Knowledge-Based Systems 298 (2024) 111950

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A dual-mode local search algorithm for solving the minimum dominating set problem

Enqiang Zhu [a,1], Yu Zhang [b,1], Shengzhi Wang [a], Darren Strash [c], Chanjuan Liu [d,*]

[a] Institute of Computing Science and Technology, Guangzhou University, Guangzhou, China
[b] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China
[c] Department of Computer Science, Hamilton College, Clinton, USA
[d] School of Computer Science and Technology, Dalian University of Technology, Dalian, China

ARTICLE INFO

Keywords:
Minimum dominating set
Heuristics
Local search
Dual-mode
Perturbation
Greedy strategy
Vertex selection

ABSTRACT

Given a graph $G = (V, E)$, the minimum dominating set (MinDS) problem is to identify a smallest set of vertices D such that every vertex in $V \setminus D$ is adjacent to at least one vertex in D. The MinDS problem is a classic $\mathcal{NP}$-hard problem and has been extensively studied because of its many disparate applications in network analysis. To solve this problem in practice, many heuristic approaches have been proposed to obtain a high-quality solution within a given time limit. However, existing heuristic algorithms are limited by various tie-breaking cases when selecting vertices, which slows down the effectiveness of the algorithms. In this paper, we design an efficient local search algorithm for MinDS, named DmDS – a dual-mode local search framework that probabilistically chooses between two distinct vertex-swapping schemes. We further address limitations of other algorithms by introducing vertex selection criterion based on the frequency of vertices added to solutions to address tie-breaking cases, and by improving the quality of the initial solution via a greedy strategy with perturbation. We evaluate DmDS against the state-of-the-art algorithms on real-world datasets, consisting of 382 instances (or families) with up to tens of millions of vertices. Experimental results show that DmDS obtains the best performance in accuracy for almost all instances and finds significantly better solutions than state-of-the-art MinDS algorithms on a broad range of large real-world graphs; specifically, DmDS computes the smallest solution on 352 (out of 382) instances, and on 119 instances DmDS finds smaller solutions than all other algorithms in our comparison.

1. Introduction

A dominating set (DS) – a set of vertices D in which each vertex of the graph is in D or adjacent to at least one vertex in D – is an important structure in graph theory. DSs have applications spanning the sciences, including social networks [1], epidemic control [2], and biological network analysis [3,4]. One key example problem is in protein–protein interaction networks: to identify a smallest subset D of proteins where each protein outside D can be reached by interacting with a protein within D. This problem is equivalent to finding a dominating set of minimum cardinality — a minimum dominating set (MinDS) [4].

Given its many applications, the MinDS problem has attracted considerable interest in recent years. Due to its $\mathcal{NP}$-hardness [5], significant effort has been put into improved exponential-time algorithms. The first improvement to the standard brute-force $O^*(2^n)$-time algorithm for MinDS was introduced by Fomin et al. [6] and, since then, many improved algorithms have been proposed [7–10]. The current state-of-the-art exact MinDS algorithm, with $O(1.4864^n)$ time and has polynomial space [11], is based on a variation of Grandoni's algorithm [12]. Researchers have also developed increasingly sophisticated heuristic algorithms for finding a close-to-minimum DS. Greedy approaches provide a simple and efficient way for constructing DSs, but such algorithms often perform poorly, especially on large instances [13]. The highest-quality heuristic algorithms first apply data reduction rules to decrease the input size [13], then generate an initial solution with a greedy approach, and finally adopt heuristics to improve solution quality [14–16].

* Corresponding author.
E-mail addresses: zhuenqiang@gzhu.edu.cn (E. Zhu), zhangyu@e.gzhu.edu.cn (Y. Zhang), wangshengzhi@e.gzhu.edu.cn (S. Wang), dstrash@hamilton.edu (D. Strash), chanjuanliu@dlut.edu.cn (C. Liu).
[1] The first two authors contributed equally to this work.

https://doi.org/10.1016/j.knosys.2024.111950
Received 7 August 2023; Received in revised form 3 March 2024; Accepted 12 May 2024
Available online 23 May 2024

1.1. Related work

Many recent studies show the significance of heuristic approaches to solve intractable problems, such as partition coloring [17], critical node [18], weighted total domination [19], and minimum k-dominating set [20]. Here we give a brief review of the heuristic algorithms for solving the MinDS problem.

Heuristics, including ant-colony optimization [21], genetic algorithms [22,23], and local search algorithms [13,24,25] have been proposed for the MinDS problem. In particular, local search approaches have attracted much attention in recent years. Vertex swapping is an important component in local search for graph problems. If vertex swapping depends heavily on greedy methods, then the algorithms will repeatedly visit candidate solutions that have been encountered recently and result in a local optimum. This phenomenon is called the cycling problem, which is an issue inherent in local search algorithms. To address this problem, Cai et al. [26] proposed a configuration checking (CC) strategy, adopting Boolean array markers to prevent recently visited vertices from being revisited, where the restriction of a vertex v is relaxed if the state of a vertex adjacent to v is changed. In 2017, an algorithm named CC^{2}FS based on a variant of CC (CC^2) was proposed to solve the minimum weight dominating set (MinWDS) problem [27], in which CC was extended to *two-level* mode: a vertex v is reset to an allowed state if a vertex within distance two from v is removed from (or added to) the solution. Subsequently, Wang et al. [28] introduced another variant CC^{2}V3 of CC that adds a new state to CC^2 for the purpose of further distinguishing the priority of visitable vertices and showed that CC^{2}V3 can enhance the performance of CC^{2}FS on large sparse real-world graphs. Based on CC^{2}V3, an algorithm called FastMWDS was proposed to identify a MinWDS. In 2019, by combining score checking (SC) with a probabilistic walk, Fan et al. [25] proposed a local search algorithm named ScBppw for the MinDS problem on large graphs, which outperforms FastMWDS on many large graphs. In the same year, Cai et al. [13] introduced a two-goal local search framework for the MinDS problem and developed a MinDS algorithm named FastDS based on three data reduction rules (called inference rules in their article). In contrast to the classic local search MinDS framework, FastDS aims to improve a given k-sized feasible solution to a $(k-1)$- or $(k-2)$-sized feasible solution each time, which accelerates the convergence speed and expands the search region. Recently, Jiang et al. [29] proposed an independent-set-based method to calculate a lower bound of the domination number and an exact algorithm named EMOS to find a MinDS. The EMOS algorithm achieved excellent performance on standard benchmarks UDG and T1. To date, the state-of-the-art algorithms for MinDS are ScBppw, FastDS, and EMOS.

1.2. Motivation and our contributions

1.2.1. Motivation

Note that the ScBppw algorithm can obtain small solutions on large graphs, but it does not perform as well as FastDS on medium-sized graphs. One reason is that the SC strategy consumes a lot of time during local search. In addition, the SC strategy is stronger than the CC^2 strategy [25], which results in more vertices being forbidden and limits the algorithm's search space. Regarding FastDS, we observe that there are many vertices with equal priority when selecting vertices, even though the algorithm uses two criteria to address the ties. Moreover, the initial solution in FastDS is constructed by a greedy strategy with non-dominated degrees, which can be improved.

1.2.2. Contributions

This paper focuses on designing an efficient heuristic algorithm to find a high-quality (small) dominating set in a given graph. We propose a new local search algorithm for the MinDS problem named DmDS, which includes three main ideas: a novel local search framework, a new approach for constructing an initial solution, and a new criterion for selecting vertices. First, to prevent cycling, we propose a dual-mode local search framework in the search phase, which allows the algorithm to run in two modes: when finding a k-sized dominating set, DmDS searches for a solution with size $(k-1)$ by either (2, 1)-swaps or (3, 2)-swaps, where a $(j, j-1)$-swap removes j vertices from the solution and replaces them with $j-1$ vertices [30]. Second, a high-quality initial solution can accelerate the convergence speed of local search algorithms. The common methods for generating an initial DS are greedy-based strategies [13] and random-based strategies [31], because these strategies are simple and efficient. However, the quality of solutions generated by these strategies is often poor and the time complexity would be increased significantly when adding complicated strategies to improve solutions' quality. Based on the above considerations, DmDS uses two approaches to generate solutions and chooses the better one as the initial solution, including a greedy-based algorithm according to dominating profits and an approach integrating the greedy strategy with a perturbation mechanism. Third, in the process of swapping vertices, selecting vertices according to a single criterion would suffer from many tie-breaking cases, i.e., there are multiple vertices with the best value. Previous work used "age" to address this problem. However, on large sparse real-world graphs, the improvement is not obvious and there are still multiple best vertices that can be selected at the same time. Therefore, we propose a new vertex selection criterion based on the number of times each vertex has been added to the solution.

1.3. Organization of this paper

The remainder of this paper is organized as follows. Section 2 introduces notation and terminology, as well as a brief review of local search and related techniques. Section 3 describes the DmDS algorithm. Section 4 is devoted to the design and analysis of experiments, and Section 5 provides concluding remarks.

2. Preliminaries

We first introduce some necessary notations and terminologies, followed by a brief review of local search.

2.1. Notation and terminology

All graphs considered in this paper are unweighted, undirected, and simple. We use $G = (V, E)$ to denote a graph with *vertex set* V and *edge set* E. A vertex v is *adjacent* to (or a *neighbor* of) another vertex u in G if they are connected by an edge, i.e., $uv \in E$. For a vertex $v \in V$, the *degree* of v, denoted by $d(v)$, is the number of edges incident with v; the *neighborhood* of v, denoted by $N(v)$, is the set of neighbors of v, i.e., $N(v) = \{u \mid u \in V \text{ and } uv \in E\}$; the *closed neighborhood* of v, denoted by $N[v]$, is the union of $N(v)$ and $\{v\}$, i.e., $N[v] = N(v) \cup \{v\}$; and the *two-level closed neighborhood* of v, denoted by $N^2[v]$, is the set of vertices that have distance at most two from v, i.e., $N^2[v] = N[v] \cup (\bigcup_{u \in N(v)} N(u))$. Moreover, for a subset $S \subseteq V$, let $N(S) = \bigcup_{v \in S}(N(v) \setminus S)$ and $N[S] = N(S) \cup S$.

Definition 1. In a graph $G = (V, E)$, a vertex u is dominated by a vertex v if $v \in N[u]$. A dominating set of G is a subset $D \subseteq V$ such that every vertex in V is dominated by a vertex in D, i.e., $N[D] = V$. A dominating set of minimum cardinality is called a minimum dominating set (MinDS). The MinDS problem is to find a MinDS in G.

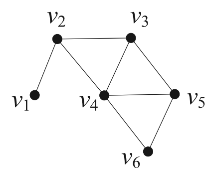

Fig. 1. An example graph where $\{v_1, v_4\}$ is a MinDS.

An example illustrating the MinDS problem is given in Fig. 1. Observe that this graph has six vertices and has maximum degree four, which implies that a MinDS contains at least two vertices. On the other hand, $N[\{v_1, v_4\}] = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. Therefore, $\{v_1, v_4\}$ is a MinDS. Note that there are other MinDSs, such as $\{v_2, v_5\}$, $\{v_2, v_6\}$, and $\{v_2, v_4\}$. Thus, a graph may have more than one MinDS.

2.2. Local search

The principle of local search can be described as follows: Start with an initial solution and then iteratively improve it by removing vertices, adding vertices, and/or swapping vertices. Local search often incorporates some effective techniques to enhance its search results, such as tabu strategies, data reduction, and heuristic vertex selection strategies. For example, CC [26], CC^2 [27], CC^{2}V3 [28], and SC [25] are tabu strategies that are designed to avoid wasting time in low-quality search space and address the cycling problem by forbidding vertices from being added to the solution. The aim of designing reduction rules is to simplify instances, with the advantage of reducing the search space by determining a part of vertices that definitely belong (or not belong) to an optimal solution. Additionally, in each iteration of local search, the greedy-based strategies for selecting vertices are time-consuming and easily result in a local optimum. For this, Cai et al. [32] proposed a heuristic vertex selection strategy named Best from Multiple Selections (BMS), which selects the best vertex from t random vertices (t is a predefined parameter, in general setting to 50) instead of traversing all vertices. The BMS strategy has been used to improve the efficiency of local search algorithms for solving the MinDS problem [13,28].

3. Main algorithms

We first present the top-level local search framework (Algorithm 1), followed by the construction of initial solutions (Algorithm 2). Finally, we give the local search algorithm DmDS (Algorithm 3) with a time complexity analysis.

For presentation purposes, we specify some notation used in this section. Given a graph G, let $D \subseteq V$ and $X \subseteq V$ be two disjoint sets, where D is a partial dominating set of G and X is a set of vertices forbidden from being added to D. We use a Boolean function $dom_D : V \to \{1, 0\}$ to characterize whether a vertex is dominated by a vertex in D, which is defined as follows: for every vertex $v \in V$, if $v \in N[D]$, then $dom_D(v) = 1$; otherwise, $dom_D(v) = 0$. Denote by UD the set of vertices on which dom_D evaluates to 0, i.e., $UD = \{v \mid v \in V$ and $dom_D(v) = 0\}$. For a vertex $u \in D$, denote by $loss_D(u)$ the number of vertices v such that $dom_D(v) = 1$ and $dom_{D\setminus\{u\}}(v) = 0$; and for a vertex $w \notin D$, denote by $gain_D(w)$ the number of vertices v such that $dom_D(v) = 0$ and $dom_{D\cup\{w\}}(v) = 1$. Moreover, we use the $age(v)$ to denote the number of iterations that the status of v has not changed, and $freq(v)$ to denote the number of times v has been added to the solution. The large age of a vertex implies that the vertex has been in (or not in) a solution for a long time, and in this case, the vertex should have a priority to be removed from (or added to) a solution. In addition, if a vertex has a smaller $freq$ value, then it has been added to a solution fewer times, and it should have a priority to be added to a solution.

Algorithm 1: Local search framework for MinDS

```
   Input: A graph G = (V, E), the cutoff time
   Output: A dominating set D* of G
1  D ← Initialization(G);
2  while elapsed_time < cutoff do
3      if D is feasible then
4          if D contains redundant vertices then
5              remove all redundant vertices from D;
6          D* ← D;
7          remove a vertex with minimum loss from D;
8          continue;
9      if with probability α then
10         search by (1, 1)-swap;
11     else
12         search by (2, 2)-swap;
13 return D*;
```

3.1. Local search framework

The details of the local search framework are shown in Algorithm 1. First, an initial solution D is constructed by a procedure *Initialization()* (line 1), where some data reduction techniques are adopted to reduce the search space. The next is the main part of the framework (lines 2–12), which tries iteratively reducing the size of a feasible solution within a given period of time. Specifically, each time a feasible solution D is obtained, the algorithm checks and removes all redundant vertices from D (lines 3–5), updates the best solution D^* (line 6), and removes a vertex with the minimum $loss$ value from D (line 7), where a vertex $v \in D$ is redundant if $loss_D(v) = 0$ and $D \setminus \{v\}$ is a feasible solution. The algorithm attempts to make an infeasible solution with size $|D| - 1$ feasible. When the current solution D is not feasible, two vertex-swapping modes (1, 1)-swap and (2, 2)-swap are implemented with probabilities α (lines 9–10) and $1 - \alpha$ (lines 11–12), respectively, to exchange vertices until D becomes feasible, where a (k, k)-swap removes k vertices from D and adds k new vertices to D for $k = 1, 2$. It should be noted that within each iteration, the algorithm first removes one vertex (line 7) and then carries out a (1, 1)-swap or (2, 2)-swap (lines 9–12); therefore the algorithm effectively performs a (2, 1)-swap or a (3, 2)-swap. Finally, the best solution D^* is returned when the time limit is reached (line 13).

