Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 5** : 1–13.
ISSN-0037-8712
doi:10.5269/bspm.81878

Solving the Minimum Dominating Set Problem Using an Apiary Organizational–Based Optimization Algorithm

Hussein Saadi Abbas, Manal Naji Al-Harere and Ahmed T. Sadiq Al-Obaidi

ABSTRACT: The Minimum Dominating Set (MDS) problem is a fundamental challenge in graph theory due to its computational difficulty and broad relevance to coverage and optimization tasks. The problem seeks the smallest subset of vertices capable of dominating all other vertices in a graph, a requirement that becomes increasingly complex as graph size and structure grow. Because the MDS problem is NP-hard, exact methods are often impractical for large or irregular graphs, which motivates the use of heuristic strategies. In this study, an adaptation of the Apiary Organizational-based Optimization Algorithm (AOOA) is employed to generate approximate solutions for selected instances of the MDS problem. The algorithm, inspired by collective behavior in bee colonies, provides a flexible search mechanism that balances exploration and exploitation. Experimental results obtained from applying the algorithm to benchmark graph instances illustrate its ability to produce competitive dominating sets within reasonable computational time. The findings highlight the value of using nature-inspired heuristics to complement theoretical analysis and provide practical insight into domination behavior across different graph structures. The results were compared with two metaheuristic algorithms: The Bees algorithm and the Genetic algorithm. The improvement ratios showed that the AOOA algorithm outperformed both, with a 21.8% improvement in minimum domination compared to 14% for the Bees algorithm. Regarding average time, the improvement was 22.7% for GA and 34.4% for the Bees algorithm. This demonstrates the superiority of the AOOA over the other two.

Keywords: Dominating set, minimum dominating set, apiary organizational optimization algorithm, heuristic algorithm.

Contents

1. Introduction

Given an undirect graph $G = (V, E)$ consist of a set of vertices V and a set of edges E that represent the connections between these vertices. A degree of a vertex v of any graph G is defined as the number of edges incident on v. The minimum and maximum degrees of vertices in G denoted by $\delta(G)$ and $\Delta(G)$, respectively. The open neighborhood of v is the set $N(v) = \{ u \in V : uv \in E \}$ and the close neighborhood of v is the set $N[v] = N(v) \cup \{v\}$. The s adjacency matrix A of G is the n x n matrix whose ij -th entry is the number of edges joining vertex i and vertex j .

2020 *Mathematics Subject Classification:* 05C69.
Submitted February 10, 2026. Published April 09, 2026

For graph theoretic terminology, we refer to [23]. A set $D \subseteq V$ of vertices in a graph G is called dominating set if every vertex $v \in V$ is either an element of D or is adjacent to an element of D [19]. If D has no proper subset as a dominating set then it is a minimal dominating set. The domination number $\gamma(G)$ is the minimum cardinality of a dominating set D of G [13,14,18]. The minimum dominating set (MDS) problem refers to finding the smallest possible dominating set. This concept provides a fundamental way to study coverage and influence within graph networks is regarded as an essential topic in domination theory. The (MDS) problem is Known to be NP-hard which means that finding an exact (MDS) becomes increasingly difficult as the size and complexity of the graph increase as a result many studies have focused on understanding domination numbers deriving mathematical bounds and exploring how DS behave across difference classes of graph. These studies contribute to a deeper understanding of how graph structure influences domination and highlight the theoretical complexity of the problem. There are many domination types in graph which was driven by the applications in real life needs, which the researchers had offered by implying a condition on the dominating set [7,8,11,17], or on the remaining set in the graph [3,4,16,21], or on both sets [5,10,20]. Also, there are studies on stability of the dominating set in a graph [6,9].

As graphs grow larger obtaining exact solve becomes more challenging. For this reason, approximate and heuristic methods are often used to provide near-optimal solutions when exact computation is impractical. these approaches support theoretical research by allowing the examination of larger example, the identification of structural patterns and the testing of mathematical ideas on a broader range of graph instances. They do not replace the mathematical foundation of the problem. but they offer practical tools that complement theoretical analysis in this research the Apiary organization of Algorithm is used as supportive heuristic tool to generate approximate solutions for select domination problems related to the minimum dominating set.

Although the proposed algorithm is inspired by the cooperative behavior of bees in the apiary, its role in this study goes beyond a merely illustrative purpose. The algorithm is employed as a practical and effective computational tool for analyzing the minimum dominating set problem, producing consistent and reliable results that can be systematically compared with theoretical expectations. The obtained outcomes demonstrate the capability of the algorithm to capture meaningful domination patterns across different graph structures.

Overall, this study combines rigorous mathematical analysis with a carefully designed computational approach to enhance the understanding of the Minimum Dominating Set problem. This balanced perspective provides clearer insight into domination behavior in graphs and offers a solid foundation for future research in both theoretical and applied aspects of domination problems.

