Appl Intell (2015) 43:512–529
DOI 10.1007/s10489-015-0654-1

A hybrid evolutionary algorithm with guided mutation for minimum weight dominating set

Sachchida Nand Chaurasia · Alok Singh

Published online: 23 April 2015

Abstract This paper presents a hybrid evolutionary algorithm with guided mutation (EA/G) to solve the minimum weight dominating set problem (MWDS) which is $\mathcal{NP}$-hard in nature not only for general graphs, but also for unit disk graphs (UDG). MWDS finds practical applications in diverse domains such as clustering in wireless networks, intrusion detection in adhoc networks, multi-document summarization in information retrieval, query selection in web databases etc. EA/G is a recently proposed evolutionary algorithm that tries to overcome the shortcomings of genetic algorithms (GAs) and estimation of distribution algorithms (EDAs) both, and that can be considered as a cross between the two. The solution obtained through EA/G algorithm is further improved through an improvement operator. We have compared the performance of our hybrid evolutionary approach with the state-of-the-art approaches on general graphs as well as on UDG. Computational results show the superiority of our approach in terms of solution quality as well as execution time.

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

1 Introduction

For any graph, a dominating set is a set of nodes such that each node of the graph either belongs to the dominating set or is adjacent to a node in the dominating set. A minimum dominating set is a dominating set with minimum cardinality. Use of dominating sets is widespread in wireless networks for clustering and forming a virtual backbone for routing, and several efficient centralized and distributed algorithms have been proposed for this purpose in the literature, e.g. [2, 3, 5, 10, 16]. However, all these approaches work under the assumption that all nodes are the same in all respects. These approaches may not yield good results in heterogeneous environments where the capabilities of the nodes differ. For example, different nodes may have different residual energy, allotted bandwidth or transmission range. Under these circumstances, instead of minimizing the cardinality of the dominating set, it is more appropriate to minimize the weighted sum of the nodes present in the dominating set. This problem of minimizing the weighted sum of the nodes present in the dominating set is called the minimum weight dominating set problem. Both minimum dominating set problem (MDS) and the minimum weight dominating set problem (MWDS) are $\mathcal{NP}$-hard [6].

Many approaches have been proposed in the literature for MWDS problem, but most of them are for unit disk graphs (UDG) which are used to model wireless networks. In UDG, nodes lie on an Euclidean plane and an edge exists between a pair of nodes, if and only if the Euclidean distance between them does not exceed a fixed threshold. A characteristic feature of UDG is that the number of independent neighbors of a node is polynomially bounded. To form a backbone for adhoc wireless networks, Wang et al. [15] proposed new centralized and distributed approximation algorithm, where node has some cost and a minimum weight dominating set is created. These approaches assume that node weights are smooth, i.e., ratio of weights of adjacent node does not exceed a fixed constant. Dai et al. [4] proposed a

S. N. Chaurasia · A. Singh (✉)
School of Computer and Information Sciences,
University of Hyderabad, Hyderabad 500 046, India
e-mail: alokcs@uohyd.ernet.in

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

$5 + \epsilon$-approximation algorithm to form a minimum weight dominating set for UDG. This goal is achieved in two steps. In the first step, the plane in which the nodes of the UDG lie is partitioned into $k\mu \times k\mu$ blocks, where $\mu = \frac{\sqrt{2}}{2}$ and k is some large integer constant, and in the second step, each block is partitioned into k^2 squares. Finally, results of both the steps are combined to get a minimum weight dominating set. Another $4 + \epsilon$-approximation scheme [20] is based on a polynomial-time dynamic programming algorithm for the minimum weight chromatic disk cover problem. Last two approaches do not assume node weights to be smooth. Zhu et al. [19] proposed another polynomial time approximation scheme to form a minimum weight dominating set for UDG, where the weights of the nodes in the graph are smooth.

However, use of MWDS is not limited to clustering and routing in wireless networks. In fact, MWDS has applications in several diverse domains. It finds application in gateway placement in a wireless mesh network. A polynomial time near-optimal algorithm is proposed in [1] to recursively compute minimum weight dominating set in each iteration, so as to minimize the number of gateways and also satisfy the QoS (quality of service) requirements. During the installation of wavelength division multiplexing optical-networks, MWDS is used to minimize the number of full wavelength converters required in order to cut down the installation cost and to subdue the technological limitations [7]. MWDS was also used in determining the nodes in an adhoc network where intrusion detection software for squandering detection needs to be installed [14]. Shen et al. [12] used the concept of dominating sets in information retrieval for the task of multi-document summarization. In this approach, first a graph is constructed where nodes represent the sentences from various documents and an edge exists between two sentences if they are similar. Then a minimum dominating set is computed over this graph to get a summary of all the documents. The application of MWDS is extended to the task of query selection, so as to harvest the data records from hidden web database efficiently [17]. For this purpose each web database is modeled as an undirected attribute-value graph (AVG) and the task of the optimal query selection for the web database is shown equivalent to solving MWDS in corresponding AVG. In none of the applications cited in this paragraph, we can assume the underlying graph to be UDG. However, all the approximation schemes available in the literature work for UDG only, and therefore, these schemes cannot be used for aforementioned applications. In fact, underlying graphs in these applications vary depending on the situation and as a result only those approaches, which can be applied to any general graph can be used. In the absence of the approximation schemes for general graphs, heuristics and metaheuristics are attractive alternatives.

To the best of our knowledge, only two metaheuristic approaches have been proposed in the literature for MWDS. Jovanovic et al. [8] developed an ant colony optimization (ACO) algorithm which was inspired by the ant-colony optimization approach of Shyu et al. [13] for the minimum weight vertex cover problem and tested it on general graphs. This ACO approach will be referred to as Raka-ACO subsequently. Potluri and Singh [11] proposed four greedy heuristics, a hybrid genetic algorithm and two versions of hybrid ant-colony optimization algorithm for MWDS. The solutions obtained through the genetic algorithm and the ACO algorithms are improved further though a minimization heuristic. The genetic algorithm uses steady-state population replacement method, selects parents using binary tournament selection and employs fitness-based crossover and simple bit flip mutation to produce offspring. Hereafter, this hybrid genetic algorithm will be referred to as HGA. Except for pheromone initialization, the two ACO versions are same in all other respects. Starting with an empty set S, these two ACO versions construct dominating set by performing a random walk on the graph and adding the nodes visited to S until S becomes a dominating set. The next node to be visited is selected based on pheromone concentration on the nodes. The first version of ACO initializes all pheromone trails to the same value without any bias, whereas the second version uses a pre-processing step to initialize the pheromone trails according to the quality and compositions of the solutions generated in the preprocessing step. The first version will be referred to as ACO-LS and the second version as ACO-PP-LS subsequently. All the approaches presented in [11] have been tested on the same general graph instances as used in [8]. In addition, some UDG instances have also been used. HGA, ACO-LS and ACO-PP-LS approaches obtained much better results in comparison to Raka-ACO. In fact, three of the four greedy heuristics presented also outperformed Raka-ACO. As far as the relative performance of HGA, ACO-LS and ACO-PP-LS approaches are concerned, HGA obtained better quality solutions with increase in average node degree and graph size, but at the expense of much larger execution time. There is not much difference in solution quality of ACO-LS and ACO-PP-LS, but ACO-PP-LS is faster than ACO-LS.

In this paper, we present a hybrid approach comprising evolutionary algorithm with guided mutation (EA/G) and an improvement operator for MWDS. EA/G is a relatively recent evolutionary technique developed by Zhang et al. [18] that uses a guided mutation operator to produce offspring (solutions). This operator uses a combination of global statistical information about the search space and the location information of the solutions found so far to generate the offspring. Zhang et al. [18] used EA/G to solve the maximum clique problem. The success of EA/G in solving the standard benchmark instances of the maximum clique problem has motivated us to develop an EA/G approach for MWDS. In addition, we present the modified version of a greedy heuristic proposed in [11] and use it inside our hybrid EA/G approach for the initial solution generation and repairing an infeasible solution. We have compared our hybrid EA/G approach with the approaches presented in [8, 11] on the same benchmark instances as used in [8, 11]. In comparison to these approaches, our hybrid approach obtained better quality solutions in shorter time.

The remainder of this paper is organized as follows: Section 2 formally defines the minimum weight dominating set problem and introduces the notational conventions used in this paper. Section 3 describes the proposed modifications in one of the heuristics of [11]. Section 4 provides an overview of EA/G. Our hybrid EA/G approach is presented in Section 5. Section 6 reports the computational results and provide a comparative analysis of our hybrid EA/G approach vis-à-vis existing approaches for MWDS. Finally, Section 7 outlines some concluding remarks and directions for future research.

2 Problem formulation

In an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, two nodes u and v are called neighbor of each other or adjacent to each other, iff, there exists an edge between them, i.e., $(u, v) \in E$. A dominating set $D \subseteq V$ is a set of nodes such that each node $v \in V$ either belongs to D or is neighbor of at least one node in D. Any node belonging to D is called a dominating node or dominator. Any node not in D is called a dominatee or non-dominating node. The minimum dominating set problem (MDS) seeks a dominating set on G whose cardinality is least among all the possible dominating sets on G. In minimum weight dominating set problem (MWDS), a weight is assigned to each node of V through a weight function $w : V \to \Re^+$ and the goal is to find a dominating set with the minimum sum of node weights. More formally, MWDS seeks $\arg\min\limits_{D \in DS} \sum_{v \in D} w(v)$, where DS is the set of all dominating sets of G. Both MDS and MWDS have been proven to be $\mathcal{NP}$-hard [6]. In fact, MWDS is a generalization of MDS as it reduces to MDS when $w(v) = 1 \forall v \in V$. Important notational conventions used in this paper are given in Table 1. Additional notational conventions will be introduced as and when required.

