	1279.42	17	**923.77**	12	779.35	18.53
500_3	4618.27	300	4767.25	250	1474.32	17	1300.60	16	**953.13**	12	808.50	17.89

Table 6 Results of O_ABC^{DT} [15], ACO_DT [15], EA/G [2], SSGA [13], VNS [3] and ABC_DTP on the instances with transmission range 100m

Instance	O_ABC^{DT} [15]					ACO_DT [15]					EA/G [2]					SSGA [13]					VNS [3]					ABC_DTP				
	Best	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*
50_1	**1204.41**	**1204.41**	0.00	19.00	25.57	**1204.41**	**1204.41**	0.00	19.00	2.41	**1204.41**	**1204.41**	0.00	19.00	6.75	**1204.41**	1204.43	0.03	18.75	0.73	**1204.41**	1247.10	2.57	20.85	1.93	**1204.41**	**1204.41**	0.00	19.00	0.44
50_2	**1340.44**	**1340.44**	0.00	21.00	21.46	**1340.44**	**1340.44**	0.00	21.00	4.18	**1340.44**	**1340.44**	0.00	21.00	8.13	**1340.44**	**1340.44**	0.00	21.00	1.31	**1340.44**	1358.73	1.56	23.25	2.86	**1340.44**	1340.69	1.07	21.00	0.64
50_3	**1316.39**	**1316.39**	0.00	19.00	22.99	**1316.39**	**1316.39**	0.00	19.00	2.50	**1316.39**	**1316.39**	0.00	19.00	6.82	**1316.39**	1317.10	0.30	18.15	0.73	**1316.39**	1334.81	1.25	21.00	2.01	**1316.39**	**1316.39**	0.00	19.00	0.56
100_1	**1217.47**	1218.15	0.69	18.45	28.64	**1217.47**	**1217.47**	0.00	19.00	12.71	**1217.47**	1217.61	0.43	18.90	11.06	**1217.47**	1217.98	0.27	18.00	1.74	1222.65	1247.89	1.83	23.90	7.78	**1217.47**	1218.59	0.53	18.00	1.11
100_2	**1128.40**	1128.42	0.09	17.90	27.58	1152.85	1152.85	0.00	17.00	10.86	**1128.40**	1128.54	0.20	17.30	9.93	**1128.40**	1128.79	0.09	16.10	1.56	**1128.40**	1137.19	0.98	16.05	17.21	**1128.40**	1136.50	4.77	16.00	1.12
100_3	**1252.99**	**1253.14**	0.23	19.70	28.39	1253.49	1253.49	0.00	19.00	8.96	1253.49	1257.37	4.29	19.00	11.97	**1252.99**	1253.26	0.25	19.45	1.98	**1252.99**	1257.58	0.65	19.35	16.04	**1252.99**	1253.30	0.26	19.00	1.20
200_1	**1206.79**	1209.52	2.69	18.25	84.10	**1206.79**	1207.61	3.58	18.05	81.13	**1206.79**	1208.26	2.10	18.55	19.76	**1206.79**	**1207.28**	0.23	18.05	4.41	**1206.79**	1209.24	0.45	18.30	34.54	**1206.79**	1210.25	2.32	19.00	4.31
200_2	1216.41	1219.74	2.15	18.90	87.78	**1216.23**	1217.73	2.61	17.65	78.72	1216.41	1222.23	7.67	18.95	21.55	**1216.23**	**1217.63**	3.34	17.15	4.30	**1216.23**	1220.22	0.24	19.05	40.73	1216.41	1219.38	1.57	20.00	4.17
200_3	1253.02	1258.06	3.42	22.15	90.44	**1247.25**	**1248.94**	2.99	20.90	97.93	1247.63	1250.78	2.37	21.80	21.31	**1247.25**	1251.69	2.60	20.90	4.89	**1247.25**	1259.63	0.95	21.05	74.23	1247.73	1252.15	3.52	23.00	4.81
300_1	1229.97	1237.47	2.89	21.75	145.17	1228.24	1243.70	9.71	22.85	352.89	1225.22	1230.48	3.86	21.75	29.57	1226.11	1230.06	4.25	20.15	8.91	1226.11	1236.09	0.65	20.05	229.72	**1215.48**	**1220.39**	3.53	23.00	11.50
300_2	1182.52	1200.79	7.82	19.60	162.59	1176.45	1193.95	10.51	21.10	260.30	**1170.85**	1171.30	0.69	19.00	28.56	**1170.85**	1173.57	7.04	18.80	7.65	**1170.85**	1181.43	0.95	18.70	177.83	**1170.85**	**1171.15**	0.60	19.00	9.73
300_3	1257.21	1271.20	6.74	20.50	145.75	1261.18	1276.75	9.27	24.60	251.91	1252.14	1260.83	5.61	21.80	29.66	**1247.51**	**1253.73**	8.40	20.35	8.35	1262.57	1279.01	0.87	21.15	113.83	1249.54	1254.67	3.91	21.00	11.39
400_1	1223.61	1241.75	7.88	21.90	263.13	1220.62	1237.45	9.50	26.05	600.74	**1211.72**	1220.79	5.31	22.70	40.26	1213.45	1224.87	8.16	21.55	13.93	1211.33	1225.21	0.56	20.45	432.83	1212.51	**1214.36**	2.19	22.00	27.15
400_2	1220.54	1235.29	6.97	22.45	249.39	1209.69	1246.14	21.41	24.40	591.44	1199.92	**1202.82**	1.98	21.30	40.01	1199.67	1205.69	4.79	20.55	13.04	1202.59	1212.25	0.66	20.15	386.49	**1199.23**	1202.90	1.67	20.00	25.44
400_3	1266.41	1276.80	4.59	22.30	216.95	1254.10	1270.34	9.42	25.85	530.58	1248.29	1268.38	8.80	22.70	39.80	1248.29	1265.25	9.75	20.85	12.59	1252.87	1267.72	0.68	21.30	374.35	**1246.94**	**1258.76**	3.84	23.00	24.52
500_1	1233.14	1241.60	4.56	21.40	379.72	1219.66	1240.05	9.17	26.50	1163.20	1206.07	1222.12	11.19	22.30	44.75	1203.64	1215.34	10.93	20.45	18.12	1200.06	1224.82	1.12	20.30	834.19	**1200.06**	**1208.73**	4.18	25.00	54.61
500_2	1245.59	1258.33	5.40	22.35	364.04	1273.86	1295.51	13.39	28.65	1031.81	1226.78	1240.62	6.66	23.50	50.11	1233.89	1247.42	7.13	21.90	22.84	1227.18	1260.31	1.42	20.45	258.57	**1220.68**	**1230.07**	5.18	23.00	54.86
500_3	1249.17	1278.67	11.96	21.60	338.25	1232.71	1259.08	20.03	24.35	917.73	1232.15	1250.48	16.59	21.65	48.09	1231.92	1251.23	16.52	20.50	17.76	**1231.81**	1252.86	1.00	21.10	575.95	1231.95	**1236.33**	3.79	19.00	37.90

