An Efficient Greedy-Randomized Heuristic for the Minimum Weakly Connected Dominating Set Problem

Kaushal Kumar Ojha[†], Tara Hemaliya[†], and Sachchida Nand Chaurasia[⋆,†][0000−0002−0635−0808]

Department of Computer Science, Institute of Science,
Banaras Hindu University, Varanasi, 221005, India.
{kkojha, tarakushwaha, snchaurasia}@bhu.ac.in

Abstract. The Minimum Weakly Connected Dominating Set Problem (MWCDSP) seeks the smallest weakly connected dominating set (WCDS) on an undirected connected graph where each vertex is either in the WCDS or a neighbor of at least one node in the set, and also each node in the WCDS is at a maximum distance of two hops from at least one other node in the set. MWCDSP is a $\mathcal{NP}$-hard problem. The MWCDSP has applications in network design and wireless communication. This paper proposes a greedy randomized heuristic to obtain near-optimal solutions with very low computational cost. The method is evaluated on multiple instances and compared with state-of-the-art approaches self-stabilizing algorithm (SMWCDS), the Memetic Algorithm (MA) and the Greedy Randomized Adaptive Search Procedure (GRASP). The experimental results demonstrate that the proposed approach outperforms SMWCDS in both the best and average cases across all instances, and when compared to MA and GRASP, it achieves better or equal performance on small and medium instances, while on large instances, both metaheuristics outperform it due to their broader search capability, but at the cost of higher computation time. The proposed heuristic is simple, fast and effective, making it well-suited for scenarios that require quick and reasonably good solutions under limited computational resources.

Keywords: Combinatorial optimization, Minimum weakly connected dominating set problem, $\mathcal{NP}$-hard problem, Greedy heuristic

1 Introduction

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected connected graph. The set $\mathcal{V}$ contains all nodes and $\mathcal{E}$ contains all edges. A set of vertices is said to be a dominating set if each vertex in the graph is either a member of the set or adjacent to at least one vertex in the set. Mathematically, the dominating set is defined as follows.

$$\mathcal{D}_{set} \subseteq \mathcal{G} \tag{1}$$

$$\forall u \in \mathcal{V},\ u \in \mathcal{D}_{Set} \ \vee\ \exists v \in \mathcal{D}_{Set} \text{ such that } (u, v) \in \mathcal{E} \tag{2}$$

⋆ Corresponding author, Supported by IoE-BHU

† These authors contributed equally.

A dominating set $\mathcal{D}_{set}$ is called a connected dominating set (CDS) if all vertices in $\mathcal{D}_{set}$ are mutually reachable within the subgraph induced by the set. This implies that a path exists between any two vertices within $\mathcal{D}_{set}$. Thus, the vertices in $\mathcal{D}_{set}$ not only dominate the graph but also form a single connected component. Equation (3) describes the connected dominating set. It follows that each vertex is either part of the dominating set or is within one hop of at least one dominating vertex. Formally, the CDS is defined as:

$$\begin{aligned} &(\forall u \in \mathcal{V},\, u \in \mathcal{D}_{set} \;\vee\; \exists v \in \mathcal{D}_{set} : (u,v) \in \mathcal{E}) \\ &\wedge\, (\forall u, v \in \mathcal{D}_{set}),\, \exists \text{ a path from } u \text{ to } v \text{ entirely in } \mathcal{D}_{set} \end{aligned} \tag{3}$$

WCDS relaxes the connectivity requirement compared to CDS. A set $\mathcal{D}_{set}$ forms a weakly connected dominating set (WCDS) if it dominates the graph and the vertices in the set satisfy the condition that each vertex is either adjacent to another vertex in the set or separated by no more than two hops from some vertex in $\mathcal{D}_{set}$. The objective is to find a weakly connected dominating set with minimum number of vertices. We use the notation $hop(u,v)$ to denote the distance (in hops) between nodes u and v:

- $hop(u,v) = 1$ means u and v are direct neighbors.
- $hop(u,v) = 2$ means u and v are two hops apart.

In this work, we primarily consider hop distances of at most two. Equation (4) describes that for any given node in the WCDS, either the node is at one hop or two hop distance of any other node in the $\mathcal{D}_{set}$. The MWCDS can be formally defined as follows:

$$\begin{aligned} &(\forall u \in \mathcal{V},\, u \in \mathcal{D}_{set} \;\vee\; \exists v \in \mathcal{D}_{set} : (u,v) \in \mathcal{E}) \\ &\wedge (\forall u \in \mathcal{D}_{set},\, \exists v \in \mathcal{D}_{set} : v \neq u \wedge hop(u,v) \leq 2) \end{aligned} \tag{4}$$

The utility of an MWCDS stems from its dual functionality, it not only covers all nodes in the graph, but it also simplifies network management. Due to its fundamental properties, an MWCDS typically consists of fewer critical (dominating) nodes, allowing efficient network control with minimal overhead. It has been established that the task of computing a minimum weakly connected dominating set is $\mathcal{NP}$-hard [9]. This challenge is compounded by the fact that real-world networks are dynamic and constantly evolving.

Formally, a subgraph S_w induced by a dominating set W is defined as $S_w = (\mathcal{N}[W], E \cap (W \times \mathcal{N}[W])$, where $\mathcal{N}[W]$ includes nodes in W and their one-hop neighbors. Set W is weakly connected if S_w forms a connected subgraph. Thus, our objective reduces to finding a WCDS W of minimum cardinality.

The concept of MWCDS was first introduced by Dunbar et al. in 1997 [9]. Since then, WCDS has been widely applied in clustering strategies for distributed sensor networks [11] and mobile ad hoc networks [8,12,20]. Research by Yu et al. [27] suggests that a WCDS includes lesser nodes than a CDS while being no more computationally challenging to compute, making it an efficient alternative for clustering. Studies by Han et al. [12] further demonstrate its effectiveness in cluster formation, while Pathan et al. [20] and Du et al. [8] highlight its utility in reducing cluster head density and addressing secure clustering challenges in distributed sensor networks.

