Applied Soft Computing 108 (2021) 107437

Contents lists available at ScienceDirect

Applied Soft Computing

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

Multi-start iterated local search, exact and matheuristic approaches for minimum capacitated dominating set problem

Mallikarjun Rao Nakkala [a], Alok Singh [a,*], André Rossi [b]

[a] School of Computer and Information Sciences, University of Hyderabad, Hyderabad 500 046, Telangana, India
[b] Université Paris-Dauphine PSL, LAMSADE UMR CNRS 7243, Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France

ARTICLE INFO

Article history:
Received 25 May 2020
Received in revised form 16 March 2021
Accepted 19 April 2021
Available online 24 April 2021

Keywords:
Capacitated dominating set
Dominating set
Iterated local search
Integer linear programming
Matheuristic

ABSTRACT

This paper presents a multi-start iterated local search (MS-ILS), an exact approach based on an improved integer linear programming (ILP) model, and a matheuristic combining the ILP approach with MS-ILS through solution merging for the minimum capacitated dominating set (CAPMDS) problem for undirected graphs. This problem is an extension of the well-known minimum dominating set problem, where there is a cap on the number of nodes that each node can dominate, and the goal is to find a dominating set of minimum cardinality where the number of dominated nodes for each dominating node is within this cap. This $\mathcal{NP}$-hard problem has several real world applications in resource constrained environments such as clustering of wireless sensor networks along with selection of cluster heads, document summarization in information retrieval. Computational results on the standard benchmark instances of this problem show the superiority of our approaches over state-of-the-art approaches available in the literature in their respective categories.

1. Introduction

Given an undirected graph $G = (V, E)$ where V denotes the set of vertices or nodes and E denotes the set of edges, a subset D of V is called a dominating set if all the other nodes of V is adjacent to at least one node of D. Nodes belonging to D are called dominator or dominating nodes and the remaining nodes of V are called dominated or dominatee or non-dominating nodes. A dominating node u ($u \in D$) is said to dominate a non-dominating node v ($v \notin D$) and v is said to be dominated by u whenever there is an edge between u and v, *i.e.*, $(u, v) \in E$. Hence, we can alternatively define a dominating set D to be set of nodes such that each node not in D is dominated by at least one node of D. Many different dominating sets are possible for a graph. The minimum dominating set (MDS) problem seeks a dominating set with minimum cardinality. The minimum capacitated dominating set (CAPMDS) problem is an extension of the MDS problem where there is an additional constraint that places an upper limit on the number of non-dominating nodes that a dominating node can dominate. The MDS problem is an $\mathcal{NP}$-hard problem [1], and so is the CAPMDS problem as it generalizes the MDS problem. If the capacity of each node in the graph is greater than or equal to its degree then the CAPMDS problem is same as the MDS problem. Throughout this paper, we will use node and vertex interchangeably.

Applications of the CAPMDS problem are in resource constrained environments where the MDS problem cannot be applied due to capacity restrictions. Mostly, the CAPMDS problem finds applications in allocation of network centres in communication networks, document summarization and clustering. A number of heuristics have been presented and used broadly for clustering of wireless sensor networks [2–4], in information retrieval for summarization of documents [5,6].

A number of approximation schemes [7–12], heuristics [13, 14], metaheuristics [15] and metaheuristics [16] have been proposed in the literature for solving CAPMDS problem and its variants. We have discussed these approaches in the next section.

In this paper, we present three approaches for the CAPMDS problem. Our first approach is a multi-start iterated local search algorithm. The local search is done by a heuristic which is also responsible for generating an initial solution. This heuristic, which is composed of various greedy and random strategies, iteratively constructs a capacitated dominating set by selecting during each iteration not only a dominating node, but also the dominated nodes corresponding to the selected dominating node. Both dominating and dominated nodes are selected with the intention of minimizing the cardinality of the final capacitated dominating set obtained. If the capacity of the selected dominating node remains unutilized even after selecting the dominated nodes then this excess capacity is used to redundantly cover other nodes. Once the dominating set is constructed, its cardinality is minimized

* Corresponding author.
E-mail addresses: mrao877@gmail.com (M.R. Nakkala), alokcs@uohyd.ernet.in (A. Singh), andre.rossi@dauphine.psl.eu (A. Rossi).

https://doi.org/10.1016/j.asoc.2021.107437
1568-4946/

further by iteratively removing the redundant nodes, *i.e.*, the dominating nodes which can be removed from the constructed capacitated dominating set without affecting its feasibility. Our second approach is an exact approach based on an improved integer linear programming (ILP) model that utilizes the concept of strong vertices in addition to other improvements. Our third approach is a matheuristic that combines the multi-start iterated local search with the ILP through solution merging [17]. Computational results show the effectiveness of our approaches in solving the CAPMDS problem via-a-vis state-of-the-art approaches available in the literature in their respective categories. Our approaches obtained better quality solutions in shorter times.

The rest of the paper is organized as follows: Section 2 provides a survey of related works. Section 3 presents our multi-start iterated local search algorithm. Section 4 is devoted to our ILP approach, whereas 5 describes our matheuristic approach. Computational results on benchmark instances are presented and analysed in Section 6. Finally, the paper concludes with Section 7 listing the contributions made and possible direction for future research.

2. Related work

A number of approximation algorithms, heuristics and metaheuristics have been proposed in the literature to solve the CAPMDS problem. In this section, we will provide a survey of these approaches. Bar-Ilan et al. [7] presented a logarithmic approximation algorithm utilizing a greedy strategy for a problem called ρ-dominating set problem. At the abstract level, this problem is the CAPMDS problem with uniform capacity (every node have same capacity) on unit disk graphs (UDGs) [18]. Starting with an empty dominating set C, this algorithm iteratively adds to C, a vertex, whose addition will enable C to cover the maximum number of non-dominating vertices. For determining this, a maximum integer flow problem is solved in each iteration. This process is repeated till no uncovered vertex remains.

Kao and Chen [8] proposed logarithmic approximation algorithms with respect to different capacity assignment models for the CAPMDS problem on general graphs. A distributed approximation algorithm for the CAPMDS problem with uniform capacity on UDGs is proposed by Kuhn and Moscibroda [9] based on the concept of maximum independent set. They further concluded that CAPMDS problem with variable capacity is inherently non-local for general graphs as well as UDGs, whereas the CAPMDS problem with uniform capacity is non-local on general graphs. Cygan et al. [10] developed an algorithm for solving the CAPMDS problem in $\mathcal{O}(1.89^{|V|})$ time based on the concept of maximum matching. They also presented an exponential approximation scheme for the CAPMDS problem. Liedloff et al. [11] further improved the time complexity of the algorithm of [10] to $\mathcal{O}(1.8463^{|V|})$ by utilizing dynamic programming over subsets. Becker [12] developed a polynomial-time approximation scheme for solving the problem of capacitated domination on planar graphs under bounded maximum capacity and bounded maximum demand.

Potluri and Singh [13] proposed three heuristics for the CAPMDS problem, viz. maximum coverage lowest degree heuristic (MC-LDEG), cluster-subdivision heuristic (C-SUBD) and relax minimum independent heuristic (DS-RELAX). MC-LDEG can be considered as an extension of the optimal approximation algorithm for MDS problem [19] to solve CAPMDS problem. This heuristic uses a concept, which we call *effective capacity* (explained in Section 3), primarily to select dominating nodes. C-SUBD and DS-RELAX both were based on the concept of maximum independent set (MIS), and both make use of MC-LDEG to extend the dominating set obtained by computing a MIS into a capacitated dominating set. Performance of these heuristics have been evaluated on general graphs as well as UDGs with both variable and uniform capacities. Overall, MC-LDEG heuristic performed the best, though C-SUBD heuristic performed the best for UDGs with low uniform capacity.

Potluri and Singh [15] proposed two hybrid metaheuristic approaches based on ant colony optimization (ACO) and genetic algorithm (GA) for the CAPMDS problem, where ACO and GA are hybridized with a minimization heuristic that removes the redundant nodes. Furthermore, 20% of the initial population of GA is seeded with solutions returned by MC-LDEG heuristic, and pheromone values in ACO are initialized based on 250 solutions generated through MC-LDEG heuristic. Performance of these two metaheuristic approaches are tested on the same instances as used in [13] against MC-LDEG. These metaheuristic approaches offer significant advantage over MC-LDEG when capacities of nodes are small. However, the difference in performance declines when capacity increases. Similar effect is observed with respect to degree when capacity remains the same. The two metaheuristic approaches performed quite similar as far as their relative performance is concerned.

Li et al. [14], proposed a novel local search algorithm to address the CAPMDS problem. They used a dynamic scoring method based on vertex penalizing to iteratively build a capacitated dominating set. The node with the highest score is selected as a dominating node and in case of a tie in score, the node with highest capacity is selected. Two-mode (greedy strategy and random strategy) dominated node selection method is used by them to yield more favourable solutions. They also used an intensification scheme at the end of every iteration after dominating and dominated nodes are selected to further improve the partial solution by adding more dominated nodes at the expense of reducing the degree of redundancy of redundantly covered nodes. Computational results show the superiority of this approach, which was called LS_PD, over ACO, GA and MC-LDEG approaches.

Recently, Pinacho-Davidson et al. [16] developed a matheuristic called construct, merge, solve and adapt (CMSA), where an ILP model is iteratively solved on a reduced set of vertices V' whose composition changes dynamically during each iteration. Initially, V' is initialized to $\emptyset$. Then during each iteration, this set of vertices is updated by building a capacitated dominating set C in each iteration through a GRASP based heuristic utilizing the concept of effective capacity. The vertices belonging to C are added to V' and ILP model is solved to obtain a new CAPMDS solution C'. The vertices which are in V' over a certain maximum number of iterations without appearing in solution C' are deleted from V'. This process is repeated again and again for fixed amount of time and best solution found during this time is returned as the solution found by CMSA approach. Computational results show the superiority of CMSA over LS_PD, ACO, GA and MC-LDEG approaches albeit at the cost of high computational time.

3. Multi-start iterated local search approach for the CAPMDS problem

To address the CAPMDS problem, we have developed a multi-start iterated local search approach. Iterated local search (ILS) is a simple and powerful metaheuristic for solving combinatorial optimization problems [20,21]. ILS maintains a single solution which is improved in an iterative manner. ILS possess several desirable features like simplicity, ease of implementation, robustness, and effectiveness. ILS comprises four principal components, viz. initial solution generation, local search, perturbation procedure and acceptance criterion. ILS starts with the generation of an initial solution which is improved through a local search. Then this solution becomes the current solution and an iterative process ensues. During each iteration, the current solution is perturbed, and the

Algorithm 1: Basic ILS framework

Input: An instance of the problem under consideration and parameters for ILS
Output: Best solution found
$S \leftarrow$ Generate_Initial_Solution();
$S \leftarrow$ Local_Search(S);
best $\leftarrow S$;
while *termination criteria not met* **do**
  $S_1 \leftarrow$ Perturb(S);
  $S_1 \leftarrow$ Local_Search(S_1);
  $S \leftarrow$ Acceptance_Criteria(S, S_1);
  if *S is better than best* **then**
    best $\leftarrow S$;
return *best*;

local search is applied on the perturbed solution to obtain a new solution. Then depending on the acceptance criteria, this new solution may replace the current solution. Common acceptance criteria are mandatorily replacing the current solution with the new solution, and, replacing only if the new solution is better than the current one. The latter criteria yields a first improvement strategy, whereas the former one results into a random-walk sort of strategy. This iterative process continues till the termination condition is met. Already, ILS has been used to solve numerous combinatorial optimization problems, e.g. [22–27], where it has shown its effectiveness in comparison to other state-of-the-art metaheuristic approaches. Algorithm 1 provides the pseudo-code of basic ILS framework.

Our ILS based approach makes multiple starts and the overall best solution among the best solutions found in each start is returned as the solution found by our approach. The local search in our approach is done by a heuristic which is also responsible for generating an initial solution. This heuristic, which is a proper mix of various greedy and random strategies, and, which we call H-MECU (Heuristic for maximum effective capacity utilization) is described first. Local search immediately after initial solution generation (as in Algorithm 1) is not required in our ILS approach as same heuristic is used for local search as well as initial solution generation. Acceptance criteria used in our ILS approach is to replace the current solution with new solution only when new solution is better than the current solution. We have represented a dominating set directly by the set of dominating nodes. Like [15], we are also maintaining a count called reference count for each node (irrespective of whether it is dominating or dominated) in the input graph G, regarding the number of dominating nodes in the solution covering it, and this count is updated for each affected node whenever there is a change in the set of dominating nodes. We have followed the convention that a dominating node covers itself while defining these reference counts. The concept of reference count aids in redundant node removal (Section 3.1.3).

3.1. Heuristic (H-MECU) for the CAPMDS problem

This heuristic builds a capacitated dominating set in an iterative manner (iterations of H-MECU should not be confused with the iterations of the multi-start iterated local search). During each iteration, H-MECU selects a dominating node and corresponding dominated nodes on the current graph G_c. For initial solution generation, this heuristic begins with the empty solution, and the current graph G_c initialized to input graph G. When this heuristic is used for local search, this heuristic begins with the partial solution and the current graph G_c both returned by the perturbation procedure (described in Section 3.2). During each iteration, this heuristic selects a dominating node and the corresponding dominated nodes in such a manner that the selected dominating node can utilize its capacity as much as possible and lower degree nodes can be avoided from becoming dominating in a bid to decrease the cardinality of the final solution. This point will be clear when we discuss the selection of dominating and dominated nodes. At the end of the every iteration, the selected nodes (both dominating and dominated) are removed from the current graph G_c, and then the degree of nodes in G_c are updated. Then another iteration begins. This process continues until a feasible solution has been obtained. This solution is improved further by the redundant node removal procedure. Important definitions and notational conventions in the context of our heuristic are presented in Table 1. Our heuristic uses the concept of effective capacity which was introduced in [13] in the context of MC-LDEG heuristic though it was not given any name there. The procedures for selecting dominating nodes, dominated nodes and redundant node removal are described below.

Table 1
Important notational conventions.

Symbol	Meaning
G_c	Current graph
V_c	Set of nodes present in G_c
E_c	Set of unordered pair of nodes present in G_c
$d_G(v_i)$	Degree of node v_i in G
$d_{G_c}(v_i)$	Degree of node v_i in G_c
$C(v_i)$	Capacity of node v_i
$EC(v_i)$	Effective capacity of node v_i in G_c which is defined to be $\min(C(v_i), d_{G_c}(v_i))$
$N_{EC_{sum}}(v_i)$	Sumtotal of the effective capacities of adjacent nodes of v_i in G_c

3.1.1. Selection of dominating node

A dominating node is selected from nodes in G_c through one of the following three steps. In each of these steps, unless stated otherwise, ties are broken arbitrarily wherever they occur, and a higher numbered step is used only when all the lower numbered steps fail to find a dominating node.

1. **Isolated Node:** If a node $v_i \in G$ is an isolated node, then v_i must be a part of the solution and is added to the solution in the starting iterations. If $v_i \in G$ is not an isolated node, but after certain number of iterations v_i becomes an isolated node in G_c, then a non-dominating node $v_j \in G$ with highest effective capacity in G which is adjacent to v_i in G is inserted into the graph G_c. The graph G_c is updated by connecting v_j to the nodes present in G_c, which are adjacent to v_j in G and current iteration ends without adding any dominating node to the solution (and obviously without selecting any dominated nodes). On the other hand, if no non-dominating adjacent node is available, then the node v_i itself is made dominating and added to the candidate solution. The motivation for this step comes from the fact that instead of an isolated node, it is always better to add a non-isolated highest effective capacity adjacent node to the candidate solution if we are interested in minimizing the cardinality of the obtained solution as it can dominate more nodes. The reason for not immediately making newly inserted non-dominating adjacent node v_j dominating is to increase the randomness of the heuristic. Our heuristic always search for isolated nodes in G_c in natural order of nodes and the first isolated node found is processed. If no isolated node is found in this step then we proceed to step 2, otherwise process exits.

2. **Node with degree one:** If the graph G_c have at least one node with degree one, then its only adjacent node is selected as a dominating node. As already mentioned, ties are broken arbitrarily by selecting one node randomly for processing in case there are more than one node with degree one. This step is also motivated by the argument analogous to the previous step. If there is no node in G_c with degree one, we proceed to step 3, otherwise process exits.
3. **Multiple adjacent nodes:** If there is no isolated node and degree one node present in the current graph G_c, then a node v_i is selected as a dominating node based on its effective capacity, and adjacent effective capacity sum in G_c ($EC(v_i)$, $N_{EC_{sum}}(v_i)$ as defined in Table 1). A node having highest effective capacity in G_c is selected as a dominating node. If there are two or more nodes with highest effective capacity, then a node having smallest adjacent effective capacity sum in G_c among these nodes is selected as a dominating node. If still there are two or more candidate nodes, then ties are broken arbitrarily.
   The concept of effective capacity aids in minimizing the number of nodes in the dominating set as a node can have high capacity, but less degree, thereby effectively reducing its capacity to degree. Selecting such a high capacity node may lead to a very poor solution. In case of tie on effective capacity, we are selecting a node with minimum adjacent effective capacity sum. This is done to avoid a lesser effective capacity node from becoming dominating at a latter iteration, thereby hampering the minimization of cardinality of capacitated dominating set.

3.1.2. Selection of dominated nodes

Dominated nodes are selected greedily. The number of dominated nodes selected for a dominating node should not exceed the capacity constraint of that node. If the capacity of a dominating node is greater than or equal to its degree in G_c, then all its adjacent nodes in G_c are made dominated. Otherwise, only the adjacent nodes equal to capacity of the dominating node can be made dominated. In this case adjacent nodes are selected one-by-one in non-decreasing order of their degree in G_c (Ties are broken arbitrarily) till their number reaches the capacity of the newly added dominating node. Here lower degree nodes are preferred for making them dominated so as to avoid them from becoming dominating during latter iterations. If the dominating node has some excess capacity after covering all its adjacent nodes in G_c, then it utilizes this excess capacity in covering other nodes which are adjacent to it in G. These nodes are already marked dominating or dominated in the partially constructed solution and are selected randomly. For each dominating node v, we are also maintaining a list ℓ_v of the nodes that it covers. This aids in further minimizing the cardinality of a capacitated dominating set through redundant node removal which is described in the next section.