3 Heuristic

In this section, we present an improved version of a heuristic (viz. heuristic 2) which has been presented in [11]. Actually, four different heuristics have been presented in [11]. The reason for choosing heuristic 2 is that among these four heuristics, it performed the best. In all these heuristics, initially, all nodes are assumed to have color WHITE. Starting with an empty set D, these heuristics build a dominating set by iteratively adding a node to D till it satisfies the property of a dominating set. A node that is added to D is re-colored BLACK and all its neighbors are re-colored GREY. Obviously, with this coloring scheme, a heuristic stops when no WHITE node exists. The manner in which the nodes are selected for inclusion into the partially constructed dominating set leads to different heuristics. We have made two modifications in heuristic 2 of [11]. Suppose D is the the partially constructed dominating set, then $S = V - D$ is the set from which next node is added to the dominating set D. Also suppose $c : V \to \{0, 1\}$ is a function such that $c(v) = 1$, iff color of node v is WHITE at present, 0 otherwise. Our first modification utilizes closed neighborhood instead of open neighborhood while choosing the next node to be added to the dominating set. In heuristic 2 of [11], to determine the next node to be added to the dominating set, the sum of weights of WHITE nodes belonging to open neighborhood of each node $u \in S$ (1) is first calculated and then next node is selected using (2), whereas in our heuristic, sum of weights of WHITE nodes belonging to the closed neighborhood of each node $u \in S$ is first calculated (3) and then next node to be added is determined using (4). The motivation for using the closed neighborhood instead of the open neighborhood lies in the fact that when a node to be added is itself WHITE, its weight should be added to the sum of the weights of its neighboring WHITE nodes because addition of this node in the partially constructed dominating set not only satisfies the property of the dominating set with respect to the neighboring WHITE nodes of this node adjacent but also with respect to this node itself, i.e., once this node is added, each node belonging to the closed neighborhood of this node will be either adjacent to a

Table 1 Notational Convention

Notation	Definition
$w(v)$	Weight of node $v \in V$.
$ON(v) \subseteq V$	$\{u : u \in V$ 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 $v \in V$.
$dc(v)$	$\|ON(v)\|$, i.e., Total number of neighboring nodes of node $v \in V$ or degree of v in G

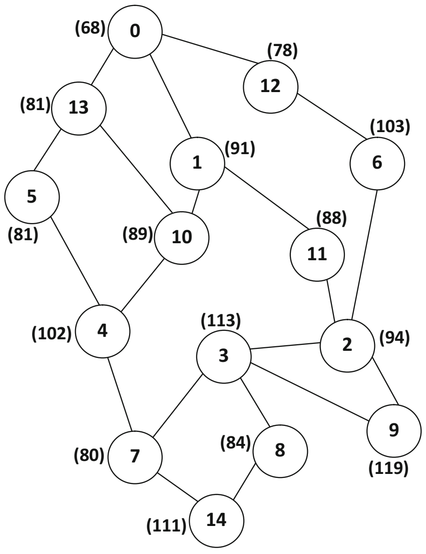

Fig. 1 Input graph. Initially, all nodes are colored WHITE and $weight = 0$

node in the dominating set or itself present in the dominating set.

$$W(u) \leftarrow \sum_{v \in ON(u)} w(v) \times c(v) \tag{1}$$

$$v \leftarrow \arg\max_{u \in S} \frac{W(u)}{w(u)} \tag{2}$$

$$\overline{W}(u) \leftarrow \sum_{v \in CN(u)} w(v) \times c(v) \tag{3}$$

$$v \leftarrow \arg\max_{u \in S} \frac{\overline{W}(u)}{w(u)} \tag{4}$$

Figures 1, 2, 3, 4, 5, 6, 7 illustrate our first modification with the help of an example. Figure 1 shows an undirected

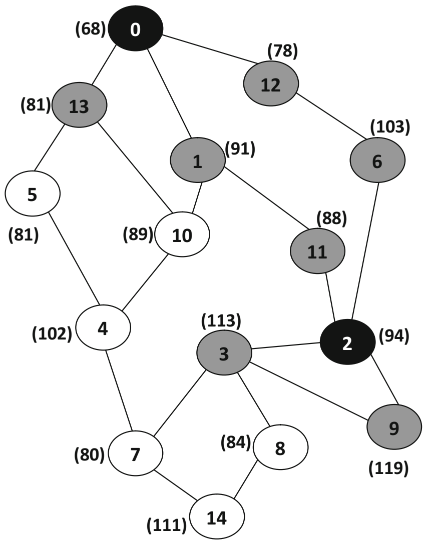

Fig. 3 Node 0 is selected by both heuristics and $weight = 162$

connected weighted graph with 15 nodes where the number inside a circle indicates the ID of the node represented by that circle and the number outside a circle indicates the weight of the corresponding node. All nodes are colored WHITE in this figure to show the initial state. Both the heuristics select node 2 during the first iteration for inclusion into the partially constructed dominating set. Node 2 is colored BLACK now and after that all the neighbors of node 2, viz. nodes 9, 3, 11 and 6 are colored GREY. This situation is shown in Fig. 2. Both the heuristics continue to select identical nodes for three more iterations leading to the further inclusion of node 0, 7 and 13 in that order to the partially constructed dominating set.

Figures 3, 4, 5 depict the partially constructed dominating set at the end of each of these three iterations. The

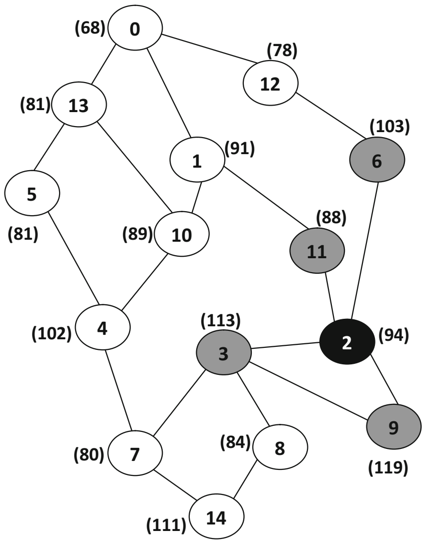

Fig. 2 Node 2 is selected by both heuristics and $weight = 94$

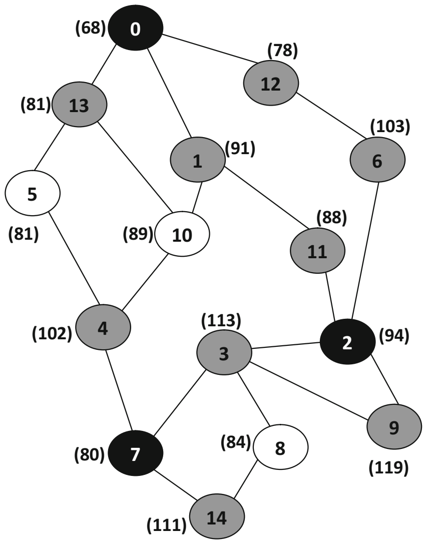

Fig. 4 Node 7 is selected by both heuristics and $weight = 242$

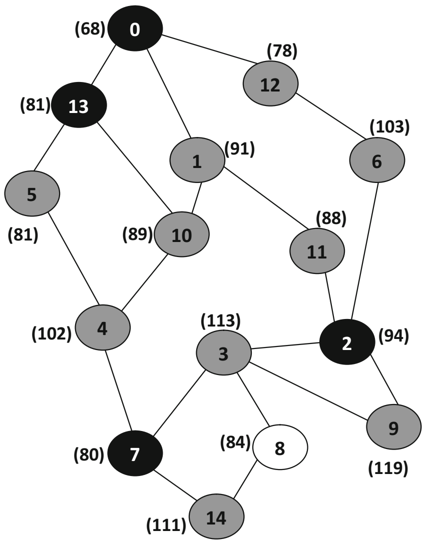

Fig. 5 Node 13 is selected by both heuristics and $weight = 323$

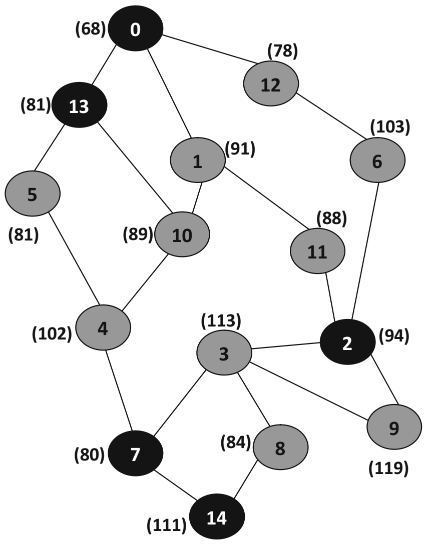

Fig. 7 Node 14 is selected with heuristic of [11] and $weight = 434$

partially constructed dominating set depicted in Fig. 5 has weight 323. Now, the only WHITE node is node 8. Our heuristic selects node 8 as it has the maximum ratio of 1 as per (4) in the next iteration and produces a dominating set with nodes 2, 0, 7, 13 and 8, and total weight 407, which is shown in Fig. 6.

Heuristic 2 of [11], on the other hand, returns GREY node 14 to cover WHITE node 8, and produces a dominating set with nodes 2, 0, 7, 13 and 14, and total weight 434, which is shown in Fig. 7. As open neighborhood is used here, so node 8 ironically has the minimum ratio of 0 and node 14 has the maximum ratio of $\frac{84}{111}$ as per (2), and, that is why node 14 is selected. It is to be noted that even if we swap the weights of node 8 and 14, our heuristic will select the appropriate node according to the changed scenario. In the changed scenario node 8 will have the weight of 111 and node 14 will have the weight of 84. Under this situation node 14 has the maximum ratio of $\frac{111}{84}$ and node 8 again has the ratio 1 as per (4), so our heuristic will correctly select node 14. In this case, however, heuristic 2 of [11] will also select node 14.