3.2. Generating an initial solution

The graph reduction strategy is a key mechanism for designing effective heuristic algorithms for intractable graph problems on large sparse graphs [14]. To generate a high-quality initial solution for DmDS, we first use three simple graph reduction rules, proposed by Cai et al. [13], to reduce the search space (Algorithm 2; lines 1–4). The sets D and X in the following three reduction rules are the same as described above.

Reduction Rule 1 (*Degree 0*)**.** *Let v be a vertex with degree zero. If $dom_D(v) = 0$, then add v to D.*

Reduction Rule 2 (*Degree 1*)**.** *Let v be a vertex of degree one and u the neighbor of v. If $dom_D(u) = 0$, then add u to D and v to X.*

Reduction Rule 3 (*Triangle*)**.** *Let u, v, w be a triangle such that $d(u) = d(v) = 2$. If $dom_D(w) = 0$, then add w to D and u and v to X.*

As shown in Algorithm 2, the sets D^* and X (of vertices in an optimal solution and excluded from that optimal solution, respectively) can be obtained by applying Reduction Rules 1–3. We invite the

Algorithm 2: $Initialization(G)$

Input: A graph $G = (V, E)$
Output: A dominating set D of G
1 **foreach** u *in* V **do**
2     $D^*, X \leftarrow$ apply **Reduction Rule 1** on u;
3     $D^*, X \leftarrow$ apply **Reduction Rule 2** on u;
4     $D^*, X \leftarrow$ apply **Reduction Rule 3** on u;
5 $V \leftarrow V \setminus X, D \leftarrow D^*, D' \leftarrow D^*$;
6 $D \leftarrow$ GreedyConstructDS(G, D);
7 $D' \leftarrow$ PerturbedConstructDS(G, D');
8 **if** $|D'| < |D|$ **then**
9     $D \leftarrow D'$;
10 **return** D;

Procedure GreedyConstructDS(G, D);
11 **while** $UD \neq \emptyset$ **do**
12     $v \leftarrow$ a vertex with maximum $gain_D$ in $V \setminus D$;
13     $D \leftarrow D \cup \{v\}$;
14 remove all redundant vertices from D;
15 **return** D;

Procedure PerturbedConstructDS(G, D');
16 **while** $UD \neq \emptyset$ **do**
17     $v \leftarrow$ a vertex with maximum $gain_{D'}$ in $V \setminus D'$;
18   save $gain_{D'}(v)$, $D' \leftarrow D' \cup \{v\}$;
19     $u \leftarrow$ a vertex with minimum $loss_{D'}$ in $D' \setminus D^*$;
20   **if** $loss_{D'}(u) < gain_{D'}(v)$ **then**
21         $D' \leftarrow D' \setminus \{u\}$;
22 remove all redundant vertices from D';
23 **return** D';

interested reader to refer to [13] for the proofs of correctness. Next, after initializing three sets V, D, D' (line 5), two distinct strategies GreedyConstructDS(G, D) and PerturbedConstructDS(G, D') are used to construct two initial solutions D and D', respectively (lines 6–7), and the best one is selected as the final initial solution (lines 8–9). GreedyConstructDS(G, D) iteratively adds a vertex v with the maximum $gain_D(v)$ to D until D becomes a feasible solution (lines 11–13), and finally removes redundant vertices from D (line 14). PerturbedConstructDS(G, D') first selects a vertex v with the maximum $gain_{D'}(v)$, saves the value of $gain_{D'}(v)$, and updates D' by adding v to D' (lines 17–18); next, if there is a vertex u with minimum loss in D' such that $loss_{D'}(u) < gain_{D'}(v)$, then v is removed from D' (lines 19–21); the algorithm repeatedly conducts the above procedure until D' becomes a feasible solution; finally, the algorithm removes all redundant vertices from D' (line 22). Note that these two initial solution strategies are independent of each other, we merely use them each to produce a better overall initial solution.

3.3. The DmDS algorithm

We now describe the DmDS algorithm, which is based on the proposed dual-mode local search framework, using Algorithm 2 to improve the quality of initial solutions and a novel vertex selection criterion $freq$ to break ties. Moreover, to optimize the quality of solutions and explore the search space, DmDS introduces dual-mode search, i.e., performing either $(2,1)$-swaps and $(3,2)$-swaps.

The pseudocode of DmDS is presented in Algorithm 3. The algorithm starts with an initial solution D obtained by Algorithm 2 (line 1), and then improves D iteratively (lines 2–17). In each iteration, if D is a feasible solution, the algorithm first removes the redundant vertices from D (lines 3–6), and then removes a vertex v with minimum $loss_D(v)$ from D (line 7). Next, the algorithm conducts a tiny perturbation by randomly removing a vertex u_1 from D, records the number *rm_num* of removed vertices in each iteration (lines 8–9), removes another vertex u_2 according to the BMS heuristic with a probability α, and updates *rm_num* (lines 10–12). Note that if u_2 is removed, then a $(2,2)$-swap (lines 8–17) is implemented; otherwise, a $(1,1)$-swap (lines 8–9, 13–14) is implemented. After removing vertices from D, it chooses vertices from $N[UD]$ and adds them to D to obtain a feasible solution D. More specifically, the algorithm first selects a vertex $w_1 \in N[UD]$ with the greatest $gain_D(w_1)$ and adds it to D (lines 13–14); if three vertices are removed in this iteration, the algorithm further adds a vertex w_2 (selected by the same criteria as w_1) to D (lines 16–17). Finally, the best solution found D^* is returned when the cutoff time is reached. Note that there are many tie-breaking cases when selecting a vertex to be removed from D or to be added to D. The algorithm breaks the ties by age and $freq$. Specifically, when removing a vertex from D, the algorithm selects vertices with largest age value (breaking ties by largest $freq$ value if more than one vertex has the same largest age); when adding a vertex to D, the algorithm selects vertices with largest age value (breaking ties by selecting the vertex with smallest $freq$ value if more than one vertex has the same largest age).

Note that for a vertex v, $gain_D(v) = 0$ when $v \in D$ and $loss_D(v) = 0$ when $v \notin D$, and the $loss$ and $gain$ values of a vertex in $N^2[v]$ may change when v is removed from or added to the current solution D. We use the example in Fig. 1 to illustrate this procedure. Suppose that $D = \{v_1, v_3, v_6\}$; we have $gain_D(v_i) = 0$ for $i = 1, 2, \ldots, 6$, $loss_D(v_i) = 1$ for $i = 1, 3, 6$ and $loss_D(v_i) = 0$ for $i = 2, 4, 5$. Now, we remove v_3 from D. Then, v_2, v_4, and v_5 are dominated by only one vertex in D and v_3 is not dominated by any vertex in D. Therefore, $gain_D(v_i) = 0$ for $i = 1, 6$ and $gain_D(v_i) = 1$ for $i = 2, 3, 4, 5$, and $loss_D(v_1) = 2$, $loss_D(v_6) = 3$, and $loss_D(v_i) = 0$ for $i = 2, 3, 4, 5$. Here, the $gain$ (and $loss$) value for vertices v_2, v_3, v_4 and v_5 (v_1, v_3, and v_6) have changed when v_3 is removed from D.

Algorithm 3: DmDS

Input: A graph $G = (V, E)$, the *cutoff* time
Output: A dominating set D^* of G
1 $D \leftarrow Initialization(G)$;
2 **while** *elapsed_time* < *cutoff* **do**
3   **if** *D is feasible* **then**
4   **while** *there exists a vertex* v *with* $loss_D(v) = 0$ **do**
5   remove v from D;
6         $D^* \leftarrow D$;
7   remove a vertex with minimum $loss_D$ from D, breaking ties by age and $freq$;
8     $u_1 \leftarrow$ a random vertex in D;
9     $D \leftarrow D \setminus \{u_1\}$, *rm_num* $\leftarrow 2$;
    // (3, 2)-swap, else (2,1)-swap
10   **if** *with probability* α **then**
11         $u_2 \leftarrow$ a vertex in D selected by BMS heuristic, breaking ties by age and $freq$;
12         $D \leftarrow D \setminus \{u_2\}$, *rm_num* $\leftarrow 3$;
13     $w_1 \leftarrow$ a vertex in $N[UD]$ with the greatest $gain_D$, breaking ties by age and $freq$;
14     $D \leftarrow D \cup \{w_1\}$;
15   **if** $UD \neq \emptyset$ *and* *rm_num* $= 3$ **then**
16         $w_2 \leftarrow$ a vertex in $N[UD]$ with the greatest $gain_D$, breaking ties by age and $freq$;
17         $D \leftarrow D \cup \{w_2\}$;
18 **return** D^*;

3.4. Complexity analysis

In this section, we analyze the time complexity of DmDS. Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, we use an adjacency list to represent G.

Lemma 1. *Reduction Rules 1–3 run in* $O(n)$ *time.*

Proof. The procedure traverses all vertices once to verify the conditions of the rules. The time complexity for determining whether a vertex v satisfies the conditions of Reduction Rules 1 and 2 is $O(1)$. For Reduction Rule 3, computing the common neighbors of two neighboring vertices u and v each with degree 2 takes $O(1)$ time. Furthermore, the operation of adding a vertex to a set takes $O(1)$ time. Therefore, the overall time complexity of the procedure is $O(n)$.

Lemma 2. *Algorithm 2 runs in $O(n + m \log n)$ time.*

Proof. We utilize two heaps for the $gain$ and $loss$, respectively, to accelerate the initial solution generation, which can be constructed in $O(n)$ time. Note that selecting vertices with the maximum $gain$ and $loss$ values takes $O(1)$ time. Updating $gain$ and $loss$ values when adding a vertex to D or removing a vertex v from D takes at worst $O(d(v))$ time, and adjusting the heaps takes $O(d(v) \log n)$ time. Adjusting heaps for every vertex v is $O(m \log n)$ time. In addition, Reduction Rules 1–3 run in $O(n)$ time by Lemma 1, and deleting redundant vertices can be done in $O(n)$ time. Therefore, the time complexity of Algorithm 2 is $O(n + m \log n)$.

Theorem 1. *DmDS (Algorithm 3) runs in $O(tn + m \log n)$ time, where t is the number of iterations of the while loop on lines 2–17.*

Proof. First, by Lemma 2, initial solution generation takes $O(n+m \log n)$ time. Second, removing redundant vertices takes $O(n)$ time. Third, removing a vertex with a minimum $loss$ from D requires traversing D, taking $O(n)$ time; removing a vertex randomly from D, implementing the BMS heuristic are $O(1)$, and updating the gain and loss values takes $O(\Delta)$ time. Note that each time a vertex is added to D, it has to examine $|UD| \leq n$ vertices, implying the adding operation takes $O(n)$ time. In addition, in one round of iteration, at most three vertices are removed from D and at most two vertices are added to D. Therefore, the time complexity of each iteration is $O(n)$, and DmDS runs in $O(tn + m \log n)$.

4. Experiments

We conduct experiments on an extensive collection of benchmark instances to evaluate the performance of DmDS and existing state-of-the-art MinDS algorithms. We first introduce the seven benchmark datasets that we use, followed by a description of the experimental setup. Then we report the experimental results, including the size of the dominating sets obtained by the algorithms under identical conditions, along with an assessment of their time to converge to a final solution. Finally, an empirical evaluation of the parameters in the DmDS algorithm is provided, followed by a detailed discussion of the experimental results.

4.1. Benchmark datasets

We evaluate DmDS on seven benchmark datasets, which include four standard benchmarks for MinDS and three benchmarks of large sparse real-world graphs. All instances are processed as undirected unweighted simple graphs.

UDG[2]*:* This dataset is widely used in the literature on the MinDS problem [13,21,22,24,31,33]. Graphs in this dataset simulate wireless sensor networks, in which each vertex is a center of a unit disk, with edges between vertices whose corresponding disks overlap. This dataset contains 12 families, and each family contains 10 instances that are unit disk graphs with 50 to 1000 vertices.

T1/T2[2]*:* These two datasets consist of 53 families with a total of 530 instances, where each family contains 10 instances of the same size (from 50 to 1000).

BHOSLIB[3]*:* This dataset was translated from hard random SAT instances, which have been widely used to test heuristic algorithms [34]. Instances in this dataset have between 760 and 4000 vertices.

SNAP[4]*:* This dataset is from the Stanford Large Network Dataset Collection and was used by Cai et al. [13] to test MinDS algorithms. We collect all instances they used in the dataset to evaluate DmDS; the instances have between 7115 and 3,774,768 vertices.

DIMACS10[5]*:* These are synthetic and real-world instances from the 10th DIMACS implementation challenge on graph partitioning and graph clustering [35]. These instances are sparse graphs with between 22,963 and 18,483,186 vertices.

Network Repository[6]*:* This dataset is an interactive scientific network data repository [36], which contains a large number of graphs of various categories and has been widely used for different graph problems [13,14,32]. The instances we consider have between 70 and 58,790,782 vertices.

4.2. Experimental setup

All algorithms have been implemented in C++ and were compiled using gcc 7.1.0 using the '-O3' optimization flag. All experiments are run under CentOS Linux 7.6.1810 with an Intel(R) Xeon(R) Gold 6254 CPU running at 3.10 GHz and 128 GB RAM. The parameters of FastDS are set to the same as those used by its original authors [13], and the parameter α of DmDS is set to 0.5. We experiment with different values of α in Section 4.3.3. Regarding the parameter t in the BMS heuristic, we follow the authors' suggestion (near 50) and set t randomly in the range of 45 to 55.

We compare DmDS against two state-of-the-art heuristic MinDS algorithms, ScBppw [25], FastDS [13], and one exact MinDS algorithm, EMOS [29]. ScBppw is designed to solve large graphs for MinDS, which combines a tabu strategy called score checking with best-picking with probabilistic walk and can obtain smaller solutions than FastMWDS (proposed by Wang et al. [28] in 2018) on many large graphs. FastDS uses a two-goal local search framework together with inference rules that reduce the input graph, and is the best-performing heuristic algorithm at present. Both of the above algorithms are local search algorithms. EMOS introduces an independent set (IS)-based approach to determine a lower bound on the domination number, which is used for pruning during branch-and-bound. The code for ScBppw[7] and EMOS[8] is available online, and the code of FastDS was kindly provided by its authors.

We run ScBppw, FastDS, and DmDS 10 times on each instance with random seeds 1, 2, 3, …, 10; EMOS is run one time on each instance; each run is made on its own dedicated core of the CPU. The cutoff time is set to 1000 s. For each instance (excluding UDG and T1/T2 benchmark datasets), we report the best size (Min.) and the average size (Avg.) of the solutions found by ScBppw, FastDS, and DmDS over the 10 runs, and report only the best size (Min.) of the solutions found by EMOS within the time limit.[9] For the UDG and T1/T2 benchmarks, we follow the method by Cai et al. [13] and report the average $\overline{D_{min}}$ of the smallest solutions over all instances in a family.

