NuMDS: An Efficient Local Search Algorithm for Minimum Dominating Set Problem

Rui Sun[1,2] , **Zhaohui Liu**[1] , **Yiyuan Wang**[1,2*] , **Han Xiao**[1] , **Jiangnan Li**[1] , **Jiejiang Chen**[3*]
[1]School of Computer Science and Information Technology, Northeast Normal University, China
[2]Key Laboratory of Applied Statistics of MOE, Northeast Normal University, China
[3]School of Information Science and Technology, Hangzhou Normal University, China
{ruisun, liuch, wangyy912, xiaohan, lijn101}@nenu.edu.cn, chenjj016@hznu.edu.cn

Abstract

The minimum dominating set (MDS) problem is a crucial NP-hard combinatorial optimization problem with wide applications in real-world scenarios. In this paper, we propose an efficient local search algorithm namely NuMDS to solve the MDS, which comprises three key ideas. First, we introduce a dominate propagation-based reduction method that fixes a portion of vertices in a given graph. Second, we develop a novel two-phase initialization method based on the k-shell decomposition method. Third, we propose a multi-stage local search procedure, which adopts three different search manners according to the current stage of the search. We conduct extensive experiments to demonstrate the outstanding effectiveness of NuMDS, and the results clearly indicate that NuMDS outperforms previous state-of-the-art algorithms on almost all instances.

1 Introduction

Given a graph $G = (V, E)$, a dominating set D is a subset of V such that every vertex in V belongs to D or is adjacent to at least one vertex in D. The minimum dominating set (MDS) problem aims to find a dominating set with the minimum size. The MDS is one of the most important combinational optimization problems in graph theory. In some real-world domains, the MDS has numerous applications, including wireless communication [Wu, 2002; Stojmenovic *et al.*, 2002], infectious diseases analysis [Liu *et al.*, 2011; Takaguchi *et al.*, 2014; Stefan, 2014], and computer vision [Potluri and Bhagvati, 2012; Yao and Li, 2012]. For instance, Shen et al. [2010] formulated the multi-document information extraction problem as the MDS and extracted vital information from a group of files to generate a refined summary.

Because the MDS is NP-hard, no polynomial algorithm can solve this problem unless P=NP [Raz and Safra, 1997]. The approximation algorithms can guarantee an approximation ratio between the solution produced by the algorithm and the optimal solution. Up to now, various studies have proposed approximation algorithms for the MDS [Sanchis, 2002; Eubank *et al.*, 2004; Campan *et al.*, 2015; Chalupa, 2017]. However, it has been proven even obtaining a $(1-\varepsilon)\ln(|V|)$-approximation for the MDS is NP-hard for any $\epsilon > 0$ [Dinur and Steurer, 2014]. Thus, the asymptotic worst-case performance ratio achieved by the greedy algorithm can hardly be further improved. In practice, approximation algorithms usually have poor performance in solving the MDS.

Besides approximation algorithms, there are two types of algorithms, i.e., exact and heuristic algorithms for the MDS. Recent works of exact algorithms for the MDS have been developed in the theoretical aspect, which focuses on improving the upper bound of the time complexity of these algorithms [Grandoni, 2004; Grandoni, 2006; Van Rooij and Bodlaender, 2011]. The state-of-the-art exact algorithms for the MDS are proposed in [Jiang and Zheng, 2023] and [Xiong and Xiao, 2024]. In detail, Jiang et al. [2023] propose a novel lower bound for the MDS based on the definition of 2-hop adjacency of vertices. Xiao et al. [2024] propose two branch-and-bound algorithms to enhance the performance of exact algorithms for the MDS. One of them uses LP relaxations as lower bounds for pruning the search space, and the other one is a pure combinatorial algorithm. Results show that those exact algorithms for the MDS can successfully solve some small instances, but they fail to solve large-scale instances.

To solve large-scale instances in an acceptable time limit, researchers turn to designing many heuristic algorithms for the MDS. Heuristic algorithms for the MDS can be categorized into evolutionary algorithms [Alber *et al.*, 2005; Ho *et al.*, 2006; Hedar and Ismail, 2010a; Potluri and Singh, 2011; Hedar and Ismail, 2012; Giap and Ha, 2014; Chalupa, 2018; Reixach and Blum, 2024] and local search algorithms [Chalupa, 2017; Fan *et al.*, 2019; Cai *et al.*, 2020; Okumuş and Karcı, 2024; Zhu *et al.*, 2024]. Despite the above algorithms, all the heuristic algorithms for the minimum weighted dominating set (MWDS) problem [Jovanovic *et al.*, 2010; Potluri and Singh, 2013; Chaurasia and Singh, 2015; Bouamama and Blum, 2015; Wang *et al.*, 2017; Wang *et al.*, 2018; Chen *et al.*, 2023] can be adapted to solve the MDS by assigning the weight of all vertices as 1. According to the literature, the current best heuristic algorithms for the MDS are DmDS [Zhu *et al.*, 2024] and FastDS [Cai *et al.*, 2020]. The best-performing heuristic algorithm for the MWDS is DeepOpt-MWDS [Chen *et al.*, 2023].

This work proposes a local search algorithm namely

*Corresponding author

NuMDS[1] to solve the MDS. It contains three novel ideas. First, we propose a dominate propagation-based reduction method (DPRM) as a preprocessing procedure to fix a portion of vertices in an optimal solution and exclude some other vertices from an optimal solution. The proposed DPRM comprises four propagation-based reduction rules, which take into account three vertex relationships, including the strong cover relationship, the strong ignore relationship, and the unique dominate relationship. The proposed rules significantly promote each other, leading to reducing a large portion of the search space.

The second strategy is a two-phase initialization method. Previous initialization methods usually use greedy ways to generate an initial solution. However, according to preliminary experiments, dominating vertices in sparse regions typically requires a larger amount of vertices than in dense regions. To address this issue, we propose a two-phase initialization procedure, including priority and greedy phases. In the first phase, our proposed initialization method focuses on dominating vertices in the sparse regions of a given graph by utilizing the k-shell decomposition method. Subsequently, in the second phase, vertices are added in a greedy manner. By employing this initialization method, our search procedure starts from a good starting point.

The third idea of NuMDS is a multi-stage search mechanism. Previous local search algorithms for the MDS usually use a uniform search strategy throughout the entire procedure, and do not consider the current search state. Unlike previous local search algorithms for the MDS, we introduce a novel local search procedure, which considers three search stages. When the current solution exhibits evident potential for improving the quality of solution, the algorithm employs a greedy search approach to accelerate the convergence speed. But, when the current solution reaches a state where further improvement becomes challenging, the algorithm switches to a randomized search manner and an exploitation search manner sequentially.

We carry out extensive experiments to illustrate the performance of our proposed algorithm. Experimental results show that NuMDS outperforms all the competitors on almost all instances.

2 Preliminaries

Given an undirected graph $G = (V, E)$, V denotes the vertex set and $E \subseteq V \times V$ denotes the edge set. For each $e = (u, v)$, we call vertices u and v are the endpoints of edge e, and u and v are neighbors. We use $N(v) = \{u \in V | (v, u) \in E\}$ to denote v's neighbors, and the degree of v is represented as $|N(v)|$. Moreover, $N[v]$ is the closed neighborhood of v, defined as $N(v) \cup \{v\}$. For a vertex set $S \subseteq V$, $N[S] = \bigcup_{v \in S} N[v]$ is used to denote the closed neighborhood of S. The MDS is defined as follows.