2. Related Work

The Minimum Dominating Set (MDS) problem is a classical topic in graph theory and combinatorial optimization. Owing to its NP-hard nature, a large body of research has focused on developing heuristic and metaheuristic methods that can produce high-quality solutions within reasonable computational time. These approaches are particularly important for large or dense graphs, where exact methods quickly become infeasible. Among the most successful techniques for solving the MDS problem are local search–based algorithms. Several studies have demonstrated that carefully designed neighborhood exploration strategies can significantly improve solution quality. For instance, fast local search frameworks that incorporate inference or reduction rules have been shown to efficiently handle large-scale graph instances. Such methods emphasize intensification around promising solutions while reducing unnecessary search effort, resulting in strong performance on benchmark datasets. To further enhance local search methods, researchers have proposed algorithms that integrate preprocessing and adaptive strategies. These approaches typically apply domination-based reduction rules and multi-stage search mechanisms to guide the algorithm more effectively toward smaller dominating sets. The results reported in these studies highlight the importance of combining problem-specific knowledge with local improvement strategies to achieve stable and competitive performance. In addition to local search, metaheuristic approaches such as greedy-based and trajectory-based algorithms have also been widely explored. Iterated greedy methods, for example, rely on partial destruction and reconstruction of solutions to escape local optima. Despite their relative simplicity, such algorithms have been shown to achieve near-optimal results on a wide range

of benchmark graphs, demonstrating that well-designed heuristics can be highly effective for domination problems. In [15] they proposed a new hybrid method based on genetic algorithm (GA) to solve the MDS problem, called shortly HGA-MDS. The proposed method invokes a new fitness function to effectively measure the solution qualities. The search process in HGA-MDS uses local search and intensification schemes beside the GA search methodology in order to achieve faster performance. Also, the performance of the HGA-MDS is compared with the standard GA. The new invoked design elements in HGA-MDS show its promising performance compared with standard GA. In [22] they propose an efficient local search algorithm namely NuMDS to solve the MDS, which comprises three key ideas. They introduced a dominate propagation-based reduction method that fixes a portion of vertices in a given graph. They developed a novel two phase initialization method based on the k-shell decomposition method. Also, they proposed a multistage local search procedure, which adopts three different search manners according to the current stage of the search. We conduct extensive experiments to demonstrate the outstanding effectiveness of NuMDS, and the results clearly indicate that NuMDS outperforms previous state-of-the-art algorithms on almost all instances. More recently, memory-based techniques, such as tabu search [12], have been applied to the MDS problem. By incorporating short-term memory and diversification strategies, tabu-based algorithms can avoid cycling and premature convergence. These methods have demonstrated strong performance on difficult instances, reinforcing the value of memory-guided search in complex combinatorial optimization problems. Despite the extensive research on local search, genetic algorithms, and tabu-based methods, relatively limited attention has been given to organizational-based metaheuristics inspired by structured social systems. In particular, apiary-inspired optimization frameworks, which model hierarchical interactions between queens, workers, and drones, remain underexplored in the context of the Minimum Dominating Set problem. This observation motivates the present study, which investigates the effectiveness of an Apiary Organizational Optimization Algorithm (AOOA) for computing minimum dominating sets and provides a complementary perspective to existing approaches.

Example 2.1 Let G be a graph of 9 vertices that is illustrated in the following figure, there are only five minimal dominating sets, as follows:

$D_1 = \{v_4\ ,\ v_5\ \}$
$D_2 = \{v_1\ ,\ v_3\ ,\ v_5\}$
$D_3 = \{v_2\ ,\ v_5)$
$D_4 = \{v_6\ ,\ v_1\}$
$D_5 = \{v_2\ ,\ v_8,\ v_9\}$, where the minimum dominating set are $D_1,\ D_3,\ and\ D_4$.
Therefore, $\gamma(G) = |D| = 2$

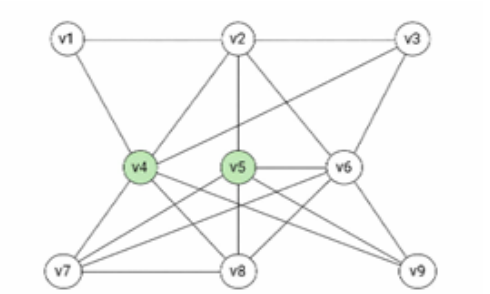

Figure 1: Minimum dominating set

3. AOOA Description

Let $G = (V, E)$ be a graph of order n . The concept of domination in a graph depends on the vertex's adjacency.

First step: In the beginning representation in the AOOA algorithm is done in the form of a binary vector with a length equal to the number of vertices (nodes) of a graph. The vertex (v_i) belonging to the dominating set is represented as ($x_i = 1$) , and the vertex (v_i) not belonging to the dominating set is represented as ($x_i = 0$). This representation allows for representing any dominating set within the graph.

In the second step: Population Generation (Initialization) a number (N) of random solutions is generated (meaning the population), where the solution represents any potential dominating set. After that, the fitness function is calculated for each solution. Subsequently, the solutions are evaluated and divided into groups of hives, and the best solution in every hive is chosen as a queen. The queen represents the best dominating set in the hive in terms of the lowest number of nodes and covering all nodes of the graph.