2 https://github.com/yiyuanwang1988/MDSClassicalBenchmarks

3 http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm

4 https://snap.stanford.edu/data/

5 https://www.cc.gatech.edu/dimacs10/downloads.shtml

6 https://networkrepository.com

7 https://github.com/Fan-Yi

8 https://github.com/huajiang-ynu/ijcai23-mds/

9 We briefly note that because EMOS is an exact algorithm, in most cases it does not finish within the 1000-second time limit.

Table 1
Summarizing the number of instances where an algorithm's solution is smaller than the solution of all other algorithms.

Benchmark	DmDS	FastDS	ScBppw	EMOS
UDG	**1**	0	0	0
T1/T2	**7**	1	0	2
BHOSLIB	0	**1**	0	0
SNAP	**5**	3	0	0
DIMACS10	**12**	5	0	0
Network Repository	**94**	16	2	0
Total	**119**	26	2	2

Table 2
Experimental results on UDG.

Instance Family	DmDS $\overline{D}_{min}$	FastDS $\overline{D}_{min}$	ScBppw $\overline{D}_{min}$	EMOS $\overline{D}_{min}$
V50U150	**12.9**	**12.9**	**12.9**	**12.9**
V50U200	**9.4**	**9.4**	**9.4**	**9.4**
V100U150	**17.0**	**17.0**	17.3	**17.0**
V100U200	**10.4**	**10.4**	10.6	**10.4**
V250U150	**18.0**	**18.0**	19.0	20.2
V250U200	**10.8**	**10.8**	11.5	**10.8**
V500U150	**18.5**	**18.5**	20.1	25.9
V500U200	**11.2**	**11.2**	12.4	15.3
V800U150	**19.0**	**19.0**	20.9	27.2
V800U200	**11.7**	11.8	12.6	15.8
V1000U150	**19.1**	**19.1**	21.3	27.4
V1000U200	**12.0**	**12.0**	13.0	16.8

4.3. Experimental results

We report and discuss the experimental results from running ScBppw, FastDS, EMOS, and DmDS on all benchmark datasets. We first summarize the quality of the solutions computed by each algorithm. Table 1 gives a concise report on the competitive superiority of DmDS and the other three algorithms on different benchmarks. Specifically, it shows the number of instances on which an algorithm exclusively finds a smallest dominating set; that is, its solution is smaller than the solutions of all other algorithms. As can be seen from Table 1, DmDS significantly outperforms the other three algorithms on all benchmarks except BHOSLIB.

4.3.1. Comparison of solution quality

The results on UDG, T1/T2, BHOSLIB, SNAP, DIMACS10, and Network Repository benchmarks are reported in Tables 2, 3, 4, 6, and 7, respectively (Tables 6 and 7 are in Appendix). We mark the best solution found by all algorithms in bold. Regarding UDG and T1/T2 benchmarks in Tables 2 and 3, we follow Cai et al. [13] and report the average best solution for each family. It can be seen that DmDS has the best performance in terms of the "Min." values, FastDS has performance similar to DmDS in the "Avg." values, and ScBppw and EMOS perform worse than DmDS and FastDS. We now provide a more detailed discussion of the results.

Regarding standard benchmarks, the DmDS, FastDS, ScBppw, and EMOS algorithms obtain the best solutions on 12 (all), 11, two, and five families in the UDG benchmark, respectively; on 50 (out of 53), 43, nine, and 12 families in the T1/T2 benchmark, respectively; and on 30, 31 (all), zero, and zero instances in the BHOSLIB benchmark, respectively. In addition, DmDS obtains the best "Avg." value on 21 (out of 31) BHOSLIB instances, outperforming other algorithms. These results indicate that DmDS outperforms the other three algorithms on the standard benchmarks, though FastDS obtains a slightly smaller dominating set than DmDS on one BHOSLIB instance.

Regarding large real-world sparse graphs, the DmDS, FastDS, ScBppw, and EMOS algorithms obtain the best solutions respectively on 19 (out of 22), 17, 11, and seven instances in the SNAP benchmark, respectively on 22 (out of 27), 15, two, and one instance(s) in the DIMACS10 benchmark, and respectively on 219 (out of 237), 141, 81, and 46 instances in the Network Repository benchmark. In particular, DmDS obtains a smaller dominating set than other three algorithms on five SNAP benchmark instances, 12 DIMACS10 benchmark instances, and 94 Network Repository benchmark instances; FastDS obtains a smaller dominating set than other three algorithms on three SNAP benchmark instances, five DIMACS10 benchmark instances, and 16 Network Repository benchmark instances; ScBppw obtains a smaller dominating set than other three algorithms only on two Network Repository benchmark instances; and EMOS obtains a smaller dominating set

Table 3
Experimental results on T1/T2.

Instance Family	DmDS $\overline{D}_{min}$	FastDS $\overline{D}_{min}$	ScBppw $\overline{D}_{min}$	EMOS $\overline{D}_{min}$	Instance Family	DmDS $\overline{D}_{min}$	FastDS $\overline{D}_{min}$	ScBppw $\overline{D}_{min}$	EMOS $\overline{D}_{min}$
V50E50	**17.0**	**17.0**	**17.0**	**17.0**	V250E1000	**36.0**	**36.0**	36.6	41.8
V50E100	**11.9**	**11.9**	12.0	**11.9**	V250E2000	**21.6**	21.8	22.4	24.6
V50E250	**6.0**	**6.0**	**6.0**	**6.0**	V250E3000	**16.1**	16.2	16.6	18.0
V50E500	3.9	3.9	3.9	**3.8**	V250E5000	**11.0**	**11.0**	11.5	12.7
V50E750	3.0	3.0	3.0	**2.4**	V300E300	**100.0**	**100.0**	100.1	119.9
V100E100	**33.6**	**33.6**	**33.6**	**33.6**	V300E500	**77.7**	**77.7**	78.4	89.2
V100E250	**19.9**	**19.9**	20.1	**19.9**	V300E750	**59.6**	**59.6**	60.0	68.9
V100E500	**12.2**	**12.2**	12.5	**12.2**	V300E1000	**48.6**	**48.6**	49.5	57.1
V100E750	**9.0**	9.1	9.5	**9.0**	V300E2000	**29.4**	**29.4**	30.6	33.8
V100E1000	**7.5**	**7.5**	7.8	**7.5**	V300E3000	**22.2**	**22.2**	22.9	24.9
V100E2000	**4.1**	**4.1**	4.3	**4.1**	V300E5000	**15.3**	15.5	15.9	16.8
V150E150	**50.0**	**50.0**	**50.0**	**50.0**	V500E500	**167.0**	**167.0**	**167.0**	200.7
V150E250	**39.1**	**39.1**	39.3	42.3	V500E1000	**114.7**	**114.7**	116.3	132.5
V150E500	**24.6**	**24.6**	24.9	28.7	V500E2000	**71.2**	**71.2**	72.9	83.5
V150E750	**18.3**	**18.3**	18.8	20.7	V500E5000	**37.0**	**37.0**	38.7	42.3
V150E1000	**15.0**	**15.0**	15.4	17.1	V500E10000	**22.6**	22.7	23.3	24.8
V150E2000	**9.0**	**9.0**	9.5	10.0	V800E0	**267.0**	**267.0**	**267.0**	320.6
V150E3000	**6.9**	**6.9**	**6.9**	7.0	V800E1000	**242.5**	**242.5**	246.5	280.5
V200E250	**61.1**	**61.1**	61.7	70.3	V800E2000	**158.3**	**158.3**	162.1	185.8
V200E500	**39.6**	**39.6**	40.0	46.0	V800E5000	**82.7**	**82.7**	86.3	95.4
V200E750	**30.0**	**30.0**	30.4	34.8	V800E10000	**50.7**	50.8	53.1	56.4
V200E1000	**24.4**	**24.4**	25.0	27.8	V1000E1000	**333.7**	**333.7**	**333.7**	399.1
V200E2000	**15.0**	**15.0**	15.3	16.7	V1000E5000	**121.1**	**121.1**	126.0	141.8
V200E3000	**11.0**	**11.0**	11.3	12.3	V1000E10000	74.3	**74.2**	77.6	84.0
V250E250	**83.3**	**83.3**	**83.3**	99.6	V1000E15000	**55.8**	55.9	58.2	61.0
V250E500	**57.8**	**57.8**	58.1	67.0	V1000E20000	**45.7**	45.8	47.4	49.5
V250E750	**44.0**	**44.0**	44.9	51.7					

Table 4
Experimental results on BHOSLIB.

Instance			DmDS		FastDS		ScBppw		EMOS
Name	$\|V\|$	$\|E\|$	Avg.	Min.	Avg.	Min.	Avg.	Min.	Min.
frb40-19-1	760	41,314	14.0	**14**	14.1	**14**	15.8	15	17
frb40-19-2	760	41,263	14.3	**14**	14.4	**14**	16.5	16	17
frb40-19-3	760	41,095	15.0	**15**	15.0	**15**	16.5	16	17
frb40-19-4	760	41,605	14.3	**14**	14.5	**14**	16.2	15	17
frb40-19-5	760	41,619	14.2	**14**	14.2	**14**	15.2	15	16
frb45-21-1	945	59,186	16.2	**16**	16.3	**16**	18.0	17	18
frb45-21-2	945	58,624	16.2	**16**	16.5	**16**	18.1	18	19
frb45-21-3	945	58,245	16.1	**16**	16.0	**16**	17.8	17	18
frb45-21-4	945	58,549	16.2	**16**	16.2	**16**	18.0	17	19
frb45-21-5	945	58,579	16.0	**16**	16.2	**16**	18.2	18	19
frb50-23-1	1,150	80,072	18.1	**18**	18.1	**18**	19.9	19	20
frb50-23-2	1,150	80,851	18.0	**18**	18.2	**18**	19.8	19	21
frb50-23-3	1,150	81,068	18.1	**18**	18.1	**18**	20.0	20	21
frb50-23-4	1,150	80,258	18.2	**18**	18.0	**18**	20.0	19	20
frb50-23-5	1,150	80,035	18.1	**18**	18.2	**18**	19.9	19	21
frb53-24-1	1,272	94,227	19.0	**19**	19.0	**19**	21.1	21	22
frb53-24-2	1,272	94,289	19.4	**19**	19.2	**19**	21.2	21	22
frb53-24-3	1,272	94,127	19.0	**19**	19.0	**19**	20.4	20	21
frb53-24-4	1,272	94,308	18.6	**18**	18.4	**18**	20.2	19	21
frb53-24-5	1,272	94,226	19.0	**19**	19.2	**19**	21.5	20	23
frb56-25-1	1,400	109,676	20.0	**20**	20.1	**20**	22.2	21	23
frb56-25-2	1,400	109,401	20.1	**20**	20.2	**20**	22.5	21	22
frb56-25-3	1,400	109,379	20.1	**20**	20.1	**20**	22.2	21	23
frb56-25-4	1,400	110,038	20.1	**20**	20.2	**20**	22.5	22	23
frb56-25-5	1,400	109,601	19.9	**19**	19.9	**19**	21.8	21	22
frb59-26-1	1,534	126,555	20.8	**20**	20.7	**20**	22.6	22	23
frb59-26-2	1,534	126,163	21.0	21	20.8	**20**	22.6	22	23
frb59-26-3	1,534	126,082	21.2	**21**	21.0	**21**	23.1	23	24
frb59-26-4	1,534	127,011	21.3	**21**	21.1	**21**	23.6	23	23
frb59-26-5	1,534	125,982	21.3	**21**	21.2	**21**	24.0	23	25
frb100–40	4,000	572,774	36.4	**36**	36.3	**36**	40.1	39	40

Table 5
Averaged convergence time on three standard benchmarks.

Benchmark	DmDS	FastDS	ScBppw	EMOS
UDG	0.199 s	4.061 s	0.002 s	3.915 s
T1/T2	4.508 s	4.591 s	0.001 s	5.453 s
BHOSLIB	38.236 s	51.112 s	0.003 s	0.418 s

than other three algorithms only on two T1/T2 benchmark instances. Note that there are 78 brain network instances in Network Repository benchmark with vertex counts ranging from 300,000 to 1,000,000. Excluding these instances, DmDS, FastDS, ScBppw, and EMOS each compute a smaller dominating set than the other three algorithms on 25, 12, two, and zero Network Repository instances, respectively. Regarding the "Avg." value, DmDS obtains the best solutions on 17 (out of 22) SNAP instances, 19 (out of 27) DIMACS10 instances, and 214 (out of 237) Network Repository instances, which outperforms ScBppw, EMOS, and FastDS except for SNAP benchmark (FastDS obtains the best solutions on 18 SNAP instances). These results suggest that the DmDS algorithm significantly outperforms the other three algorithms on large sparse real-world benchmarks.

4.3.2. Comparison of convergence time

We now compare the *convergence time* – the time an algorithm takes to compute its final solution on an instance – for each algorithm. Note that since EMOS is an exact algorithm with a higher time complexity compared to heuristics, it cannot obtain the optimal solution within the time limit on most instances. Regarding the standard benchmarks, since the instances are small, all four algorithms converge quickly to their final solutions. Here we report the averaged convergence time of the four algorithms on these benchmarks. See Table 5 for the results: while ScBppw has the shortest convergence time (due to the lack of further improvement after the solution is obtained), DmDS has a faster convergence time than FastDS.

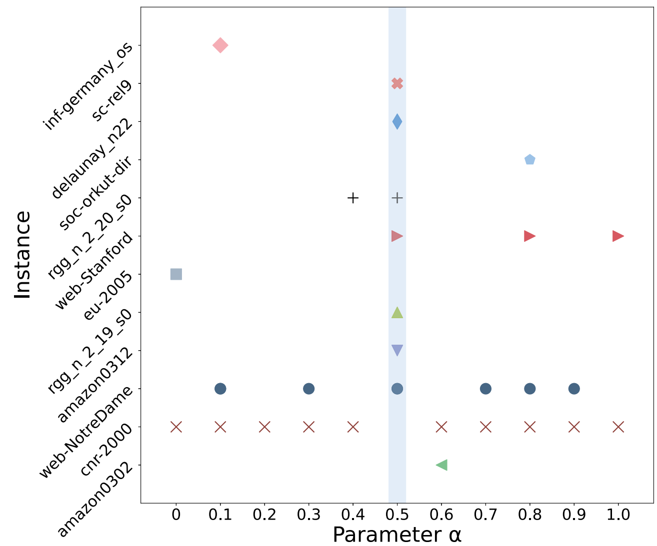

Fig. 2. Results of parameter tuning for α in DmDS.

