An Exact Algorithm for the Minimum Dominating Set Problem

Hua Jiang[*] , **Zhifei Zheng**
Engineering Research Center of Cyberspace & School of Software, Yunnan University, China
huajiang@ynu.edu.cn, zzfreya@mail.ynu.edu.cn

Abstract

The Minimum Dominating Set (MDS) problem is a classic NP-hard combinatorial optimization problem with many practical applications. Solving MDS is extremely challenging in computation. Previous work on exact algorithms mainly focuses on improving the theoretical time complexity and existing practical algorithms for MDS are almost based on heuristic search. In this paper, we propose a novel lower bound and an exact algorithm for MDS. The algorithm implements a branch-and-bound (BnB) approach and employs the new lower bound to reduce search space. Extensive empirical results show that the new lower bound is efficient in reduction of the search space and the new algorithm is effective for the standard instances and real-world instances. To the best of our knowledge, this is the first effective BnB algorithm for MDS.

1 Introduction

Given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Two vertices are neighbors if there is an edge between them. A dominating set D is a subset of V such that every vertex of G is either in D or has at least one neighbor in D. The Minimum Dominating Set (MDS) problem consists in finding a dominating set D with the smallest cardinality for a given graph G.

MDS and its variants find applications in diverse areas, including multi-document summarization [Shen and Li, 2010], influences among individuals in social networks [Dinh *et al.*, 2014], routing in ad hoc wireless networks [Dai and Wu, 2004], graph mining [Chalupa, 2018] and action recognition in computer vision [Yao and Fei-Fei, 2012].

Solving MDS in general graphs is NP-Hard [Karp, 1972; Garey and Johnson, 1979]. On the theoretical side, Rooij and Bodlaender [2011] proved that there is a branch-and-reduce algorithm that solves MDS in $O(1.4969^n)$ time and polynomial space, where n is the number of vertices. Dinur and Steurer [2014] proved that obtaining a $(1 - \varepsilon)\ln(n)$-approximation for MDS is NP-hard for every $\varepsilon > 0$ unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$). Gast et al. [2015] proved that MDS on power law graphs cannot be approximated within an approximation ratio of $\Omega(\ln(n))$. Raz and Safra [1997] gave a lower bound of approximation of $c\ln(n)$ under the assumption of $P \neq NP$. In general, theoretical work for MDS mainly focuses on improving the approximation ratio and the upper bound of time complexity of exact algorithms.

Due to the relevance of MDS to practical applications, there is a lot of work on heuristic algorithms for MDS. Experimental analysis and comparison of early heuristic algorithms can be found in [Sanchis, 2002]. Hybrid genetic and ant-colony optimization algorithms for MDS were proposed in [Hedar and Ismail, 2010; Potluri and Singh, 2011]. Hybrid meta-heuristic algorithms based on evolutionary and memetic approaches were proposed for the Minimum Weight Dominating Set (MWDS) problem in [Chaurasia and Singh, 2015; Lin *et al.*, 2016], which is a generalization of MDS. In recent years, several local search algorithms were proposed for MDS and its variants. Wang et al. [2017; 2018] proposed two local search algorithms based on configuration checking strategy and frequency scoring function for MWDS. Fan et al. [2019] proposed a local search based on tabu and probabilistic walk strategy for MDS in large graphs. Cai et al. [2020] proposed a two-goal local search and inference rules for MDS. Lei and Cai formulated MDS into maximum satisfiability and proposed a specific local search to solve MDS [Lei and Cai, 2020]. Li et al. [2020] proposed a local search for the minimum connected dominating set problem.

Compared to heuristic algorithms, exact algorithms for solving MDS are almost absent in the literature. Although MDS is a subgraph problem in general, solving MDS is quite different from solving other subgraph problems, such as the Maximum Clique (MC) problem. When an algorithm searches on a partial solution for MC, those vertices that are inconsistent with the partial solution can be discarded safely, resulting in a substantial reduction of search space. Whereas, each vertex outside the partial solution for MDS could be a candidate for the final solution. The actual search space for MDS is much bigger than the search space for MC. Therefore, solving MDS exactly is more challenging than solving other subgraph problems like MC in the view of practice.

In this paper, we present an effective exact algorithm for MDS, which implements a branch-and-bound (BnB) approach. To reduce the search space, we propose a novel lower bound for MDS, which is derived from a new notion of 2-hop

[*]Corresponding author

adjacency of vertices of G. Two vertices are 2-hop adjacent if and only if they are neighbors or have common neighbors. We prove that two vertices cannot be dominated by a single vertex if they are not 2-hop adjacent. With this property, we define a notion called 2-hop graph of G and prove that the cardinality of any independent set of the 2-hop graph is a lower bound of MDS in G. We design a sophisticated algorithm to exploit the new lower bound to reduce the number of branches at each search tree node in the BnB algorithm.

Extensive experiments were conducted to evaluate the performance of the new algorithm and the lower bound. The results show that the lower bound can reduce the search space efficiently and the proposed BnB algorithm is effective for the standard instances for MDS and even can solve MDS in real-world large graphs. To the best of our knowledge, this is the first effective BnB algorithm for MDS.

The paper is organized as follows: Section 2 gives basic graph notations and definitions. Section 3 presents the new lower bound. Section 4 gives a BnB algorithm for MDS. Section 5 reports on the empirical results. Section 6 concludes.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph with n vertices and m edges. Two vertices u and v are neighbors or adjacent, if there is an edge between u and v, i.e., $(u, v) \in E$. We use $N_G(v)$ to denote the set of neighbors of a vertex v in G, i.e., $N_G(v) = \{u|(u, v) \in E\}$. $N_G(v)$ is also denoted by $N(v)$ if the context is clear. The degree of a vertex v, denoted by $deg(v)$, is the cardinality of $N_G(v)$. Following previous literature, $N_G[v]$ is defined as $N_G(v) \cup \{v\}$ in this paper. Let S be a subset of V, $N_G[S]$ is defined as $\bigcup_{v \in S} N_G[v]$.

A subset D of V is a dominating set of G if and only if it holds that $N(v) \cap D \neq \emptyset$ for each vertex $v \in V \setminus D$. A dominating set D of the smallest size is the minimum dominating set in G. If a vertex u is adjacent to v, we say that u can dominate v in G, vice versa. Moreover, we give a definition of subset dominating.

Definition 1 (Subset Dominating). *Let $G = (V, E)$, D and U be two subsets of V. We say D can dominate U if it holds that $u \in D$ or $N(u) \cap D \neq \emptyset$ for each $u \in U$.*

A subset of V is an Independent Set (IS) if every pair of vertices in the set is nonadjacent. An IS becomes *conflicting* if it contains adjacent vertices and the adjacent vertices are called *conflicting* vertices. We use $G[S]$ to denote the subgraph of $G = (V, E)$ induced by a subset $S \subseteq V$, i.e., $G[S] = (S, E')$, where $E' = \{(u, v)|u, v \in S, (u, v) \in E\}$.