Definition 1. *Given an undirected graph $G = (V, E)$, a dominating set of G is a vertex subset $D \subseteq V$ such that $\forall v \in V, v \in N[D]$. The minimum dominating set (MDS) problem aims to find a dominating set with the minimum size.*

During the local search, a candidate solution D is a vertex subset of V, and $N[D]$ is the vertex set that dominated by D. $V_{und}^{D} = V \setminus N[D]$ is the set of vertices that do not dominated by D. The scoring function used in our proposed algorithm is defined as $score(v) = |V_{und}^{D}| - |V_{und}^{D'}|$, where $D' = D \setminus \{v\}$ if $v \in D$, and $D' = D \cup \{v\}$ otherwise. Obviously, if $v \in D$, $\mathrm{score}(v) \leq 0$, and if $v \notin D$, $\mathrm{score}(v) \geq 0$.

3 Reduction Method

The reduction method is an effective technique to deal with various NP-hard problems, such as the MDS [Xiong and Xiao, 2024], the MWDS [Chen *et al.*, 2023], and the maximum weighted clique problem [Wang *et al.*, 2020]. It aims to reduce the search space by determining the values of some variables (the states of some vertices in the MDS). In this section, we propose a dominate propagation-based reduction method (DPRM). First, we review previous reduction methods for the MDS. Then, we present the dominate propagation-based reduction rules and provide detailed information about the DPRM algorithm.

3.1 Previous Reduction Methods for MDS

In the last decades, various kernelization methods have been designed for special types of graphs for the MDS, such as planar graphs [Guo and Niedermeier, 2007; Chen *et al.*, 2007] and graph classes of bounded expansion [Drange *et al.*, 2014]. Given a graph, these kernelization methods compute an equivalent but smaller kernel whose size can be bounded by a function of a specified parameter. The kernelization methods remain challenging to apply to the general graph for the MDS, because it belongs to the class W[2], which formalizes the intractability of the problem from the perspective of parameterized complexity [Downey and Fellows, 1995; Guo and Niedermeier, 2007].

To address the general graph for the MDS, numerous effective reduction methods have been developed. Alber et al. [2004] introduced two neighborhood relationship checking-based reduction rules, namely RLNSV and RLNPV. To further improve these rules, they incorporated several simple reduction rules, resulting in RLNSV_imp and RLNPV_imp, with time complexities of $O(|V|^3)$ and $O(|V|^4)$, respectively [Alber *et al.*, 2006]. Very recently, Xiong and Xiao [2024] formalized the MDS as the extended minimum dominating set problem and proposed three reduction rules, denoted as EMDSP_reduce, which has a time complexity of $O(|V| \times |E| \times \Delta)$, where Δ denotes the largest degree of a given graph. Additionally, a representative reduction method for the MDS utilizes three inference rules to fix a portion of the vertices quickly for large-scale graphs [Cai *et al.*, 2020; Zhu *et al.*, 2024]. This method can be viewed as a specific case of the above five methods, which has a time complexity of $O(|V|)$.

3.2 Dominate Propagation-Based Reduction Method

In this section, we propose a dominate propagation-based reduction method (DPRM) to further improve the reduction effectiveness for the MDS. Our method is built upon the "usefulness" of the vertices in a graph G. The criterion is whether

[1]Source code and supplementary materials are available at https://github.com/yiyuanwang1988/NuMDS.

the vertices in $N[v]$ need v to be dominate. If so, v is useful, otherwise it is useless. We use V_{in} to denote these useful vertices and V_{out} to denote the useless vertices, respectively. In addition, for the vertices that can be confirmed to be dominated, we mark them as ignorable vertices, and their set is denoted as V_{ignore}. These vertices do not need to check again whether they should be dominated in the reduction process. The introduction of V_{ignore} can help further expand the size of V_{in} and V_{out}. Formally, for any $S \subseteq V \setminus V_{out}$, if $V \setminus V_{ignore} \subseteq N[S]$, then $V \subseteq N[S]$. Moreover, we guarantee there is an optimal solution S_{opt} such that $V_{in} \subseteq S_{opt}$ and $V_{out} \subseteq V \setminus S_{opt}$.

After the reduction procedure, vertices in V_{in} are fixed in the candidate solution, vertices in V_{out} are fixed out from the candidate solution, and only vertices in $V \setminus (V_{in} \cup V_{out})$ can be operated (i.e., to be added or removed) during the local search. Based on these vertex sets, we define three important relationship of vertex sets (or vertices), which can help us develop the corresponding reduction rules.

Definition 2. Strong Cover Relationship. *Given an undirected graph $G = (V, E)$, a fix-in vertex set V_{in}, an ignored vertex set V_{ignore}, two vertex sets $V_S, V_{S'} \subseteq V$ and $V_S \cap V_{S'} = \emptyset$, if $N[V_{S'}] \setminus (N[V_{in}] \cup V_{ignore}) \subseteq N[V_S] \setminus (N[V_{in}] \cup V_{ignore})$, then V_S strongly covers $V_{S'}$, denoted as $V_S \rhd V_{S'}$.*

Definition 3. Strong Ignore Relationship. *Given an undirected graph $G = (V, E)$, a fix-out vertex set V_{out}, and two vertex sets $V_S, V_{S'} \subseteq V$, if $N[V_{S'}] \setminus V_{out} \subseteq N[V_S]$, then V_S strongly ignores $V_{S'}$, denoted as $V_S \blacktriangleright V_{S'}$.*

Definition 4. Unique Dominate Relationship. *Given an undirected graph $G = (V, E)$, a fix-out vertex set V_{out}, an undominated vertex $v \in V$ and one vertex in its closed neighborhood $v_{adj} \in N[v]$, if $N[v] \setminus \{v_{adj}\} \subseteq V_{out}$, then v_{adj} uniquely dominates v, denoted as $v_{adj} \succ v$.*

Then we propose our reduction rules used in DPRM, and the detailed proofs can be found in the supplementary material.

Fix-I Reduction Rule 1. For two vertices $v, v' \in V \setminus (N[V_{in}] \cup V_{ignore})$, if $\{v\} \blacktriangleright \{v'\}$, then $V_{ignore} = V_{ignore} \cup \{v\}$.

Fix-Out Reduction Rule 2. For a vertex $v \in V \setminus V_{out}$ and another vertex $v' \in V \setminus (V_{in} \cup V_{out})$, if $\{v\} \rhd \{v'\}$, then $V_{out} = V_{out} \cup \{v'\}$.

Fix-In Reduction Rule 3. For an undominated vertex $v \in V$ and its closed neighbor $v_{adj} \in V$, if $v_{adj} \succ v$, then $V_{in} = V_{in} \cup \{v_{adj}\}$.