Table 7 Results of O_ABC^{DT} [15], ACO_DT [15], EA/G [2] and ABC_DTP on the instances with transmission range 125m

Instance	O_ABC^{DT} [15]					ACO_DT [15]					EA/G [2]					ABC_DTP				
	Best	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*
50_1	**802.95**	**802.95**	0.00	10.00	11.26	802.95	803.26	1.36	10.05	2.16	**802.95**	**802.95**	0.00	10.00	5.01	**802.95**	**802.95**	0.00	10.00	0.24
50_2	**1055.10**	**1055.10**	0.00	12.00	11.50	**1055.10**	**1055.10**	0.00	12.00	2.01	**1055.10**	**1055.10**	0.00	12.00	4.75	**1055.10**	**1055.10**	0.00	12.00	0.32
50_3	**877.77**	**877.77**	0.00	10.00	9.14	**877.77**	**877.77**	0.00	10.00	1.40	**877.77**	**877.77**	0.00	10.00	3.90	**877.77**	877.83	0.26	10.05	0.32
100_1	**943.01**	**943.01**	0.00	12.00	18.12	**943.01**	946.37	1.86	12.75	7.29	**943.01**	**943.01**	0.00	12.00	7.56	**943.01**	**943.01**	0.00	12.00	0.77
100_2	**917.00**	**917.03**	0.09	13.00	19.34	935.71	938.71	1.26	12.00	6.81	917.95	917.95	0.00	12.00	7.13	**917.00**	917.38	0.47	12.60	0.71
100_3	**998.18**	998.82	1.18	14.00	18.09	**998.18**	1006.11	7.57	13.80	7.61	**998.18**	**998.18**	0.00	14.00	8.27	**998.18**	999.91	2.63	14.85	1.06
200_1	**910.17**	911.61	0.72	14.00	57.35	**910.17**	910.50	0.79	14.15	43.25	**910.17**	**910.17**	0.00	14.00	14.73	**910.17**	911.66	0.58	14.00	2.65
200_2	**921.76**	**922.62**	1.05	12.65	54.85	928.84	942.72	5.20	13.15	39.29	**921.76**	923.03	1.40	12.55	14.29	**921.76**	925.38	1.88	12.50	2.62
200_3	942.32	944.93	2.14	14.15	57.36	951.36	959.63	6.83	13.10	41.12	**939.58**	949.18	3.68	13.00	15.79	**939.58**	**943.20**	2.73	13.75	2.68
300_1	981.31	984.63	1.67	14.30	118.68	978.91	980.11	1.11	16.65	157.13	**977.65**	**981.04**	1.35	15.95	22.88	979.81	981.85	1.27	15.40	8.46
300_2	917.31	926.87	4.14	14.35	117.74	918.40	949.05	11.73	14.55	111.61	**913.01**	914.08	2.14	14.80	21.40	**913.01**	**913.88**	1.40	14.90	7.75
300_3	974.98	979.95	2.28	13.95	117.45	981.15	981.33	0.14	13.80	121.29	974.85	979.34	2.22	13.65	21.57	**974.78**	**978.35**	2.54	13.90	10.15
400_1	967.34	971.07	2.63	13.25	221.76	968.66	980.60	15.64	15.70	323.06	**965.99**	**966.59**	0.39	13.00	28.75	**965.99**	966.71	0.31	13.00	19.99
400_2	947.57	952.49	2.77	13.80	200.00	941.52	961.71	21.90	14.70	250.91	**941.02**	943.53	3.05	14.55	29.49	**941.02**	**942.59**	0.90	15.20	19.82
400_3	1003.24	1007.05	2.59	14.30	200.38	**1002.61**	1009.07	8.13	13.90	236.62	1002.97	1003.62	0.22	14.90	28.21	**1002.61**	**1003.33**	0.54	13.65	24.32
500_1	967.32	975.25	3.85	13.80	358.22	986.49	991.85	3.33	17.10	513.13	**963.89**	**963.89**	0.00	14.00	36.20	**963.89**	964.80	0.35	14.00	43.66
500_2	954.89	965.45	5.96	14.50	314.63	953.77	996.85	14.55	13.95	406.90	**948.57**	952.96	4.11	15.15	49.80	948.96	**950.12**	0.71	16.10	44.55
500_3	992.30	1001.21	4.39	16.15	342.23	1006.23	1007.36	1.53	15.85	456.67	**980.67**	992.64	7.19	15.90	45.36	981.90	**986.01**	2.95	17.55	44.39

