Proceedings of the Conference on March 14 - International Day of Mathematics

Vuković, Mirjana, urednik; Nurkanović, Mehmed, urednik

2024-12-26

Academy of Sciences and Arts of Bosnia and Herzegovina

https://bastina.anubih.ba/handle/123456789/798

PROCEEDINGS OF THE CONFERENCE ON MARCH 14 – INTERNATIONAL DAY OF MATHEMATICS
Special Editions ANUBiH, Book CCXVI, OPMN Book 30
Sarajevo 2024, **p. 81–89** DOI: 10.5644/PI2024.216.10

FINDING A MINIMAL DOMINATING SET OF A GRAPH COMBINING VARIOUS HEURISTIC APPROACHES BASED ON VARIABLE NEIGHBORHOOD SEARCH

ANTON VRDOLJAK

Dedicated to the 75th birthday of our dear Professor Mirjana Vuković

ABSTRACT. Many complex combinatorial optimization problems cannot be solved by traditional optimization techniques. Therefore, these types of problems often require the application and sometimes a combination of other scientific approaches to obtain, at the very least, adequate solutions. As demonstrated in numerous papers, heuristic approaches are usually more efficient and effective compared to classical methods, especially in managing neighboring uncertainty. Therefore, this paper focuses on combining various heuristic approaches based on the basic variable neighborhood search (BVNS) metaheuristic. This paper aims to find a minimal dominating set of a graph, which is a well-known NP-complete problem.

1. INTRODUCTION

Combinatorial optimization problems are often modeled using graphs because many such problems can be represented as searching for an optimal structure within a graph. Next, their invariance to permutation is crucial for such quests.

Definition 1.1. *A graph is an ordered triple $G = (V, E, \varphi)$, where V is a non-empty set of vertices, E is a set of edges that is disjoint from V, and φ is a function that assigns to each edge from E two, not necessarily different, vertices from V. A graph is usually represented by the ordered pair $G = (V, E)$ or simply G.*

Definition 1.2. *Let $G = (V, E)$ be any graph. A set $D \subseteq V$ is a dominating set of a graph G if for every vertex $v \in V \setminus D$ there is at least one $u \in D$ that is adjacent to v, i.e. for which $uv \in E$ holds.*

Definition 1.3. *A dominating set D of a graph G is called a minimal dominating set, if there is no proper subset $D' \subset D$ that is a dominating set of the graph G. A dominating set of the smallest cardinality (size) is called a minimum dominating set, and its cardinal number determines the (lower) domination number $\gamma(G)$ for the given graph G.*

Hence, a dominating set of a graph $G = (V, E)$ is a subset D of V, such that every vertex from the graph G is either in a dominating set or its neighbor is in a dominating

2020 *Mathematics Subject Classification.* 05C69.
Key words and phrases. Dominating set, greedy algorithm, variable neighborhood search.

set [7]. A minimum dominating set is always a minimal dominating set, but the converse does not necessarily hold [15] (see Figure 1 for a counterexample). Furthermore, every graph has at least one dominating set: if $D = V$ = the set of all vertices of the graph G then the set D is by definition a dominating set of a graph G, since $V \setminus D$ is necessarily the empty set. Clearly, a graph without edges has the whole vertex set V as its unique dominating set. For graphs without isolated vertices (an isolated vertex is a vertex with degree zero), there are always two disjoint dominating sets: if D_m = a minimal dominating set of a graph G, then $V \setminus D_m$ is a dominating set.

Unfortunately, finding a dominating set of size k represents an NP-complete decision problem in computational complexity theory, and for now, there is no efficient algorithm that finds a minimal dominating set for a selected graph [8]. Accordingly, checking whether the domination number

$$\gamma(G) = \min\{|D| : D \subseteq V \text{ and } D \text{ is a dominating set of a graph } G\}$$

of an arbitrary graph G is less than a given integer is also an NP-complete problem. If the graph G does not contain isolated vertices, then surely $\gamma(G) \leq \frac{n}{2}$ holds, where $n = |V|$. The domination number $\gamma(G)$ for the graph G from Figure 1, i.e. for the Petersen graph, is 3.