3 A New Lower Bound for MDS

In this section, we propose a novel lower bound for MDS. The lower bound is based on a notion called 2-hop adjacent. We first give the definitions of 2-hop adjacent and 2-hop graphs, then introduce the new lower bound for MDS.

Definition 2 (2-hop adjacent). *Given a graph $G = (V, E)$, two vertices u and v are 2-hop adjacent in G if and only if u and v are adjacent or they have common neighbors, i.e., $(u, v) \in E$ or $N_G(u) \cap N_G(v) \neq \emptyset$.*

We call u and v 2-hop adjacent because the distance between u and v is smaller than or equal to 2 (hops) in G.

Definition 3 (2-hop graphs). *Given a graph $G = (V, E)$, we define a new graph $G^2 = (V, E^2)$ for G, where $E^2 = \{(u, v)|(u, v) \in E$ or $N_G(u) \cap N_G(v) \neq \emptyset\}$, i.e., two vertices u and v are adjacent in G^2 if and only if u and v are 2-hop adjacent in G. G^2 is called the 2-hop graph of G.*

We use $G^2[U]$ to denote the subgraph of G^2 induced by U and $G[U]^2$ to denote the 2-hop graph of the induced subgraph $G[U]$. With Definition 2, we introduce the following lemma.

Lemma 1. *Given a graph $G = (V, E)$, two vertices u and v cannot be dominated by a single vertex if they are not 2-hop adjacent in G.*

Proof. If u and v are not 2-hop adjacent in G, then u and v are not adjacent and cannot be dominated by each other. If there is a vertex w that can dominate both u and v, w must be adjacent to u and v simultaneously, which is inconsistent with that u and v are not 2-hop adjacent in G. Thus, u and v cannot be dominated by a single vertex. □

With the definition of 2-hop graphs and Lemma 1, we have the following lemma to derive a lower bound for MDS.

Lemma 2. *Given a graph $G = (V, E)$, let G^2 be the 2-hop graph of G. If S is an IS of G^2, then the cardinality of S is a lower bound of the minimum dominating set of G.*

Proof. Since S is an IS in G^2, any two vertices u and v in S are not 2-hop adjacent in G. According to Lemma 1, u and v cannot be dominated by a single vertex. Thus, it needs $|S|$ different vertices to dominate vertices in S. Note that S is a subset of V, thus the size of any dominating set to dominate all the vertices of G is not smaller than $|S|$. Therefore, $|S|$ is a lower bound of the minimum dominating set of G. □

We call the lower bound in Lemma 2 the *IS-based Lower Bound* (ISLB). Figure 1 illustrates the new lower bound.

Example 1. *In Figure 1, the right subfigure* b *is the corresponding 2-hop graph G^2 of G in the left subfigure* a*. The dashed lines denote the new added edges in G^2. The subset $\{v_1, v_2, v_9\}$ is an IS of G^2. According to Lemma 2, its cardinality of 3 is a lower bound of MDS of G in subfigure* a*. In fact, 3 is a tight lower bound for MDS of G, because $\{v_2, v_3, v_7\}$ is a minimum dominating set of G.*

Further, we give the following lemma, which is a generalization of Lemma 2. It gives a lower bound of dominating sets that can dominate a given subset of vertices of G.

Lemma 3. *Given a graph $G = (V, E)$ and its G^2. Let U be a subset of V and S be an IS in $G^2[U]$. For any subset $D \subseteq V$ that can dominate U in G, it holds that $|D| \geq |S|$.*

Proof. Since S is an IS of $G^2[U]$, then any two vertices u and v in S are not 2-hop adjacent in G. According to Lemma 1, there is no single vertex that can dominate both u and v. Therefore, it needs $|S|$ different vertices to dominate vertices in S. Since $S \subseteq U$, $|S|$ is a lower bound of the cardinality of any subset D that can dominate U, i.e., $|D| \geq |S|$. □

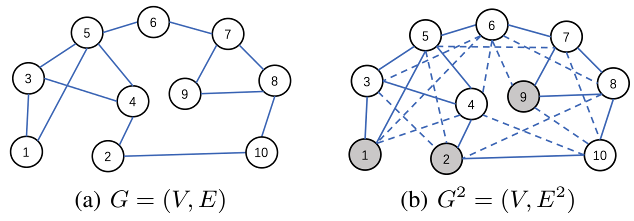

Figure 1: An example for IS-based lower bound for MDS.

Note that the computation of the optimal ISLB for a graph G is equivalent to solving the Maximum Independent Set (MIS) problem in its G^2, which is NP-hard [Garey and Johnson, 1979]. In the next section, we will present a sophisticated algorithm that exploits ISLB to reduce the number of branches in BnB algorithms rather than to compute a lower bound for the subproblem of each search tree node.

4 An Exact Algorithm for MDS

We present an exact algorithm for MDS, called EMOS (Exact Minimum dOminating Set), which is depicted in Algorithm 1. EMOS first calls a preprocessing procedure to compute an initial solution D_0 and a fixed partial solution D_f, and to reduce the input graph G to G'. Then, the algorithm calls a BnB search procedure to search for an optimal solution in the reduced graph G' with D_f as the starting partial solution and D_0 as the incumbent solution D^*. We first describe the preprocessing procedure, then the BnB search algorithm.

4.1 The Preprocessing Procedure

The preprocessing procedure performs three tasks: compute an initial solution D_0, reduce the input graph G and obtain a fixed partial solution D_f.

To compute an initial solution D_0 for a graph $G = (V, E)$, we adopt the following heuristic: Let D_0 be an empty set at the beginning, select a vertex v that can dominate the most number of undominated vertices, mark v and its neighbors as dominated and insert it into D_0. The selection step executes iteratively till all the vertices of V have been dominated. Then, D_0 is an initial solution. Using a maximum heap, the process can be done in $O((|E| + |V|) \cdot \log |V|)$ time.

Reduction has been proven to be an effective strategy for solving subgraph problems in real-world graphs [Jiang *et al.*, 2017; Gao *et al.*, 2018; Zhou *et al.*, 2021]. To reduce the input graph G, we employ the reduction rule for MDS proposed in [Alber *et al.*, 2004]. The rule identifies vertices that must be in an optimal solution of MDS by exploiting local structures of graphs. We implement the rule in our preprocessing procedure, which partitions the set $N(v)$ of vertex v into three different subsets $N_1(v)$, $N_2(v)$ and $N_3(v)$.

- $N_1(v)$, the set of neighbors of v that have at least one neighbor not in $N[v]$.
- $N_2(v)$, the set of neighbors of v that are not in $N_1(v)$ but have at least one neighbor in $N_1(v)$.
- $N_3(v)$, the set of neighbors of v not in $N_1(v) \cup N_2(v)$.

