Soft Comput
DOI 10.1007/s00500-014-1513-4

METHODOLOGIES AND APPLICATION

A hybrid heuristic for dominating tree problem

Sachchida Nand Chaurasia · Alok Singh

Abstract Given an undirected, connected, edge-weighted graph, the dominating tree problem (DTP) seeks on this graph a tree of minimum weight such that each node of the graph either belongs to the tree or is adjacent to a node in the tree. This problem is $\mathcal{NP}$-hard. In this paper, we present an evolutionary algorithm with guided mutation (EA/G) to solve the DTP. This problem has several practical applications in the field of wireless sensor networks. EA/G is a recently proposed evolutionary algorithm that tries to overcome the shortcomings of genetic algorithms (GAs) and estimation of distribution algorithms both, and has the characteristics of both. We have compared the performance of our proposed approach with the state-of-the-art approaches presented in the literature. Computational results show the superiority of our approach in terms of solution quality as well as execution time.

Keywords Constrained optimization · Dominating tree · Estimation of distribution algorithm · Guided mutation · Heuristic

1 Introduction

Consider an undirected, connected, edge-weighted graph $G = (V, E)$, where V denotes the set of vertices or nodes and E denotes the set of edges. The dominating tree problem (DTP) is concerned with finding a tree DT of minimum total edge weight on G in such a way that each node $v \in V$ either belongs to DT or is adjacent to a node belonging to DT. Nodes in DT are said to be dominating nodes, whereas nodes which do not belong to DT are said to be non-dominating nodes. DTP is $\mathcal{NP}$-hard in general (Shin et al. 2010; Zhang et al. 2008). However in some special cases DTP can be solved in polynomial time, e.g. cases where the underlying graph is complete or is a tree. In case of a complete graph, each node is a minimum dominating tree in itself with cost 0. In case the underlying graph G is a tree, the minimum dominating tree is the subgraph (subtree) of G induced by non-leaf nodes.

The DTP, a relatively new problem, finds applications in the area of wireless sensor networks (WSNs). One such application of the DTP is to provide a virtual backbone for routing (Wu and Hailan 1999). In this scheme, routing information are stored only on the dominating nodes after computing a DT. Since non-dominating nodes are one hop away from nodes of the DT, in order to forward a message from one node (sender) to another node (receiver), the message can always be first forwarded to the nearest dominating node of the sender, then routed to the nearest dominating node of the receiver with the help of the DT, and finally forwarded to the receiver. Non-dominating nodes only need to know the nearest dominating node. The advantage of this scheme is that the number of dominating nodes is small in comparison to the total nodes (Wu and Hailan 1999), thereby significantly reducing the size of the routing tables. Such a scheme is more resilient to faults also as these tables need to be recalculated only when topological changes in the network affect one of the dominating nodes.

In the literature, the connected dominating set concept has been widely used for constructing a routing backbone in WSNs with minimum energy consumption (Thai et al.

Communicated by V. Loia.

S. N. Chaurasia · A. Singh (✉)
School of Computer and Information Sciences,
University of Hyderabad, Hyderabad 500046, India
e-mail: alokcs@uohyd.ernet.in

S. N. Chaurasia
e-mail: mc10pc13@uohyd.ernet.in

Published online: 13 November 2014

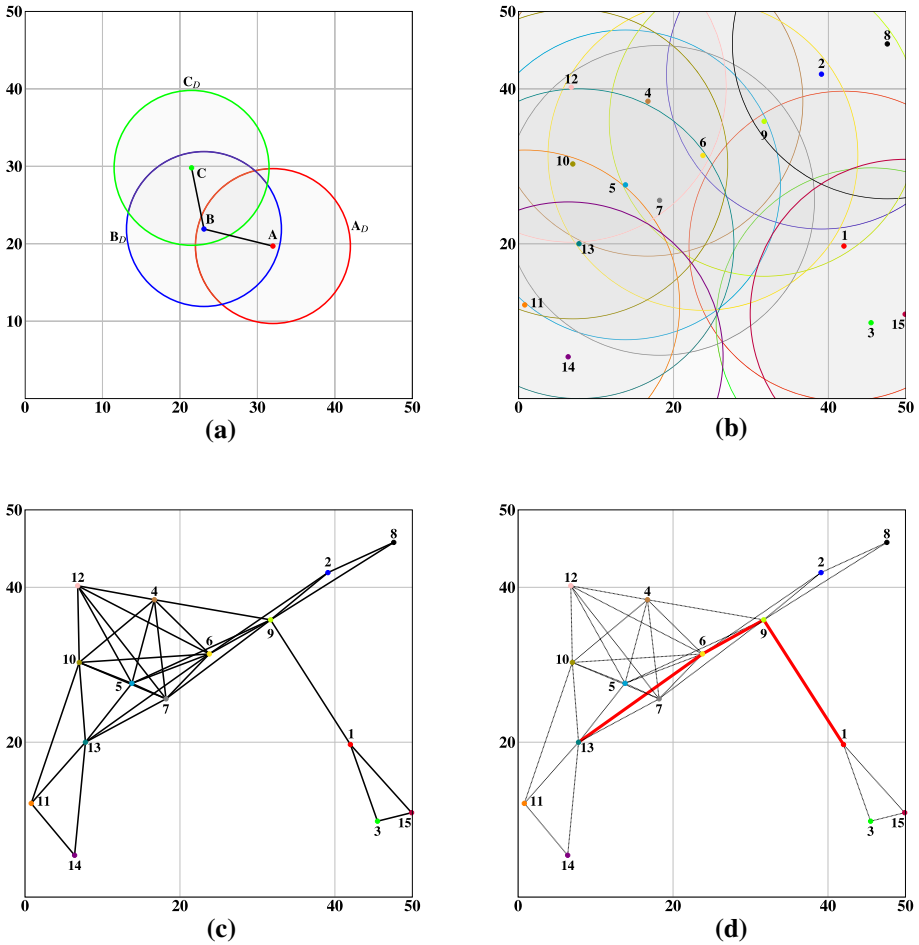

Fig. 1 Illustration of a disk graph and a dominating tree. **a** Edges in a disk graph. **b** A WSN with 15 nodes. **c** Disk graph corresponding to WSN of **b**. **d** A dominating tree on disk graph of **c**

2007; Wan et al. 2002; Guha and Khuller 1998; Park et al. 2007; Thaiand et al. 2008). However, these approaches focus on the nodes instead of the edges in order to minimize the energy consumption. Actually, the energy consumption at each edge directly affects the energy consumption of routing. Therefore, one has to consider the energy consumption by each edge to minimize the energy consumption of routing. With this intention, the DTP was formulated (Shin et al. 2010; Zhang et al. 2008). There exists another related problem called the tree cover problem that has been studied in the literature (Arkin et al. 1993; Fujito 2001, 2006), but this problem is different from the DTP. A tree in the tree cover problem is defined as an edge dominating set, whereas a tree in the DTP is defined as a node dominating set.

Actually, communication in wireless sensor networks can be modelled using disk graphs where each disk around a sensor node represents the transmission range of that node. There exists an edge between a pair of nodes if these two nodes lie in the intersection area of their respective disks. In other words, an edge exists between a pair of nodes only when these two nodes are within the transmission range of each other. Obviously, only those nodes which are connected by an edge can communicate directly with each other. Figure 1a explains this concept where three nodes A, B and C are placed randomly in a 50×50 m area. The transmission range of each node is assumed to be 10 m. A_D, B_D and C_D are the disks associated with nodes A, B and C, respectively. In this figure, for the sake of clarity, each node and its associated disk is represented with the same colour which is different from the colours assigned to other nodes and their associated disks. Nodes A and B lie in the intersection area of disks A_D and B_D. Similarly, nodes B and C lie in the intersection area of disks B_D and C_D. Hence, an edge exist between A and B and another edge exists between B and C. On the other hand, nodes A and C do not lie in the intersection area of their respective disks, viz. A_D and C_D, and hence there exists no edge between A and C. From this figure, it can also be observed that if the radius of the disks increases, the number of edges can increase.

Figure 1b shows a wireless sensor network consisting of 15 nodes, each with a transmission range of 20 m, placed randomly in a 50×50 m area. The disks associated with each nodes are also shown. This figure uses the same colouring scheme as used in Fig. 1a. The corresponding disk graph is shown in the Fig. 1c. A possible dominating tree on this disk graph is shown in Fig. 1d with thick red colour edges.

Shin et al. (2010) and Zhang et al. (2008) both proved the $\mathcal{NP}$-hardness of the DTP, provided the inapproximability results and introduced an approximation framework for solving the DTP. Since approximation algorithm is quasi-polynomial ($|V|^{O(lg|V|)}$), each of them developed a polynomial time heuristic for the DTP. Later, Sundar and Singh (2013) proposed one more heuristic and two metaheuristic techniques, viz. artificial bee colony (ABC) algorithm and ant colony optimization (ACO) algorithm for the DTP. To the best of our knowledge, only these two metaheuristic approaches have been proposed in the literature for the DTP. The heuristic of Sundar and Singh (2013) outperformed the heuristics proposed by Shin et al. (2010) and Zhang et al. (2008).

In this paper, we present a heuristic and an evolutionary algorithm with guided mutation (EA/G) for the DTP. Our heuristic is derived from the heuristic proposed by Sundar and Singh (2013). EA/G is a relatively new evolutionary technique that employs a guided mutation operator to create offsprings (solutions). EA/G was developed by Zhang et al. (2005). The guided mutation operator makes use of global statistical information about the search space and location information of the solutions found so far to generate the offsprings. We have compared our approaches with the previously proposed approaches. Computational results show the effectiveness of our approaches.

The organization of the remaining part of the paper is as follows: Sect. 2 presents the formal problem formulation and introduces the notational conventions used in this paper. Section 3 describes the modifications proposed in the heuristic of Sundar and Singh (2013). Overview of EA/G is provided in Sect. 4, whereas Section 5 describes our EA/G approach for the dominating tree problem. Computational results are presented in Sect. 6. Finally, Sect. 7 presents some concluding remarks and directions for future research.

2 Problem formulation

Let $G = (V, E)$ be an undirected connected graph, where V is the set of vertices or nodes and E is the set of edges. Two nodes u and v are called neighbors of each other or adjacent to each other, iff, there exists an edge between them, i.e., $(u, v) \in E$. Similarly, two edges $e_{i,j}$ and $e_{k,l}$ are called neighbors of each other or adjacent to each other, iff, they have a node in common. Given a non-negative weight function $w : E \rightarrow \Re^+$ associated with the edges of G, the dominating tree problem (DTP) seeks on G a tree DT such that for each node $v \in V$, v is either in DT or adjacent to a node in DT and has minimum total edge weight among all such trees, i.e., $\sum_{e_{i,j} \in DT} w(e_{i,j})$ is minimum. Nodes in DT are called dominating nodes, whereas nodes not in DT are called non-dominating or dominatee nodes. In this paper, we will also call any edge belonging to DT dominating edge, and, any edge not in DT non-dominating or dominatee edge.

Table 1 Notational convention

Notation	Definition
$e_{u,v}$	Edge between nodes u and v, i.e., $(u, v) \in E$
$w(e_{u,v})$	Weight of the edge $e_{u,v}$
$ON(v) \subseteq V$	$\{u : u \in V$ and $(u, v) \in E\}$ is called open neighborhood of node $v \in V$
$CN(v) \subseteq V$	$ON(v) \cup \{v\}$ is called closed neighborhood of node $v \in V$.
$wd(v) \subseteq CN(v)$	Set of WHITE nodes in the closed neighborhood of node $v \in V$
$c(i, j)$	has value 1 if at least one of i and j have colour WHITE, 0 otherwise

Throughout this paper, while constructing a dominating tree, we will follow the convention that a node which is neither in the tree nor adjacent to a node in the tree is assumed to have colour WHITE, a node which does not belong to the tree, but is adjacent to a node in the tree is assumed to have colour GREY, and a node belonging to the tree is assumed to have colour BLACK. Initially, all nodes are assumed to have colour WHITE. When construction of dominating tree is complete then nodes belonging to the tree will have BLACK colour and all other (non-dominating) nodes will have GREY colour.

Important notational conventions used throughout this paper are given in the Table 1. Additional notational conventions will be introduced wherever those will be used.