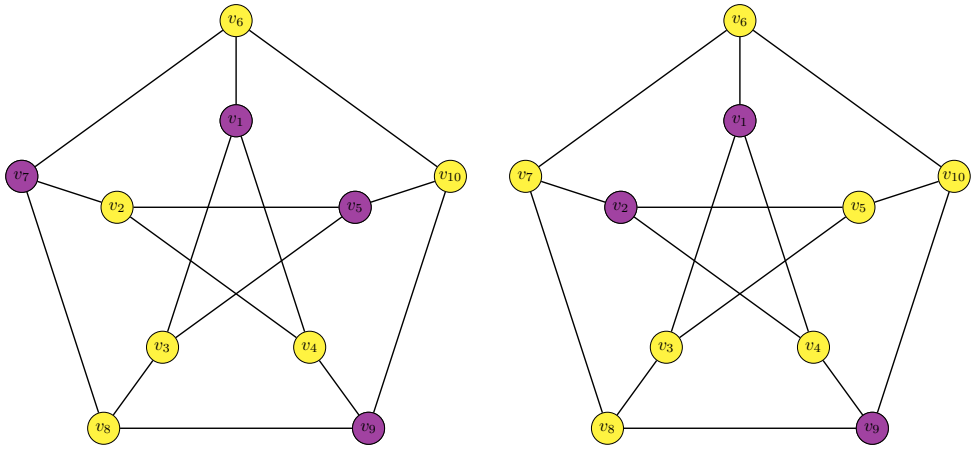

FIGURE 1. A minimal (on the left) and a minimum dominating sets of the Petersen graph.

The primary research problem of this paper is finding a minimal dominating set of a simple undirected graph, i.e. our primary goal is to devise an efficient algorithm for solving such an NP-complete problem. We propose a method that links several heuristic approaches based on variable neighborhood search metaheuristic.

2. ALGORITHM

Graphs are an unparalleled mathematical way to represent a network of interconnected objects that model real-life problems. One such problem was addressed as the

second research problem in the dissertation submitted to the Faculty of Science (Department of Mathematics and Computer Sciences) at the University of Sarajevo [14]. The main results, as well as the algorithm, regarding this problem, i.e. the construction (finding) of the minimal dominating set of amino-acid scales, are submitted to Mathematical Medicine and Biology: A Journal of the Institute of Mathematics & its Applications [13]. To summarize the proposed approaches, we will provide a brief review.

Clearly, to achieve the primary goal, it is necessary to create some initial dominating set. This task (optimization problem) is correctly solved using the greedy algorithm (first approach) and cluster analysis (second approach).

A greedy algorithm is any algorithm that uses a metaheuristics to solve a problem, in a way that it solves the problem by choosing a locally optimal solution at each step (hoping to reach the global optimum that way) [3]. In most combinatorial optimization problems it proceeds, starting from the partial solution $X = \emptyset$, by repeatedly adding to X an element x from the ground set E [1]. Although such an approach can be disastrous for some computational tasks, it is still optimal for many others. Several advantages of greedy algorithms are of particular importance, for example, their design and implementation are usually simple, execution is fast, and we practically never revisit previous choices.

Cluster analysis is a set of methods (algorithms) that allows us to classify (divide) observed data (objects) into groups (classes or clusters) that are meaningful, useful or both. In other words, it allows us to conceptualize similarities and differences between observed data. Cluster analysis is closely related to different fields of research and has been present in the literature for several decades, so it undoubtedly possesses one of the desirable features that any clustering algorithm should have: the ability to work with different types of data. It is also insensitive to the order of steps in the algorithm, it enables the detection of clusters of arbitrary sizes and shapes, as well as the identification of high-quality clusters in the presence of noise.

Greedy approach. In the first approach, we begin with an initial graph $G_0 = G$ of size n and a partial solution $D = \emptyset$. In each step i, we select a vertex v_i with the maximal degree in a graph G_{i-1} and then construct the graph

$$G_i = G_{i-1} - v_i - N_{G_{i-1}}(v_i),$$

where $N_{G_{i-1}}(v_i)$ denotes the set of neighbors of v_i in a graph G_{i-1}. If there are multiple vertices with an equal largest degree, we randomly select one of them. Hence, in each step, we need to determine a maximal degree which takes $o(v(G_i)) \leq o(n)$ operations and eliminate all its neighbors which takes $o(d_{G_i}(v_i)) \leq o(n)$ operations. Hence, this algorithm can be executed in less than or equal to $o(n^2)$ operations. A briefly introduced custom-tailored greedy algorithm (greedy approach) is formally described by Algorithm 1.