In the third step: The fitness function is defined as follows:

Fitness Function $(x) = |D| + a \times u$, Where:

$|D|$: Is the number of vertices in the dominating set.

u: Is the number of uncovered vertices.

a: Is a large penalty factor/coefficient.

This function ensures the rejection of any solution that does not dominate all vertices and prefers dominating sets with the smallest size.

Drone Exchange: Two random hives are selected, and some drones (meaning "solutions") are exchanged between them. This step provides diverse dominating sets and facilitates the discovery of new vertex distributions within the dominating set. *Queen Fertilization*: In every hive, the best drones are selected, and their solutions are merged with the queen's solution to generate new solutions within the hive. This means merging two dominating sets to produce a new set that combines the advantages of both. This results in reducing the size [of the set] while maintaining coverage. *Worker Life Cycle*: This is the life cycle of the workers. The workers perform local adjustments based on age, for example: adding a (node) to the set, removing an unnecessary (node), or swapping one node for another. This step represents a local improvement for the dominating set (removing redundant nodes while preserving the condition of dominance). *Queen Investiture (Replacing the Queen):* As for the process of replacing the queen (Queen Investiture): If a new solution is found that is better than the current queen, the queen is replaced by the new solution, such that the new queen represents a smaller (minimal) dominating set or is more efficient in covering the nodes. *Fading Out & Swarming: In* the Fading Out stage, the worst drone or worst worker is deleted, and large [inefficient] dominating sets are discarded. In the Swarming stage, if the size of the hive increases, it is divided into two hives. This allows for the prevention of premature convergence and the exploration of different regions within the dominating sets. *Conclusion/Final Selection*: In the end, the final solution is selected when the iterations differ/end. The best queen is chosen at the level of all Hives. This solution represents the approximate minimum dominating set for the graph.

4. The Proposed Apiary Organizational–Based Optimization Algorithm for Finding MDS

In [2] the nature inspired using metaheuristic optimization algorithm (AOOA) to solve the FJSSP. In [1] a new metaheuristic algorithm is introduced based on the behavior of bees which have one of the most intelligent and organized societies. This is achieved by investigating bees' behavior and activities during the lifecycle inside the apiary. The rationale behind proposing this algorithm is that for NP-hard optimization problems, according to the "no free lunch" (NFL) theorems, no one optimization algorithm can solve all the optimization problems. Therefore, there is room for developing new algorithms to enhance the quality of solutions. A beehive comprises several components that collectively create the perfect environment for bees to live and thrive. By examining the intelligent behavior of bees inside the apiary, it was found that seven main stages can be converted into a nature inspired metaheuristic algorithm that simulates the organizational behavior of a honeybees inside their apiary.

4.1. AOOA Pseudo Code

The following algorithm applies an AOOA algorithm [1], based on adjacency matrix and then utilized it to find the minimum dominating set.

algorithm: AOOA for MDS

 # Initialization
Use Eq. $N = b \times h$ (1)
to make N solution
Find f of solution
Choose Q_i
Split the rest of the Solutions into two groups: (w) and (d)
Set w^g to 1
while (not termination condition) do
 # Drone Exchange
for $i = 1$ to N do
for $j = 1$ to *exchratio* do
Choose two hives ha and *ho* randomly
Choose two (d)s randomly, d_1 from ha and d_2 from ho
$h_a \longleftarrow d_2$
$h_0 \longleftarrow d_1$
end for
end for
 # Bees and Fertilization
loop: For $m = 1$ to *fert – radio*, do
Choose d_{best} and w_{best} in h_i
Use Eq. (2):

$$New\_b_{H_i} = \begin{cases} Q_{Hi} + r.\ (d_{hi} - Q_{Hi}, & if\ \left(\frac{d_{Hi} - Q_{Hi}}{d_{Hi}}\right) < 0.5 \\ \frac{1}{2}\ (Q_{Hi}) + r\ .\frac{1}{2}\ (d_{Hi}),\ 0.W & \end{cases} \quad (2)$$

to fertilize Q_i with d_{best} and make a few new B_s
end for
 # The worker's life cycle
Fertilize Q_i with w_{best} using Eq. (3):

$$NW_{Hi} = W_{Hi} + r.\ (\frac{W_{H_i}^{max\text{-}g} - W_{H_i}^{g}}{W_{H_i}^{max\text{-}g}}) \times W_{Hi} \quad (3)$$

to generate Several $new - b(s)$
for each W_s do
case Age of W_s do
1 to 10:
Apply simple fertilization to US.
11 to 25:
use 2 – optima fertilization.
26 to 50:
use 3- optima fertilization.
51 to 60:
use 2 -optima fertilization.
otherwise:
drop W_s
end case
end for
 # Queen Investiture
if $(f\ (new - b) - f\ (Q_i) \geq \theta$
or no convergence to the optimal solution in h_i) then
Drop Q_i
proceed to step loop
end if
 # fade out