3 Heuristic

This section describes our heuristic, which is an improved version of the heuristic H_DT proposed in Sundar and Singh (2013). We will refer to our heuristic as M_DT hereafter. Similar to the H_DT, the M_DT consists of two phases. The first phase is the initialization phase in which the shortest path between all pairs of nodes in graph G are computed. The second phase consists of an iterative procedure to construct a dominating tree. At the beginning of the second phase of the M_DT, all nodes are assumed to have colour WHITE, and, we start with an empty tree DT. During each iteration, an edge $e_{i,j}$ is selected using the following expression:

$$e_{i,j} \leftarrow \arg\max_{e_{u,v} \in E} \frac{W(e_{u,v}) \times nc(e_{u,v})}{w(e_{u,v})} \tag{1}$$

where $W(e_{u,v})$ is $(\sum_{x \in ON(u)} w(e_{u,x}) \times c(u, x) + \sum_{y \in ON(v)} w(e_{v,y}) \times c(v, y) - w(e_{u,v}) \times c(u, v))$, i.e., $W(e_{u,v})$ is the sum of the weights of all those edges which can potentially be

avoided in DT in case the edge $e_{u,v}$ is selected and $nc(e_{u,v})$ is $|wd(u) \cup wd(v)|$, i.e., $nc(e_{u,v})$ gives the number of white nodes in the closed neighborhood of nodes u and v. Therefore, Expression 1 selects an edge considering not only its own characteristics but also the characteristics of its adjacent edges. The characteristics that are considered are the weight of an edge and the colour of its end points. Further processing in the iteration depends on the colour of the nodes i and j. Depending on the colour of the nodes i and j, following cases can occur:

Case A : *If both the nodes* i and j *are WHITE.* A shortest path SP between nodes $\{i, j\}$ and the partially constructed dominating tree DT is searched in G. If two or more than two shortest paths exist then the tie is broken by selecting a path which has the maximum number of WHITE nodes. If a tie occurs in case of the number of WHITE nodes on the paths as well, then arbitrarily one such path is selected. All edges belonging to SP are added to DT and all nodes belonging to SP are recoloured BLACK (if not already) and all WHITE neighbors of such nodes are recoloured GREY. The edge $e_{i,j}$ will be added to DT only if one of the nodes among i and j which does not lie on the shortest path SP has at least one WHITE neighbor. Otherwise, there is no point in adding the edge $e_{i,j}$ as both its endpoint nodes are non WHITE after adding the nodes of SP to DT. If the edge $e_{i,j}$ got added to DT then nodes i and j are recoloured BLACK (if not already) and all their WHITE neighbors are recoloured GREY.

Case B : *If one node is WHITE and the other is GREY.* Check which one is WHITE node. Suppose node j is WHITE. Now, find a shortest path SP between partially constructed dominating tree DT and node j (ties are broken in the same manner as previous case). Let k be the node which lies one hop away from node j on SP. All edges belonging to SP except $e_{k,j}$ are added to DT and all nodes belonging to SP except j are recoloured BLACK (if not already) and all WHITE neighbors of such nodes are recoloured GREY. Now check whether node j has any WHITE neighboring nodes, if yes, then add the edge $e_{k,j}$ into DT and recolour node j BLACK and also recolour GREY, all the WHITE neighboring nodes of node j.

Case C : *If both the nodes* i and j *are GREY.*

C1 : *If both the nodes have WHITE neighboring nodes.* Find, which one among i and j has shortest path SP from DT (ties are broken arbitrarily). Suppose SP connects i to DT. Add all the edges on the shortest path SP into DT and recolour all the nodes lying on SP BLACK and all the WHITE neighboring nodes of such nodes GREY. Again recheck that node j still has WHITE neighboring nodes, if yes, then proceed as in case C2.

C2 : *If only one has WHITE neighboring nodes.* Suppose node j has WHITE neighboring nodes. Now, find the shortest path SP between partially constructed dominating tree DT and node j. Add all the edges on the shortest path SP into DT and recolour all the nodes lying on SP BLACK and all the WHITE neighboring nodes of such nodes GREY.

Case D : *If one node is BLACK and the other is GREY with at least one WHITE neighbor.* Add the edge $e_{i,j}$ into partially constructed dominating tree DT and recolour BLACK the node whose colour is GREY and also recolour GREY all its WHITE neighboring nodes.

After this another iteration begins. This process continues till the construction of the dominating tree is complete, i.e., till no WHITE node remains.

After the completion of the second phase of the heuristic, all nodes in DT are reconnected by computing a minimum spanning tree (MST) (Prim 1957) on the subgraph of G induced by these nodes, thereby possibly reducing the cost further as MST is a spanning tree of least cost among all spanning trees over a graph with given set of nodes. After computing MST, a pruning operator is called to remove all the redundant nodes from DT. A redundant node is a node such that if we remove that node from DT, DT still satisfy the property of a dominating tree. Detail of pruning operator can be found in Sect. 3.2.

The following points highlight the differences between the H_DT of Sundar and Singh (2013) and our heuristic M_DT.

1. The determination of next edge $e_{i,j}$ to be added into DT differs for the H_DT and the M_DT. In the H_DT, next edge to be added is an edge whose edge weight is least among all available edges, whereas in M_DT, next edge is selected with the help of Expression 1.
2. In the heuristic H_DT, the edge $e_{i,j}$ is always included into DT, but in case of heuristic M_DT, the edge $e_{i,j}$ will be added to DT only when absolutely necessary as explained already.
3. The heuristic H_DT applies two times pruning and two times reconnection by computing a minimum spanning tree (MST) in the following order: pruning $\rightarrow$ MST $\rightarrow$ pruning $\rightarrow$ MST, whereas in our heuristic M_DT, we applied only once the pruning and MST in the order of MST followed by pruning. Actually, if we apply MST first then we may get a dominating tree of lesser cost

in comparison to applying the pruning first because we may lose some nodes in pruning which are vital for reducing the cost of the dominating tree. Empirical observations also favoured this strategy. It is computationally less expensive, as well.

Algorithm 1 provides the pseudo-code of M_DT.

Algorithm 1: The pseudo-code for heuristic M_DT

```
//Initially all nodes in V are coloured WHITE
W_n ← V;   DT ← ∅;   E_r ← E;   nd ← ∅;
Compute shortest path between all pairs of nodes in G;
e_{i,j} ← arg max_{e_{k,l}∈E_r} W(e_{k,l})×nc(e_{k,l}) / w(e_{k,l});
nd ← {p : p ∈ ON(i) ∪ ON(j) ∩ W_n};
make i and j BLACK;
make all nodes ∈ nd GREY;
W_n ← W_n\(nd ∪ {i, j});
DT ← DT ∪ {e_{i,j}};
while W_n ≠ ∅ do
    e_{i,j} ← arg max_{e_{k,l}∈E_r} W(e_{k,l})×nc(e_{k,l}) / w(e_{k,l});
    if Both the nodes i and j are WHITE then
        Apply Case A;
    else if One is WHITE and the other is GREY then
        Apply Case B;
    else if Both the nodes i and j are GREY then
        Apply Case C;
    else if One is BLACK and the other is GREY then
        Apply Case D;
    Remove all BLACK and GREY nodes from W_n;
Reconnect nodes in DT via a minimum spanning tree;
Apply Pruning operator on nodes in DT;
return DT;
```

3.1 Illustrating H_DT and M_DT with an example

With the help of the Fig. 2, we demonstrate the process of construction of a dominating tree using the heuristic H_DT of Sundar and Singh (2013). In the Fig. 2a, initially all the nodes are assumed to have colour WHITE. The edge $e_{1,5}$, which has least weight among all the edges, is selected and added into dominating tree DT. Now, the nodes {1, 5} are recoloured BLACK and also their neighboring nodes {0, 2, 4} are recoloured GREY. This situation is shown in the Fig. 2b. In the next iteration, the edge $e_{9,10}$ is added into DT. As a result, the nodes {9, 10} are recoloured BLACK and their WHITE neighboring nodes {6, 12, 13} are recoloured GREY. The two sub-trees $\{e_{1,5}\}$ and $\{e_{9,10}\}$ are connected via the shortest path between nodes 5 and 9. The node 4 which is on this shortest path is recoloured BLACK and the edges $\{e_{5,4}, e_{4,9}\}$ are included into DT. This situation is depicted in the Fig. 2c where the shortest path between two sub-trees is shown with thick and dotted line. Now, the edge $e_{7,11}$ is selected and included into DT which leads to the recolouring of the nodes {7, 11} with BLACK colour and their WHITE neighboring nodes {8, 14} with GREY colour. The two resulting sub-trees, viz. $\{e_{1,5}, e_{5,4}, e_{4,9}, e_{9,10}\}$ and $\{e_{7,11}\}$ are connected via the shortest path between nodes 10 and 11. The node 6 which is on this shortest path is recoloured BLACK and the edges $e_{11,6}$ and $e_{6,10}$ are added into DT. This situation is shown in the Fig. 2d. In the last iteration, the edge $e_{2,3}$ is selected and included into DT and the nodes {2, 3} are recoloured BLACK. The edge $e_{2,6}$ which constitutes the shortest path between the two sub-trees $\{e_{1,5}, e_{5,4}, e_{4,9}, e_{9,10}, e_{10,6}, e_{6,11}, e_{11,7}\}$ and $\{e_{2,3}\}$ is included into DT yielding a total weight of 36 for DT (Fig. 2e). Now, no WHITE node remains, and as a result, the second phase of heuristic H_DT stops. Hereafter, a *pruning procedure* (Sect. 3.2) is applied to remove all the redundant edges as well as the redundant nodes. In the dominating tree DT $\{e_{1,5}, e_{5,4}, e_{4,9}, e_{9,10}, e_{10,6}, e_{6,11}, e_{11,7}, e_{2,6}, e_{2,3}\}$, the nodes {3, 7} are redundant. After the removal of these two redundant nodes and their corresponding redundant edges $\{e_{2,3}, e_{11,7}\}$, DT becomes $\{e_{1,5}, e_{5,4}, e_{4,9}, e_{9,10}, e_{10,6}, e_{6,11}, e_{6,2}\}$ with total weight 25. After the *pruning procedure* , a minimum spanning tree (MST) is constructed on the subgraph induced by the set of nodes in DT to explore the possibility of reconnecting these nodes via this MST in case it leads to reduction in cost. In this example, nodes in DT are already connected via a MST, so MST procedure fails to reduce the cost of DT any further. Once again the *pruning procedure* is applied, but in vain as there are no redundant nodes. Finally, MST procedure is also applied unsuccessfully on the nodes of DT and then the heuristic H_DT stops. The final dominating tree with weight 25 is shown in the Fig. 2f.

To illustrate the M_DT, the same input graph with 15 nodes as used for the H_DT is taken. Initially, all nodes are assumed to have colour WHITE as shown in the Fig. 3a. At the first iteration, the edge $e_{7,11}$ is selected by the Expression 1 as this edge has the maximum ratio. Now, the edge $e_{7,11}$ is included into empty dominating tree DT. The nodes {7, 11} are coloured BLACK and also all the WHITE neighboring nodes {2, 6, 8, 13, 14} of the nodes {7, 11} are coloured GREY. This situation is shown in the Fig. 3b. In the next iteration, Expression 1 returns the edge $e_{9,10}$. Here both the nodes, viz. 9 and 10 are WHITE, and therefore, the *Case A* is applicable. Now, to connect the sub-trees $\{e_{7,11}\}$ and $\{e_{9,10}\}$ a shortest path between these two sub-trees is searched. Here the shortest path is between the nodes 11 and 10 with node 6 as the only intermediate node. The edges lying on the shortest path, viz. $e_{11,6}$ and $e_{6,10}$ are added into DT. The nodes 6 and 10 are recoloured BLACK. Now, the $wd(9)$ is calculated and $wd(9) \geq 1$, therefore, the edge $e_{10,9}$ is added into DT and the node 9 is recoloured BLACK and also all the WHITE neighboring nodes of the node 9, viz. 4 and 12 are recoloured GREY. This situation is shown in Fig. 3c.