The minimum weakly connected dominating set problem in general graphs is known to be $\mathcal{NP}$-hard [9]. Researchers have thoroughly investigated the difficulty in finding

MWCDSP on various kinds of graphs [7,9,22,24], underscoring its theoretical significance. To address this challenge, self-stabilizing algorithms have been developed to efficiently construct MWCDS in connected graphs. In [6], Ding et al. developed a self-stabilizing solution for this problem that terminates in $O(n)$ steps for an n-node network. However, applying such methods to complex networks—which often contain multiple locally optimal solutions—remains difficult. Recently, Niu et al. [18] introduced a self-stabilizing memetic algorithm (MA) for MWCDSP, leveraging the algorithm SMWCDS to refine infeasible solutions during the search process.

While heuristics and exact algorithms have been extensively applied to variants of the dominating set problems [13,15,19,1,16] in real-world graphs. The GRASP approach, put forward by Feo and Resende in 1989, has been widely used to obtain near-optimal solutions in combinatorial optimization [19,21,23,5,10,15,26]. Heuristic techniques addressing related optimization problems—including the minimum weight dominating set [3], the dominating tree problem [3], and the MWCDSP [17]—also offer valuable perspectives. These advancements motivate the development of more efficient heuristics and metaheuristics for high-quality solutions. We propose an efficient greedy randomized heuristic for MWCDSP, building upon the heuristic introduced in [2]. The proposed approach leverages the principle of weak connectivity while prioritizing nodes that maximize coverage. A purely greedy approach may get stuck in poor solutions, so we added measures to avoid this problem. The proposed method selects a dominating node based on its potential to cover the maximum number of its one-hop neighbors. To avoid local optima, it incorporates randomness by choosing a node from a set of high-potential candidates. The computational experiments on test instances demonstrate the effectiveness of our heuristic compared to IBM CPLEX Solver, the memetic algorithm (MA) [18], GRASP [17] (the current best-performing approach), and the SMWCDS [6].

The layout of paper: Section 2 introduces details of the problem definition and integer linear programming (ILP) formulation for MWCDSP. Section 3 outlined the suggested heuristic framework. Section 4 discusses the performance and analysis of the proposed approach on different sets of instances. Finally, Section 5 provides a conclusion to the paper, recaps the key contributions, and identifies future research prospects.

2 Problem Definition

This section presents the MWCDSP formulation as outlined in [17]. Consider a connected, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. Let $\mathcal{N}_i(v)$ denote the set of nodes in $\mathcal{V}$ that are exactly i hops away from node v. We define $\mathcal{E}_{weak}$ as an adjacency matrix representing edges that connect nodes in $\mathcal{V}$ such that for any node $u \in \mathcal{V}$, $\mathcal{E}_{weak}$ includes connections to nodes in $\mathcal{N}_2(v)$. For every pair of vertices in the graph $(u, v) \in \mathcal{V}$, $\mathcal{E}_{weak}[u, v]$ is 1 if an edge exists between vertices u and v, and 0 otherwise. $\mathcal{S} = \{e_1, e_2, \ldots, e_k\}$ (where $\forall e_i \subseteq \mathcal{E}_{weak}$) denote a candidate solution set. x_{uv} is a binary decision variable that equals 1 if the edge between nodes u and v is selected in the solution, and 0 otherwise. The binary variable y_t is defined such that it takes the value 1 when vertex t is selected in the solution, and 0 otherwise. The ILP of the problem is mentioned below:

$$\min \mathcal{F}(\mathcal{S}) = 1 + \sum_{(u,v)\in\mathcal{V}} x_{uv} \tag{5}$$

subject to,

$$x_{uv} \leq \mathcal{E}_{weak}[u,v], \forall u,v \in \mathcal{V} \tag{6a}$$

$$x_{uv} = x_{vu}, \forall u,v \in \mathcal{V} \tag{6b}$$

$$y_u + y_v \geq 2 \times x_{uv}, \forall u,v \in \mathcal{V} \tag{6c}$$

$$\sum_{u,v\in\mathcal{S}} x_{uv} \leq |\mathcal{S}| - 1, \mathcal{S} \subset \mathcal{V} \& \sum_{u,v\in\mathcal{S}} \mathcal{E}_{weak}[u,v] > |\mathcal{S}| - 1$$

$$\& |\mathcal{S}| \leq \text{UB} \tag{6d}$$

$$\sum_{u,v\in\mathcal{V}} x_{uv} = \sum_{t\in\mathcal{V}} y_t - 1 \tag{6e}$$

$$\sum_{t\in N[v]} y_t \geq 1, \forall v \in \mathcal{V} \tag{6f}$$

The objective function in Equation (5) minimizes the total number of vertices included in the WCDS. The corresponding spanning tree is composed of edges with $x_{uv} = 1$. For a connected graph, the vertex count can be derived by adding one to the total number of edges present in its spanning tree, as established in Equation (5). The Constraints Equations (6a) to (6f) guarantee that all selected edges in the solution belong to $\mathcal{E}_{weak}$, and that the solution satisfies the connectivity and domination requirements.

The third constraint, Equation (6c), ensures the connectivity of vertices in an optimal solution, ensuring that each selected edge includes its corresponding end vertices. Constraints Equation (6d) and Equation (6e) employ a minimum spanning tree formulation to guarantee a valid tree-based solution. Unlike [25], our formulation introduces two additional subset-selection criteria in Equation (6d).

A solution is a valid MWCDS only if it satisfies all the constraints, including cycle elimination, which requires $\sum_{\{u,v\}\in\mathcal{S}} \mathcal{E}_{weak}[u,v] > |\mathcal{S}| - 1$ for any subset $\mathcal{S} \subseteq \mathcal{V}$, and a cardinality bound that restricts the size to $|\mathcal{S}| \leq$ UB (UB denoted a upper bound).

3 Proposed Heuristic

This section presents a greedy randomized heuristic algorithm, called GrH, inspired by [3,2,4]. The closed neighborhood search strategy is based on two principles. It permits non-dominating nodes that are already covered to be candidates for the dominating set, as a node with a higher degree can be more competent than a white node, potentially yielding a better solution. Furthermore, it dynamically updates the neighborhood count of non-dominating nodes after each iteration to ensure an accurate assessment of their selection potential in the future course of selection.