To facilitate understanding the above three reduction rules, we provide an illustrative example in Figure 1. First, we initialize $V_{in} = V_{out} = V_{ignore} = \emptyset$. Because $\{v_7\} \rhd \{v_6, v_9, v_{10}\}$, $\{v_2\} \rhd \{v_1, v_3\}$, $\{v_7\} \blacktriangleright \{v_6, v_9, v_{10}\}$, and $\{v_2\} \blacktriangleright \{v_1, v_3\}$, we can obtain $V_{out} = \{v_1, v_3, v_6, v_9, v_{10}\}$ based on the Fix-out Reduction Rule 2 and $V_{ignore} = \{v_2, v_7\}$ based on the Fix-I Reduction Rule 1. Then, we can deduce $v_7 \succ v_{10}$, so we put $\{v_7\}$ into V_{in} according to Fix-in Reduction Rule 3. After that, $N[V_{in}] = \{v_2, v_3, v_5, v_6, v_7, v_8, v_9, v_{10}\}$. So we can deduce $\{v_4\} \rhd \{v_5, v_8\}$, $\{v_1\} \rhd \{v_4\}$, resulting in $V_{out} = \{v_1, v_3, v_6, v_9, v_{10}\} \cup \{v_5, v_8\} \cup \{v_4\} =$

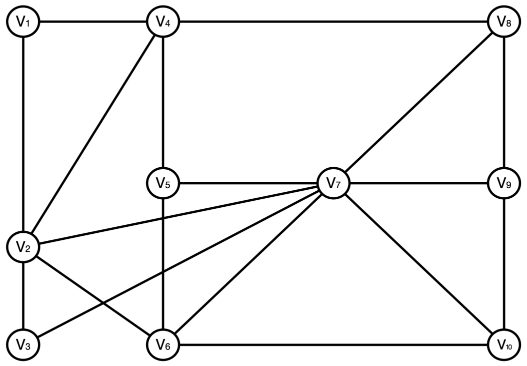

Figure 1: An example for the first three reduction rules, where $V_{in} = \{v_2, v_7\}$, $V_{out} = \{v_1, v_3, v_4, v_5, v_6, v_8, v_9, v_{10}\}$.

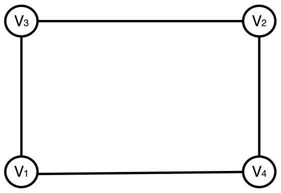

Figure 2: An example for the fourth reduction rule, where $V_{in} = \{v_1, v_2\}$, $V_{out} = \{v_3, v_4\}$.

$\{v_1, v_3, v_4, v_5, v_6, v_8, v_9, v_{10}\}$ according to Fix-out Reduction Rule 2. Then, $v_2 \succ v_1$, so $V_{in} = V_{in} \cup v_2$. As a result, all vertices are added into V_{in} or V_{out}.

The time complexity of the Fix-I Reduction Rule 1, Fix-Out Reduction Rule 2, and the Fix-In Reduction Rule 3 are $O(|E| \times \Delta)$, $O(|E| \times \Delta)$, and $O(|V| + |E|)$, respectively, where Δ is the largest degree in G. These three reduction rules cover the all scenarios in which rules applied in EMDSP [Xiong and Xiao, 2024], which is the state-of-the-art exact algorithm for the MDS. Key differences between the above reduction rules and EMDSP_reduce, along with their respective time complexity derivation procedures, are provided in the supplementary material. Despite the above reduction rules, we further propose another new reduction rule.

Fix-Out-Set Reduction Rule 4. For two vertices $v_1, v_2 \in V$, a vertex set $V_{S'} \subseteq V \setminus V_{out}$ and $|V_{S'}| > 1$, if $\{v_1, v_2\} \rhd V_{S'}$, then

$$V_{out} = V_{out} \cup \bigcup_{v' \in CSet} V_{S'} \setminus N[v']$$

where $CSet = \{v' \in V_{S'} \setminus N[V_{in}] \mid N[v'] \subseteq (V_{S'} \cup V_{out} \cup \{v_1, v_2\})\}$.

The time complexity of the above reduction rule is $O(|V|^2 \times \Delta^2)$. The detailed derivation procedure is provided in the supplementary material. We also provide a small example to illustrate this reduction rule in Figure 2. Obviously, this square example cannot be reduced by the first three reduction rules. But it can be reduced by the Fix-Out-Set Reduction Rule 4 as follows: First, the V_{in}, V_{out}, and V_{igore} are initialized as $\emptyset$. Additionally, we can obtain $\{v_1, v_2\} \rhd \{v_3, v_4\}$, and $CSet = \{v_3, v_4\}$. After that, according to the Fix-Out-

Algorithm 1: DPRM

```
   Input: An graph $G = (V, E)$
   Output: $\langle V_{in}, V_{out} \rangle$
1  $V_{in} := V_{out} := V_{ignore} := \emptyset$;
2  repeat
3      foreach $v \in V_{und}^{V_{in}} \setminus V_{ignore}$ do
4          if $\exists v' \in N(v) \cap V_{und}^{V_{in}} \setminus V_{ignore}, \{v\} \blacktriangleright \{v'\}$
             then
5              $V_{ignore} = V_{ignore} \cup \{v\}$;
6      foreach $v \in V \setminus V_{out}$ do
7          if $\exists v' \in N(v) \cap (V \setminus (V_{in} \cup V_{out})), \{v\} \triangleright \{v'\}$
             then
8              $V_{out} = V_{out} \cup \{v'\}$;
9      foreach $v \in V \setminus V_{und}^{V_{in}}$ do
10         if $\exists v' \in N[v], N[v] \setminus V_{out} = \{v'\}$ then
11             $V_{in} := V_{in} \cup \{v'\}$;
12 until no vertices are added into $V_{in}, V_{out}$ and $V_{ignore}$;
13 if $V \setminus (V_{in} \cup V_{out})| < M$ then
14     apply Fix-out-Set Reduction Rule 4 thoroughly;
15     apply Fix-In-Set Reduction Rule 3 thoroughly;
16 else
17     sample $N$ vertex pairs from $V \setminus (V_{in} \cup V_{out})$;
18     apply Fix-out-Set Reduction Rule 4 for the above
         vertex pairs;
19     apply Fix-In-Set Reduction Rule 3 thoroughly;
20 return $\langle V_{in}, V_{out} \rangle$
```

Set Reduction Rule 4, $V_{out} = V_{out} \cup \{\{v_3, v_4\} \setminus N[v_3]\} \cup \{\{v_3, v_4\} \setminus N[v_4]\} = \{v_3, v_4\}$. Then, we will get $v_1 \succ v_1$, $v_2 \succ v_2$. As a result, $V_{in} = \{v_1, v_2\}$.

3.3 The DPRM Algorithm

We present the DPRM algorithm in Algorithm 1. The DPRM algorithm begins by initializing V_{in}, V_{out} and V_{ignore}. Then, it entering a loop that sequentially applies the Fix-I Reduction Rule 1, Fix-Out Reduction Rule 2, and Fix-In Reduction Rule 3 (Lines 2-12). For each vertex v, we traverse each of its neighbors v' and check whether v and v' satisfy the conditions specified by the three reduction rules (Lines 4, 7, and 10). This loop continues until no vertices are added to V_{in}, V_{out}, and V_{ignore} (Line 12).

Afterward, since Fix-Out-Set Reduction Rule 4 have a higher time complexity, we apply this reduction rule as follows: If the number of remaining vertices is less than M after the reduction, we thoroughly apply the Fix-Out-Set Reduction Rule 4 and Fix-In Reduction Rule 3 (Lines 13-15). Otherwise, we randomly select N pairs of vertices and treat each of them as the $\{v_1, v_2\}$ of the two reduction rules. Then we conduct these two reduction rules for the selected N vertex pairs (Lines 17-19). Based on our preliminary experiments, we set M as 1000 and N as 10000.

4 Two-Phase Initialization Method

In this section, we first review previous initialization methods adopted by the local search algorithms for the MDS. Then, our two-phase initialization method is presented.

4.1 Previous Initialization Methods for MDS

The initialization procedure commonly used by previous local search algorithms for the MDS is typically a greedy approach, denoted as Init$_{greedy}$ [Fan *et al.*, 2019; Cai *et al.*, 2020]. This method greedily adds vertices with the largest score to the candidate set until all the vertices are dominated. Furthermore, Zhu et al. [2024] integrate a perturbation process into Init$_{greedy}$, resulting in a perturbation initialization method denoted as Init$_{perturb}$. In DmDS [Zhu *et al.*, 2024], both Init$_{greedy}$ and Init$_{perturb}$ are conducted, and the best initial solution obtained from these two methods is selected.