Algorithm 1 EMOS(G), an exact algorithm for MDS

```
Input: A graph $G = (V, E)$
Output: a minimum dominating set $D^*$ in $G$.
1: $(D_0, D_f, G' = (V', E')) \leftarrow$ Preprocessing($G$);
2: Let $U = V' \setminus N_{G'}[D_f]$;
3: return BnBSearch($G'$, $D_f$, $U$, $D_0$);
```

For a vertex $v \in V$, if $N_3(v)$ is not empty, the rule removes those vertices in $N_2(v)$ and $N_3(v)$ from G and adds a new vertex v' to G as a neighbor of v. Alber et al. [2004] proved that the optimality of MDS is preserved and v must be in an optimal solution after the transformation. We take v as a fixed vertex and insert it into the fixed partial solution D_f.

After the rule is applied to every vertex in G, we obtain a fixed partial solution D_f and the reduced graph G'. We use D_f as the starting point of the following search in G'. Time complexity of the reduction is $O(|V|^3)$.

4.2 The Branch-and-Bound Search Procedure

Algorithm 2 describes the BnB search procedure for MDS, *BnBSearch(G, D, U, D*)*, where D is the partial solution and U is the set of vertices that haven't been dominated by D. The procedure explores the search space to find a minimum dominating set of containing D and size smaller than the incumbent D^* to dominate all the vertices in G.

Theoretically, every vertex outside the partial solution D could be a candidate of the final solution, resulting in a huge search space for solving MDS. To prune the search space, Algorithm 2 first calls a function *ReduceBranches* to identify a minimal set B of branching vertices. Let $b_1 > b_2 > \cdots > b_{|B|}$ be the decreasing ordering of vertices of B *w.r.t.* the number of undominated neighbors of each b_i. The algorithm branches only on each $b_i \in B$ from $i = 1$ to $|B|$. Once it branches on b_i, b_i is marked as *branched* status. When all the vertices in B have branched, they resume to *unbranched* status before the algorithm backtracks to the upper level of the search tree. Obviously, the smaller the branching set B, the less search space needs to be explored.

We divide the set $V \setminus D$ into following four subsets, according to the branching and dominating status of vertices.

- $\mathbb{S}_1$: the set of unbranched and undominated vertices
- $\mathbb{S}_2$: the set of unbranched but dominated vertices
- $\mathbb{S}_3$: the set of branched but undominated vertices
- $\mathbb{S}_4$: the set of branched and dominated vertices

At the current search tree node, all the vertices in $\mathbb{S}_1$ and $\mathbb{S}_2$ could join in the partial solution D. The vertices in $\mathbb{S}_3$ cannot join in D but must be dominated by the vertices in $\mathbb{S}_1 \cup \mathbb{S}_2$. The vertices in $\mathbb{S}_4$ can be omitted completely. Thus, $C = \mathbb{S}_1 \cup \mathbb{S}_2$ is the entire candidate set of branching vertices.

To reduce the number of branches, we adopt the BnB search approach proposed in [Jiang *et al.*, 2016]. The function ReduceBranches is designed to partition the union $\mathbb{S}_1 \cup \mathbb{S}_2 \cup \mathbb{S}_3$ into B and P. Let $U_P \subseteq P$ be the set of undominated vertices in P. The partition of B and P satisfies that the cardinality of the minimum dominating set in $G[P]$ for dominating the subset U_P is greater than or equal to $|D^*| - |D|$, which

Algorithm 2 BnBSearch(G, D, U, D^*)

```
Input: A graph $G$=($V$,$E$), a partial solution $D$, the set of undominated vertices $U$ and the best solution $D^*$ found so far
Output: the best solution $D^*$ in $G$
1: if $U$ is empty then
2:     return $D$;
3: end if
4: $B$ ←ReduceBranches($G$, $D$, $U$, $D^*$);
5: if $B$ is empty then
6:     return $D^*$;
7: end if
8: sort vertices in $B$ as $b_1 > b_2 >...> b_{|B|}$ in decreasing ordering w.r.t. the number of undominated neighbors.
9: for $i = 1$ to $|B|$ do
10:     Let $U' = U \setminus N[b_i]$;
11:     mark $b_i$ as branched;
12:     $D' \leftarrow$ BnBSearch($G$, $D \cup \{b_i\}$, $U'$, $D^*$);
13:     if $|D'| < |D^*|$ then
14:         $D^* \leftarrow D'$;
15:     end if
16: end for
17: mark every $b_i \in B$ as unbranched;
18: return $D^*$;
```

implies that it is impossible to derive a dominating set of containing D and size smaller than D^* to dominate the whole G using vertices in P solely. Consequently, $B = C \setminus P$ is the set of branching vertices of the current search tree node.

The function ReduceBranches is depicted in Algorithm 3, which employs the new lower bound ISLB to implement the partition. Let $U = \mathbb{S}_1 \cup \mathbb{S}_3$ be the set of undominated vertices. At the beginning, P is an empty set. The algorithm constructs the partition in two steps. At the first step (line 3-22), the algorithm tries to identify a subset $P \subseteq U$ such that the lower bound of MDS of $G[P]$ is greater than or equal to $|D^*|-|D|$. If such a set P cannot be identified, the whole candidate set C is returned as the branching set (line 24). In this case, the number of branches of the current search tree node has not been reduced. Otherwise, algorithm 3 conducts the second step (line 26-36) to further enlarge P by inserting more vertices in $\mathbb{S}_2$ into P, subject to the lower bound of dominating sets for dominating all the undominated vertices in $G[P]$ is still not smaller than $|D^*|-|D|$. Finally, Algorithm 3 returns $C \setminus P$ as the branching set B.

The First Step to Identify P

The essence of the first step of Algorithm 3 is to construct a subgraph $G[P]$ ($P \subseteq U$) with a MDS of size greater than or equal to $|D^*|-|D|$. According to Lemma 2, the cardinality of any IS of $G[P]^2$ is a lower bound of MDS of $G[P]$. To obtain an IS of $G[P]^2$ as big as possible, we maintain a set Π of at most k ISs of $G[P]^2$ during the contruction. Let $u_1 < u_2 < \cdots < u_{|U|}$ be the natural ordering of vertices in U. We try to insert each $u_i \in U$ into P from $i = |U|$ to 1 and construct ISs in Π. A vertex u_i is allowed to join into many ISs of Π.