GrH combines greedy selection with guided randomized exploration. This greedy mechanism prioritizes nodes that can cover the greatest number of vertices, with the goal of minimizing the cardinality of the resulting set, while randomization introduces

diversity for efficient search space exploration. The algorithm executes as follows. Firstly, it randomly selects a vertex $v \in \mathcal{W}_n$ as a dominating node. It then updates node states: setting color(v) = BLACK and color(u) = GREY for all $u \in \mathcal{ON}(v)$. The node v is removed from the working graph, $\mathcal{W}_n = \mathcal{W}_n \setminus \{v\}$, and added to the dominating set $\mathcal{D}_{\text{set}} = \mathcal{D}_{\text{set}} \cup \{v\}$.

For each remaining vertex v, we compute its potential set $\mathcal{P}_{\text{set}}(v)$ as all white vertices within 2 hops: $\mathcal{P}_{\text{set}}(v) = \{u \in \mathcal{V} \mid \text{color}(u) = \texttt{WHITE} \wedge d(u,v) \leq 2\}$, where $\mathcal{ON}(v)$ denotes v's neighborhood and $d(u,v)$ represents the distance between u and v.

$$\begin{aligned} \mathcal{P}_{set} &= \left\{ u_i \in \Delta_n \;\middle|\; f(u_i) = \max_{u_j \in \Delta_n} f(u_j) \right\}, \\ \text{where} \quad f(u_j) &= \mathcal{NC}(u_j) - \left| \mathcal{ON}(u_j) \cap \mathcal{D}_{set} \right| \end{aligned} \tag{7}$$

Following the initial random node selection, our algorithm constructs a candidate set $\mathcal{P}_{set}$ that guides subsequent dominating nodes selection. This set construction represents a key innovation in our heuristic approach.

The process begins by creating Δ_n, which contains all WHITE nodes within two hops of the current dominating node. This dynamic set evolves as new dominating nodes are incorporated into the solution. $\mathcal{P}_{set}$ is derived from Δ_n by selecting nodes whose one-hop neighborhoods contain the maximum number of WHITE nodes. The selection mechanism incorporates a greedy strategy, prioritizing nodes that cover the most uncovered (WHITE) nodes. However, to maintain diversity in the search process, Algorithm 1 introduces randomness by selecting the next dominating node uniformly at random from a potential set $\mathcal{P}_{set}$. This balanced approach combines greedy optimization with stochastic exploration.

The complete mathematical formulation of this process appears in Equation (7), which precisely defines the set construction and selection criteria.

The neighbor count $\mathcal{NC}(v)$ represents the number of one-hop neighbors for vertex v. Our selection process identifies the next dominating node by choosing the vertex with maximum potential, where potential is measured as the count of adjacent WHITE nodes. The algorithm executes through successive iterations. First, it selects the node v^* from $\mathcal{P}_{set}$ that maximizes $\mathcal{NC}(v)$. Then, it updates node colors: the selected node v^* becomes BLACK while nodes in $\mathcal{ON}(v^*)$ turn GREY. Finally, these dominated nodes are removed from the working set $\mathcal{W}_n$. This procedure continues until $\mathcal{W}_n$ becomes empty, at which point all nodes have been recolored either BLACK (dominating nodes) or GREY (dominated nodes). The result is a valid WCDS. The complete procedure is formally described in Algorithm 1, which includes an example (Figure 2) demonstrating the node selection mechanism.

Algorithm 1 processes an input graph $\mathcal{W}_n = \mathcal{V}$, where $\mathcal{V} = v_1, \ldots, v_n$ denotes the vertex set. Each vertex is assigned one of three color states: WHITE (unprocessed), GREY (dominated), or BLACK (dominating). Initialization involves three steps: First, all vertices in $\mathcal{W}_n$ (visualized in Figure 1(a)) are set to WHITE. Second, the dominating set $\mathcal{D}_{set}$ is initialized as empty. Third, the candidate set $\mathcal{P}_{set}$ is likewise initialized empty. The algorithm proceeds iteratively, with each execution performing three key actions: selecting candidate vertices based on maximal coverage of WHITE neighbors, updating colors to reflect domination status (GREY for dominated, BLACK for dominator), and

Algorithm 1 Pseudo-code of Proposed Heuristic (GrH)

Initialization:
$\mathcal{W}_n \leftarrow \mathcal{V}$; ▷ All nodes are WHITE (unprocessed)
$\mathcal{D}_{set} \leftarrow \emptyset$; ▷ Initialize dominating set
$\Delta_n \leftarrow \emptyset$; ▷ Track 2-hop neighborhood
First Node Selection:
$v \leftarrow$ RandomSelect($\mathcal{W}_n$); ▷ Pick a random seed node
$\mathcal{D}_{set} \leftarrow \mathcal{D}_{set} \cup \{v\}$; ▷ Add to dominating set
Mark v as BLACK; ▷ Mark as dominator
$\mathcal{W}_n \leftarrow \mathcal{W}_n \setminus \{v\}$; ▷ Remove from unprocessed set
$\mathcal{D}_h \leftarrow \{u \mid u \in \mathcal{ON}\}$; ▷ Immediate neighbors of v
$\mathcal{W}_n \leftarrow \mathcal{W}_n \setminus \mathcal{D}_h$; ▷ Remove neighbors from unprocessed
Mark all $u \in \mathcal{D}_h$ GREY; ▷ Mark neighbors as dominated
$\Delta_n \leftarrow \mathcal{N}_2(\mathcal{D}_{set})$; ▷ Initialize 2-hop neighborhood
Main Loop (Greedy Selection):
while $\mathcal{W}_n \neq \emptyset$ **do**; ▷ While unprocessed nodes remain
    $\mathcal{P}_{set} \leftarrow$ Apply Equation (7) ; ▷ Select candidate nodes
    $v \leftarrow$ RandomSelect($\mathcal{P}_{set}$); ▷ Break ties randomly
    $\mathcal{D}_{set} \leftarrow \mathcal{D}_{set} \cup \{v\}$; ▷ Add to dominating set
    Mark v BLACK; ▷ Mark as dominator
    $\mathcal{W}_n \leftarrow \mathcal{W}_n \setminus \{v\}$; ▷ Remove from unprocessed
    $\mathcal{D}_h \leftarrow \{u \mid u \in \mathcal{ON}(v) \cap \mathcal{W}_n\}$; ▷ New WHITE neighbors
    $\mathcal{W}_n \leftarrow \mathcal{W}_n \setminus \mathcal{D}_h$; ▷ Mark them as processed
    Color all $u \in \mathcal{D}_h$ GREY; ▷ Dominated nodes
    $\Delta_n \leftarrow \Delta_n \setminus \{v \cup \mathcal{D}_h\}$; ▷ Update neighborhood
    $\Delta_n \leftarrow \Delta_n \cup (\mathcal{N}_2(v) \setminus \mathcal{D}_{set}) \cap \mathcal{W}_n$; ▷ Add new 2-hop neighbors
end while
Output:
return $\mathcal{D}_{set}$; ▷ Return the computed dominating set