3.1.3. Redundant node removal

A solution constructed by H-MECU is subjected to redundant node removal procedure. If D is a capacitated dominating set, and even after removal of a node $v_i \in D$, the set D remains the capacitated dominating set, then the node v_i is termed as a redundant node. Obviously, a dominating node v_i is a redundant node if its own reference count and reference count of all the nodes it covers is at least two. To remove the redundant nodes from the generated solution, we have used a strategy which is a modified version of the strategy used in Potluri and Singh [15]. This strategy is a mix of random and greedy strategies, and removes the redundant nodes one-by-one in an iterative manner. It begins by computing the set of redundant nodes, and then during each iteration, one of the random or greedy strategy is used to remove a redundant node. The random strategy is used with probability p_r (otherwise, greedy strategy is used), where p_r is a parameter to be determined empirically. In the random strategy, like [15], a node from the set of redundant nodes is selected randomly and removed from the dominating set. In the greedy strategy, unlike [15], the redundant node with lowest degree (in G) is removed. If there are two or more nodes with lowest degree then ties broken arbitrarily. Actually, greedy strategy of [15] removes a redundant node v_i with largest residual capacity ($max(C(v_i) - d_G(v_i), 0)$), and only in case of a tie, node with lowest degree is removed. For variable capacity graphs, a redundant node with capacity 10 and degree 7 (residual capacity 3) is better than a node with capacity 6 and degree 5 (residual capacity 1) as former node is covering more nodes than the latter node due to higher degree, and it is more likely that a capacitated dominating set with smaller cardinality is obtained in case we retain the former instead of latter. Hence, our approach selects the former node. After removal of a redundant node, reference counts of all affected nodes are updated and the set of redundant nodes is recomputed. Then another iteration begins. This process continues until there is no redundant node in the dominating set.

Algorithm 2 provides the pseudo code of H-MECU. As H-MECU plays the double role of generating an initial solution and performing the local search, it takes as input a partial solution S, input graph G, and the current graph G_c corresponding to S. For initial solution generation, $S = \emptyset$ and $G_c = G$. On the other hand, for performing the local search, S and G_c both are returned by the perturbation procedure (Section 3.2).

Algorithm 2: Pseudo-code of H-MECU

```
Input: A partial solution S, input graph G = (V, E) and the current graph G_c = (V_c, E_c)
Initialize reference count r_v for all nodes v ∈ V as per S;
while (V_c ≠ ∅) do
    if (an isolated node u exists in G_c) then
        if ((d_G(u) = 0) ∨ (all nodes adjacent to u in G are already in S)) then
            S := S ∪ {u};
            v := u;
        else
            v := argmax_{x:(x∈V)∧((u,x)∈E)} min(C(x), d_G(x));
            V_c := V_c ∪ {v};
            E_c := E_c ∪ {(v, x) : ((v, x) ∈ E) ∧ (x ∈ V_c)};
            continue;
    else if (a degree one node exists in G_c) then
        Compute set D_1 of all degree one nodes in G_c;
        v := Select a node adjacent to a node in D_1 randomly;
        S := S ∪ {v};
    else
        ECmax := 0;
        N_ECmin := +∞;
        for (u ∈ V_c) do
            if ((EC(u) > ECmax) ∨ ((EC(u) = ECmax) ∧ (N_ECsum(u) < N_ECmin))) then
                v := u;
                ECmax := EC(u);
                N_ECmin := N_ECsum(u);
        S := S ∪ {v};
    Select the set of dominated nodes Y corresponding to v in G_c;
    if ((min(C(v), d_G(v)) − |Y|) > 0) then
        Utilize this excess capacity of v to cover other adjacent nodes of v in G;
    Update the reference count r_u of each affected node u in G;
    Update G_c = (V_c, E_c) by deleting nodes in Y ∪ {v} along with their corresponding edges;
R := {v : (v ∈ S) ∧ (r_v(v) ≥ 2) ∧ (r_u(u) ≥ 2∀u ∈ ℓ_v)};
while (R ≠ ∅) do
    if (u_01 ≤ p_r) then
        v := Choose a node from R randomly;
    else
        v := argmin_{u∈R} d_G(u);
    S := S \ {v};
    R := R \ {v};
    Update the reference count r_u of each affected node u in G;
    R' := {v : (v ∈ R) ∧ (r_v(v) ≥ 2) ∧ (r_u(u) ≥ 2∀u ∈ ℓ_v)};
    R := R';
return S;
```

The working of H-MECU is illustrated with the help of an example in Fig. 1. Input graph G is given in Fig. 1(a). The graph G is an uniform capacity graph, where every node in the graph have the capacity 2. Hence, any dominating node in a candidate

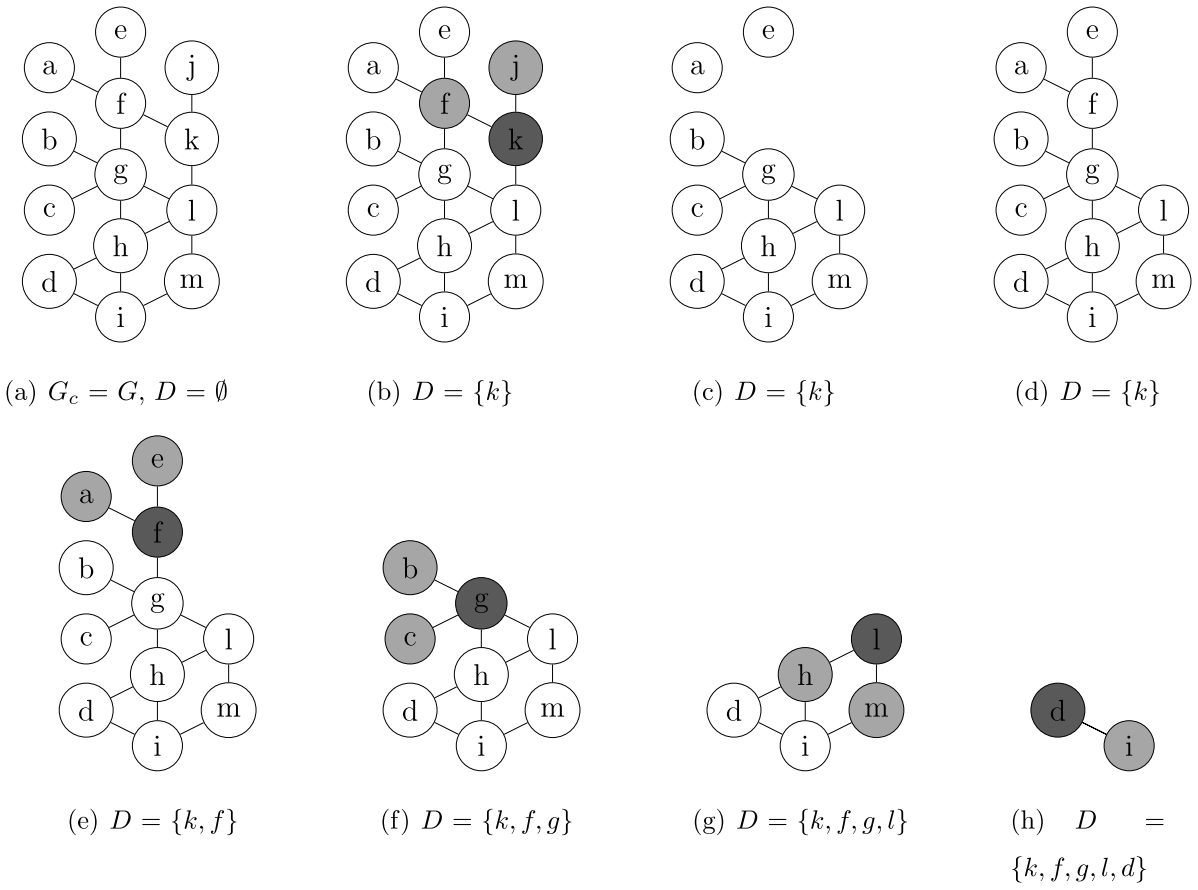

Fig. 1. Illustrating H-MECU on a uniform capacity graph G where every node has capacity 2.

solution can dominate at most two adjacent nodes. The H-MECU begins by initializing current graph G_c to G and the candidate solution D to $\emptyset$ (empty set).

In the first iteration, H-MECU first searches for the isolated nodes in G_c. However, there is no isolated node in G_c, then this approach search for the degree one nodes in G_c. There are five nodes with degree one, viz. a, b, c, e, and j and their corresponding adjacent nodes are f, g, g, f, and k respectively. Node j is selected randomly for processing and its adjacent node k is added to the candidate solution D. In G_c, node k has three adjacent nodes viz. j, l and f, and their degrees are 1, 4, and 4 respectively. Here node j have the lowest degree, so it is made a dominated node. Nodes l and f have the same degree, so randomly node f is made a dominated node. This situation is shown in Fig. 1(b), where newly determined dominating node is shown in dark grey colour and newly determined dominated nodes are shown in light grey colour. Now, the dominating node k along with its dominated nodes j and f are deleted from G_c, thereby creating two isolated nodes a and e (Fig. 1(c)).

In the second iteration, there are two isolated nodes a and e in G_c. Among these two nodes, a being lower in natural (lexicographic) order is selected for processing. Then it checks for a non-dominating node in G with the highest effective capacity that is adjacent to a. In G, a has only one non-dominating adjacent node f. The node f is inserted into G_c and is connected to all the nodes present in G_c to which it is adjacent in G (Fig. 1(d)). This iteration ends without adding any node to D.

In next iteration, no isolated nodes exists in G_c, so H-MECU searches for degree one nodes. There are four nodes with degree one, viz. a, b, c, e and their corresponding adjacent nodes are f, g, g and f respectively. Now, node a is selected randomly for processing and its adjacent node f is added to the candidate solution D. Node f has three adjacent nodes in G_c, viz. a, e and g, out of which a and e have degree 1 and g has degree 5, So a and e are now made dominated nodes. This situation is shown in Fig. 1(e), where newly determined dominating node is shown in dark grey colour and newly determined dominated nodes are shown in light grey colour. Now, the dominating node f along with its dominated nodes a and e are deleted from G_c.

In the next iteration, isolated nodes are not found in G_c then it searches for degree one nodes. There are two nodes with degree one, viz. b and c, and node g is the adjacent to both. Node c is randomly selected for processing and its adjacent node g is added to the candidate solution. Node g has four neighbours, viz. b, c, h and l with degree 1, 1, 4 and 3 respectively, so nodes b and c are made dominated. This situation is shown in Fig. 1(f). The dominating node g along with its dominated nodes b and c are now deleted from G_c.

In the next iteration, isolated nodes and degree one nodes are not found in the graph G_c. Then it tries to select a node based on the effective capacity. However, all the nodes in G_c have the same capacity, and have degree greater than or equal to the uniform capacity, so effective capacity is same for all the nodes. Hence, adjacent effective capacity sum of nodes is utilized for breaking the tie. The adjacent effective capacity sum of nodes d, h, i, l and m in G_c are 4, 6, 6, 4, and 4 respectively. The nodes d, l and m have the same minimum value, so the node l is selected randomly, and its two adjacent nodes h and m are made dominated. This situation is shown in Fig. 1(g). Now, the nodes l, m, and h are deleted from G_c.

In the next iteration, only two nodes d and i with degree one are present in G_c. The node i is selected at random for processing and its only adjacent node d is made dominating and added to the candidate solution D. Obviously, node i will become dominated.

This situation is shown in Fig. 1(h). Now, nodes d and i are deleted from G_c. The graph G_c is now empty and a feasible solution $D = \{k, f, g, l, d\}$ to the CAPMDS problem has been obtained. Now, this solution is checked for redundant nodes. However, there is no redundant node in D, so H-MECU terminates and D is returned as a solution found by it.

3.2. Perturbation procedure

Perturbation procedure perturbs the current solution by copying only fraction of nodes from the current solution to the new candidate solution. This procedure copy $\lfloor \frac{|D|}{A} \rfloor$ dominating nodes from the current solution D to the new candidate solution, where A is a parameter to be determined empirically. This procedure begins by initializing the current graph G_c to the input graph G and new candidate solution to the empty set, and then an iterative process ensues. During each iteration a node in D is selected uniformly at random and copied to the new candidate solution. In each iteration, after the selection of a dominating node, its adjacent nodes are made dominated by using the dominated node selection procedure discussed in Section 3.1.2. And the selected nodes (both dominating and dominated nodes) are removed from the current graph G_c and D. After this, the next iteration begins. This process is repeated till the required number of nodes gets copied to the new candidate solution, and, then the new candidate solution (which is actually only a partial solution) along with current graph G_c is returned by this procedure. Please note that dominated nodes returned for a dominating nodes can be different from the current solution as the order of selection of dominating nodes is different from their order of selection in the current solution.

3.3. Two lower bounds

In this section, we compute two lower bounds on the minimum capacitated dominating set. The first lower bound which we call LB_1 can be computed as follows:

$$x := \max_{v \in V}(\min(C(v), d_G(v))) + 1 \tag{1}$$

$$LB_1 := \left\lceil \frac{|V|}{x} \right\rceil \tag{2}$$

With regard to (1), please note that a dominating node cover itself in addition to other nodes. A tighter lower bound which we call LB_2 can be obtained by sorting all the $|V|$ values in non-increasing order of $min(C(v), d_G(v)) + 1 \ \ \forall v \in V$ into a list and finding first LB_2 numbers in this list whose sum is as large as $|V|$. We have utilized LB_2 in terminating our multi-start iterated local search as explained in the next section.

It is to be noted that LB_1 purely depends on maximum effective capacity in the graph. Hence, this lower bound will not be tight if only a few nodes in the graph have effective capacity equal to maximum value. For example, the value of LB_1 for a 50 node graph having only one node with maximum effective capacity 4 is 10. But in this case, we cannot get a CAPMDS with 10 nodes, because only the maximum effective capacity node can cover 5 nodes including itself, and to cover remaining 45 nodes we need more than 9 nodes (in fact, at least 12). However, when a graph have all nodes with same effective capacity, this bound is tight. LB_2 is tighter than LB_1 as it is computed using information about effective capacities of more nodes. However, this bound also cannot be tight when nodes involved in computing this bound cover the same set of nodes including covering one another.

3.4. Other features

Our multi-start iterated local search approach makes a maximum of N_0 fresh starts. If a solution with cardinality LB_2 is found then it terminates immediately without making any further starts as this solution is optimal. During each start, iterated local search terminates either when best solution of that start has not improved over N_1 iterations or a solution with cardinality LB_2 has been found. The best solution found over all the starts is returned as the solution found by our approach. Hereafter, our multi-start iterated local search approach will be referred to as MS-ILS. It is to be noted that N_0 and N_1 are parameters to be determined empirically. So MS-ILS have four parameters in total. In addition to these two parameters, we have p_r (Section 3.1.3) and A (Section 3.2).

Algorithm 3 provides the pseudo-code of MS-ILS, where Perturb(S_i, G) is a procedure that takes as input a solution S_i and the input graph G, and implements the perturbation procedure described in Section 3.2. H-MECU(S, G, G_c) implements Algorithm 2.

Algorithm 3: Pseudo code for MS-ILS

Input: An undirected Graph $G := (V, E)$ with $V := \{v_1, v_2, ..v_n\}$ and $C(v_i) \geq 1, \forall v_i \in V$.
Output: Best solution found
Compute LB_2;
$i := 0$;
$best := V$;
while $((i < N_0) \wedge (|best| > LB_2))$ **do**
  $S_i :=$ H-MECU($\emptyset$, G, G);
  $not\_improved := 0$;
  while $((not\_improved < N_1) \wedge (|best| > LB_2))$ **do**
    $(S, G_c) :=$ Perturb(S_i, G);
    $S :=$ H-MECU(S, G, G_c);
    if $(|S| < |S_i|)$ **then**
      $S_i := S$;
      if $(|S_i| < |best|)$ **then**
        $best := S_i$;
      $not\_improved := 0$;
    else
      $not\_improved := not\_improved + 1$;
  $i := i + 1$;
return $best$;

4. An improved integer linear programming model for the CAPMDS problem

This section is devoted to our exact approach based on integer linear programming (ILP). Our ILP model is derived from the ILP model of Pinacho-Davidson et al. [16] by introducing several improvements. The ILP model of [16] itself is a slightly improved version of ILP model introduced in [14]. As our exact approach uses the concept of strong vertices, so we will introduce this concept first, and then we will describe our ILP approach.

4.1. Strong vertices in a graph for the CAPMDS problem

Let v_i and v_j be two (adjacent) vertices in the graph $G = (V, E)$. We say that vertex v_i is stronger than v_j if the following two conditions hold:

1. $\min(C(v_i), d_G(v_i)) > \min(C(v_j), d_G(v_j))$, *i.e.*, effective capacity of v_i in G is greater than that of v_j.
2. $N[v_j] \subseteq N[v_i]$ where $N[v_j] = \{v_k : (v_j, v_k) \in E\} \cup \{v_j\}$ and $N[v_i] = \{v_k : (v_i, v_k) \in E\} \cup \{v_i\}$. As per standard graph theory terminology, $N[v_i]$ and $N[v_j]$ are closed neighbourhoods of vertices v_i and v_j respectively. Hence, this condition can be restated as the closed neighbourhood of v_j should be a subset of closed neighbourhood of v_i.

Less formally, vertex v_i is stronger than v_j if any capacitated dominating set (CDS) containing v_j and not v_i can be turned into another CDS by removing v_j and by introducing v_i (this does not change the cardinality of the CDS). If we add the constraint vertex v_j cannot be selected in any CDS if vertex v_i is not already selected whenever v_i is stronger than v_j, then we can curtail the size of the solution space to be searched by the exact solver. Depending on the topology of the underlying graph, this can significantly reduce the size of the solution space. Please note that in the first condition above substituting $\geq$ in place of $>$ will not work because v_i not only has to cover all the vertices covered by v_j, but v_j itself also.