The algorithm's correct operation is guaranteed by Proposition 2.1 and its execution in polynomial time is guaranteed by Proposition 2.2, which is stated without proof (see the notes in the paragraph above).

Algorithm 1 Pseudocode for a custom-tailored greedy algorithm [13]

Require: Graph $G = (V(G), E(G))$
Ensure: D = Graph-Dominating Set
1: $G_0 \leftarrow G$
2: $i \leftarrow 0$
3: $D \leftarrow \emptyset$
4: **repeat**
5: $\quad i \leftarrow i + 1$
6: $\quad$ Find a vertex v_i with the maximal degree in a graph G_{i-1}
7: $\quad D \leftarrow v_i$
8: $\quad G_i \leftarrow G_{i-1} - v_i -$ **AllNeighbors**(v_i)
9: **until** $(G_i = \emptyset)$
10: **return** (D)

Proposition 2.1. The set D is a subset of $V(G)$ and a dominating set of a graph $G = (V(G), E(G))$ at the end of the Algorithm 1 (custom-tailored greedy algorithm).

Proof. According to the initial declarations and repeat loop of Algorithm 1 at Step 7, the only elements of the set D are vertices v_i taken from $V(G)$, hence D is necessarily a subset of $V(G)$. Next, assume that D is not a dominating set in G at the end of Algorithm 1. Let $H = V \setminus D$, then there is at least one vertex $u \in H \subseteq V$ whose neighbors are not in the D. According to the stop condition for repeat loop, H is an empty set, but H is not. Therefore, according to the performance of Algorithm 1, the algorithm should not be ended, which is a contradiction. □

Proposition 2.2. The time complexity of Algorithm 1 is at most $o(n^2)$.

To illustrate the functioning of Algorithm 1, we will refer to the graph presented in Figure 1. In a given example with the Petersen graph, all vertices have equal vertex degrees, i.e. $d_G(v_i) = 3$, $\forall i \in \{1, 2, \ldots, 10\}$. Hence, the algorithm starts with an initial graph $G_0 = G$, a dominating set $D = \emptyset$ and in step 1 it will choose one vertex randomly among all vertices. For instance, let vertex v_2 be the chosen one. Therefore, at the end of step 1, we have $G_1 = G_0 \setminus \{v_2, v_4, v_5, v_7\} \neq \emptyset$ and $D = \{v_2\}$. Since G_1 is not an empty set, the algorithm will continue and in step 2 it will search for a vertex with maximal degree in a "surviving" graph G_1. In this particular graph, all vertices also have equal vertex degree, but this time $d_{G_1}(v_j) = 2$, $j \in \{1, 3, 6, 8, 9, 10\}$. So, the algorithm has to choose again randomly one of them. For instance, let vertex v_1 be chosen in step 2. Consequently, at the end of step 2, we will have $G_2 = G_1 \setminus \{v_1, v_3, v_6\} \neq \emptyset$ and $D = \{v_1, v_2\}$. The algorithm will still operate and in step 3 search for a vertex with a maximal degree in the new surviving graph G_2. In this particular case, only one vertex has a maximal degree. It is vertex v_9, so by adding it to the set D, and then eliminating it and all its neighbours from graph G_2, a new surviving graph G_3 will necessarily be an empty set. Therefore, according to the performance of Algorithm 1, the procedure should be finished, and as an output we will have $D = \{v_1, v_2, v_9\}$.