expanding $\mathcal{D}_{set}$ while preserving solution feasibility. This process continues until $\mathcal{D}_{set}$ meets all criteria for a valid WCDS. The initialization phase begins with the random selection of vertex v_{13} via Algorithm 1, which is included in the empty dominating set $\mathcal{D}_{set}$. The algorithm then executes two color updates: v_{13} is marked BLACK as a dominating node, while its immediate neighbors $v_{12}, v_{10}, v_{11}, v_7$, and v_{14} are marked GREY to indicate their dominated status, as visualized in Figure 1(b). Following this assignment, the algorithm removes both v_{13} and its neighbors from the working graph $\mathcal{W}_n$. The subsequent step involves computing the potential for each remaining vertex using two criteria: (1) membership in the two-hop neighborhood of v_{13}, and (2) current WHITE status. These potentials are calculated according to Equation (7), which quantifies each vertex's coverage capability.

The algorithm selects vertex v_2 from the remaining graph $\mathcal{W}_n = \{v_0, v_1, v_4, v_5, v_3, v_8, v_9\}$ based on its maximal potential. Upon selection, v_2 is colored BLACK as a dominating node, while its neighbors $\{v_1, v_6, v_3\}$ are marked GREY and removed from $\mathcal{W}_n$, as illustrated in Figure 1(c). For cases of equal potential among vertices, the algorithm employs random selection. First, from equally potential vertices $\{v_9, v_5, v_8\}$, v_5 is randomly cho-

sen - marked BLACK with neighbor v_4 updated to GREY (Figure 1(d)). Next, among remaining equal-potential vertices $\{v_9, v_8, v_0\}$, v_9 becomes the subsequent dominating node (BLACK) with corresponding neighborhood updates shown in Figure 1(e). Finally, v_8 is selected from the remaining candidates, completing the dominating set formation as shown in Figure 1(f).

The final stage, in Figure 1(f), constructs a single remaining WHITE vertex v_0 within the two-hop neighborhood $\mathcal{N}_2$ of the current dominating set. The algorithm completes by marking v_0 as BLACK and adding it to the solution set, producing the final WCDS $\mathcal{D}_{set} = \{v_{13}, v_2, v_5, v_9, v_8, v_0\}$. The complete construction process, illustrated in Figures 1(b) to 1(g), demonstrates two key aspects through red dotted lines: the sequence of dominating node selections and the formation of the weakly induced subgraph $\mathcal{G}_{weak}$ as shown in Figure 1(h). This subgraph is formally defined as the pair $\mathcal{G}_{weak} = (\mathcal{V}_{dom}, \mathcal{E}_{weak})$, where $\mathcal{V}_{dom} = \bigcup_{d \in \mathcal{D}_{set}} \mathcal{N}[d]$ represents the union of closed neighborhoods of all dominating nodes, and $\mathcal{E}_{weak} = \mathcal{E} \cap \left(\bigcup_{d \in \mathcal{D}_{set}} \{d\} \times \mathcal{N}[d]\right)$ captures the induced edges between dominators and their neighbors.

Figure 1(h) depicts the final solution where the green arrows indicate the WCDS of the graph $\mathcal{G}$. For comparison, the SMWCDS [6] operates differently: it begins by randomly selecting an initial root vertex, then in subsequent iterations chooses vertices at exactly two-hop distance from the root to form the dominating set. When applied to small graphs like our test case, both heuristics may coincidentally select identical vertices due to the limited node count. However, two key distinctions emerge from the algorithm: First, our approach incorporates inherent randomness in node selection throughout the process, not just in initialization. Second, our selection criteria consider both neighborhood coverage and vertex potential. The worst-case time complexity the algorithm is $O(|\mathcal{V}|^2\Delta + |\mathcal{V}|\Delta^2)$.

The improved version of the algorithm, called GRH-IMP, modifies the selection process by continuously updating the information it uses. After every selection, it updates the neighborhood count for all non-dominating nodes. Then, as described by Equation (7), any of these updated nodes can become a candidate for the next dominating node, ensuring decision-making in any given scenario.

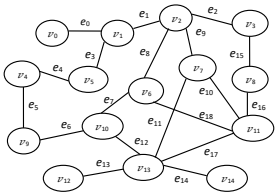
4 Computational Results

We compare the computational performance of the proposed method with the state-of-the-art approaches SMWCDS[6], MA[18], and GRASP[17]. The proposed heuristic, GRH, implementation uses C language compiled with gcc 11.4.0 under -O3 flag optimization. The experimental evaluation was performed on an Ubuntu 22.04 LTS platform featuring a 12th-Gen Intel® Core™ i7-12700 processor (2.10 GHz base frequency, 25MB cache) and 32GB DDR4 RAM. For performance evaluation, the approaches used four standard test instances: Random UDG [1], common UDG [14,15], LPNMR'09[2] and NDR[3] graphs. The description of all test instances is outlined in Table 1.