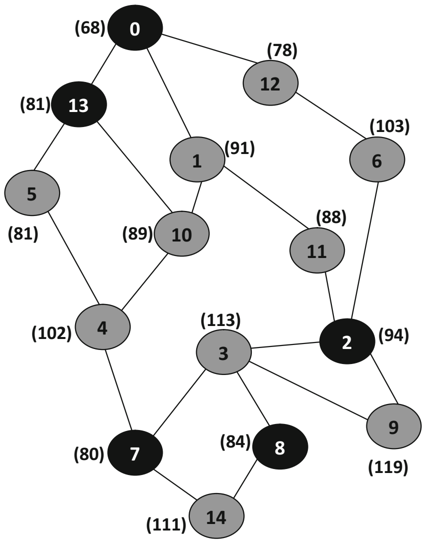

Fig. 6 Node 8 is selected with our heuristic and $weight = 407$

Our second modification is a tie-breaking rule. Actually, when there are more than one node satisfying (4), then the next node to be added is selected using this tie-breaking rule. Suppose S' is the set of nodes satisfying (4), then our heuristic computes the number of WHITE nodes belonging to the closed neighborhood of each node $u \in S'$ (5) and then the next node to be added is determined using (6). In case there is more than one node satisfying even (6), then the next node to be added is selected arbitrarily from among these nodes. The motivation behind this tie-breaking rule lies in experimental observation that using this rule helps in getting better results on more instances.

$$wd(u) \leftarrow \sum_{v \in CN(u)} c(v) \tag{5}$$

$$v \leftarrow \arg\max_{u \in S'} \frac{wd(u)}{w(u)} \tag{6}$$

Hereafter, original heuristic 2 of [11] will be referred to as Heu_A and our improved version as Heu_I.

4 Overview of EA/G

An evolutionary algorithm with guided mutation (EA/G) [18] is a recent addition to the family of evolutionary algorithms. Zhang et al. [18] developed EA/G with the intention of removing, as far as possible, the drawbacks of the genetic algorithms (GAs) and the estimation of distribution algorithms (EDAs). Both GAs and EDAs are population based evolutionary approaches. GAs conventionally use genetic operators, such as crossover and mutation, to produce offspring (solutions) from selected parents. In order to produce offspring, GAs directly utilize the location information of the solutions found so far. However, no global information about the search space is used to produce the offspring. As a matter of fact, this global information can be extracted in GAs using all the solutions produced so far since the beginning. Here, by location information of a solution, we mean the information that can uniquely characterize a solution in the space of all possible solutions, e.g., in case of MWDS, the location information consists of the set of nodes present in a solution as this information can uniquely characterize the dominating set represented by that solution. Rather than using genetic operators, EDAs rely on a probability model to produce offspring. This probability model characterizes the distribution of promising solutions at each generation, and, offspring are produced by sampling this model. This probability model is updated at each generation using the global statistical information extracted from the population members present in that generation. EDAs do not directly utilize the location information of the solutions found so far.

An ideal algorithm should take into account both the factors, viz. global information about the search space and location information of the solution found so far, while producing offspring. EA/G was developed by Zhang et al. [18] exactly with this intention. EA/G employs a mutation operator called *guided mutation (GM)* to produce offspring. In guided mutation, an offspring is created partly by sampling a probability model characterising global statistical information and partly by copying elements from its parent, i.e.,offspring are produced by taking into account the global statistical information as well as the location information of the solutions found so far.

Zhang et al. [18] proposed EA/G approach in the context of maximum clique problem. Computational results demonstrated the superiority of EA/G approach over other evolutionary approaches in solving the maximum clique problem. Motivated by the success of EA/G in solving the maximum clique problem, we have developed an EA/G approach for MWDS, which is described in the next section.

5 Hybrid EA/G approach for MWDS

We have developed a hybrid approach to MWDS combining EA/G algorithm with an improvement operator. The solutions generated through EA/G algorithm are passed through the improvement operator in a bid to improve them further. Hereafter, we will refer to our hybrid EA/G approach with improvement operator as EA/G-IR (EA/G with improvement operator). Before starting our EA/G-IR approach, we precompute the set of neighbors for each node $v \in V$. The salient features of our EA/G-IR approach for MWDS are described in the subsequent subsections.

5.1 Solution encoding

Subset encoding has been used to represent a solution, i.e., each solution is represented directly by the set of the nodes it contains.

5.2 Initial solution

First member of the initial population is always constructed through our heuristic (see Section 3). The remaining members of the initial population are also constructed in the same manner as in our heuristic except for the fact that during each iteration (4) (and (6) if needed) is used for selecting the next node for inclusion in the partially constructed dominating set with probability π_i, otherwise the next node is selected randomly. Each member of the initial population, irrespective of how it got constructed, is passed through an improvement operator in a bid to improve it further.

Pseudo-code for constructing an initial solution except first solution is presented in Algorithm 1.

Algorithm 1: The pseudo-code for generation of each initial solution except first solution

```
// Initially all nodes in $V$ are colored WHITE
$I \leftarrow V$;
$W_n \leftarrow V$;
$WD \leftarrow \Phi$;
while $W_n \neq \Phi$ do
    Generate a random number $r$ such that $0 \leq r \leq 1$;
    if $r < \pi_i$ then
        $v \leftarrow \arg\max_{u \in I} \frac{\overline{W}(u)}{w(u)}$;
    else
        $v \leftarrow random(W_n)$;
    end
    if $v$ is WHITE then
        $W_n \leftarrow W_n \setminus \{v\}$;
    end
    $wd(v) \leftarrow \{u : u \in ON(v) \cap W_n\}$;
    $W_n \leftarrow W_n \setminus wd(v)$;
    $WD \leftarrow WD \cup \{v\}$;
    make $v$ BLACK;
    make all vertices $\in wd(v)$ GREY;
    $I \leftarrow I \setminus \{v\}$;
end
return $WD$;
```

5.3 Initialization and update of the probability vector

Like [18], we have also used a univariate marginal distribution (UMD) model to maintain the distribution of the promising solutions in the search space. 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

V, i.e., number of nodes in the graph. Each $p_i \in p$ gives the probability of node i of V to be present in a dominating set. Our *guided mutation operator* (described in Section 5.4) uses this probability vector to generate offspring at each generation g. This probability vector is initialised using the N_c initial solutions. The pseudo-code for initializing a probability vector p is given in Algorithm 2.

Algorithm 2: The pseudo-code for initializing a probability vector p

```
Compute $n_v \leftarrow$ number of initial solutions containing node $v$, $\forall v \in V$;
Compute $p_v \leftarrow \frac{n_v}{N_c}$, $\forall v \in V$;
```

At each generation g, a parent set *parent(g)* is formed by selecting the best $\frac{N_c}{4}$ 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 3, where $\lambda \in (0, 1]$ is the learning rate, and it governs the contribution of the solutions in *parent(g)* to the updated probability vector p, i.e., the higher the value of λ, the greater the contribution of the solutions in *parent(g)*.

Algorithm 3: The pseudo-code for updating a probability vector p in generation g

```
Compute $n_v \leftarrow$ number of solutions in parent(g) containing node $v$, $\forall v \in V$;
Compute $p_v \leftarrow (1-\lambda)p_v + \lambda \frac{n_v}{\frac{N_c}{4}}$, $\forall v \in V$;
```

5.4 *Guided mutation (GM)* operator

As mentioned already, *GM* operator uses both global statistical information stored in the form of probability vector p and location information of the solutions found so far for generating new solutions. *GM* operator is applied on the best solution b_s in the current population to generate new solutions. The pseudo-code for generating a new solution through *GM* operator is given in Algorithm 4, where β is an adjustable parameter in [0, 1] and D is a new solution constructed through *GM* operator whose nodes are either sampled from probability vector p or directly copied from best solution b_s in *parent(g)*. Parameter β controls the relative contribution of probability vector p and best solution b_s in the generation of a new solution. As the value of β increases, more nodes of D are sampled from probability vector p. On the other hand, as the value of β decreases, more nodes are directly copied from b_s. There is no guarantee of the feasibility of the solution D generated through *GM* operator, i.e., D may not be a dominating set. Therefore, every solution D generated through *GM* operator is checked for feasibility and if D is found to be infeasible, then it is passed through a repair operator which transforms D into a feasible solution. This repair operator is described in the next section.

Algorithm 4: The pseudo-code of generating a solution through *GM* operator

```
$D \leftarrow \Phi$;
foreach node $v \in V$ do
    Generate a random number $r_1$ such that $0 \le r_1 \le 1$;
    if $r_1 < \beta$ then
        Generate a random number $r_2$ such that $0 \le r_2 \le 1$;
        if $r_2 < p_v$ then
            $D \leftarrow D \cup \{v\}$;
        end
    else
        if $v \in b_s$ then
            $D \leftarrow D \cup \{v\}$;
        end
    end
end
return $D$;
```

5.5 Repair operator

The repair operator is applied only on an infeasible solution. Our repair operator is quite different from the repair heuristic of [11] and computationally much more efficient. In the repair heuristic of [11], initially, S is the set of $V \setminus D$ nodes, where V is the set of nodes and D is the infeasible solution to be repaired. The repair heuristic iteratively selects a node from S as per (2) and adds it to D after deleting it from S. This process is repeated till D becomes a dominating set. Actually, set S also contains those GREY nodes which have no neighboring WHITE node. Such GREY nodes can never be selected for inclusion in D, but in every iteration of repair heuristic used in [11], corresponding ratios needed for Eq. 2 are calculated even for these GREY nodes. From the empirical observations, it is found that there are a large number of such GREY nodes in S which unnecessarily increases the computational burden of the repair heuristic used in [11]. To overcome this shortcoming, in our repair operator, we begin by computing the set of WHITE nodes W_n, i.e., W_n contain those nodes which are not the neighbor of any node in the infeasible solution D. Then in every iteration, we initialize set S to the closed neighborhood of a randomly chosen node in W_n. With this initialization of S in every iteration, the cardinality of set S is always much smaller, especially for graphs with a large number of nodes. In every iteration, after initializing S, our repair operator selects a node from set S using (4) (and (6) if needed) and includes it into the infeasible solution D. In addition, after each iteration in our repair operator, we delete all those nodes from W_n which are no longer WHITE. This further helps in speeding-up the repair operator. These iterations continue until no WHITE nodes are left, i.e., until D becomes feasible. In the repair heuristic of [11], with probability p_h (which was set to 0.70), the infeasible solution is repaired as mentioned already, and otherwise it is repaired randomly, i.e., some infeasible solutions are repaired by selecting the node from S in a totally random manner. This is done to maintain the diversity of the population as the other approach was purely greedy. Our repair operator is a proper mix of randomness and greediness. As such, all solutions are repaired as described above, and no infeasible solution needs to be repaired in a purely random manner just for the sake of diversity. The pseudo-code of repair operator is presented in Algorithm 5.