4.2 Two-Phase Initialization Procedure

The classic Init$_{greedy}$ method adds the vertex with the largest score at each iteration. These high-score vertices are usually located in dense areas of the graph, which often leads to the neglect of vertices in sparse areas. In fact, how to dominate these vertices is critical to the initialization process. This is because vertices in sparse areas tend to dominate a small number of neighbors, meaning that a larger number of vertices is needed to dominate the same number of vertices in sparse areas compared to dense areas.

Different from previous methods, our initial solution prioritizes the addition of vertices that can dominate sparse areas. Specifically, we adopt k-shell decomposition [Kitsak *et al.*, 2010] to partition the graph into different areas. There are several cluster sets $CS = \{V_1, V_2, \ldots, V_n\}$ in the k-shell decomposition, and each set V_i is associated with a shell value denoted as $shell(V_i)$, where a higher $shell(V_i)$ value indicates V_i is located in a denser area. At the beginning, CS is initialized as an empty set, then the k-shell algorithm iteratively removes a set of vertices from the graph. In the ith iteration, the minimum degree of all the vertices is recorded as d_{min}, and vertices with the degree value not higher than d_{min} are removed from the given graph and then they are stored in V_i until the degree value of all the remaining vertices is larger than d_{min}. Then, the set of removed vertices V_i is added to CS. After that, the shell value of the current vertex subset $V_i \in CS$ is set to d_{min}, i.e., $shell(V_i) = d_{min}$. The shell decomposition process continues until all vertices have been removed from the graph. Its time complexity is $O(|V|+|E|)$.

The complete two-phase initialization method is shown in Algorithm 2. At first, the algorithm proceeds with the k-shell decomposition procedure and stores all vertex subsets with a $shell$ value less than del_thre in the candidate set CS_{sp}, where del_thre is a parameter (Lines 1–2). The initial solution D is set to an empty set (Line 3). In the following, there are two phases, including the priority phase (Lines 4–9) and the greedy phase (Lines 10–12). In the first phase, if CS_{sp} is not empty, the algorithm iteratively selects a vertex subset V_i with the smallest $shell$ value, breaking ties randomly (Line 5). For each vertex $v_p \in V_j \cap V_{und}^{D}$, the algorithm selects an added vertex from $N[v_p] \setminus V_{out}$ with the largest score value,

Algorithm 2: Init$_{tp}$

```
Input: A graph $G = (V, E)$, fix-in set $V_{in}$, fix-out set $V_{out}$
Output: An initial solution $D$
1  applying k-shell decomposition to obtain a cluster set $CS$;
2  $CS_{sp} := \{V_i \in CS \mid shell(V_i) < del\_thre\}$;
3  $D := V_{in}$;
4  while $CS_{sp} \neq \emptyset$ do
5      select a vertex subset $V_j$ with the minimum $shell$ value;
6      foreach $v_p \in V_j \cap V^D_{und}$ do
7          select a vertex $v_{add} \in N[v_p] \setminus V_{out}$ with the largest
             $score$ value;
8          $D := D \cup \{v_{add}\}$;
9      $CS_{sp} := CS_{sp} \setminus \{V_j\}$;
10 while $V^D_{und} \neq \emptyset$ do
11     select a vertex $v_{add} \in V \setminus V_{out}$ with the largest $score$
         value;
12     $D := D \cup \{v_{add}\}$;
13 return $D$;
```

and then adds it into D (Lines 7–8). The vertex subset V_i is removed from CS_{sp} (Line 9). In the second phase, a vertex with the largest $score$ value and outside the V_{out} is iteratively added until all the vertices are dominated (Lines 10–12). Finally, the algorithm returns an initial solution D (Line 13).

5 The NuMDS Algorithm

We present a novel multi-stage search mechanism, followed by introducing our proposed algorithm namely NuMDS.

5.1 Multi-Stage Search Mechanism

Previous local search algorithms for the MDS and MWDS have commonly employed a unified search approach throughout the entire search procedure [Chalupa, 2017; Fan *et al.*, 2019; Cai *et al.*, 2020; Chen *et al.*, 2023]. However, this type of search pattern lacks the flexibility to adapt various search approaches based on the current search state. The local search usually faces two main challenges, including slow convergence speed during the early stages of the search and difficulty in escaping from local optima during the later stages of the search. To deal with these challenges, we propose a multi-stage local search mechanism that incorporates different search manners at different stages of the search. In the early stages of the search, we employ a greedy manner to accelerate the convergence speed of the search. If improving the quality of current solution becomes gradually challenging, we adopt two kinds of manners sequentially, including random and exploitation search manners.

During the local search, we use a variable $unimprove$ to measure the current state of the search, which is defined as the number of consecutive steps during the local search where the best solution has not been improved. As the value of $unimprove$ increases, it indicates that improving the quality of the current solution is pretty hard. Moreover, we define a hybrid scoring function to evaluate each vertex in the candidate solution D, formally expressed as follows: $\forall v \in D, score_{hybrid}(v) = \frac{age(v)}{|score(v)|+1}$, where the age value of vertex v is defined as the number of steps for which the vertex has not been operated (i.e., to be added or removed). This function considers both the diversification effect and the direct impact of operating a vertex in D. Its goal is to prioritize selecting vertices that have not been operated for a long period while simultaneously resulting in few undominated vertices.

The proposed mechanism modifies the vertex removal procedure of the local search to align with three search manners. Thus, how to define search manner is a key issue. We employ three variations of the best from multiple selections heuristic (BMS) [Cai, 2015] by adjusting the parameter settings to match the respective search manners. The BMS strategy works as follows, i.e., randomly sampling a certain amount of vertices and then selecting the best one among them. In the following, we give three search manners.

Greedy Search Manner. If $unimprove < thre$, the algorithm randomly samples t_1 vertices from $D \setminus V_{out}$ and removes the one with the smallest score value, breaking ties with the largest age value.

Random Search Manner. If $thre \leq unimprove < 2 \times thre$, the algorithm removes a random vertex from $D \setminus V_{out}$.

Exploitation Search Manner. If $unimprove \geq 2 \times thre$, the algorithm randomly samples t_1 vertices from $D \setminus V_{out}$, and removes the one with the largest $score_{hybrid}$ value, breaking ties randomly.

Where $thre$ and t_1 are two parameters. Note that the score-based vertex removal procedure indicates a greedy manner, which limits the total number of undominated vertices. Regarding the transition to the second and third search manners, the algorithm gradually enhances the exploration capability of the search. As the value of $unimprove$ increases, the search procedure naturally shifts its inclination from exploration to exploitation, i.e., from the random search manner to the exploitation search manner. Specifically, for the third search manner, removing vertices with the largest $score_{hybrid}$ value can be beneficial for escaping from local optima.

5.2 Details of the NuMDS

The proposed algorithm NuMDS is presented in Algorithm 3. First, the DPRM is applied to generate the V_{in} and V_{out}, and $unimprove$ is initialized as 0 (Lines 1–2). Subsequently, the proposed initialization method is employed to obtain an initial solution, which serves as the starting point of the following local search procedure (Line 3). In each iteration, the algorithm begins by checking whether the undominated vertex set V^D_{und} is empty. If so, the algorithm iteratively removes some redundant vertices (Lines 6–8). The best solution D_{best} is updated by D and $unimprove$ is reset to 0 (Line 9). Lastly, the algorithm utilizes the BMS heuristic to remove a vertex with the largest score value, resulting in an unfeasible candidate solution smaller than D_{best} (Lines 10–11).