4.2. ILP model improvements

First, we will describe the ILP model of Pinacho-Davidson et al. [16]. This model uses the concept of open neighbourhood which is the set of vertices adjacent to a vertex. The open neighbourhood of a vertex v_i is denoted by $N(v_i)$, so $N(v_i) = N[v_i] \setminus \{v_i\}$. In this model, binary variables $x_i \forall v_i \in V$ are introduced to indicate whether the vertex v_i belongs to the dominating set ($x_i = 1$) or not ($x_i = 0$). A pair of binary variables y_{ij} and y_{ji} are introduced for each edge $(i, j) \in E$. $y_{ij} = 1$ indicates that node v_i dominates node v_j and $y_{ij} = 0$ indicates that node v_i does not dominate node v_j. With the help of these binary variables, the ILP formulation of Pinacho-Davidson et al. [16] for the CAPMDS problem is given below:

$$\textbf{Minimize} \sum_{v_i \in V} x_i \tag{3a}$$

$$\textbf{Subject to} \sum_{v_j \in N(v_i)} y_{ji} \geq 1 - x_i \qquad \forall v_i \in V \tag{3b}$$

$$\sum_{v_j \in N(v_i)} y_{ij} \leq C(v_i) \qquad \forall v_i \in V \tag{3c}$$

$$y_{ij} \leq x_i \qquad \forall v_i \in V,\ v_j \in N(v_i) \tag{3d}$$

$$x_i \in \{0, 1\} \qquad \forall v_i \in V \tag{3e}$$

$$y_{ij}, y_{ji} \in \{0, 1\} \qquad \forall (i, j) \in E \tag{3f}$$

(3a) is the objective function which needs to be minimized. Constraint (3b) enforces the constraint that a vertex is either a dominating vertex itself or it should be dominated by at least one dominating vertex. Constraint (3c) enforces the capacity restrictions on nodes. Constraint (3d) enforces the constraint that y_{ij} can be non-zero only when v_i is a dominating vertex. Constraints (3e) and (3f) enforce the binary nature of variables x_i and y_{ij}. This model will be referred to as ILP-PD. We have made four improvements in ILP-PD. These improvements are described below.

4.2.1. Improvement 1: Variable types

It can be seen that only the x variables have to be binary, the y variables can be set to continuous. Indeed, when all the x variables are integer, then all the y variables are binary too. Doing so is likely to decrease the computational effort of the solver, as the number of variables for branching is $|V|$ instead of $|V| + 2|E|$.

In addition, constraint (3b) states that if a vertex is a dominatee, then the number of entering arcs can be larger than one. However, it could be exactly one with no inconvenience, so this constraint can be transformed into an equality.

4.2.2. Improvement 2: Capacity constraint

Constraint (3c) enforces the capacity constraint on dominating nodes. However, since it is active for dominating nodes only, it can be improved to $\sum_{v_j \in N(v_i)} y_{ij} \leq \min(C(v_i), d_G(v_i))x_i$. Hence, this implies that a dominatee cannot dominate any node, and does not change the original constraint for dominators. Since $\min(C(v_i), d_G(v_i))x_i \leq \min(C(v_i), d_G(v_i))$, the new constraint is tighter than the original one.

4.2.3. Improvement 3: Utilizing strong vertices

As explained in Section 4.1, adding the constraint that vertex v_j cannot be chosen in any CDS if vertex v_i is not already chosen whenever vertex v_i is stronger than vertex v_j, can reduce the size of the solution space to be searched by the exact solver thereby boosting its performance. Hence, we add at most $|E|$ additional constraints of the form $x_j \leq x_i$ whenever vertex v_i is stronger than vertex v_j.

4.2.4. Improvement 4: Lower bound on the cardinality of a capacitated dominating set

For each vertex $v_i \in V$, let $w(v_i) = \min(C(v_i), d_G(v_i)) + 1 > 0$ be the maximum number of vertices that can be covered by vertex v_i. Since any CDS should cover all the vertices, the following constraint holds: $\sum_{v_i \in V} w(v_i)x_i \geq |V|$. Let W be the set of all the pairwise different values of $w(v_i)$ (the cardinality of W is at most $|V|$). The previous constraint can be attempted for strengthening as follows:

$$\sum_{v_i \in V} \left\lceil \frac{w(v_i)}{q} \right\rceil x_i \geq \left\lceil \frac{|V|}{q} \right\rceil \qquad \forall q \in W \tag{4}$$

Since it is not guaranteed that one of these constraint dominate the original one, they are all enforced. If some of them are redundant, they can be removed at presolve stage.

4.2.5. New ILP formulation

The new ILP model incorporating the improvements described above is given below:

$$\textbf{Minimize} \sum_{v_i \in V} x_i \tag{5a}$$

$$\textbf{Subject to} \sum_{v_j \in N(v_i)} y_{ji} = 1 - x_i \qquad \forall v_i \in V \tag{5b}$$

$$\sum_{v_j \in N(v_i)} y_{ij} \leq \min(C(v_i), d_G(v_i))x_i \qquad \forall v_i \in V \tag{5c}$$

$$y_{ij} \leq x_i \qquad \forall v_i \in V,\ v_j \in N(v_i) \tag{5d}$$

$$x_j \leq x_i \qquad \forall v_i, v_j \in V \times V \mid v_i \text{ is stronger than } v_j \tag{5e}$$

$$\sum_{v_i \in V} \left\lceil \frac{w(v_i)}{q} \right\rceil x_i \geq \left\lceil \frac{|V|}{q} \right\rceil \qquad \forall q \in W \tag{5f}$$

$$x_i \in \{0, 1\} \qquad \forall v_i \in V \tag{5g}$$

$$y_{ij}, y_{ji} \in [0, 1] \qquad \forall (i, j) \in E \tag{5h}$$

(5a) represents the objective function of the CAPMDS problem that has to be minimized. Constraint (5b) enforces the constraint that each vertex is either part of the dominating set or covered by a vertex belonging to the dominating set (Improvement 1). Constraint (5c) enforces the constraint on the maximum number of vertices that a vertex can dominate. Constraint (5d) enforces the constraint that y_{ij} can be non-zero only when v_i is a dominating vertex. Constraint (5e) enforces the constraint due to strong vertices. Constraint (5f) enforces the fourth improvement.

Constraint (5g) restricts the value of each $x_i \forall v_i \in V$ to 0 or 1. Constraint (5h) confines each y_{ij} and y_{ji} $\forall (i,j) \in E$ to interval [0, 1]. We will refer to this new model as ILP only. Our exact approach consists of solving this ILP model through an exact solver such as CPLEX.

5. A matheuristic for the CAPMDS problem

We have developed a matheuristic combining MS-ILS with the ILP problem formulation described in previous section. If MS-ILS is able to find an optimal solution (a solution with value equal to LB_2 as mentioned in Section 3.4) then the ILP is not solved and this solution is returned as solution found by the matheuristic. Otherwise, the ILP is solved, but it is executed on a subset $V' \subseteq V$ of vertices. This subset consists of those vertices which occur as dominating vertices either with certain minimum frequency in different CDS computed by MS-ILS or in best solution computed by MS-ILS or their respective stronger vertices. This way of utilizing the information about dominating vertices is called solution merging in [17], and it is based on the assumption that if a vertex is a dominating vertex in one good solution, then chances of it being a dominating vertex in several good solutions are high. Therefore, limiting the search for dominating vertices to this set can aid the exact approach in finding an even better solutions quickly.

To compute V', we count the frequency of occurrences of each vertex as a dominating vertex in N_0 solutions computed by MS-ILS. Now, V' is initialized with those vertices of V whose frequency of occurrence is at least K_f, where K_f is a parameter to be determined empirically. Moreover, vertices belonging to the best solution of MS-ILS is added to V' if they are not already there. Now, whenever a vertex $v_j \in V'$ is such that a vertex v_i is stronger than v_j and $v_i \notin V'$, then $v_j \in V'$ is replaced with v_i. This process is repeated until no such vertex v_j van be found in V'. This is done to increase the possibility of finding a better solution as replacing a vertex in V' with a stronger vertex can reduce the size of final dominating set obtained. If even after this process, a vertex v_i that is stronger than some other vertex and no other vertex is stronger than v_i is left out of V' then v_i is added to V'. This process is also repeated until no such vertex is left out of V'. This is done so as not to left out any promising vertex from V'. By solving the ILP model on a reduced set of vertices, we loose the proof of optimality. However, when executed for a fixed amount of time, the solution obtained with this approach can be better than executing the exact approach on V for the same amount of time, specially on large instances. For small instances where ILP can be solved in reasonable amount of time, K_f can be set to 0 to retain the proof of optimality. Hereafter, this matheuristic approach will be referred to as MH.

6. Computational results

In this section, we will present the computational results and their analysis. First, we discuss about the benchmark instances used to evaluate the performance of various approaches. Next, we present the results of MS-ILS along with other state-of-the-art heuristic approaches available in the literature. At last, we present the results of our ILP approach and metaheuristics along with their comparison with state-of-the-art approaches. All our algorithms are implemented in C and executed on an Intel core i7-7700 CPU based system with 32 GB of RAM running at 3.60 GHz under Ubuntu 18.04. IBM CPLEX 12.7 library is used to solve the ILP models and their relaxed versions, and is restrained to use a single core.

6.1. Benchmark instances

The performance of our approach has been evaluated on same benchmark instances as used in the literature [13–16]. These instances consist of two kinds of graphs. One kind of graphs are Unit Disk Graphs (UDG) which are created using the topology generator proposed in [28], the other kind of graphs are general graphs which have been taken from the Type-I instances in [29]. Each kind of these graphs have six different sizes, viz. 50, 100, 250, 500, 800 and 1000 nodes. In case of UDG, nodes are randomly distributed in an area of 1000 $\times$ 1000 units. Two different communication ranges 150 and 200 units are used to study the effect of degree of connectivity on the solution. In case of general graphs, the number of edges varies from 100 to 10 000 based on the number of nodes in the graphs. With regard to capacity, there are two types of benchmark instances, viz. uniform capacity instances and variable capacity instances [13]. In case of uniform capacity instances, capacity of every node in the graph is same. Three values of uniform capacities are taken, viz. 2, 5 and average degree α. On the other hand, in case of variable capacity graphs, three scenarios are considered. In the first scenario, capacity of each node in the graph is set randomly to either 2 or 5, in the second scenario, capacity of each node is set randomly to either $\frac{\alpha}{5}$ or $\frac{\alpha}{2}$. These two scenarios are real world scenarios based on the characteristics of wireless networks. In the third scenario, capacity of each node is randomly chosen from the set $\{1, 2, \ldots, \alpha\}$. For each combination of nodes, range and capacity in UDGs, and nodes, edges and capacity in general graphs, there are 10 different instances. Reported results are the average results over each group of these 10 instances after executing our approach once on each instance. This is inline with the practice followed in the literature [13–16].

6.2. MS-ILS results

In this section, we will discuss the values of different parameters used in MS-ILS, comparative performance of MS-ILS vis-à-vis state-of-the-art approaches. As MS-ILS is a heuristic approach, we have compared it with 4 state-of-the-art heuristic approaches, viz. MC-LDEG [13], GA and ACO [15], LS_PD [14]. Results of these 4 approaches are taken from their respective papers. In particular, comparison of MS-ILS with LS_PD is most important as latter is the best heuristic approach available in the literature.

6.2.1. Parameters

As discussed in Section 3.4, MS-ILS makes use of four parameters, viz. N_0, N_1, p_r and A. In all our experiments with MS-ILS, the number of starts N_0 is set to 75 and the maximum number of iterations N_1 without improvement in solution quality is set to 150 for each start. The random strategy is used with probability p_r = 0.4 while removing redundant nodes. The parameter A is set to 5, *i.e.*, only 20% of the dominating nodes are retained after perturbation. The values for all these parameters are set empirically. As MS-ILS was beating the heuristic approaches available in the literature easily, we tried to reduce its execution time without significantly reducing the quality of solutions returned by it. This was important for our matheuristic also which needs to be run for a fixed amount of time like the previous matheuristic/exact approaches available in the literature.

6.2.2. Results on uniform capacity instances

Performance on uniform capacity instances are reported separately for Unit Disk Graphs (UDGs) and General Graphs.

Performance on UDGs: Table 2 provides the computational results of different approaches on uniform capacity UDG instances. First two columns represent the number of nodes ($|V|$) and the

Table 2
Cardinality of CAPMDS computed using MC-LDEG, ACO, HGA, LS_PD and MS-ILS for UDG graph instances with an uniform capacity of 2, 5, and average degree (α) for every node.

$\|V\|$	Range	MC-LDEG			ACO			HGA			LS_PD			MS-ILS		
		2	5	α	2	5	α	2	5	α	2	5	α	2	5	α
50	150	21.1	15.6	17.9	17.9	13.5	15.2	18.1	13.0	14.7	**17.2**	**12.9**	**14.4**	**17.2**	13.0	**14.4**
50	200	20.4	12.7	12.7	17.6	10.8	10.8	17.7	10.5	10.5	**17.0**	**10.0**	**10.0**	**17.0**	10.2	10.2
100	150	41.2	23.3	23.3	35.9	20.8	20.8	36.4	20.6	20.6	34.5	**18.8**	18.9	**34.0**	**18.8**	**18.8**
100	200	41.4	21.0	14.1	35.5	18.7	12.5	35.8	18.7	11.8	34.1	**17.4**	**11.0**	**34.0**	**17.4**	**11.0**
250	150	104.0	48.4	25.1	92.6	46.5	22.6	92.5	46.4	21.8	86.0	43.7	**19.4**	**84.0**	**42.0**	19.6
250	200	106.4	47.8	15.3	92.1	45.9	13.7	91.9	44.9	13.0	85.4	43.0	**11.5**	**84.0**	**42.0**	11.9
500	150	212.7	95.7	26.3	187.7	93.3	24.5	187.4	91.2	22.7	171.6	86.2	**20.7**	**167.0**	**84.0**	21.2
500	200	213.5	95.4	15.7	187.0	92.1	14.2	186.5	90.4	13.8	170.3	85.0	**12.1**	**167.0**	**84.0**	12.8
800	150	343.4	152.7	27.4	303.5	149.7	26.1	301.1	145.4	24.0	273.9	137.0	**21.3**	**267.0**	**134.0**	22.3
800	200	344.3	152.0	16.3	302.6	146.8	14.7	301.1	144.0	14.5	272.4	135.7	**12.6**	**267.0**	**134.0**	13.1
1000	150	428.4	190.8	27.2	380.8	186.7	25.9	377.6	181.5	24.5	342.6	171.0	**21.5**	**334.0**	**167.0**	23.1
1000	200	430.5	190.6	17.0	379.5	183.7	15.2	377.3	179.8	14.5	340.4	169.6	**12.9**	**334.0**	**167.0**	13.2
Avg:		192.3	87.2	19.9	169.4	84.0	18.0	168.6	82.2	17.2	153.8	77.5	**15.5**	**150.5**	**76.1**	16.0

Table 3
Cardinality of CAPMDS computed using MC-LDEG, ACO, HGA, LS_PD and MS-ILS for general graph instances with an uniform capacity of 2, 5, and average degree (α) for every node.

$\|V\|$	$\|E\|$	MC-LDEG			ACO			HGA			LS_PD			MS-ILS		
		2	5	α	2	5	α	2	5	α	2	5	α	2	5	α
50	100	20.3	14.2	15.2	17.3	13.0	13.4	17.5	12.1	12.7	**17.0**	**11.9**	**12.0**	**17.0**	12.0	**12.0**
50	250	19.9	10.1	7.0	**17.0**	9.1	6.9	17.2	9.1	6.5	**17.0**	**9.0**	**6.0**	**17.0**	**9.0**	6.1
50	500	19.3	9.5	4.2	**17.0**	**9.0**	4.1	**17.0**	**9.0**	**3.8**	**17.0**	**9.0**	**3.8**	**17.0**	**9.0**	3.9
100	100	43.6	39.5	43.6	36.9	35.7	36.9	35.5	35.0	35.5	**34.0**	**33.6**	**34.0**	**34.0**	**33.6**	**34.0**
100	250	39.3	26.1	26.1	36.2	22.8	22.8	36.0	22.4	22.4	**34.0**	20.3	**20.2**	**34.0**	**20.2**	**20.2**
100	500	39.6	21.1	15.3	35.6	19.2	14.9	35.3	19.1	14.1	**34.0**	**17.0**	**12.2**	**34.0**	**17.0**	12.5
250	250	108.8	100.0	108.8	92.6	93.1	92.6	91.0	89.5	91.0	**84.0**	83.5	**84.0**	**84.0**	**83.3**	**84.0**
250	500	99.4	71.1	77.7	95.2	66.7	68.7	93.5	64.4	68.5	**84.0**	59.9	61.7	**84.0**	**59.1**	**60.3**
250	1000	98.3	55.8	45.4	93.2	54.3	43.7	92.2	52.7	42.4	**84.0**	45.0	**37.9**	**84.0**	**44.3**	38.5
500	500	216.5	201.2	216.5	190.2	190.0	190.2	185.9	181.5	185.9	168.6	168.3	168.4	**167.0**	**167.0**	**167.0**
500	1000	199.2	141.2	153.4	194.7	136.8	143.5	191.2	131.9	140.2	170.2	121.5	126.4	**167.0**	**119.8**	**122.4**
500	2000	197.0	109.4	91.2	189.3	107.7	89.7	186.9	107.3	88.6	167.5	92.2	**78.6**	**167.0**	**90.9**	79.8
800	1000	330.8	281.3	330.8	312.8	279.8	312.8	303.3	264.9	303.3	274.2	253.2	274.0	**267.0**	**244.1**	**267.0**
800	2000	317.1	209.4	209.4	308.7	207.4	207.4	307.0	200.8	200.8	272.7	176.2	178.1	**267.0**	**174.2**	**174.2**
800	5000	313.0	156.8	104.3	301.9	152.5	101.6	298.0	154.2	101.9	267.8	140.4	**92.2**	**267.0**	**139.0**	97.9
1000	1000	431.7	399.5	431.7	385.9	385.0	385.9	377.5	370.3	377.5	338.4	338.7	338.4	**334.0**	**333.7**	**334.0**
1000	5000	391.6	205.7	152.6	380.0	200.8	149.6	375.6	202.3	150.8	336.0	181.0	**137.4**	**334.0**	**177.2**	142.2
1000	10 000	391.3	187.7	88.7	378.3	183.4	85.6	372.1	184.1	86.4	**334.0**	171.0	**81.3**	**334.0**	**169.7**	89.5
Avg:		182.0	124.4	117.9	171.3	120.4	109.5	168.5	117.3	107.4	151.9	107.3	**97.0**	**150.5**	**105.7**	**97.0**

Table 4
Cardinality of CAPMDS computed using MC-LDEG, ACO, HGA, LS_PD and MS-ILS for UDG graph instances where capacity of each node vary randomly over $\{2, 5\}$, $\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$ and $\{1, \dots, \alpha\}$.