Table 8 (see Appendix) reports the convergence time of the four algorithms running on large sparse real-world instances such that at least two algorithms obtain the smallest "Min." value (in total 149 instances). For each of such instances, the shortest convergence time among the four algorithms is highlighted in bold, and we use the notation "-" to indicate that the corresponding algorithm did not find a best solution (the smallest "Min." value). In addition, a running time less than 10^{-4} is reported as 0. As shown in the table, DmDS converges faster than the other three algorithms on 57 instances, while FastDS, ScBppw, and EMOS have this advantage on 46, 72, and 16 instances, respectively. These results indicate that in terms of the convergence time, ScBppw performs better than DmDS, and DmDS performs better than FastDS and EMOS. Note that ScBppw and EMOS do not find the best solution on 57 and 95 instances (out of 149), respectively, and both DmDS and FastDS can find a best solution on the 149 instances. This indicates that DmDS is superior to the other three algorithms on these instances.

4.3.3. Parameter tuning

We conduct an experiment to tune the parameter α in DmDS (Algorithm 3). Specifically, we select four instances from each of the SNAP, DIMACS10, and Network Repository benchmarks as representatives, in total 12 instances with vertex counts from 10^5 to 10^7. We measure the performance of DmDS on the 11 distinct values of α in {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}, and follow the experimental setup from Section 4.2. We run DmDS with a fixed value of α and 10 random seeds, and report the best solution. Fig. 2 shows the experimental results, where the x-axis represents values of α, and the y-axis represents instances. When DmDS finds the smallest dominating set on instance A using value $\alpha = a$, we plot a marker at coordinate (a, A). It can be seen from Fig. 2 that when $\alpha = 0.5$ DmDS finds a best solution on seven instances, and that this is the largest number of instances DmDS solves from this experiment. Therefore, we select $\alpha = 0.5$ for DmDS. Note that when $\alpha = 0.1$ and $\alpha = 1.0$, each iteration of DmDS is fixed to implement a $(2, 1)$-swap and $(3, 2)$-swap, respectively.

4.3.4. Initial solutions

Table 9 (see Appendix) reports the size of initial solutions of large sparse real-world benchmark instances obtained by GreedyConstructDS() and PerturbedConstructDS(), and the running time constructing the initial solutions. Note that only the instances for which the two solutions obtained by the two algorithms have different sizes

Table 6
Experimental results on SNAP and DIMACS10.

Instance	Properties		DmDS		FastDS		ScBppw		EMOS
Name	$\|V\|$	$\|E\|$	Avg.	Min.	Avg.	Min.	Avg.	Min.	Min.
Wiki-Vote	7,115	100,762	1,116.0	**1,116**	1,116.0	**1,116**	1,116.0	**1,116**	**1,116**
p2p-Gnutella04	10,876	39,994	2,227.0	**2,227**	2,227.0	**2,227**	2,227.0	**2,227**	2,245
p2p-Gnutella25	22,687	54,705	4,519.0	**4,519**	4,519.0	**4,519**	4,519.0	**4,519**	4,522
cit-HepTh	22,908	2,444,798	1,011.0	**1,011**	1,011.0	**1,011**	1,019.6	1,018	1,066
p2p-Gnutella24	26,518	65,369	5,418.0	**5,418**	5,418.0	**5,418**	5,418.0	**5,418**	**5,418**
cit-HepPh	28,093	3,148,447	894.0	**894**	894.0	**894**	907.0	905	972
p2p-Gnutella30	36,682	88,328	7,169.0	**7,169**	7,169.0	**7,169**	7,169.3	**7,169**	**7,169**
p2p-Gnutella31	62,586	147,892	12,582.0	**12,582**	12,582.0	**12,582**	12,582.0	**12,582**	**12,582**
soc-Epinions1	75,879	405,740	15,734.0	**15,734**	15,734.0	**15,734**	15,734.0	**15,734**	**15,734**
soc-Slashdot0811	77,360	469,180	14,312.0	**14,312**	14,312.0	**14,312**	14,312.0	**14,312**	14,316
soc-Slashdot0902	82,168	504,230	15,305.0	**15,305**	15,305.0	**15,305**	15,305.0	**15,305**	15,313
amazon0302	262,111	899,792	35,599.7	35,593	35,597.2	**35,587**	36,210.7	36,178	38,305
email-EuAll	265,009	364,481	17,976.0	**17,976**	17,976.0	**17,976**	17,976.0	**17,976**	**17,976**
web-Stanford	281,903	1,992,636	13,199.2	**13,197**	13,198.3	**13,197**	13,320.9	13,312	14,067
web-NotreDame	325,729	1,090,108	23,733.8	**23,733**	23,735.0	23,735	23,747.1	23,745	23,772
amazon0312	400,727	2,349,869	45,488.3	**45,480**	45,490.2	45,487	45,936.4	45,908	48,262
amazon0601	403,394	2,443,408	42,289.9	**42,283**	42,287.8	42,284	42,730.3	42,708	45,208
amazon0505	410,236	2,439,437	47,314.9	**47,309**	47,315.4	47,310	47,743.9	47,718	50,121
web-BerkStan	685,230	6,649,470	28,438.8	28,438	28,435.1	**28,434**	28,623.6	28,603	29,611
web-Google	875,713	4,322,051	79,699.0	79,699	79,698.0	**79,698**	79,758.7	79,751	80,073
wiki-Talk	2,394,385	4,659,565	36,960.0	**36,960**	36,960.0	**36,960**	36,960.0	**36,960**	**36,960**
cit-Patents	3,774,768	16,518,947	621,661.4	**621,644**	621,736.2	621,705	623,185.8	623,121	643,409
as-22july06	22,963	48,436	2,026.0	**2,026**	2,026.0	**2,026**	2,026.0	**2,026**	**2,026**
cond-mat-2005	39,577	175,693	5,664.0	**5,664**	5,664.0	**5,664**	5,690.9	5,686	5,710
kron_g500-logn16	55,321	2,456,071	3,885.0	**3,885**	3,885.0	**3,885**	3,885.0	**3,885**	3,888
luxembourg_os	114,599	119,666	37,753.2	**37,751**	37,751.5	**37,751**	38,444.8	38,421	39,460
rgg_n_2_17_s0	131,070	728,753	12,319.3	**12,317**	12,332.7	12,329	13,848.9	13,827	15,298
wave	156,317	1,059,331	11,681.6	11,654	11,646.3	**11,609**	13,927.0	13,887	14,780
caidaRouterLevel	192,244	609,066	40,523.0	**40,523**	40,523.0	**40,523**	40,580.4	40,573	41,035
coAuthorsCiteseer	227,320	814,134	33,197.0	**33,197**	33,197.0	**33,197**	33,358.1	33,343	33,267
citationCiteseer	268,495	1,156,647	43,412.0	**43,412**	43,412.0	**43,412**	43,434.3	43,430	44,022
coAuthorsDBLP	299,067	977,676	43,978.0	**43,978**	43,978.0	**43,978**	44,094.9	44,084	44,088
cnr-2000	325,557	2,738,969	22,010.4	**22,010**	22,011.2	**22,010**	22,030.9	22,027	22,100
coPapersCiteseer	434,102	16,036,720	26,082.0	**26,082**	26,082.0	**26,082**	27,008.0	26,989	26,961
rgg_n_2_19_s0	524,284	3,269,766	44,397.1	**44,378**	44,413.6	44,404	50,420.2	50,368	55,555
coPapersDBLP	540,486	15,245,729	35,596.7	**35,595**	35,598.7	35,597	37,090.3	37,060	37,439
eu-2005	862,664	16,138,468	32,287.8	**32,281**	32,288.4	32,284	32,325.2	32,316	32,456
audikw_1	943,695	38,354,076	9,486.2	9,469	9,467.3	**9,459**	10,568.5	10,534	11,375
ldoor	952,203	22,785,136	19,725.2	**19,710**	19,724.4	19,717	21,546.4	21,527	24,079
ecology1	1,000,000	1,998,000	202,057.3	**201,491**	201,664.9	201,505	238,250.4	238,052	250,334
rgg_n_2_20_s0	1,048,575	6,891,620	84,690.6	**84,665**	84,708.7	84,693	96,675.2	96,556	106,499
in-2004	1,382,867	13,591,473	77,781.9	**77,781**	77,787.0	77,786	77,861.6	77,850	77,990
belgium_osm	1,441,295	1,549,970	468,925.8	468,909	468,854.8	**468,846**	476,811.0	476,674	496,735
rgg_n_2_21_s0	2,097,148	14,487,995	162,242.7	**162,170**	162,303.2	162,273	185,435.2	185,346	204,017
rgg_n_2_22_s0	4,194,301	30,359,198	312,553.2	**312,471**	312,886.3	312,621	356,871.8	356,750	392,443
cage15	5,154,859	47,022,346	457,967.5	457,686	456,650.6	**456,524**	476,883.6	476,574	500,135
rgg_n_2_23_s0	8,388,607	63,501,393	608,116.9	**607,682**	609,133.6	608,149	687,953.9	687,850	755,935
rgg_n_2_24_s0	16,777,215	132,557,200	1,217,145.7	1,209,519	1,207,476.8	**1,205,162**	1,328,143.5	1,327,935	1,458,184
uk-2002	18,483,186	261,787,258	1,038,649.7	**1,038,641**	1,038,674.3	1,038,662	1,040,090.5	1,040,064	1,044,567

are reported, in total 213 instances. From the data in the table, we can see that the GreedyConstructDS() and the PerturbedConstructDS() have similar performance, obtaining better initial solutions than the other approach on 99 and 114 instances, respectively. In addition, the running time of these two algorithms is very close. This provides us with a method of improving the quality of the initial solution, that is, constructing two solutions by the two algorithms and selecting a better one as the initial solution.

4.3.5. Discussion

We briefly discuss our observations, which attempt to explain the strong performance of DmDS. First, DmDS can find a variety of solutions with different sizes within the 1000-second time limit, due in part to the intrinsic mechanism of local search. However, we have found that using a single search strategy may lead local search to become trapped in a local optimum, which may reduce the efficiency of solution improvement. DmDS integrates two distinct search strategies (i.e., $(2,1)$-swaps and $(3,2)$-swaps) and conducts tiny perturbations in the process of searching solutions, by which the diversity of solutions is enhanced and the issue of local optima is also weakened. Next, we observe that a strict tabu strategy restricts the search space, which also reduces the rate of improving a solution. Therefore, DmDS does not use tabu strategies — all vertices have an opportunity to be searched. The above analysis may explain why DmDS performs so well on the benchmark instances.

5. Conclusion

We proposed an efficient local search algorithm for the MinDS problem named DmDS. DmDS introduces a dual-mode local search framework in the search phase, computes an improved initial solution based on a greedy strategy and a perturbation mechanism, and introduces a new criterion for vertex selection. The experimental results show that DmDS significantly outperforms other state-of-the-art MinDS algorithms on seven benchmark datasets, consisting of 382 instances (or families). In the future, we would like to apply the three proposed techniques to solve other intractable graph problems, such as the minimum vertex cover.

Table 7
Experimental results on Network Repository.

Instance	Properties		DmDS		FastDS		ScBppw		EMOS
Name	$\lvert V\rvert$	$\lvert E\rvert$	Avg.	Min.	Avg.	Min.	Avg.	Min.	Min.
soc-highschool-moreno	70	274	11.0	**11**	11.0	**11**	11.0	**11**	**11**
reptilia-tortoise-network-cs	73	132	21.0	**21**	21.0	**21**	21.0	**21**	**21**
ENZYMES8	88	133	25.0	**25**	25.0	**25**	25.7	**25**	**25**
ENZYMES123	90	127	26.0	**26**	26.0	**26**	26.1	**26**	**26**
ENZYMES118	95	121	29.0	**29**	29.0	**29**	30.0	30	**29**
ENZYMES297	121	149	37.0	**37**	37.0	**37**	38.0	38	**37**
ENZYMES295	123	139	41.0	**41**	41.0	**41**	41.4	**41**	**41**
ENZYMES296	125	141	40.0	**40**	40.0	**40**	41.2	**40**	**40**
reptilia-tortoise-network-bsv	136	374	34.0	**34**	34.0	**34**	34.8	**34**	**34**
ia-enron-employees	150	1,526	9.0	**9**	9.0	**9**	9.0	**9**	**9**
soc-student-coop	185	311	48.0	**48**	48.0	**48**	48.6	**48**	**48**
soc-ANU-residence	217	1,839	17.0	**17**	17.0	**17**	17.7	**17**	20
bio-celegans	453	2,025	29.0	**29**	29.0	**29**	29.0	**29**	**29**
bio-celegans-dir	453	2,025	29.0	**29**	29.0	**29**	29.4	**29**	**29**
fb-pages-food	620	2,091	118.0	**118**	118.0	**118**	118.0	**118**	**118**
bio-SC-TS	636	3,959	124.0	**124**	124.0	**124**	124.0	**124**	125
bio-DM-LC	658	1,129	163.0	**163**	163.0	**163**	163.0	**163**	**163**
socfb-Caltech36	769	16,656	62.0	**62**	62.0	**62**	62.0	**62**	**62**
BA-1_10_60-L5	804	46,410	3.0	**3**	3.0	**3**	3.0	**3**	**3**
ia-crime-moreno	829	1,473	211.0	**211**	211.0	**211**	211.0	**211**	**211**
bio-CE-GT	924	3,239	126.0	**126**	126.0	**126**	126.8	**126**	**126**
gene	1,103	1,672	315.0	**315**	315.0	**315**	315.3	**315**	**315**
mammalia-voles-plj-trapping	1,263	3,380	280.0	**280**	280.0	**280**	281.5	**280**	**280**
bio-yeast	1,458	1,948	353.0	**353**	353.0	**353**	353.0	**353**	**353**
bio-yeast-protein-inter	1,846	2,203	514.0	**514**	514.0	**514**	514.0	**514**	**514**
bio-CE-PG	1,871	47,754	180.0	**180**	180.0	**180**	181.0	181	182
fb-messages	1,899	13,838	276.0	**276**	276.0	**276**	276.0	**276**	**276**
bio-grid-fission-yeast	2,026	12,637	280.0	**280**	280.0	**280**	280.0	**280**	**280**
tech-routers-rf	2,113	6,632	479.0	**479**	479.0	**479**	479.0	**479**	**479**
bio-CE-GN	2,220	53,683	195.0	**195**	195.0	**195**	195.7	**195**	202
bio-HS-HT	2,570	13,691	456.0	**456**	456.0	**456**	456.6	**456**	460
bio-CE-HT	2,617	2,985	690.0	**690**	690.0	**690**	690.0	**690**	**690**
bio-DM-HT	2,989	4,660	734.0	**734**	734.0	**734**	734.4	**734**	739
bio-DR-CX	3,289	84,940	310.0	**310**	310.0	**310**	311.2	**310**	344
bio-grid-worm	3,507	6,531	578.0	**578**	578.0	**578**	578.0	**578**	**578**
bio-DM-CX	4,040	76,717	513.0	**513**	513.0	**513**	513.5	**513**	540
ca-GrQc	4,158	13,422	776.0	**776**	776.0	**776**	778.4	**776**	**776**
bio-HS-LC	4,227	39,484	380.0	**380**	380.0	**380**	380.0	**380**	382
web-EPA	4,271	8,909	262.0	**262**	262.0	**262**	262.0	**262**	**262**
bio-HS-CX	4,413	108,818	495.0	**495**	495.0	**495**	495.0	**495**	516
web-spam	4,767	37,375	831.0	**831**	831.0	**831**	831.0	**831**	**831**
ca-Erdos992	5,094	7,515	440.0	**440**	440.0	**440**	440.0	**440**	**440**
soc-advogato	5,167	39,432	808.0	**808**	808.0	**808**	808.0	**808**	**808**
bio-grid-yeast	6,008	156,945	286.0	**286**	286.0	**286**	286.0	**286**	305
soc-wiki-elec	7,115	100,753	1,116.0	**1,116**	1,116.0	**1,116**	1,116.0	**1,116**	**1,116**
bio-grid-fruitfly	7,274	24,894	1,522.0	**1,522**	1,522.0	**1,522**	1,522.0	**1,522**	**1,522**
bio-dmela	7,393	25,569	1,453.0	**1,453**	1,453.0	**1,453**	1,453.0	**1,453**	1,454
rec-movielens-user-movies-10 m	7,601	55,384	341.0	**341**	341.0	**341**	341.0	**341**	**341**
bio-grid-human	9,436	31,182	1,785.0	**1,785**	1,785.0	**1,785**	1,785.0	**1,785**	1,790
tech-pgp	10,680	24,316	2,711.0	**2,711**	2,711.0	**2,711**	2,711.3	**2,711**	2,712
Oregon-1	11,174	23,409	992.0	**992**	992.0	**992**	992.0	**992**	**992**
Oregon-2	11,461	32,730	961.0	**961**	961.0	**961**	961.0	**961**	**961**
skirt	12,595	91,961	909.0	**909**	909.0	**909**	1,121.9	1,111	1,014
cbuckle	13,681	331,417	290.8	**290**	291.8	291	368.7	363	326
cyl6	13,681	350,280	241.0	**241**	241.0	**241**	308.0	301	**241**
bio-human-gene2	14,022	9,027,024	392.0	**392**	392.0	**392**	392.6	**392**	426
case9	14,453	72,171	2,403.0	**2,403**	2,403.0	**2,403**	2,403.0	**2,403**	3,591
bio-CE-CX	15,229	245,952	2,549.0	**2,549**	2,549.0	**2,549**	2,552.7	2,552	2,704
Dubcova1	16,129	118,440	1,024.0	**1,024**	1,024.0	**1,024**	1,183.2	1,175	1,351
olafu	16,146	499,505	285.0	**285**	285.0	**285**	364.1	349	371
bio-WormNet-v3	16,347	762,822	2,072.4	**2,072**	2,072.1	**2,072**	2,075.8	2,073	2,202
rec-movielens-tag-movies-10 m	16,528	71,067	2,992.0	**2,992**	2,992.0	**2,992**	2,992.0	**2,992**	2,995
ca-AstroPh	17,903	196,972	2,055.0	**2,055**	2,055.0	**2,055**	2,073.0	2,068	2,111
soc-political-retweet	18,470	48,053	3,277.0	**3,277**	3,277.0	**3,277**	3,277.0	**3,277**	**3,277**
raefsky4	19,779	654,416	322.0	**322**	322.0	**322**	404.1	393	374
raefsky3	21,200	733,784	300.0	**300**	300.0	**300**	375.4	369	308
ca-CondMat	21,363	91,286	2,990.0	**2,990**	2,990.0	**2,990**	3,011.9	3,008	3,035
bio-human-gene1	21,890	12,323,680	825.3	**825**	825.8	**825**	827.1	826	884
rgg_n_2_15_s0	32,766	160,240	3,473.4	**3,472**	3,475.8	3,475	3,857.2	3,838	4,258
bio-pdb1HYS	36,417	2,154,174	413.5	**412**	422.5	421	501.2	497	559
c-62ghs	41,731	258,806	15,161.0	**15,161**	15,161.0	**15,161**	15,248.2	15,240	15,200