After that, the local search turns to begin a swap procedure, involving removing two vertices and adding two other vertices (Lines 13–22). Specifically, during the vertex removing procedure, the first removal vertex is selected using the BMS strategy to choose a vertex with the largest score value, whereas the second removal vertex is selected according to the proposed three search manners (Lines 13–17). During the

Algorithm 3: NuMDS

```
Input: A graph $G = (V, E)$
Output: the best obtained solution $D_{best}$
1  $\langle V_{in}, V_{out} \rangle := DPRM(G)$;
2  $unimprove := 0$;
3  $D := D_{best} := Init_{tp}(G, V_{in}, V_{out})$;
4  while $elapsed\_time < cutoff$ do
5      if $V^D_{und} = \emptyset$ then
6          while there exist vertices in $D$ whose score is 0 do
7              select a vertex $v_1 \in D \setminus V_{in}$ with
                 $score(v_1) = 0$;
8              $D := D \setminus \{v_1\}$;
9          $D_{best} := D$ and $unimprove := 0$;
10         select a vertex $v_2 \in D \setminus V_{in}$ with the largest $score$
             among $t_2$ samples, breaking ties by the largest age;
11         $D := D \setminus \{v_2\}$;
12         continue;
13     remove a vertex $v_1 \in D \setminus V_{in}$ with the largest $score$
         among $t_2$ samples, breaking ties by the largest age
         value;
14     $D := D \setminus \{v_1\}$;
15     choose a search manner based on the $unimprove$ value;
16     select a vertex $v_2$ based on the corresponding manner;
17     $D := D \setminus \{v_2\}$;
18     select a vertex $u_1 \in V^D_{und} \setminus V_{out}$ with the largest score
         value, breaking ties by the largest age value;
19     $D := D \cup \{u_1\}$;
20     if $V^D_{und} = \emptyset$ then
21         select a vertex $u_2 \in V^D_{und} \setminus V_{out}$ with the largest
             score value, breaking ties by the largest age value;
22         $D := D \cup \{u_2\}$;
23     $unimprove := unimprove + 1$;
24 return $D_{best}$;
```

Parameter	Range	Final value
t_1	{5, 10, 15, 20}	10
t_2	{35, 40, 45, 50}	45
del_thre	{4, 6, 8, 10}	6
$thre$	{2000, 2500, 3000}	2500

Table 1: Tuned parameters of NuMDS.

vertex adding procedure, if undominated vertices exist, two vertices with the largest score value are sequentially added (Lines 18–22). In case multiple vertices have the same score for these two procedures, ties are broken by favoring the vertex with the largest age value. At the end of each iteration, $unimprove$ is increased by 1 (Line 23). The local search terminates when the current run time reaches the predefined *cutoff* time (Line 4). Finally, the best solution D_{best} is returned (Line 24).

6 Experiments

We conduct extensive experiments to evaluate the performance of NuMDS. According to previous studies [Cai *et al.*, 2020; Zhu *et al.*, 2024], DmDS [Zhu *et al.*, 2024] and FastDS [Cai *et al.*, 2020] significantly outperform other heuristic algorithms for the MDS. Thus, we first select these two algorithms as our competitors. Besides, we compare NuMDS with the state-of-the-art MWDS solver DeepOpt-MWDS [Chen *et al.*, 2023]. The source codes of all algorithms are kindly provided by the authors. All the algorithms are implemented in C++ and compiled by g++ with -O3 option. All experiments are conducted on Intel Xeon Gold 6238 CPU @ 2.10GHz with 256GB RAM under CentOS 7.9. For all comparative algorithms, we use the parameters specified in the corresponding literature. Note that the experiments in DmDS were conducted on a significantly more powerful CPU than ours. To achieve the results reported in the literature, we set the time limit to 1800 seconds. For each instance, all algorithms are executed 10 times with the random seeds from 1 to 10.

We collect all instances that have already been tested by DmDS [Zhu *et al.*, 2024] and FastDS [Cai *et al.*, 2020], comprising 4 standard benchmarks and 4 real-world benchmarks. The selected benchmarks are presented as follows.

UDG. This benchmark is widely used in many algorithms for the MDS [Hedar and Ismail, 2010b; Potluri and Singh, 2011; Cai *et al.*, 2020; Zhu *et al.*, 2024]. UDG comprises 12 families, and each family contains 10 instances.

T1. This benchmark consists 54 families, with each family containing 10 instances of the same size. Each instance has two weight functions [Romania, 2010] and we set the weight of each vertex to 1.

DIMACS. This benchmark is sourced from the Second DIMACS Implementation Challenge (1992-1993) [2]. It has been extensively tested in various significant combinatorial optimization problems, including 61 instances.

BHOSLIB. This benchmark is generated from RB model [Xu *et al.*, 2007], containing 41 instances.

SNAP. The Stanford Large Network Dataset Collection comprises a collection of real-world graphs ranging from 10^4 vertices to 10^7 vertices in size.

DIMACS10. This benchmark is sourced from the 10th DIMACS Implementation Challenge (2010) [3], containing many challenging instances.

Network Repository. The Network Data Repository [Rossi and Ahmed, 2015] is a comprehensive collection of real-world graphs gathered from various domains.

The standard benchmarks have 168 instances, including UDG, T1 and DIMACS and BHOSLIB, and the remaining benchmarks have 260 real-world instances. In total, we select 428 instances. In detail, for the instances selection of SNAP and DIMACS10, we follow the DmDS and FastDS [Cai *et al.*, 2020; Zhu *et al.*, 2024]. Only instances with a minimum of 30,000 vertices are chosen, resulting in a set of 53 instances. For the Network Repository, many previous studies for the MDS and MWDS select vertices with more than 10^5 and edges with more than 10^6 [Wang *et al.*, 2018; Cai *et al.*, 2020; Chen *et al.*, 2023], resulting in 65 instances. In 2024, Zhu et al. [2024] add 142 additional instances from NDR to test the performance of DmDS. Consequently, we divided the instances in the Network Repository into two sets:

[2]ftp://dimacs.rutgers.edu/pub/challenges
[3]https://www.cc.gatech.edu/dimacs10/

Benchmark	#ins	NuMDS #min(#avg)	DmDS min(#avg)	FastDS #min(#avg)	DeepOpt-MWDS #min(#avg)
UDG	12	**12(12)**	**12(12)**	11(9)	**12(12)**
T1	54	**54(53)**	44(29)	49(41)	47(43)
DIMACS	61	**61(60)**	55(47)	59(53)	59(55)
BHOSLIB	41	**41(41)**	33(13)	35(26)	26(17)
DIMACS10	31	**23(24)**	12(11)	10(10)	10(9)
SNAP	22	**20(20)**	14(11)	13(11)	16(13)
NDR1	65	**62(61)**	37(31)	32(27)	26(22)
NDR2	142	**137(117)**	58(70)	51(44)	45(42)
#total	428	**410(388)**	265(224)	260(221)	241(213)

Table 2: Experimental results on all benchmarks. #min and #avg denote the number of instances where the corresponding algorithm finds the best minimum and average solutions among all algorithms.

NDR1 and NDR2. NDR1 consists of the 65 instances tested by previous studies for the MDS, and NDR2 comprises the 142 additional instances added by DmDS.

We utilize the automatic configuration tool irace [López-Ibáñez *et al.*, 2016] to tune the parameters. For the training set, we randomly selected 10% instances from each benchmark. The tuning process is given a budget of 4000 runs for the training set with a time budget of 1000 seconds per run. Table 1 presents the selected parameter values. For all competitors, we used the parameters from the relevant literature and optimized them for the newly added instances using the irace tool.