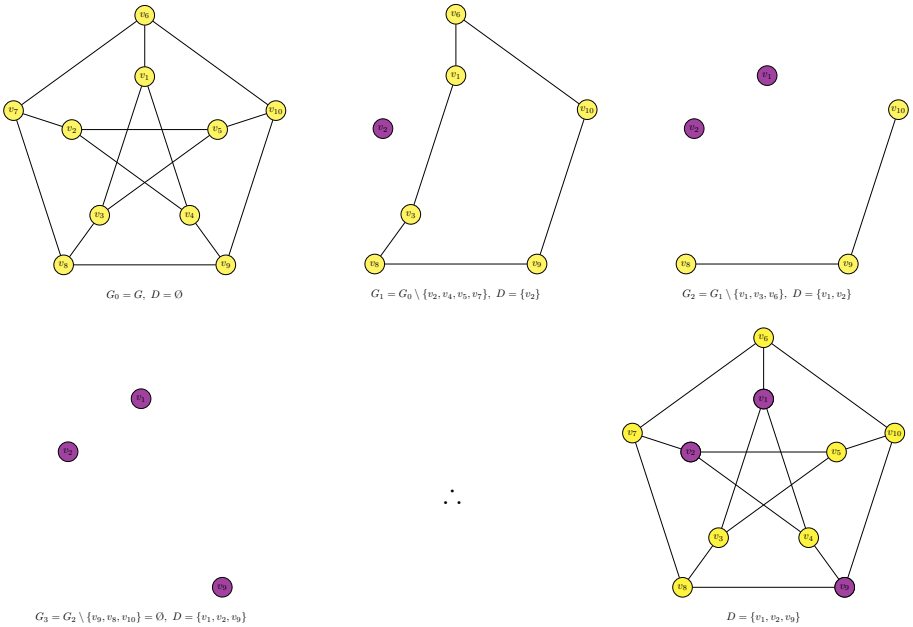

FIGURE 2. A demonstration of Algorithm 1 in action.

The Figure 2 showcases a demonstration of Algorithm 1 in action. We shall notice that in this simple example the set D is the most optimal solution candidate, i.e. it is the minimum dominating set for a given graph. Otherwise, the greedy algorithm does not have to find a solution to the problem. Even if it finds one, that solution may not be optimal, because the greedy algorithm does not use the objective function anywhere, but only the choice function.

Upon creating the initial dominating set, we apply another heuristic method – VNS or variable neighborhood search [11], [6] in order to reduce the size of that primal solution found by a greedy algorithm. Variable neighborhood search is a relatively recently developed metaheuristic. It was introduced in the nineties of the twentieth century, after which it underwent many applications and thus expansion. In this paper, only the basic variable neighborhood search (BVNS) was used, and in the following text, it will mainly be called variable neighborhood search, or VNS metaheuristic for the sake of simplicity. VNS is formally described by Algorithm 2.

The basic idea of the method is very simple (see Figure 3): a systematic change of neighborhoods within a local search (LS). The VNS metaheuristic is based on the following three key principles [5]:

[1]: A local minimum with respect to one neighborhood structure is not necessary so with another;
[2]: A global minimum is a local minimum with respect to all possible neighborhood structures;
[3]: For many problems, local minima with respect to one or several neighborhoods are relatively close to each other.

Algorithm 2 Pseudocode for VNS [2], [13]

Require: Neighborhoods
Ensure: D_{best}
$D_{\text{best}} \leftarrow$ RandomInitialSolution()
while (¬StopCondition()) **do**
  foreach (Neighborhood$_i \in$ Neighborhoods) **do**
    Neighborhood$_{\text{current}} \leftarrow$ CalculateNeighborhood(D_{best}, Neighborhood$_i$)
    $D_{\text{perturbare}} \leftarrow$ RandomPerturbareInNeighborhood(Neighborhood$_{\text{current}}$)
    $D_{\text{perturbare}} \leftarrow$ LocalSearch($D_{\text{perturbare}}$)
    if ($D_{\text{perturbare}}$ **is better than** D_{best}) **then**
      $D_{\text{best}} \leftarrow D_{\text{perturbare}}$
      break
    end if
  end foreach
end while
return (D_{best})

A part of the algorithm used to deal with the determination of solution quality tightly relies on the VNS metaheuristic. We propose two criteria to determine the quality of the solution obtained by a greedy algorithm:

[1]: One of the two solutions (dominating sets) is considered better than the other if it has a smaller number of vertices;
[2]: If two solutions have the same number of vertices, the one whose vertices have a larger sum of squared degrees in the original set is better.

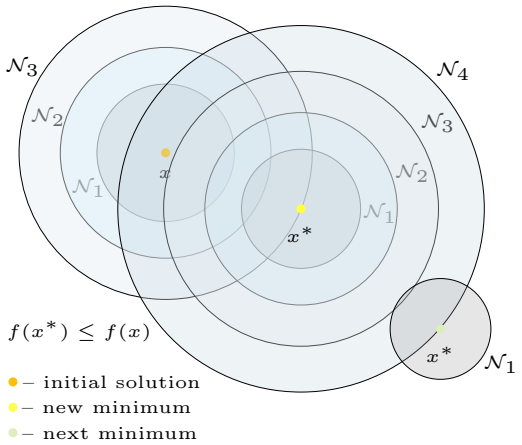

FIGURE 3. VNS introduces multiple neighborhoods for local search.