$\|V\|$	Range	MC-LDEG			ACO			HGA			LS_PD			MS-ILS		
		$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\dots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\dots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\dots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\dots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\dots,\alpha\}$
50	150	16.1	31.0	19.1	15.0	25.6	17.1	14.7	25.6	17.2	**14.5**	**25.4**	**16.7**	**14.5**	**25.4**	16.8
50	200	12.7	21.1	13.7	12.0	17.9	12.7	11.4	18.1	12.5	**11.1**	**17.3**	**12.1**	**11.1**	**17.3**	12.3
100	150	25.8	37.4	25.6	23.7	33.1	24.1	23.4	33.2	23.3	**21.8**	30.3	**21.4**	21.9	**29.9**	21.8
100	200	21.3	21.0	15.2	19.2	19.5	14.3	19.2	19.4	13.8	17.8	**17.8**	**13.0**	**17.7**	**17.8**	13.2
250	150	49.7	37.6	26.8	46.3	35.0	24.7	47.0	35.7	24.3	43.0	32.4	**21.9**	**42.1**	**31.9**	22.6
250	200	47.6	22.6	16.9	45.2	20.7	15.5	45.1	21.4	15.0	**42.0**	19.2	**13.0**	**42.0**	**18.9**	14.1
500	150	95.8	39.2	30.0	92.1	35.7	26.6	92.2	37.0	26.8	84.9	32.9	**22.6**	**84.0**	**32.0**	24.2
500	200	95.2	24.2	17.6	91.9	21.4	16.0	90.3	22.3	15.8	**84.0**	**19.4**	**13.0**	**84.0**	**19.4**	14.5
800	150	151.7	40.4	29.7	148.0	36.7	26.9	146.1	38.0	27.7	134.9	33.3	**22.7**	**134.0**	**31.8**	24.6
800	200	151.7	24.3	18.0	148.2	21.9	16.5	144.7	23.1	16.6	134.1	19.8	**13.2**	**134.0**	**19.6**	14.7
1000	150	189.2	41.0	29.8	185.5	37.4	27.3	183.0	38.6	28.4	168.6	33.5	**22.6**	**167.0**	**32.5**	25.5
1000	200	189.0	25.1	18.4	185.7	22.1	16.7	180.9	23.1	16.5	167.5	20.0	**13.0**	**167.0**	**19.2**	14.9
Avg:		87.1	30.4	21.7	84.4	27.2	19.9	83.2	28.0	19.8	77.0	25.1	**17.1**	**76.6**	**24.6**	18.3

transmission range (*Range*), The last row Avg represents the overall average values obtained by each approach for different capacities. The best values obtained by all the approaches are represented in bold for ease of identification. This table clearly shows that MS-ILS obtained better results in comparison to MC-LDEG, ACO, HGA. As far as comparison with LS_PD is concerned, MS-ILS obtained better or equal results on 25 instance groups out of 36. We can observe that, in most of the cases MS-ILS has obtained optimal results whenever the graphs have capacity 2 or 5 by comparing these values with respective lower bounds (LB_2). By observing these instances, it can clearly be seen that the average degree of the graph is more than the value of uniform capacity 2 or 5 as the case may be. On the other hand, for graphs with capacity same as the average degree, MS-ILS has slightly poor performance. Actually, UDG graphs that we considered are highly dense due to the large values of transmission ranges. So when capacity is equal to average degree, dominating set has smaller size and there will be fewer iterations of H-MECU. As a result, each decision have wider impact and even one error of judgement

Table 5
Cardinality of CAPMDS computed using MC-LDEG, ACO, HGA, LS_PD and MS-ILS for general graph instances with capacity of each node vary randomly over $\{2, 5\}$, $\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$ and $\{1, \ldots, \alpha\}$.

$\|V\|$	$\|E\|$	MC-LDEG			ACO			HGA			LS_PD			MS-ILS		
		$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$
50	100	15.1	21.7	16.3	14.3	18.3	15.0	13.2	18.9	14.5	**13.0**	**17.7**	**13.7**	**13.0**	**17.7**	**13.7**
50	250	10.3	10.5	8.6	9.8	9.6	8.4	9.2	9.6	7.8	**9.0**	**9.0**	**7.2**	**9.0**	**9.0**	7.6
50	500	9.8	6.0	4.5	**9.0**	5.2	4.3	**9.0**	5.1	**3.9**	**9.0**	**5.0**	**3.9**	**9.0**	**5.0**	4.1
100	100	41.6	63.9	47.6	36.0	51.6	41.1	35.4	51.7	40.7	**33.7**	**50.0**	**40.1**	**33.7**	**50.0**	**40.1**
100	250	26.8	42.6	28.5	26.2	38.2	27.1	24.7	38.4	26.0	22.3	34.9	**23.7**	**22.2**	**34.3**	24.0
100	500	21.4	21.4	16.2	19.9	19.9	15.6	20.0	19.6	15.5	17.2	**17.3**	**14.1**	**17.1**	**17.3**	14.5
250	250	103.3	156.4	113.4	93.1	132.7	104.8	90.7	132.9	102.6	**83.7**	**125.0**	99.0	**83.7**	**125.0**	**98.9**
250	500	75.4	107.6	82.2	74.6	102.2	81.5	72.1	102.0	77.7	66.5	91.4	71.5	**65.0**	**88.9**	**70.3**
250	1000	57.6	65.6	49.5	55.7	61.7	47.0	56.1	63.1	47.4	**48.4**	**54.9**	44.6	48.5	**54.9**	**44.2**
500	500	209.4	314.0	231.2	190.1	269.7	216.6	185.3	270.0	212.3	168.2	251.1	203.2	**167.0**	**250.0**	**202.7**
500	1000	153.1	214.0	166.2	151.9	205.8	163.0	146.8	206.1	162.0	135.2	183.5	147.7	**131.7**	**178.0**	**146.0**
500	2000	115.0	131.1	98.6	111.0	124.9	94.3	112.4	127.2	96.2	**98.9**	**110.8**	92.6	99.3	111.5	**92.1**
800	1000	298.7	496.7	356.6	295.7	437.4	348.0	279.4	437.4	339.2	261.7	401.6	310.5	**252.4**	**400.0**	**304.6**
800	2000	222.4	339.6	228.7	219.9	328.8	223.0	219.0	329.6	225.3	199.1	292.2	208.2	**193.6**	**283.1**	**205.4**
800	5000	162.8	146.6	109.3	157.1	140.6	105.2	159.0	142.7	107.1	**144.5**	**127.0**	**104.8**	145.8	131.3	110.5
1000	1000	413.0	628.7	457.5	385.0	545.8	437.5	374.0	545.5	428.3	338.6	505.6	405.0	**333.8**	**500.0**	**400.8**
1000	5000	213.1	213.2	164.7	206.4	206.6	157.7	209.5	208.3	160.3	**189.4**	**188.0**	**156.4**	191.8	189.8	162.0
1000	10 000	191.4	118.4	92.9	186.2	113.7	**88.4**	188.0	115.5	90.3	172.2	**104.7**	92.4	**172.1**	113.6	97.8
Avg:		130.0	172.1	126.2	124.5	156.3	121.0	122.4	156.9	119.8	111.7	142.8	**113.3**	**110.5**	**142.2**	**113.3**

Table 6
Execution times (in seconds) of ACO, HGA, LS_PD and MS-ILS for UDG graph instances with an uniform capacity of 2, 5, and average degree (α) for every node.

$\|V\|$	Range	ACO			HGA			LS_PD			MS-ILS		
		2	5	α	2	5	α	2	5	α	2	5	α
50	150	2.7	2.8	2.3	1.0	0.9	1.0	0.1	**0.0**	**0.0**	**0.0**	0.1	0.1
50	200	2.4	2.8	2.8	1.0	0.8	0.8	**0.0**	**0.1**	**0.0**	**0.0**	**0.1**	0.1
100	150	8.8	8.1	7.7	3.5	2.6	2.4	**0.0**	2.7	2.6	**0.0**	**0.5**	**0.5**
100	200	7.1	6.9	9.1	3.3	3.1	2.3	**0.0**	**0.0**	**0.0**	**0.0**	0.3	0.5
250	150	38.0	28.2	34.8	18.9	24.1	8.0	**0.0**	0.4	8.3	**0.0**	**0.0**	**2.3**
250	200	31.2	23.0	27.9	18.2	19.8	7.2	**0.0**	**0.0**	**1.7**	**0.0**	**0.0**	2.5
500	150	129.3	84.8	99.9	83.1	100.0	20.3	0.1	0.1	79.4	**0.0**	**0.0**	**7.2**
500	200	130.0	74.1	82.9	91.0	69.3	16.0	**0.0**	**0.0**	22.5	**0.0**	**0.0**	**8.1**
800	150	360.6	207.1	264.9	241.4	238.6	44.1	0.2	0.2	250.1	**0.0**	**0.0**	**15.8**
800	200	394.2	218.2	198.3	244.1	152.7	35.2	0.1	**0.0**	129.5	**0.0**	**0.0**	**19.6**
1000	150	582.2	337.2	340.3	420.8	346.0	65.5	0.2	1.0	319.7	**0.0**	**0.0**	**23.1**
1000	200	631.1	365.4	289.4	414.4	253.4	41.0	0.1	0.1	**21.4**	**0.0**	**0.0**	30.1
Avg:		193.1	113.2	113.4	128.4	100.9	20.3	0.1	0.4	69.6	**0.0**	**0.1**	**9.2**

can drastically deteriorate the final solution obtained. For example, if a node has degree equal to average degree of the graph, which lets us assume is 10, then this node will have effective capacity of 10, and, by selecting this node as a dominating node, it will end-up using its maximum capacity, but the problem is subsequent deletion of this node along with all its dominated node. By deleting these 11 nodes, there is a chance of deleting more than 100 edges from G_c, thereby creating the possibility of isolated nodes in G_c. This is the possible reason for slightly poor performance of MS-ILS in comparison to LS_PD on uniform capacity UDG instances where capacity is equal to average degree.

Performance on General Graphs: Table 3 reports the computational results of different approaches on general graph instances with uniform capacity. First two columns represents the number of nodes ($|V|$) and the number of edges ($|E|$). We can observe that, in almost all the cases MS-ILS is able to obtain better results when compared with MC-LDEG, ACO and HGA. Compared with LS_PD, MS-ILS has obtained better or equal results on 45 instance groups out of 54 groups. Furthermore, all the graphs with uniform capacity 2 are solved to optimality as the solutions obtained match the respective lower bounds (LB_2). In comparison to UDG graphs with capacity equal to average degree, MS-ILS performed better on general graphs with capacity equal to average degree. This is due to the fact that average degree of general graphs are comparatively less.

From these two tables, we can see that compared with LS_PD, our approach MS-ILS has obtained 70 better or equal results out of 90 uniform capacity instance groups.

6.2.3. Results on variable capacity instances

In this subsection, we will discuss the computational results of various approaches on variable capacity UDGs as well as general graphs.

Performance on UDGs: Table 4 presents the computational results of different approaches on UDG instances with variable capacity. For the UDG instances with variable capacity, MS-ILS has outperformed MC-LDEG, ACO, HGA for all cases. In comparison to LS_PD, MS-ILS has obtained better or equal results on 23 instance groups out of 36.

Performance on General Graphs: Table 5 reports the computational results of different approaches on general graph instances with variable capacity. Here also similar conclusions to previous tables can be drawn with regard to comparative performance of different approaches. MS-ILS has obtained better or equal results on 39 instance groups out of 54 in comparison to LS_PD.

So out of 90 variable capacity instance groups, MS-ILS has obtained better or equal results in 62 groups in comparison to LS_PD.

6.2.4. Comparison of execution times

Tables 6 to 9 report the execution times of various approaches, viz. HGA, ACO, LS_PD and MS-ILS. MC-LDEG being a constructive heuristic has negligible execution time in comparison to other approaches, and hence, its execution time is not reported. MS-ILS is executed on an Intel Core i7-7700 CPU based system with 32 GB of RAM running at 3.60 GHz under Ubuntu 18.04. Whereas

Table 7
Execution times (in seconds) of ACO, HGA, LS_PD and MS-ILS for general graph instances with an uniform capacity of 2, 5, and average degree (α) for every node.

$\|V\|$	$\|E\|$	ACO			HGA			LS_PD			MS-ILS		
		2	5	α	2	5	α	2	5	α	2	5	α
50	100	2.5	2.6	2.1	1.0	12.1	1.0	**0.0**	**0.0**	**0.0**	**0.0**	0.2	0.2
50	250	2.1	2.2	2.0	1.0	9.1	0.9	**0.0**	**0.0**	**0.1**	**0.0**	**0.0**	0.3
50	500	2.2	1.8	2.0	1.1	9.0	0.8	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.2
100	100	6.1	10.4	6.2	3.9	35.0	3.7	**0.0**	**0.2**	**0.0**	**0.0**	**0.2**	**0.0**
100	250	9.0	9.3	9.1	3.9	22.4	2.7	**0.0**	5.9	5.9	**0.0**	**0.7**	**0.7**
100	500	7.2	6.6	6.0	3.5	19.1	2.7	**0.0**	0.1	2.6	**0.0**	**0.0**	**1.0**
250	250	46.2	84.6	46.4	37.2	89.5	36.2	3.6	11.6	2.1	**0.0**	**0.4**	**0.0**
250	500	82.7	90.4	76.7	42.4	64.4	25.5	5.7	46.0	17.8	**0.0**	**2.3**	**2.3**
250	1000	52.1	57.6	55.2	34.1	52.7	26.4	10.0	32.6	41.1	**0.0**	**4.1**	**3.8**
500	500	281.0	537.8	273.7	253.8	181.5	250.8	**0.0**	10.7	32.3	**0.0**	**3.6**	**0.0**
500	1000	567.0	581.0	536.2	308.6	131.9	181.1	55.5	80.5	59.0	**0.0**	**7.0**	**6.8**
500	2000	300.0	338.8	335.4	234.7	107.3	206.2	38.4	71.3	101.0	**0.0**	**12.1**	**10.9**
800	1000	2579.5	3282.1	2594.8	979.9	264.9	975.8	160.9	101.8	120.8	**0.2**	**16.5**	**0.2**
800	2000	2099.9	2240.6	2262.4	1252.9	200.8	952.3	127.1	223.6	159.2	**0.0**	**15.4**	**15.5**
800	5000	633.1	633.1	683.2	773.4	154.2	607.6	44.4	131.6	329.4	**0.0**	**40.6**	**31.0**
1000	1000	2251.9	3729.3	2231.3	1733.9	370.3	1714.9	136.5	75.7	134.7	**0.0**	**10.3**	**0.0**
1000	5000	1703.5	1916.3	2000.7	1728.1	202.3	1528.8	126.0	140.3	352.2	**0.0**	**45.7**	**35.9**
1000	10 000	734.5	614.5	726.6	1104.5	184.1	823.1	4.2	**62.7**	604.8	**0.0**	103.0	**55.4**
Avg:		631.1	785.5	658.3	472.1	117.3	407.8	39.6	55.3	109.1	**0.0**	**14.6**	**9.1**

Table 8
Execution times (in seconds) of ACO, HGA, LS_PD and MS-ILS for UDG graph instances where capacity of each node vary randomly over $\{2, 5\}$, $\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$ and $\{1, \ldots, \alpha\}$.

$\|V\|$	Range	ACO			HGA			LS_PD			MS-ILS		
		$\{2, 5\}$	$\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$	$\{1, \ldots, \alpha\}$
50	150	3.5	2.9	3.3	1.1	1.1	1.1	**0.0**	**0.0**	**0.1**	0.2	0.1	0.2
50	200	2.8	2.9	3.2	1.0	1.1	1.0	**0.0**	**0.1**	0.5	0.2	0.2	**0.2**
100	150	10.2	9.4	12.4	4.5	4.5	3.7	3.5	6.7	4.3	**0.6**	**0.6**	**0.6**
100	200	8.4	9.0	9.7	3.9	3.6	3.4	3.7	3.4	0.8	**0.6**	**0.6**	**0.5**
250	150	37.6	38.0	44.4	30.8	24.1	18.6	18.4	25.2	6.1	**0.5**	**1.0**	**2.4**
250	200	30.5	28.7	36.1	29.7	17.6	13.8	36.0	13.3	5.5	**0.0**	**0.3**	**2.6**
500	150	132.1	116.0	131.8	152.0	98.4	73.6	35.7	35.9	55.7	**0.0**	**0.1**	**7.4**
500	200	117.0	89.1	107.7	132.5	75.8	47.3	33.4	116.3	53.0	**0.0**	**0.0**	**8.5**
800	150	365.3	271.9	356.1	497.9	269.8	204.3	251.8	91.8	163.0	**0.0**	**6.9**	**16.5**
800	200	349.2	218.9	273.8	435.6	202.8	72.1	595.3	100.1	76.5	**0.0**	**0.1**	**20.3**
1000	150	617.9	409.7	470.2	881.9	404.6	265.5	235.4	173.6	500.4	**0.0**	**1.0**	**24.6**
1000	200	601.4	319.6	351.1	795.6	321.6	146.2	408.1	88.2	319.3	**0.0**	**0.3**	**31.2**
Avg:		189.7	126.3	150.0	247.2	118.8	70.9	135.1	54.6	98.8	**0.2**	**0.9**	**9.6**

Table 9
Execution times (in seconds) of ACO, HGA, LS_PD and MS-ILS for general graph instances where capacity of each node vary randomly over $\{2, 5\}$, $\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$ and $\{1, \ldots, \alpha\}$.