Algorithm 5: The pseudo-code of repair operator

```
// D is an infeasible solution on which repair operator is
    applied
$W_n \leftarrow \{v : v \in V$ and $v$ is WHITE$\}$;
while $W_n \neq \Phi$ do
    $v_r \leftarrow random(W_n)$;
    $S \leftarrow CN(v_r)$;
    $v \leftarrow \arg\max_{u \in S} \frac{\overline{W}(u)}{w(u)}$;
    $wd(v) \leftarrow \{u : u \in CN(v) \cap W_n\}$;
    make $v$ BLACK;
    $D \leftarrow D \cup \{v\}$;
    make all vertices $\in wd(v) \backslash \{v\}$ GREY;
    $W_n \leftarrow W_n \backslash wd(v)$;
end
return $D$;
```

5.6 Improvement operator

All feasible solutions are passed through an improvement operator in a bid to further improve the solution. Our improvement operator removes redundant nodes from a dominating set WD. A node $v \in WD$ is redundant if $CN(v) \subseteq (\cup_{u \in D\backslash\{v\}} ON(u))$, i.e., all nodes in $ON(v)$ are either in WD or the neighbor of a node in $WD\backslash\{v\}$ and v is the neighbor of a node in WD. If node v is redundant, then it can be removed from WD without affecting the dominating set property of WD. Our improvement operator begins by computing the set of redundant node, then an iterative process takes over, where during each iteration, the redundant node, having the highest ratio of its weight to its degree, is deleted from the dominating set, and the set of redundant nodes is recomputed. The iterative process stops when the set of redundant nodes becomes empty. Clearly, our improvement operator follows a greedy strategy. The pseudo-code of our improvement operator is given in Algorithm 6 where WD is the input dominating set and R_s is the set of redundant nodes in WD. This is different from *Minimization heuristic* of [11] where during each iteration, the redundant node to be deleted is either chosen based on highest ratio of weight of a node to its degree or chosen randomly from the set of redundant nodes. In every iteration, the first strategy is used with probability p_r (which was set to 0.60), and otherwise the random strategy is used. Our improvement operator follows the greedy strategy always. We have experimented with random strategy also, but the greedy approach produced better solution in comparison to the combination of greedy and random strategies of [11].

Algorithm 6: The pseudo-code of improvement operator

```
$R_s \leftarrow \{v : v \in WD$ and $CN(v) \subseteq (\cup_{u \in D\backslash\{v\}} ON(u))$;
while $(R_s \neq \Phi)$ do
    $v \leftarrow \arg\max_{u \in R_s} \frac{w(u)}{dc(u)}$;
    $WD \leftarrow WD \backslash \{v\}$;
    $R_s \leftarrow \{v : v \in WD$ and $CN(v) \subseteq (\cup_{u \in D\backslash\{v\}} ON(u))$;
end
return $WD$;
```

5.7 Other features

Zhang et al. [18] puts best $\frac{N_c}{2}$ solutions of *pop(g)* into *parent(g)* and creates $\frac{N_c}{2}$ new solutions in every generation. The population for the next generation is formed using $\frac{N_c}{2}$ solutions in *parent(g)* and $\frac{N_c}{2}$ newly created solutions. On the other hand, in our approach, *parent(g)* is formed using best $\frac{N_c}{4}$ solutions of *pop(g)*, and $\frac{N_c}{4}$ new solutions are produced in every generation. Best $\frac{3N_c}{4}$ solutions of *pop(g)* along with $\frac{N_c}{4}$ newly created solutions constitute *pop(g+1)* in our approach. We have also tried the same strategy as [18], but our strategy gave better results.

If in a generation, all solutions are found to be identical, then we reinitialize the population in the same manner as described in Section 5.2.

The pseudo-code of our EA/G-IR approach for MWDS problem is given in Algorithm 7.

Algorithm 7: EA/G-IR Approach for MWDS

```
1 At generation $g \leftarrow 0$, an initial population pop(g) consisting of $N_c$ solutions ($N_c$
  should be divisible by 4) are generated in a manner described in Section 5.2;
2 Initialize the probability vector $p$ for all nodes using Algorithm 2;
3 Select best $\frac{N_c}{4}$ solutions from pop(g) to form a parent set parent(g), and then
  update the probability vector $p$ using Algorithm 3;
4 Apply the GM operator $\frac{N_c}{4}$ times on the best solution $b_s$ in parent(g) in order to
  generate $\frac{N_c}{4}$ new solutions. A repair operator is applied to each generated
  solution, if necessary, and then an improvement operator is applied to each
  generated solution to improve the solution fitness. Add all $\frac{N_c}{4}$ newly generated
  solutions along with best $\frac{3N_c}{4}$ parent(g) solutions to form pop(g+1). If the
  stopping condition is met, return the dominating set with the minimum weight
  found so far ;
5 $g \leftarrow g+1$ ;
6 If all solutions are identical, then reinitialize pop(g), and go to step 2 ;
7 Go to step 3 ;
```

6 Computational results

In this section we present the computational results of our EA/G-IR approach for MWDS and compare them with the state-of-the-art approaches. In the literature, three sets of instances were used to evaluate the performance of different algorithms, viz. Type I, Type II and unit disk graphs (UDG). Type I and Type II instances were generated by Jovanovic et al. [8]. Type I and Type II instances consist of undirected connected weighted graphs. For Type I instances, the weights of the nodes are randomly distributed in the closed interval [20, 70]. For Type II instances, weights are randomly distributed in the closed interval [1, $dc(v)^2$], where $dc(v)^2$ is square of the degree of the node v. For both Type I and Type II instances, the number of nodes varies from 50 to 1000 and the number of edges varies from 50 to 20000. 10 instances were generated for each combination of the number of nodes ($|V|$) and the number of edges ($|E|$). The number of edges is also varied while keeping the number of nodes same so as to observe the variation in the solution with the number of edges. A total of 530 instances (53 × 10) were generated for each of Type I and Type II datasets leading to a grand total of 1060 instances in these two datasets.

UDG instances were generated by Potluri and Singh [11] using the topology generator of [9]. The nodes are distributed randomly in an area of 1000 × 1000 units. The number of nodes in these instances belong to {50, 100, 250, 500, 750, 1000}. Transmission range of all nodes is fixed to either 150 or 200 units. Similar to Type I and Type II instances, 10 instances are generated for each combination of number of nodes and the transmission range. This leads to a total of 120 UDG instances.

Our EA/G-IR approach for MWDS has been implemented in C and executed on an Intel core i5-2400 processor based system with 4 GB RAM running under Fedora 16 at 3.10 GHz. `gcc 4.6.3-2` compiler with `O3` flag has been used to compile the C program for our approach. In all computational experiments with our approach, we have used

Table 2 Results of Heu_A, Heu_I, Raka-ACO, HGA, ACO-LS, ACO-PP-LS, EA/G-IR and EA/G for small & medium size Type I instances

|V|	**|E|**	**Heu_A**	**Heu_I**	**Raka-ACO**	**HGA**		**ACO-LS**		**ACO-PP-LS**		**EA/G-IR**		**EA/G**	
					Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
50	50	578.30	578.30	539.80	***531.30***	26.86	***531.30***	26.86	532.60	26.26	532.90*	26.65	531.80	26.69
50	100	410.50	410.50	391.90	***371.20***	22.63	***371.20***	22.63	371.50	22.60	371.50*	22.57	371.30	22.59
50	250	197.20	197.20	195.30	***175.70***	8.38	176	8.41	***175.70***	8.38	175.70	8.387	175.70†	8.38
50	500	105.50	105.50	112.80	***94.90***	6.30	***94.90***	6.30	95.20	6.58	94.90	6.30	94.90†	6.30
50	750	72.90	72.90	69.00	***63.10***	6.15	***63.10***	6.15	63.20	6.12	63.30*	6.18	63.30	6.18
50	1000	48.10	48.10	44.70	***41.50***	1.78	***41.50***	1.78	***41.50***	1.78	41.50	1.78	41.50†	1.78
100	100	1168.00	1168.00	1087.20	1081.00	52.40	1066.90	47.85	***1065.40***	47.85	1065.50*	44.35	1065.40†	45.10
100	250	676.70	676.70	698.70	*626.20*	36.17	627.20	30.70	627.40	30.84	**620.00**	33.95	620.80	34.41
100	500	397.30	397.30	442.80	*358.30*	19.98	362.50	19.23	363.20	18.47	**355.90**	16.22	357.00‡	17.33
100	750	294.90	294.90	313.70	*261.20*	13.85	263.50	15.12	265.00	12.80	**256.70**	8.91	257.90‡	11.54
100	1000	235.10	235.10	247.80	*205.60*	8.10	209.20	8.61	208.80	9.24	**203.60**	7.07	204.80‡	6.94
100	2000	122.30	122.30	125.90	108.20	3.12	*108.10*	3.07	108.40	3.47	108.10	2.92	107.90‡	3.21
150	150	1727.60	1724.40^{++}	1630.10	1607.00	64.49	***1582.80***	57.90	1585.20	59.21	1587.40*	59.18	1594.60	58.49
150	250	1340.50	1340.50	1317.70	1238.60	48.56	***1237.20***	45.48	1238.30	44.26	**1224.50**	46.14	1240.30	54.61
150	500	830.40	830.40	899.90	***763.00***	43.02	767.70	45.69	768.60	42.79	**755.30**	41.34	762.30‡	42.91
150	750	611.40	611.40	674.40	***558.50***	27.83	565.00	31.16	562.80	33.39	**550.80**	28.12	552.70‡	27.04
150	1000	488.70	488.70	540.70	***438.70***	23.02	446.80	22.46	448.30	211.00	**435.20**	17.58	439.30	18.72
150	2000	271.50	271.50	293.10	***245.70***	13.94	259.40	9.34	255.60	9.05	**241.50**	12.20	242.10‡	12.66
150	3000	191.10	191.10	204.70	***169.20***	10.39	173.40	10.86	175.20	7.89	**168.10**	9.89	168.00‡	9.80
200	250	2095.10	2095.10	2039.20	1962.10	52.30	1934.30	52.62	***1927.00***	54.60	**1924.10**	50.76	1951.70	65.32
200	500	1380.80	1380.80	1389.40	1266.30	40.07	***1259.70***	45.80	1260.80	50.59	**1251.30**	48.03	1269.50	58.69
200	750	1020.50	1020.50	1096.20	939.80	49.22	***938.70***	44.79	940.10	44.97	**927.30**	48.30	935.30‡	44.04
200	1000	809.60	809.60	869.90	***747.80***	16.41	751.20	16.40	753.70	19.30	**731.10**	20.89	736.50‡	16.87
200	2000	467.50	467.50	524.10	***432.90***	17.46	440.20	16.52	444.70	16.61	**417.00**	12.66	422.40‡	14.35
200	3000	334.80	334.80	385.70	***308.50***	12.28	309.90	11.86	315.20	17.44	**294.70**	11.93	297.80‡	14.41
250	250	2906.00	2906.00	NA	2703.40	80.70	***2655.40***	72.91	***2655.40***	74.63	**2653.70**	60.52	2695.50	66.78
250	500	2040.90	2040.90	NA	1878.80	77.85	1850.30	55.49	***1847.90***	68.49	**1835.30**	60.11	1909.30	74.25
250	750	1533.80	1537.00^{--}	NA	1421.10	40.16	***1405.20***	36.56	1405.50	34.09	**1399.20**	41.69	1436.90	58.39
250	1000	1238.80	1238.80	NA	1143.40	43.43	1127.10	43.03	***1122.90***	26.48	**1114.90**	30.70	1137.30	43.68
250	2000	715.80	715.80	NA	656.60	30.73	***672.80***	29.18	676.40	30.88	**637.50**	35.52	657.40‡	33.90
250	3000	500.40	500.40	NA	***469.30***	22.09	474.10	20.02	476.30	21.77	**456.30**	19.49	459.50‡	19.16
250	5000	328.40	328.40	NA	***300.50***	9.65	310.40	12.18	308.70	11.56	**291.80**	5.16	295.50‡	5.42