Table 8 Results of O_ABC^{DT} [15], ACO_DT [15], EA/G [2] and ABC_DTP on the instances with transmission range 150m

Instance	O_ABC^{DT} [15]					ACO_DT [15]					EA/G [2]					ABC_DTP				
	Best	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*	*Best*	*Avg*	*SD*	*ANDN*	*ATET*
50_1	**647.75**	**647.75**	0.00	7.00	6.99	**647.75**	**647.75**	0.00	7.00	1.02	**647.75**	**647.75**	0.00	7.00	3.47	**647.75**	**647.75**	0.00	7.00	0.19
50_2	**863.69**	**863.69**	0.00	11.00	10.05	**863.69**	**863.69**	0.00	11.00	1.70	**863.69**	**863.69**	0.00	11.00	4.17	**863.69**	864.04	0.70	11.00	0.27
50_3	**743.94**	**743.94**	0.00	8.00	7.38	**743.94**	**743.94**	0.00	8.00	1.21	**743.94**	**743.94**	0.00	8.00	3.62	**743.94**	745.68	1.36	8.05	0.18
100_1	**876.69**	876.85	0.39	10.00	16.31	881.37	885.36	5.43	10.65	7.00	**876.69**	**876.69**	0.00	10.00	6.61	**876.69**	877.02	0.50	10.00	0.57
100_2	**657.35**	**657.35**	0.00	8.00	14.55	**657.35**	**657.35**	0.00	8.00	4.23	**657.35**	657.53	0.78	8.00	5.78	**657.35**	657.53	0.78	8.00	0.66
100_3	**722.87**	**722.87**	0.00	9.00	15.75	**722.87**	**722.87**	0.00	9.00	4.80	**722.87**	**722.87**	0.00	9.00	5.98	**722.87**	**722.87**	0.00	9.00	0.54
200_1	**809.90**	**809.90**	0.00	11.00	62.64	**809.90**	810.87	0.19	10.20	35.80	**809.90**	810.49	1.89	10.85	12.71	**809.90**	**809.90**	0.00	10.75	2.41
200_2	**736.23**	736.27	0.17	8.20	59.44	**736.23**	**736.23**	0.00	8.40	28.06	**736.23**	**736.23**	0.00	8.00	11.57	**736.23**	**736.23**	0.00	8.00	2.92
200_3	792.73	797.00	1.64	9.75	62.97	**792.71**	793.73	1.21	9.45	32.15	**792.71**	795.65	1.64	9.15	12.03	**792.71**	**793.48**	0.71	9.65	2.28
300_1	796.70	797.94	1.37	10.30	154.67	796.70	797.17	1.39	10.00	108.25	**796.15**	798.12	3.00	10.50	20.09	796.29	**796.99**	1.00	10.15	9.20
300_2	**741.02**	743.20	0.82	10.05	133.79	748.94	752.33	3.51	10.35	76.95	**741.02**	743.05	1.34	9.85	17.34	**741.02**	**742.88**	1.10	10.10	7.25
300_3	**819.76**	823.76	2.46	10.10	140.01	826.48	826.56	0.13	10.85	94.81	**819.76**	821.67	1.75	10.00	18.60	**819.76**	**820.45**	1.19	10.15	8.74
400_1	796.70	801.57	2.77	9.90	314.00	796.70	798.24	1.38	10.50	197.40	**795.53**	798.82	2.80	10.30	25.71	**795.53**	**797.92**	1.75	10.25	23.68
400_2	781.20	782.28	0.76	9.30	263.35	782.91	787.66	6.62	10.25	175.62	**779.63**	783.14	2.38	9.25	23.60	**779.63**	**781.40**	1.65	9.35	23.06
400_3	816.53	822.64	2.04	10.25	274.13	826.48	831.32	3.26	10.45	159.38	**814.14**	817.38	3.55	10.40	25.38	**814.14**	**815.35**	2.09	10.05	23.33
500_1	796.50	800.25	2.11	10.30	490.85	794.47	797.13	2.40	10.45	338.07	**792.21**	**793.59**	1.79	10.90	30.52	793.98	796.16	1.03	10.40	46.86
500_2	**779.35**	785.10	3.03	9.45	430.80	**779.35**	791.20	8.35	11.10	293.95	**779.35**	781.28	2.55	9.20	32.10	**779.35**	**780.04**	0.73	9.05	46.70
500_3	809.65	811.08	1.03	10.25	542.44	**808.50**	811.35	3.30	11.55	290.10	**808.50**	810.27	0.76	10.55	29.65	**808.50**	**808.50**	0.00	12.05	45.05