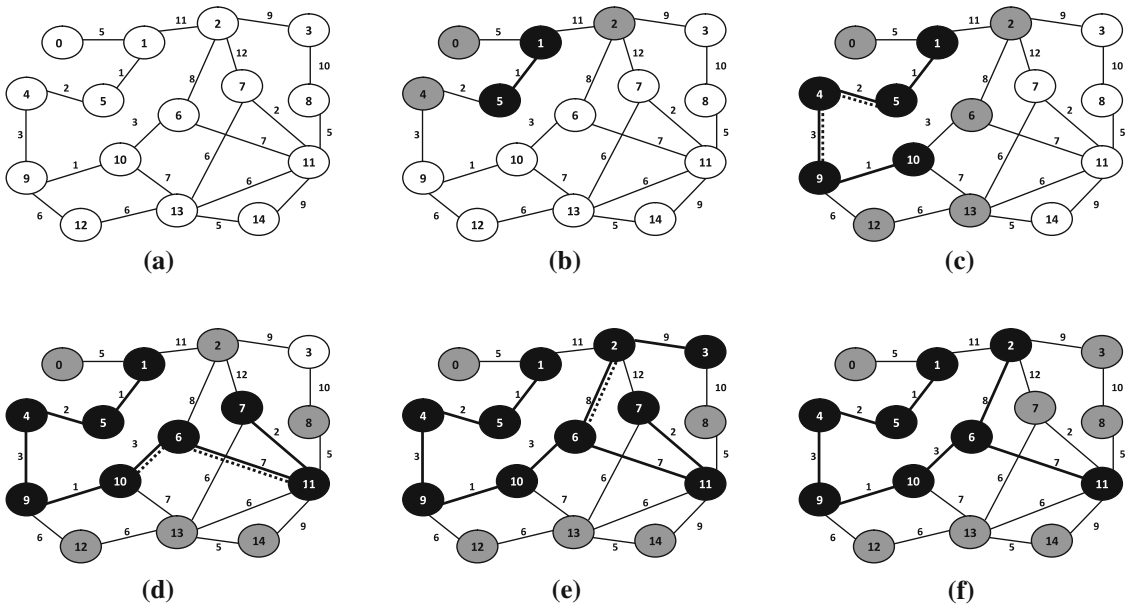

Fig. 2 Illustrating H_DT heuristic. **a** Initially, all nodes are coloured WHITE and $weight = 0$. **b** Edge $e_{1,5}$ is selected by heuristics H_DT and $weight = 1$. **c** Edge $e_{9,10}$ is selected by heuristics H_DT and $weight = 7$. **d** Edge $e_{7,11}$ is selected by heuristics H_DT and $weight = 19$. **e** Edge $e_{2,3}$ is selected by heuristics H_DT and $weight = 36$. **f** After pruning, MST, pruning and MST $weight = 25$

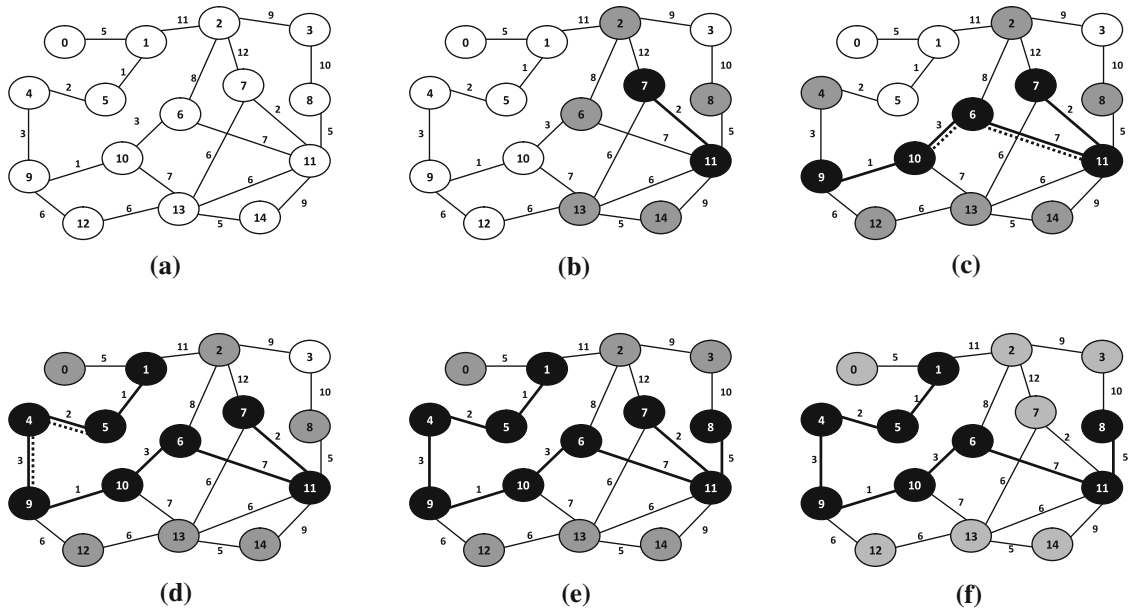

Fig. 3 Illustrating M_DT heuristic. **a** Initially, all nodes are coloured WHITE and $weight = 0$. **b** Edge $e_{7,11}$ is selected by heuristics M_DT and $weight = 2$. **c** Edge $e_{9,10}$ is selected by heuristics M_DT and $weight = 13$. **d** Edge $e_{1,5}$ is selected by heuristics M_DT and $weight = 19$. **e** Edge $e_{2,3}$ is selected by heuristics M_DT and $weight = 24$. **f** After MST and pruning $weight = 22$

Next, the edge $e_{1,5}$ is returned by the Expression 1 leading to the *Case A* again. The edges $e_{9,4}$, $e_{4,5}$, $e_{5,1}$ are added into the partially constructed dominating tree DT, the nodes 4, 5 and 1 are coloured BLACK and their WHITE neighbors GREY (in this case only the node 0). This is shown in the Fig. 3d.

In the last iteration, the edge $e_{2,3}$ is returned by the Expression 1, because the ratio of edges $e_{2,3}$, $e_{8,11}$, $e_{3,8}$, $e_{2,6}$, $e_{1,2}$ and $e_{2,7}$ are 2.11, 2, 1.9, 1.12, 0.81 and 0.75 respectively and the remaining edges have ratio zero. Here the node 2 is GREY and the node 3 is WHITE. So the *Case B* is applicable. As a result, the shortest path between neighbor-

ing nodes {2, 8} of node 3 and the partially constructed tree $\{e_{1,5}, e_{5,4}, e_{4,9}, e_{9,10}, e_{10,6}, e_{6,11}, e_{11,7}\}$ is computed. This shortest path is between the nodes 11 and 8 without any intermediate node. The edge $e_{11,8}$ is included into DT, and, the node 3, which is the only WHITE neighbor of the node 8 is recoloured GREY. Here $wd(3)$ is zero; therefore the edge $e_{8,3}$ is not included into DT. At this point M_DT stops as no WHITE node remained, and, the construction of the dominating tree DT is complete. This dominating tree has cost 24 (Fig. 3e). Comparing Fig. 2e with Fig. 3e, we can see that the cost of the dominating tree returned by the heuristic H_DT is 36, whereas the M_DT returns a dominating tree with cost 24 on the same graph. Thus, we can say that our heuristic M_DT can perform better than the heuristic H_DT of Sundar and Singh (2013). Hereafter, we compute a minimum spanning tree (MST) on the set of nodes in DT to reconnect these nodes via this MST in a bid to reduce the cost of DT further. As the nodes of DT are already connected via a MST so cost of DT remains the same. Next the pruning operator is applied to remove the redundant nodes as well as redundant edges which removes the node 7 and the edge $e_{11,7}$ leading to the final DT with cost 22 as shown in Fig. 3f.

3.2 Pruning operator

Our Pruning operator is similar to the pruning procedure of Sundar and Singh (2013). Pruning operator removes the redundant nodes of dominating tree DT. Let DN be the set of nodes belonging to DT. A node $v \in DN$ is redundant if $|ON(v) \cap DN| = 1$ *and* $CN(v) \subseteq (\cup_{u \in DN \setminus \{v\}} ON(u))$, i.e., the degree of the node v in DN must be one and all the non-dominating neighboring nodes of the node v are covered by some other dominating nodes (other than the node v) in DN. If the node v is redundant then it can be removed from DN without affecting the dominating tree characteristic of DT. Our pruning operator begins by computing the set R_n of redundant nodes and then an iterative process starts where during each iteration a node is selected and removed from DN and the set R_n is recomputed. We have selected a node for removal from R_n according to the order in which it is added into DN. We have also tried selecting a node from R_n according to the non-increasing order of the cost of their sole incident edge or according to the non-decreasing order of the number of non-dominating nodes covered by each node in R_n. Experimentally, we observed that solutions obtained through different ordering schemes did not differ much in quality and no ordering scheme has an ultimate advantage over others. Therefore, we settled for a simpler ordering scheme. Iterative process stops when the set R_n becomes empty. The pseudo-code of the pruning operator is presented in Algorithm 2 where $Select\_Node(R_n)$ is a function that returns a node from R_n which was added first into DN among all the nodes currently present in R_n.

Algorithm 2: The pseudo-code of Pruning operator

$R_n \leftarrow \{v : v \in DN \text{ and } |ON(v) \cap DN| = 1 \text{ and } CN(v) \subseteq (\cup_{u \in DN \setminus \{v\}} ON(u))\}$;
while $(R_n \neq \emptyset)$ **do**
  $v \leftarrow Select\_Node(R_n)$;
  $DN \leftarrow DN \setminus \{v\}$;
  $R_n \leftarrow \{v : v \in DN \text{ and } |ON(v) \cap DN| = 1 \text{ and } CN(v) \subseteq (\cup_{u \in DN \setminus \{v\}} ON(u))\}$;
return DN;

4 Overview of the EA/G

The evolutionary algorithm with guided mutation (EA/G) is a relatively new member in the class of evolutionary algorithms. It was developed by Zhang et al. (2005) with a motivation to overcome, as far as possible, the drawbacks of two evolutionary algorithms, viz. genetic algorithms (GAs) and estimation of distribution algorithms (EDAs).

The EA/G has the features of both GAs and EDAs. Conventionally, GAs use genetic operators such as crossover and mutation to generate an offspring from the selected parents. GAs directly utilize only the location information of the solutions and do not make use of global information about the search space which can be collected by keeping track of all the solutions generated since the beginning of the algorithm. On the other hand, EDAs rely only on a probability model to generate an offspring. The probability model characterizes the distribution of the promising solutions in the search space and is updated at each generation using the global statistical information about the search space extracted from the population members present at that generation. An offspring is generated by sampling this probability model. Contrary to GAs, EDAs do not directly utilize the location information of solutions. Here, by the location information of a solution, we mean the information that can uniquely identify a solution in the search space of all solutions. For example, in case of the DTP, the set of edges present in a dominating tree constitutes its location information as this information can uniquely identify a dominating tree.

Taking into the account this complementary aspect of GAs and EDs, Zhang et al. (2005) developed an ideal algorithm that utilizes the location information of the solutions like GAs and the global statistical information about the search space like EDAs while generating an offspring. This algorithm was named evolutionary algorithm with guided mutation (EA/G). EA/G uses a mutation operator, called *guided mutation (GM)*, to generate offsprings. Guided mutation generates a new solution considering both the location information about the parent solution as well as the global statistical information about the search space, i.e., a solution is generated partly by sampling a probability model characterizing the global statistical information and partly by copying elements from its parent.

5 Hybrid EA/G approach for DTP

Our proposed hybrid approach for the dominating tree problem (DTP) is inspired by the approach of Zhang et al. (2005) for the maximum clique problem (MCP). Success of the EA/G in solving the MCP over standard benchmark instances have motivated us to develop an EA/G approach for the DTP. Solutions obtained through the EA/G approach are further improved through the use of the same two procedures as used in M_DT, i.e., reconnecting the nodes of the solution via minimum spanning tree (MST) and pruning operator. However, each of these procedures are applied twice in the order MST → pruning → MST → pruning. Hereafter, our hybrid EA/G approach with MST and pruning operator will be referred to as EA/G-MP (EA/G with MST and pruning operator).

Before starting our EA/G-MP approach, we pre-compute the set of neighboring nodes for each node $v \in V$ and the shortest paths between all pairs of nodes in V. Subsequent subsections describe other salient features of EA/G-MP approach.

5.1 Solution encoding

Edge-set encoding has been used to represent a solution, i.e., each dominating tree is represented directly be the set of the edges it contains. The edge-set encoding was introduced by Raidl and Julstrom (2003) for representing a spanning tree. It is to be noted that a spanning tree always has $|V| - 1$ edges, whereas the number of edges in a dominating tree varies.