$\|V\|$	$\|E\|$	ACO			HGA			LS_PD			MS-ILS		
		$\{2, 5\}$	$\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$	$\{1, \ldots, \alpha\}$
50	100	3.4	3.0	3.1	1.1	1.1	0.9	**0.2**	0.4	0.8	**0.2**	**0.2**	**0.2**
50	250	2.2	2.2	2.3	0.9	1.0	0.9	**0.0**	0.2	0.6	**0.0**	**0.0**	**0.3**
50	500	2.0	1.9	2.1	1.1	0.9	0.9	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.2
100	100	12.9	12.7	11.7	3.8	3.9	4.4	**0.1**	0.1	**0.0**	0.2	**0.0**	0.3
100	250	12.0	10.8	11.5	4.1	4.6	4.2	8.9	6.7	7.4	**0.7**	**0.3**	**0.8**
100	500	7.7	7.5	8.7	3.5	3.9	3.4	6.5	6.4	5.9	**0.5**	**0.6**	**1.0**
250	250	128.7	105.7	102.9	36.1	40.4	40.7	6.6	8.9	6.6	**0.6**	**0.0**	**1.4**
250	500	115.7	95.6	106.4	53.8	48.2	46.1	20.7	19.3	18.7	**2.3**	**3.0**	**2.5**
250	1000	74.5	64.6	74.0	41.6	40.8	35.4	20.6	20.2	27.3	**4.5**	**5.0**	**4.1**
500	500	895.3	658.1	735.8	243.3	288.8	296.3	60.4	61.2	28.3	**0.8**	**0.0**	**3.9**
500	1000	828.8	632.8	706.1	320.5	352.7	346.2	104.5	36.5	55.0	**6.2**	**8.4**	**7.0**
500	2000	367.9	363.9	438.4	289.8	280.3	261.0	63.2	59.8	99.8	**13.2**	**14.7**	**11.2**
800	1000	4037.9	2368.0	2961.3	1210.3	1101.4	1341.4	180.3	85.9	90.1	**10.9**	**0.3**	**9.4**
800	2000	2651.8	2256.0	3127.4	1404.0	1316.6	1392.6	174.9	107.6	124.8	**15.4**	**26.6**	**17.2**
800	5000	838.9	900.9	1063.6	839.8	810.0	726.6	161.5	101.8	227.2	**46.9**	**44.0**	**33.8**
1000	1000	7318.9	4941.1	6013.6	1683.0	2159.6	2277.3	139.9	108.9	146.0	**9.0**	**0.0**	**12.2**
1000	5000	2609.5	2692.5	3406.4	1891.8	1759.4	1740.8	175.7	118.4	335.5	**47.9**	**49.6**	**39.2**
1000	10 000	971.3	1091.9	1542.9	1249.1	1026.1	911.9	207.2	123.0	387.0	**121.6**	**82.8**	**60.3**
Avg:		1160.0	900.5	1128.8	515.4	513.3	523.9	74.0	48.1	86.7	**15.6**	**13.1**	**11.4**

the execution times of all other algorithms are taken from [14] where these approaches had been executed on a 2.13 GHz Intel Xeon E7-4830 CPU based system. Therefore, exact comparison of execution times are not possible. However, a rough comparison can always be made. Based on the information available in the public domain regarding the performance of these two processors on various benchmark instances, our system can be 1.5 to 2 times faster than the one used in [14]. Even after compensating for this difference in processing speeds, MS-ILS is several times faster than other approaches on most of the instances. Furthermore,

Table 10
Summary table for MS-ILS Vs. LS_PD.

Capacity	Graph	Better	Equal	Worst	Total
Uniform	UDG	19	6	11	36
	General	27	18	9	54
Variable	UDG	15	8	13	36
	General	25	14	15	54
Total		86	46	48	180

following observations can be made regarding the execution time of MS-ILS:

- The execution time of MS-ILS increases with the increase in the number of nodes in the graph.
- If the number of nodes are same, then the execution time increases with increase in the number of edges in the graph.
- In general, the execution time of MS-ILS is expected to increase as effective capacity decreases. This is due to increase in size of the dominating set with decrease in effective capacity and as a result H-MECU require higher number of iterations. However, this has not been observed in most cases due to pre-mature termination resulting from finding an optimal solution.
- On no instance MS-ILS requires more than 122 s.

6.2.5. MS-ILS Vs. LS_PD

As LS_PD is the existing best heuristic approach for the CAP-MDS problem, therefore, superiority of MS-ILS over LS_PD is extremely important. To highlight this superiority, we have provided a summary table where performance of MS-ILS is compared with LS_PD in terms of number of instances where MS-ILS performed better, same or worse than LS_PD on each category of instances and overall. Table 10 provides this summary. This table clearly establishes the superiority of MS-ILS over LS_PD. In addition, as discussed in previous Section, MS-ILS is several times faster than LS_PD.

6.2.6. Sensitivity analysis of MS-ILS

This section study the impact of different parameter values and components of MS-ILS on solution quality. First, we will study the impact of parameter values on solution quality. MS-ILS has four parameters. These four parameters along with their respective values are $N_0 = 75$, $N_1 = 150$, $p_r = 0.4$ and $A = 5$. To determine how sensitive MS-ILS is to parameter settings, we have varied all parameter values one-by-one while keeping all other parameters set to their respective aforementioned values. For each parameter, we have taken one value smaller and one value larger than the chosen value. This analysis is done on eight groups of instances. These instances have the capacity α for uniform capacity graphs and $\{1, \ldots, \alpha\}$ for variable capacity graphs. Table 11 presents the result of this study. In this table, first column represents the type of the graph. Second column represents the number of nodes and number of edges (for general graphs) or range (UDG) in the instance group. The next 18 columns provides either the solution quality (column labelled "v") or execution time (column labelled "t") for a particular parameter value listed above it. Solution quality and execution time under the head "Reported" are those obtained with parameter settings used throughout this paper. We can see increasing N_0/N_1 can improve the solution quality slightly at the expense of increased execution times. On the other hand, opposite effects were observed on solution quality and execution time when we have decreased N_0/N_1. This is on expected lines as increase in values of these two parameters cannot harm solution quality. We have already explained the rationale for retaining $N_0 = 75$ and $N_1 = 150$ in Section 6.2.1. Parameter p_r does not have any impact on solution quality on these 8 instance groups. This is because a solution constructed by H-MECU either has no redundant node or only a few redundant nodes, and even in the former case, constructed solution can be the best solution sometimes. In the latter case, number of redundant nodes decreases rapidly when we start removing such nodes. As one can see, the value of parameter A is so chosen that whether we increase or decrease it, there will be deterioration in solution quality in most cases.

Table 11
Impact of different parameters of MS-ILS on solution quality.

Graph	$(\|V\|, \|E\|/R)$	Reported		$N_0 = 50$		$N_0 = 100$		$N_1 = 125$		$N_1 = 175$		$p_r = 0.3$		$p_r = 0.5$		A = 4		A = 6	
		v	t	v	t	v	t	v	t	v	t	v	t	v	t	v	t	v	t
Uni-UDG	(800, 150)	22.3	15.8	22.4	10.5	22.2	20.8	22.5	13.1	22.2	18.3	22.3	15.6	22.3	15.6	22.0	15.7	22.8	15.8
	(1000, 200)	13.2	30.1	13.4	20.2	13.2	39.6	13.2	25.1	13.2	34.6	13.2	29.8	13.2	29.8	13.6	29.8	13.7	28.1
Uni-General	(800, 5000)	97.9	31.0	97.9	20.5	97.6	40.7	98.0	25.9	97.7	35.5	97.9	30.7	97.9	30.7	98.2	29.6	97.6	31.9
	(1000, 10 000)	89.5	55.4	89.8	36.3	89.4	73.4	89.7	45.6	89.4	64.9	89.5	54.9	89.5	54.6	90.5	54.1	89.2	55.7
Var-UDG	(800, 150)	24.6	16.5	24.8	11.0	24.4	22.0	24.8	13.7	24.5	19.0	24.6	16.4	24.6	16.4	24.8	16.4	24.9	15.9
	(1000, 200)	14.9	31.2	14.9	20.7	14.8	41.5	14.9	26.1	14.9	36.0	14.9	31.1	14.9	31.0	14.7	31.6	15.2	29.4
Var-General	(800, 5000)	110.5	33.8	111.1	21.9	110.0	44.0	110.8	27.3	110.5	38.6	110.5	33.1	110.5	33.1	111.6	31.5	110.7	33.4
	(1000, 10 000)	97.8	60.3	98.0	39.1	97.6	80.0	97.9	49.1	97.7	69.9	97.8	59.7	97.8	59.7	98.4	58.2	97.6	61.2
Average:		**58.84**	**34.26**	**59.04**	**22.53**	**58.65**	**45.25**	**58.98**	**28.24**	**58.76**	**39.60**	**58.84**	**33.91**	**58.84**	**33.86**	**59.23**	**33.36**	**58.96**	**33.93**

Table 12
Impact of different components of MS-ILS on solution quality.

Graph	$(\|V\|, \|E\|/R)$	Reported		Case I		Case II		Case III	
		v	t	v	t	v	t	v	t
Uni-UDG	(800, 150)	22.3	15.8	22.5	14.4	22.6	14.8	22.6	15.5
	(1000, 200)	13.2	30.1	13.8	28.3	13.7	28.6	13.7	29.9
Uni-general	(800, 5000)	97.9	31.0	103.0	29.6	98.1	28.7	97.9	31.3
	(1000, 10 000)	89.5	55.4	94.4	54.3	90.5	54.2	90.0	55.6
Var-UDG	(800, 150)	24.6	16.5	24.8	15.0	25.0	15.6	24.6	16.3
	(1000, 200)	14.9	31.2	14.8	30.6	14.9	31.3	15.1	31.6
Var-general	(800, 5000)	110.5	33.8	117.2	32.7	111.1	29.9	110.8	33.3
	(1000, 10 000)	97.8	60.3	103.3	61.5	98.4	58.0	98.1	60.5
Average:		**58.84**	**34.26**	**61.73**	**33.30**	**59.29**	**32.64**	**59.10**	**34.25**

Table 13
Comparison of LB_1, LB_2, relaxed versions of ILP-PD and ILP for UDG graph instances with an uniform capacity of 2, 5, and average degree (α) for every node.

$\|V\|$	Range	LB1			LB2			ILP-PD			ILP			ILP-PD Time			ILP Time		
		2	5	α	2	5	α	2	5	α	2	5	α	2	5	α	2	5	α
50	150	**17.0**	9.0	13.0	**17.0**	9.1	13.0	14.1	**12.9**	13.6	**17.0**	**12.9**	**14.1**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
50	200	**17.0**	9.0	9.0	**17.0**	9.0	9.0	10.1	9.5	9.5	**17.0**	**10.0**	**10.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
100	150	**34.0**	17.0	17.0	**34.0**	17.0	17.0	18.1	17.5	17.5	**34.0**	**18.4**	**18.4**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
100	200	**34.0**	17.0	10.0	**34.0**	17.0	10.0	11.5	10.9	10.7	**34.0**	**17.4**	**11.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
250	150	**84.0**	**42.0**	16.0	**84.0**	**42.0**	16.0	18.7	18.0	17.7	**84.0**	**42.0**	**18.0**	0.5	0.3	**0.1**	**0.1**	**0.2**	0.4
250	200	**84.0**	**42.0**	10.0	**84.0**	**42.0**	10.0	11.7	11.0	10.9	**84.0**	**42.0**	**11.0**	1.4	0.9	**0.2**	**0.1**	**0.5**	1.2
500	150	**167.0**	**84.0**	16.0	**167.0**	**84.0**	16.0	19.0	18.3	**18.1**	**167.0**	**84.0**	**18.1**	8.4	5.6	**1.1**	**0.5**	**3.4**	5.5
500	200	**167.0**	**84.0**	10.0	**167.0**	**84.0**	10.0	12.0	11.2	**11.0**	**167.0**	**84.0**	**11.0**	20.5	9.7	**1.9**	**1.0**	**6.8**	25.3
800	150	**267.0**	**134.0**	16.0	**267.0**	**134.0**	16.0	19.2	18.6	**18.3**	**267.0**	**134.0**	**18.3**	61.4	32.2	**6.3**	**2.3**	**23.8**	26.0
800	200	**267.0**	**134.0**	10.0	**267.0**	**134.0**	10.0	12.0	11.6	**11.0**	**267.0**	**134.0**	**11.0**	126.2	69.9	**9.5**	**5.3**	**42.5**	230.8
1000	150	**334.0**	**167.0**	16.0	**334.0**	**167.0**	16.0	19.2	18.7	**18.2**	**334.0**	**167.0**	**18.2**	128.0	82.3	**15.7**	**4.8**	**58.8**	63.2
1000	200	**334.0**	**167.0**	10.0	**334.0**	**167.0**	10.0	12.0	11.9	**11.0**	**334.0**	**167.0**	dnf(2)	327.6	191.6	**22.7**	**11.5**	**98.1**	704.6

Table 14
Comparison of LB_1, LB_2, relaxed versions of ILP-PD and ILP for general graph instances with an uniform capacity of 2, 5, and average degree (α) for every node.

$\|V\|$	$\|E\|$	LB1			LB2			ILP-PD			ILP			ILP-PD Time			ILP Time		
		2	5	α	2	5	α	2	5	α	2	5	α	2	5	α	2	5	α
50	100	**17.0**	9.0	10.0	**17.0**	9.0	10.0	11.5	**11.5**	**11.5**	**17.0**	**11.5**	**11.5**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
50	250	**17.0**	**9.0**	5.0	**17.0**	**9.0**	5.0	5.3	5.2	**5.2**	**17.0**	**9.0**	**5.2**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
50	500	**17.0**	**9.0**	**3.0**	**17.0**	**9.0**	**3.0**	3.0	3.0	**3.0**	**17.0**	**9.0**	**3.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
100	100	**34.0**	25.0	**34.0**	**34.0**	33.0	**34.0**	**34.0**	**33.6**	**34.0**	**34.0**	**33.6**	**34.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
100	250	**34.0**	17.0	17.0	**34.0**	17.0	17.0	19.0	**18.8**	**18.8**	**34.0**	**18.8**	**18.8**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
100	500	**34.0**	**17.0**	10.0	**34.0**	**17.0**	10.0	10.2	10.2	**10.2**	**34.0**	**17.0**	**10.2**	0.1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
250	250	**84.0**	63.0	**84.0**	**84.0**	83.0	**84.0**	**84.0**	**83.3**	**84.0**	**84.0**	**83.3**	**84.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
250	500	**84.0**	42.0	50.0	**84.0**	42.0	50.0	56.5	**55.7**	55.7	**84.0**	**55.7**	**56.2**	0.1	**0.0**	**0.0**	**0.0**	0.1	0.1
250	1000	**84.0**	**42.0**	28.0	**84.0**	**42.0**	28.0	31.6	30.8	**30.8**	**84.0**	**42.0**	**30.8**	0.3	**0.1**	**0.1**	**0.1**	0.3	0.4
500	500	**167.0**	125.0	**167.0**	**167.0**	166.1	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
500	1000	**167.0**	84.0	100.0	**167.0**	84.0	100.0	111.5	110.0	110.0	**167.0**	**110.1**	**111.2**	0.3	**0.1**	**0.1**	**0.2**	0.3	0.4
500	2000	**167.0**	**84.0**	56.0	**167.0**	**84.0**	56.0	62.4	60.9	60.9	**167.0**	**84.0**	**61.0**	1.8	**0.8**	**0.3**	**0.4**	1.5	1.9
800	1000	**267.0**	134.0	**267.0**	**267.0**	180.1	**267.0**	241.2	**240.5**	241.2	**267.0**	**240.5**	**267.0**	**0.1**	0.0	**0.1**	**0.1**	0.2	0.2
800	2000	**267.0**	134.0	134.0	**267.0**	134.0	134.0	149.9	147.6	147.6	**267.0**	**148.6**	**148.6**	2.0	**0.5**	**0.5**	**0.8**	2.1	2.1
800	5000	**267.0**	**134.0**	62.0	**267.0**	**134.0**	62.0	65.2	63.5	63.5	**267.0**	**134.0**	**63.7**	12.0	**6.4**	**2.5**	**1.5**	8.5	11.7
1000	1000	**334.0**	250.0	**334.0**	**334.0**	333.0	**334.0**	**334.0**	**333.7**	**334.0**	**334.0**	**333.7**	**334.0**	**0.0**	**0.0**	**0.0**	0.1	0.1	0.1
1000	5000	**334.0**	**167.0**	91.0	**334.0**	**167.0**	91.0	100.9	98.4	**98.4**	**334.0**	**167.0**	**98.4**	15.8	**7.3**	**2.9**	**2.1**	12.2	14.7
1000	10 000	**334.0**	**167.0**	48.0	**334.0**	**167.0**	48.0	51.9	50.2	**50.2**	**334.0**	**167.0**	**50.2**	50.4	67.9	**8.2**	**2.7**	**25.5**	36.9

Table 15
Comparison of LB_1, LB_2, relaxed versions of ILP-PD and ILP for UDG graph instances where capacity of each node vary randomly over $\{2, 5\}$, $\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$ and $\{1, \ldots, \alpha\}$.