To obtain an IS of $G[P]^2$ of size greater than or equal to $|D^*|-|D|$, we define a score for each vertex u and nonempty IS $S_j \in \Pi$, denoted by $\delta(S_j, u)$, to measure the impact on the

Algorithm 3 ReduceBranches(G, D, U, D^*), algorithm to identify a branching set B

```
Input: A graph $G$=($V$,$E$), a partial solution $D$, the set of undominated vertices $U$ and the best solution $D^*$ found so far
Output: A branching set $B$
1: Let $P$=$\emptyset$, $C$=$\mathbb{S}_1 \cup \mathbb{S}_2$, $U$=$\mathbb{S}_1 \cup \mathbb{S}_3$=$\{u_1, u_2, \ldots, u_{|U|}\}$;
2: Let $\Pi = \{S_1, S_2, \ldots, S_k\}$ be the set of $k$ ISs of $G[P]^2$, each $S_i$ is initialized to $\emptyset$ at the beginning;
3: for $i = |U|$ to 1 do
4:     Let $totalScore = 0$;
5:     for each nonempty IS $S_j \in \Pi$ do
6:         $totalScore \leftarrow totalScore + \delta(S_j, u_i)$;
7:     end for
8:     if $totalScore \geq 0$ then
9:         $P \leftarrow P \cup \{u_i\}$;
10:        for each nonempty $S_j \in \Pi$ do
11:            if $\delta(S_j, u_i) > 0$ then
12:                insert $u_i$ into $S_j$;
13:            end if
14:            if $\delta(S_j, u_i) < 0$ then
15:                remove $|N(u_i) \cap S_j|$-1 conf. vertices from $S_j$
16:            end if
17:        end for
18:        if $u_i$ hasn't been inserted into any nonempty IS then
19:            insert $u_i$ into the first empty IS $S_i$;
20:        end if
21:    end if
22: end for
23: Let $lb = \max\{|S_j| \mid S_j \in \Pi\}$;
24: if $lb < |D^*| - |D|$ then return $C$;
25: Let $\mathbb{S}_2 = \{d_1, d_2, \ldots, d_{|\mathbb{S}_2|}\}$;
26: for $i = |\mathbb{S}_2|$ to 1 do
27:     Let $\Pi' \leftarrow \Pi$, $P \leftarrow P \cup \{d_i\}$;
28:     for each nonempty $S_j \in \Pi$ do
29:         if $\delta(S_j, d_i) < 0$ then
30:             remove $|N(d_i) \cap S_j|$-1 conf. vertices from $S_j$
31:         end if
32:     end for
33:     if $\max\{|S_j| \mid S_j \in \Pi\} < |D^*| - |D|$ then
34:         $\Pi \leftarrow \Pi'$, $P \leftarrow P \setminus \{d_i\}$;
35:     end if
36: end for
37: return $C \setminus P$ ;
```

size of S_j by the insertion of u into P.

Let $N_1 = N_G(u) \cap S_j$ be the set of neighbors of u in S_j and $N_2 = N_G[u] \cap N_G[S_j] \cap P \cap C$ be the set of common and unbranched neighbors of u and vertices of S_j in P. If N_1 is empty, S_j is still an IS of $G[P]$ after inserting u into P. If u is undominated and N_2 is empty, u can be inserted into S_j. The score $\delta(S_j, u)$ is defined in Equation 1.

If u is *unbranched* ($u \in \mathbb{S}_1$), it could be selected to dominate other vertices in the following search. Thus, u is used to determine the 2-hop adjacency of vertices in $G[P]$, i.e., any two neighbors of u in P are adjacent in $G[P]^2$ if u is inserted into P. If N_1 is not empty, the neighbors of u in S_j will become conflicting after inserting u into P. To keep S_j as an

IS, $|N_1| - 1$ neighbors of u must be removed from S_j. Thus, we define the score as $1 - |N_1|$ because the size of S_j is decreased by $|N_1| - 1$. Note that u is undominated thus u can join into S_j if N_1 and N_2 are empty, the score is defined as 1. The score is defined as 0 if N_1 is empty but N_2 is not empty.

If u is *branched* ($u \in \mathbb{S}_3$), the 2-hop adjacency between vertices in $G[P]$ will not change after the insertion of u into P, because u will not be selected as a candidate to dominate other vertices in the following search. In this case, we only need to consider whether u can join into S_j. If N_2 is empty, u can join into S_j and the size of S_j is increased by 1, the score is defined as 1; Otherwise, the score is defined as 0.

$$\delta(S_j, u) = \begin{cases} 1 & u \in \mathbb{S}_1 \wedge N_1 = \emptyset \wedge N_2 = \emptyset \\ 0 & u \in \mathbb{S}_1 \wedge N_1 = \emptyset \wedge N_2 \neq \emptyset \\ 1 - |N_1| & u \in \mathbb{S}_1 \cup \mathbb{S}_2 \wedge N_1 \neq \emptyset \\ 0 & u \in \mathbb{S}_2 \wedge N_1 = \emptyset \\ 0 & u \in \mathbb{S}_3 \wedge N_2 \neq \emptyset \\ 1 & u \in \mathbb{S}_3 \wedge N_2 = \emptyset \end{cases} \qquad (1)$$

If the total score of u, defined as the sum of $\delta(S_j, u)$ for every non-empty $S_j \in \Pi$, is greater than or equal to 0, Algorithm 3 inserts u into P, because it is expected to bring positive impacts on the maximum cardinality of ISs in Π. It inserts u into the ISs of positive scores and removes the conflicting vertices from ISs of negative scores. If u couldn't be inserted into any nonempty IS and there exists empty ISs, u is inserted into the first empty IS. If the total score is smaller than 0, Algorithm 3 does not insert u into P at the first step.

Let lb be the maximum cardinality of ISs in Π after the first step. If $lb \geq |D^*| - |D|$, it guarantees that using vertices in P solely cannot obtain a dominating set of containing D and size smaller than D^* to dominate G. In this case, Algorithm 3 takes the second step to insert more vertices of $\mathbb{S}_2$ into P to further reduce the number of branches.

The Second Step to Enlarge P

The vertices in $\mathbb{S}_2$ (dominated but unbranched) can join in P but cannot be inserted into ISs of Π to estimate the lower bound. Let U_p be the set of undominated vertices that are inserted into P in the first step. At the second step, we compute the lower bound of dominating sets for dominating the subset U_p. Since the ISs in Π only contain vertices in U_p, according to Lemma 3, the cardinality of the maximum IS in Π is the lower bound of dominating sets for dominating U_p. Therefore, to guarantee that a dominating set of containing D and size smaller than D^* couldn't be derived with vertices in P solely, we only insert those vertices that satisfy that the cardinality of the maximum IS in Π is still greater than or equal to $|D^*| - |D|$ after the insertion.

Finally, Algorithm 3 returns the set of candidate vertices that have not been inserted into P as the branching set B, i.e., $C \setminus P$. We use Figure 1 to illustrate How Algorithm 3 identifies a minimal branching set B.

