A Fast Metaheuristic for Finding the Minimum Dominating Set in Graphs

Alejandra Casado[1], Sergio Bermudo[2], Ana Dolores López-Sánchez[2], and Jesús Sánchez-Oro[1](✉)

[1] Universidad Rey Juan Carlos, Mostoles, Spain
{alejandra.casado,jesus.sanchezoro}@urjc.es

[2] Universidad Pablo de Olavide, Seville, Spain
{sbernav,adlopsan}@upo.es

Abstract. Finding minimum dominating sets in graphs is a problem that has been widely studied in the literature. However, due to the increase in the size and complexity of networks, new algorithms with the ability to provide high quality solutions in short computing times are desirable. This work presents a Greedy Randomized Adaptive Search Procedure for dealing with the Minimum Dominating Set Problem in large networks. The algorithm is conformed by an efficient constructive procedure to generate promising initial solutions and a local search designed to find a local optimum with respect to those initial solutions. The experimental results show the competitiveness of the proposed algorithm when comparing it with the state-of-the-art methods.

Keywords: Minimum dominating set · Grasp · Metaheuristics

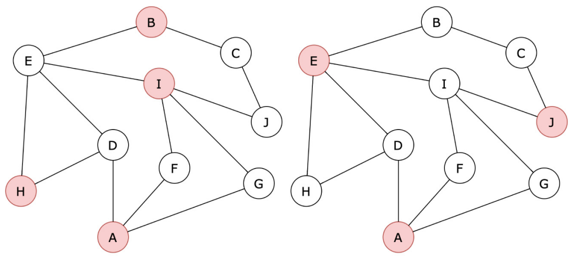
1 Introduction

Due to the development of large networks in several context such as road networks, social networks, electrical networks, communication networks, computer networks or security networks, among others, graph theory has regained the interest of researchers and practitioners. The study of problems related to graph domination is becoming more and more relevant. Since its original proposal [2] and the main definition of domination number [10], more than two thousand research papers have been published on this topic (we refer the reader to [6–8].

Given a graph $G = (V, E)$, a dominating set of vertices D is a subset of V in which every vertex $u \in V \setminus D$ is adjacent to, at least, one vertex in D, i.e., $\exists v \in D : (u, v) \in E$. The objective function of the minimum dominating set problem (MDSP) is evaluated as the number of vertices belonging to the dominating set, i.e., $MDSP(G, D) = |D|$. Then, the MDSP consists of finding a minimum dominating set $D^\star$ among all possible dominating sets of the graph under evaluation G. More formally,

$$D^\star \leftarrow \arg\min_{D \in \mathcal{D}} MDSP(G, D)$$

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
L. Di Gaspero et al. (Eds.): MIC 2022, LNCS 13838, pp. 554–559, 2023.
https://doi.org/10.1007/978-3-031-26504-4_47



(a) $D_1 = \{A, B, H, I\}$ (b) $D_2 = \{A, E, J\}$

Fig. 1. Two feasible solutions D_1 and D_2 for a graph with 10 nodes and 12 edges.

where $\mathcal{D}$ represents the set of all possible dominating sets over the graph G.

Let us illustrate this problem with a graphical example, depicted in Fig. 1. In both examples, the vertices included in the dominating set are colored. Solution $D_1 = \{A, B, H, I\}$, depicted in Fig. 1(a), represents a dominating set for the example graph with an associated objective value of $|D_1| = 4$. If we now analyze solution $D_2 = \{A, E, J\}$, the objective function value is $|D_2| = 3$, being better than D_1. In fact, D_2 is the optimal solution for the considered graph (i.e., it is not possible to find a dominating set with less than three vertices).

This problem has several applications in different fields: social networks [9], radio stations [3], network surveillance [5], etc. The MDSP has been proven to be $\mathcal{NP}$-hard for arbitrary graphs [8], exact algorithms are not practical for large-sized graphs. Although there are only a few works focused on finding an algorithm to approximate the MDSP for general graphs [1,11], it has been extensively studied for particular graphs, proposing several bounds for the minimum size of the dominating set in general graphs also [6,8].

This work deals with the MDSP from a metaheuristic point of view for providing high-quality solutions in small computing times. As far as we know, the best approach for the MDSP is an order-based randomized local search (RLS) [1], which is shown to be better than previous approaches such as a greedy heuristic or an Ant Colony Optimization algorithm [11]. This algorithm use a permutation-based representation of the solution for MDSP which is transformed into dominating sets using a greedy approach. Then, a local search method based on randomized jump moves is applied.

The remaining of the paper is organized as follows: Sect. 2 presents the algorithmic approach, Sect. 3 details the computational experiments performed to test the quality of the proposal, and Sect. 4 draws some conclusions derived from the research.