Drop D_{worst} and W_{worst} from H_i
 # Swarming
$h = 0$
if the size of h is greater than S_{ratio} then
Split h_i into two hives as shown below
the first 1/2 W_{hi} will remain in the hive
while the second 1/2 W_{hi} will be transferred to the new one $h(n+1)$
1/2 d_{hi} will remain in h_i
while the second 1/2 d_{hi} will be transferred to the new hive $h(n+1)$
Q_{hi} remains
proceed to step loop to generate the queen of the new hive $h(n+1)$
end if

$$h = h + l_i$$

$$N = N + h_i$$

end while
End
List of symbols used in algorithm as follows.
F : objective function
Q_i : best solution selected as the Queen
W : worker
d : drone
W^g : the age of the worker
$exch_{ratio}$: number of drones to exchange
O : the threshold of fitness
d_{best} : best drone
W_{best} : best worker
New_b : new generated bee
d_{worst} : drone with the worst fitness
W_{worst} : worker with the worst fitness
(termination condition either reach the desired results or reach max iteration)

5. Dominating Set Problem Solving Via AOOA

The **Minimum Dominating Set** problem is solved using the **AOOA metaheuristic algorithm in section 4.1**. To clearly illustrate the mechanism and operational steps of the proposed approach, the solution process is explained through an illustrative example based on the graph shown in Fig. 2, which consists of 9 vertices and is represented by its adjacency matrix. $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$ the relations between vertices are represented by "Adjacency Matrix".

The MDS problem is modelled using a binary representation, where each candidate solution is expressed as a binary vector composed of values {0,1}. The length of this vector is equal to the number of vertices in the graph. A value of **1** indicates that the corresponding vertex is selected as part of the dominating set, while a value of **0** indicates that it is not selected. Based on this encoding, the AOOA algorithm searches for an optimal binary vector that satisfies the domination constraints while minimizing the total number of selected vertices.

How the algorithm works:

Consider the graph in Fig. 2. We start by "Generate Initial Solution", so we choose 4 vertices for example (see Fig. 3) as part of the dominating vertices in graph G . So, if we chose $D_1 = \{v_1, v_2, v_4, v_6\}$, then we represent the solution we chose vertices as a binary vector $[0, 1]$. The mechanism of vector representation is putting 1 for the chosen vertex, and 0 for others. The solution matrix will be $[1, 1, 0, 1, 0, 1, 0, 0, 0]$.

Then we choose another set showing a selected solution, like $D_2 = \{v_1, v_3, v_5, v_6, v_9\}$, accordingly the solution matrix would be $[1, 0, 1, 0, 1, 1, 0, 0, 1]$ (see Fig. 4).

In this step we do the "Breading" procedure between Drone and Queen, to generate an offspring by combining queen and drone, as queen represent the best solution so far, while drone is a new solution. The combination between the two sets is "crossover" by selecting a random cut point, where we take the first part from queen up to the point and the second part from the drone. Hence, we have a new solution, evaluate it via fitness function to evaluate domination, and compare it to queen. Now if it shows better results than queen, it became the queen, we keep the queen.

According to the above solution:

Matrix Solution (1): $[1, 1, 0, 1, 0, 1, 0, 0, 0]$ = queen

Matrix Solution (2) : $[1, 0, 1, 0, 1, 1, 0, 0, 1]$ = drone

Offspring $[1, 1, 0, 1, 0, \ 1, 0, 0, 1]$ which represent the $\{v_1, v_2, v_4, v_6, v_9\}$

We check the dominating set and compare it to the previous solution. It is clear that the solution is valid, but it isn't better than the old one, so we keep queen as it is. We keep do the operations, till we reach the best solution, which means the smallest subset in the given example which is 2 vertices, which is the smallest dominating set $D3 = \{V_4, \ V_5\}$ (see Fig. 5).

$Queen \ = \ [0, 0, 0, 1, 1, 0, 0, 0, 0]$

Regarding the mathematical operations in equation 2 and 3, and because the special case of representing the problem, we made the summation as insertion, while subtraction became "Exclusive", which means extracting non-common characteristics. Finally, multiplication is represented as distribution.

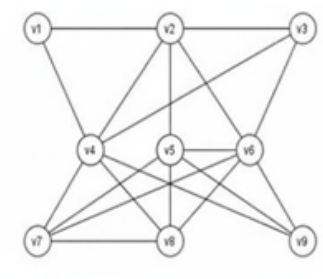

Figure 2: Graph G of 9 vertices.

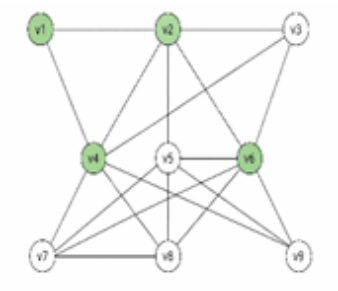

Figure 3: Set D_1 in G.