Example 2. *Suppose k is 4, the partial solution D=$\{v_4\}$, the size of D^* is 4 and only v_4 is* branched*, then we have lb=$|D^*|$-$|D|$=3, $\mathbb{S}_1$=$\{v_1, v_6, v_7, v_8, v_9, v_{10}\}$, $\mathbb{S}_2$=$\{v_2, v_3, v_5\}$, $\mathbb{S}_3$=$\emptyset$, $\mathbb{S}_4$=$\emptyset$ and U=$\mathbb{S}_1 \cup \mathbb{S}_3$=$\{v_1, v_6, \ldots, v_{10}\}$. At the first step, Algorithm 3 constructs P=$\{v_{10}, v_9, v_7, v_6, v_1\}$ and obtains a Π=$\{\{v_{10}, v_9, v_1\}, \{v_7, v_1\}, \{v_6, v_1\}, \emptyset\}$. It does not insert v_8 into P, because its score is negative. The computed lower bound of MDS for $G[P]$ is 3, which is greater than or equal to lb. At the second step, v_5, v_3 and v_2 are inserted into P because they do not change the 2-hop adjacency of vertices in the maximum IS $\{v_{10}, v_9, v_1\}$. Finally, Algorithm 3 derives a minimal branching set $B = \{v_8\}$. The number of branches is reduced from 9 to 1.*

According to our experiments, k is fixed to 16 in Algorithm 3. Let $\Gamma(G)$ be the maximum degree of vertices in G. The time complexity of determining the 2-hop adjacency of two vertices is $O(\Gamma(G))$. The time complexity of the computation of the total score for each u is $O(|C| \cdot \Gamma(G))$. Thus, the total time complexity of Algorithm 3 is $O(|C|^2 \cdot \Gamma(G))$.

5 Experimental Evaluations

We implemented the new algorithm EMOS[1] in C and complied it with GNU gcc -O3. Extensive experiments were conducted to evaluate the performance of EMOS and the effectiveness of the new lower bound ISLB for pruning branches. Experiments were performed on AMD EPYC CPUs 7702 @2.0GHz under Linux with 128GB of memory.

Since there is no available exact algorithm for MDS. We compared EMOS with three state-of-the-art heuristic algorithms, FastDS [Cai *et al.*, 2020], ScBppw [Fan *et al.*, 2019] and FastMWDS [Wang *et al.*, 2018], which are incomplete and cannot guarantee the optimality of solutions. The three algorithms implement local search approach and employ different heuristic strategies to escape from local minimums. The source code of the three algorithms was kindly provided by their authors and was compiled using GNU g++ -O3.

Experiments were conducted on two standard datasets UDG and T1 and a dataset of real-world graphs, which are widely used to evaluate algorithms for MDS [Jovanovic *et al.*, 2010; Hedar and Ismail, 2010; Potluri and Singh, 2013; Wang *et al.*, 2018; Fan *et al.*, 2019; Cai *et al.*, 2020].

- **UDG.** The set contains 120 unit disk graphs, which are generated with the topology generator described in [Potluri and Singh, 2013] and divided into 12 families, each containing 10 different graphs.
- **T1.** The set contains 530 random connected undirected graphs with vertices ranging from 50 to 1000 [Jovanovic *et al.*, 2010]. The set is divided into 53 groups, each containing 10 different graphs.
- **Real-world graphs.** The set contains 400 real-world graphs which we select from Network Repository[2]. We use this set to evaluate the performance of our algorithm for graphs encoding from real-world applications.

5.1 The Total Performance

In the first experiment, we evaluated the total performance of EMOS and compared it with the three heuristic algorithms in terms of quality of solutions.

[1]Published at https://github.com/huajiang-ynu/ijcai23-mds/
[2]http://networkrepository.com/networks.php

Instance	$\lvert V\rvert$	$\lvert E\rvert$	EMOS		FastDS	ScBppw	Fast MWDS
			$time$	opt			
100_A_0	100	297	0.01	**17**	**17**	**17**(17.9)	**17**
100_A_1	100	302	0.03	**19**	**19**	**19**	**19**
100_A_2	100	309	0.05	**17**	**17**	**17**(18.3)	**17**
100_A_3	100	286	0.05	**19**	**19**	**19**(19.3)	**19**
100_A_4	100	305	0.01	**17**	**17**	**17**(17.9)	**17**
100_A_5	100	284	0.01	**18**	**18**	19	**18**
100_A_6	100	305	0.02	**16**	**16**	**16**(16.5)	**16**
100_A_7	100	297	0.02	**15**	**15**	16	**15**
100_A_8	100	336	0.01	**16**	**16**	**16**(16.3)	**16**
100_A_9	100	306	0.01	**16**	**16**	17	**16**
100_B_0	100	519	0.22	**12**	**12**	**12**	**12**
100_B_1	100	518	0.01	**10**	**10**	11	**10**
100_B_2	100	511	0.01	**9**	**9**(9.9)	**9**(10.2)	**9**
100_B_3	100	516	0.01	**11**	**11**	**11**(11.7)	**11**
100_B_4	100	550	0.01	**11**	**11**	**11**(11.9)	**11**
100_B_5	100	502	0.02	**10**	**10**	**10**(11.6)	**10**
100_B_6	100	559	0.01	**11**	**11**	**11**(12)	**11**
100_B_7	100	598	0.01	**10**	**10**	11(11.5)	**10**
100_B_8	100	527	0.01	**10**	**10**	**10**(10.3)	**10**
100_B_9	100	507	0.03	**10**	**10**	**10**(10.3)	**10**
250_A_0	250	1902	673.8	**18**	**18**(18.4)	19(19.3)	**18**
250_A_1	250	2008	207.9	**17**	18	18(19.7)	**17**
250_A_2	250	1933	1966	**18**	**18**(18.1)	19(20.5)	**18**
250_A_3	250	1946	9376	**19**	**19**(19.1)	**19**(20.4)	**19**
250_A_4	250	1903	160.1	**18**	**18**(18.8)	19(19.7)	**18**
250_A_5	250	1857	74.07	**18**	**18**(18.5)	19(20.3)	**18**
250_A_6	250	1918	2161	**18**	**18**(18.3)	**18**(19.7)	**18**
250_A_7	250	1891	3496	**19**	**19**	**19**(21)	**19**
250_A_8	250	1872	6732	**18**	**18**	19(19.9)	**18**
250_A_9	250	1886	173.5	**17**	**17**(17.8)	18(19.1)	**17**
250_B_0	250	3269	5.3	**10**	11	**10**(11.7)	**10**
250_B_1	250	3344	107.1	**11**	**11**(11.2)	12(13.4)	**11**
250_B_2	250	3294	19.3	**11**	**11**(11.1)	**11**(12.8)	**11**
250_B_3	250	3353	6.77	**10**	**10**(10.6)	12(12.9)	**10**
250_B_4	250	3305	69.34	**11**	12	12(13.4)	**11**
250_B_5	250	3182	14.9	**11**	**11**	**11**(11.4)	**11**
250_B_6	250	3237	2.44	**10**	**10**(10.9)	11(11.3)	**10**
250_B_7	250	3287	57.29	**11**	12	12(13.3)	**11**
250_B_8	250	3218	19.71	**11**	**11**	**11**(12.4)	**11**
250_B_9	250	3158	254.0	**12**	**12**	**12**(12.8)	**12**
500_B_0	500	13289	1972	**11**	**11**(11.5)	**11**(11.9)	**11**
500_B_2	500	12969	4078	**11**	**11**(11.9)	12(12.6)	**11**
500_B_3	500	13036	2273	**11**	**11**(11.4)	12(12.9)	**11**
500_B_6	500	13032	10808	**11**	12	12(13.4)	**11**
500_B_9	500	13366	1375	**11**	**11**(11.8)	12(13)	**11**
total solved				**45**	**40(24)**	**24(2)**	**45(45)**

Table 1: Experimental comparison of EMOS and heuristic algorithms on UDG instances. The average is in brackets if it is not equal to the best for heuristic algorithms. Time is in seconds.