NA indicate that result is not available for that instance

$N_c = 100$, $\beta = 0.70$, $\lambda = 0.60$ and $\pi_i = 0.55$. All these parameter values are chosen empirically after a large number of trials. These parameter values provide good results, though they may not be optimal for all instances. We have allowed our approach to execute for 800 iterations on each instance. In each iteration $\frac{N_c}{4}$, i.e., 25 new solutions are generated. So total 800 × 25 = 20000 solutions are generated for each instance which is the same amount generated by the metaheuristic approaches proposed in [11]. ACO of [8] generates 100000 solutions. To get an idea about the role of the improvement operator in finding high quality solutions, we have also implemented another version of our approach where no improvement operator is used. This version will be referred to as EA/G.

We have compared EA/G-IR and EA/G with four different approaches, viz. HGA, ACO-LS & ACO-PP-LS approaches proposed in [11] and Raka-ACO approach proposed in [8].

Like [8] and [11], we have analysed our results according to the instance groups. By an instance group, we mean the 10 instances with the same number of nodes and edges. An instance group is characterized by an ordered pair $(|V|, |E|)$. For this analysis, EA/G-IR is executed only once on each instance like the metaheuristic approaches of [8] and [11]. Tables 2, 3, 4, 5 report the results of Heu_A, Heu_I, Raka-ACO, HGA, ACO-LS, ACO-PP-LS, EA/G-IR and EA/G on Type I and Type II instances, whereas Table 6 does the same for UDG instances except for the fact an instance group here corresponds to 10 instances with the same number of nodes and transmission range. Data for Raka-ACO is taken from [8], where only Type I and Type II instances were used, and therefore, results of Raka-ACO on UDG instances are not reported. Moreover, standard deviations of solution values and average execution times were also not reported in [8] for Raka-ACO and that is why these two quantities for Raka-ACO are not reported. As the C programs for HGA, ACO-LS and ACO-PP-LS were available, so we have re-executed these programs on our system so that execution time of these approaches can be compared directly with our approaches. This re-execution only changed the execution times from those reported in [11]. For each combination of number of nodes $(|V|)$ and number of edges

Table 3 Results of Heu_A, Heu_I, Raka-ACO, HGA, ACO-LS, ACO-PP-LS, EA/G-IR and EA/G for large size Type I instances

|V|	**|E|**	**Heu_A**	**Heu_I**	**Raka-ACO**	**HGA**		**ACO-LS**		**ACO-PP-LS**		**EA/G-IR**		**EA/G**	
					Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
300	300	3481.50	3481.50	NA	3255.20	74.13	***3198.50***	63.82	3205.90	70.39	3213.70*	75.81	3288.30	80.56
300	500	2697.90	2697.90	NA	2509.80	69.21	2479.20	80.75	***2473.3***	84.44	2474.80*	76.95	2574.90	85.18
300	750	2104.30	2099.40^{++}	NA	1933.90	81.23	***1903.30***	55.08	1913.90	64.69	**1896.30**	55.74	1977.60	78.11
300	1000	1688.00	1686.20^{++}	NA	1560.10	35.27	***1552.50***	32.51	1555.80	36.19	**1531.00**	28.43	1604.30	52.85
300	2000	965.60	965.60	NA	*909.60*	22.85	916.80	25.61	916.50	23.08	**880.10**	21.91	904.40‡	26.61
300	3000	701.00	701.00	NA	***654.90***	24.44	667.80	30.00	670.70	28.00	**638.20**	22.15	645.60‡	26.04
300	5000	459.80	459.80	NA	***428.30***	15.07	437.40	16.59	435.90	16.22	**415.70**	10.32	419.60‡	7.44
500	500	5820.50	5820.50	5476.30	5498.30	113.45	5398.30	100.57	***5387.70***	99.53	**5380.10**	89.37	5630.00	153.32
500	1000	4039.80	4038.10^{++}	4069.80	3798.60	92.08	3714.80	103.77	***3698.30***	85.61	**3695.20**	107.47	3990.20	144.51
500	2000	2484.90	2484.90	2627.50	2338.20	77.75	2277.60	60.20	***2275.90***	65.12	**2264.30**	84.53	2427.30	94.45
500	5000	1177.10	1177.10	1398.50	1122.70	30.96	1115.30	35.79	***1110.20***	41.94	**1083.50**	33.27	1128.00	30.84
500	10000	670.50	670.50	825.70	***641.10***	22.18	652.80	11.81	650.90	119.00	**606.80**	11.57	625.70‡	8.87
800	1000	8472.10	8472.10	8098.90	***8017.70***	141.01	8117.60	162.03	8068.00	178.60	**7792.20**	108.34	8426.60	134.45
800	2000	5641.40	5639.70^{++}	5739.90	***5318.70***	130.02	5389.90	151.14	5389.60	144.43	**5160.70**	76.92	5641.40	154.05
800	5000	2739.90	2739.90	3116.50	2633.40	69.07	26160	66.49	***2607.90***	62.02	**2561.90**	37.51	2702.40	53.32
800	10000	1590.60	1590.60	1923.00	1547.70	45.66	***1525.70***	32.40	1535.30	31.00	**1497.00**	33.41	1559.60	28.70
1000	1000	11666.30	11666.30	***10924.40***	11095.20	147.38	11035.50	174.92	11022.90	129.43	**10771.70**	122.15	11599.10	109.77
1000	5000	4168.50	4168.50	4662.70	***3996.60***	73.73	4012.00	81.91	4029.80	85.90	**3876.30**	64.70	4163.50	101.85
1000	10000	2379.80	2379.80	2890.30	2334.70	64.51	2314.90	64.03	***2306.60***	56.03	**2265.10**	51.68	2365.80	58.32
1000	15000	1704.00	1704.00	2164.30	1687.50	28.29	***1656.30***	44.23	1657.40	40.05	**1629.40**	30.04	1682.60	41.04
1000	20000	1351.00	1351.30	1734.30	1337.20	30.97	***1312.80***	22.52	1315.80	24.10	**1299.90**	19.32	1333.70	22.36

NA indicate that result is not available for that instance

($|E|$) (transmission range in case of UDG instances), these tables (Tables 2, 3, 4, 5, 6) report the average solution quality (Mean) and standard deviation of solution values (SD) over 10 instances. All approaches have high standard deviation as these approaches were executed once on each of the 10 instance and the value of the solution varies from one instance to the other. For the two heuristics Heu_A and Heu_I, we have only reported the average solution quality. In all these tables, a result in bold for EA/G-IR indicates that it is better than all the existing approaches in the literature. '*' indicates that EA/G-IR is worse than the best among HGA, ACO-LS and ACO-PP-LS. A result in bold italic indicates that it is the best result among Heu_A, Heu_I, Raka-ACO, HGA, ACO-LS and ACO-PP-LS. '†' indicates that the result of EA/G is equal to the best result among HGA, ACO-LS and ACO-PP-LS and '‡' indicates that the result of EA/G is better than the best result among HGA, ACO-LS and ACO-PP-LS. The results of Heu_I which are