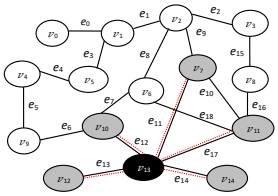
[1] https://github.com/niudd304/GRASP_WCDS

[2] https://dtai.cs.kuleuven.be/events/ASP-competition/encodings.shtml

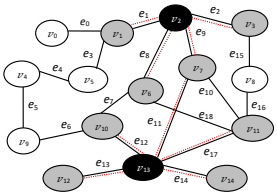
[3] https://networkrepository.com/networks.php



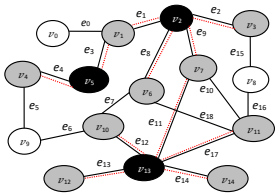
(a) Input graph. Initially, all nodes are colored WHITE



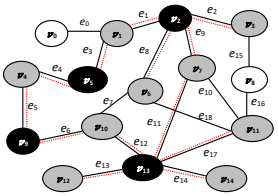
(b) Node v_{13} is selected randomly



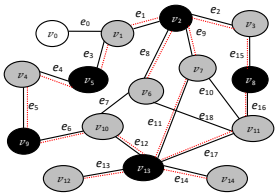
(c) Node v_2 is selected using the heuristic GRH



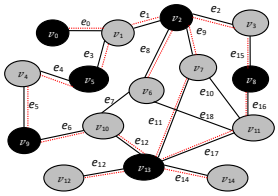
(d) Node v_5 is selected using heuristic GRH



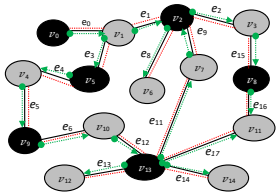
(e) Node v_9 is selected using the heuristic GRH



(f) Node v_8 is selected using the heuristic GRH



(g) Remaining node v_0 is selected using heuristic GRH



(h) Induced graph $\mathcal{E}_{weak} = \mathcal{E} \cap (\mathcal{D}_{set} \times \mathcal{N}[\mathcal{D}\text{set}])$

Fig. 1: Demonstrates the construction of $\mathcal{D}_{set}$ and selection of dominating node using Equation (7)

Table 1: Summary of Graph Test Instances

Graph Set	Instances	Nodes (Range)	Edges (Range)
Random UDG	24 (R_1 to R_{24})	15–30	18–49
Common UDG	41 (C_1 to C_{41})	80–400	262–1880
LPNMR'09	9 (L_1 to L_9)	40–150	200–500
NDR	24 (N_1 to N_{24})	100–21498	300–459277

The execution time varied significantly across algorithms. While MA and GRASP were limited to a fixed 300 seconds timeout per instance, SMWCDS completed in less than 0.01s. The GRH and GRH-IMP algorithms were even faster, averaging under 0.001s.

The subsequent subsections report the computation results and compare the performance analysis of the proposed heuristics with the considered approaches.

Figures 2 to 8 visually compare the percentage improvement by the proposed GRH heuristic over SMWCDS, MA, and GRASP algorithms. Each figure plots the percentage of improvement in the solution quality of the best case ($\%Best$) and the average case ($\%Avg.$) in all the test instances. In the figures, the abscissa enumerates the test instance, while the ordinate quantifies the relative improvement percentage, calculated as:

$$\%\text{Improvement} = \frac{Q_{ref} - Q_{\{\text{GRH, GRH-IMP}\}}}{Q_{ref}} \times 100$$

where Q_{GRH} and $Q_{\text{GRH-IMP}}$ denote the quality of the solution (best or average) of our heuristic, and Q_{ref} represents the corresponding value of the reference algorithm. The positive values demonstrate GRH's superior performance, with greater magnitudes indicating more significant improvements.

4.1 Comparative results of GRH with SMWCDS, MA, and GRASP on Common UDG Graph Test instances

Figure 2 shows that GRH delivers superior performance over SMWCDS, with improvements of 8.33% to 25% in terms of $\%Best$ solutions across various instances. For ($\%Avg.$) solutions, GRH achieves gains of 9.09% to 31.82% on several instances while matching SMWCDS on some instances. These results highlight GRH's consistency and significant advantage.

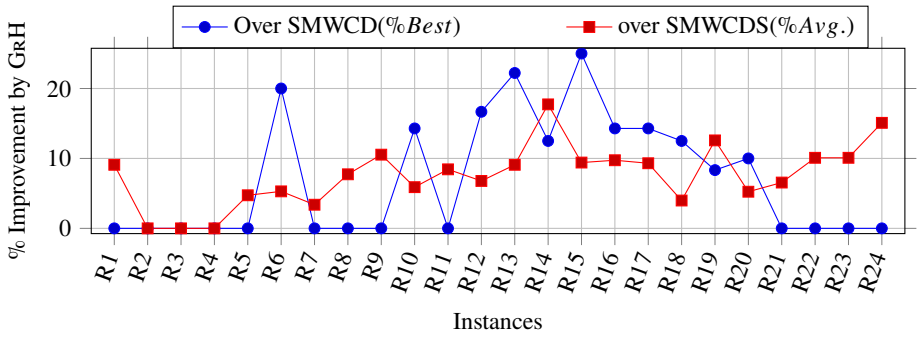


Fig. 2: Comparison of $\%Best$ and $\%Avg.$ results between GRH and SMWCDS on Random UDG Graph.

Across the 24 test instances in Figure 3, GRH matches MA in the best solutions ($\%Best$). For the percentage of average solutions ($\%Avg.$), GRH delivers identical results to MA on instances 80-15-15, 80-15-18, and 150-30-30, while remaining competitive on the remaining instances.

Compared to GRASP, GRH shows significant improvement of 15.38% and 14.29% in $\%Best$ on instances 150-30-28 and 100-20-23, respectively. It also outperforms GRASP by margins ranging from 9.09% to 14.29% on several other instances. In terms

of $\%Avg.$ performance, GRH surpasses GRASP on instances 100-20-23, 150-30-27, and 150-30-28, while delivering comparable results on the remaining instances.