(*continued on next page*)

Table 7 (*continued*).

bio-mouse-gene	43,126	14,461,095	2,713.6	**2,713**	2,713.2	**2,713**	2,715.9	2,714	2,859
c-66b	49,989	224,509	21,183.0	**21,183**	21,183.0	**21,183**	21,224.8	21,220	21,205
sc-nasasrb	54,870	1,311,227	1,044.6	**1,044**	1,045.0	1,045	1,385.7	1,374	1,102
Dubcova2	65,025	482,600	4,096.0	**4,096**	4,096.1	**4,096**	4,709.8	4,698	5,373
rgg_n_2_16_s0	65,532	342,127	6,536.7	**6,535**	6,543.5	6,539	7,287.3	7,270	8,033
sc-pkustk11	87,804	2,565,054	2,260.0	**2,260**	2,260.3	**2,260**	2,367.6	2,361	2,591
soc-BlogCatalog	88,784	2,093,195	4,894.0	**4,894**	4,894.0	**4,894**	4,894.0	**4,894**	**4,894**
sc-pkustk13	94,893	3,260,967	1,223.2	**1,222**	1,223.8	**1,222**	1,462.3	1,449	1,468
soc-buzznet	101,163	2,763,066	127.0	**127**	127.0	**127**	127.0	**127**	133
soc-LiveMocha	104,103	2,193,083	1,424.0	**1,424**	1,424.0	**1,424**	1,424.0	**1,424**	1,464
kron_g500-logn17	107,909	5,113,985	7,468.0	**7,468**	7,468.0	**7,468**	7,468.0	**7,468**	7,469
web-uk-2005	129,632	11,744,049	1,421.0	**1,421**	1,421.0	**1,421**	1,421.0	**1,421**	**1,421**
sc-shipsec1	140,385	1,707,759	7,668.9	7,664	7,666.3	**7,660**	8,805.2	8,788	9,341
Dubcova3	146,689	1,744,980	4,096.0	**4,096**	4,097.0	**4,096**	4,725.8	4,707	5,201
soc-douban	154,908	327,162	8,364.0	**8,364**	8,364.0	**8,364**	8,364.0	**8,364**	**8,364**
web-arabic-2005	163,598	1,747,269	16,901.5	**16,901**	16,901.7	**16,901**	16,908.2	16,904	17,012
rec-dating	168,791	17,351,416	11,734.0	**11,734**	11,734.3	**11,734**	11,736.4	11,735	11,745
sc-shipsec5	179,104	2,200,076	10,310.9	10,299	10,303.3	**10,296**	11,879.1	11,854	12,608
soc-academia	200,169	1,022,441	28,325.0	**28,325**	28,325.0	**28,325**	28,332.1	28,329	28,659
kron_g500-logn18	210,155	10,582,686	14,268.0	**14,268**	14,268.0	**14,268**	14,268.0	**14,268**	14,271
sc-pwtk	217,891	5,653,221	4,198.0	**4,196**	4,202.1	4,201	5,520.1	5,490	4,637
rec-libimseti-dir	220,970	17,233,144	12,954.8	**12,954**	12,954.8	**12,954**	12,956.1	**12,954**	12,995
rgg_n_2_18_s0	262,141	1,547,283	23,354.0	**23,349**	23,374.1	23,367	26,427.0	26,407	29,117
ca-dblp-2012	317,080	1,049,866	46,138.0	**46,138**	46,138.0	**46,138**	46,264.6	46,254	46,249
bn-human-Jung2015_M87119044	317,252	65,582,968	6,964.8	**6,961**	6,964.5	6,962	7,492.0	7,481	7,895
bn-human-Jung2015_M87116523	329,521	77,777,027	6,589.4	**6,586**	6,591.2	6,590	7,117.0	7,104	7,510
ca-MathSciNet	332,689	820,644	65,572.0	**65,572**	65,572.0	**65,572**	65,597.2	65,592	65,650
sc-msdoor	404,785	9,378,650	8,604.3	8,600	8,605.4	**8,596**	10,317.0	10,282	10,583
kron_g500-logn19	409,175	21,780,787	27,748.0	**27,748**	27,748.0	**27,748**	27,748.0	**27,748**	–
soc-dogster	426,816	8,543,549	26,249.0	**26,249**	26,249.0	**26,249**	26,252.5	26,251	26,409
bn-human-Jung2015_M87118347	428,842	79,114,771	7,861.5	**7,855**	7,861.3	**7,855**	8,580.9	8,547	9,063
soc-twitter-higgs	456,631	12,508,442	14,689.5	**14,689**	14,689.5	**14,689**	14,690.5	**14,689**	15,076
soc-youtube	495,957	1,936,748	89,732.0	**89,732**	89,732.0	**89,732**	89,734.9	89,733	89,792
web-it-2004	509,338	7,178,413	32,997.0	**32,997**	32,997.0	**32,997**	32,999.4	32,998	33,000
soc-flickr	513,969	3,190,452	98,062.0	**98,062**	98,062.0	**98,062**	98,064.5	98,063	98,096
soc-delicious	536,108	1,365,961	55,722.0	**55,722**	55,722.0	**55,722**	55,728.3	55,726	55,758
ca-coauthors-dblp	540,486	15,245,729	35,597.4	**35,596**	35,598.2	35,597	37,092.5	37,069	37,462
bn-human-Jung2015_M87110650	628,995	40,701,380	26,584.7	**26,579**	26,588.7	26,583	28,651.1	28,597	30,016
soc-FourSquare	639,014	3,214,986	60,979.0	**60,979**	60,979.0	**60,979**	60,979.0	**60,979**	61,016
bn-human-BNU_1_0025873*1-bg	645,518	149,547,446	19,680.2	**19,673**	19,683.5	19,677	20,997.5	20,982	–
bn-human-Jung2015_M87124152	680,788	144,225,699	17,154.5	**17,149**	17,156.0	17,150	18,571.3	18,551	–
bn-human-BNU_1_0025896*2-bg	687,745	114,530,787	21,322.9	**21,316**	21,327.6	21,318	22,897.0	22,861	–
bn-human-BNU_1_0025869*1-bg	690,519	135,038,917	23,432.2	**23,425**	23,434.7	23,428	24,987.5	24,955	–
bn-human-Jung2015_M87116517	690,931	134,151,255	19,283.1	**19,275**	19,287.7	**19,275**	20,952.5	20,927	–
bn-human-BNU_1_0025914*1-bg	692,562	118,071,816	25,186.2	**25,181**	25,186.1	25,183	26,809.0	26,788	28,308
bn-human-BNU_1_0025864*1-bg	696,338	143,158,339	23,505.2	**23,497**	23,507.8	23,501	25,000.5	24,984	–
bn-human-Jung2015_M87124670	699,442	42,402,074	25,621.4	**25,612**	25,627.2	25,615	28,093.0	28,069	29,429
bn-human-BNU_1_0025878*1-bg	699,697	127,906,128	23,361.2	**23,355**	23,362.2	**23,355**	24,930.1	24,906	–
bn-human-BNU_1_0025914*2	701,145	103,134,404	24,485.8	**24,478**	24,490.0	24,483	26,114.4	26,068	27,521
bn-human-BNU_1_0025889*1	704,694	144,469,850	23,261.2	**23,256**	23,262.7	23,260	24,865.7	24,848	–
bn-human-BNU_1_0025917*1	706,494	144,880,821	22,010.1	**22,000**	22,013.6	22,004	23,524.7	23,499	–
bn-human-Jung2015_M87104201	707,284	191,224,983	15,787.2	**15,781**	15,787.4	**15,781**	17,151.9	17,132	–
bn-human-BNU_1_0025916*1	714,571	112,519,748	23,081.3	**23,075**	23,083.8	23,078	24,740.8	24,696	26,052
bn-human-BNU_1_0025869*2-bg	716,150	151,544,915	25,097.9	**25,088**	25,100.6	25,092	26,814.0	26,746	–
bn-human-BNU_1_0025890*2	723,881	158,147,409	23,809.4	**23,802**	23,811.0	23,803	25,377.2	25,361	–
bn-human-Jung2015_M87115663	724,456	176,054,087	18,897.3	**18,889**	18,899.7	18,891	20,344.6	20,323	–
bn-human-Jung2015_M87124029	724,458	43,281,705	26,998.8	**26,988**	27,006.5	26,995	29,432.9	29,406	30,901
bn-human-Jung2015_M87127186	725,452	165,672,643	17,676.4	17,672	17,678.8	**17,671**	19,173.1	19,153	–
bn-human-BNU_1_0025913*2	726,197	183,978,766	22,474.8	**22,469**	22,476.2	22,472	24,003.5	23,976	–
bn-human-BNU_1_0025868*1-bg	727,487	150,443,553	24,490.2	**24,480**	24,492.9	24,486	26,108.8	26,093	–
bn-human-BNU_1_0025868*2-bg	728,087	158,620,929	24,939.8	**24,933**	24,941.1	24,934	26,540.4	26,482	–
bn-human-Jung2015_M87101698	728,370	144,737,265	21,064.1	**21,057**	21,066.7	21,061	22,736.2	22,709	–
bn-human-BNU_1_0025911*2	730,624	148,225,147	25,251.6	**25,241**	25,252.2	25,245	26,940.6	26,910	–
bn-human-Jung2015_M87105966	734,186	136,408,625	21,680.3	**21,675**	21,686.1	21,677	23,389.2	23,355	–
bn-human-BNU_1_0025865*1-bg	734,561	165,916,089	21,631.0	**21,621**	21,631.8	21,622	23,122.4	23,097	–
bn-human-BNU_1_0025871*2-bg	734,729	171,005,692	23,030.7	**23,026**	23,034.9	23,027	24,594.4	24,559	–
bn-human-Jung2015_M87104509	737,579	50,037,313	27,461.1	**27,448**	27,467.2	27,459	29,796.0	29,766	31,353
bn-human-BNU_1_0025867*1-bg	747,410	145,276,294	25,429.7	**25,423**	25,431.2	25,426	27,115.4	27,090	–
bn-human-BNU_1_0025918*1	748,521	159,835,566	24,366.3	24,359	24,366.4	**24,358**	26,088.8	26,057	–
bn-human-Jung2015_M87108808	750,844	175,282,659	22,359.2	**22,353**	22,360.1	22,354	24,112.5	24,079	–
bn-human-Jung2015_M87125286	753,905	209,976,387	19,797.9	**19,792**	19,800.3	19,795	21,281.7	21,264	–
rec-epinion	755,200	13,396,042	9,037.0	**9,037**	9,037.0	**9,037**	9,038.0	9,038	–
bn-human-Jung2015_M87110148	760,221	143,436,058	20,458.2	**20,448**	20,466.3	20,454	22,202.1	22,179	–
bn-human-Jung2015_M87110670	761,071	52,355,271	26,034.8	26,022	26,041.0	**26,021**	28,495.5	28,446	30,025
bn-human-Jung2015_M87125334	763,149	40,258,003	30,331.1	**30,324**	30,343.3	30,333	33,271.3	33,228	34,930

(*continued on next page*)

Table 7 (*continued*).