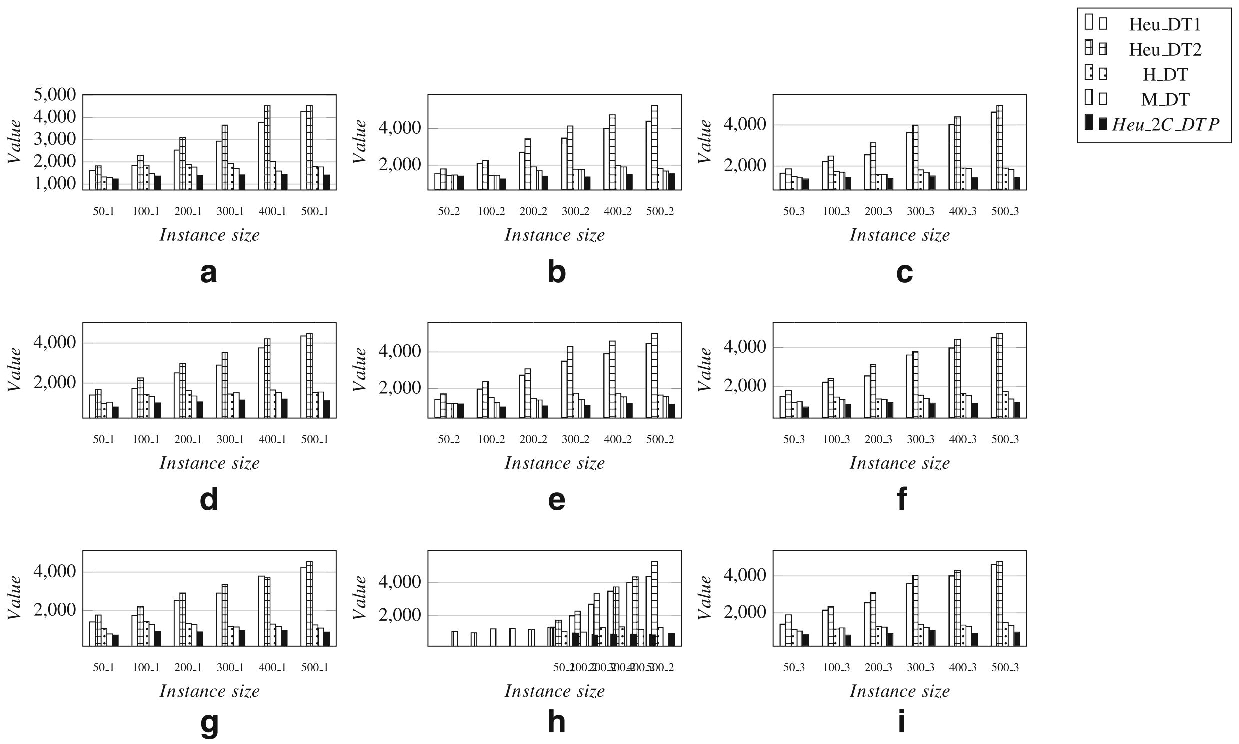

Fig. 3 Comparison of *Value* obtained by various heuristics for different transmission ranges (**a–c**) for 100m range; (**d–f**) for 125m range; and (**g–i**) for 150m range

EA/G, SSGA and VNS on instances with the transmission range 100m. Results of O_ABC^{DT}, ACO_DT, EA/G, SSGA and VNS reported in Table 6 are taken from their respective papers. Tables 7–8 report the results of ABC_DTP along with the results of O_ABC^{DT}, ACO_DT and EA/G on instances with transmission range 125m and