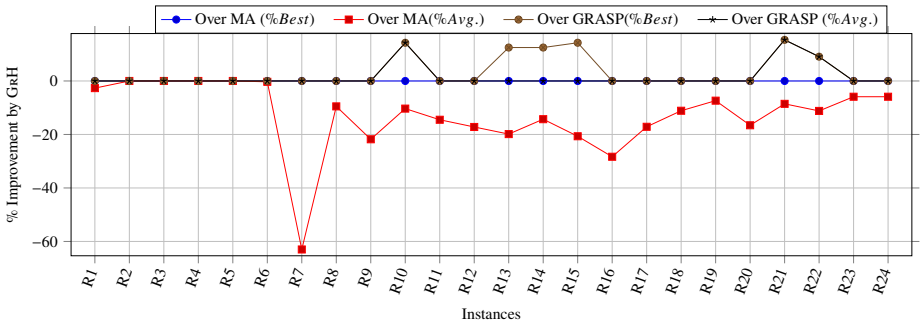


Fig. 3: Comparison of GRH, MA, and GRASP across instances in terms of $\%Best$ and $\%Avg.$ on Random UDG Graph.

4.2 Comparative results of GRH with SMWCDS, MA, and GRASP on LPNMR'09 Test instances

The comparative results for the LPNMR'09 test set are presented in Figure 4. Graph demonstrates the performance edge of proposed heuristic (GRH) over the SMWCDS against the best and average solutions across nearly all test instances.

Specifically, for the best-found solutions ($\%Best$), GRH achieves a performance improvement ranging from 11.11% to over 40% for all instances except L4. Regarding the average solution quality ($\%Avg.$), the heuristic shows substantial improvements, exceeding 38.05%, 37.11%, and 36.17% over SMWCDS for most instances.

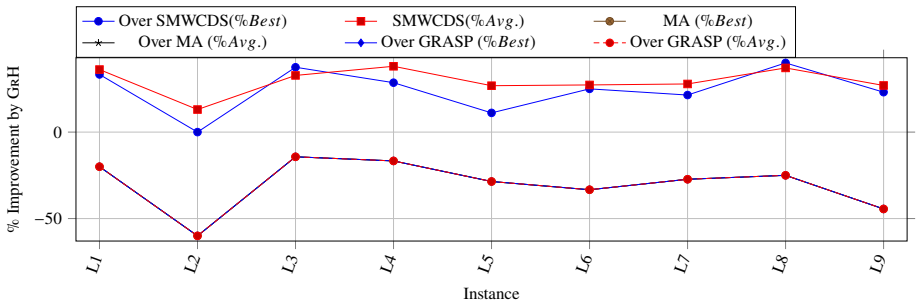


Fig. 4: Comparison of $\%Best$ and $\%Avg.$ results for GRH, MA and GRASP on LPNMR'09 test instances.

In contrast, a comparison with the MA and GRASP metaheuristics reveals a different outcome. As illustrated in Figure 4, GRH exhibits no discernible improvement over either MA or GRASP in terms of both $\%Best$ and $\%Avg.$ solution values across all instances.

4.3 Comparative results of GRH with SMWCDS, MA, and GRASP on Common UDG graph Test instances

Upon evaluating the performance of GRH against the SMWCDS algorithm on the Common UDG graph Test instances, the results demonstrate that GRH performs better in both the Best ($\%Best$) and Average ($\%Avg.$) solution quality in almost all test cases, with only a single exception. The improvement in $\%Best$ solutions exceeds 21.43% in one specific instance, while in other cases the enhancement falls within 18.89% to 5.56% improvement range. Similarly, for $\%Avg.$ solutions, the performance gains are particularly significant in some instances with improvements surpassing 27.93%, while most test cases show consistent enhancements of approximately 14% to 21%. These comparative results against SMWCDS are presented in Figure 5. However, when comparing GRH with both MA and GRASP algorithms on these same test instances, the proposed heuristic shows relatively weaker performance. This comparative analysis against MA and GRASP is illustrated in Figure 6.

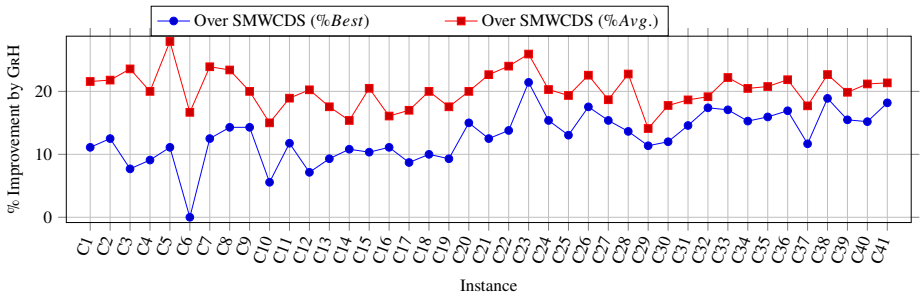


Fig. 5: Comparison of $\%Best$ and $\%Avg.$ results of GRH and SMWCDS across instances of Common UDG Graph.

Figure 4 to Figure 8 demonstrates that GRH struggles when compared to both MA and GRASP in terms of both $\%Best$ and $\%Avg.$ solution quality. This performance difference stems from fundamental algorithmic distinctions: MA and GRASP employ iterative solution generation through multiple execution runs, while GRH operates as a single-pass heuristic producing just one solution per instance. The repeated sampling and refinement capability of MA and GRASP enables them to explore the solution space more thoroughly, leading to better average-case and best-case outcomes at the cost of greater computational overhead.