Figure 4: Set D_2 in G

Figure 5: Minimum dominating set in $G(D_)$

6. Dataset Characteristics

In this study, we utilize a collection of benchmark graph instances obtained from the DIMACS Implementation Challenges, a well-known series of competitions organized by the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) at Rutgers University. These datasets were

originally created to support research in graph theory, combinatorial optimization, and algorithm design, providing standardized instances that allow fair comparison between different heuristic and exact methods. The DIMACS graphs are publicly available and can be accessed through the official repository hosted at Rutgers University (https://dimacs.rutgers.edu/programs/challenge), where researchers worldwide contribute and evaluate algorithms on common benchmark instances. The dataset includes graphs of varying sizes, densities, and structural properties, making it suitable for evaluating the performance of minimum dominating set algorithms under diverse conditions.

The DIMACS collection contains several widely used graph families, such as brock200-2, hamming10-4, keller4, c-fat200-5, and gen200-p0.9-44, each designed to test different aspects of algorithmic performance. For example, the brock graphs are dense and challenging, the hamming graphs exhibit highly regular structure, and the c-fat graphs represent clustered or community-like structures. These variations allow researchers to analyze how an

algorithm behaves on sparse vs. dense graphs, structured vs. random graphs, and small vs. large instances. In our experiments, these datasets were selected because they represent standard benchmarks used in many recent studies on domination problems, enabling meaningful and reproducible comparisons.

7. Expermeintal and Results

To evaluate the performance of the proposed Apiary Organizational Optimization Algorithm (AOOA) on the Minimum Dominating Set (MDS) problem, a series of controlled experiments were conducted using 64 benchmark instances from the widely recognized DIMACS graph dataset. These instances span a variety of graph types and structural complexities, allowing for a comprehensive assessment of the algorithm's robustness and adaptability. All experiments were implemented in Python, and each instance was executed ten independent times to reduce the influence of randomness and to ensure stable statistical observations.

For every instance, the AOOA algorithm was required to compute a dominating set and determine the smallest set size achieved across the ten runs. In addition, two key performance metrics were recorded: Average Dominating Set Size (Avg-MDS): For each instance, the dominating set size obtained from all ten executions was averaged. This metric provides insight into the algorithm's consistency and its ability to converge toward high-quality solutions across repeated trials.

Average Execution Time (Avg-Time): The computational time (in seconds) required to complete each run was recorded, and the average over ten runs was calculated. This metric reflects the efficiency of AOOA and its practical feasibility when applied to graphs of varying sizes and densities.

Conducting multiple runs per instance allows the analysis to capture both the central performance tendency (through averages) and the algorithm's stability (through the limited variance observed between runs). The use of 64 DIMACS instances further ensures that results are not biased toward a specific graph structure, as the dataset contains random graphs, structured graphs, and challenging benchmarks commonly used in evaluating graph algorithms.

Overall, the experimental results demonstrate the capability of the AOOA algorithm to generate competitive dominating set solutions within reasonable computational time. The reliance on average values across repeated executions provides a reliable performance profile that mitigates the stochastic nature of metaheuristic optimization, thereby offering a clear indication of the algorithm's practical effectiveness on the MDS problem.

Table 1: Experimental result of AOOA on DIMCS compared with Bees and Genetic algorithm

Instance	Min			Min. Avg.			Avg. Times(s)		
	GA	Bees	AOOA	GA	Bees	AOOA	GA	Bees	AOOA
brock 200 -1	3	3	3	3	3	3	0.2	0.3	0.15
brock 200 - 2	5	5	5	5.8	5.8	5.8	0.4	0.5	0.21
brock 200 - 3	4	4	4	4.5	4.4	4.2	0.2	0.3	0.11
brock 200 - 4	4	4	4	4	4	3.8	0.13	0.23	0.16