5.2 Initial solutions

Our initial solution generation method is derived from the initial solution generation method used in Sundar and Singh (2013). First we will describe the initial solution generation method of Sundar and Singh (2013) and then introduce our modifications. Let W_n be the set of WHITE nodes which is initialized to V ($W_n = V$ initially). Let I_u be the set of non-dominating nodes, DN be the set of dominating nodes and DT be the partially constructed dominating tree. Initially, these three sets, viz. I_u, DN and DT are empty. Randomly, a node say v is selected from W_n and added into DN and recoloured BLACK and node v is removed from set W_n. Now, consider a set n_b of WHITE neighboring nodes of node v, i.e., $n_b = ON(v) \cap W_n$. Remove the nodes in n_b also from W_n. After that all the nodes in n_b are recoloured GREY and added into the set I_u. From here onwards, at each step, an edge is selected by following one of the two strategies. With probability φ, first strategy is followed where a least cost edge $e_{v,u}$, connecting a node v in DN and a node u in I_u is selected. Otherwise second strategy is followed where an edge $e_{v,u}$ connecting a node v in DN to a node u in I_u is selected randomly. Here φ is a parameter to be determined empirically. Clearly, first strategy aims at quality, whereas the latter strategy aims at diversity. Therefore, φ governs the delicate balance between the quality and the diversity of initial solutions. We have made two modifications in the initial solution generation method of Sundar and Singh (2013). Our first modification here is that only those edges $e_{v,u}$ are considered where $u \in I_u$ has at least one WHITE neighboring node, i.e., where $|ON(u) \cap W_n| \geq 1$. Whereas in Sundar and Singh (2013), those nodes in I_u are also considered which have no WHITE neighboring node and as a result some edges might be unnecessarily inserted into DT. In addition, with probability $1-\varphi$, we are selecting the edges uniformly at random instead of using roulette wheel selection method like Sundar and Singh (2013). Now, the node u is added into set DN and removed from I_u and an edge $e_{v,u}$ is added into the set DT. Nodes in $n_b = ON(u) \cap W_n$ are added into I_u and nodes in n_b are removed from W_n. After that all the WHITE neighboring nodes of the node u are recoloured GREY. This whole process is repeated until the set W_n becomes empty. After construction of a feasible dominating tree DT, a pruning operator (as described in Sect. 3.2) is applied to remove the redundant nodes from DN and the redundant edges from DT. After an application of the pruning operator, a MST is constructed on the set of nodes in DN and these nodes are reconnected via this MST. Pseudo-code of the construction of an initial solution is presented in the Algorithm 3. Here, we have applied pruning first

Algorithm 3: The pseudo-code for initial solution

```
//Initially all nodes in V are coloured WHITE
W_n ← V;   DT ← ∅;   DN ← ∅;   I_u ← ∅;
v ← random(W_n);
DN ← DN ∪ {v};
make v BLACK;
n_b ← ON(v) ∩ W_n;
W_n ← W_n \ (n_b ∪ {v});
make all nodes ∈ n_b GREY;
I_u ← n_b;
while W_n ≠ ∅ do
    Generate a random number u_01 such that 0 ≤ u_01 ≤ 1;
    if u_01 < φ then
        (v, u) ← arg min_{v∈DN, {u∈I_u, |ON(u)∩W_n|≥1}} w(v, u);
    else
        v ← random(DN);
        u ← {u : random(I_u) and |ON(u) ∩ W_n| ≥ 1};
    DT ← DT ∪ {e_{v,u}};
    DN ← DN ∪ {u};
    make u BLACK;
    I_u ← I_u \ {u};
    n_b ← {u : u ∈ ON(u) ∩ W_n};
    make all nodes ∈ n_b GREY;
    I_u ← I_u ∪ n_b;
    W_n ← W_n \ n_b;
Apply pruning operator on nodes in DT;
Reconnect nodes in DT via a minimum spanning tree;
return DT;
```

and then MST with the intention of generating more diverse solutions.

5.3 Initialization and update of the probability vector

Our EA/G-MP, as in Zhang et al. (2005), models the distribution of promising solutions in the search space through the use of univariate marginal distribution (UMD) model. In this model, a probability vector $p = \{p_1, p_2, \ldots, p_{|V|}\} \in [0, 1]^{|V|}$ is used to characterize the distribution of the promising solutions in the search space, where $|V|$ is the cardinality of the set V, i.e., the number of nodes in the graph G. p_v is the probability of the node $v \in V$ to be present in a dominating tree. The probability vector is initialized using N_p initial solutions. The probability of each node is initialized to the ratio of the number of initial solutions containing that node to the total number of initial solutions. The pseudo-code for initializing the probability vector p for the DTP is presented in Algorithm 4.

Algorithm 4: The pseudo-code for initializing a probability vector p

```
Compute n_v ← number of initial solutions containing
node v, ∀v ∈ V;
Compute p_v ← n_v / N_p, ∀v ∈ V;
```

At each generation g, a parent set *parent(g)* is formed by selecting the best L solutions from current population *pop(g)*. Once *parent(g)* is formed, it is used for updating the probability vector p. The pseudo-code for updating the probability vector is given in Algorithm 5, where $\lambda \in (0, 1]$ is the learning rate and it governs the contribution of solutions in *parent(g)* to the updated probability vector p, i.e., higher the value of λ, more is the contribution of solutions in *parent(g)*. The probability of a node increases after update if the ratio of solutions containing this node in *parent(g)* to the total number of solutions in *parent(g)* is more than its current probability. The probability decreases in case this ratio is less than its current value. The probability remains the same in case this ratio is exactly equal to its current value.

Algorithm 5: The pseudo-code for updating the probability vector p in generation g

```
Compute n_v ← number of solutions in parent(g)
containing node v, ∀v ∈ V;
Compute p_v ← (1 − λ) p_v + λ n_v / L, ∀v ∈ V;
```

5.4 *Guided mutation (GM)* operator

As we have already discussed in Sect. 4, the GM operator uses both the global statistical information stored in the form of probability vector p and the location information of the parent solution for generating new offsprings. Zhang et al. (2005) applied GM operator M times on the best solution of the current population to generate M offsprings. On the other hand, our GM operator is applied on M best solutions of current population $pop(g)$ to generate M new offsprings. In other words, on the set of M best solutions $\{m_1, m_2, \ldots, m_M\}$, GM is applied once on each $m_i, i = 1, 2 \ldots, M$ to generate $\{o_1, o_2, \ldots, o_M\}$ offsprings. The pseudo-code of our GM operator is presented in Algorithm 6 where $\beta \in [0, 1]$ is an adjustable parameter and DT is a new offspring constructed through GM operator whose nodes are either sampled randomly from the probability vector p or directly copied from the solution m_i in $pop(g)$. In case of sampling from probability vector p, a node is copied only when either its colour is WHITE according to partially constructed DT or its colour is GREY and it has at least one WHITE neighbor. Whereas in case a node is to be directly copied from the solution m_i, it is copied only when it is a dominating node in m_i, its colour is GREY according to partially constructed DT and it has at least one WHITE neighbor. The reason behind such a policy lies in the fact that by copying a node from m_i only when its colour is GREY according to DT will help in getting some more edges in DT from m_i. As m_i is among the best M solutions, this may help in improving the solution quality. There is no guarantee of the feasibility of the offspring generated through GM operator, i.e., it may

Algorithm 6: The pseudo-code of generating a solution through GM operator

```
Set the colour of all nodes in V to WHITE;
DT ← ∅;
foreach node v ∈ V in some random order do
    Generate a random number r_1 such that 0 ≤ r_1 ≤ 1;
    if r_1 < β then
        Generate a random number r_2 such that 0 ≤ r_2 ≤ 1;
        if (r_2 < p_v) and ((v is WHITE) or (v is GREY with at
        least one WHITE neighbor)) then
            Find the shortest path SP between node v and a node
            u in DT;
            Add all the edges of SP into DT;
            Colour BLACK all the nodes on the path SP;
            Colour GREY all the WHITE neighboring nodes of
            nodes on the path SP;

    else
        if v is a dominating node in m_i and v is GREY with at
        least one WHITE neighbor then
            Find the shortest path SP between node v and a node
            u in DT;
            Add all the edges of SP into DT;
            Colour BLACK all the nodes on the path SP;
            Colour GREY all the WHITE neighboring nodes of
            nodes on the path SP;
return DT;
```

not be a dominating tree. Therefore, each infeasible offspring generated through GM operator is passed through a repair operator (Sect. 5.5) so that it can be made feasible.

5.5 Repair operator

Repair operator is applied only on an infeasible offspring generated through GM operator. After the application of GM operator, there is a possibility that some WHITE nodes remain, i.e., some nodes may remain uncovered. Let U_{cn} be the set of such WHITE nodes. Such WHITE nodes are covered by making use of the repair operator which follows an iterative procedure. During each iteration, a node with the highest number of WHITE neighboring nodes is selected from U_{cn} (ties are broken in favour of the node having lower index). If none of the nodes in the set U_{cn} has WHITE neighboring nodes, then the node with lowest index is selected from U_{cn}. After selecting the node i from the set U_{cn} , a shortest path between the node i and the partially constructed tree DT is found, and, all the edges on this path are added into DT. All the nodes on this path are recoloured BLACK (if not already) and their neighboring nodes GREY. Then all BLACK and GREY nodes are removed from the set U_{cn}. After this another iteration begins. This whole process is repeated until set U_{cn} becomes empty. The pseudo-code of the repair operator is given in Algorithm 7.

Algorithm 7: The pseudo-code of repair operator

while $U_{cn} \neq \emptyset$ **do**
  $v \leftarrow \arg\max_{u \in U_{cn}}(wd(u) > 0)$;
  if $v = \emptyset$ **then**
    Select a node v with lowest index from U_{cn};
  Find a shortest path SP between node v and a node u in DT;
  Add all the edges on the path SP into DT;
  Make BLACK all nodes $\in SP$;
  Make GREY all WHITE neighboring nodes of nodes $\in SP$;
  Remove all BLACK and GREY nodes from U_{cn};
return DT;

5.6 Others features

Zhang et al. (2005) kept best $\frac{N_p}{2}$ solutions of $pop(g)$ into $parent(g)$ and generated $\frac{N_p}{2}$ new offsprings through GM operator in each generation (iteration). The population of the next generation is formed by using $\frac{N_p}{2}$ newly created offsprings through GM operator and best $\frac{N_p}{2}$ solutions of $pop(g)$. Therefore, in each next generation the population size remains the same as in previous generation. On the other hand, in our approach $parent(g)$ is formed by using best L solutions of $pop(g)$ and M new offsprings are generated through GM operator in each generation. The best $N_p - M$ solutions of $pop(g)$ along with M newly generated offsprings constitute $pop(g+1)$. Therefore, also in this case population size remains the same throughout the execution of the algorithm.

Unlike Zhang et al. (2005), we never found all the solutions of the population to be same. We also observed that the best solution does not improve for a large number of generations. Therefore, to avoid getting stuck into a local optimum, if the best solution does not improve over S_c generations, then, except for the best solution, we reinitialize the entire population in the same manner as described in Sect. 5.2. So, in a way, we have followed the 1-elitism policy as best solution is retained always.

The pseudo-code of our EA/G-MP approach for DTP is given in Algorithm 8.

Algorithm 8: EA/G-MP Approach for DTP

1. At generation $g \leftarrow 0$, an initial population *pop(g)* consisting of N_p solutions, is generated randomly;
2. Initialize the probability vector p for all nodes using Algorithm 4;
3. Select best L solutions from *pop(g)* to form a parent set *parent(g)*, and then update the probability vector p using Algorithm 5;
4. Apply the *GM* operator once on each of the M best solutions in *pop(g)* in order to generate M new solutions. A repair operator is applied to each generated solution, if necessary, and then MST, pruning operator, MST and pruning operator are applied to each generated solution to improve its fitness. Add all M newly generated solutions along with $N_p - M$ best solutions in *pop(g)* to form *pop(g+1)*. If the stopping condition is met, return the dominating tree with minimum weight found so far ;
5. $g \leftarrow g + 1$;
6. If the best solution of the population did not improve over S_c generations, then reinitialize entire *pop(g)* except for the best solution, and then go to step 2 ;
7. Go to step 3 ;

6 Computational results

Our approaches, viz. M_DT and EA/G-MP have been implemented in C and executed on an Intel Core 2 Duo processor based system with 2 GB RAM running under Fedora 12 at 3.0 GHz which is exactly the same system as used for executing the approaches of Sundar and Singh (2013). Likewise `gcc 4.4.4-10` compiler with `O3` flag has been used to compile the C programs of our approaches. We have used a super set of test instances used in Sundar and Singh (2013) to test our approaches. Due to unavailability of the test instances used in Zhang et al. (2008) and Shin et al. (2010), Sundar and Singh (2013) generated a set of 18 test instances in the same manner as in Zhang et al. (2008) and Shin et al. (2010). These instances were generated considering a disk graph $G = (V, E)$, where each disk around a

node represents the transmission range of that node. There exists an edge between a pair of nodes if these two nodes are within the transmission range of each other. The weight on each edge $e_{i,j}$ in E is assigned through a weight function $w : E \rightarrow \Re^+$ which is defined as $w(e_{i,j}) = d_{i,j}^2$, where $d_{i,j}$ is the Euclidean distance between the nodes i and j. It was assumed that nodes in $|V|$ are randomly deployed in a $500m \times 500m$ area and transmission range of each node is 100m. For each value of $|V|$ in {50, 100, 200, 300, 400, 500}, three different test instances were generated leading to a total of 18 instances. In addition to the transmission range of 100m, we consider two more values for transmission range of each node, viz. 125 and 150 m and generated three different test instances for each combination of values of V mentioned above and one of these two values of transmission range. This results in generation of 36 additional instances leading to a grand total of 54 instances. All these 54 test instances can be downloaded from http://dcis.uohyd.ernet.in/~alokcs/dtp.zip. Actually, density of a disk graph depends on the transmission range of its constituent nodes. The longer the transmission range of nodes, the higher will be the density of the corresponding disk graph. Therefore, to show that effectiveness of our proposed approaches is not limited to graphs with a particular density, it is necessary to consider different values of transmission range. The values of transmission range that we have considered leads to difficult randomly generated feasible DTP instances. We have also considered the transmission range of 75 and 200 m. At the transmission range of 75m, not all instances with 50 nodes were connected. At the transmission range of 200 m, generated instances were highly dense, and therefore, all dominating trees on these instances had few edges only, and as a result,finding a minimum dominating tree among them was not that difficult. As the C programs for the approaches considered in Sundar and Singh (2013) were available, we have executed them on these additional 36 instances under the same setup as used for our approaches.