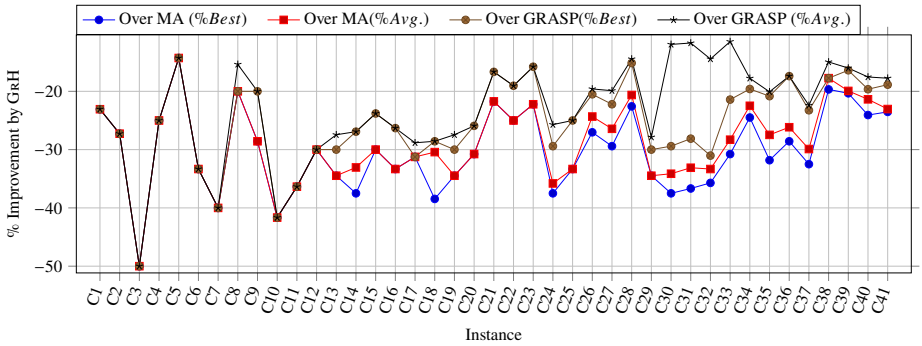


Fig. 6: Comparison of the $\%Best$ and $\%Avg.$ results of GRH, MA, and GRASP across instances of the Common UDG graph.

4.4 Comparative results of GRH with SMWCDS, MA, and GRASP on NDR graph Test instances

For NDR graph test instances the results are presented in Figures 7 and 8. In terms of percentage improvement, the proposed approach GRH achieves a 3.77% to 37.72% improvement across all instances compared to the $\%Best$ solutions of SMWCDS, and a 4.81%–41.56% improvement compared to the $\%Avg.$ solutions, as shown in Figure 7. In Figure 8, when compared with MA, our method demonstrates an improvement of 3.73%–18.47% on six instances for $\%Best$ solutions, and 4.26% to 18.51% on six instances for $\%Avg.$ solutions. Against GRASP, the proposed approach achieves a 6.77%–8.28% improvement on two instances for $\%Avg.$ solutions. However, the heuristic performance, while competitive, does not significantly exceed that of MA and GRASP for the remaining instances.

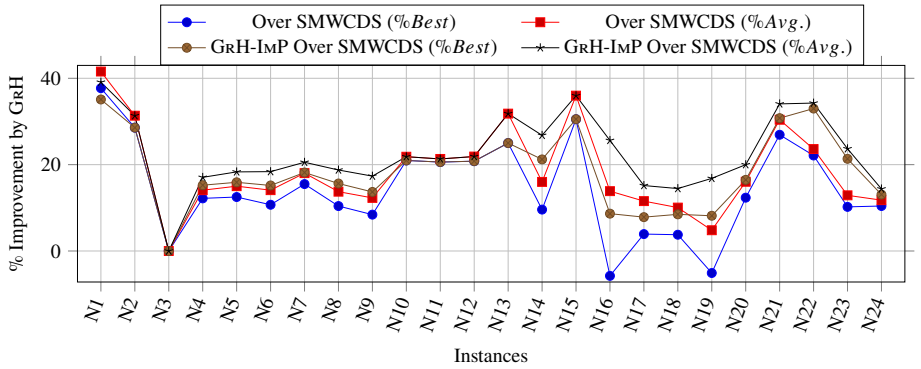


Fig. 7: Comparison of $\%Best$ and $\%Avg.$ results of GRH and GRH-IMP over SMWCDS and NDR Graph.

When the improvement operator is applied to the proposed heuristic GRH on the NDR graph instances, it exhibits only minimal improvement due to the greedy nature of the approach and its limited search space. Specifically, it achieves a 1.04% better $\%Best$ result on one instance and matches another instance compared to GRASP, while the $\%Avg.$ improvement ranges from 5.78% to 19.64% across two instances. In contrast, the NDR graph's larger instances and structural complexity provide a broader search space, enabling greater potential for improvement. Notably, while MA and GRASP are metaheuristics, the proposed method is a heuristic-based approach and the execution timeout per instance is 300 seconds.

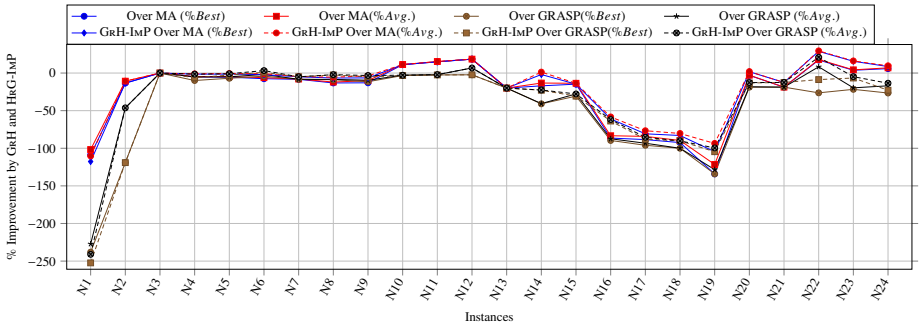


Fig. 8: Comparison of $\%Best$ and $\%Avg.$ results of GRH and GRH-IMP over MA and GRASP on NDR graph.

4.5 Some other observations from the computational results

GRASP performs better than most existing methods, mainly due to its thorough local search strategy. However, this carries the penalty of longer execution time. When comparing our proposed approach (GRH) with SMWCDS, GRH delivers better results in both the best and average cases (see Figure 5 and Figure 7). However, GRH does not outperform metaheuristic approaches such as MA and GRASP (Figure 6) on many instances, which is expected since GRH is a simpler heuristic. Still, in some cases, such as instances (2000-300-230) to (2000-300-260) GRH comes close to GRASP's performance. This suggests that combining GRH with a metaheuristic framework could further improve its searching capabilities.

On NDR graphs (Figure 8), GRH performs similarly to SMWCDS in most of the instances and even slightly better than MA and GRASP over four instances. However, there are still cases where its search efficiency could be improved.

Overall, GRH shows great potential, especially for large instances. Future improvements could include the strategies discussed in Section 5.

5 Conclusion and Future Direction

The experimental results demonstrate that a balanced integration of greedy and randomized strategies can effectively address the MWCDSP, consistently yielding high-quality solutions. The core principle underlying this approach lies in the deliberate combination of these two complementary strategies, greedy and random, which, when applied strategically, improve both exploitation and exploration within the search space. By randomly selecting nodes from the predefined candidate set $\mathcal{P}_{set}$, the algorithm introduces diversity, thereby escaping premature convergence, while the construction of $\mathcal{P}_{set}$ itself ensures that the search remains guided toward promising regions of the solution space. This careful tradeoff between randomness and greediness not only improves solution quality but also mitigates the risk of premature convergence, which is a common challenge in combinatorial optimization problems.