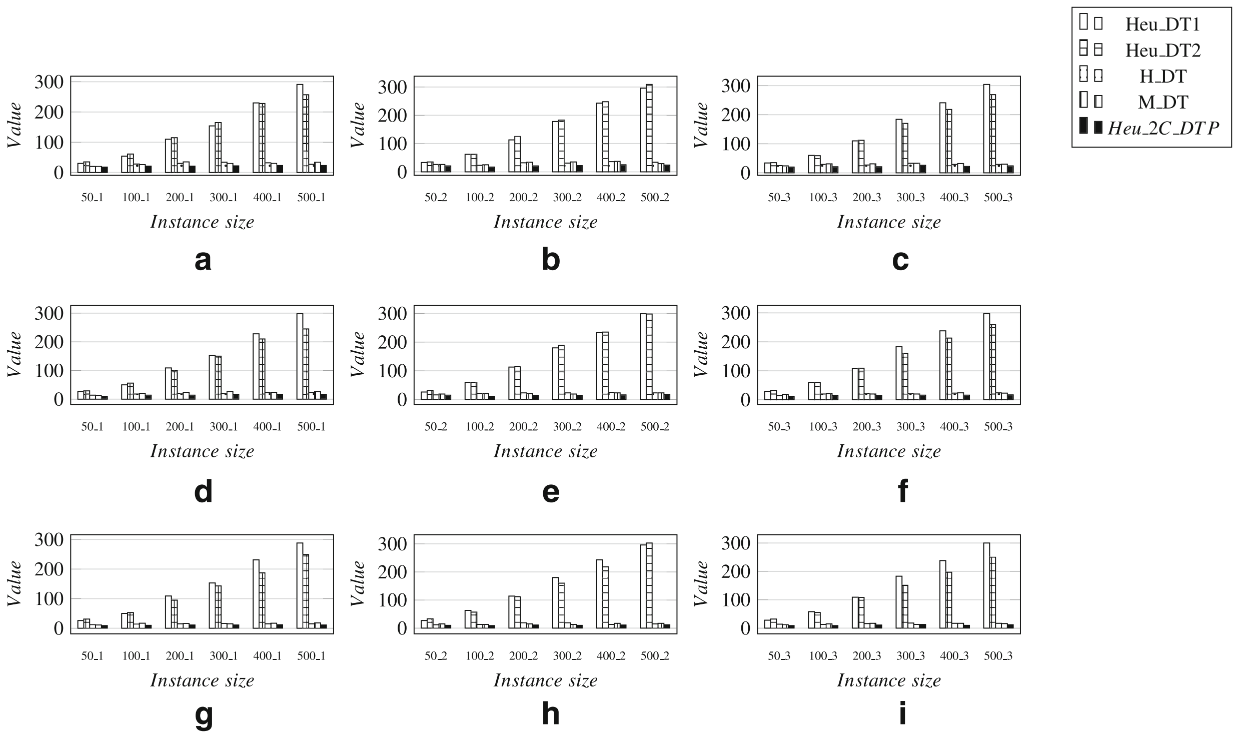

Fig. 4 Comparison of *NDN* obtained by various heuristics for different transmission ranges (**a–c**) for 100m range; (**d–f**) for 125m range; and (**g–i**) for 150m range

150m respectively. Results of O_ABC^{DT}, ACO_DT and EA/G reported in Tables 7 and 8 are taken from [2]. For each instance in Tables 6–8, column *Instance* denotes the name of instance; and for each metaheuristic technique, columns *Best*, *Avg*, *SD*, *ANDN* and *TET* respectively denote the best value, the average solution quality, standard deviation, the average number of dominating vertices, and the average total execution time obtained over 20 runs. For each instance, the best value *Best* and the best average solution quality *Avg* among O_ABC^{DT}, ACO_DT, EA/G, SSGA, VNS and ABC_DTP are highlighted in bold.

Table 6 that reports the results of O_ABC^{DT}, ACO_DT, EA/G, SSGA, VNS and ABC_DTP on 18 instances with the transmission range 100m shows the effectiveness of ABC_DTP in comparison to all other approaches. Comparing with O_ABC^{DT}, ABC_DTP, in terms of solution quality (*Best*), is better on 10 and equals on 8; ABC_DTP, in terms of average solution quality (*Avg*), is better on 11, equals on 2 and is worse on 5; ABC_DTP, in terms of average number of dominating nodes (*ANDN*), is better on 6, equals on 3 and is worse on 9. Comparing with ACO_DT, ABC_DTP, in terms of *Best*, is better on 11, equals on 2 and is worse on 5; ABC_DTP, in terms of *Avg*, is better on 11, equals on 2 and is worse on 5; ABC_DTP, in terms of *ANDN*, is better on 10, equals on 4 and is worse on 4. Comparing with EA/G, ABC_DTP, in terms of *Best*, is better on 8, equals on 8 and is worse on 2; ABC_DTP, in terms of *Avg*, is better on 10, equals on 2 and is worse on 6; ABC_DTP, in terms of *ANDN*, is better on 7, equals on 5 and is worse on 6. Comparing with SSGA, ABC_DTP, in terms of *Best*, is better on 6, equals on 8 and is worse on 4; ABC_DTP, in terms of *Avg*, is better on 10 and is worse on 8; ABC_DTP, in terms of *ANDN*, is better on 4, equals on 2 and is worse on 12. Comparing with VNS, ABC_DTP, in terms of *Best*, is better on 7, equals on 8 and is worse on 3; ABC_DTP, in terms of *Avg*, is better on 17 and is worse on 1; ABC_DTP, in terms of *ANDN*, is better on 9 and is worse on 9.

Table 7 that reports the results of O_ABC^{DT}, ACO_DT, EA/G and ABC_DTP on 18 instances with the transmission range 125m shows the effectiveness of ABC_DTP in comparison to all other approaches. Comparing with O_ABC^{DT}, ABC_DTP, in terms of solution quality (*Best*), is better on 10 and equals on 8; ABC_DTP, in terms of average solution quality (*Avg*), is better on 10, equals on 3 and is worse on 5; ABC_DTP, in terms of average number of dominating nodes (*ANDN*), is better on 6, equals on 4 and is worse on 8. Comparing with ACO_DT, ABC_DTP, in terms of *Best*, is better on 10, equals on 7 and is worse on 1; ABC_DTP, in terms of *Avg*, is better on 13, equals on 1 and is worse on 4; ABC_DTP, in terms of *ANDN*, is better on 8, equals on 1 and is worse on 9. Comparing with EA/G, ABC_DTP, in terms of *Best*, is better on 3, equals on 12 and is worse on 3;

Table 9 Comparison of ABC_DTP with O_ABC^{DT}, ACO_DT, EA/G, SSGA and VNS approaches

Comparision	Transmission Range 100m										Transmission Range 125m						Transmission Range 150m					
	O_ABC^{DT}		ACO_DT		EA/G		SSGA		VNS		O_ABC^{DT}		ACO_DT		EA/G		O_ABC^{DT}		ACO_DT		EA/G	
	Best	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*
better	10	11	11	11	8	10	6	10	7	17	10	10	10	13	3	8	8	12	8	12	0	10
equal	8	2	2	2	8	2	8	0	8	0	8	3	1	4	12	3	10	3	10	3	16	4
worse	0	5	5	5	2	6	4	8	3	1	0	5	7	1	3	7	0	3	0	3	2	4