For EA/G-MP, we have used a population size of 60, i.e., $N_p = 60$, generated $M = 25$ new solutions through guided mutation and used $L = 15$ best solutions of current population to update probability vector. The value of β is set to 0.50 in the guided mutation. The value $\lambda = 0.50$ is used in the update of probability vector and the value $\varphi = 0.20$ is used in the initial solution generation. If the best solution does not improve over $S_c = 400$ generations, entire population minus the best solution and the probability vector are reinitialized. We have allowed our EA/G-MP approaches to execute till the best solution does not improve over 3,000 generations and it has executed at least for a total of 10,000 generations. All these parameters are set empirically after a large number of trials. These parameter values provide good results on all instances, though they may not be optimal for all instances. Like ABC_DT and ACO_DT approaches of Sundar and Singh (2013), EA/G-MP has been executed 20 independent times on each test instance.

We first present the results of M_DT and other problem specific heuristics. Tables 2, 3 and 4 report the results of M_DT on instances with transmission range 100, 125 and 150 m respectively and compare them with previously proposed heuristic approaches, viz. heuristics of Zhang et al. (2008), Shin et al. (2010) and Sundar and Singh (2013), which will be referred to as Heu_DT1, Heu_DT2 and H_DT respectively. In addition, we have also included a simple heuristic which computes a MST on the input graph and then removes leaf nodes from the computed MST to obtain a dominating tree. This heuristic, which will be referred to as MST-L was used in Zhang et al. (2008) and Shin et al. (2010) for comparison against their respective heuristics. For each heuristic, these tables report the cost of the dominating tree obtained (column labelled Value) and the number of nodes in the dominating tree (column labelled NDN) on each instance. In Table 2, data for Heu_DT1, Heu_DT2, H_DT and MST-L are taken from Sundar and Singh (2013). Whereas Tables 3 and 4 contain the results obtained after executing various approaches on 36 new instances. These three tables also report the % improvement in cost of the dominating tree obtained by M_DT over other approaches. Though not the objective of DTP, we have reported the number of nodes in the dominating tree due to past precedences. Zhang et al. (2008), Shin et al. (2010) and Sundar and Singh (2013), all reported the number of nodes in the dominating tree obtained by various approaches. These tables clearly show the superiority of M_DT over other approaches in terms of the cost of the dominating tree obtained. Except for 8 instances (3 with transmission range 100 m, 4 with transmission range 125 m and 1 with transmission range 150 m) where H_DT has slightly better cost (as indicated by negative value for % improvement of M_DT over H_DT in Tables 2, 3 and 4), cost of the dominating tree obtained by M_DT is always better than all the other approaches. As far as number of dominating nodes in a solution is concerned, performance of M_DT is far superior in comparison to Heu_DT1, Heu_DT2 and MST-L on all instances. However, H_DT performs slightly better on this count on most of the instances. Execution times of various heuristics are not reported as all of them hardly need a second on any instance.

Tables 5, 6 and 7 report the results of EA/G-MP on instances with transmission range 100, 125 and 150 m respectively and compare them with ABC_DT and ACO_DT approaches of Sundar and Singh (2013). For each test instance, these tables report the best solution (column Best), average solution quality (column Avg), standard deviation of solution values (column SD), average number of dominating nodes (column ANDN) and average total execution time in seconds (column ATET) obtained over 20 runs for EA/G-MP, ABC_DT and ACO_DT. Data for ABC_DT

Table 2 Results of MST_L, Heu_DT1, Heu_DT2, H_DT and M_DT on the instances with transmission range 100 m

Instance	MST_L		Heu_DT1		Heu_DT2		H_DT		M_DT		% Improvement			
	Value	NDN	Value	NDN	Value	NDN	Value	NDN	Value	NDN	MST_L	Heu_DT1	Heu_DT2	H_DT
50_1	1,860.67	38	1,608.11	30	1,819.91	35	1,321.83	20	1,288.80	20	30.73	19.86	29.18	2.50
50_2	1,780.66	37	1,564.05	33	1,795.89	35	1,427.65	26	1,467.03	26	17.61	6.20	18.31	−2.75
50_3	1,860.12	39	1,659.70	34	1,863.01	35	1,494.12	25	1,429.76	24	23.14	13.85	23.26	4.31
100_1	2,491.23	76	1,836.57	54	2,290.28	61	1,852.86	28	1,482.11	26	40.51	19.30	35.29	20.00
100_2	2,515.82	78	2,096.97	62	2,265.68	62	1,449.25	23	1,454.16	25	42.20	30.65	35.82	−0.32
100_3	2,670.84	77	2,213.15	60	2,488.15	59	1,732.35	29	1,704.19	31	36.17	22.97	31.48	1.58
200_1	3,652.20	154	2,530.57	110	3,093.01	115	1,880.50	30	1,766.97	35	51.62	30.18	42.87	6.04
200_2	3,597.99	150	2,709.42	113	3,437.79	125	1,909.86	32	1,695.43	34	52.88	37.42	50.68	11.23
200_3	3,592.74	152	2,561.99	110	3,132.56	112	1,587.48	27	1,589.81	31	55.75	37.95	49.25	−0.14
300_1	4,445.38	231	2,932.26	154	3,653.64	165	1,929.91	34	1,695.08	30	61.87	42.20	53.61	12.17
300_2	4,498.58	233	3,480.73	178	4,136.57	183	1,781.00	31	1,773.32	35	60.58	49.05	57.13	0.43
300_3	4,673.49	239	3,640.35	184	3,990.55	170	1,815.28	33	1,673.31	33	64.20	54.03	58.07	7.82
400_1	5,110.49	311	3,776.71	230	4,524.29	228	2,017.50	32	1,587.43	30	68.94	57.97	64.91	21.32
400_2	5,225.01	310	4,004.41	243	4,744.41	248	1,972.89	36	1,904.82	37	63.54	52.43	59.85	3.45
400_3	5,227.94	314	4,026.04	241	4,394.95	218	1,907.05	29	1,883.79	32	63.97	53.21	57.14	1.22
500_1	5,761.72	390	4,276.57	291	4,534.93	257	1,795.28	27	1,771.82	34	69.25	58.57	60.93	1.31
500_2	5,953.15	398	4,399.44	296	5,251.35	309	1,824.03	34	1,683.54	29	71.62	61.60	67.83	7.38
500_3	5,840.50	390	4,629.12	304	4,944.21	269	1,903.86	29	1,837.40	30	68.54	60.31	62.84	3.49

Table 3 Results of MST_L, Heu_DT1, Heu_DT2, H_DT and M_DT on the instances with transmission range 125 m

Instance	MST_L		Heu_DT1		Heu_DT2		H_DT		M_DT		% Improvement			
	Value	NDN	Value	NDN	Value	NDN	Value	NDN	Value	NDN	MST_L	Heu_DT1	Heu_DT2	H_DT
50_1	1,860.67	37	1,404.49	26	1,679.80	29	982.61	14	1,047.25	13	43.72	25.59	37.66	−6.58
50_2	1,780.66	36	1,407.53	26	1,705.67	31	1,165.63	16	1,179.19	19	33.78	16.22	30.87	−1.16
50_3	1,860.12	38	1,488.20	29	1,774.12	32	1,154.05	14	1,201.88	19	35.39	19.24	32.25	−4.14
100_1	2,517.76	75	1,737.47	50	2,258.10	56	1,442.11	18	1,331.99	20	47.10	23.34	41.01	7.64
100_2	2,515.82	77	1,969.81	59	2,372.79	60	1,511.53	21	1,238.10	20	50.79	37.15	47.82	18.09
100_3	2,670.84	76	2,213.15	59	2,402.43	59	1,445.39	20	1,311.60	21	50.89	40.74	45.41	9.26
200_1	3,652.20	153	2,510.88	109	2,990.67	100	1,639.11	20	1,355.67	24	62.88	46.01	54.67	17.29
200_2	3,597.99	149	2,733.43	113	3,074.32	115	1,436.93	23	1,367.08	20	62.00	49.98	55.53	4.85
200_3	3,592.74	151	2,540.70	108	3,118.57	109	1,345.50	21	1,307.22	20	63.61	48.55	58.08	2.85
300_1	4,445.38	230	2,899.16	153	3,537.02	150	1,454.30	19	1,516.07	26	65.90	47.71	57.14	−4.25
300_2	4,498.58	232	3,500.06	180	4,310.97	189	1,739.35	23	1,387.87	19	69.15	60.35	67.81	20.21
300_3	4,673.49	238	3,612.75	183	3,802.58	160	1,541.75	21	1,370.75	20	70.67	62.06	63.95	11.09
400_1	5,110.49	310	3,758.23	228	4,211.58	210	1,654.87	23	1,528.15	24	70.10	59.34	63.72	7.66
400_2	5,225.01	309	3,901.81	233	4,596.69	235	1,739.40	25	1,539.59	23	70.53	60.54	66.51	11.49
400_3	5,227.94	313	3,981.63	238	4,421.78	213	1,630.39	23	1,524.44	24	70.84	61.71	65.52	6.50
500_1	5,761.72	389	4,354.11	298	4,472.74	245	1,563.24	23	1,551.67	26	73.07	64.36	65.31	0.74
500_2	5,953.15	397	4,471.75	299	5,005.20	298	1,638.64	23	1,548.49	23	73.99	65.37	69.06	5.50
500_3	5,840.50	389	4,508.65	297	4,715.44	259	1,731.32	24	1,344.64	23	76.98	70.18	71.48	22.33

and ACO_DT is taken from Sundar and Singh (2013) for Table 5. On the other hand, Tables 6 and 7 report the results obtained after executing ABC_DT and ACO_DT on 36 new instances. These three tables also report the results of Mann–Whitney *U* test between EA/G-MP and ABC_DT (column ABC_EA/G) and between EA/G-MP and ACO_DT

Table 4 Results of MST_L, Heu_DT1, Heu_DT2, H_DT and M_DT on the instances with transmission range 150 m