Continued on next page

Table 1 – *Continued from previous page*

Instance	Min			Min. Avg.			Avg. Times(s)		
	GA	Bees	AOOA	GA	Bees	AOOA	GA	Bees	AOOA
brock 400 -1	3	3	3	3.8	3.8	3.8	0.13	0.23	0.16
brock 400 - 2	3	3	3	3.3	3.4	3.8	0.32	0.45	0.252
brock 400 - 3	3	3	3	3.3	3.4	3.8	0.35	0.43	0.262
brock 400 - 4	4	4	4	4	4	4	0.4	0.5	0.25
brock 800 - 1	5	5	5	5.9	5.9	5.8	0.8	0.9	0.46
brock 800 - 2	5	5	5	5.8	5.8	5.6	0.8	0.95	0.46
brock 800 - 3	5	5	5	5.4	5.4	5.2	0.9	1	0.54
brock 800 - 4	5	5	5	5	5	5	0.85	0.97	0.498
C125 - 9	2	2	2	2	2	2	0.4	0.5	0.0934
C250 - 9	2	2	2	2	2	2	0.2	0.3	0.138
C500 - 9	3	3	3	3	3	3	0.4	0.5	0.218
C-fat 200-2	9	8	7	9	8	7	7.3	8.2	2.35
C-fat 200 - 5	3	3	3	3	3	3	0.4	0.5	0.144
hamming 6-8	2	2	2	2	2	2	0.08	0.1	0.074
hamming 6-4	4	4	4	4	4	4	0.65	0.7	0.196
hamming 8-2	2	2	2	2	2	2	0.69	0.78	0.198
hamming 8-4	2	2	2	2	2	2	0.32	0.45	0.18
hamming 10-2	2	2	2	2	2	2	1.1	1.4	0.85
hamming 10-4	2	2	2	2.8	2.8	2.8	1.3	1.8	0.912
keller 4	2	2	2	2	2	2	0.32	0.5	0.146
keller 5	3	3	3	3.9	3.9	3.8	0.73	0.9	0.452
Main – a27	2	2	2	2	2	2	0.3	0.4	0.2
Main - a9	2	2	2	2	2	2	0.1	0.1	0.098
Main - a45	2	2	2	2.8	2.8	2.6	0.7	0.8	0.576
p-hat 300-3	3	3	3	3	3	3	0.4	0.5	0.228
p-hat 500-2	7	7	6	7	7	6.4	0.6	0.7	0.48
p-hat 500-3	4	4	4	4	4	4	0.4	0.5	0.32
p-hat 700-2	10	9	8	10.2	9.4	8.2	0.63	0.83	0.5
p-hat 700-3	4	4	4	4.8	4.8	4.4	0.59	0.63	0.4
p-hat 1000 -2	10	11	9	9.5	10.5	8.6	0.62	0.93	0.58
p-hat 500-3	5	5	5	5.4	5.4	5.4	1.9	2.3	0.8
c-fat 300-5	8	7	6	8	7	6	0.7	0.84	0.42
c-fat 300-10	3	3	3	3	3	3	0.4	0.53	0.3
Dsjc 500-5	8	7	6	8	7	6.2	0.41	0.52	0.3
Dsjc 1000-5	10	9	8	10	9	8	0.75	0.9	0.52
gen 200-p0-9-44	2	2	2	2	2	2	0.15	0.25	0.134
gen 200-p0-9-55	2	2	2	2	2	2	0.28	0.37	0.156
gen 400-p0-9-55	2	2	2	2	2	2	0.32	0.4	0.2
gen 400-p0-9-65	3	3	3	3	3	3	0.28	0.31	0.16
gen 400-p0-9-75	2	2	2	2	2	2.4	0.2	0.3	0.148
johnson 16-2-4	3	3	3	3	3	3	0.1	0.15	0.082
johnson 32-2-4	3	3	3	3	3	3	0.2	0.25	0.2
C 1000 – 9	3	3	3	3	3	3	0.63	0.82	0.42
san 200 - 0-7-1	2	2	2	2	2	2.8	0.15	0.2	0.1
san 200 -0-7-2	3	3	3	3	3	3	0.14	0.2	0.1
san 200 -0-9-1	2	2	2	2	2	2	0.19	0.2	0.1
san 200 -0-9-2	2	2	2	2	2	2	0.15	0.23	0.108
san 200 0-9-3	2	2	2	2	2	2	0.18	0.25	0.104
san 400 0-5-1	4	4	4	4	4	4	0.21	0.3	0.18

Continued on next page

Table 1 – *Continued from previous page*

Instance	Min			Min. Avg.			Avg. Times(s)		
	GA	Bees	AOOA	GA	Bees	AOOA	GA	Bees	AOOA
san 400 0-7-1	3	3	3	3.6	3.6	3.6	0.28	0.32	0.178
san 400 0-7-2	4	4	4	4	4	4	0.15	0.2	0.184
san 400 0-7-3	3	3	3	3.8	3.8	3.8	0.2	0.3	0.176
san 400 0-9-1	2	2	2	2	2	2	0.22	0.31	0.174
san 1000	4	4	4	4	4	4	0.75	0.8	0.44
san r 200-0-7	3	3	3	3	3	3	0.2	0.2	0.1
san r 200-0-9	2	2	2	2	2	2	0.2	0.2	0.1
san r 400-0-5	8	7	6	8	7	6	0.21	0.3	0.19
san r 400-0-7	4	4	4	4	4	4	0.27	0.35	0.186
johnson 8-2-4	3	3	3	3	3	3	0.15	0.1	0.082
johnson 8-4-4	2	2	2	2	2	2	0.13	0.1	0.07

8. Discussion

This section presents the main results of the AOOA algorithm and compares them with both the Genetic Algorithm (GA) and the Bees Algorithm when applied to the DIMCS benchmark datasets. The comparison is based on three main performance metrics: minimum dominating set, minimum average dominating, and average execution time.

For the small and medium datasets, the results of the minimum dominating and minimum average dominating are nearly similar among the compared algorithms. However, for the large datasets, the use of the AOOA algorithm shows noticeable superiority, particularly in terms of minimum average dominating.

In terms of execution time, the AOOA algorithm is faster than both GA and Bees. Regarding the minimum dominating set, the improvement percentage between GA and AOOA is 21.8%, while the improvement percentage between Bees and AOOA reaches 14 % for the 64 instances. In addition, the improvement percentage in average time between GA and AOOA is 22.7%, whereas between Bees and AOOA it reaches 34.4%.