Instance	$\lvert V\rvert$	$\lvert E\rvert$	EMOS		FastDS	ScBppw	Fast MWDS
			$time$	opt			
150_150_0	150	150	59.43	**50**	**50**	**50**	**50**
150_150_1	150	150	167.9	**50**	**50**	**50**	**50**
150_150_2	150	150	259.7	**50**	**50**	**50**(50.5)	**50**
150_150_3	150	150	108.7	**50**	**50**	**50**	**50**
150_150_4	150	150	181.5	**50**	**50**	**50**	**50**
150_150_5	150	150	92.02	**50**	**50**	**50**	**50**
150_150_6	150	150	187.0	**50**	**50**	**50**(50.3)	**50**
150_150_7	150	150	190.6	**50**	**50**	**50**(50.7)	**50**
150_150_8	150	150	179.1	**50**	**50**	**50**(50.8)	**50**
150_150_9	150	150	67.36	**50**	**50**	**50**(50.5)	**50**
150_250_0	150	250	2084	**38**	39	39(39.1)	**38**
150_250_1	150	250	2951	**40**	**40**	**40**(40.1)	**40**
150_250_2	150	250	687.2	**39**	**39**	**39**(39.8)	**39**
150_250_3	150	250	712.4	**39**	**39**(39.8)	40(41.1)	**39**
150_250_4	150	250	2807	**39**	**39**	**39**(39.9)	**39**
150_250_5	150	250	4436	**40**	**40**	**40**	**40**
150_250_6	150	250	964.8	**39**	**39**(39.2)	**39**(39.9)	**39**
150_250_7	150	250	2820	**40**	**40**	**40**(40.9)	**40**
150_250_8	150	250	918.2	**38**	**38**	**38**(39.2)	**38**
150_250_9	150	250	217.3	**39**	**39**(39.2)	**39**(40.1)	**39**
150_500_7	150	500	12617	**23**	**23**(24.3)	24(24.5)	**23**
150_2000_0	150	2000	10505	**9**	10	**9**(9.9)	**9**
150_2000_1	150	2000	11088	**9**	**9**	**9**(9.4)	**9**
150_2000_2	150	2000	8379	**9**	**9**	10	**9**
150_2000_3	150	2000	9697	**9**	10	10	**9**
150_2000_4	150	2000	11919	**9**	**9**	**9**(9.7)	**9**
150_2000_5	150	2000	11382	**9**	**9**	**9**(9.3)	**9**
150_2000_6	150	2000	10054	**9**	10	**9**(9.7)	**9**
150_2000_7	150	2000	10326	**9**	10	10	**9**
150_2000_8	150	2000	12839	**9**	10	10	**9**
150_2000_9	150	2000	9764	**9**	**9**	10	**9**
150_3000_0	150	3000	1906	**7**	**7**	**7**	**7**
150_3000_1	150	3000	1929	**7**	**7**	**7**	**7**
150_3000_2	150	3000	1656	**7**	**7**	**7**	**7**
150_3000_3	150	3000	1836	**7**	**7**	**7**	**7**
150_3000_4	150	3000	1662	**7**	**7**	**7**	**7**
150_3000_5	150	3000	131.7	**6**	**6**(6.9)	**6**(6.7)	**6**
150_3000_6	150	3000	1607	**7**	**7**	**7**	**7**
150_3000_7	150	3000	1893	**7**	**7**	**7**	**7**
150_3000_8	150	3000	1836	**7**	**7**	**7**	**7**
150_3000_9	150	3000	1450	**7**	**7**	**7**(7.2)	**7**
200_250_2	200	250	13454	**61**	**61**(61.2)	**61**(62.6)	**61**
200_250_3	200	250	15234	**61**	**61**	**61**(62.5)	**61**
200_250_9	200	250	17279	**61**	**61**(61.2)	**61**(62.7)	**61**
total solved				**44**	**38(31)**	**36(14)**	**44(44)**

Table 2: Experimental comparison of EMOS and heuristic algorithms on T1 instances. The average is in brackets if it is not equal to the best for heuristic algorithms. Time is in seconds.

We tested EMOS on the three datasets with a cutoff time of 5 hours for each instance. For those instances that EMOS can solve within the cutoff time, we ran the heuristic algorithms on them 10 times using random seeds 1 to 10 and a cutoff time of 1800s for each run. So, the total running time of heuristic algorithms for each instance is also 5 hours. We report the running time ($time$) in seconds and the optimal solution (opt) of EMOS and the best and the average of solutions of 10 runs of heuristic algorithms for each instance in Table 1-3. The average is put in brackets if it is not equal to the best.

For UDG instances, EMOS solves all of the instances in groups of 50, 100 and 250 and a few of instances in group 500. To save space, we discard group 50 and report the remaining 45 instances in Table 1. For the heuristic algorithms, FastMWDS finds the optimum at each run for every instance. FastDS and ScBppw find 40 (24) and 24 (2) optimums in terms of best (average) solutions, respectively.

For T1 instances, EMOS solves all of the instances in group 50 and 100 and most instances in group 150 and a few of instances in group 200. The results of group 150 and 200 are reported in Table 2. For the heuristic algorithms, FastMWDS can find the optimum at each run for the 44 instances. FastDS and ScBppw find 38 (31) and 36 (14) optimums in terms of best (average) solutions, respectively.

For the real-world instances, EMOS solves 203 instances out of the total 400 instances. Moreover, among the 203 solved instances, there are 162 instances for which the initial solutions derived by EMOS are the optimal solutions and EMOS can solve most of them within 1s. We exclude those easy instances and report the remaining 41 instances. Results are shown in Table 3. From the table, we can see that among the 41 instances solved by EMOS, there are 31 instances that can be solved by EMOS within 100s. However, FastDS, ScBppw and FastMWDS can only find 24 (21), 19 (12) and 31(27) optimums in terms of best (average) solutions respectively, showing a weak performance for real-world graphs.