Instance	MST_L		Heu_DT1		Heu_DT2		H_DT		M_DT		% Improvement			
	Value	NDN	Value	NDN	Value	NDN	Value	NDN	Value	NDN	MST_L	Heu_DT1	Heu_DT2	H_DT
50_1	1,860.67	37	1,408.01	26	1,766.43	31	1,058.69	12	784.03	11	57.86	44.32	55.61	25.94
50_2	1,780.66	36	1,309.87	27	1,728.53	33	1,061.02	12	1,047.55	15	41.17	20.03	39.40	1.27
50_3	1,860.12	38	1,389.86	28	1,887.77	32	1,104.85	14	1,010.84	12	45.66	27.27	46.45	8.51
100_1	2,517.76	75	1,737.47	50	2,219.63	53	1,420.84	14	1,278.74	17	49.21	26.40	42.39	10.00
100_2	2,515.82	77	2,014.17	63	2,276.44	57	1,009.99	13	964.16	13	61.68	52.13	57.65	4.54
100_3	2,670.84	76	2,148.23	58	2,331.72	55	1,124.55	13	1,184.35	15	55.66	44.87	49.21	−5.32
200_1	3,652.20	153	2,530.52	109	2,911.73	95	1,319.86	15	1,286.15	16	64.78	49.17	55.83	2.55
200_2	3,597.99	149	2,703.77	114	3,327.77	112	1,300.02	19	1,203.19	15	66.56	55.50	63.84	7.45
200_3	3,592.74	151	2,561.99	109	3,112.19	108	1,258.29	16	1,224.62	17	65.91	52.24	60.65	2.68
300_1	4,445.38	230	2,908.12	153	3,344.34	143	1,170.23	16	1,144.65	15	74.25	60.64	65.77	2.19
300_2	4,498.58	232	3,493.12	180	3,740.08	160	1,324.59	19	1,224.33	13	72.78	64.95	67.26	7.57
300_3	4,673.49	238	3,589.75	183	4,016.39	151	1,382.82	18	1,195.94	13	74.41	66.68	70.22	13.51
400_1	5,110.49	310	3,790.75	231	3,704.20	187	1,295.98	15	1,166.44	17	77.18	69.23	68.51	10.00
400_2	5,225.01	309	4,017.53	243	4,350.94	218	1,174.12	13	1,171.00	17	77.59	70.85	73.09	0.27
400_3	5,227.94	313	4,006.47	238	4,304.85	197	1,335.58	17	1,272.84	17	75.65	68.23	70.43	4.70
500_1	5,761.72	389	4,253.35	288	4,540.70	249	1,252.10	15	1,089.90	18	81.08	74.38	76.00	12.95
500_2	5,953.15	397	4,379.35	296	5,269.14	303	1,286.67	15	1,279.42	17	78.51	70.79	75.72	0.56
500_3	5,840.50	389	4,618.27	300	4,767.25	250	1,474.32	17	1,300.60	16	77.73	71.84	72.72	11.78

(column ACO_EA/G) on each instance as best and average solution quality of these approaches are close to each other. For Mann–Whitney U test, we have used the online calculator available at http://www.socscistatistics.com/tests/mannwhitney/Default2.aspx. For this test, we have used two-tailed hypothesis and 5 % significance criterion (p value ≤ 0.05) leading to a critical U value of 127.

6.1 Comparison of EA/G-MP, ABC_DT and ACO_DT approaches on instances with transmission range 100 m

Out of 18 test instances with transmission range 100 m, EA/G-MP is better than ABC_DT on 10 test instances and equal to ABC_DT on 8 test instances in terms of quality of the best solution found. Whereas in terms of average solution quality, EA/G-MP is better than ABC_DT on 12 test instances and worse than ABC_DT on 3 test instances and on the remaining 3 test instances EA/G-MP is equal to ABC_DT. Results of Mann–Whitney U test between EA/G-MP and ABC_DT indicate that out of 18 test instances, results of EA/G-MP is statistically significant on 12 test instances in comparison to ABC_DT, whereas on 3 test instances (100_2, 200_1 and 200_2) results are not significant. On the remaining three instances, results of Mann–Whitney U test are not meaningful as both the approaches obtained the same results in all 20 runs. As far as comparison in terms of average number of dominating nodes is concerned, EA/G-MP is better than ABC_DT on 5 test instances, worse than ABC_DT on 9 test instances and equal to ABC_DT on 4 test instances. Our EA/G-MP approach is much faster than ABC_DT approach. On all 18 test instances the average total execution time (ATET) of EA/G-MP is better than ABC_DT. From Table 5, it can be observed that as the size of the input graph increases, the gap in terms of computational time between ABC_DT and EA/G-MP increases as well. On an average, EA/G-MP is 3 times faster than ABC_DT on the instances with 50 nodes, 3 times faster than ABC_DT on the instances with 100 nodes, 4 times faster than ABC_DT on the instances with 200 nodes, 5 times faster than ABC_DT on the instances with 300 nodes, 6 times faster than ABC_DT on the instances with 400 nodes and 8 times faster than ABC_DT on the instance with 500 nodes.

Now, we compare EA/G-MP with ACO_DT. Out of 18 test instances, the best solution of EA/G-MP is better than ACO_DT on 11 test instances, worse than ACO_DT on 2 test instances and on the remaining 5 test instances both approaches obtained the same best solution. In terms of average solution quality, EA/G-MP is better than ACO_DT on 10 test instances, worse than ACO_DT on 5 test instances and on the remaining 3 test instances both the approaches have the same average solution quality. Results of Mann–Whitney U test between EA/G-MP and ACO_DT show that out of 18 test instances, results of EA/G-MP is statistically significant in comparison to ACO_DT on 12 test instances whereas on 3 test instances (100_1, 400_3 and 500_3) results are not significant. On the remaining three instances, results of Mann–

Table 5 Results of ABC_DT, ACO_DT and EA/G-MP on the instances with transmission range 100 m

Instance	ABC_DT					ACO_DT					EA/G-MP					ABC_EA/G		ACO_EA/G	
	Best	Avg	SD	ANDN	TET	Best	Avg	SD	ANDN	TET	Best	Avg	SD	ANDN	TET	*U value*	*p value*	*U value*	*p value*
50_1	1,204.41	1,204.41	0.00	19.00	25.57	1,204.41	1,204.41	0.00	19.00	2.41	1,204.41	1,204.41	0.00	19.00	6.75	200.00	0.99202[a]	200.0	0.99202[a]
50_2	1,340.44	1,340.44	0.00	21.00	21.46	1,340.44	1,340.44	0.00	21.00	4.18	1,340.44	1,340.44	0.00	21.00	8.13	200.00	0.99202[a]	200.0	0.99202[a]
50_3	1,316.39	1,316.39	0.00	19.00	22.99	1,316.39	1,316.39	0.00	19.00	2.50	1,316.39	1,316.39	0.00	19.00	6.82	200.00	0.99202[a]	200.0	0.99202[a]
100_1	1,217.47	1,218.15	0.69	18.45	28.64	1,217.47	1,217.47	0.00	19.00	12.71	1,217.47	1,217.61	0.43	18.90	11.06	112.00	0.01778	180.00	0.59612
100_2	1,128.40	1,128.42	0.09	17.90	27.58	1,152.85	1,152.85	0.00	17.00	10.86	1,128.40	1,128.54	0.20	17.30	9.93	143.50	0.13104	0.00	0.00000
100_3	1,252.99	1,253.14	0.23	19.70	28.39	1,253.49	1,253.49	0.00	19.00	8.96	1,253.49	1,257.37	4.29	19.00	11.97	33.00	0.00000	110.00	0.01552
200_1	1,206.79	1,209.52	2.69	18.25	84.10	1,206.79	1,207.61	3.58	18.05	81.13	1,206.79	1,208.26	2.10	18.55	19.76	128.00	0.05360	5.00	0.00000
200_2	1,216.41	1,219.74	2.15	18.90	87.78	1,216.23	1,217.73	2.61	17.65	78.72	1,216.41	1,222.23	7.67	18.95	21.55	167.50	0.38430	65.00	0.00028
200_3	1,253.02	1,258.06	3.42	22.15	90.44	1,247.25	1,248.94	2.99	20.90	97.93	1,247.63	1,250.78	2.37	21.80	21.31	14.00	0.00000	89.00	0.00278
300_1	1,229.97	1,237.47	2.89	21.75	145.17	1,228.24	1,243.70	9.71	22.85	352.89	1,225.22	1,230.48	3.86	21.75	29.57	41.00	0.00000	37.00	0.00000
300_2	1,182.52	1,200.79	7.82	19.60	162.59	1,176.45	1,193.95	10.51	21.10	260.30	1,170.85	1,171.30	0.69	19.00	28.56	0.00	0.00000	0.00	0.00000
300_3	1,257.21	1,271.20	6.74	20.50	145.75	1,261.18	1,276.75	9.27	24.60	251.91	1,252.14	1,260.83	5.61	21.80	29.66	54.00	0.00000	27.00	0.00000
400_1	1,223.61	1,241.75	7.88	21.90	263.13	1,220.62	1,237.45	9.50	26.05	600.74	1,211.72	1,220.79	5.31	22.70	40.26	11.00	0.00000	10.00	0.00000
400_2	1,220.54	1,235.29	6.97	22.45	249.39	1,209.69	1,246.14	21.41	24.40	591.44	1,199.92	1,202.82	1.98	21.30	40.01	0.00	0.00000	0.00	0.00000
400_3	1,266.41	1,276.80	4.59	22.30	216.95	1,254.10	1,270.34	9.42	25.85	530.58	1,248.29	1,268.38	8.80	22.70	39.80	70.00	0.00046	177.00	0.54186
500_1	1,233.14	1,241.60	4.56	21.40	379.72	1,219.66	1,240.05	9.17	26.50	1163.20	1,206.07	1,222.12	11.19	22.30	44.75	33.00	0.00000	37.00	0.00000
500_2	1,245.59	1,258.33	5.40	22.35	364.04	1,273.86	1,295.51	13.39	28.65	1031.81	1,226.78	1,240.62	6.66	23.50	50.11	9.00	0.00000	0.00	0.00000
500_3	1,249.17	1,278.67	11.96	21.60	338.25	1,232.71	1,259.08	20.03	24.35	917.73	1,232.15	1,250.48	16.59	21.65	48.09	37.00	0.00000	160.00	0.28462

[a] Both the approaches obtained the same solution in all the runs

Table 6 Results of ABC_DT, ACO_DT and EA/G-MP on the instances with transmission range 125 m

Instance	ABC_DT					ACO_DT					EA/G-MP					ABC_EA/G		ACO_EA/G	
	Best	Avg	SD	ANDN	TET	Best	Avg	SD	ANDN	TET	Best	Avg	SD	ANDN	TET	*U value*	*p value*	*U value*	*p value*
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	200.00	0.99202[a]	190.00	0.79486
50_2	1,055.10	1,055.10	0.00	12.00	11.50	1,055.10	1,055.10	0.00	12.00	2.01	1,055.10	1,055.10	0.00	12.00	4.75	200.00	0.99202[a]	200.00	0.99202[a]
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	200.00	0.99202[a]	200.00	0.99202[a]
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	200.00	0.99202[a]	40.00	0.00000
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	0.00	0.00000	0.00	0.00000
100_3	998.18	998.82	1.18	14.00	18.09	998.18	1,006.11	7.57	13.80	7.61	998.18	998.18	0.00	14.00	8.27	100.00	0.00714	60.00	0.00016
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	30.00	0.00000	170.00	0.42372
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	173.00	0.47152	0.00	0.00000
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	72.50	0.00058	4.50	0.00000
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	18.00	0.00000	92.00	0.00362
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	4.00	0.00000	0.00	0.00000
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	153.00	0.20766	75.00	0.00076
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	0.00	0.00000	0.00	0.00000
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	12.00	0.00000	65.00	0.00028
400_3	1,003.24	1,007.05	2.59	14.30	200.38	1,002.61	1,009.07	8.13	13.90	236.62	1,002.97	1,003.62	0.22	14.90	28.21	20.00	0.00000	119.00	0.02926
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	0.00	0.00000	0.00	0.00000
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	27.00	0.00000	9.00	0.00000
500_3	992.30	1,001.21	4.39	16.15	342.23	1,006.23	1,007.36	1.53	15.85	456.67	980.67	992.64	7.19	15.90	45.36	70.00	0.00046	0.00	0.00000

[a] Both the approaches obtained the same solution in all the runs

Table 7 Results of ABC_DT, ACO_DT and EA/G-MP on the instances with transmission range 150m

Instance	ABC_DT					ACO_DT					EA/G-MP					ABC_EA/G		ACO_EA/G	
	Best	Avg	SD	ANDN	TET	Best	Avg	SD	ANDN	TET	Best	Avg	SD	ANDN	TET	*U value*	*p value*	*U value*	*p value*
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	200.00	0.99202[a]	200.00	0.99202[a]
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	200.00	0.99202[a]	200.00	0.99202[a]
50_3	743.74	743.74	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	200.00	0.99202[a]	200.00	0.99202[a]
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	170.00	0.42372	0.00	0.00000
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	190.00	0.79486	190.00	0.79486
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	200.00	0.99202[a]	200.00	0.99202[a]
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	140.00	0.10740	125.00	0.04338
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	190.00	0.79486	200.00	0.99202
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	198.50	0.97606	85.50	0.00208
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	145.00	0.14156	188.00	0.75656
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	185.00	0.69654	0.00	0.00000
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	106.50	0.01174	11.00	0.00000
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	82.50	0.00158	198.00	0.96810
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	188.00	0.75656	70.00	0.00046
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	65.00	0.00028	0.00	0.00000
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	2.00	0.00000	64.00	0.00024
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	51.00	0.00000	58.50	0.00014
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	100.50	0.00736	194.00	0.88076

[a] Both the approaches obtained the same solution in all the runs

Whitney U test are not meaningful as both the approaches obtained the same results in all 20 independent runs. As far as comparison in terms of average number of dominating nodes is concerned, EA/G-MP is better than ACO_DT on 10 test instances, worse than ACO_DT on 4 test instances and equal to ACO_DT on 4 test instances. From Table 5, it can be observed that as the size of the input graph increases, the gap in terms of computational time between ACO_DT and EA/G-MP increases as well. On an average, EA/G-MP is 4 times faster than ACO_DT on the instances with 200 nodes, 10 times faster than ACO_DT on the instances with 300 nodes, 14 times faster than ACO_DT on the instances with 400 nodes and 22 times faster than ACO_DT on the instance with 500 nodes. On the other hand, EA/G-MP is 3 times slower than ACO_DT on the instances with 50 nodes and slightly slower than ACO_DT on the instances with 100 nodes.