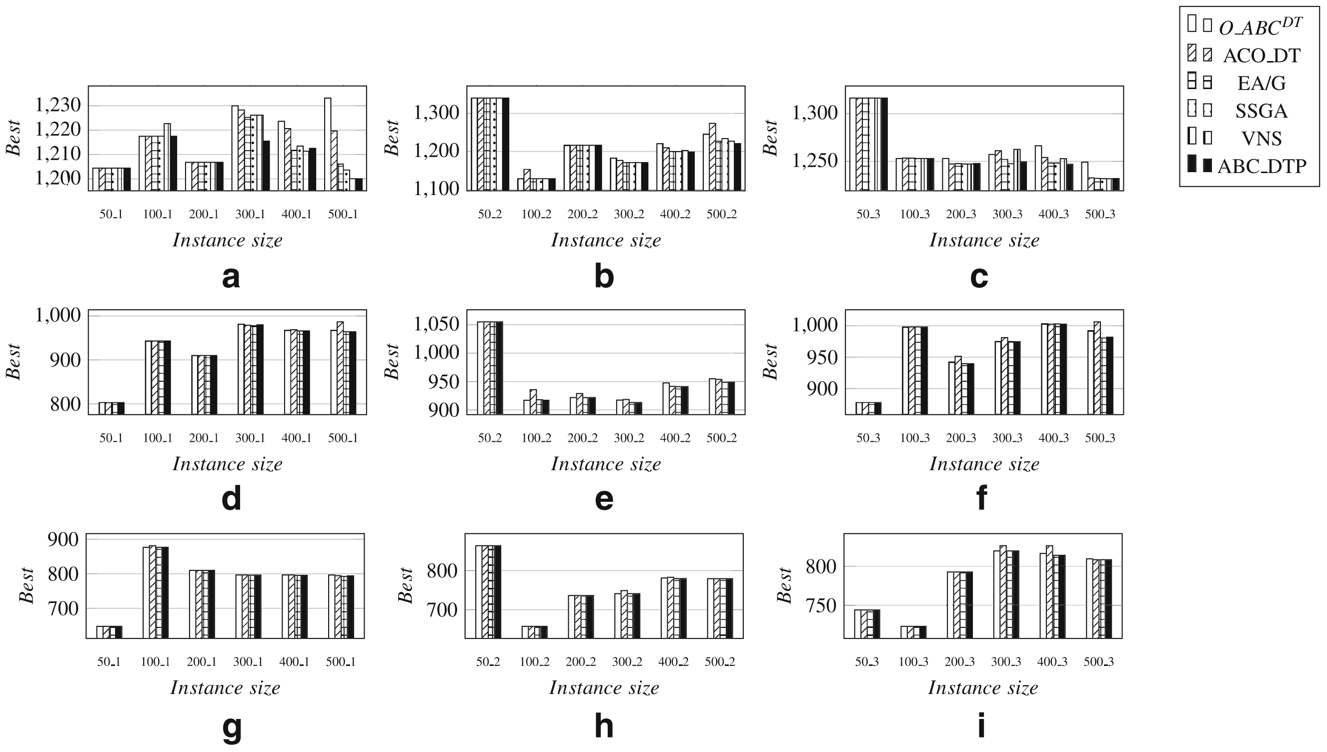

Fig. 5 Comparison of *Value* obtained by various metaheuristics for different transmission ranges (**a–c**) for 100m range; (**d–f**) for 125m range; and (**g–i**) for 150m range