In general, the empirical results show that EMOS is effec-

Instance	$\|V\|$	$\|E\|$	EMOS		FastDS	ScBppw	Fast MWDS
			time	*opt*			
bio-CE-GT	924	3239	10.12	**126**	**126**	**126**(126.8)	**126**
bio-CE-PG	1871	47754	6671	**180**	**180**(180.4)	181	**180**
bio-DM-LC	658	1129	0.03	**163**	**163**	**163**	**163**
bio-SC-TS	636	3959	2705	**124**	**124**	**124**	**124**
bio-WormNet-v3	2445	78736	7243	**139**	140	141(141.8)	**139**
bio-celegans	453	2025	0.01	**29**	**29**	**29**(29.1)	**29**
bio-celegans-dir	453	2025	0.01	**29**	31	31	55(71.5)
ca-GrQc	4158	13422	7.14	**776**	777	777(779.1)	**776**
BA-1_10_60-L5	804	46410	12.37	**3**	**3**(3.7)	**3**	**3**
ENZYMES118	96	121	0.2	**30**	**30**	31	**30**
ENZYMES123	90	127	0.01	**26**	**26**	**26**(26.2)	**26**
ENZYMES295	124	139	0.41	**42**	**42**	**42**	**42**
ENZYMES296	126	141	2.13	**41**	**41**	**41**(41.5)	**41**
ENZYMES297	122	149	2.34	**38**	**38**	39(39.1)	**38**
ENZYMES8	88	133	0.01	**25**	**25**	**25**(25.7)	**25**
fb-pages-food	620	2091	0.03	**118**	**118**	**118**	**118**
gene	1103	1672	0.01	**315**	**315**	**315**(315.6)	**315**
ia-crime-moreno	829	1473	0.19	**211**	**211**	**211**	**211**
ia-enron-employees	151	1526	0.1	**10**	14	14	15(17)
ia-fb-messages	1266	6451	2.07	**249**	**249**	**249**	**249**
insecta-ant-colony1	41	256	0.01	**4**	5	5(5.3)	**4**(4.2)
insecta-ant-colony2	39	245	0.01	**5**	6	6(6.9)	**5**(5.9)
mammalia-voles-plj-trapping	1263	3380	763.5	**280**	282(283.6)	297(298.1)	281(285.3)
rec-movielens-tag-movies-10m	16528	71067	1770	**2992**	3026	3019(3022)	3011(3018)
rec-movielens-user-movies-10m	7601	55384	0.4	**341**	352	352(353.4)	345(349.9)
reptilia-tortoise-network-bsv	136	374	0.01	**34**	35(36.5)	36(36.7)	35
reptilia-tortoise-network-cs	73	132	0.01	**21**	22	22(22.2)	**21**(21.7)
soc-ANU-residence	217	1839	12238	**17**	22(22.9)	21(22.1)	21(21.9)
soc-BlogCatalog-ASU	10312	333983	41.5	**218**	**218**	**218**	**218**
soc-Epinions1	75888	405740	434.5	**15743**	15862	15870(15878)	15877(15887)
soc-advogato	6551	39432	3399	**2192**	2211	2208(2210)	2206(2212)
soc-highschool-moreno	70	274	0.03	**11**	**11**	12(12.5)	**11**
soc-political-retweet	18470	48053	1.01	**3277**	3300	3297(3301)	3285(3287)
soc-student-coop	185	311	0.02	**48**	**48**(48.8)	50	**48**(48.1)
soc-tribes	16	58	0.01	**2**	3	**2**(2.8)	**2**
soc-wiki-elec	8297	100753	1041	**2298**	2300	2299(2300)	**2298**
socfb-Caltech36	769	16656	1.06	**62**	**62**	**62**	**62**
tech-pgp	10680	24316	4127	**2711**	**2711**	2712	**2711**
tech-routers-rf	2113	6632	0.26	**479**	**479**	**479**	**479**
web-EPA	4772	8909	75.87	**763**	**763**	**763**	**763**
web-spam	4767	37375	20.29	**831**	**831**	**831**	**831**
total solved				**41**	**24(21)**	**19(12)**	**31(27)**

Table 3: Experimental comparison of EMOS and heuristic algorithms on real-world graphs. The average is in brackets if it is not equal to the best for heuristic algorithms. Time is in seconds.

tive on the three datasets. Especially, EMOS shows a promising performance for graphs from real-world applications.

5.2 Effects of the Lower Bound and Graph Reduction

To investigate the effects of the new lower bound ISLB for pruning branches and the reduction rule for graph reduction in the preprocessing. We compared EMOS with the following two variants using the 130 instances reported in Table 1-3.

- **EMOS\islb.** It is EMOS but the partition of C in function ReduceBranches is disabled. It uses the whole set C as the branching set B.
- **EMOS\reduction.** It is EMOS but the reduction rule in the preprocessing is disabled, i.e., no vertices are removed and fixed. It uses $\emptyset$ as the initial partial solution.

Figure 2 plots the cumulative numbers of instances solved by EMOS and the two variants. Compared to EMOS,

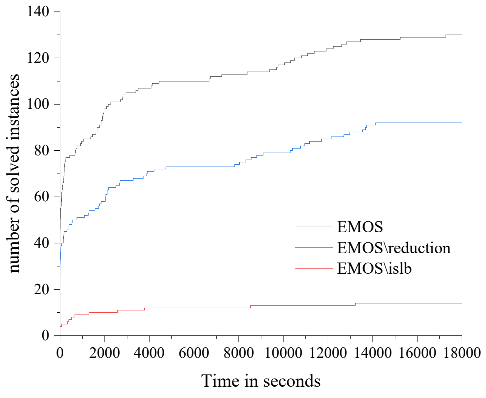

Figure 2: Cumulative numbers of instances solved by EMOS, EMOS\islb and EMOS\reducion. The cutoff time is 18000s.

EMOS\reduction fails to solve 38 instances (24 from real-world dataset and 14 from UDG and T1) in 1800s and the performance of EMOS\islb declines dramatically, showing that the reduction rule in the preprocessing is very effective and the lower bound is crucial to the performance of EMOS.

To investigate the efficiency of the new lower bound ISLB for reducing the number of branches in EMOS, we compute the sum of the size of the branching set B, denoted by $\Sigma|B|$, and the sum of the size of the candidate set C, denoted by $\Sigma|C|$, of every search tree node for each instance in Table 1-3. We call the ratio of $(\Sigma|C| - \Sigma|B|)$ to $\Sigma|C|$ the *pruning rate* of instances by ISLB. We report the percentages of the numbers of instances with different pruning rates in Table 4.

It is easy to see that for more than 80% instances, the pruning rates are greater than or equal to 0.95, and for 93% instances, the pruning rates are greater than or equal to 0.90, showing that the new lower bound ISLB is very efficient for pruning branches and is crucial to the performance of EMOS.

Pruning rates	≥ 0.20	≥ 0.80	≥ 0.90	≥ 0.95
Percentages	100%	98%	93%	81%

Table 4: The percentages of the numbers of instances with different pruning rates on 130 tested instances in Table 1-3.

6 Conclusions

MDS is extremely challenging in computation. In this paper, we propose a novel lower bound and an exact algorithm for MDS. The algorithm implements a branch-and-bound approach and employs the new lower bound to reduce search space. Empirical results show that the new lower bound is very efficient in reducing the number of branches and the new algorithm EMOS is very effective for the tested instances, especially for instances encoding from real-world applications. To the best of our knowledge, this is the first effective branch-and-bound algorithm for MDS.

Acknowledgments

This work is funded by the National Natural Science Foundation of China (Grant NO. 62162066, 62162064 and 62162067), and in part supported by the project of Research and Application of Object Detection Based on Artificial Intelligence, by the project of Research on Data Driven Intelligent Logistics System Construction and Key Technologies (Grant NO. 202302AD080006), Science and Technology Department of Yunnan Province. The experiments were performed on Hefei advanced computing center.