The argument for both conditions (criteria) is apparent since we are searching for the smallest dominating set. Next, considering that VNS more exhaustively analyzes

a set of greater similarities, this pushes searching for the optimal solution in the right direction.

The distance between sets of vertices is the cardinality of their symmetric difference. Since we can binary code sets of vertices (with 0-1 characters), Hamming distance is chosen as the metric. If some vertex set of size n has x ones and $n-x$ zeros, then there are $\binom{n}{d}$ vertex sets with distance d from that set. Clearly, in the case that we are dealing with large sets of vertices, the number $\binom{n}{d}$ will grow rapidly with increasing the distance d, i.e. it tends to grow in the order of n^d. Hence, this rationale leads us to exhaustively search neighborhoods at distances up to d_{min} and probabilistically neighborhoods at distances up to d_{max}. Determining if a selected set is a dominating set of vertices, is very often an action within the VNS improvement algorithm. The standard algorithm used here has a complexity of $o\left(n+\sum_{v\in D} d_G(v)\right)$ which is less than or equal to $o(n+m)$ where m is the number of edges in the observed graph. Hence, the VNS improvement algorithm has complexity $o(n^d \cdot (n+m)) = o(n^{d+1}+n^d \cdot m)$.

Proposition 2.3. The time complexity of Algorithm 2 is at most $o(n^d \cdot (n+m))$.

Clustering approach. In the second approach, we created the initial dominating set using a $k-$means iterative clustering algorithm with the Euclidean metric. Usually, we can fix the number of iterations required for convergence (to get stable centroid values). In a data-set that does have a clustering structure it is often small, as most changes typically occur in the first few iterations. For a fixed number of iterations i, the overall complexity of this algorithm is $o(k \cdot n \cdot m \cdot i)$ for a data-set with n objects ($m-$dimensional vectors), each with m attributes [10]. Thus, in practice, $k-$means is linear in all relevant factors, although it is in the worst case superpolynomial when performed until convergence. The usage of $k-$means is formally described by Algorithm 3.

Algorithm 3 Pseudocode for basic $k-$means algorithm [12]

Require: Data-set with n objects (each with m features), fixed number k
Ensure: k distinct non-overlapping clusters
1: Select k objects as initial centroids
2: **repeat**
3: Form k clusters by assigning each object to its closest centroid
4: Re-compute the centroid of each cluster
5: **until** Centroids do not change

However, this algorithm has several significant drawbacks [4]. One is manifested in the fact that the number of clusters k (in the vast majority of problems) is fixed and must be chosen by the user (inappropriate k may yield misleading results). Furthermore, this algorithm often converges to a local optimum. Also, the running time of the algorithm is unbounded (unlimited), although it usually works well on real-life problems (real

networks) [12]. Sometimes we only have a set of data, but we don't know how many different groups to expect in that data. In general, there is no method for determining the exact value of k, and it is often an *ad hoc* decision based on prior knowledge, assumptions, and practical experience. Nevertheless, if we want to determine the number of clusters for some data-set, that is, if we want to achieve a balance between the accuracy of joining objects to a cluster and the minimization of the objective function as a function of k, an adequate estimate can be obtained using the *average silhouette* method [9], *gap statistic* method, as well as the so-called *elbow* method [4].

After determining the desired number of clusters, we apply again a greedy algorithm from the first approach, but this time to each cluster. If there were singleton clusters (clusters with only one object), the method was slightly modified. The algorithm will merge such clusters with another cluster to obtain more meaningful initial sets. This is achieved through a single function that can be executed in $o(k \cdot n)$ operations. Ending up with a singleton cluster is frequent if a data-set contains many *outliers*, objects that are far from any other objects and therefore do not fit well into any cluster. Often, if an outlier is chosen as an initial seed, then no other vector is assigned to it during subsequent iterations [10]. For our second approach, we combine all initial sets of vertices (a union) obtained from each cluster by a greedy algorithm. The initial set, i.e. dominating set, found by a clustering algorithm is then fixed using the VNS strategy too.

3. Conclusions

In this paper, we have concisely reviewed the summary of the findings of a minimal graph-dominating set combining various heuristic approaches based on variable neighborhood search. With the paper's algorithms, a minimal dominating set of a simple undirected graph can be efficiently constructed by applying the VNS strategy on an initial dominating set obtained either by a greedy approach or by combining clustering and greedy approaches. Both implemented algorithms for constructing the initial set of vertices are relatively fast, i.e. require a very small amount of time, while the algorithm with regard to the VNS strategy operates at a quite leisurely pace. This is reasonable since NP-complete does not necessarily imply unsolvable. It just means any proposed solution will be slow.