These results clearly indicate the superiority of the AOOA algorithm over both GA and Bees in terms of solution quality and execution time for the minimum dominating set problem. In general, the results confirm that AOOA not only improves solution quality compared to traditional algorithms, but also outperforms them in several cases in terms of stability and execution speed.

This demonstrates that the AOOA algorithm represents an effective and promising approach for solving DIMACS problems, especially in datasets that require high computational efficiency and reduced execution time.

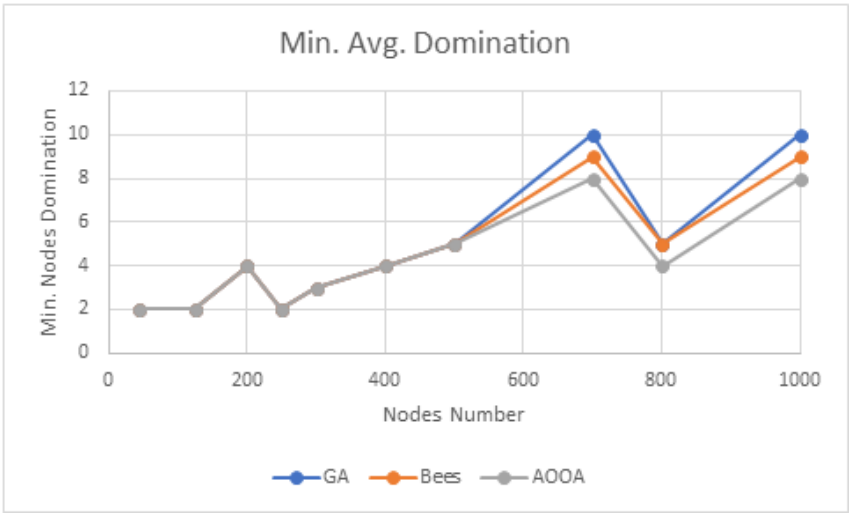

Figure 6: minimum average domination compared with BEES nd GA algorithm.

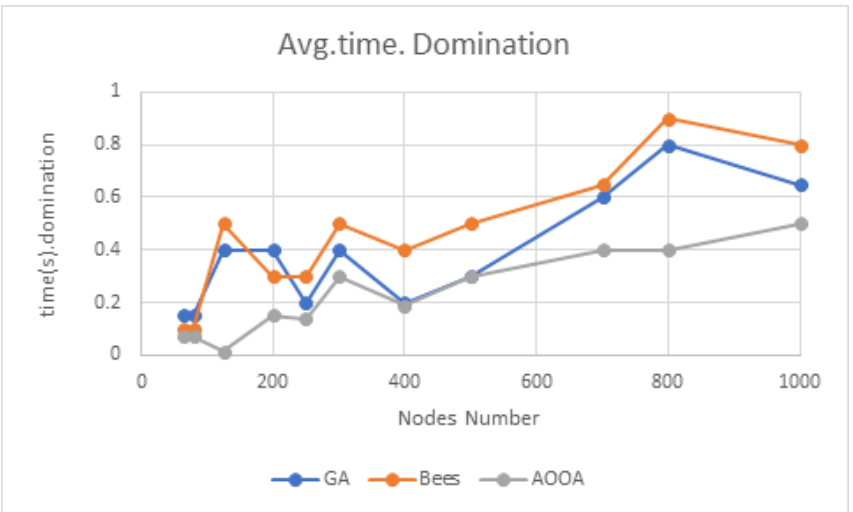

Figure 7: Average time(s) domination compared with BEES and GA algorithm.

The above Fig. 6, shows the obtained the average of the minimum dominating problem. The row represents the node number, and the column represents the minimum dominating. The table presents a comparison of the results using the same dataset. The results of three algorithms—GA, Bees, and AOOA—are compared. The results show the superiority of AOOA on the large dataset, and these results are highlighted using gray-colored text. This indicates that as the number of nodes increases, the differences between the algorithms begin to appear.

Regarding the second Fig. 7, it shows the average time (s) required in seconds. The average time was calculated for all the data set Found in the Table 1. The average time was computed over 10 iterations,

after which the average value was obtained. The results are illustrated using graphical representations, showing the relationship between time and the size of the graph for both GA and Bees algorithms. The AOOA algorithm demonstrates better performance in terms of time efficiency, especially for large-scale graphs when the number of nodes reaches 600 or more, where the algorithms become time-consuming.

9. Conclusion

This paper applied an algorithm for solving minimum dominating set problem in graphs by developing metaheuristic optimization algorithm which was the so called AOOA. In terms of execution time, the AOOA algorithm exhibits faster execution than both GA and Bees. Regarding the minimum dominating set, the improvement percentage between GA and AOOA is 21.8%, while the improvement percentage between Bees and AOOA reaches 14 % for the 64 instances. In addition, the improvement percentage in average time between GA and AOOA is 22.7%, whereas between Bees and AOOA it reaches 34.4%.