References

[Alber *et al.*, 2004] Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Polynomial-time data reduction for dominating set. *J. ACM*, 51(3):363–384, 2004.

[Cai *et al.*, 2020] Shaowei Cai, Wenying Hou, Yiyuan Wang, Chuan Luo, and Qingwei Lin. Two-goal local search and inference rules for minimum dominating set. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1467–1473. ijcai.org, 2020.

[Chalupa, 2018] David Chalupa. An order-based algorithm for minimum dominating set with application in graph mining. *Inf. Sci.*, 426:101–116, 2018.

[Chaurasia and Singh, 2015] Sachchida Nand Chaurasia and Alok Singh. A hybrid evolutionary algorithm with guided mutation for minimum weight dominating set. *Appl. Intell.*, 43(3):512–529, 2015.

[Dai and Wu, 2004] F. Dai and J. Wu. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 15(10):908–920, 2004.

[Dinh *et al.*, 2014] Thang N. Dinh, Yilin Shen, Dung T. Nguyen, and My T. Thai. On the approximability of positive influence dominating set in social networks. *J. Comb. Optim.*, 27(3):487–503, 2014.

[Dinur and Steurer, 2014] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633. ACM, 2014.

[Fan *et al.*, 2019] Yi Fan, Yongxuan Lai, Chengqian Li, Nan Li, Zongjie Ma, Jun Zhou, Longin Jan Latecki, and Kaile Su. Efficient local search for minimum dominating sets in large graphs. In Guoliang Li, Jun Yang, João Gama, Juggapong Natwichai, and Yongxin Tong, editors, *Database Systems for Advanced Applications - 24th International Conference, DASFAA 2019, Chiang Mai, Thailand, April 22-25, 2019, Proceedings, Part II*, volume 11447 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2019.

[Gao *et al.*, 2018] Jian Gao, Jiejiang Chen, Minghao Yin, Rong Chen, and Yiyuan Wang. An exact algorithm for maximum k-plexes in massive graphs. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1449–1455. ijcai.org, 2018.

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

[Gast *et al.*, 2015] Mikael Gast, Mathias Hauptmann, and Marek Karpinski. Inapproximability of dominating set on power law graphs. *Theor. Comput. Sci.*, 562:436–452, 2015.

[Hedar and Ismail, 2010] Abdel-Rahman Hedar and Rashad Ismail. Hybrid genetic algorithm for minimum dominating set problem. In David Taniar, Osvaldo Gervasi, Beniamino Murgante, Eric Pardede, and Bernady O. Apduhan, editors, *Computational Science and Its Applications - ICCSA 2010, International Conference, Fukuoka, Japan, March 23-26, 2010, Proceedings, Part IV*, volume 6019 of *Lecture Notes in Computer Science*, pages 457–467. Springer, 2010.

[Jiang *et al.*, 2016] Hua Jiang, Chu Min Li, and Felip Manyà. Combining efficient preprocessing and incremental maxsat reasoning for maxclique in large graphs. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, editors, *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 939–947. IOS Press, 2016.

[Jiang *et al.*, 2017] Hua Jiang, Chu Min Li, and Felip Manyà. An exact algorithm for the maximum weight clique problem in large graphs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, California, USA*, pages 830–838, 2017.

[Jovanovic *et al.*, 2010] Raka Jovanovic, Milan Tuba, and Dana Simian. Ant colony optimization applied to minimum weight dominating set problem. In *Proceedings of the 12th WSEAS International Conference on Automatic Control, Modelling & Simulation*, ACMOS'10, page 322–326, Stevens Point, Wisconsin, USA, 2010. World Scientific and Engineering Academy and Society (WSEAS).

[Karp, 1972] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[Lei and Cai, 2020] Zhendong Lei and Shaowei Cai. Solving set cover and dominating set via maximum satisfiability. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The*

Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 1569–1576. AAAI Press, 2020.

[Li *et al.*, 2020] Bohan Li, Xindi Zhang, Shaowei Cai, Jinkun Lin, Yiyuan Wang, and Christian Blum. Nucds: An efficient local search algorithm for minimum connected dominating set. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1503–1510. ijcai.org, 2020.

[Lin *et al.*, 2016] Geng Lin, Wenxing Zhu, and M. Montaz Ali. An effective hybrid memetic algorithm for the minimum weight dominating set problem. *IEEE Trans. Evol. Comput.*, 20(6):892–907, 2016.

[Potluri and Singh, 2011] Anupama Potluri and Alok Singh. Two hybrid meta-heuristic approaches for minimum dominating set problem. In Bijaya K. Panigrahi, Ponnuthurai Nagaratnam Suganthan, Swagatam Das, and Suresh Chandra Satapathy, editors, *Swarm, Evolutionary, and Memetic Computing - Second International Conference, SEMCCO 2011, Visakhapatnam, Andhra Pradesh, India, December 19-21, 2011, Proceedings, Part II*, volume 7077 of *Lecture Notes in Computer Science*, pages 97–104. Springer, 2011.

[Potluri and Singh, 2013] Anupama Potluri and Alok Singh. Hybrid metaheuristic algorithms for minimum weight dominating set. *Appl. Soft Comput.*, 13(1):76–88, 2013.

[Raz and Safra, 1997] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 475–484. ACM, 1997.

[Sanchis, 2002] Laura A. Sanchis. Experimental analysis of heuristic algorithms for the dominating set problem. *Algorithmica*, 33(1):3–18, 2002.

[Shen and Li, 2010] Chao Shen and Tao Li. Multi-document summarization via the minimum dominating set. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, pages 984–992. Tsinghua University Press, 2010.

[van Rooij and Bodlaender, 2011] Johan M. M. van Rooij and Hans L. Bodlaender. Exact algorithms for dominating set. *Discret. Appl. Math.*, 159(17):2147–2164, 2011.

[Wang *et al.*, 2017] Yiyuan Wang, Shaowei Cai, and Minghao Yin. Local search for minimum weight dominating set with two-level configuration checking and frequency based scoring function. *J. Artif. Intell. Res.*, 58:267–295, 2017.

[Wang *et al.*, 2018] Yiyuan Wang, Shaowei Cai, Jiejiang Chen, and Minghao Yin. A fast local search algorithm for minimum weight dominating set problem on massive graphs. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1514–1522. ijcai.org, 2018.

[Yao and Fei-Fei, 2012] Bangpeng Yao and Li Fei-Fei. Action recognition with exemplar based 2.5d graph matching. In Andrew W. Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part IV*, volume 7575 of *Lecture Notes in Computer Science*, pages 173–186. Springer, 2012.

[Zhou *et al.*, 2021] Yi Zhou, Shan Hu, Mingyu Xiao, and Zhang-Hua Fu. Improving maximum k-plex solver via second-order reduction and graph color bounding. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12453–12460. AAAI Press, 2021.