Moreover, some results and the efficiency of the given approaches when applied on the particular real-life problem have been reported in an article [13] and a dissertation [14]. There a simple mathematical framework regarding modeling of a reduction to a dominating set of a graph with the set of 507 amino-acid scales (indices), constructed by Kawashima[1] and collaborators, is presented . Concerning guidelines for future research, given in the dissertation [14], it might be interesting to see whether similar generalized reduction in other contexts, especially in those major real-world instances (networks) where computationally intensive modeling is a dominant issue, will benefit from techniques used in these papers. Furthermore, it is also interesting to study is it possible to achieve even more significant improvements through the use of different settings, mainly regarding the desired number of clusters k and wanted numbers for dis-

[1] Amino acid index database: `https://www.genome.jp/aaindex/`

tances $d_{\min}$, $d_{\max}$. Finally, one can consider changing a default metric within the VNS improvement algorithm.

ACKNOWLEDGMENTS

The author is thankful to academician Mirjana Vuković and professor Amela Muratović-Ribić, supervisor for the 3rd cycle of study "Mathematical Sciences in Southeast Europe" at the Faculty of Science University of Sarajevo, for the invitation to participate in this special Conference on the occasion of March 14 – International Day of Mathematics.

REFERENCES

[1] G. Bendall and F. Margot, *Greedy-type resistance of combinatorial problems*, Discrete Optim., **3** (2006), 288–298, https://doi.org/10.1016/j.disopt.2006.03.001.

[2] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*, Open source book, 2012., http://www.cleveralgorithms.com/.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press, Massachusetts, 2001.

[4] B. S. Everitt, S. Landau, M. Leese and D. Stahl, *Cluster Analysis*, 5th edn. Wiley, New York, 2011, https://doi.org/10.1002/9780470977811.

[5] P. Hansen and N. Mladenović, *Variable neighborhood search*. In: F. Glover and G. A. Kochenberger (eds) Handbook of Metaheuristics. Springer, New York, 2003, pp. 145–184, https://doi.org/10.1007/b101874.

[6] P. Hansen, N. Mladenović and J. A. Moreno Pérez, *Variable neighbourhood search: methods and applications*, Ann. Oper. Res., **175** (2010), 367–407, https://doi.org/10.1007/s10479-009-0657-6.

[7] T. W. Haynes, S. T. Hedetniemi and P. J. Slater, *Fundamentals of Domination in Graphs*, Marcel Dekker, New York, 1998.

[8] R. M. Karp, *Reducibility among Combinatorial Problems*. In: Miller, R. E., Thatcher, J. W., Bohlinger, J. D. (eds) Complexity of Computer Computations. The IBM Research Symposia Series. Springer, Boston, Massachusetts, 1972, pp. 85–103, https://doi.org/10.1007/978-1-4684-2001-2_9.

[9] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, 1st edn. Wiley, New York, 1990, https://doi.org/10.1002/9780470316801.

[10] C. D. Manning, P. Raghavan and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, Cambridge, 2009, http://www.informationretrieval.org/

[11] N. Mladenović and P. Hansen, *Variable neighborhood search*, Comput. Oper. Res., **24** (1997), 1097–1100, https://doi.org/10.1016/S0305-0548(97)00031-2.

[12] P-N. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining*, 1st edn. Pearson Addison-Wesley, Boston, 2005.

[13] A. Vrdoljak and D. Vukičević, *Selector of Amino-Acid Scales Set*, Math. Med. Biol., **41** (2024), 157–168, https://doi.org/10.1093/imammb/dqae007.

[14] A. Vrdoljak, *Automati grafova*, disertacija, Univerzitet u Sarajevu, Prirodno–matematički fakultet, Odsjek za matematičke i kompjuterske nauke, 2024.

[15] E. W. Weisstein, "Minimum Dominating Set". From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/MinimumDominatingSet.html

(Received: May 06, 2024)
(Revised: August 08, 2024)

Anton Vrdoljak
University of Mostar
Faculty of Civil Engineering, Architecture and Geodesy
Matice hrvatske b.b., 88000 Mostar, BiH
e-mail: *anton.vrdoljak@fgag.sum.ba*