6.2 Comparison of EA/G-MP, ABC_DT and ACO_DT approaches on instances with transmission range 125 m

Out of 18 test instances with transmission range 125 m, the best solution obtained by EA/G-MP is better than ABC_DT on 10 test instances, worse than ABC_DT on 1 test instances and equal to ABC_DT on 7 test instances. Whereas in terms of average solution quality, EA/G-MP is better than ABC_DT on 11 test instances and worse than ABC_DT on 3 test instances and on the remaining 4 test instances EA/G-MP is equal to ABC_DT. Results of Mann–Whitney U test between EA/G-MP and ABC_DT indicate that out of 18 test instances, results of EA/G-MP is statistically significant on 12 test instances in comparison to ABC_DT, whereas on 2 test instances (200_2 and 300_3) results are not significant. On the remaining four instances, results of Mann–Whitney U test are meaningless as both the approaches obtained the same results in all 20 independent runs. As far as comparison in terms of average number of dominating nodes is concerned, EA/G-MP is better than ABC_DT on 6 test instances, worse than ABC_DT on 6 test instances and equal to ABC_DT on 6 test instances. Our EA/G-MP approach is much faster than ABC_DT approach. On all 18 test instances the average total execution time (ATET) of EA/G-MP is better than ABC_DT. From Table 6, it can be observed that as the size of the input graph increases, execution time of ABC_DT increases at a faster pace than EA/G-MP. On an average, EA/G-MP is 2 times faster than ABC_DT on the instances with 50 and 100 nodes, 4 times faster than ABC_DT on the instances with 200 nodes, 5 times faster than ABC_DT on the instances with 300 nodes, 7 times faster than ABC_DT on the instances with 400 nodes and 8 times faster than ABC_DT on the instance with 500 nodes.

As far as comparison between EA/G-MP and ACO_DT is concerned, out of 18 test instances, the best solution obtained by EA/G-MP is better than ACO_DT on 11 test instances, worse than ACO_DT on 1 test instances and on the remaining 6 test instances both approaches obtained the same best solution. In terms of average solution quality, EA/G-MP is better than ACO_DT on 16 test instances, worse than ACO_DT on 1 test instance and on the remaining 1 test instance both the approaches have the same average solution quality. Results of Mann–Whitney U test between EA/G-MP and ACO_DT show that out of 18 test instances, results of EA/G-MP is statistically significant in comparison to ACO_DT on 15 test instances whereas on 2 test instances (50_1 and 200_1) results are not significant. On the remaining one instances, result of Mann–Whitney U test is not meaningful as both the approaches obtained the same results in all 20 independent runs. As far as comparison in terms of average number of dominating nodes is concerned, EA/G-MP is better than ACO_DT on 10 test instances, worse than ACO_DT on 5 test instances and equal to ACO_DT on 3 test instances. From Table 6, it can be observed that as the size of the input graph increases, execution time of ACO_DT increases at a faster pace in comparison to EA/G-MP. On an average, EA/G-MP is 3 times faster than ACO_DT on the instances with 200 nodes, 6 times faster than ACO_DT on the instances with 300 nodes, 9 times faster than ACO_DT on the instances with 400 nodes and 10 times faster than ACO_DT on the instance with 500 nodes. On smallest instances with 50 nodes, EA/G-MP is 3 times slower than ACO_DT. On instances with 100 nodes, EA/G-MP is slightly slower than ACO_DT.

6.3 Comparison of EA/G-MP, ABC_DT and ACO_DT approaches on instances with transmission range 150 m

On 18 test instances with transmission range 150 m, EA/G-MP is better than ABC_DT on 7 test instances, worse than ABC_DT on 1 test instance and equal to ABC_DT on 10 test instances in terms of quality of the best solution obtained. On the other hand, average solution quality of EA/G-MP is better than ABC_DT on 10 test instances, worse than ABC_DT on 4 test instances and same as ABC_DT on the remaining 4 instances. Results of Mann–Whitney U test between EA/G-MP and ABC_DT indicate that out of 18 test instances, results of EA/G-MP is statistically significant on 6 test instances in comparison to ABC_DT, whereas on 8 test instances (100_1, 100_2, 200_1, 200_2, 200_3, 300_1, 300_2 and 400_2) results are not significant. On the remaining four instances, results of Mann–Whitney U test are not meaningful as both the approaches obtained the same results in all 20 independent runs. As far as comparison in terms of average number of dominating nodes is concerned, EA/G-MP is better than ABC_DT on 7 test instances, worse than ABC_DT on 5 test instances and equal to ABC_DT on 6 test instances. Our EA/G-MP approach is much faster than ABC_DT approach. On all 18 test instances the average total

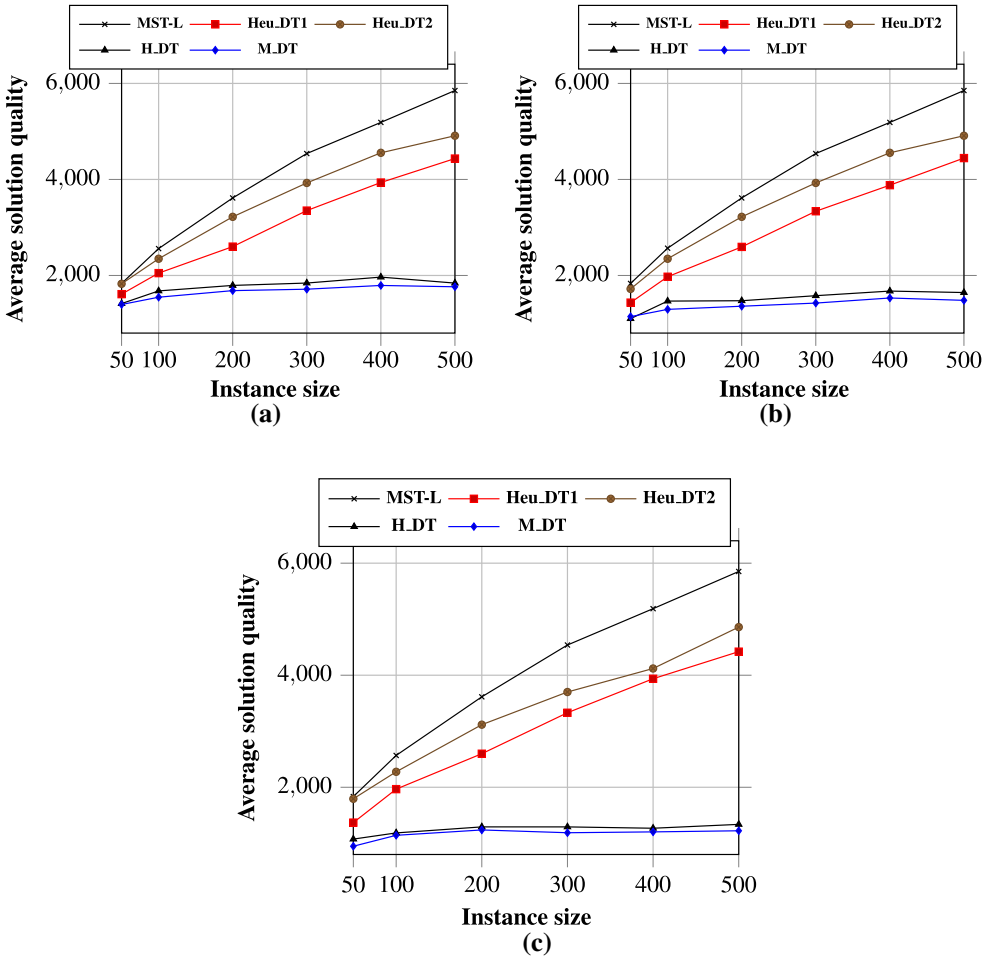

Fig. 4 Average solution quality of various heuristics over all the instances of each size for different transmission ranges. **a** Transmission range 100 m. **b** Transmission range 125 m. **c** Transmission range 150 m

execution time (ATET) of EA/G-MP is better than ABC_DT. From Table 7, it can be observed that as the size of the input graph increases, the gap in terms of computational time between ABC_DT and EA/G-MP increases as well. On an average, EA/G-MP is 2 times faster than ABC_DT on the instances with 50 nodes, 3 times faster than ABC_DT on the instances with 100 nodes, 5 times faster than ABC_DT on the instances with 200 nodes, 8 times faster than ABC_DT on the instances with 300 nodes, 11 times faster than ABC_DT on the instances with 400 nodes and 16 times faster than ABC_DT on the instance with 500 nodes.

Next, we compare EA/G-MP with ACO_DT. Out of 18 test instances, the best solution of EA/G-MP is better than ACO_DT on 8 test instances and on the remaining 10 test instances both approaches obtained the same best solution. In terms of average solution quality, EA/G-MP is better than ACO_DT on 8 test instances, worse than ACO_DT on 5 test instances and on the remaining 5 test instances both the approaches have the same average solution quality. Results of Mann–Whitney U test between EA/G-MP and ACO_DT show that out of 18 test instances, results of EA/G-MP is statistically significant in comparison to ACO_DT on 9 test instances whereas on 5 test instances (100_2, 200_2, 300_1, 400_1 and 500_3) results are not significant. On the remaining four instances, results of Mann–Whitney U test are meaningless as both the approaches obtained the same results in all 20 independent runs. As far as comparison in terms of average number of dominating nodes is concerned, EA/G-MP is better than ACO_DT on 10 test instances, worse than ACO_DT on 3 test instances and equal to ACO_DT on remaining 5 test instances.

ACO_DT is also slower than EA/G-MP. From Table 7, it can also be observed that as the size of the graph increases, execution time of ACO_DT increases at a much faster rate compared to EA/G-MP. On an average, EA/G-MP is 3 times faster than ACO_DT on the instances with 200 nodes, 5 times faster than ACO_DT on the instances with 300 nodes, 7 times faster than ACO_DT on the instances with 400 nodes and 10 times faster than ACO_DT on the instance with 500

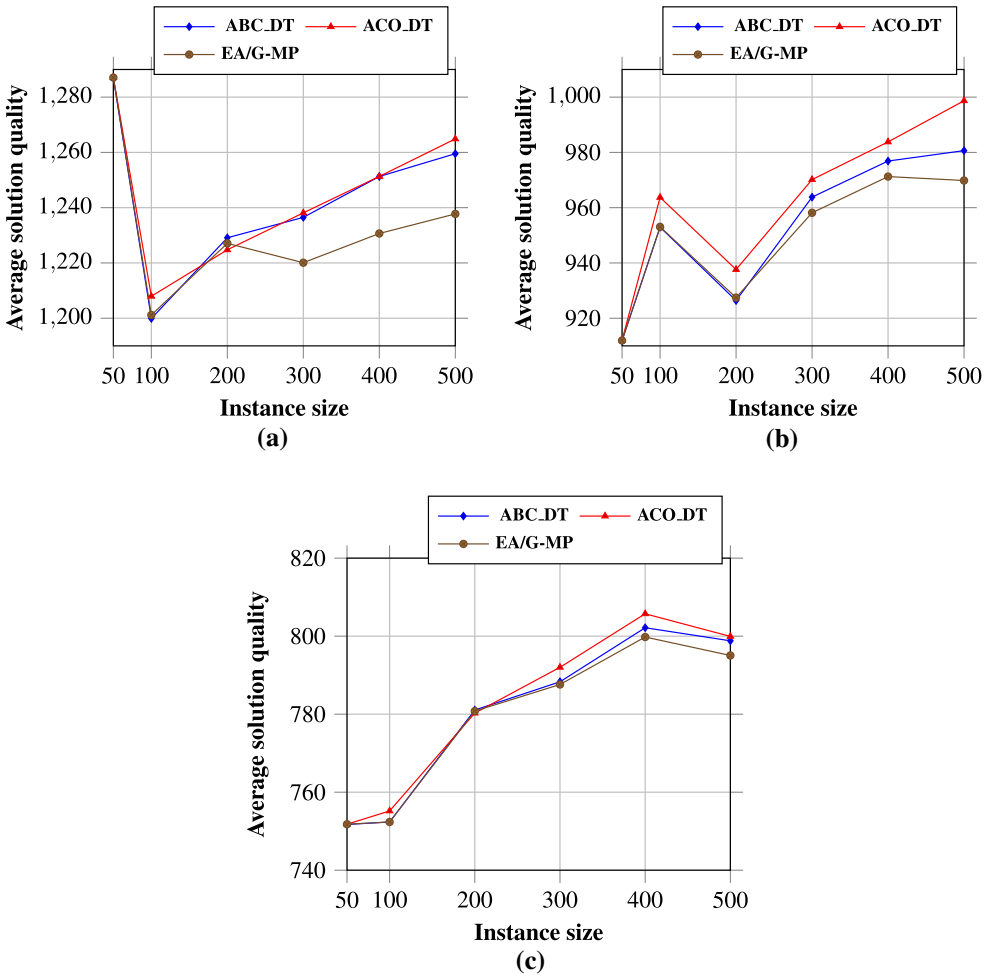

Fig. 5 Average solution quality of ABC_DT, ACO_DT and EA/G-MP over all the instances of each size for different transmission ranges. **a** Transmission range 100 m. **b** Transmission range 125 m. **c** Transmission range 150 m

nodes. On the other hand, EA/G-MP is 3 times slower than ACO_DT on the instances with 50 nodes and slightly slower than ACO_DT on the instances with 100 nodes.