2 Greedy Randomized Adaptive Search Procedure

The Greedy Randomized Adaptive Search Procedure (GRASP) is a multi-start metaheuristic originally introduced for solving the set covering problem [4]. It consists of two well-differenced phases. First, the construction phase is responsible for generating an initial feasible solution and, then, the local improvement stage finds a local optimum with respect to some predefined neighborhood. The distinguishing feature of GRASP is the inclusion of randomization in the construction phase, with the aim of increasing the diversity of the search.

2.1 Construction

The construction phase is one of the key parts of GRASP metaheuristic. It is responsible for including randomization in the search, which prevents the algorithm from stagnating in local optima. Algorithm 1 shows the pseudocode of the proposed constructive procedure, named SNF (Select Non-monitored First).

Algorithm 1. $SNF(G = (V, E), \alpha)$

```
1:  $v \leftarrow Random(V)$
2:  $D \leftarrow \{v\}$
3:  $CL \leftarrow V \setminus \{v\}$
4:  $M \leftarrow \{u \in V : (u, v) \in E\}$
5:  while $|M| \leq |V \setminus D|$ do
6:      $g_{\min} \leftarrow \min_{c \in CL} g(c)$
7:      $g_{\max} \leftarrow \max_{c \in CL} g(c)$
8:      $\mu \leftarrow g_{\max} - \alpha \cdot (g_{\max} - g_{\min})$
9:      $RCL \leftarrow \{c \in CL : g(c) \geq \mu\}$
10:     $d \leftarrow Random(RCL)$
11:     $D \leftarrow D \cup \{d\}$
12:     $CL \leftarrow CL \setminus \{d\}$
13:     $M \leftarrow M \cup \{u \in V : (u, d) \in E\}$
14: end while
15: return $D$
```

The method starts by randomly selecting the first vertex to be included in the solutions, as it is customary in GRASP to increase diversity (step 1). Then, the solution D is initialized with the selected vertex (step 2) and the candidate list CL is constructed with all the vertices except the selected one (step 3). Additionally, the set of monitored vertices is created with those vertices which are adjacent to the selected one, since those are now monitored (step 4). Then, the method iteratively adds a new vertex to the solution under construction until all the vertices are monitored (steps 5–14). In each iteration, a greedy function value g is considered to evaluate how promising a node is. In the context of MDSP, a vertex is promising if it is able to monitor a large number of non-previously monitored vertices. Therefore, the greedy function is evaluated as the number of adjacent vertices to the node under evaluation which are not monitored by any other node, i.e., $g(v) = |\{u \in V \setminus M : (u, v) \in E\}|$. The procedure then evaluates the minimum and maximum greedy function value (steps 6–7) to calculate a threshold μ (step 8). The threshold directly depends on the value of an input parameter named $\alpha \in [0, 1]$, which controls the randomness/greediness of the method, i.e., if $\alpha = 0$, then $\mu = g_{\max}$ and the method is completely greedy, while $\alpha = 1$ results in $\mu = g_{\min}$, providing a completely random method. This threshold is used to construct a restricted candidate list *RCL* which contains all the candidate vertices whose greedy function value is better (larger) than μ (step 9). Then, the next vertex is selected at random from the *RCL* (step 10), including it in the solution under construction (step 11) and removing it from the *CL* (step 12). Additionally, the set of monitored nodes M is updated with all the adjacent vertices to the selected one (step 13). Finally, the method returns the constructed solution (step 15).

2.2 Local Improvement

The first element to be defined in order to propose a local search method is the move operator. In this work, we propose the use of swap moves, which removes a vertex from the solution and replaces it with a new one. In mathematical terms,

$$\mathit{Swap}(D, i, j) \leftarrow D \backslash \{i\} \cup \{j\}$$

Notice that this movement is not able to produce any improvement by itself, since the number of vertices in the dominating set will remain the same. However, the movement may result in a solution in which some vertices are redundant, i.e., they are covering vertices which are also covered by another vertex. All those redundant vertices are removed after each swap move, eventually leading to an improvement. Given this move operator, the neighborhood to be explored is defined as all the solutions that can be reached with a single swap move. More formally,

$$N(D) \leftarrow \{D' \leftarrow \mathit{Swap}(D, i, j)\ \forall i \in D \wedge\ \forall j \in V \backslash D\}$$

Having defined the move operator and the neighborhood explored, it is necessary to establish the strategy followed by the local search to explore the neighborhood. There are two main strategies to traverse the neighborhood: first improvement and best improvement. Since the problem has constraints related to computing time, we select first improvement as exploration strategy. Then, the first swap move that leads to an improvement is performed in each iteration of the search, avoiding exploring the complete neighborhood in each iteration, thus leading to reduce the computational effort without deteriorating the quality of the obtained solutions.