References

1. M. A. Al-Sharqi, A. T. S. Al-Obaidi and S. O. Al-Mamory, *Apiary organizational-based optimization algorithm: a new nature-inspired metaheuristic algorithm*, Int. J. Intell. Eng. Syst. 17,3, 2024, 783-801, (2024).
2. M. A. Al-Sharqi, A.T. Sadiq, and S. O. Al-mamory, *Flexible Job Shop Scheduling Problem-Solving Using Apiary Organizational-Based Optimization Algorithm*, ARO-The Scientific Journal of Koya University 12,2, 94-106, (2024).
3. M. A. A. Abdlhusein, and M. N. Al-Harere , *Some modified types of pitchfork domination and its inverse*, Boletim da Sociedade Paranaense de Matemática, 40, 1–9,(2022).
4. D. A. Abbass, M. N. Al-Harere and E.B. Al-Zangana , *Inverse Planar Domination and Independent Planar Domination in Graphs*, Boletim Da Sociedade Paranaense De Matematica, 43,1-7,(2025)
5. M. N. Al-Harere, and A. T. Breesam, *Further results on bi-domination in graphs*, *AIP Conf. Proc.* 2096 (1), 020013, (2019).
6. M. A. A. Abdlhusein, and M. N. Al-Harere , *Stabile and Critical Pitchfork Domination in Graphs,* Boletim da Sociedade Paranaense de Matemática, 43,1-10,(2025).
7. D. A. Abbass, M. N. Al-Harere and E.B. Al-Zangana, On planar domination in graphs, Journal of Discrete Mathematical Sciences and Cryptography, 28,4-B, 1355–1360**,** **(**2025).
8. M. N. Al-Harere, and P.A. Khuda Bakhash,*Tadpole domination in duplicated graphs*, Discrete Mathematics, Algorithms and Applications, 13, 2150003, (2021).
9. M. N. Al-Harere, and P.A. Khuda Bakhash, *Changing and unchanging on tadpole domination in $G-e$, $G+e$ graphs,* Turkish World Mathematical Society Journal of Applied and Engineering Mathematics,13, 1151–1159, (2022).
10. M. A. A. Abdlhusein, and M. N. Al-Harere,*Doubly connected pitchfork domination and its inverse in graphs*, Turkish World Mathematical Society Journal of Applied and Engineering Mathematics, 12, 82-91, (2022).
11. S. S. Abed, and M. N. Al-Harere, *On rings domination in graphs*, International Journal of Mathematics and Computer Science. 17, 1313–1319,(2022).
12. J. M. Colmenar, M. Laguna and R. Martín-Santamaría, *Tabu search: an application to the minimum dominating set problem*, 33, 304-326, (2025).
13. T. W. Haynes, S. T. Hedetniemi and P.J. Slater, *Domination in Graphs Advanced Topics*, Marcel Dekker Inc., (1998).
14. S.T. Hedetneimi and R. Laskar, *Topics in Domination in Graphs*, Discrete Math., (2025).
15. A.R. Hedar and R. Ismail, *Hybrid genetic algorithm for minimum dominating set problem*, Lect. Notes Comput. Sci. 6019 , 457-467, (2010).
16. S. Sh. Kahatand, and M. N. Al-Harare, *Inverse equality co-neighborhood domination in graphs,* Ibn Al-Haitham International Conference for Pure and Applied Sciences, Journal of Physics: Conference Series, 1879, 3, 032036, (2021).
17. S. S. Majeed, A. A. Omran, and M. N. Yaqoob, *Modern Roman domination of corona of cycle graph with some certain graphs*, International Journal of Mathematics and Computer Science,17, 317–324, (2022).
18. O. Ore, *Theory of graphs*, American Mathematical Soc, (1962).
19. A. A. Omran, *Domination and independence in cubic chessboard*, Engineering and Technology Journal ,34, 64-59,(2016).
20. A. A. Omran, *Domination and independence on square chessboard*, Engineering and Technology Journal,35, 68-75,(2017).
21. S. Salah, A.A. Omran, M.N. Al-Harere,*Calculating modern roman domination of fan graph and double fan graph*, Journal of Applied Sciences and Nanotechnology 2 ,2, 47-54,(2022).

22. R. Sun, Z. Liu, Y. Wang, H. Xiao, J. Li and J. Chen, *NuMDS: An efficient local search algorithm for minimum dominating set problem*, Proc. Int. Joint Conf. Artif. Intell. (IJCAI-25,(2025).

23. R. J. Wilson, *Introduction to graph theory,* Longman, (1996).

Hussein Saadi Abbas,
Department of Mathematics and computer Applications,
College of Applied Sciences,University of Technology,
Iraq.
E-mail address: as.24.31@grad.uotechnology.edu.iq

and

Manal Naji Al-Harere,
Department of Mathematics and computer Applications,
College of Applied Sciences,University of Technology,
Iraq.
E-mail address: Manal.N.Alharere@uotechnology.edu.iq

and

Ahmed T. Sadiq Al-Obaidi,
College of Computer science, University of Technology,
Iraq.
E-mail address: Ahmed.t.sadiq@uotechnology.edu.iq