$\|V\|$	Range	LB1			LB2			ILP-PD			ILP			ILP-PD Time			ILP Time		
		$\{2, 5\}$	$(\alpha/5,\ \alpha/2)$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$(\alpha/5,\ \alpha/2)$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$(\alpha/5,\ \alpha/2)$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$(\alpha/5,\ \alpha/2)$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$(\alpha/5,\ \alpha/2)$	$\{1, \ldots, \alpha\}$	$\{2, 5\}$	$(\alpha/5,\ \alpha/2)$	$\{1, \ldots, \alpha\}$
50	150	9.0	25.0	13.4	9.7	25.0	13.4	13.7	15.0	14.4	**14.3**	**25.2**	**16.3**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
50	200	9.0	16.6	9.0	9.0	16.6	9.4	9.9	10.5	10.0	**11.1**	**17.1**	**11.6**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
100	150	17.0	28.6	15.8	17.0	28.6	16.3	17.7	18.1	17.9	**21.0**	**29.3**	**20.7**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
100	200	17.0	17.0	9.7	17.0	17.0	9.7	11.1	11.1	10.8	**17.6**	**17.7**	**12.3**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
250	150	42.0	**31.6**	16.1	42.0	**31.6**	16.1	18.5	18.2	18.1	**42.1**	**31.6**	**20.1**	0.4	0.4	**0.2**	**0.2**	**0.3**	0.3
250	200	**42.0**	18.8	10.0	**42.0**	18.8	10.0	11.2	11.0	11.0	**42.0**	**18.9**	**12.0**	1.2	**0.8**	**0.6**	**0.3**	0.9	0.9
500	150	**84.0**	**32.0**	16.6	**84.0**	**32.0**	16.6	18.7	18.1	18.2	**84.0**	**32.0**	**20.3**	6.6	**3.9**	**3.6**	**2.1**	4.9	6.6
500	200	**84.0**	**19.4**	10.0	**84.0**	**19.4**	10.0	11.6	11.0	11.0	**84.0**	**19.4**	**12.0**	13.4	**7.0**	**5.7**	**3.5**	15.7	11.3
800	150	**134.0**	**31.7**	16.1	**134.0**	**31.7**	16.7	18.8	18.3	18.3	**134.0**	**31.7**	**19.9**	41.8	**26.0**	**20.4**	**12.0**	55.5	47.6
800	200	**134.0**	**19.6**	10.0	**134.0**	**19.6**	10.0	12.0	11.0	11.0	**134.0**	dnf(1)	**12.0**	89.4	**39.0**	**33.5**	**13.0**	376.1	83.8
1000	150	**167.0**	**32.5**	16.5	**167.0**	**32.5**	16.6	18.9	18.2	18.3	**167.0**	**32.5**	**19.8**	104.7	**63.9**	**51.2**	**13.4**	387.4	132.6
1000	200	**167.0**	**19.2**	10.0	**167.0**	**19.2**	10.0	12.0	11.1	11.0	**167.0**	dnf(10)	**12.0**	239.5	**104.1**	**80.7**	**27.0**	1000.2	226.7

To understand the impact of different components of MS-ILS on solution quality, we have studied the following cases:

- Case I: Steps 1 and 2 (Section 3.1.1) in dominating node selection is not used. This means no preferential treatment to isolated or degree one nodes
- Case II: Step 1 (Section 3.1.1) in dominating node selection is not used. This mean no preferential treatment for isolated nodes. However, degree one nodes are still getting preferential treatment
- Case III: Redundant node removal procedure (Section 3.1.3) is not used

We have utilized the same 8 instance groups as used in sensitivity analysis of parameter settings. Table 12 presents the results of this study in the same format as in Table 11. This table clearly vindicates our dominating node selection strategy. Benefit of redundant node removal procedure can also be seen.

6.3. ILP and MH results

In this section, we will present the results of ILP and MH, and compare them with CMSA and ILP-PD approaches of [16]. As MH is a combination of MS-ILS and ILP, we have also included MS-ILS in this comparison. Results of CMSA and ILP-PD are taken from [16]. Like the approaches of [16], we have allowed ILP and MH to execute for a maximum of 1000 s. When these approaches terminate before this time, then a proven optimal solution is found. In case of MH, it also terminates when it finds the optimal solution on reduced set of vertices because that is the best solution MH can achieve with that particular reduced set of vertices. For MH, we have set K_f to 1 for the graphs having number of nodes greater than 250, otherwise K_f is set to 0. We have found that a higher value of K_f leads to faster termination, but solutions of slightly inferior quality. With maximum permissible time of 1000 s, $K_f = 1$ provides best results on large instances. If we

Table 16
Comparison of LB_1, LB_2, relaxed versions of ILP-PD and ILP for general graph instances where capacity of each node vary randomly over $\{2,5\}$, $\{\frac{\alpha}{5},\frac{\alpha}{2}\}$ and $\{1,\ldots,\alpha\}$.

$\|V\|$	$\|E\|$	LB1			LB2			ILP-PD			ILP			ILP-PD Time			ILP Time		
		{2, 5}	(α/5, α/2)	{1, ..., α}	{2, 5}	(α/5, α/2)	{1, ..., α}	{2, 5}	(α/5, α/2)	{1, ..., α}	{2, 5}	(α/5, α/2)	{1, ..., α}	{2, 5}	(α/5, α/2)	{1, ..., α}	{2, 5}	(α/5, α/2)	{1, ..., α}
50	100	9.0	17.0	10.0	9.0	17.0	11.4	11.5	12.0	11.7	**12.4**	**17.7**	**13.7**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
50	250	**9.0**	**9.0**	5.0	**9.0**	**9.0**	5.6	5.2	5.2	5.2	**9.0**	**9.0**	**6.8**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
50	500	**9.0**	**5.0**	**3.0**	**9.0**	**5.0**	**3.0**	3.0	3.0	**3.0**	**9.0**	**5.0**	**3.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
100	100	25.9	**50.0**	34.0	33.1	**50.0**	34.0	**33.7**	34.0	34.0	**33.7**	**50.0**	**40.1**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
100	250	17.0	34.0	17.0	17.0	34.0	18.6	18.8	19.9	19.1	**21.6**	**34.3**	**22.8**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
100	500	**17.0**	**17.0**	10.0	**17.0**	**17.0**	10.0	10.2	10.2	10.2	**17.0**	**17.0**	**12.3**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.1
250	250	67.2	**125.0**	84.0	83.2	**125.0**	84.0	**83.7**	84.0	84.0	**83.7**	**125.0**	**98.9**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
250	500	42.0	84.0	50.0	42.7	84.0	52.6	56.2	58.6	57.3	**62.0**	**86.8**	**66.5**	**0.0**	0.1	**0.0**	0.1	**0.0**	0.1
250	1000	42.0	50.0	28.0	42.0	50.0	29.1	31.1	32.6	31.3	**43.5**	**51.4**	**37.0**	**0.2**	0.3	**0.2**	0.3	**0.2**	0.3
500	500	129.2	**250.0**	167.0	166.8	**250.0**	167.0	**167.0**	167.0	167.0	**167.0**	**250.0**	**202.7**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
500	1000	84.0	167.0	100.0	84.2	167.0	108.3	110.8	116.0	113.1	**122.2**	**172.9**	**134.7**	**0.2**	0.4	**0.2**	0.4	**0.2**	0.3
500	2000	84.0	100.0	56.0	84.0	100.0	58.6	61.3	64.3	61.9	**85.9**	**101.3**	**74.0**	**1.3**	1.6	**1.0**	1.8	**0.9**	1.7
800	1000	134.0	**400.0**	267.0	199.5	**400.0**	267.0	240.9	249.3	246.1	**246.7**	**400.0**	**300.2**	**0.1**	0.2	**0.1**	0.2	**0.1**	**0.1**
800	2000	134.0	267.0	134.0	134.0	267.0	142.2	148.5	157.6	151.5	**171.8**	**273.4**	**179.7**	**1.1**	2.5	**1.3**	1.9	**0.5**	1.6
800	5000	134.0	**115.0**	62.0	134.0	**115.0**	63.7	64.3	64.0	64.3	**134.1**	**115.0**	**78.6**	10.0	9.0	**6.2**	**5.5**	**7.6**	15.2
1000	1000	258.4	**500.0**	334.0	333.1	**500.0**	334.0	**333.8**	334.0	334.0	**333.8**	**500.0**	**400.8**	**0.0**	**0.0**	**0.0**	0.1	0.1	0.1
1000	5000	167.0	167.0	91.0	167.0	167.0	95.1	99.4	99.2	99.4	**169.0**	**168.1**	**119.8**	11.6	11.0	**7.6**	**7.4**	**7.4**	14.6
1000	10 000	**167.0**	**91.0**	48.0	**167.0**	**91.0**	49.1	51.0	50.2	50.7	**167.0**	**91.0**	**60.9**	69.6	78.8	**30.1**	**16.0**	**30.0**	85.3

Table 17
Cardinality of CAPMDS computed using ILP-PD, CMSA, MS-ILS, ILP and MH for UDG graph instances with an uniform capacity of 2, 5, and average degree (α) for every node.

$\|V\|$	Range	ILP-PD			CMSA			MS-ILS			ILP			MH		
		2	5	α	2	5	α	2	5	α	2	5	α	2	5	α
50	150	**17.2**	**12.9**	**14.4**	**17.2**	**12.9**	**14.4**	**17.2**	13.0	**14.4**	**17.2**	**12.9**	**14.4**	**17.2**	**12.9**	**14.4**
50	200	**17.0**	**10.0**	**10.0**	**17.0**	**10.0**	**10.0**	**17.0**	10.2	10.2	**17.0**	**10.0**	**10.0**	**17.0**	**10.0**	**10.0**
100	150	**34.0**	**18.7**	**18.7**	**34.0**	**18.7**	**18.7**	**34.0**	18.8	18.8	**34.0**	**18.7**	**18.7**	**34.0**	**18.7**	**18.7**
100	200	**34.0**	**17.4**	**11.0**	**34.0**	**17.4**	**11.0**	**34.0**	**17.4**	**11.0**	**34.0**	**17.4**	**11.0**	**34.0**	**17.4**	**11.0**
250	150	**84.0**	**42.0**	**18.5**	**84.0**	**42.0**	18.6	**84.0**	**42.0**	19.6	**84.0**	**42.0**	**18.5**	**84.0**	**42.0**	**18.5**
250	200	**84.0**	**42.0**	**11.3**	**84.0**	**42.0**	11.4	**84.0**	**42.0**	11.9	**84.0**	**42.0**	**11.3**	**84.0**	**42.0**	**11.3**
500	150	**167.0**	**84.0**	**18.6**	**167.0**	**84.0**	18.8	**167.0**	**84.0**	21.2	**167.0**	**84.0**	**18.6**	**167.0**	**84.0**	18.7
500	200	**167.0**	**84.0**	**11.3**	**167.0**	**84.0**	11.7	**167.0**	**84.0**	12.8	**167.0**	**84.0**	**11.3**	**167.0**	**84.0**	11.6
800	150	**267.0**	**134.0**	19.4	**267.0**	**134.0**	19.5	**267.0**	**134.0**	22.3	**267.0**	**134.0**	19.3	**267.0**	**134.0**	**19.1**
800	200	**267.0**	**134.0**	12.3	**267.0**	**134.0**	12.0	**267.0**	**134.0**	13.1	**267.0**	**134.0**	**11.8**	**267.0**	**134.0**	**11.8**
1000	150	**334.0**	**167.0**	25.8	**334.0**	**167.0**	19.7	**334.0**	**167.0**	23.1	**334.0**	**167.0**	192.4	**334.0**	**167.0**	**19.5**
1000	200	**334.0**	173.3	14.7	**334.0**	**167.0**	**12.0**	**334.0**	**167.0**	13.2	**334.0**	**167.0**	12.8	**334.0**	**167.0**	12.1
Avg:		**150.5**	76.6	15.5	**150.5**	**76.1**	14.8	**150.5**	**76.1**	16.0	**150.5**	**76.1**	29.2	**150.5**	**76.1**	**14.7**

reduce this maximum permissible time, higher value of K_f can provide better results.

6.3.1. Results of relaxed versions of ILP-PD and ILP

Before we present the results of ILP and MH, we compare the linear programming bounds (*i.e.*, the continuous relaxation) of ILP-PD and ILP to ascertain the benefits of our proposed improvements to ILP-PD. For doing this, we will make use of our own implementation of ILP-PD. We will also include the two lower bounds proposed by us, viz. LB_1 and LB_2 in this comparison. Results are averages of integer lower bounds over group of 10 instances. As LB_1 and LB_2 can be computed in negligible times, we have not reported the times to compute them. Both relaxed versions are allowed to execute for maximum time of 1000 s. If they cannot find the optimal solution in this permissible time on any of the instances in the group, then we will have *dnf* (did not finish) in the corresponding column with number of instances where corresponding relaxed version is not able to finish in parenthesis.

Table 13 provides the lower bounds obtained by different methods on UDG instances with uniform capacity. This table clearly shows that relaxed ILP has obtained tighter bounds compared with other three methods. On most of the instances LB_1 and LB_2 are equal. On the other hand, ILP-PD has obtained 15 better and 21 worst bounds when compared with LB_1 and LB_2. And it got 1 better (the instance group where ILP is not able to finish on two instances in maximum permissible time), 6 equal and 29 worst bounds when compared with ILP. From the table, it is clear that all instance groups, except for one instance group, got tightest lower bounds through relaxed ILP. Computational time of ILP is lesser than ILP-PD for the graphs with capacity 2 and 5, and it requires higher computational time on the graphs with average degree (α) as capacity.

Table 14 reports the lower bounds obtained by different methods on general graph instances with uniform capacity. Here, LB_2 is better than LB_1. ILP-PD returns a tighter bounds in comparison to LB_1 and LB_2. But compared with ILP, ILP-PD has got 30 inferior bounds. ILP returns the tightest bounds on all 54 instance groups. Here also, ILP-PD has taken higher time compared with ILP for the graphs with capacity 2 or 5, and has taken lesser time for the graphs with α as capacity.

Table 15 presents the lower bounds obtained by different methods on UDG instances with variable capacity. LB_1 and LB_2 are tighter than the bound returned by ILP-PD, and LB_2 is tighter than LB_1. Here also, due to the size of the instances, ILP did not finish on two instance groups. LB_2 and LB_1 are equal to the bound of ILP on 12 instance groups, and worse on 22 groups. On the other hand, ILP-PD has got worst lower bounds on 34 instance groups compared with ILP. The remaining two instance groups are those where ILP could not finish. ILP has higher computational cost on the UDG graphs having capacity $\{\frac{\alpha}{5},\frac{\alpha}{2}\}$ and $\{1,\ldots,\alpha\}$ in comparison to ILP-PD.

Table 16 reports the lower bounds obtained by different methods on general graph instances with variable capacity. Similar to the previous tables here also ILP has obtained the tightest lower bounds on all 54 instance groups. ILP-PD has obtained tightest bounds on 5 instance groups. LB_1 and LB_2 have reached tightest bounds on 15 instance groups. Computational time of ILP is less than that of ILP-PD on the graphs having capacity $\{2,5\}$ and $\{\frac{\alpha}{5},\frac{\alpha}{2}\}$, whereas on the graphs having capacity $\{1,\ldots,\alpha\}$ ILP has taken more computational time.

These four tables clearly show the superiority of the relaxed version of ILP over relaxed version ILP-PD as it is able to obtain as good as or better lower bounds on 177 instance groups out of 180 in comparison to ILP-PD. Same is the case with respect to LB_1 and LB_2. The remaining three instance groups are those where ILP

Table 18
Cardinality of CAPMDS computed using ILP-PD, CMSA, MS-ILS, ILP and MH for general graph instances with an uniform capacity of 2, 5, average degree (α) for every node.

$\|V\|$	$\|E\|$	ILP-PD			CMSA			MS-ILS			ILP			MH		
		2	5	α	2	5	α	2	5	α	2	5	α	2	5	α
50	100	**17.0**	**11.9**	**12.0**	**17.0**	**11.9**	**12.0**	**17.0**	12.0	**12.0**	**17.0**	**11.9**	**12.0**	**17.0**	**11.9**	**12.0**
50	250	**17.0**	**9.0**	**6.0**	**17.0**	**9.0**	**6.0**	**17.0**	**9.0**	6.1	**17.0**	**9.0**	**6.0**	**17.0**	**9.0**	**6.0**
50	500	**17.0**	**9.0**	**3.8**	**17.0**	**9.0**	**3.8**	**17.0**	**9.0**	3.9	**17.0**	**9.0**	**3.8**	**17.0**	**9.0**	**3.8**
100	100	**34.0**	**33.6**	**34.0**	**34.0**	**33.6**	**34.0**	**34.0**	**33.6**	**34.0**	**34.0**	**33.6**	**34.0**	**34.0**	**33.6**	**34.0**
100	250	**34.0**	**20.0**	**20.0**	**34.0**	**20.0**	20.1	**34.0**	20.2	20.2	**34.0**	**20.0**	**20.0**	**34.0**	**20.0**	**20.0**
100	500	**34.0**	**17.0**	**12.2**	**34.0**	**17.0**	**12.2**	**34.0**	**17.0**	12.5	**34.0**	**17.0**	**12.2**	**34.0**	**17.0**	**12.2**
250	250	**84.0**	**83.3**	**84.0**	**84.0**	**83.3**	**84.0**	**84.0**	**83.3**	**84.0**	**84.0**	**83.3**	**84.0**	**84.0**	**83.3**	**84.0**
250	500	**84.0**	**57.8**	**58.3**	**84.0**	**57.8**	**58.3**	**84.0**	59.1	60.3	**84.0**	**57.8**	**58.3**	**84.0**	**57.8**	**58.3**
250	1000	**84.0**	**42.0**	36.7	**84.0**	**42.0**	36.7	**84.0**	44.3	38.5	**84.0**	**42.0**	**36.4**	**84.0**	**42.0**	**36.4**
500	500	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**	**167.0**
500	1000	**167.0**	**114.9**	116.2	**167.0**	115.1	116.8	**167.0**	119.8	122.4	**167.0**	115.0	**116.1**	**167.0**	115.1	**116.1**
500	2000	**167.0**	84.1	75.3	**167.0**	**84.0**	**73.7**	**167.0**	90.9	79.8	**167.0**	**84.0**	74.1	**167.0**	**84.0**	74.2
800	1000	**267.0**	**242.5**	**267.0**	**267.0**	243.1	**267.0**	**267.0**	244.1	**267.0**	**267.0**	**242.5**	**267.0**	**267.0**	242.6	**267.0**
800	2000	**267.0**	164.8	164.9	**267.0**	**162.8**	**162.8**	**267.0**	174.2	174.2	**267.0**	163.6	163.7	**267.0**	163.3	163.0
800	5000	**267.0**	**134.0**	91.1	**267.0**	**134.0**	93.0	**267.0**	139.0	97.9	**267.0**	**134.0**	90.3	**267.0**	**134.0**	**90.2**
1000	1000	**334.0**	**333.7**	**334.0**	**334.0**	**333.7**	**334.0**	**334.0**	**333.7**	**334.0**	**334.0**	**333.7**	**334.0**	**334.0**	**333.7**	**334.0**
1000	5000	**334.0**	**167.0**	132.5	**334.0**	**167.0**	135.0	**334.0**	177.2	142.2	**334.0**	**167.0**	132.0	**334.0**	**167.0**	**131.7**
1000	10 000	**334.0**	**167.0**	87.6	**334.0**	**167.0**	86.8	**334.0**	169.7	89.5	**334.0**	**167.0**	**80.9**	**334.0**	**167.0**	81.0
Avg:		**150.5**	103.3	94.6	**150.5**	**103.2**	94.6	**150.5**	105.7	97.0	**150.5**	**103.2**	94.0	**150.5**	**103.2**	**93.9**

Table 19
Cardinality of CAPMDS computed using ILP-PD, CMSA, MS-ILS, ILP and MH for UDG graph instances where capacity of each node vary randomly over $\{2, 5\}$, $\{\frac{\alpha}{5}, \frac{\alpha}{2}\}$ and $\{1, \ldots, \alpha\}$.