Despite using irace for parameter tuning, we manually analyze the parameter sensitivity of NuMDS as follows. Initially, we select 10% instances from each benchmark, resulting in a total of 41 instances. We vary t_1 with values 5, 10, and 15, t_2 with values 40, 45, and 50, and $thre$ with values 2000, 2500, and 3000. In total, we tested 27 different parameter combinations. NuMDS is executed within a cutoff time of 1800 seconds using random seeds of 1 to 10. According to our experiments, under the optimal parameter settings, the best solution achieved is, on average, 0.0021% smaller than the best solutions obtained using other parameter combinations. Additionally, we selected del_thre as 4 and 8. The initial solution achieved with del_thre set to 6 is, on average, 0.0017% smaller than the initial solutions obtained with del_thre set to 4 and 8. These results show that the performance our algorithm is not sensitive to the parameters.

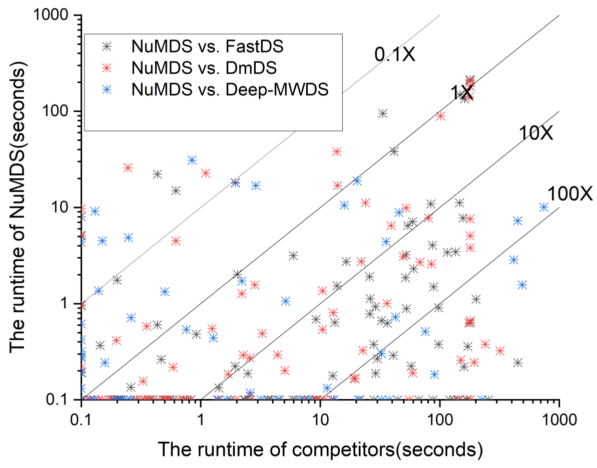

Figure 3: Comparison of run time of NuMDS and its corresponding competitor.

6.1 Results on All the Benchmarks

Results on all the benchmarks are summarized in Table 2. We present the detailed results of NuMDS and all competitors in the supplementary material.

Results on standard and real-world benchmarks

For all standard benchmarks, NuMDS can find the best solution for all the instances. Specifically, NuMDS outperforms FastDS, DmDS, and DeepOpt-MWDS for 14, 24, and 24 instances. Furthermore, among 168 standard instances, it only fails to obtain the minimal average solution on only 2 instances, while FastDS, DmDS, and DeepOpt-MWDS fail to achieve the minimal average solution for 39, 67, and 41 instances. As for the real-world instances, NuMDS performs best on all the benchmarks. In detail, it obtains the 23 best solutions (out of 31 instances) on DIMACS10 benchmark,

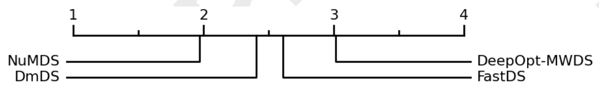

Figure 4: Critical difference analysis for all algorithms.

Benchmark	#ins	vs. Gurobi #bet(#wor)	vs. BIBLP #bet(#wor)
Standard	168	**73**(1)	**72**(1)
SNAP&DIMACS10	53	**22**(10)	**43**(0)
NDR1	65	**31** (4)	**62**(0)
NDR2	142	**86**(6)	**113**(0)
#total	428	**212**(21)	**290**(1)

Table 3: Comparing NuMDS with BIBLP and Guribi. #bet and #wor represent the number of instances where NuMDS achieve better and worse minimal solutions, respectively.

Benchmark	#ins	vs. NuMDS1 #bet(#wor)	vs. NuMDS2 #bet(#wor)	vs. NuMDS3 #bet(#wor)	vs. NuMDS4 #bet(#wor)	vs. NuMDS5 #bet(#wor)	vs. NuMDS6 #bet(#wor)	vs. NuMDS7 #bet(#wor)
Standard	168	0(0)	0(0)	0(0)	**6**(0)	0(0)	0(0)	**8**(0)
S&D	53	**20**(8)	**19**(2)	**13**(6)	**17**(5)	**19**(3)	**12**(6)	**13**(0)
NDR1	65	**25**(3)	**13**(3)	**15**(8)	**23**(5)	**18**(3)	**16**(8)	**21**(2)
NDR2	142	**80**(12)	**45**(12)	**44**(21)	**83**(3)	**64**(7)	**61**(7)	**66**(4)
#total	428	**125**(23)	**77**(17)	**72**(35)	**129**(13)	**101**(13)	**89**(21)	**108**(6)

Table 4: Comparing NuMDS with its seven alternative versions. #bet and #wor represent the number of instances where NuMDS achieves better and worse minimal solutions, respectively. Due to space limitations, SNAP&DIMACS10 is represented by S&D.

Benchmark	Infe_rules		RLNSV		RLNSV_imp		EMDSP_reduce		DPRM	
	fix_ratio	time	fix_ratio	time	fix_ratio	time	fix_ratio	time	fix_ratio	time
Standard	0.18%	<1	0.69%	<1	1.31%	<1	12.54%	<1	13.32%	<1
SNAP&DIMACS10	21.82%	<0.1	32.67%	1.04	47.81%	1.94	56.15%	20.03	68.47%	6.11
NDR1	29.07%	<0.1	36.75%	4.29	62.98%	18.3	67.94%	22.68	72.83%	30.27
NDR2	12.53%	<0.1	16.10%	8.77	27.36%	8.22	32.74%	278.10	53.41%	39.64

Table 5: The average fix ratio and time consumption of five reduction methods on each benchmark.

Benchmark	#ins	vs. FastDS #bet(#wor)	vs. DmDS bet(#wor)
Standard	168	0(0)	0(0)
SNAP&DIMACS10	53	**24**(6)	**19**(3)
NDR1	65	**25** (6)	**25**(6)
NDR2	142	**81**(9)	**61**(23)
#total	428	**130**(21)	**105**(32)

Table 6: Comparing FastDS$_{\text{DPRM}}$ and DmDS$_{\text{DPRM}}$ with FastDS and DmDS. #bet and #wor represent the number of instances where DmDS$_{\text{DPRM}}$ or FastDS$_{\text{DPRM}}$ achieves better and worse minimal solutions, respectively.

20 best solutions (out of 22 instances) on SNAP benchmark, 62 best solutions (out of 65 instances) on NDR1 benchmark, 130 best solutions (out of 142 instances) on NDR2 benchmark. Every competitor fails to obtain over 50% best solutions among 260 real-world instances. Moreover, when NuMDS discards its training set in the parameter tuning procedure, it achieves 371 best solutions out of 387 instances, while DmDS, FastDS, and DeepOpt-MWDS achieve 242, 236, and 217 best solutions, respectively.

Run time analysis

Then, we conduct a run time comparison between the NuMDS algorithm and three comparison algorithms, focusing on cases where both algorithms achieved the same best and average solution values for all instances. The instances where the run time of both NuMDS and the corresponding competitors is below 0.1 seconds are excluded from the analysis. As shown in Figure 3, the run time obtained by NuMDS is quite smaller than its competitors for the most instances.