6.4 The overall picture

Figures 4 and 5 graphically compare the average solution quality of various approaches over all the instances of each size viz. 50, 100, 200, 300, 400 and 500 for different transmission ranges. Figure 4 compares different problem specific heuristics, viz. MST-L, Heu_DT1, Heu_DT2, H_DT and M_DT, whereas Fig. 5 compares various metaheuristic approaches, viz. ABC_DT, ACO_DT and EA/G-MP. Figure 4 clearly shows that as the problem size increase, the cost of the dominating tree grows rapidly for MST-L, Heu_DT1 and Heu_DT2, thereby restricting the utility of these three heuristics to small size instances only. Like M_DT, H_DT also scales well, but its results are always inferior on an average to those of M_DT. If we look at Fig. 4a–c together, it can be observed that with the increase in transmission range, the average cost of dominating tree returned by H_DT and M_DT decreases for the same node size, which is also expected theoretically. On the other hand, average cost of the dominating tree returned by other heuristics seem to remain unaffected by the increase in transmission range.

Figure 5 shows that all the metaheuristic approaches scale well with regard to average solution quality as size of the problem increases. Except for instances of size 200 and range 100 m, average solution quality of EA/G-MP is better than ABC_DT and ACO_DT on all other sizes and transmission ranges. Looking at the Fig. 5a–c together, we can observe that the average cost of the dominating tree returned by all the three metaheuristic approaches decreases with increase in transmission range.

Figure 6 graphically compares the average total execution time of ABC_DT, ACO_DT and EA/G-MP over all the instances of each size for the transmission range 100, 125 and 150 m. The average total execution time of ABC_DT and ACO_DT grows rapidly with increase in instance size. Only EA/G-MP scales well with increase in instance size in terms

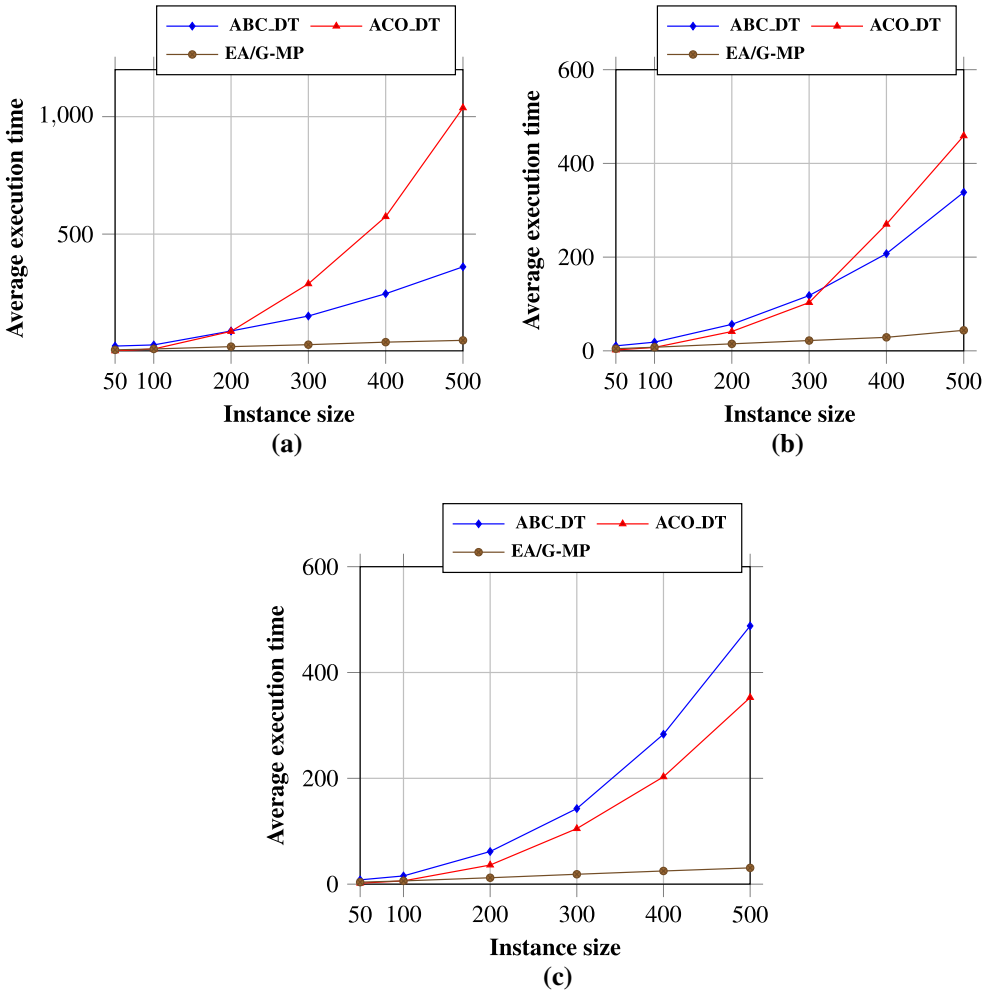

Fig. 6 Average total execution time of ABC_DT, ACO_DT and EA/G-MP over all the instances of each size for different transmission ranges. **a** Transmission range 100 m. **b** Transmission range 125 m. **c** Transmission range 150m

of average total execution time. Some interesting observations can be made if we look at Fig. 6a–c together. Average total execution time of ABC_DT increases with increase in transmission range. On the other hand, average total execution time of ACO_DT decreases with increase in transmission range. For EA/G-MP also average total execution time decreases, but only slightly when compared to ACO_DT.

To get an idea about the effect of increase in transmission range on number of dominating nodes, we have found the best value among average number of dominating nodes returned by ABC_DT, ACO_DT and EA/G-MP on each size for different transmission ranges. We have plotted these best values for each transmission range against node sizes. Resulting plot is shown in Fig. 7. This figure clearly shows that the number of dominating nodes decreases on an average with increase in transmission range. This is also expected theoretically as with increase in transmission range, degree of underlying disk graphs increases, leading to dominating trees with lesser number of nodes.

Though better than other approaches in their respective classes in terms of solution quality, both of our approaches obtain dominating tree with slightly more number of nodes on some instances when we compare M_DT to H_DT, and, EA/G-MP to ABC_DT and ACO_DT. Actually, sometimes even when a pair of node is connected directly by an edge, it may not be the shortest path between them. Instead a shortest path may involve one or more intermediate nodes. As a result, a dominating tree which has lesser cost than another may not have a lesser number of nodes as well. As our approaches favour shortest paths over direct edges more often in comparison to corresponding previous approaches, dominating trees obtained through our approaches are more likely to have more nodes.

7 Conclusions

In this paper we have presented two approaches, viz. a problem specific heuristic called M_DT and a hybrid approach

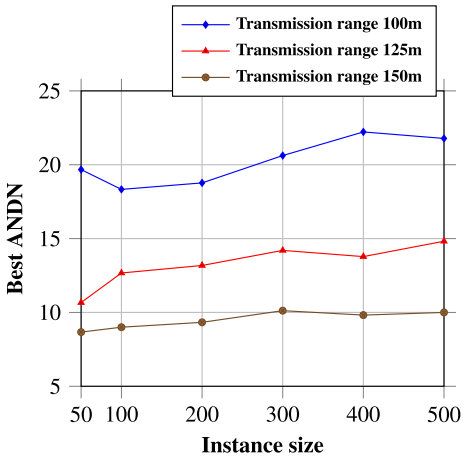

Fig. 7 Best ANDN among ABC_DT, ACO_DT and EA/G-MP on each problem size for different transmission ranges

called EA/G-MP that combines evolutionary algorithm with guided mutation with two improvement procedures for the dominating tree problem (DTP). On 54 benchmark instances of various sizes and transmission ranges, M_DT produced better results in comparison to state-of-the-art problem specific heuristic approaches available in the literature. Similarly, EA/G-MP is able to find better solutions on most of the test instances when compared to two state-of-the-art metaheuristic approaches, viz. ABC_DT and ACO_DT in a much shorter time.

As a future work, we intend to extend our approach to other related $\mathcal{NP}$-hard problems like the connected minimum weight dominating set problem and the capacitated minimum weight dominating set problem. Analogous approaches can be developed for the set covering problem, the minimum weight vertex cover problem and various target coverage problems in wireless sensor networks etc.

Acknowledgments Authors are grateful to two anonymous reviewers for their valuable comments and suggestions which has helped in improving the quality of this paper.

References

Arkin EM, Halldórsson MM, Hassin R (1993) Approximating the tree and tour covers of a graph. Inf Process Lett 47:275–282

Fujito T (2001) On approximability of the independent/connected edge dominating set problems. Inf Process Lett 79:261–266

Fujito T (2006) How to trim an mst: a 2-approximation algorithm for minimum cost tree cover. In: Bugliesi M, Preneel B, Sassone V, Wegener I (eds) Automata, languages and programming, Lecture Notes in Computer Science, vol 4051. Springer, Berlin, Heidelberg, pp 431–442

Guha S, Khuller S (1998) Approximation algorithms for connected dominating sets. Algorithmica 20(4):374–387

Park MA, Willson J, Wang C, Thai M, Wu W, Farago A (2007) A dominating and absorbent set in a wireless ad-hoc network with different transmission ranges. In: Proceedings of the 8th ACM international symposium on mobile ad hoc networking and computing, MobiHoc '07. ACM, New York, pp 22–31

Prim RC (1957) Shortest connection networks and some generalizations. Bell Syst Tech J 36:1389–1401

Raidl G, Julstrom B (2003) Edge-sets: an effective evolutionary coding of spanning trees. IEEE Trans Evolut Comput 7:225–239

Shin I, Shen Y, Thai MT (2010) On approximation of dominating tree in wireless sensor networks. Optim Lett 4(3):393–403

Sundar S, Singh A (2013) New heuristic approaches for the dominating tree problem. Appl Soft Comput 13(12):4695–4703

Thai MT, Wang F, Liu D, Zhu S, Du D-Z (2007) Connected dominating sets in wireless networks with different transmission ranges. IEEE Trans Mobile Comput 6(7):721–730

Thaiand MT, Tiwari R, Du D-Z (2008) On construction of virtual backbone in wireless ad hoc networks with unidirectional links. IEEE Trans Mobile Comput 7(9):1098–1109

Wan P-J, Alzoubi KM, Frieder O (2002) Distributed construction of connected dominating set in wireless ad hoc networks. In: Proceedings of the twenty-first annual joint conference of the IEEE Computer and Communications Societies (INFOCOM 2002), vol 3, pp 1597–1604

Wu J, Hailan L (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, DIALM '99. ACM, New York, pp 7–14

Zhang N, Shin I, Li B, Boyaci C, Tiwari R, Thaiand MT (2008) New approximation for minimum-weight routing backbone in wireless sensor network. Wireless algorithms, systems, and applications, Lecture Notes in Computer Science, vol 5258. Springer, Berlin, Heidelberg, pp 96–108

Zhang Q, Sun J, Tsang E (2005) An evolutionary algorithm with guided mutation for the maximum clique problem. IEEE Trans Evolut Comput 9:192–200