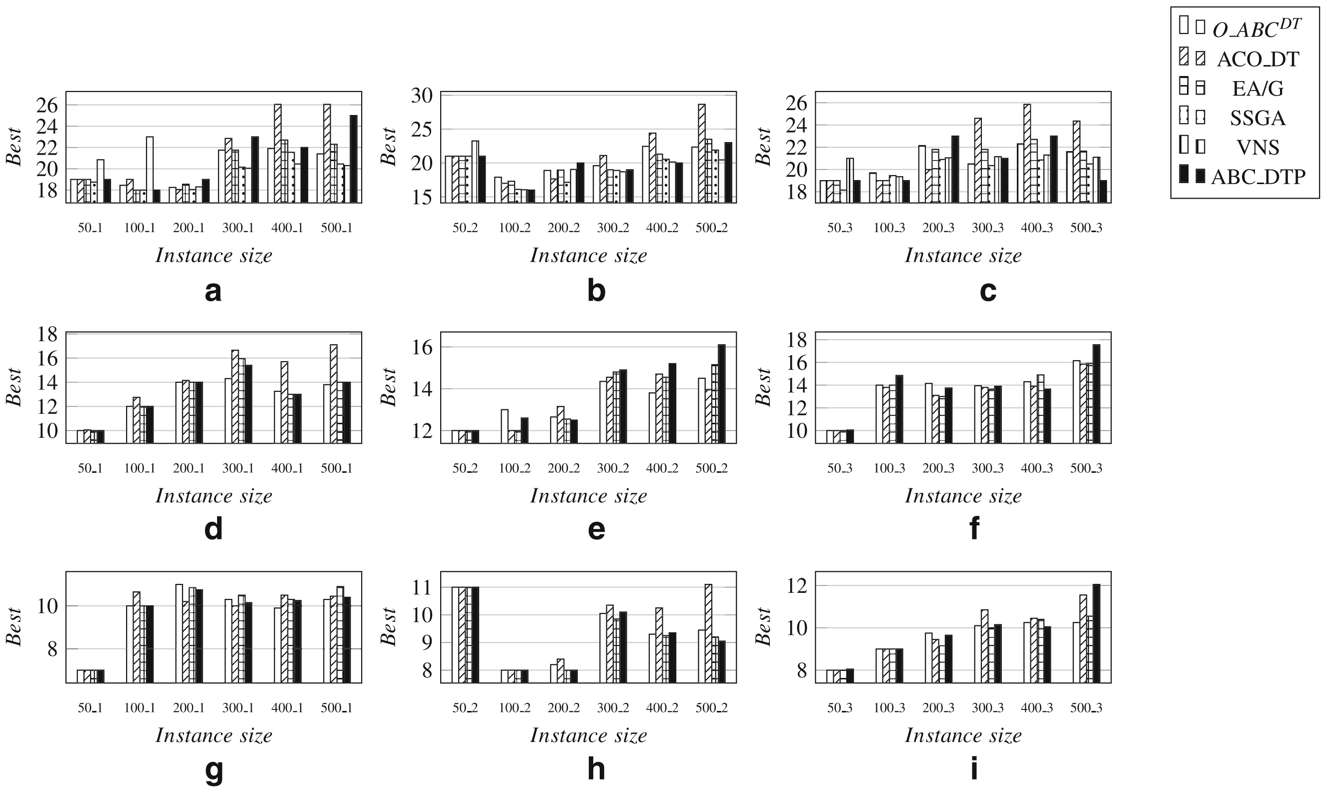

Fig. 6 Comparison of *ANDN* obtained by various metaheuristics for different transmission ranges (**a–c**) for 100m range; (**d–f**) for 125m range; and (**g–i**) for 150m range

ABC_DTP, in terms of *Avg*, is better on 8, equals on 3 and is worse on 7; ABC_DTP, in terms of *ANDN*, is better on 3, equals on 6 and is worse on 9.

Table 8 that reports the results of O_ABC^{DT}, ACO_DT, EA/G and ABC_DTP on 18 instances with the transmission range 150m shows the effectiveness of ABC_DTP in comparison to all other approaches. Comparing with O_ABC^{DT}, ABC_DTP, in terms of solution quality (*Best*), is better on 8 and equals on 10; ABC_DTP, in terms of average solution quality (*Avg*), is better on 12, equals on 3 and is worse on 3; ABC_DTP, in terms of average number of dominating nodes (*ANDN*), is better on 6, equals on 5 and is worse on 7. Comparing with ACO_DT, ABC_DTP, in terms of *Best*, is better on 8 and equals on 10; ABC_DTP, in terms of *Avg*, is better on 12, equals on 3 and is worse on 3; ABC_DTP, in terms of *ANDN*, is better on 10, equals on 4 and is worse on 4. Comparing with EA/G, ABC_DTP, in terms of *Best*, equals on 16 and is worse on 2; ABC_DTP, in terms of *Avg*, is better on 10, equals on 4 and is worse on 4; ABC_DTP, in terms of *ANDN*, is better on 6, equals on 6 and is worse on 6.

5.4 Collective picture

This subsection presents a collective picture with respect to all problem-specific heuristics (Heu_DT1, Heu_DT2, H_DT, M_DT and Heu_2C_DTP) and all metaheuristic techniques (O_ABC^{DT}, ACO_DT, EA/G, SSGA, VNS and ABC_DTP) for the DTP. Clearly, in terms of solution quality (*Value* and *NDN*), Heu_2C_DTP shows superiority over all other existing problem-specific heuristics (Heu_DT1, Heu_DT2, H_DT, and M_DT) on all 54 instances except on one instance, i.e., *300_3* in 150m range whose *NDN* value is similar to that of M_DT. One can observe the performance of Heu_2C_DTP in terms of *Value* and *NDN* in Figs. 3 and 4 respectively.

As far as comparison with all metaheuristic techniques (O_ABC^{DT}, ACO_DT, EA/G, SSGA, VNS and ABC_DTP), ABC_DTP performs better overall in terms of solution quality (*Best* and *Avg*). One can observe solution quality of ABC_DTP (*Best* and *Avg*) in Table 9. ABC_DTP finds new values (*Best*) for 6 instances (*300_1*, *400_2*, *400_3* and *500_2* in 100m range, and *300_3* in 125m range). One can notice in Fig. 5 that in terms of *Best*, ABC_DTP is overall better than all existing metaheuristic techniques for the DTP. ABC_DTP has improved average solution quality (*Avg*) for 23 instances (7 instances in 100m range, 7 instances in 125m range and 9 instances in 150m range) out of 54 instances. In terms of average number of dominating nodes (*ANDN*), ABC_DTP is comparable with all existing metaheuristic techniques for the DTP. This can be observed in Fig. 6.

In terms of computational time, ABC_DTP is many times faster than O_ABC^{DT}, ACO_DT and VNS. Overall, ABC_DTP is faster than EA/G. ABC_DTP is slower than SSGA, but ABC_DTP provides better solution quality (*Best* and *Avg*).

It is to be noted that Tables 3, 4 and 5 report best known solution value (*BKS*) for each instance taken from all approaches for the DTP, i.e., problem-specific heuristics including Heu_2C_DTP and metaheuristic techniques including ABC_DTP. Tables 3, 4 and 5 also report the percentage relative deviation (PRD) of each instance for Heu_2C_DTP. The *PRD* is defined in (1),

$$\left[PRD = \frac{Value^{Heu\_2C\_DTP} - BKS}{BKS} \times 100\% \right] \tag{1}$$

In *equation* (1), $Value^{Heu\_2C\_DTP}$ denotes the value (*Value*) of each instance obtained by Heu_2C_DTP. One can observe from Tables 3, 4 and 5 that the results of Heu_2C_DTP on all instances are nearer to *BKS* whose maximum *PRD* value is 27.31 for *300_3* in 150m and minimum *PRD* value is 0.00 for *50_1* in 125m. It shows that Heu_2C_DTP can be a good choice for obtaining a $dTree$ (solution) to the DTP in a very short time.