The future research could explore the incorporation of advanced metaheuristic techniques such as Evolutionary Algorithm with Guided Mutation, hyper-heuristics, and deep reinforcement learning (DRL) to further refine the algorithm's performance. This would help the algorithm handle larger problems and find near-optimal solutions across a wider range of test cases. Using probabilistic models, automatic heuristic selection, and neural network-based decisions, future studies may discover new ways to solve complex rule-based problems more efficiently.

References

1. Abu-Khzam, F.N.: An improved exact algorithm for minimum dominating set in chordal graphs. Information Processing Letters **174**, 106206 (2022)
2. Chaurasia, S.N., Kim, J.H.: An evolutionary algorithm based hyper-heuristic framework for the set packing problem. Information Sciences **505**, 1–31 (2019)
3. Chaurasia, S.N., Singh, A.: A hybrid evolutionary algorithm with guided mutation for minimum weight dominating set. Applied Intelligence **43**(3), 512–529 (2015)
4. Chaurasia, S.N., Singh, A.: A hybrid heuristic for dominating tree problem. Soft Computing **20**(1), 377–397 (2016)
5. Cravo, G.L., Amaral, A.R.S.: A grasp algorithm for solving large-scale single row facility layout problems. Computers & Operations Research **106**, 49–61 (2019)
6. Ding, Y., Wang, J.Z., Srimani, P.K.: A linear time self-stabilizing algorithm for minimal weakly connected dominating sets. International Journal of Parallel Programming **44**(1), 151–162 (2016)
7. Domke, G.S., Hattingh, J.H., Markus, L.R.: On weakly connected domination in graphs ii. Discrete Mathematics **305**(1-3), 112–122 (2005)
8. Du, H., Wu, W., Shan, S., Kim, D., Lee, W.: Constructing weakly connected dominating set for secure clustering in distributed sensor network. Journal of combinatorial optimization **23**, 301–307 (2012)
9. Dunbar, J.E., Grossman, J.W., Hattingh, J.H., Hedetniemi, S.T., McRae, A.A.: On weakly connected domination in graphs. Discrete Mathematics **167**, 261–269 (1997)
10. Ferdi, I., Layeb, A.: A grasp algorithm based new heuristic for the capacitated location routing problem. Journal of Experimental & Theoretical Artificial Intelligence **30**(3), 369–387 (2018)
11. Ghaffari, M., Nesterenko, M., Tixeuil, S., Tucci, S., Yamauchi, Y.: Stabilization, Safety, and Security of Distributed Systems. Springer (2019)

12. Han, B., Jia, W.: Clustering wireless ad hoc networks with weakly connected dominating set. Journal of Parallel and Distributed Computing **67**(6), 727–737 (2007)
13. Hu, S., Liu, H., Wang, Y., Li, R., Yin, M., Yang, N.: Towards efficient local search for the minimum total dominating set problem. Applied Intelligence pp. 1–15 (2021)
14. Jovanovic, R., Tuba, M.: Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem. Computer Science and Information Systems **10**(1), 133–149 (2013)
15. Li, R., Hu, S., Gao, J., Zhou, Y., Wang, Y., Yin, M.: Grasp for connected dominating set problems. Neural Computing and Applications **28**, 1059–1067 (2017)
16. Nakkala, M.R., Singh, A., Rossi, A.: Swarm intelligence, exact and matheuristic approaches for minimum weight directed dominating set problem. Engineering Applications of Artificial Intelligence **109**, 104647 (2022)
17. Niu, D., Nie, X., Zhang, L., Zhang, H., Yin, M.: A greedy randomized adaptive search procedure (grasp) for minimum weakly connected dominating set problem. Expert Systems with Applications **215**, 119338 (2023)
18. Niu, D., Yin, M.: A self-stabilizing memetic algorithm for minimum weakly connected dominating set problems. In: the 2nd international workshop on heuristic search in industry (HSI), In: conjunction with the 31st international joint conference on artificial intelligence and the 25th European conference on artificial intelligence (IJCAI-ECAI 2022) (2022)
19. Pan, S., Ma, Y., Wang, Y., Zhou, Z., Ji, J., Yin, M., Hu, S.: An improved master-apprentice evolutionary algorithm for minimum independent dominating set problem. Frontiers of Computer Science **17**(4), 174326 (2023)
20. Pathan, A.S.K., Hong, C.S.: A key-predistribution-based weakly connected dominating set for secure clustering in dsn. In: High Performance Computing and Communications: Second International Conference, HPCC 2006, Munich, Germany, September 13-15, 2006. Proceedings 2. pp. 270–279. Springer (2006)
21. Pitsoulis, L.S., Resende, M.G.: Greedy randomized adaptive search procedures. Handbook of applied optimization pp. 168–183 (2002)
22. Raczek, J., Cyman, J.: Weakly connected roman domination in graphs. Discrete Applied Mathematics **267**, 151–159 (2019)
23. Resende, M.G., Ribeiro, C.C.: Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. Handbook of metaheuristics pp. 283–319 (2010)
24. Sandueta, E.P.: Weakly connected total domination critical graphs. Advances and Applications in Discrete Mathematics **25** (2020)
25. Shin, I., Shen, Y., Thai, M.T.: On approximation of dominating tree in wireless sensor networks. Optimization Letters **4**(3), 393–403 (2010)
26. Wang, Y., Cai, S., Chen, J., Yin, M.: A fast local search algorithm for minimum weight dominating set problem on massive graphs. In: IJCAI. pp. 1514–1522 (2018)
27. Yu, J., Wang, N., Wang, G.: Constructing minimum extended weakly-connected dominating sets for clustering in ad hoc networks. Journal of Parallel and Distributed Computing **72**(1), 35–47 (2012)