bn-human-BNU_1_0025874*2-bg	769,392	163,523,401	25,406.0	**25,400**	25,409.4	**25,400**	27,106.0	27,090	–
soc-digg	770,799	5,907,132	66,155.0	**66,155**	66,155.0	**66,155**	66,156.2	**66,155**	66,188
bn-human-Jung2015_M87101705	776,644	201,198,184	18,646.9	**18,638**	18,642.6	18,639	20,267.4	20,239	–
bn-human-Jung2015_M87124563	776,845	160,996,800	16,617.1	16,609	16,617.2	**16,608**	18,212.9	18,182	–
bn-human-BNU_1_0025876*2-bg	779,330	139,871,242	29,366.0	**29,349**	29,366.2	29,355	31,183.8	31,159	–
bn-human-Jung2015_M87125521	779,372	44,010,434	30,312.1	**30,297**	30,318.0	30,308	33,179.4	33,153	34,930
bn-human-BNU_1_0025886*1	780,185	158,184,747	24,542.1	**24,528**	24,539.2	24,534	26,228.3	26,182	–
bn-human-Jung2015_M87107242	781,156	49,597,087	31,874.1	**31,862**	31,882.8	31,876	34,356.0	34,326	36,189
bn-human-BNU_1_0025912*2	781,747	147,562,552	24,889.2	**24,880**	24,889.4	24,881	26,554.8	26,540	–
bn-human-Jung2015_M87113878	784,262	267,844,669	14,327.7	**14,321**	14,330.8	14,325	15,588.2	15,568	–
bn-human-Jung2015_M87121714	784,507	50,740,434	28,440.6	**28,432**	28,449.3	28,441	31,131.7	31,096	32,811
bn-human-BNU_1_0025876*1-bg	789,979	140,362,426	29,380.3	**29,370**	29,387.7	29,382	31,254.5	31,218	–
bn-human-Jung2015_M87118219	791,219	121,907,663	23,066.9	**23,062**	23,074.7	23,064	24,902.6	24,875	26,320
bn-human-Jung2015_M87127677	792,187	208,902,538	23,412.1	**23,401**	23,412.7	23,403	25,081.4	25,052	–
bn-human-Jung2015_M87118759	794,819	58,208,860	28,597.4	**28,587**	28,604.9	28,597	31,307.9	31,270	32,906
kron_g500-logn20	795,241	44,619,402	53,350.0	**53,350**	53,350.0	**53,350**	53,350.0	**53,350**	–
bn-human-BNU_1_0025870*1-bg	797,293	148,754,618	27,821.9	**27,811**	27,824.6	27,817	29,553.7	29,531	–
bn-human-Jung2015_M87125691	800,878	48,117,719	31,566.3	**31,552**	31,575.0	31,567	34,417.3	34,384	36,152
bn-human-Jung2015_M87111392	801,214	175,576,604	23,242.1	**23,219**	23,241.2	23,231	25,009.9	24,985	–
bn-human-Jung2015_M87109786	801,996	60,580,327	29,334.7	**29,324**	29,339.7	29,331	32,097.8	32,072	33,696
bn-human-Jung2015_M87102230	806,303	59,411,819	28,964.7	**28,957**	28,972.1	28,965	31,551.3	31,497	33,125
bn-human-BNU_1_0025870*2-bg	810,505	166,806,043	27,226.1	**27,218**	27,228.0	27,222	28,981.8	28,943	–
bn-human-Jung2015_M87128194	822,606	134,045,684	23,101.5	**23,088**	23,106.6	23,097	25,121.4	25,094	26,493
bn-human-Jung2015_M87129974	826,616	53,227,127	32,349.2	**32,334**	32,351.0	32,336	35,012.9	34,941	36,787
bn-human-Jung2015_M87125330	826,828	55,672,197	30,379.7	**30,366**	30,387.6	30,378	33,276.7	33,228	34,929
bn-human-Jung2015_M87119472	835,832	59,548,327	26,788.3	**26,775**	26,799.1	26,787	29,618.7	29,599	31,177
bn-human-Jung2015_M87103674	839,591	58,903,564	29,427.4	**29,404**	29,434.9	29,425	32,025.4	31,999	33,674
bn-human-Jung2015_M87118954	840,687	72,908,732	19,318.8	**19,306**	19,325.6	19,316	21,721.8	21,703	22,852
bn-human-Jung2015_M87123142	846,535	53,825,327	30,256.5	**30,253**	30,264.6	30,255	33,153.3	33,121	34,905
bn-human-Jung2015_M87125989	848,695	51,882,604	32,248.0	**32,242**	32,259.1	32,249	35,412.7	35,368	37,272
bn-human-Jung2015_M87104300	851,113	67,658,067	24,855.2	**24,848**	24,864.3	24,851	27,446.4	27,375	28,862
bn-human-Jung2015_M87127667	853,543	73,862,001	19,365.5	**19,355**	19,368.0	19,359	21,772.6	21,734	22,935
bn-human-Jung2015_M87128519	861,636	169,367,851	21,196.1	**21,186**	21,199.7	21,191	23,111.7	23,087	–
bn-human-Jung2015_M87113679	874,741	62,333,443	28,895.0	**28,886**	28,904.4	28,888	31,765.1	31,713	33,600
bn-human-Jung2015_M87115834	877,902	57,190,876	29,321.9	**29,309**	29,331.4	29,310	32,194.8	32,152	33,987
bn-human-Jung2015_M87117515	891,589	48,669,606	28,704.7	**28,688**	28,719.0	28,707	31,984.5	31,960	33,692
bn-human-Jung2015_M87123456	895,179	51,025,656	32,750.9	**32,737**	32,762.7	32,749	35,956.5	35,874	37,987
ca-IMDB	896,305	3,782,447	120,570.3	**120,570**	120,570.8	**120,570**	120,573.7	120,572	120,788
sc-ldoor	909,537	20,770,807	19,746.2	19,740	19,743.0	**19,731**	23,621.6	23,582	24,249
bn-human-Jung2015_M87122310	924,284	94,370,886	16,712.0	**16,701**	16,716.1	16,707	18,810.1	18,772	19,877
bn-human-Jung2015_M87102575	935,265	87,273,967	17,672.6	**17,659**	17,675.6	17,664	19,982.8	19,965	21,100
bn-human-Jung2015_M87117093	939,474	124,038,605	22,191.3	**22,179**	22,194.1	22,188	24,643.5	24,605	25,993
bn-human-Jung2015_M87126525	975,930	146,109,300	15,326.8	**15,316**	15,333.4	15,323	17,448.5	17,424	18,439
ca-hollywood-2009	1,069,126	56,306,653	48,733.5	**48,728**	48,746.6	48,743	50,369.7	50,351	51,564
inf-roadNet-PA	1,087,562	1,541,514	326,866.6	**326,845**	326,939.6	326,921	332,187.0	332,099	345,133
rt-retweet-crawl	1,112,702	2,278,852	75,740.0	**75,740**	75,740.0	**75,740**	75,742.0	**75,740**	**75,740**
soc-youtube-snap	1,134,890	2,987,624	213,122.0	**213,122**	213,122.0	**213,122**	213,123.0	**213,122**	213,133
soc-lastfm	1,191,805	4,519,330	67,226.0	**67,226**	67,226.0	**67,226**	67,226.0	**67,226**	**67,226**
kron_g500-logn21	1,544,087	91,040,932	102,238.0	**102,238**	102,238.0	**102,238**	102,238.0	**102,238**	–
soc-pokec	1,632,803	22,301,964	207,311.2	207,307	207,308.7	**207,303**	207,394.9	207,385	213,833
tech-as-skitter	1,694,616	11,094,209	181,718.0	**181,717**	181,717.9	**181,717**	181,867.8	181,856	184,351
soc-flickr-und	1,715,255	15,555,041	295,700.0	**295,700**	295,700.0	**295,700**	295,704.8	295,702	295,757
web-wikipedia2009	1,864,433	4,507,315	346,582.3	**346,580**	346,583.9	346,582	346,684.3	346,670	347,984
web-wikipedia-growth	1,870,709	36,532,531	116,814.5	**116,814**	116,814.6	**116,814**	116,819.7	116,817	–
inf-roadNet-CA	1,957,027	2,760,388	586,395.0	**586,364**	586,492.8	586,481	595,852.8	595,760	621,705
web-baidu-baike	2,140,198	17,014,946	276,745.0	**276,745**	276,745.0	**276,745**	276,748.3	276,747	–
tech-ip	2,250,498	21,643,497	154.9	**154**	154.8	**154**	156.7	155	–
soc-flixster	2,523,386	7,918,801	91,019.0	**91,019**	91,019.0	**91,019**	91,019.0	**91,019**	**91,019**
memchip	2,707,524	6,621,370	464,234.4	**464,194**	464,263.9	464,215	466,616.9	466,512	482,280
socfb-B-anon	2,937,612	20,959,854	187,030.0	**187,030**	187,030.0	**187,030**	187,030.2	**187,030**	187,082
soc-orkut	2,997,166	106,349,209	110,566.1	**110,550**	110,599.3	110,578	111,107.2	111,080	119,557
soc-orkut-dir	3,072,441	117,185,083	93,619.5	**93,603**	93,655.4	93,630	94,005.9	93,982	100,329
socfb-A-anon	3,097,165	23,667,394	201,690.0	**201,690**	201,690.0	**201,690**	201,690.3	**201,690**	201,845
wikipedia_link_en	3,371,708	31,024,475	212,876.0	**212,876**	212,876.0	**212,876**	212,876.0	**212,876**	212,880
Freescale1	3,428,754	8,472,832	597,141.5	**597,132**	597,162.7	597,152	598,622.1	598,580	617,222
patents	3,750,822	14,970,766	632,687.8	**632,665**	632,776.5	632,733	634,033.5	633,989	654,447
dblp-author	4,000,150	8,649,002	1,411,236.0	**1,411,236**	1,411,236.3	**1,411,236**	1,411,237.5	**1,411,236**	1,411,245
delicious-ti	4,006,816	14,976,217	140,676.0	**140,676**	140,676.0	**140,676**	140,676.4	**140,676**	–
soc-livejournal	4,033,137	27,933,062	793,887.9	**793,887**	793,888.6	793,888	793,994.6	793,987	797,153
delaunay_n22	4,194,304	12,582,869	641,622.1	641,480	641,542.6	**641,394**	688,804.3	688,628	760,044
channel-500 × 100x100-b050	4,802,000	42,681,372	343,731.6	343,328	343,003.4	**342,733**	392,503.0	392,415	434,881
ljournal-2008	5,363,186	49,514,271	1,005,858.0	**1,005,858**	1,005,858.0	**1,005,858**	1,005,986.5	1,005,974	1,008,292
soc-ljournal-2008	5,363,186	49,514,271	1,005,858.0	**1,005,858**	1,005,858.0	**1,005,858**	1,005,980.1	1,005,970	1,008,275
sc-rel9	5,921,786	23,667,162	119,305.4	119,234	119,247.4	**119,208**	130,424.5	123,108	150,161
indochina-2004	7,414,758	150,984,819	399,928.9	**399,922**	399,952.9	399,949	400,516.5	400,507	–
soc-livejournal-user-groups	7,489,073	112,305,407	1,071,123.0	**1,071,123**	1,071,123.0	**1,071,123**	1,071,124.0	**1,071,123**	–
delaunay_n23	8,388,608	25,165,784	1,294,633.7	**1,291,532**	1,295,994.0	1,293,999	1,377,716.0	1,377,473	1,519,995

(*continued on next page*)

Table 7 (*continued*).

friendster	8,658,744	45,671,471	656,463.0	**656,463**	656,463.0	**656,463**	656,466.4	656,464	656,688
wb-edu	9,450,404	46,236,105	833,550.9	**833,546**	833,557.0	833,553	834,338.9	834,307	837,052
twitter_mpi	9,862,150	99,940,317	566,313.0	**566,313**	566,313.0	**566,313**	566,313.5	**566,313**	–
inf-germany_os	11,548,845	12,369,181	3,815,331.6	3,813,153	3,800,840.1	**3,800,262**	3,847,240.5	3,846,783	3,993,482
hugetrace-00010	12,057,441	18,082,179	3,301,282.3	3,274,050	3,256,924.8	**3,253,894**	3,392,443.6	3,391,758	3,649,279
dbpedia-link	13,980,908	96,786,150	1,968,836.0	**1,968,836**	1,968,837.9	1,968,837	1,968,839.0	1,968,839	–
inf-road_central	14,081,816	16,933,413	4,581,139.1	**4,580,173**	4,585,195.9	4,584,680	4,615,658.1	4,615,457	4,771,095
road_central	14,081,816	16,933,413	4,580,701.5	**4,579,690**	4,583,212.3	4,582,927	4,615,769.5	4,615,468	4,771,095
hugetrace-00020	16,002,413	23,998,813	4,501,029.6	4,496,071	4,428,935.2	**4,412,314**	4,509,342.4	4,508,908	4,893,492
delaunay_n24	16,777,216	50,331,601	2,690,868.0	2,672,734	2,661,379.5	**2,651,597**	2,755,337.8	2,755,004	3,040,275
hugebubbles-00020	21,198,119	31,790,179	6,082,340.5	6,037,152	6,034,858.7	6,028,600	5,985,251.5	**5,984,831**	6,543,902
inf-road-usa	23,947,347	28,854,312	7,817,568.8	**7,804,560**	7,826,434.1	7,818,748	7,855,036.8	7,854,395	8,114,432
inf-europe_os	50,912,018	54,054,660	17,385,174.3	17,359,873	17,205,514.2	17,191,561	17,019,317.7	**17,017,316**	17,471,789
socfb-uci-uni	58,790,782	92,208,195	865,675.0	**865,675**	865,675.0	**865,675**	865,675.0	**865,675**	865,678

Table 8
Averaged convergence time for instances with ties.