$\|V\|$	Range	ILP-PD			CMSA			MS-ILS			ILP			MH		
		$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$
50	150	**14.5**	**25.4**	**16.7**	**14.5**	**25.4**	**16.7**	**14.5**	**25.4**	16.8	**14.5**	**25.4**	**16.7**	**14.5**	**25.4**	**16.7**
50	200	**11.1**	**17.3**	**12.1**	**11.1**	**17.3**	**12.1**	**11.1**	**17.3**	12.3	**11.1**	**17.3**	**12.1**	**11.1**	**17.3**	**12.1**
100	150	**21.8**	**29.9**	**21.4**	**21.8**	**29.9**	**21.4**	21.9	**29.9**	21.8	**21.8**	**29.9**	**21.4**	**21.8**	**29.9**	**21.4**
100	200	**17.6**	**17.8**	**12.8**	**17.6**	**17.8**	**12.8**	17.7	**17.8**	13.2	**17.6**	**17.8**	**12.8**	**17.6**	**17.8**	**12.8**
250	150	**42.1**	**31.6**	**20.8**	**42.1**	**31.6**	**20.8**	**42.1**	31.9	22.6	**42.1**	**31.6**	**20.8**	**42.1**	**31.6**	**20.8**
250	200	**42.0**	**18.9**	**12.8**	**42.0**	**18.9**	**12.8**	**42.0**	**18.9**	14.1	**42.0**	**18.9**	**12.8**	**42.0**	**18.9**	**12.8**
500	150	**84.0**	**32.0**	21.9	**84.0**	**32.0**	21.1	**84.0**	**32.0**	24.2	**84.0**	**32.0**	**20.9**	**84.0**	**32.0**	21.2
500	200	**84.0**	79.6	13.2	**84.0**	**19.4**	**12.5**	**84.0**	**19.4**	14.5	**84.0**	**19.4**	**12.5**	**84.0**	**19.4**	12.8
800	150	**134.0**	732.0	22.7	**134.0**	**31.7**	**21.1**	**134.0**	31.8	24.6	**134.0**	**31.7**	21.8	**134.0**	**31.7**	21.3
800	200	**134.0**	796.6	726.9	**134.0**	**19.6**	12.7	**134.0**	**19.6**	14.7	**134.0**	91.9	91.9	**134.0**	**19.6**	**12.6**
1000	150	**167.0**	997.8	909.2	**167.0**	**32.5**	**21.3**	**167.0**	**32.5**	25.5	**167.0**	592.9	414.8	**167.0**	**32.5**	21.5
1000	200	**167.0**	996.0	1000.0	**167.0**	**19.2**	**12.6**	**167.0**	**19.2**	14.9	**167.0**	935.0	661.5	**167.0**	**19.2**	12.7
Avg:		**76.6**	314.6	232.5	**76.6**	**24.6**	**16.5**	**76.6**	**24.6**	18.3	**76.6**	153.7	110.0	**76.6**	**24.6**	16.6

could not finish on some instances in maximum permissible time of 1000 s. Such an amount of time can be considered too large for the computation of a lower bound, and it shows that linear programs are not good candidates for providing lower bounds for large instances.

6.3.2. Results on uniform capacity instances

Now, we will present the results of ILP and MH along with those of ILP-PD and CMSA. The results are presented in Tables 17 to 20. The meaning of first two columns, last row and the bold font remain the same in these tables as in Tables 2 to 5. In this section, we will report the results on uniform capacity instances, and in the next section on variable capacity instances. Again the performance will be reported separately for Unit Disk Graphs (UDGs) and General Graphs.

Performance on UDGs: Table 17 provides the computational results of different approaches on uniform capacity UDG instances. When compared with ILP-PD, our approaches ILP and MH obtained better or equal solutions in 35 and 34 cases respectively out of 36 cases. Compared with CMSA, our approaches ILP and MH obtained better or equal solutions in 34 and 35 cases respectively out of 36. If the capacity of a graph is 2 or 5 then almost all approaches got optimal solutions. If the capacity of a graph is equal to its average degree (α) then ILP and MH has shown better performance in comparison to other three approaches. ILP and MH each of them have obtained as good as or better solutions on 33 instance groups out of 36 in comparison to other approaches. If we compare ILP and MH among themselves, they performed same on 31 instances with ILP better on remaining 2 and MH on 3. If we look at the averages of different approaches in the last row, we can see MH performed as good as or better than all other approaches.

Performance on General Graphs: Table 18 reports the computational results of different approaches on general graph instances with uniform capacity. In comparison to ILP-PD, our approaches ILP and MH obtained as good as or better solutions in 53 and 52 cases respectively out of a total of 54 cases. Compared with CMSA, our approaches ILP and MH obtained better or equal solutions in 51 cases out of 54 cases. If the ILP and MH are compared among themselves then they performed same in 46 cases with ILP and MH performing better on 4 each out of 8 remaining cases. If the capacity of the graph is 2 then all approaches got optimal solutions. For the large graphs with capacity of a graph is equal to 5 or its average degree (α), then ILP and MH has shown much better performance than other three approaches. On overall, ILP and MH each of them has obtained better or equal solutions on 48 instance groups out of 54 in comparison to other approaches. In comparison to other approaches, MH again obtained as good as or better averages.

From these two tables, compared with ILP-PD, CMSA, and MS-ILS, each of our approaches ILP and MH have obtained 81 better or equal results out of 90 uniform capacity graph instances.

Table 20
Cardinality of CAPMDS computed using ILP-PD, CMSA, MS-ILS, ILP and MH for general graph instances where capacity of each node vary randomly over $\{2,5\}$, $\{\frac{\alpha}{5},\frac{\alpha}{2}\}$ and $\{1,\ldots,\alpha\}$.

$\lvert V\rvert$	$\lvert E\rvert$	ILP-PD			CMSA			MS-ILS			ILP			MH		
		$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$
50	100	**13.0**	**17.7**	**13.7**	**13.0**	**17.7**	**13.7**	**13.0**	**17.7**	**13.7**	**13.0**	**17.7**	**13.7**	**13.0**	**17.7**	**13.7**
50	250	**9.0**	**9.0**	**7.2**	**9.0**	**9.0**	**7.2**	**9.0**	**9.0**	7.6	**9.0**	**9.0**	**7.2**	**9.0**	**9.0**	**7.2**
50	500	**9.0**	**5.0**	**3.9**	**9.0**	**5.0**	**3.9**	**9.0**	**5.0**	4.1	**9.0**	**5.0**	**3.9**	**9.0**	**5.0**	**3.9**
100	100	**33.7**	**50.0**	**40.1**	**33.7**	**50.0**	**40.1**	**33.7**	**50.0**	**40.1**	**33.7**	**50.0**	**40.1**	**33.7**	**50.0**	**40.1**
100	250	**21.9**	**34.3**	**23.3**	**21.9**	**34.3**	**23.3**	22.2	**34.3**	24.0	**21.9**	**34.3**	**23.3**	**21.9**	**34.3**	**23.3**
100	500	**17.0**	**17.0**	**13.5**	**17.0**	**17.0**	13.6	17.1	17.3	14.5	**17.0**	**17.0**	**13.5**	**17.0**	**17.0**	**13.5**
250	250	**83.7**	**125.0**	**98.9**	**83.7**	**125.0**	**98.9**	**83.7**	**125.0**	**98.9**	**83.7**	**125.0**	**98.9**	**83.7**	**125.0**	**98.9**
250	500	**63.2**	**86.8**	**67.3**	63.3	**86.8**	67.4	65.0	88.9	70.3	**63.2**	**86.8**	**67.3**	**63.2**	**86.8**	**67.3**
250	1000	43.7	**51.4**	40.5	**43.6**	**51.4**	40.2	48.5	54.9	44.2	**43.6**	**51.4**	**40.0**	**43.6**	**51.4**	**40.0**
500	500	**167.0**	**250.0**	**202.7**	**167.0**	**250.0**	**202.7**	**167.0**	**250.0**	**202.7**	**167.0**	**250.0**	**202.7**	**167.0**	**250.0**	**202.7**
500	1000	125.5	**172.9**	**135.9**	125.4	**172.9**	136.3	131.7	178.0	146.0	**125.2**	**172.9**	**135.9**	**125.2**	**172.9**	**135.9**
500	2000	88.2	**101.3**	83.1	87.2	**101.3**	81.4	99.3	111.5	92.1	87.2	**101.3**	81.3	**87.1**	**101.3**	**81.1**
800	1000	**248.1**	**400.0**	**300.2**	248.2	**400.0**	**300.2**	252.4	**400.0**	304.6	**248.1**	**400.0**	**300.2**	248.3	**400.0**	**300.2**
800	2000	181.2	**273.4**	186.6	181.5	**273.4**	186.2	193.6	283.1	205.4	**179.1**	**273.4**	184.9	179.3	**273.4**	**184.8**
800	5000	**134.1**	**115.0**	100.3	**134.1**	115.1	98.1	145.8	131.3	110.5	**134.1**	**115.0**	96.9	**134.1**	**115.0**	**96.5**
1000	1000	**333.8**	**500.0**	**400.8**	**333.8**	**500.0**	**400.8**	**333.8**	**500.0**	**400.8**	**333.8**	**500.0**	**400.8**	**333.8**	**500.0**	**400.8**
1000	5000	**169.0**	**168.1**	149.5	**169.0**	**168.1**	143.8	191.8	189.8	162.0	**169.0**	**168.1**	**142.2**	**169.0**	**168.1**	142.7
1000	10 000	**167.0**	105.7	132.9	**167.0**	107.1	90.1	172.1	113.6	97.8	**167.0**	102.4	90.1	**167.0**	**99.9**	**86.2**
Avg:		106.0	137.9	111.1	106.0	138.0	108.2	110.5	142.2	113.3	**105.8**	137.7	107.9	**105.8**	**137.6**	**107.7**

Table 21
Execution times (in seconds) of CMSA, MS-ILS, ILP, MH and time to reach best solution of ILP-PD for UDG graph instances with an uniform capacity of 2, 5, and average degree (α) for every node.

$\lvert V\rvert$	Range	ILP-PD[a]			CMSA			MS-ILS			ILP			MH		
		2	5	α	2	5	α	2	5	α	2	5	α	2	5	α
50	150	0.1	0.1	0.1	1000.0	1000.0	1000.0	**0.0**	0.1	0.1	**0.0**	**0.0**	**0.0**	0.1	0.1	0.1
50	200	0.1	0.1	0.1	1000.0	1000.0	1000.0	**0.0**	0.1	0.1	**0.0**	**0.0**	**0.0**	**0.0**	0.2	0.2
100	150	0.1	0.3	0.3	1000.0	1000.0	1000.0	**0.0**	0.5	0.5	0.1	**0.1**	**0.1**	**0.0**	0.6	0.6
100	200	0.2	0.4	0.4	1000.0	1000.0	1000.0	**0.0**	0.3	0.5	**0.0**	**0.2**	**0.1**	**0.0**	0.3	0.5
250	150	0.8	3.5	4.5	1000.0	1000.0	1000.0	**0.0**	**0.0**	**2.3**	0.2	1.1	4.0	**0.0**	**0.0**	5.6
250	200	0.9	3.5	6.3	1000.0	1000.0	1000.0	**0.0**	**0.0**	**2.5**	0.3	0.9	4.5	**0.0**	**0.0**	6.2
500	150	3.1	25.7	164.1	1000.0	1000.0	1000.0	**0.0**	**0.0**	**7.2**	1.4	5.9	62.6	**0.0**	**0.0**	45.0
500	200	4.4	50.7	90.9	1000.0	1000.0	1000.0	**0.0**	**0.0**	**8.1**	1.8	8.3	66.7	**0.0**	**0.0**	33.7
800	150	8.0	215.9	906.2	1000.0	1000.0	1000.0	**0.0**	**0.0**	**15.8**	4.2	25.9	639.4	**0.0**	**0.0**	359.3
800	200	11.7	387.9	733.4	1000.0	1000.0	1000.0	**0.0**	**0.0**	**19.6**	8.6	17.4	630.0	**0.0**	**0.0**	267.9
1000	150	13.9	459.6	942.4	1000.0	1000.0	1000.0	**0.0**	**0.0**	**23.1**	6.5	52.7	947.7	**0.0**	**0.0**	741.8
1000	200	19.1	787.6	953.4	1000.0	1000.0	1000.0	**0.0**	**0.0**	**30.1**	19.0	43.1	928.5	**0.0**	**0.0**	445.0
Avg:		5.2	161.3	316.8	1000.0	1000.0	1000.0	**0.0**	**0.1**	**9.2**	3.5	13.0	273.6	**0.0**	**0.1**	158.8

[a]Time to reach best solution.

Table 22
Execution times (in seconds) of CMSA, MS-ILS, ILP, MH and time to reach best solution of ILP-PD for general graph instances with an uniform capacity of 2, 5, and average degree (α) for every node.

$\lvert V\rvert$	$\lvert E\rvert$	ILP-PD[a]			CMSA			MS-ILS			ILP			MH		
		2	5	α	2	5	α	2	5	α	2	5	α	2	5	α
50	100	0.1	0.1	0.2	1000.0	1000.0	1000.0	**0.0**	0.2	0.2	**0.0**	**0.0**	**0.0**	**0.0**	0.2	0.2
50	250	0.1	0.2	0.5	1000.0	1000.0	1000.0	**0.0**	**0.0**	0.3	**0.0**	**0.0**	**0.1**	**0.0**	**0.0**	0.4
50	500	0.1	0.2	0.9	1000.0	1000.0	1000.0	**0.0**	**0.0**	**0.2**	**0.0**	**0.0**	0.4	**0.0**	**0.0**	0.6
100	100	0.2	0.1	0.2	1000.0	1000.0	1000.0	**0.0**	0.2	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.2	**0.0**
100	250	0.1	0.9	1.0	1000.0	1000.0	1000.0	**0.0**	0.7	0.7	**0.0**	**0.3**	**0.3**	**0.0**	0.9	1.0
100	500	0.1	0.6	5.5	1000.0	1000.0	1000.0	**0.0**	**0.0**	**1.0**	**0.0**	0.2	4.1	**0.0**	**0.0**	3.8
250	250	1.7	0.2	2.1	1000.0	1000.0	1000.0	**0.0**	0.4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.4	**0.0**
250	500	0.4	9.7	30.6	1000.0	1000.0	1000.0	**0.0**	**2.3**	**2.3**	0.1	9.2	20.6	**0.0**	12.5	24.7
250	1000	0.6	15.7	999.8	1000.0	1000.0	1000.0	**0.0**	**4.1**	**3.8**	0.1	4.4	1000.0	**0.0**	7.7	1000.0
500	500	10.8	0.9	15.8	1000.0	1000.0	1000.0	**0.0**	3.6	**0.0**	**0.0**	**0.1**	**0.0**	**0.0**	3.6	**0.0**
500	1000	1.7	821.4	966.2	1000.0	1000.0	1000.0	**0.0**	**7.0**	**6.8**	0.6	841.5	1000.0	**0.0**	697.0	978.0
500	2000	1.7	254.1	1000.0	1000.0	1000.0	1000.0	**0.0**	**12.1**	**10.9**	0.7	57.9	1000.0	**0.0**	38.5	1000.0
800	1000	2.7	5.8	3.6	1000.0	1000.0	1000.0	**0.2**	16.5	**0.2**	0.4	**5.6**	0.4	**0.2**	20.0	**0.2**
800	2000	3.4	1000.0	1001.0	1000.0	1000.0	1000.0	**0.0**	**15.4**	**15.5**	1.5	1000.0	1000.0	**0.0**	1000.0	1000.0
800	5000	4.6	69.3	993.6	1000.0	1000.0	1000.0	**0.0**	40.6	**31.0**	1.7	**13.8**	1000.1	**0.0**	54.8	1000.0
1000	1000	145.8	3.1	131.7	1000.0	1000.0	1000.0	**0.0**	10.3	**0.0**	0.1	**0.2**	0.1	**0.0**	10.3	**0.0**
1000	5000	5.7	205.6	951.3	1000.0	1000.0	1000.0	**0.0**	45.7	**35.9**	3.3	**31.5**	1000.0	**0.0**	78.6	1000.0
1000	10 000	12.6	149.7	993.9	1000.0	1000.0	1000.0	**0.0**	103.0	**55.4**	2.9	**27.9**	1000.1	**0.0**	131.8	1000.1
Avg:		10.7	141.0	394.3	1000.0	1000.0	1000.0	**0.0**	**14.6**	**9.1**	0.6	110.7	390.3	**0.0**	114.2	389.4

[a]Time to reach best solution.

Table 23
Execution times (in seconds) of CMSA, MS-ILS, ILP, MH and time to reach best solution of ILP-PD for UDG graph instances where capacity of each node vary randomly over $\{2,5\}$, $\{\frac{\alpha}{5},\frac{\alpha}{2}\}$ and $\{1,\ldots,\alpha\}$.