It is worth mentioning that when following a first improvement approach, the order in which the neighbors is explored is relevant. Although the neighborhood can be traversed at random, exploring first the most promising neighbors usually result in better solutions. In the context of MDSP, the nodes to be removed are selected in ascending order with respect to the number of nodes that remains non-dominated after its removal, with the aim of increasing the number of redundant nodes after the swap move. Additionally, the vertex which replaces it will be the one which dominates the maximum number of non-dominated nodes after the removal. This heuristic selection of the nodes to be removed and the ones to be added allows the local search procedure to find improvements in a small number of iterations, resulting in an efficient and effective local search method, as it can be seen in Sect. 3.

3 Computational Experiments

This section has two main objectives: 1) configure the parameters of the proposed algorithm and 2) perform a competitive testing against the state-of-the-art method, named RLS [1], previously described in Sect. 1.

The proposed algorithm have been implemented in Java 17 and all the experiments has been carried out in an Intel Core i7 2.7 GHz and 16 GB RAM, while the state of the art is executed in a similar computer but with 64 GB of RAM. In order to have a fair comparison, we have used the same testbed of instances as in the best previous work, conformed by 21 instances with nodes ranging from 34 to 16726, extracted from different applications of the problem. The results reports the following metrics: Time (s), average computing time required by the algorithm in seconds; Avg., average objective function value; Dev (%), average percentage deviation with respect to the best value found in the experiment; and #Best, number of times in which the best solution of the experiment is reached by the algorithm.

The preliminary experimentation is designed to select the best configuration of the proposed algorithm and, in order to avoid overfitting, a subset of 9 out of 21 representative instances are selected for this stage. The GRASP algorithm proposed has a single input parameter, which is the α parameter of the constructive procedure, which has to be tuned. In particular, the values tested are $\alpha = \{0.25, 0.50, 0.75, RND\}$, where *RND* indicates that it is selected at random in the range [0, 1] in each construction. In all the cases, 100 constructions followed by its corresponding local improvement are performed. The best results are achieved when considering $\alpha = 0.25$, reaching 8 out of 9 best solutions with a deviation of 0.01%.

The competitive testing is performed against RLS to evaluate the performance of GRASP. In this experiment, the complete set of instances is considered. Table 1 shows the results obtained by both algorithms.

As it can be seen from the results, both algorithms provide similar performance in terms of quality. Although RLS is able to reach an additional best solution, the average deviation of GRASP is smaller than the one of RLS. Finally, the computing time required by GRASP is considerably smaller than the 10 min required by RLS. These results show that GRASP emerges as a competitive algorithm for finding a minimum dominating set in large graphs.

Table 1. Competitive testing between GRASP and RLS.

Algorithm	Time (s)	Avg.	Dev(%)	#Best
GRASP	33.66	603.19	0.40	19
RLS	600.00	603.05	0.68	20

4 Conclusion

The proposed GRASP algorithm is conformed by a greedy randomized adaptive constructive procedure and a local search method. The constructive procedure, named SNF, is able to generate high quality and diverse solutions in small computing times, providing promising starting points for the local search. The proposed local search is able to find a local optimum with respect to the initial point without requiring high computational efforts, resulting in an efficient and effective algorithm for tackling the MDSP. The results obtained show the competitiveness of GRASP when comparing it with the state of the art, providing similar results, being almost twenty times faster than RLS. As a future work, the testbed will be enlarged with more challenging instances derived from real-life networks to show the potential of GRASP.

References

1. Chalupa, D.: An order-based algorithm for minimum dominating set with application in graph mining. Inf. Sci. **426**, 101–116 (2018)
2. De Jaenisch, C.F.: Applications de l'Analuse mathematique an Jen des Echecs. Petrograd (1862)
3. Erwin, D.: Dominating broadcasts in graphs. Int. J. Comput. Eng. Technol. **42**, 89–105 (2004)
4. Feo, T.A., Resende, M.G.: A probabilistic heuristic for a computationally difficult set covering problem. Oper. Res. Lett. **8**(2), 67–71 (1989)
5. Haynes, T.W., Hedetniemi, S.M., Hedetniemi, S.T., Henning, M.A.: Domination in graphs applied to electric power networks. SIAM J. Discret. Math. **15**(4), 519–529 (2002)
6. Haynes, T.W., Hedetniemi, S., Slater, P.: Fundamentals of Domination in Graphs. CRC Press, Boca Raton (2013)
7. Haynes, T.W., Hedetniemi, S.T., Henning, M.A.: Structures of Domination in Graphs. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-58892-2
8. Haynes, T.: Domination in Graphs: Volume 2: Advanced Topics. Routledge (2017)
9. Lozano-Osorio, I., Sánchez-Oro, J., Duarte, A., Cordón, Ó.: A quick grasp-based method for influence maximization in social networks. J. Ambient Intell. Human. Comput. 1–13 (2021)
10. Ore, O.: Theory of Graphs. AMS, Providence (1962)
11. Potluri, A., Singh, A.: Hybrid metaheuristic algorithms for minimum weight dominating set. Appl. Soft Comput. **13**, 76–88 (2013)