Instance	DmDS	FastDS	ScBppw	EMOS	Instance	DmDS	FastDS	ScBppw	EMOS
Wiki-Vote	0.032	0.019	**0.008**	0.25	Oregon-2	0.009	0.009	**0.002**	0.06
p2p-Gnutella04	**0.038**	0.057	**0.038**	–	skirt	**3.509**	7.143	–	–
p2p-Gnutella25	0.028	0.016	**0.005**	–	cyl6	0.103	0.083	–	**0.08**
cit-HepTh	0.602	**0.444**	–	–	bio-human-gene2	**48.279**	84.489	296.277	–
p2p-Gnutella24	0.038	**0.032**	0.048	12.09	case9	17.372	51.687	**0.163**	–
cit-HepPh	**297.148**	336.827	–	–	bio-CE-CX	**36.664**	45.116	–	–
p2p-Gnutella30	0.051	0.039	**0.019**	0.1	Dubcova1	**0.208**	0.364	–	–
p2p-Gnutella31	0.082	0.062	0.051	**0.03**	olafu	**2.789**	4.427	–	–
soc-Epinions1	0.152	0.098	**0.071**	173.65	bio-WormNet-v3	**194.874**	390.686	–	–
soc-Slashdot0811	0.176	**0.129**	0.176	–	rec-movielens-tag-movies-10 m	0.035	0.02	**0.012**	–
soc-Slashdot0902	0.208	**0.186**	0.25	–	ca-AstroPh	**0.171**	0.27	–	–
email-EuAll	0.126	0.116	**0.057**	0.45	soc-political-retweet	0.014	0.018	**0.007**	0.04
web-Stanford	584.935	**573.48**	–	–	raefsky4	**0.43**	0.554	–	–
wiki-Talk	2.853	2.64	**1.376**	96.06	raefsky3	**0.162**	0.292	–	–
as-22july06	0.011	0.015	**0.006**	0.12	ca-CondMat	**0.985**	1.784	–	–
cond-mat-2005	**0.254**	0.56	–	–	bio-human-gene1	252.077	**153.672**	–	–
kron_g500-logn16	1.712	0.893	**0.236**	–	c-62ghs	0.099	**0.049**	–	–
luxembourg_os	663.454	**537.778**	–	–	bio-mouse-gene	**91.291**	185.752	–	–
caidaRouterLevel	**52.34**	120.622	–	–	c-66b	0.074	**0.048**	–	–
coAuthorsCiteseer	**1.374**	3.094	–	–	Dubcova2	**0.356**	0.369	–	–
citationCiteseer	**16.331**	20.197	–	–	sc-pkustk11	599.457	**441.788**	–	–
coAuthorsDBLP	**1.192**	3.308	–	–	soc-BlogCatalog	0.27	0.303	**0.074**	20.59
cnr-2000	**27.704**	94.148	–	–	sc-pkustk13	**186.771**	443.231	–	–
coPapersCiteseer	**353.499**	433.596	–	–	soc-buzznet	794.95	628.716	**0.087**	–
soc-highschool-moreno	**0**	**0**	**0**	0.01	soc-LiveMocha	0.846	0.418	**0.114**	–
reptilia-tortoise-network-cs	**0**	**0**	**0**	**0**	kron_g500-logn17	1.585	**1.25**	1.894	–
ENZYMES8	**0**	**0**	**0**	**0**	web-uk-2005	0.503	0.491	**0.191**	1.62
ENZYMES123	**0**	**0**	**0**	0.01	Dubcova3	**13.441**	198.132	–	–
ENZYMES118	**0**	**0**	–	0.01	soc-douban	0.096	0.088	**0.055**	0.19
ENZYMES297	**0.008**	0.013	–	2.57	web-arabic-2005	172.16	**39.468**	–	–
ENZYMES295	**0**	**0**	**0**	0.07	rec-dating	**25.435**	196.067	–	–
ENZYMES296	**0**	**0**	**0**	0.9	soc-academia	**2.854**	3.341	–	–
reptilia-tortoise-network-bsv	**0**	**0**	**0**	**0**	kron_g500-logn18	4.024	2.086	**0.709**	–
ia-enron-employees	**0**	**0**	**0**	**0**	rec-libimseti-dir	**75.205**	443.944	248.972	–
soc-student-coop	**0**	**0**	**0**	0.02	ca-dblp-2012	**4.946**	12.918	–	–
soc-ANU-residence	0.001	0.003	**0**	–	ca-MathSciNet	**0.919**	1.424	–	–
bio-celegans	**0**	**0**	**0**	**0**	kron_g500-logn19	9.599	5.073	**1.759**	–
bio-celegans-dir	0.001	**0**	**0**	**0**	soc-dogster	**7.427**	9.707	–	–
fb-pages-food	0.001	**0**	**0**	**0**	bn-human-Jung2015_M87118347	926.687	**892.579**	–	–
bio-SC-TS	0.001	0.001	**0**	–	soc-twitter-higgs	**384.006**	427.345	717.939	–
bio-DM-LC	**0**	**0**	**0**	**0**	soc-youtube	1.579	**1.493**	–	–
socfb-Caltech36	0.002	0.002	**0**	0.02	web-it-2004	1.169	**0.664**	–	–
BA-1_10_60-L5	0.012	0.006	**0.005**	0.15	soc-flickr	**5.324**	11.946	–	–
ia-crime-moreno	0.001	**0**	**0**	0.06	soc-delicious	**1.088**	1.392	–	–
bio-CE-GT	0.001	**0**	**0**	0.61	soc-FourSquare	2.18	1.507	**1.152**	–
gene	0.001	0.001	**0**	**0**	bn-human-Jung2015_M87116517	**972.048**	981.889	–	–
mammalia-voles-plj-trapping	0.005	**0.002**	0.005	638.21	bn-human-BNU_1_0025878_session_1-bg	969.876	**968.036**	–	–
bio-yeast	0.001	0.001	**0**	**0**	bn-human-Jung2015_M87104201	972.712	**945.369**	–	–
bio-yeast-protein-inter	0.001	0.001	**0**	**0**	bn-human-BNU_1_0025874_session_2-bg	976.101	**975.953**	–	–
bio-CE-PG	**0.026**	0.035	–	–	soc-digg	3.453	2.326	**2.08**	–
fb-messages	0.003	0.003	**0**	0.02	kron_g500-logn20	25.652	14.377	**4.697**	–
bio-grid-fission-yeast	0.005	0.004	**0**	0.31	ca-IMDB	241.238	**228.911**	–	–
tech-routers-rf	0.002	0.002	**0.001**	0.64	rt-retweet-crawl	2.863	1.662	**1.334**	2.77
bio-CE-GN	0.019	0.012	**0.009**	–	soc-youtube-snap	3.14	**2.631**	3.729	–
bio-HS-HT	0.006	0.005	**0**	–	soc-lastfm	2.078	2.326	**1.357**	16.63
bio-CE-HT	**0**	0.001	**0**	**0**	kron_g500-logn21	78.568	41.929	**19.113**	–

(*continued on next page*)

Table 8 (*continued*).

Instance	DmDS	FastDS	ScBppw	EMOS	Instance	DmDS	FastDS	ScBppw	EMOS
bio-DM-HT	0.005	0.004	**0.002**	–	soc-pokec	–	**940.477**	–	–
bio-DR-CX	**0.359**	0.666	2.071	–	tech-as-skitter	**489.562**	720.78	–	–
bio-grid-worm	0.001	0.002	**0**	**0**	soc-flickr-und	10.336	**7.794**	–	–
bio-DM-CX	**0.107**	0.135	8.192	–	web-wikipedia-growth	**486.887**	550.192	–	–
ca-GrQc	0.007	0.004	**0.001**	0.06	web-baidu-baike	25.677	**23.666**	–	–
bio-HS-LC	**0.013**	0.015	0.03	–	tech-ip	611.044	**293.681**	–	–
web-EPA	0.004	0.004	**0**	0.25	soc-flixster	3.45	4.112	**2.195**	32.61
bio-HS-CX	0.218	**0.142**	2.076	–	socfb-B-anon	24.674	15.171	**12.04**	–
web-spam	0.017	0.011	**0.004**	2.89	socfb-A-anon	30.082	**27.239**	41.825	–
ca-Erdos992	0.002	0.002	0.002	**0**	wikipedia_link_en	38.313	22.6	**22.081**	–
soc-advogato	0.016	0.009	**0.001**	4.92	dblp-author	**16.355**	347.831	639.67	–
bio-grid-yeast	0.057	**0.046**	0.062	–	delicious-ti	31.237	20.971	**7.518**	–
soc-wiki-elec	0.035	0.019	**0.008**	0.27	ljournal-2008	**187.72**	445.812	–	–
bio-grid-fruitfly	0.007	0.007	**0.002**	0.01	soc-ljournal-2008	**127.606**	422.618	–	–
bio-dmela	**0.004**	0.006	**0.004**	–	soc-livejournal-user-groups	190.952	**126.395**	300.159	–
rec-movielens-user-movies-10 m	0.02	0.011	**0.004**	0.27	friendster	147.051	**80.066**	–	–
bio-grid-human	0.016	0.013	**0.004**	–	twitter_mpi	140.402	72.306	**56.794**	–
tech-pgp	0.012	**0.006**	0.01	–	socfb-uci-uni	409.628	237.207	**70.993**	–
Oregon-1	0.008	0.006	**0.003**	0.04					

Table 9
Comparison of initial solutions constructed by two approaches, where PCDS() and GCDS() represent PerturbedConstructDS() and GreedyConstructDS(), respectively.

Instance	PCDS()		GCDS()		Instance	PCDS()		GCDS()	
Name	Solution	Time	Solution	Time	Name	Solution	Time	Solution	Time
ENZYMES8	27	0.001	**26**	0.000	rec-libimseti-dir	13 000	3.075	**12 992**	2.968
ENZYMES296	45	0.001	**44**	0.000	coAuthorsCiteseer	33 336	0.151	**33 327**	0.134
soc-student-coop	51	0.001	**50**	0.000	amazon0302	**38 124**	0.200	38 130	0.185
bio-SC-TS	126	0.002	125	0.001	rgg_n_2_18_s0	28 521	0.232	**28 504**	0.182
bio-DM-LC	164	0.001	**163**	0.001	citationCiteseer	**43 971**	0.311	43 984	0.293
bio-CE-GT	**127**	0.003	128	0.002	web-Stanford	13 968	0.349	**13 953**	0.328
mammalia*trapping	**283**	0.002	284	0.001	coAuthorsDBLP	44 124	0.224	**44 120**	0.204
bio-CE-PG	182	0.019	**181**	0.018	ca-dblp-2012	**46 277**	0.232	46 279	0.205
bio-grid-fission-yeast	**281**	0.002	282	0.002	bn-human-*M87119044	7891	12.232	**7877**	11.666
tech-routers-rf	482	0.003	481	0.002	cnr-2000	**22 079**	0.280	22 087	0.259
bio-CE-GN	204	0.006	**202**	0.005	web-NotreDame	23 777	0.151	**23 774**	0.132
bio-HS-HT	**461**	0.003	462	0.002	bn-human-*M87116523	**7504**	14.141	7512	14.147
bio-DR-CX	341	0.011	**338**	0.010	ca-MathSciNet	65 659	0.216	**65 645**	0.191
bio-DM-CX	**536**	0.009	538	0.010	amazon0312	**47 885**	0.505	47 910	0.481
bio-HS-LC	**384**	0.006	385	0.005	amazon0601	**44 824**	0.519	44 856	0.504
web-EPA	266	0.004	264	0.003	kron_g500-logn19	**27 751**	4.164	27 752	3.926
web-spam	**835**	0.011	836	0.010	amazon0505	**49 692**	0.521	49 718	0.506
soc-advogato	812	0.015	**810**	0.014	soc-dogster	26 404	1.551	**26 396**	1.472
bio-grid-yeast	**299**	0.021	304	0.018	bn-human-*M87118347	**9048**	13.867	9052	15.092
bio-grid-fruitfly	**1522**	0.006	1524	0.005	coPapersCiteseer	27 359	1.466	**27 345**	1.463
bio-dmela	**1453**	0.006	1454	0.005	soc-twitter-higgs	**15 054**	2.547	15 066	2.586
bio-grid-human	1792	0.007	**1789**	0.006	soc-youtube	**89 767**	0.531	89 772	0.482
p2p-Gnutella04	2249	0.006	**2247**	0.005	web-it-2004	33 003	0.503	**33 001**	0.477
bio-human-gene2	419	0.725	**418**	0.739	soc-flickr	**98 092**	0.656	98 102	0.598
bio-CE-CX	2681	0.033	**2669**	0.030	rgg_n_2_19_s0	54 466	0.437	**54 424**	0.427
bio-WormNet-v3	**2183**	0.091	2184	0.089	soc-delicious	**55 758**	0.366	55 759	0.324
rec-movie*tag*10 m	**2993**	0.030	2994	0.028	ca-coauthors-dblp	37 739	1.622	**37 726**	1.601
ca-AstroPh	2121	0.021	**2117**	0.020	coPapersDBLP	**37 743**	1.632	37 746	1.611
ca-CondMat	**3032**	0.017	3035	0.015	bn-human-*M87110650	30 108	6.786	**30 097**	6.876
bio-human-gene1	**879**	1.038	880	1.026	soc-FourSquare	61 015	0.878	**61 013**	0.804
p2p-Gnutella25	4524	0.009	**4523**	0.008	bn-human*25873*_1-bg	22 094	17.030	**22 090**	17.979
cit-HepTh	**1061**	0.212	1063	0.209	bn-human-*M87124152	**19 542**	30.825	19 611	30.198
cit-HepPh	**960**	0.271	965	0.269	web-BerkStan	**29 523**	0.695	29 541	0.655
rgg_n_2_15_s0	4189	0.018	**4186**	0.016	bn-human*25896*_2-bg	24 187	12.883	**24 154**	13.718
p2p-Gnutella30	7171	0.016	**7170**	0.014	bn-human*25869*_1-bg	**26 345**	17.822	26 347	17.570
cond-mat-2005	5722	0.024	**5718**	0.022	bn-human-*M87116517	**22 120**	27.513	22 125	27.647
c-62ghs	**15 194**	0.028	15 196	0.026	bn-human*25914*_1-bg	**28 251**	13.629	28 259	14.282
bio-mouse-gene	**2837**	1.746	2847	1.484	bn-human*25864*_1-bg	26 387	19.222	**26 337**	19.254
c-66b	**21 193**	0.026	21 197	0.024	bn-human-*M87124670	**29 437**	7.396	29 439	6.644
kron_g500-logn16	**3885**	0.329	3887	0.315	bn-human*25878*_1-bg	26 378	15.334	**26 359**	16.307
rgg_n_2_16_s0	**7888**	0.038	7903	0.034	bn-human*25914*_2	**27 510**	16.953	27 521	17.632
soc-Slashdot0902	15 310	0.082	**15 309**	0.076	bn-human*25889*_1	**26 243**	17.245	26 270	18.275
soc-buzznet	**132**	0.386	133	0.372	bn-human*25917*_1	**24 903**	18.050	24 947	17.998
soc-LiveMocha	1470	0.316	**1468**	0.294	bn-human-*M87104201	18 172	36.854	**18 125**	37.286
kron_g500-logn17	**7470**	0.797	7471	0.763	bn-human*25916*_1	26 078	20.488	**26 065**	17.291
luxembourg_os	40 169	0.028	**40 137**	0.023	bn-human*25869*_2-bg	**28 227**	20.039	28 260	19.826
rgg_n_2_17_s0	15 000	0.087	**14 989**	0.076	bn-human*25890*_2	**26 721**	19.507	26 795	20.023
sc-shipsec1	9267	0.178	**9261**	0.168	bn-human-*M87115663	21 515	38.623	**21 461**	38.154
web-arabic-2005	17 007	0.135	**16 998**	0.119	bn-human-*M87124029	**31 005**	7.393	31 010	7.073

(*continued on next page*)

Table 9 (*continued*).