Table 4 Results of Heu_A, Heu_I, Raka-ACO, HGA, ACO-LS, ACO-PP-LS, EA/G-IR and EA/G for small & medium size Type II instances

|V|	|E|	Heu_A	Heu_I	Raka-ACO	HGA		ACO-LS		ACO-PP-LS		EA/G-IR		EA/G	
					Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
50	50	64.90	64.90	62.30	***60.80***	5.71	***60.80***	5.71	***60.80***	5.71	60.80	5.71	60.80†	5.71
50	100	100.10	100.10	98.40	***90.30***	17.21	***90.30***	17.21	***90.30***	17.21	90.30	17.21	90.30†	17.21
50	250	165.40	165.40	202.40	***146.70***	40.54	***146.70***	40.54	***146.70***	40.54	146.70	40.54	146.70†	40.54
50	500	205.50	205.50	312.90	*179.90*	63.14	***179.90***	63.14	***179.90***	63.14	179.90	63.13	179.90†	63.13
50	750	178.10	178.10	386.30	***171.10***	913.00	***171.10***	913.00	***171.10***	913.00	171.10	91.33	171.10†	91.33
50	1000	162.80	162.80	NA	*146.50*	97.30	***146.50***	97.3	***146.50***	97.30	146.50	97.30	146.50†	97.30
100	100	133.00	132.10^{++}	126.50	124.50	15.26	123.60	14.77	***123.50***	14.74	123.50	14.74	123.70	14.74
100	250	229.20	229.20	236.60	211.40	21.76	***210.20***	21.95	210.40	22.43	**209.20**	21.33	209.60‡	21.33
100	500	340.80	340.80	404.80	***306.00***	45.17	307.80	44.70	308.40	44.46	**305.70**	45.27	305.70‡	45.27
100	750	437.80	437.80	615.10	***385.30***	82.76	385.70	83.08	386.30	82.64	**384.50**	82.16	384.50‡	82.16
100	1000	470.30	470.30	697.30	***429.10***	76.13	430.30	79.43	430.30	79.43	**427.30**	77.05	427.30‡	77.05
100	2000	615.40	615.40	1193.90	***550.60***	171.77	558.80	169.7	559.80	171.82	550.60	171.77	550.60†	171.77
150	150	198.00	198.30^{--}	190.10	186.00	18.24	***184.70***	17.99	184.90	18.21	**184.50**	17.70	185.40	17.70
150	250	255.40	255.40	253.90	234.90	21.47	***233.20***	21.02	233.40	20.84	**232.80**	20.52	233.50	20.52
150	500	376.80	376.80	443.20	***350.00***	36.99	351.90	38.66	351.90	38.66	**349.70**	37.17	351.30	37.17
150	750	513.50	513.50	623.30	455.80	77.80	456.90	77.96	***454.70***	78.06	**452.40**	77.44	453.60‡	77.44
150	1000	622.80	622.80	825.30	***547.50***	82.12	551.40	83.67	549.00	81.68	548.20*	82.97	551.00	82.97
150	2000	833.40	833.40	1436.40	***720.10***	180.32	725.70	179.45	725.70	179.45	720.10	180.32	720.10†	180.32
150	3000	888.50	888.50	1751.90	***792.60***	218.03	794.00	220.10	806.20	245.43	**792.40**	217.92	793.20	217.92
200	250	297.40	296.70^{++}	293.20	275.10	21.55	***272.60***	20.12	***272.60***	20.31	**272.30**	20.38	274.80	20.38
200	500	427.00	427.00	456.50	390.70	55.66	388.60	56.42	***388.40***	56.64	388.40	56.80	389.00	56.80
200	750	560.50	560.50	657.90	507.00	60.41	501.70	50.44	***501.40***	50.10	**497.20**	51.94	501.70	51.94
200	1000	668.10	668.10	829.20	***601.10***	52.84	605.90	47.97	605.80	49.16	**598.20**	51.77	599.20‡	51.77
200	2000	1004.70	1004.70	1626.00	893.50	150.64	***891.00***	136.47	892.90	133.15	**885.80**	106.64	885.80‡	106.64
200	3000	1140.80	1140.80	2210.30	***1021.30***	162.54	1027.00	164.38	1034.40	167.73	**1019.70**	162.83	1023.80	162.83
250	250	327.20	327.30^{--}	NA	310.10	19.60	***306.50***	17.89	306.70	17.98	306.50	17.47	312.50	17.47
250	500	485.70	485.30^{++}	NA	444.00	30.35	443.80	32.84	***443.20***	32.47	**441.60**	31.53	448.50	31.53
250	750	635.40	635.40	NA	578.20	42.33	***573.10***	40.82	575.90	42.70	**569.20**	40.67	579.90	40.67
250	1000	749.70	749.70	NA	672.80	59.42	***671.80***	58.56	675.10	60.79	**671.70**	62.27	677.40	62.27
250	2000	1154.30	1154.30	NA	***1030.80***	139.83	1033.90	131.02	1031.50	129.71	**1010.30**	126.98	1019.60‡	126.98
250	3000	1438.80	1438.80	NA	***1262.00***	216.63	1288.50	212.74	1277.00	228.16	**1250.60**	214.08	1253.80‡	214.08
250	5000	1741.10	1741.10	NA	***1480.90***	307.17	1493.60	306.43	1520.10	349.30	**1464.20**	278.38	1470.30‡	278.38

NA indicate that result is not available for that instance

Table 5 Results of Heu_A, Heu_I, Raka-ACO, HGA, ACO-LS, ACO-PP-LS, EA/G-IR and EA/G for large size Type II instances

|V|	|E|	**Heu_A**	**Heu_I**	**Raka-ACO**	**HGA**		**ACO-LS**		**ACO-PP-LS**		**EA/G-IR**		**EA/G**	
					Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
300	300	397.90	397.80	NA	375.60	24.79	***371.10***	23.14	***371.10***	23.44	**370.50**	23.14	380.00	24.62
300	500	516.00	516.00	NA	484.20	39.19	***480.80***	40.13	481.2	40.05	***480.00***	41.24	491.00	40.46
300	750	680.70	679.90^{++}	NA	623.80	52.76	621.60	48.76	***618.30***	48.90	**613.80**	52.25	631.50	55.27
300	1000	828.70	828.70	NA	751.10	75.91	744.90	77.80	***743.50***	74.20	**742.20**	72.28	767.10	75.65
300	2000	1240.60	1240.60	NA	***1106.70***	116.09	1111.60	114.61	1107.50	112.03	**1094.90**	106.64	1107.70	113.15
300	3000	1555.30	1555.30	NA	***1382.10***	125.93	1422.80	153.78	1415.30	167.50	**1359.50**	129.06	1371.90‡	141.51
300	5000	1941.10	1941.10	NA	***1686.30***	294.44	1712.10	291.41	1698.60	300.02	**1683.60**	294.89	1686.20‡	295.82
500	500	668.60	668.20^{++}	651.20	632.90	29.54	627.50	30.06	***627.30***	30.13	**625.80**	30.41	650.90	34.35
500	1000	987.50	987.40^{++}	1018.10	919.20	41.71	913.00	35.69	***912.60***	36.56	**906.00**	42.37	955.70	44.21
500	2000	1508.10	1507.50^{++}	1871.80	1398.20	131.90	1384.90	121.03	***1383.90***	121.77	**1376.70**	116.91	1441.30	139.79
500	5000	2668.10	2668.10	4299.80	***2393.20***	222.03	2459.10	272.38	2468.80	260.35	**2340.30**	210.28	2412.50	248.49
500	10000	3723.30	3723.30	8543.50	***3264.90***	4218.00	3377.90	470.35	3369.40	482.89	**3216.40**	389.63	3236.90‡	400.42
800	1000	1193.10	1192.50^{++}	1171.20	1128.20	48.22	1126.40	51.56	***1125.10***	50.79	**1107.90**	45.43	1187.60	50.73
800	2000	1813.00	1811.60^{++}	1938.70	***1679.20***	74.70	1693.70	80.25	1697.90	80.26	**1641.70**	75.40	1797.50	93.04
800	5000	3321.50	3321.50	4439.00	***3003.60***	204.03	3121.90	227.35	3120.90	229.21	**2939.30**	213.91	3200.40	243.65
800	10000	4725.30	4725.30	8951.10	***4268.10***	379.71	4404.10	380.67	4447.90	371.23	**4155.10**	346.83	4353.50	349.66
1000	1000	1338.50	1335.90^{++}	1289.30	1265.20	30.99	1259.30	33.44	***1258.60***	34.35	**1240.80**	30.45	1333.10	38.86
1000	5000	3596.40	3596.40	4720.10	***3320.10***	221.66	3411.60	228.22	3415.10	209.28	**3222.00**	196.89	3586.20	234.63
1000	10000	5432.60	5432.60	9407.70	***4947.50***	330.77	5129.10	308.66	5101.90	306.17	**4798.60**	291.65	5247.00	295.84
1000	15000	6857.00	6857.00	14433.50	***6267.60***	463.09	6454.60	445.76	6470.60	467.53	**5958.10**	427.56	6364.40	498.45
1000	20000	7785.60	7785.60	19172.60	***7088.50***	659.71	7297.40	598.98	7340.80	604.06	**6775.80**	571.69	7066.80‡	498.82

NA indicate that result is not available for that instance

Table 6 Results of Heu_A, Heu_I, HGA, ACO-LS, ACO-PP-LS, EA/G-IR and EA/G for UDG instances

|V|	**Range**	**Heu_A**	**Heu_I**	**HGA**		**ACO-LS**		**ACO-PP-LS**		**EA/G-IR**		**EA/G**	
				Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
50	150	429.60	429.60	394.30	59.42	***393.90***	58.89	***393.90***	58.89	393.90	58.98	393.90†	58.89
50	200	264.50	264.50	247.80	48.06	***247.80***	48.06	***247.80***	48.06	247.80	48.06	247.80†	48.06
100	150	484.40	484.40	450.40	73.23	***449.70***	73.71	***449.70***	73.71	449.70	73.71	449.70†	73.71
100	200	237.50	237.50	217.30	18.56	***216.00***	18.34	***216.00***	18.34	216.00	18.34	216.00†	18.34
250	150	323.80	323.80	***294.20***	53.42	294.60	52.92	294.50	52.92	**293.70**	52.48	293.90‡	52.86
250	200	135.10	135.10	119.10	17.80	***118.90***	17.95	***118.90***	17.95	**118.70**	18.02	118.70‡	18.02
500	150	183.80	183.80	172.70	36.91	***170.80***	35.88	170.90	35.45	**169.90**	34.66	170.50‡	35.18
500	200	77.40	77.40	68.10	11.29	68.00	11.37	68.00	11.37	68.00	11.37	68.00	11.37
800	150	129.50	129.50	115.90	18.33	114.10	16.58	*114.00*	16.45	**113.90**	16.47	114.50	16.93
800	200	48.20	48.20	42.60	8.04	41.00	7.60	***40.80***	7.47	40.80	7.56	40.90	7.56
1000	150	136.30	136.30	1250	16.17	***121.90***	14.12	121.90	13.64	**121.10**	13.53	121.50‡	13.31
1000	200	53.40	53.40	47.20	6.12	***46.00***	5.72	46.20	5.55	**45.90**	5.70	46.80	6.43

worse than Heu_A are marked with "- -", and those which are better than Heu_A are marked with "++". Average time taken by different approaches is shown in Tables 7, 8, 9, 10, 11. For better analysis of results, instances are divided into two classes, viz. small & medium size and large size. Instances with 50 to 250 nodes are classified as small & medium size instances and instances with 300 to 1000 nodes are classified as large size instances.