Critical Difference Plots for All Algorithms

We use statistical analysis to evaluate the performance of the NuMDS algorithm and its competitors by presenting the difference of the best solutions obtained by each algorithm in Figure 4. First, we conduct the Friedman Test [Friedman, 1937] with the null hypothesis that the NuMDS algorithm performs equally to all the competitors. If the null hypothesis is rejected, we perform the Nemenyi post-hoc test for pairwise comparisons. The results are then presented using a critical difference diagram [Garcıa and Herrera, 2008]. A lower rank indicates better performance. If the difference between algorithms is not statistically significant at a significance level of 0.05, it indicates a connection between those algorithms. According to the critical difference plots, NuMDS consistently outperforms all the competitors.

6.2 Further Results with Exact Algorithms

In this part, we compare the NuMDS with the state-of-the-art exact algorithms. According to the literature [Jiang and Zheng, 2023; Xiong and Xiao, 2024], the state-of-the-art algorithm for the MDS is BIBLP [Xiong and Xiao, 2024]. So we compare NuMDS with this algorithm. Moreover, we also compare NuMDS with one of the most powerful commercial mixed integer programming (MIP) solvers Gurobi [Gurobi Optimization, 2021]. The version used is 10.0.3.

Note that the Gurobi is conducted on the instances reduced by DPRM. The time limit of the two exact algorithms are set to 1800 seconds, and we compare them with NuMDS under the cutoff time of 1800 seconds and seed 1. We summary the compared results in Table 3. As shown in Table 3, NuMDS clearly outperforms the two algorithms.

6.3 Analysis on the Proposed Strategies

First, we evaluate the proposed strategies by comparing NuMDS with its seven alternative versions: (1) NuMDS1 replaces the DPRM with the Infe-rules; (2) NuMDS2 substitutes the DPRM with the RLNSV_imp; (3) NuMDS3 substitutes the DPRM with the EMDSP_reduce; (4) NuMDS4 eliminates the multi-stage mechanism and instead adopts the same second vertex selection method as DmDS [Zhu *et al.*, 2024] and FastDS [Cai *et al.*, 2020]. Specifically, in each iteration of the NuMDS, the second removal vertex is selected

Benchmark	#ins	$Init_{tp}$ #win(#time)	$Init_{greedy}$ #win(#time)	$Init_{perturb}$ #win(#time)
realworld	260	**209**(**4.75**)	47(14.45)	50(16.18)

Table 7: Comparing $Init_{tp}$ with $Init_{greedy}$ and $Init_{perturb}$ on the reduced instances, where #win denotes the number of minimal results that each algorithm achieved, and #time is the average time consumption of each algorithm on the 260 real-world instances.

Benchmark	#ins	$Init_{tp}$ #win(#time)	$Init_{greedy}$ #win(#time)	$Init_{perturb}$ #win(#time)
realworld	260	**232**(**5.36**)	33(16.4)	29(16.21)

Table 8: Comparing $Init_{tp}$ with $Init_{greedy}$ and $Init_{perturb}$ on the original instances, where #win denotes the number of minimal results that each algorithm achieved, and #time is the average time consumption of each algorithm on the 260 real-world instances.

randomly. NuMDS5, NuMDS6, and NuMDS7, each removing one of the search modes: greedy search, random search, and exploitation search, respectively. Table 4 shows that both DPRM and the multi-stage mechanism are indispensable in the proposed NuMDS.

Then, we present the fix ratio and time consumption among DPRM, RLNSV [Alber *et al.*, 2004], RLNSV_imp [Alber *et al.*, 2006], the inference rules used in FastDS [Cai *et al.*, 2020] (referred to as Infe-rules), and the EMDSP_reduce in Table 5. We don't present the comparison result of RLNPV and RLNPV_imp due to their extremely long time consumption on real-world instances, making them unsuitable to be applied in local search algorithms. Specifically, both RLNPV and RLNPV_imp have a time consumption exceeding 1800 seconds on 87% real-world instances.

Table 5 shows that the fix ratio achieved by the DPRM reduction method is consistently better than that of other reduction techniques, except when compared to the EMDSP_reduce on standard instances. When compared to Infe_rules, RLNSV, and RLNSV_imp on real-world instances, DPRM significantly outperforms these algorithms in terms of fix ratio. Although the DPRM method has a longer time consumption than these algorithms, its run time remains within an acceptable range. Moreover, when we compare DPRM to EMDSP_reduce on real-world instances, DPRM also shows an improvement on all benchmarks. Overall, DPRM demonstrates robust performance in both fix ratio and time consumption.

Subsequently, we investigate the impact of DPRM on the local search performance. We incorporate DPRM into FastDS and DmDS, resulting in FastDS$_{\text{DPRM}}$ and DmDS$_{\text{DPRM}}$, and compare the best solutions obtained by these two algorithms with the best solution obtained by the original algorithms. The results are presented in Table 6. It is evident that DPRM greatly enhances the performance of FastDS and DmDS.

After that, we conduct experiments to evaluate the performance of two-phase initialization method. First, we compare $Init_{tp}$ with $Init_{greedy}$ and $Init_{perturb}$. Then, we don't use any reduction rules and directly evaluate the performance of the three initialization procedures. Note that the instances in the standard benchmark are significantly small, and the impact of the initial solution on the local search procedure can be omitted. Therefore, we only focus on the real-world benchmarks. As shown in Tables 7 and 8, $Init_{tp}$ outperforms previous initialization methods in terms of both initial solution quality and time consumption. It demonstrates the good performance regardless of whether the reduction process is employed. The reduced time consumption is a result of the vertex selection procedure in the priority phase, which only needs to transverse the neighbors of specific vertices, rather than traversing all vertices not in the candidate solution.

7 Conclusion

This paper proposes an efficient local search algorithm NuMDS for the MDS, which comprises three key ideas. First, a dominate propagation-based reduction method is presented to fix portion of vertices. Then, we develop an efficient two-phase initialization method to start the search procedure from a good point. Finally, a multi-stage local search procedure is proposed, which adapts some suitable search manners. Extensive experimental results clearly demonstrate the effectiveness of NuMDS.

Acknowledgments

This work is supported by National Cryptologic Science Fund of China 2025NCSF02046, NSFC under Grant No. 61806050, the Fundamental Research Funds for the Central Universities 2412023YQ003, Jilin Science and Technology Department YDZJ202201ZYTS412.

References

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

[Alber *et al.*, 2005] Jochen Alber, Hongbing Fan, Michael R. Fellows, Henning Fernau, Rolf Niedermeier, Fran Rosamond, and Ulrike Stege. A refined search tree technique for dominating set on planar graphs. *Journal of Computer and System Sciences*, 2005.

[Alber *et al.*, 2006] Jochen Alber, Nadja Betzler, and Rolf Niedermeier. Experiments on data reduction for optimal domination in networks. *Annals of Operations Research*, 146(1):105–117, 2006.

[Bouamama and Blum, 2015] Salim Bouamama and Christian Blum. A randomized population-based iterated greedy algorithm for the minimum weight dominating set problem. In *ICICS*, pages 7–12. IEEE, 2015.

[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 *IJCAI*, pages 1467–1473, 2020.

[Cai, 2015] Shaowei Cai. Balance between complexity and quality: Local search for minimum vertex cover in massive graphs. In Qiang Yang and Michael J. Wooldridge, editors, *IJCAI*, pages 747–753, 2015.

[Campan *et al.*, 2015] Alina Campan, Traian Marius Truta, and Matthew Beckerich. Fast dominating set algorithms for social networks. In *MAICS*, pages 55–62, 2015.

[Chalupa, 2017] David Chalupa. An order-based algorithm for minimum dominating set with application in graph mining. *Information Sciences*, 426:101–116, 2017.

[Chalupa, 2018] David Chalupa. An order-based algorithm for minimum dominating set with application in graph mining. *Information Sciences*, 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. *Applied Intelligence*, 43(3):512–529, 2015.