$\lvert V\rvert$	Range	ILP-PD[a]			CMSA			MS-ILS			ILP			MH		
		$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$
50	150	0.1	0.1	0.1	1000.0	1000.0	1000.0	0.2	0.1	0.2	**0.0**	**0.0**	**0.0**	0.2	0.1	0.2
50	200	0.1	0.1	0.1	1000.0	1000.0	1000.0	0.2	0.2	0.2	**0.0**	**0.0**	**0.0**	0.2	0.2	0.2
100	150	1.1	0.2	0.3	1000.0	1000.0	1000.0	**0.6**	0.6	0.6	**0.6**	**0.1**	**0.1**	1.1	0.7	0.7
100	200	0.4	0.4	1.5	1000.0	1000.0	1000.0	0.6	0.6	0.5	**0.1**	**0.1**	**0.4**	0.6	0.7	0.9
250	150	2.6	4.4	38.0	1000.0	1000.0	1000.0	**0.5**	**1.0**	**2.4**	0.7	2.7	12.9	**0.5**	2.6	14.2
250	200	2.3	26.3	58.2	1000.0	1000.0	1000.0	**0.0**	**0.3**	**2.6**	0.6	3.8	15.2	**0.0**	0.5	18.1
500	150	11.7	661.5	940.6	1000.0	1000.0	1000.0	**0.0**	**0.1**	**7.4**	3.9	27.6	405.4	**0.0**	**0.1**	127.1
500	200	15.2	602.3	926.1	1000.0	1000.0	1000.0	**0.0**	**0.0**	**8.5**	5.3	54.1	397.6	**0.0**	**0.0**	54.1
800	150	55.4	936.5	973.2	1000.0	1000.0	1000.0	**0.0**	**6.9**	**16.5**	16.2	300.2	954.7	**0.0**	10.9	567.7
800	200	71.1	1000.0	997.4	1000.0	1000.0	1000.0	**0.0**	**0.1**	**20.3**	13.7	698.3	912.3	**0.0**	**0.1**	429.2
1000	150	105.1	1000.0	1000.0	1000.0	1000.0	1000.0	**0.0**	**1.0**	**24.6**	20.5	846.5	1000.2	**0.0**	**1.0**	1000.2
1000	200	119.0	1000.0	1000.0	1000.0	1000.0	1000.0	**0.0**	**0.3**	**31.2**	23.6	1000.3	1000.3	0.1	0.4	550.7
Avg:		32.0	436.0	494.6	1000.0	1000.0	1000.0	**0.2**	**0.9**	**9.6**	7.1	244.5	391.6	**0.2**	1.4	230.3

[a]Time to reach best solution.

Table 24
Execution times (in seconds) of CMSA, MS-ILS, ILP, MH and time to reach best solution of ILP-PD for general graph instances where capacity of each node vary randomly over $\{2,5\}$, $\{\frac{\alpha}{5},\frac{\alpha}{2}\}$ and $\{1,\ldots,\alpha\}$.

$\lvert V\rvert$	$\lvert E\rvert$	ILP-PD[a]			CMSA			MS-ILS			ILP			MH		
		$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$	$\{2,5\}$	$\{\frac{\alpha}{5},\frac{\alpha}{2}\}$	$\{1,\ldots,\alpha\}$
50	100	0.1	0.1	0.1	1000.0	1000.0	1000.0	0.2	0.2	0.2	**0.0**	**0.0**	**0.0**	0.2	0.2	0.2
50	250	0.1	0.1	0.4	1000.0	1000.0	1000.0	**0.0**	**0.0**	0.3	**0.0**	**0.0**	**0.1**	**0.0**	**0.0**	0.4
50	500	0.1	0.4	1.2	1000.0	1000.0	1000.0	**0.0**	**0.0**	**0.2**	**0.0**	0.1	0.4	**0.0**	**0.0**	0.6
100	100	0.1	0.1	0.1	1000.0	1000.0	1000.0	0.2	**0.0**	0.3	**0.0**	**0.0**	**0.0**	0.2	**0.0**	0.3
100	250	0.7	0.1	0.7	1000.0	1000.0	1000.0	0.7	0.3	0.8	**0.2**	**0.0**	**0.2**	0.8	0.4	0.9
100	500	0.4	0.4	5.7	1000.0	1000.0	1000.0	0.5	0.6	**1.0**	**0.1**	**0.1**	2.4	0.5	0.6	3.4
250	250	0.7	0.2	0.3	1000.0	1000.0	1000.0	0.6	**0.0**	1.4	**0.0**	**0.0**	**0.0**	0.6	**0.0**	1.4
250	500	11.9	0.4	3.4	1000.0	1000.0	1000.0	**2.3**	3.0	2.5	**2.3**	**0.1**	**1.4**	4.6	3.1	3.9
250	1000	177.1	1.0	948.8	1000.0	1000.0	1000.0	**4.5**	5.0	**4.1**	28.2	**0.5**	683.1	32.9	5.5	740.2
500	500	4.5	0.8	0.9	1000.0	1000.0	1000.0	0.8	**0.0**	3.9	**0.1**	**0.0**	**0.0**	0.8	**0.0**	3.9
500	1000	702.0	1.4	51.8	1000.0	1000.0	1000.0	**6.2**	8.4	**7.0**	381.9	**0.3**	23.9	345.6	8.7	34.3
500	2000	725.3	5.7	1000.0	1000.0	1000.0	1000.0	**13.2**	14.7	**11.2**	818.1	**2.5**	1000.0	771.8	16.9	1000.0
800	1000	23.1	2.1	2.1	1000.0	1000.0	1000.0	10.9	**0.3**	9.4	**9.2**	**0.3**	**0.2**	16.1	0.4	9.5
800	2000	1000.0	3.1	1000.0	1000.0	1000.0	1000.0	**15.4**	26.6	**17.2**	1000.0	**0.8**	1000.0	1000.0	27.4	1000.0
800	5000	34.9	74.3	985.7	1000.0	1000.0	1000.0	46.9	44.0	**33.8**	**9.2**	**27.9**	1000.0	54.0	69.8	1000.0
1000	1000	31.0	3.0	3.6	1000.0	1000.0	1000.0	9.0	**0.0**	12.2	**0.2**	0.1	**0.1**	9.0	**0.0**	12.3
1000	5000	123.7	131.6	949.9	1000.0	1000.0	1000.0	47.9	49.6	**39.2**	**20.9**	**16.6**	1000.0	66.2	64.7	1000.0
1000	10 000	94.8	841.6	1000.0	1000.0	1000.0	1000.0	121.6	**82.8**	**60.3**	**16.3**	1000.1	1000.1	133.7	1000.1	1000.1
Avg:		162.8	59.2	330.8	1000.0	1000.0	1000.0	**15.6**	**13.1**	**11.4**	127.0	58.3	317.3	135.4	66.5	322.9

[a]Time to reach best solution.

6.3.3. *Results on variable capacity instances*

In this subsection, we will discuss the computational results of various approaches on variable capacity UDGs as well as general graphs.

Performance on UDGs: Table 19 presents the computational results of different approaches on UDG instances with variable capacity. When compared with ILP-PD, both of our approaches ILP and MH have obtained better or equal solutions in all 36 cases. Compared with CMSA, our approaches ILP and MH obtained better or equal solutions in 29 and 31 cases respectively out of 36. The UDG instances with variable capacity $\{2,5\}$ are solved to optimality by almost all five approaches. For the large graphs with capacity $\{\frac{\alpha}{5},\frac{\alpha}{2}\}$ and $\{1,\ldots,\alpha\}$, both ILP-PD and ILP have shown very poor performance and unable to find good solutions. Also, MH has shown slightly poor performance compared with CMSA on these graphs. The reason behind this is the reduced effectiveness of node frequency information provided by MS-ILS. By observing the averages of different approaches at the last row, CMSA performed slightly better than MH.

Performance on General Graphs: Table 20 reports the computational results of different approaches on general graph instances with variable capacity. The table clearly shows that ILP and MH performed better than the other three approaches. ILP has obtained better or equal results than ILP-PD and CMSA on all 54 cases. Likewise MH also obtained better or equal results in 53 cases out of 54 cases. When compared with ILP, MH has obtained better or equal solutions on 51 cases out of 54. Like in case of uniform general graphs, here also for large graphs ILP and MH have shown much better performance. By observing the averages of different approaches at the last row of the table, MH is the best approach.

From these two tables, compared with ILP-PD, CMSA, and MS-ILS, our approaches ILP and MH has obtained 77 and 82 better or equal results respectively out of 90 variable capacity graph instances.

From all the four tables, ILP has obtained 158 and MH has obtained better or equal results in 163 cases out of 180.

6.3.4. *Comparison of execution times*

Tables 21 to 24 report the execution times of various approaches, viz. CMSA, MS-ILS, ILP and MH and time to reach best solution for ILP-PD. Actually, [16] reports the time to reach the best solutions for ILP-PD and CMSA instead of execution times allowing each approach a maximum permissible time of 1000 s. CMSA was executed on each instance in [16] for maximum permissible time, so we can conclude that execution time of CMSA is 1000 s on each instance. However, ILP-PD can terminate in less than 1000 s in case it finds an optimal solution and hence, its execution time cannot be determined. So we will report time to reach best solution for ILP-PD. Furthermore, the data for ILP-PD

Table 25
Performance comparison summary on various types of instance groups.

Graph	Method	ILP-PD			CMSA			MS-ILS			ILP			MH		
		B	E	W	B	E	W	B	E	W	B	E	W	B	E	W
Uni-UDG	ILP-PD	–	–	–	5	27	4	11	22	3	1	31	4	2	29	5
	CMSA	4	27	5	–	–	–	13	23	0	2	28	6	1	28	7
	MS-ILS	3	22	11	0	23	13	–	–	–	1	23	12	0	23	13
	ILP	4	31	1	6	28	2	12	23	1	–	–	–	2	31	3
	MH	5	29	2	7	28	1	13	23	0	3	31	2	–	–	–
Uni-General	ILP-PD	–	–	–	6	43	5	23	31	0	1	44	9	2	43	9
	CMSA	5	43	6	–	–	–	23	31	0	3	43	8	3	44	7
	MS-ILS	0	31	23	0	31	23	–	–	–	0	31	23	0	31	23
	ILP	9	44	1	8	43	3	23	31	0	–	–	–	4	46	4
	MH	9	43	2	7	44	3	23	31	0	4	46	4	–	–	–
Var-UDG	ILP-PD	–	–	–	0	25	11	12	16	8	0	25	11	0	25	11
	CMSA	11	25	0	–	–	–	16	20	0	7	28	1	5	30	1
	MS-ILS	8	16	12	0	20	16	–	–	–	6	17	13	0	20	16
	ILP	11	25	0	1	28	7	13	17	6	–	–	–	2	27	7
	MH	11	25	0	1	30	5	16	20	0	7	27	2	–	–	–
Var-General	ILP-PD	–	–	–	8	37	9	32	21	1	0	43	11	1	42	11
	CMSA	9	37	8	–	–	–	33	21	0	0	40	14	1	38	15
	MS-ILS	1	21	32	0	21	33	–	–	–	0	21	33	0	21	33
	ILP	11	43	0	14	40	0	33	21	0	–	–	–	3	45	6
	MH	11	42	1	15	38	1	33	21	0	6	45	3	–	–	–

Table 26
Performance comparison summary of ILP and MH with ILP-PD, CMSA and MS-ILS over all 180 instance groups.

Method	ILP-PD			CMSA			MS-ILS			ILP		
	B	E	W	B	E	W	B	E	W	B	E	W
ILP	35	143	2	29	139	12	81	92	7	–	–	–
MH	36	139	5	30	140	10	85	95	0	20	149	11

and CMSA are taken from [16], where these approaches had been executed on a 3.4 GHz Intel Core i7-3770 CPU based system which is different from the system used to execute our approaches. Therefore, exact comparison of execution times are not possible. However, a rough comparison can always be made. Based on the information available in the public domain regarding the performance of these two processors on various benchmark instances, our system is only slightly faster (less than 10%) than the one used in [16]. Even after compensating for this difference in processing speeds, MH is faster than CMSA on almost all instances. Furthermore, if we compare the execution times of ILP and MH with time to reach best solution for ILP-PD, we can safely conclude ILP and MH to be faster than ILP-PD on most instances. MS-ILS being a heuristic approach is fastest among all.

6.3.5. Performance comparison summary

Tables 25 and 26 are summary tables which report the performance of various approaches in terms of number of instance groups on which an approach mentioned in the first column obtained better (B), same (E) or worse (W) solution with respect to the approach mentioned in first row. Table 25 does this separately for various types of instance groups, whereas Table 26 does this overall instance groups. These tables clearly show the superiority of our approaches over other approaches.

7. Conclusions

In this paper, we have presented a multi-start iterated local search (MS-ILS), an exact approach based on an integer linear programming (ILP) model that utilizes the concept of strong vertices and other improvements to an existing integer linear programming model, and a matheuristic combining the ILP approach with MS-ILS through solution merging for the minimum capacitated dominating set (CAPMDS) problem for undirected graphs. Computational results on standard benchmark instances of the CAPMDS problem show that our approaches outperformed state-of-the-art approaches in their respective categories not only in terms of solution quality, but also in terms of execution time.

Our approaches can be extended to related variants of dominating set problem particularly variants involving directed graphs which are relatively understudied. A key feature of our MS-ILS approach is that same heuristic is used for initial solution generation and local search. This shows that how degree of preference for nodes changes drastically by retaining a fraction of dominating nodes but recomputing the set of dominated nodes corresponding to them. Similar perturbation procedure can be tried for other dominating set variants also. Similarly the concept of strong vertices introduced in the context of ILP can be tried for other dominating set variants too. As a future work, we intend to develop new matheuristic approaches for the CAPMDS and related problems by exploring different ways of combining integer linear programming with problem specific heuristics and metaheuristics.

CRediT authorship contribution statement

Mallikarjun Rao Nakkala: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing - original draft, Visualization. **Alok Singh:** Supervision, Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing - review & editing, Resources. **André Rossi:** Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The first author acknowledges the financial support received from the Council of Scientific and Industrial Research (CSIR), Government of India in the form of a senior research fellowship.

References

[1] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.

[2] K. Erciyes, O. Dagdeviren, D. Cokuslu, D. Ozsoyeller, Graph theoretic clustering algorithms in mobile ad hoc networks and wireless sensor networks, Appl. Comput. Math. 6 (2) (2007) 162–180.

[3] Y.P. Chen, A.L. Liestman, J. Liu, Clustering algorithms for ad hoc wireless networks, in: Y. Xiao, Y. Pan (Eds.), Ad Hoc and Sensor Networks, Nova Science Publisher, 2006, pp. 145–164.

[4] T. Firdaus, M. Hasan, A survey on clustering algorithms for energy efficiency in wireless sensor network, in: 3rd International Conference on Computing for Sustainable Global Development, IEEE, 2016, pp. 759–763.

[5] D.J. Lawrie, W.B. Croft, A.L. Rosenberg, Finding topic words for hierarchical summarization, in: SIGIR, ACM, 2001, pp. 349–357.

[6] C. Shen, T. Li, Multi-document summarization via the minimum dominating set, in: COLING, Tsinghua University Press, 2010, pp. 984–992.

[7] J. Bar-Ilan, G. Kortsarz, D. Peleg, How to allocate network centers, J. Algorithms 15 (3) (1993) 385–415.

[8] M. Kao, H. Chen, Approximation algorithms for the capacitated domination problem, 2010, http://arxiv.org/abs/1004.2839, CoRR abs/1004.2839.

[9] F. Kuhn, T. Moscibroda, Distributed approximation of capacitated dominating sets, Theory Comput. Syst. 47 (4) (2010) 811–836.

[10] M. Cygan, M. Pilipczuk, J.O. Wojtaszczyk, Capacitated domination faster than $O(n^2)$, Inform. Process. Lett. 111 (23–24) (2011) 1099–1103.

[11] M. Liedloff, I. Todinca, Y. Villanger, Solving Capacitated Dominating Set by using covering by subsets and maximum matching, Discrete Appl. Math. 168 (2014) 60–68.

[12] A. Becker, Capacitated dominating set on planar graphs, 2016, http://arxiv.org/abs/1604.04664, CoRR abs/1604.04664.

[13] A. Potluri, A. Singh, A greedy heuristic and its variants for minimum capacitated dominating set, Commun. Comput. Inf. Sci. 306 (2012) 28–39.

[14] R. Li, S. Hu, P. Zhao, Y. Zhou, M. Yin, A novel local search algorithm for the minimum capacitated dominating set, J. Oper. Res. Soc. 69 (6) (2018) 849–863.

[15] A. Potluri, A. Singh, Metaheuristic algorithms for computing capacitated dominating set with uniform and variable capacities, Swarm Evol. Comput. 13 (2013) 22–33.

[16] P. Pinacho-Davidson, S. Bouamama, C. Blum, Application of CMSA to the minimum capacitated dominating set problem, in: Proceedings of the Genetic and Evolutionary Computation Conference 2019, GECCO 2019, ACM, 2019, pp. 321–328.

[17] V. Maniezzo, T. Stützle, S. Voß, Matheuristics: Hybridizing Metaheuristics and Mathematical Programming, Springer, New York, NY, USA, 2009.

[18] B.N. Clark, C.J. Colbourn, D.S. Johnson, Unit disk graphs, Discrete Math. 86 (1–3) (1990) 165–177.

[19] R. Wattenhoffer, Distributed dominating set approximation, 2004, http://www.disco.ethz.ch/lectures/ss04/distcomp/lecture/chapter12.pdf.

[20] H.R. Lourenço, O.C. Martin, T. Stutzle, Iterated Local Search, in: International Series in Operations Research and Management Science, Springer, 2003, pp. 321–354.

[21] H.R. Lourenço, O.C. Martin, T. Stützle, Iterated local search: Framework and applications, in: Handbook of Metaheuristics, Springer, 2010, pp. 363–397.

[22] A. Subramanian, M. Battarra, An iterated local search algorithm for the travelling salesman problem with pickups and deliveries, J. Oper. Res. Soc. 64 (3) (2013) 402–409.

[23] P.H.V. Penna, A. Subramanian, L.S. Ochi, An iterated local search heuristic for the heterogeneous fleet vehicle routing problem, J. Heuristics 19 (2) (2013) 201–232.

[24] M.M. Silva, A. Subramanian, L.S. Ochi, An iterated local search heuristic for the split delivery vehicle routing problem, Comput. Oper. Res. 53 (2015) 234–249.

[25] P. Venkatesh, G. Srivastava, A. Singh, A multi-start iterated local search algorithm with variable degree of perturbation for the covering salesman problem, in: Harmony Search and Nature Inspired Optimization Algorithms, Springer, 2019, pp. 279–292.

[26] D.P. Cuervo, P. Goos, K. Sörensen, E. Arráiz, An iterated local search algorithm for the vehicle routing problem with backhauls, European J. Oper. Res. 237 (2) (2014) 454–464.

[27] P. Venkatesh, A. Singh, R. Mallipeddi, A multi-start iterated local search algorithm for the maximum scatter traveling salesman problem, in: IEEE Congress on Evolutionary Computation, CEC 2019, Wellington, New Zealand, June 10-13, 2019, IEEE, 2019, pp. 1390–1397.

[28] M. Mastrogiovanni, The clustering simulation framework : A simple manual, 2007, http://www.michele-mastrogiovanni.net/software/download/README.pdf.

[29] R. Jovanovic, M. Tuba, D. Simian, 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, 2010, pp. 322–326.