After comprehensive analysis of results returned by various approaches, following observations can be made:

1. From Tables 2, 3, 4, 5, it can be observed that EA/G-IR outperformed Raka-ACO in terms of solution quality on all Type I and Type II instance groups by a huge margin. As the graph size increases, the differences in solution quality also increases. As expected, EA/G-IR obtained much better solutions in comparison to Heu_A and Heu_I also on all instance groups including groups of UDG instances.
2. For small & medium size Type I instances, EA/G-IR performed worse than HGA, ACO-LS and ACO-PP-LS on two instance groups ((50, 50) and (50, 750)), worse than best among HGA, ACO-LS and ACO-PP-LS for three instance groups ((50, 100), (100, 100) and (150, 150)), but equal to ACO-PP-LS for instance group (50, 100), better than HGA and ACO-PP-LS for

Table 7 Average execution time of HGA, ACO-LS, ACO-PP-LS, EA/G-IR and EA/G in seconds on small & medium size Type I instances

|V|	|E|	HGA	ACO-LS	ACO-PP-LS	EA/G-IR	EA/G
50	50	0.98	0.44	0.40	0.21	0.15
50	100	0.87	0.33	0.30	0.23	0.15
50	250	0.77	0.20	0.20	0.18	0.12
50	500	0.80	0.20	0.20	0.16	0.13
50	750	0.70	0.10	0.10	0.10	0.13
50	1000	0.70	0.10	0.10	0.10	0.11
100	100	3.45	1.54	1.40	0.76	0.45
100	250	3.30	1.22	1.10	0.72	0.39
100	500	2.61	1.03	0.90	0.60	0.33
100	750	2.02	0.84	0.80	0.52	0.31
100	1000	2.30	0.80	0.80	0.49	0.32
100	2000	2.86	0.70	0.70	0.45	0.35
150	150	9.31	3.94	3.50	1.64	0.81
150	250	8.60	3.55	3.10	1.71	0.78
150	500	6.68	2.66	2.40	1.45	0.67
150	750	5.69	2.33	2.10	1.27	0.61
150	1000	5.90	2.25	2.00	1.11	0.56
150	2000	4.74	1.70	1.70	0.88	0.55
150	3000	4.74	1.71	1.60	0.82	0.59
200	250	17.99	7.63	6.60	3.68	1.26
200	500	14.81	5.85	5.10	3.30	1.11
200	750	13.74	5.18	4.50	2.78	1.01
200	1000	12.45	4.5	4.00	2.39	0.94
200	2000	8.61	3.68	3.40	1.68	0.82
200	3000	8.27	3.27	3.10	1.42	0.81
250	250	31.23	13.96	12.20	4.65	1.85
250	500	27.08	11.35	9.90	4.66	1.67
250	750	26.55	9.65	8.40	4.25	1.52
250	1000	23.34	8.54	7.50	3.69	1.42
250	2000	14.63	6.50	6.00	2.65	1.22
250	3000	13.80	5.70	5.40	2.16	1.15
250	5000	12.92	4.98	4.80	1.85	1.16

instance group (100, 100) and better than HGA for instance group (150, 150). Out of 32 instance groups, average solution quality of EA/G-IR is worse than best among HGA, ACO-LS and ACO-PP-LS on 5 instances, same as best among HGA, ACO-LS and ACO-PP-LS on 4 instances and better than best among HGA, ACO-LS and ACO-PP-LS on remaining 23 instance groups.

3. For small and medium size Type II instances, EA/G-IR performed worse than the best among HGA, ACO-LS and ACO-PP-LS for instance group (150, 1000), but better than ACO-LS and ACO-PP-LS. Out of 32 instance groups, EA/G-IR is worse than best among HGA, ACO-LS and ACO-PP-LS on 1 group, equal to best among HGA, ACO-LS and ACO-PP-LS on 11 groups and better than best among HGA, ACO-LS and ACO-PP-LS on 20 groups.
4. EA/G-IR outperformed HGA, ACO-LS and ACO-PP-LS on all large size Type I and Type II instance groups except for two large Type I instances group (300, 300) and (300, 500), where ACO-PP-LS obtained better solution quality. For both types of instances, difference in solution quality grows with the number of nodes and the degree of the nodes.

It can also be observed that improvement in solution quality by EA/G-IR for small & medium size instances

Table 8 Average execution time of HGA, ACO-LS, ACO-PP-LS, EA/G-IR and EA/G in seconds on small & medium size Type II instances

|V|	|E|	HGA	ACO-LS	ACO-PP-LS	EA/G-IR	EA/G
50	50	1.267	0.40	0.40	0.19	0.16
50	100	1.04	0.30	0.30	0.27	0.15
50	250	1.46	0.34	0.30	0.23	0.14
50	500	0.93	0.20	0.20	0.09	0.14
50	750	1.25	0.25	0.20	0.07	0.14
50	1000	0.95	0.15	0.10	0.06	0.12
100	100	4.12	1.57	1.50	0.86	0.46
100	250	4.47	1.42	1.30	0.92	0.43
100	500	4.90	1.30	1.20	0.78	0.38
100	750	3.74	1.04	1.00	0.70	0.36
100	1000	4.47	1.05	1.00	0.67	0.35
100	2000	4.68	0.96	0.90	0.54	0.39
150	150	12.00	3.70	3.50	1.85	0.82
150	250	12.68	3.75	3.40	2.03	0.82
150	500	11.46	3.27	2.90	1.95	0.75
150	750	10.12	2.74	2.60	1.78	0.70
150	1000	8.88	2.48	2.30	1.61	0.68
150	2000	8.96	2.25	2.10	1.20	0.64
150	3000	29.94	7.00	7.00	1.07	0.68
200	250	21.77	6.90	6.30	4.38	1.31
200	500	22.22	6.50	5.70	4.51	1.22
200	750	22.33	6.08	5.40	4.18	1.14
200	1000	17.92	5.38	4.90	3.89	1.08
200	2000	16.93	4.72	4.30	2.78	0.98
200	3000	18.37	4.422	4.30	2.16	0.95
250	250	41.76	13.29	12.10	5.26	1.90
250	500	43.77	12.48	11.20	6.10	1.80
250	750	39.30	11.72	10.50	6.09	1.70
250	1000	39.05	10.95	10.00	5.89	1.61
250	2000	31.97	8.95	8.50	4.23	1.41
250	3000	29.76	8.50	8.20	3.50	1.35
250	5000	29.69	7.09	6.90	2.59	1.34

Table 9 Average execution times of HGA, ACO-LS, ACO-PP-LS, EA/G-IR and EA/G in seconds on large Type I instances

|V|	|E|	HGA	ACO-LS	ACO-PP-LS	EA/G-IR	EA/G
300	300	52.64	22.27	19.10	8.73	2.38
300	500	49.77	18.72	16.30	7.21	2.18
300	750	44.42	16.21	14.30	6.48	1.95
300	1000	40.54	14.27	12.40	5.70	1.80
300	2000	26.60	10.50	9.50	4.03	1.46
300	3000	22.43	9.26	8.60	3.27	1.34
300	5000	17.12	7.81	7.50	2.59	1.30
500	500	204.83	89.57	77.50	37.52	6.02
500	1000	250.38	100.00	81.00	26.36	5.11
500	2000	145.95	60.13	54.40	25.54	4.05
500	5000	75.35	33.79	30.10	9.02	3.06
500	10000	43.62	25.15	24.40	6.08	2.77
800	1000	841.64	443.92	409.20	129.94	14.63
800	2000	576.34	301.57	292.20	102.09	11.76
800	5000	346.05	165.34	148.90	53.019	7.11
800	10000	162.32	97.28	95.90	31.17	5.12
1000	1000	2193.48	1023.79	922.40	249.82	23.76
1000	5000	626.10	341.12	326.50	107.41	13.03
1000	10000	380.27	207.39	194.70	63.22	9.02
1000	15000	254.64	162.73	150.80	45.86	7.44
1000	20000	174.50	142.40	139.20	36.35	6.57

is less in comparison to large size instances. Actually, for small size instances, the results obtained by all the approaches are either optimal or are very close to the optimal and that is why EA/G-IR either obtained the same solution or was not able to improve them much. On the other hand, for large size instances, existing approaches could not scale well and their solution quality deteriorates, and that is why EA/G-IR obtained noticeable improvement in solution quality on almost all the instances.

5. For UDG instances, out of 12 instance groups, average solution quality returned by EA/G-IR is equal to the best among HGA, ACO-LS and ACO-PP-LS on 6 groups and better than all three on the remaining 6 groups.

   All the observations made so far clearly show the superiority of EA/G-IR over existing approaches in terms of solution quality which again indicates the advantage of using both the global statistical information about the search space as well as location information of the solution generated so far while generating offspring.

6. From Tables 7, 8, 9, 10, 11, we can clearly say that EA/G-IR is much faster than HGA, ACO-LS and ACO-PP-LS. From these tables, it can also be seen that as the size of the graph increases, the average execution time taken by HGA, ACO-LS and ACO-PP-LS grow faster than that of EA/G-IR. The superiority of EA/G-IR over HGA in terms of execution time is due to the use of subset encoding in EA/G-IR in place of bit vector encoding, which is used in HGA, and efficient implementation of repair operator (see Section 5.5). The size of the dominating set is smaller than the total number of nodes present, and as such, the use of subset encoding yields a significant advantage in terms of time over bit vector encoding. ACO-LS and ACO-PP-LS also use subset encoding, but they are slower than EA/G-IR because each solution in ACO-LS and ACO-PP-LS are constructed from scratch by performing a random walk on the graph where the probability vector needs to be updated whenever a node is added to the partially constructed solution.