[Chen *et al.*, 2007] Jianer Chen, Henning Fernau, Iyad A Kanj, and Ge Xia. Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. *SIAM Journal on Computing*, 37(4):1077–1106, 2007.

[Chen *et al.*, 2023] Jiejiang Chen, Shaowei Cai, Yiyuan Wang, Wenhao Xu, Jia Ji, and Minghao Yin. Improved local search for the minimum weight dominating set problem in massive graphs by using a deep optimization mechanism. *Artificial Intelligence*, 314:103819, 2023.

[Dinur and Steurer, 2014] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *STOC*, pages 624–633, 2014.

[Downey and Fellows, 1995] Rod G Downey and Michael R Fellows. Fixed-parameter tractability and completeness i: Basic results. *SIAM Journal on computing*, 24(4):873–921, 1995.

[Drange *et al.*, 2014] Pål Grønås Drange, Markus S Dregi, Fedor V Fomin, Stephan Kreutzer, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, Felix Reidl, Saket Saurabh, Fernando Sánchez Villaamil, et al. Kernelization and sparseness: the case of dominating set. *arXiv preprint arXiv:1411.4575*, 2014.

[Eubank *et al.*, 2004] Stephen Eubank, VS Anil Kumar, Madhav V Marathe, Aravind Srinivasan, and Nan Wang. Structural and algorithmic aspects of massive social networks. In *ACM-SIAM*, pages 718–727, 2004.

[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 *DASFAA*, pages 211–228. Springer, 2019.

[Friedman, 1937] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[Garcıa and Herrera, 2008] Salvador Garcıa and Francisco Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[Giap and Ha, 2014] Cu Nguyen Giap and Dinh Thi Ha. Parallel genetic algorithm for minimum dominating set problem. In *ICAC*, 2014.

[Grandoni, 2004] Fabrizio Grandoni. *Exact Algorithms for Hard Graph Problems*. PhD thesis, Università di Roma 'Tor Vergata', Roma, Italy, 2004.

[Grandoni, 2006] Fabrizio Grandoni. A note on the complexity of minimum dominating set. *Journal of Discrete Algorithms*, 4(2):209–214, 2006.

[Guo and Niedermeier, 2007] Jiong Guo and Rolf Niedermeier. Linear problem kernels for np-hard problems on planar graphs. In *ICALP*, pages 375–386. Springer, 2007.

[Gurobi Optimization, 2021] LLC Gurobi Optimization. *Gurobi optimizer reference manual*. 2021.

[Hedar and Ismail, 2010a] Abdel-Rahman Hedar and Rashad Ismail. Hybrid genetic algorithm for minimum dominating set problem. In *ICCSA*, pages 457–467. Springer, 2010.

[Hedar and Ismail, 2010b] Abdel-Rahman Hedar and Rashad Ismail. Hybrid genetic algorithm for minimum dominating set problem. In *ICCSA*, pages 457–467. Springer, 2010.

[Hedar and Ismail, 2012] Abdel-Rahman Hedar and Rashad Ismail. Simulated annealing with stochastic local search for minimum dominating set problem. *International Journal of Machine Learning and Cybernetics*, 3(2):97–109, 2012.

[Ho *et al.*, 2006] Chin Kuan Ho, Yashwant Prasad Singh, and Hong Tat Ewe. An enhanced ant colony optimization metaheuristic for the minimum dominating set problem. *Applied Artificial Intelligence*, 20(10):881–903, 2006.

[Jiang and Zheng, 2023] Hua Jiang and Zhifei Zheng. An exact algorithm for the minimum dominating set problem. In *IJCAI*, pages 5604–5612, 2023.

[Jovanovic *et al.*, 2010] Raka Jovanovic, Milan Tuba, and Dana Simian. Ant colony optimization applied to minimum weight dominating set problem. In *ICCAS*, 2010.

[Kitsak *et al.*, 2010] Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. Identification of influential spreaders in complex networks. *Nature physics*, 6(11):888–893, 2010.

[Liu *et al.*, 2011] Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Controllability of complex networks. *nature*, 473(7346):167, 2011.

[López-Ibáñez *et al.*, 2016] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[Okumuş and Karcı, 2024] Fatih Okumuş and Şeyda Karcı. Mdsa: A dynamic and greedy approach to solve the minimum dominating set problem. *Applied Sciences*, 14(20):9251, 2024.

[Potluri and Bhagvati, 2012] Anupama Potluri and Chakravarthy Bhagvati. *Novel Morphological Algorithms for Dominating Sets on Graphs with Applications to Image Analysis*. 2012.

[Potluri and Singh, 2011] Anupama Potluri and Alok Singh. Two hybrid meta-heuristic approaches for minimum dominating set problem. In *SEMCCO*, pages 97–104. Springer, 2011.

[Potluri and Singh, 2013] Anupama Potluri and Alok Singh. Hybrid metaheuristic algorithms for minimum weight dominating set. *Applied Soft Computing Journal*, 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 *UIST*, 1997.

[Reixach and Blum, 2024] Jaume Reixach and Christian Blum. Extending cmsa with reinforcement learning: Application to minimum dominating set. In *MIC*, pages 354–359. Springer, 2024.

[Romania, 2010] Qatar Serbia Romania. Ant colony optimization applied to minimum weight dominating set problem. In *WSEAS*, pages 29–31, 2010.

[Rossi and Ahmed, 2015] Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, volume 29, 2015.

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

[Shen and Li, 2010] Chao Shen and Tao Li. Multi-document summarization via the minimum dominating set. In *COLING*, pages 984–992. Coling, 2010.

[Stefan, 2014] Wuchty Stefan. Controllability in protein interaction networks. *Proc Natl Acad Sci U S A*, 111(19):7156–7160, 2014.

[Stojmenovic *et al.*, 2002] Ivan Stojmenovic, Mahtab Seddigh, and Jovisa Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on parallel and distributed systems*, 13(1):14–25, 2002.

[Takaguchi *et al.*, 2014] Taro Takaguchi, Takehisa Hasegawa, and Yuichi Yoshida. Suppressing epidemics on networks by exploiting observer nodes. *Physical Review E*, 90(1):012807, 2014.

[Van Rooij and Bodlaender, 2011] Johan MM Van Rooij and Hans L Bodlaender. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 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. *Journal of Artificial Intelligence Research*, 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 *IJCAI*, pages 1514–1522, 2018.

[Wang *et al.*, 2020] Yiyuan Wang, Shaowei Cai, Jiejiang Chen, and Minghao Yin. Sccwalk: An efficient local search algorithm and its improvements for maximum weight clique problem. *Artificial Intelligence*, 280:103230, 2020.

[Wu, 2002] Jie Wu. Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):866–881, 2002.

[Xiong and Xiao, 2024] Ziliang Xiong and Mingyu Xiao. Exactly solving minimum dominating set and its generalization. In *IJCAI*, 2024.

[Xu *et al.*, 2007] Ke Xu, Frédéric Boussemart, Fred Hemery, and Christophe Lecoutre. Random constraint satisfaction: Easy generation of hard (satisfiable) instances. *Artificial intelligence*, 171(8-9):514–534, 2007.

[Yao and Li, 2012] Bangpeng Yao and Fei Fei Li. Action recognition with exemplar based 2.5d graph matching. In *ECCV*, 2012.

[Zhu *et al.*, 2024] Enqiang Zhu, Yu Zhang, Shengzhi Wang, Darren Strash, and Chanjuan Liu. A dual-mode local search algorithm for solving the minimum dominating set problem. *Knowledge-Based Systems*, page 111950, 2024.