6 Conclusions

In this paper, we have proposed a new and effective problem-specific heuristic for the DTP (Heu_2C_DTP) that produces much better results on a set of benchmark instances than existing problem-specific heuristics in the literature. Heu_2C_DTP is so effective that it can be a good choice for obtaining a $dTree$ (solution) to the DTP in a very short time. In addition, we have also proposed an artificial bee colony algorithm for the DTP (ABC_DTP) which is different from the existing ABC algorithm for the DTP in the literature on its two main components: initial solution generation; and determining a neighboring solution. These two components coordinated with other components of ABC_DTP help in making ABC_DTP more effective and robust. ABC_DTP demonstrates the superiority over not only previous ABC algorithm for the DTP, but also other existing metaheuristic techniques in the literature.

In future, similar strategies used in ABC_DTP can also be applied to develop ABC algorithms for other $\mathcal{NP}$-Hard graph problems.

References

1. Arkin E, Halldorssom M, Hassin R (1993) Approximating the tree and tour covers of a graph. Inf Process Lett 47:275–282
2. Chaurasia SN, Singh A (2016) A hybrid heuristic for dominating tree problem. Soft Comput Springer 20:1–21

3. Drazic Z, Cangalovic M, Kovacevic-Vujcic V (2017) A metaheuristic approach to the dominating tree problem. Optim Lett Springer 11:1155–1167
4. Fujito T (2001) On approximability of the independent/connected edge dominating set problems. Inf Process Lett 79:261–266
5. Fujito T (2006) How to trim an MST: A 2-approximation algorithm for minimum cost tree cover. In: Proceedings of the 33rd international colloquium on automata, languages and programming, ICALP, Part I. Venice, July 10-14, 2006, pp 431–442
6. Guha S, Khuller S (1998) Approximation algorithms for connected dominating sets. Algorithmica 20:374–387
7. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Computer Engineering Department Engineering Faculty. Erciyes University, Turkey
8. Karaboga D, Gorkemli B, Ozturk C, Karaboga N (2014) A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intell Rev 42(1):21–57
9. Park M, Wang C, Willson J, Thai M, Wu W, Farago A (2007) A dominating and absorbent set in wireless ad-hoc networks with different transmission range. In: Proceedings of the 8th ACM international symposium on mobile ad hoc networking and computing (MOBIHOC)
10. Prim R (1957) Shortest connection networks and some generalizations. Bell Syst Techn J 36:1389–1401
11. Shin I, Shen Y, Thai M (2010) On approximation of dominating tree in wireless sensor networks. Optim Lett 4:393–403
12. Singh A (2009) An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Appl Soft Comput Elsevier 9:625–631
13. Sundar S (2014) A steady-state genetic algorithm for the dominating tree problem. In: Proceedings of the tenth international conference on simulated evolution and learning (SEAL 2014), LNCS, vol 8886. Springer-Verlag, Dunedin, pp 48–57
14. Sundar S, Singh A (2010) A swarm intelligence approach to the quadratic minimum spanning tree problem. Inf Sci Elsevier 180:3182–3191
15. Sundar S, Singh A (2013) New heuristic approaches for the dominating tree problem. Appl Soft Comput 13:4695–4703
16. Sundar S, Suganthan PN, Jin CT, Xiang CT, Soon CC (2017) A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint. Soft Comput Springer 21:1193–1202
17. Thai M, Wang F, Liu D, Zhu S, Du DZ (2007) Connected dominating sets in wireless networks with different transmission ranges. IEEE Trans Mob Comput 6:721–730
18. Thai M, Tiwari R, Du DZ (2008) On construction of virtual backbone in wireless ad hoc networks with unidirectional links. IEEE Trans Mob Comput 7:1–12
19. Wan P, Alzoubi KM, Frieder O (2002) Distributed construction on connected dominating set in wireless ad hoc networks. In: Proceedings IEEE INFOCOM 2002, the 21st annual joint conference of the IEEE computer and communications societies. New York, June 23-27, 2002, pp 1597–1604
20. Wu J, Li H (1999) On calculating connected dominating set for efficient routing in ad hoc wireless networks. In: Proceedings of the 3rd international workshop on discrete algorithms and methods for mobile computing and communications (DIAL-M 1999). Seattle, Washington, August 20, 1999, pp 7–14
21. Zhang N, Shin I, Li B, Boyaci C, Tiwari R, Thai M (2008) New approximation for minimum-weight routing backbone in wireless sensor network. In: Lecture notes in computer science, vol 5258. Springer-Verlag, Berlin, pp 96–108

Kavita Singh received the Master of Computer Applications degree from Pt. Ravishankar Shukla University Raipur, Raipur, India in 2012. She is currently pursuing Ph.D. from Department of Computer Applications, National Institute of Technology Raipur, Raipur, India. She works mainly in the area of heuristic and metaheuristic techniques for combinatorial optimization problems.

Shyam Sundar received the Master of Computer Applications and Ph.D. in Computer Science degrees from University of Hyderabad, Hyderabad, India in 2008 and 2012 respectively. He spent one year (May 2012–May 2013) as a post doctoral researcher in the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. Currently, he is an Assistant Professor in the Department of Computer Applications at the National Institute of Technology Raipur, Raipur, India. He works mainly in the area of heuristic and metaheuristic techniques for combinatorial optimization problems.