Combining the observation made here regarding execution times with observations made previously regarding the average solution quality of EA/G-IR compared to HGA, ACO-LS and ACO-PP-LS, we can say that EA/G-IR, in general, returns solutions of better quality

Table 10 Average execution times of HGA, ACO-LS, ACO-PP-LS, EA/G-IR and EA/G in seconds on large Type I instances

|V|	|E|	HGA	ACO-LS	ACO-PP-LS	EA/G-IR	EA/G
300	300	69.70	21.06	19.20	9.01	2.41
300	500	72.50	20.22	18.10	8.83	2.32
300	750	64.33	19.07	17.00	7.57	2.17
300	1000	59.28	17.68	16.00	8.96	2.06
300	2000	55.25	15.31	14.30	6.67	1.72
300	3000	46.13	15.55	13.60	5.41	1.58
300	5000	44.85	12.24	11.70	4.02	1.50
500	500	284.90	81.38	73.90	31.06	6.36
500	1000	268.61	77.31	68.50	28.27	5.69
500	2000	233.33	72.37	65.00	23.41	4.58
500	5000	122.30	51.34	51.00	17.36	3.17
500	10000	118.96	37.41	37.60	10.80	2.38
800	1000	1091.83	300.80	268.30	132.36	14.58
800	2000	927.69	307.11	282.20	111.84	12.49
800	5000	525.34	227.41	224.00	68.14	8.64
800	10000	387.92	148.32	148.20	40.15	6.02
1000	1000	2149.54	574.95	514.20	202.08	23.57
1000	5000	1112.49	464.20	461.30	132.94	15.05
1000	10000	739.45	323.16	324.40	84.82	10.91
1000	15000	617.16	251.67	251.80	61.64	8.96
1000	20000	536.83	212.02	213.80	59.20	7.78

in much less time when compared with HGA, ACO-LS and ACO-PP-LS.

7. Another important point about EA/G-IR is its fast rate of convergence. EA/G-IR on most of the instances reached the best solution after few iterations only. As already mentioned, EA/G-IR generates 20000 solutions for each instance, but on most of the instances, it reached the best solution after generating 300 to 10000

Table 11 Average execution times of HGA, ACO-LS and ACO-PP-LS, EA/G-IR and EA/G in seconds on UDG instances

|V|	Range	HGA	ACO-LS	ACO-PP-LS	EA/G-IR	EA/G
50	150	0.48	0.30	0.30	0.23	0.13
50	200	0.45	0.30	0.30	0.20	0.12
100	150	1.58	1.10	1.10	0.47	0.36
100	200	1.47	0.90	0.80	0.47	0.31
250	150	8.99	6.20	6.10	2.30	1.16
250	200	5.05	5.20	5.20	1.77	1.03
500	150	27.05	23.50	23.40	3.43	2.48
500	200	22.01	19.80	19.70	2.34	2.51
800	150	79.82	63.90	63.50	10.23	2.99
800	200	64.80	51.40	50.90	7.86	5.42
1000	150	115.31	98.40	97.50	13.76	6.71
1000	200	83.61	79.20	78.40	5.42	3.45

solutions. Very few instances took more than 15000 solutions to reach the best solution. Especially on Type II instances, it took much fewer iterations to reach the best solution. On most of Type II instances, it has generated less than 5000 solutions to reach the best solution.

8. As far as comparison between Heu_I and Heu_A is concerned, out of a total of 106 groups of Type I and Type II instances where each group contains 10 instances with the same number of nodes and edges, the average solution quality of Heu_I is worse than that of Heu_A on 3 groups, equal to Heu_A on 87 groups and better than Heu_A on 16 groups. For UDG, Heu_I found the same results as Heu_A for all instances. Heu_I could not improve the results of Heu_A for UDG; the reason could be the fewer edges in these graphs. Overall, the results of Heu_I and Heu_A vindicate our two modifications (Section 3).
9. When we compare EA/G with EA/G-IR, we can see that EA/G is quite capable of finding a high quality solution on its own on small and medium size instances. On the majority of these instances, it is able to find solutions as good as or better than HGA, ACO-LS and ACO-PP-LS, all of which use a local search. There is not much difference in solution quality obtained by EA/G and EA/G-IR on most of these instances. On the other hand, on large instances the benefits of the improvement operator are evident as there is a large difference in solution quality between EA/G and EA/G-IR on all the instances. However, this improvement in solution quality comes at the expense of increased execution time. EA/G-IR is much slower in comparison to EA/G on large instances.

   In Table 2, there are four instance groups where the average solution quality of EA/G is better than EA/G-IR, which seems to be anomalous. However, this is due to the difference in the execution sequence between EA/G and EA/G-IR. Actually, when the solutions obtained by EA/G and EA/G-IR produce offspring through guided mutation, then these may have a different set of white nodes, and as a result, execution sequences in repair operator may be completely different. This difference in execution sequence in some rare cases can produce a solution which is even better than the one obtained through the improvement operator.

7 Conclusions

In this paper, we have presented a hybrid approach called EA/G-IR combining the evolutionary algorithm with guided mutation (EA/G) and an improvement operator for the minimum weight dominating set problem. We have compared the performance of our EA/G-IR approach with the state-of-the-art approaches available in the literature on standard benchmark instances comprising general graphs and unit disk graphs. Computational results clearly show the superiority of EA/G-IR over the state-of-the-art approaches as it is able to find better quality solutions in general in much shorter time.

As a future work, we intend to extend our approach to the connected minimum weight dominating set problem and the capacitated minimum weight dominating set problem. Similar approaches can also be developed for the set covering problem, the minimum weight vertex cover problem and the 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

1. Aoun B, Boutaba R, Iraqi Y, Kenward G (2006) Gateway placement optimization in wireless mesh networks with QoS constraints. IEEE J Sel Areas Commun 24:2127–2136
2. Basagni S (1999) Distributed clustering for ad hoc networks. In: Proceedings ISPAN - 99 International Symposium on Parallel Architectures Algorithms and Networks, pp 310–315
3. Bevan D, Vaduvur B (1997) Routing in ad-hoc networks using minimum connected dominating sets. In: Proceedings of the 1997 IEEE International Conference on Communications (ICC '97). IEEE, pp 376–380
4. Dai D, Yu C (2009) A $5+\epsilon$ -approximation algorithm for minimum weighted dominating set in unit disk graph. Theor Comput Sci 410:756–765
5. Erciyes K, Dagdeviren O, Cokuslu D, Ozsoyeller D (2007) Graph theoretic clustering algorithms in mobile ad hoc networks and wireless sensor networks. Appl Comput Math 2:162–180
6. Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York
7. Houmaidi M, Bassiouni M (2003) K-weighted minimum dominating sets for sparse wavelength converters placement under non-uniform traffic. In: Proceedings International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, vol MASCOTS03, pp 56–61
8. Jovanovic R, Tuba M Simian D (2010) Ant colony optimization applied to minimum weight dominating set problem. In: Proceedings of the 12th WSEAS international conference on Automatic control, modelling and simulation (ACMOS'10), World Scientific and Engineering Academy and Society (WSEAS), Stevens Point. Wisconsin, USA, pp 322–326
9. Mastrogiovanni M (2007) The clustering simulation framework: a simple manual. http://www.michele-mastrogiovanni.net/software/download/README.pdf

10. Nocetti F, Gonzalez J, Stojmenovic I (2003) Connectivity-based k-hop clustering in wireless networks. Telecommun Syst 22(1-4):205–220
11. Potluri A, Singh A (2013) Hybrid metaheuristic algorithms for minimum weight dominating set. Appl Soft Comput 13:76–88
12. Shen C, Li T (2010) Multi-document summarization via the minimum dominating set. In: Proceedings 23rd International Conference on Computational Linguistics, Coling 2010, pp 984–992
13. Shyu S, Yin PY, Lin B (2004) An ant colony optimization algorithm for the minimum weight vertex cover problem. Ann Oper Res 13(1-4):283–304
14. Subhadrabandhu D, Sarkar S, Anjum F (2004) Efficacy of misuse detection in adhoc networks. In: Proceedings of the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, pp 97–107
15. Wang Y, Wang W, Li XZ (2006) Efficient distributed low-cost backbone formation for wireless networks. IEEE Trans Parallel Distrib Syst 17:681–693
16. 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 Communication (DIALM 1999), Seattle, USA, pp 7–14
17. Wu P, Wen JR, Liu H, Ma WY (2006) Query selection techniques for efficient crawling of structured web sources. In: Proceedings of the 22nd International Conference on Data Engineering, ICDE06, p 47
18. Zhang Q, J S ET (2005) An evolutionary algorithm with guided mutation for the maximum clique problem. IEEE Trans Evol Comput 9:192–200
19. Zhu X, Wang W, Shan S, Wang Z, Wu W (2012) A ptas for the minimum weighted dominating set problem with smooth weights on unit disk graphs. J Comb Optim 23(4):443–450
20. Zou F, Wang Y, Xu XH, Li X, Du H, Wan P, Wu W (2011) New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs. Theor Comput Sci 412(3):198–208

Sachchida Nand Chaurasia received the Bachelor of Computer Applications degree from MCRP University, Bhopal, India in 2004 and and the Master of Computer Applications degree from University of Hyderabad, Hyderabad, India in 2010. He is currently pursuing his Ph.D. in Computer Science in the School of Computer and Information Sciences at the University of Hyderabad, Hyderabad, India. He works mainly in the area of evolutionary algorithms.

Alok Singh received the Bachelors and the Masters degrees in Computer Science from Banaras Hindu University, Varanasi, India in 1996 and 1998 respectively. He received the Ph.D. degree in Computer Science from University of Allahabad, Allahabad, India in 2006. He is currently an Associate Professor in the School of Computer and Information Sciences at the University of Hyderabad, Hyderabad, India. His primary research interests lie in the area of combinatorial optimization using heuristic and metaheuristic techniques.