Instance	PCDS()		GCDS()		Instance	PCDS()		GCDS()	
sc-shipsec5	**12 513**	0.236	12 557	0.226	bn-human-*M87127186	**20 255**	35.611	20 292	36.168
caidaRouterLevel	**40 907**	0.142	40 929	0.130	bn-human*25913*_2	**25 437**	22.590	25 458	23.534
soc-academia	28 610	0.234	**28 592**	0.217	bn-human*25868*_1-bg	**27 543**	20.935	27 558	20.854
kron_g500-logn18	**14 269**	1.859	14 271	1.747	bn-human*25868*_2-bg	**28 078**	22.322	28 089	21.883
sc-pwtk	**4704**	0.340	4709	0.320	bn-human-*M87101698	**23 988**	25.383	24 027	25.335
bn-human*25911*_2	28 427	17.791	**28 409**	19.042	bn-human-*093	**26 004**	23.094	26 014	23.608
bn-human-*M87105966	**24 661**	29.072	24 670	29.768	bn-human-*525	18 473	25.081	**18 461**	25.324
bn-human*25865*_1-bg	24 410	22.290	**24 386**	22.455	rgg_n_2_20_s0	**104 388**	1.276	104 399	1.255
bn-human*25871*_2-bg	**25 965**	20.847	25 970	21.851	ca-hollywood-2009	**51 409**	9.349	51 437	9.035
bn-human-*M87104509	**31 413**	9.289	31 440	8.679	inf-roadNet-PA	344 442	0.712	**344 398**	0.534
bn-human*25867*_1-bg	28 573	20.013	**28 564**	20.124	soc-youtube-snap	**213 131**	1.129	213 135	1.015
bn-human*25918*_1	**27 466**	20.545	27 535	20.855	in-2004	77 997	1.475	**77 990**	1.350
bn-human-*M87108808	**25 510**	35.519	25 525	35.411	belgium_osm	503 400	0.743	**501 650**	0.673
bn-human-*M87125286	**22 401**	42.058	22 407	42.424	kron_g500-logn21	**102 246**	25.065	102 248	26.030
rec-epinion	9057	3.524	**9055**	3.366	soc-pokec	**213 224**	8.133	213 324	8.691
bn-human-*M87110148	23 523	30.698	**23 499**	28.078	tech-as-skitter	**183 884**	2.916	183 889	2.697
bn-human-*M87110670	**30 011**	8.022	30 012	7.893	soc-flickr-und	**295 751**	4.091	295 758	3.868
bn-human-*M87125334	**34 962**	6.482	35 064	6.111	web-wikipedia2009	**347 849**	2.402	347 917	2.277
bn-human*25874*_2-bg	28 677	20.769	**28 675**	21.882	web-wiki*growth	**117 807**	15.423	117 809	14.910
soc-digg	**66 174**	1.337	66 178	1.270	inf-roadNet-CA	**619 587**	1.117	620 043	0.933
bn-human-*M87101705	21 453	53.213	**21 448**	49.726	rgg_n_2_21_s0	**199 980**	2.947	200 053	2.730
bn-human-*M87124563	19 306	31.539	**19 249**	33.387	web-baidu-baike	**277 019**	9.583	277 024	9.187
bn-human*25876*_2-bg	33 025	16.994	**33 000**	17.808	tech-ip	**176**	5.244	177	5.096
bn-human-*M87125521	34 982	7.748	**34 943**	7.169	memchip	**479 145**	2.084	479 158	1.950
bn-human*25886*_1	**27 749**	19.993	27 799	20.884	socfb-B-anon	187 069	10.239	**187 064**	9.886
bn-human-*M87107242	36 153	9.212	**36 118**	8.620	soc-orkut	**119 979**	53.192	120 062	52.525
bn-human*25912*_2	28 096	17.443	**28 062**	18.610	soc-orkut-dir	100 639	72.095	**100 557**	71.883
bn-human-*M87113878	**16 467**	62.757	16 470	60.195	socfb-A-anon	**201 827**	12.328	201 832	11.944
bn-human-*M87121714	**32 745**	8.413	32 815	8.203	wikipedia_link_en	212 878	14.861	**212 876**	14.428
bn-human*25876*_1-bg	**33 023**	17.369	33 033	17.889	Freescale1	608 329	2.782	**608 162**	2.581
bn-human-*M87118219	**26 292**	23.245	26 316	23.599	patents	**653 953**	11.875	653 967	11.712
bn-human-*M87127677	26 478	45.005	**26 448**	42.335	cit-Patents	**641 775**	15.070	641 946	16.107
bn-human-*M87118759	**32 969**	10.436	32 988	10.620	delicious-ti	140 725	12.511	**140 723**	12.323
bn-human*25870*_1-bg	**31 272**	20.766	31 276	20.560	soc-livejournal	**796 756**	16.245	796 910	15.828
bn-human-*M87125691	36 231	8.042	**36 230**	7.465	rgg_n_2_22_s0	385 079	6.192	**384 882**	5.759
bn-human-*M87111392	**26 392**	38.040	26 402	37.394	delaunay_n22	**751 637**	4.741	751 665	4.507
bn-human-*M87109786	33 794	9.510	**33 772**	10.024	channel*-b050	**434 264**	7.336	434 313	7.266
bn-human-*M87102230	33 268	12.465	**33 261**	10.875	cage15	503 412	30.221	**503 381**	25.954
bn-human*25870*_2-bg	**30 652**	23.526	30 690	23.999	ljournal-2008	1 008 173	33.097	**1 007 911**	31.118
bn-human-*M87128194	26 530	25.114	**26 521**	25.687	soc-ljournal-2008	1 008 211	28.282	**1 007 895**	27.316
bn-human-*M87129974	36 857	9.099	**36 828**	9.167	sc-rel9	143 570	26.014	**143 244**	25.816
bn-human-*M87125330	**34 915**	9.187	35 001	8.754	indochina-2004	401 559	15.083	**401 524**	14.284
bn-human-*M87119472	31 168	11.373	**31 161**	11.640	soc-livejo*groups	**1 071 441**	53.229	1 071 446	52.440
bn-human-*M87103674	33 791	10.547	**33 718**	10.878	rgg_n_2_23_s0	**741 341**	14.116	741 546	13.555
bn-human-*M87118954	22 917	12.496	**22 895**	12.430	delaunay_n23	1 502 976	10.717	**1 502 944**	10.292
bn-human-*M87123142	34 887	9.050	**34 880**	8.972	friendster	656 689	36.958	**656 688**	35.978
bn-human-*M87125989	37 310	8.943	**37 298**	8.840	wb-edu	836 877	9.882	**836 640**	8.624
bn-human-*M87104300	28 896	11.016	**28 879**	10.971	inf-germany_os	4 054 144	8.876	**4 035 903**	7.616
bn-human-*M87127667	**22 967**	12.790	22 984	12.771	dbpedia-link	**1 968 854**	95.694	1 968 864	92.473
bn-human-*M87128519	24 381	34.280	**24 340**	33.890	inf-road_central	**4 754 295**	16.641	4 754 615	15.449
eu-2005	32 415	1.672	**32 413**	1.682	road_central	**4 754 295**	18.446	4 754 615	16.238
bn-human-*M87113679	33 590	11.447	**33 551**	10.654	rgg_n_2_24_s0	1 430 548	31.453	**1 430 305**	30.846
web-Google	**80 047**	1.084	80 059	1.008	delaunay_n24	**3 004 962**	28.032	3 005 258	25.657
bn-human-*M87115834	**33 945**	10.227	33 961	9.689	uk-2002	**1 043 790**	35.016	1 043 969	33.128
bn-human-*M87117515	**33 690**	8.270	33 695	8.378	inf-road-usa	8 093 641	26.477	**8 081 092**	22.328
bn-human-*M87123456	38 037	9.129	**37 970**	9.283	inf-europe_os	17 810 003	47.382	**17 757 076**	41.144
ca-IMDB	**120 791**	1.453	120 826	1.382					
bn-human-*M87122310	**19 852**	14.124	19 860	14.383					
bn-human-*M87102575	**21 114**	13.882	21 139	13.740					

CRediT authorship contribution statement

Enqiang Zhu: Conceptualization, Supervision, Writing – original draft, Writing – review & editing. **Yu Zhang:** Data curation, Methodology, Software, Writing – original draft. **Shengzhi Wang:** Formal analysis, Visualization. **Darren Strash:** Formal analysis, Methodology, Writing – review & editing. **Chanjuan Liu:** Writing – original draft, Writing – review & editing.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants 62272115, 62172072.

Appendix. Tables

See Tables 6–9.

References

[1] F. Wang, E. Camacho, K. Xu, Positive influence dominating set in online social networks, COCOA 9 (2009) 313–321.
[2] D. Zhao, G. Xiao, Z. Wang, L. Wang, L. Xu, Minimum dominating set of multiplex networks: Definition, application, and identification, IEEE Trans. Syst. Man Cybern. 51 (12) (2020) 7823–7837.
[3] J.C. Nacher, T. Akutsu, Minimum dominating set-based methods for analyzing biological networks, Methods 102 (2016) 57–63.
[4] S. Wuchty, Controllability in protein interaction networks, Proc. Natl. Acad. Sci. 111 (19) (2014) 7156–7160.
[5] M.R. Garey, D.S. Johnson, Computers and Intractability, vol. 174, freeman San Francisco, 1979.
[6] F.V. Fomin, D. Kratsch, G.J. Woeginger, Exact (exponential) algorithms for the dominating set problem, Lecture Notes in Comput. Sci. (2004) 245–256.
[7] F. Grandoni, A note on the complexity of minimum dominating set, J. Discrete Algorithms 4 (2) (2006) 209–214.
[8] I. Schiermeyer, Efficiency in exponential time for domination-type problems, Discrete Appl. Math. 156 (17) (2008) 3291–3297.
[9] J.M. van Rooij, H.L. Bodlaender, Design by measure and conquer, a faster exact algorithm for dominating set, in: STACS 2008, IBFI Schloss Dagstuhl, 2008, pp. 657–668.
[10] J.M. Van Rooij, H.L. Bodlaender, Exact algorithms for dominating set, Discrete Appl. Math. 159 (17) (2011) 2147–2164.
[11] Y. Iwata, A faster algorithm for dominating set analyzed by the potential method, in: Parameterized and Exact Computation: 6th International Symposium, IPEC 2011, SaarbrÜCken, Germany, September 6-8, 2011. Revised Selected Papers 6, Springer, 2012, pp. 41–54.
[12] F. Grandoni, Exact algorithms for hard graph problems (Ph.D. thesis), Universita di Roma "Tor Vergata", Roma, Italy, 2004.
[13] S. Cai, W. Hou, Y. Wang, C. Luo, Q. Lin, Two-goal local search and inference rules for minimum dominating set, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 1467–1473.
[14] S. Lamm, P. Sanders, C. Schulz, D. Strash, R.F. Werneck, Finding near-optimal independent sets at scale, in: 2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments, ALENEX, SIAM, 2016, pp. 138–150.
[15] J. Dahlum, S. Lamm, P. Sanders, C. Schulz, D. Strash, R.F. Werneck, Accelerating local search for the maximum independent set problem, in: Experimental Algorithms: 15th International Symposium, SEA 2016, St. Petersburg, Russia, June 5-8, 2016, Proceedings 15, Springer, 2016, pp. 118–133.
[16] T. Akiba, Y. Iwata, Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover, Theoret. Comput. Sci. 609 (2016) 211–225.
[17] E. Zhu, F. Jiang, C. Liu, J. Xu, Partition independent set and reduction-based approach for partition coloring problem, IEEE Trans. Cybern. 52 (6) (2022) 4960–4969, http://dx.doi.org/10.1109/TCYB.2020.3025819.
[18] C. Liu, S. Ge, Y. Zhang, Identifying the cardinality-constrained critical nodes with a hybrid evolutionary algorithm, Inform. Sci. 642 (2023) 119140, http://dx.doi.org/10.1016/j.ins.2023.119140.
[19] Y. Ma, Q. Cai, S. Yao, Integer linear programming models for the weighted total domination problem, Appl. Math. Comput. 358 (2019) 146–150, http://dx.doi.org/10.1016/j.amc.2019.04.038.
[20] R. Li, S. Liu, F. Wang, J. Gao, H. Liu, S. Hu, M. Yin, A restart local search algorithm with relaxed configuration checking strategy for the minimum k-dominating set problem, Knowl.-Based Syst. 254 (2022) 109619.
[21] A. Potluri, A. Singh, Two hybrid meta-heuristic approaches for minimum dominating set problem, in: Swarm, Evolutionary, and Memetic Computing: Second International Conference, SEMCCO 2011, Visakhapatnam, Andhra Pradesh, India, December 19-21, 2011, Proceedings, Part II 2, Springer, 2011, pp. 97–104.
[22] A.R. Hedar, R. Ismail, Hybrid genetic algorithm for minimum dominating set problem, in: Computational Science and Its Applications–ICCSA 2010: International Conference, Fukuoka, Japan, March 23-26, 2010, Proceedings, Part IV 10, Springer, 2010, pp. 457–467.
[23] S. Alharbi, I. Venkat, A genetic algorithm based approach for solving the minimum dominating set of queens problem, J. Optim. 2017 (2017).
[24] A.-R. Hedar, R. Ismail, Simulated annealing with stochastic local search for minimum dominating set problem, Int. J. Mach. Learn. Cybern. 3 (2012) 97–109.
[25] Y. Fan, Y. Lai, C. Li, N. Li, Z. Ma, J. Zhou, L.J. Latecki, K. Su, Efficient local search for minimum dominating sets in large graphs, in: Database Systems for Advanced Applications: 24th International Conference, DASFAA 2019, Chiang Mai, Thailand, April 22–25, 2019, Proceedings, Part II, Springer, 2019, pp. 211–228.
[26] S. Cai, K. Su, A. Sattar, Local search with edge weighting and configuration checking heuristics for minimum vertex cover, Artificial Intelligence 175 (9–10) (2011) 1672–1696.
[27] Y. Wang, S. Cai, M. Yin, Local search for minimum weight dominating set with two-level configuration checking and frequency based scoring function, J. Artificial Intelligence Res. 58 (2017) 267–295.
[28] Y. Wang, S. Cai, J. Chen, M. Yin, A fast local search algorithm for minimum weight dominating set problem on massive graphs, in: IJCAI, 2018, pp. 1514–1522.
[29] H. Jiang, Z. Zheng, An exact algorithm for the minimum dominating set problem, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI, 2023, pp. 5604–5612.
[30] D.V. Andrade, M.G. Resende, R.F. Werneck, Fast local search for the maximum independent set problem, J. Heuristics 18 (2012) 525–547.
[31] S.A. Abed, H.M. Rais, J. Watada, N.R. Sabar, A hybrid local search algorithm for minimum dominating set problems, Eng. Appl. Artif. Intell. 114 (2022) 105053.
[32] S. Cai, Balance between complexity and quality: Local search for minimum vertex cover in massive graphs, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[33] D. Chalupa, An order-based algorithm for minimum dominating set with application in graph mining, Inform. Sci. 426 (2018) 101–116.
[34] K.B. Xu, Benchmarks with hidden optimum solutions for graph problems, 2007, URL http://www.nlsde.buaa.edu.cn/kexu/benchmarks/graph-benchmarks.htm.
[35] D.A. Bader, H. Meyerhenke, P. Sanders, D. Wagner (Eds.), Graph Partitioning and Graph Clustering, 10th DIMACS Implementation Challenge Workshop, Georgia Institute of Technology, Atlanta, GA, USA, February 13-14, 2012. Proceedings, in: Contemporary Mathematics, vol. 588, American Mathematical Society, 2013, http://dx.doi.org/10.1090/conm/588.
[36] R.A. Rossi, N.K. Ahmed, The network data repository with interactive graph analytics and visualization, in: AAAI, 2015.