Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 3, pp. 887–905
DOI : 10.47974/JIOS-1583

Optimizing minimum dominating set using an enhanced binary whale algorithm

Belkacem Zouilekh [†]
Sadek Bouroubi *
L'IFORCE Labortory
Faculty of Mathematics
University of Sciences and Technology Houari Boumediene
Bab Ezzouar P. B. 32 El-Alia
Algiers 16111
Algeria

Abstract

The Whale Optimization Algorithm (WOA) is an innovative metaheuristic inspired by the social hunting behavior of humpback whales. This paper presents a binary version of the WOA applied to the Minimum Dominating Set (MDS) problem, a challenging combinatorial optimization problem known to be NP-hard. However, the WOA suffers from premature convergence, potentially causing the algorithm to become trapped in local optima and fail to reach the global optimum. To address this issue, the paper introduces an enhanced version of WOA called the Local Search Optimization Binary Whale (Lobw), which incorporates a Local Search technique to improve exploitation ability and prevent the algorithm from getting stuck in local optima. The Lobw algorithm is evaluated on several benchmark datasets, demonstrating its promising performance in terms of solution quality and stability compared to other metaheuristic algorithms. The source code of the Lobw algorithm and the datasets used in the experiments can be accessed via the following link: https://github.com/elkacem/LOBW-for-MDS.

Subject Classification (2010): *05C69, 90C59, 90C27.*

Keywords: *Minimum dominating set, Metaheuristics, Whale optimization algorithm, Local search.*

1. Introduction

One of the most significant problems in graph theory is the Minimum Dominating Set (MDS). It is also an NP-hard problem, meaning it cannot be solved in polynomial time [1]. In an undirected graph $G = (V, E)$, where V

[†] *E-mail:* **bzouilekh@usthb.dz**
* *E-mail:* **sbouroubi@usthb.dz (Corresponding Author)**

denotes the set of vertices and E denotes the set of edges, a subset S of V is considered a dominating set of G if every vertex $v \in V$ is either in S or adjacent to a vertex in S. This subset is called a dominating set of G, and we say that S dominates G or G is dominated by S. The smallest size of a dominating set in G is denoted by $\gamma(G)$, known as the domination number. If S dominates G and $|S| = \gamma(G)$, then S is called the Minimum Dominating Set (MDS). Therefore, the problem of finding a dominating set involves determining the value of $\gamma(G)$ and identifying a dominating set with the minimum number of vertices. The mathematical formulation of this problem is as follows:

$$\min f(x) = \sum_{v_i \in V} x_i \tag{1}$$

$$\text{subject to:} \begin{cases} \sum_{v_j \in N[v_i]} x_j \geq 1, & \forall v_i \in V \\ x_i \in \{0,1\} \end{cases} \tag{2}$$

where x_i is a binary indicator that takes the value 1 if the vertex v_i is part of the solution S, and 0 if it is not, and $N[v_i]$ is the closed neighborhood of v_i. The first set of constraints ensures that every vertex is either in the minimum dominating set or has a neighbor in the set.

Minimum dominating sets, together with their variations such as minimum connected dominating set and minimum weighted dominating set, find diverse applications across various fields. These applications include ad hoc wireless networks [2], sensor networks, and MANETs [3] for routing purposes. They are also utilized in determining main subject sentences for document summarization via sentence network design [4]. Additionally, MDS can model and study the positive impact in social networks [5], control the spread of epidemics, and diagnose and manage epidemic spread in domains ranging from virus propagation in computer systems to misinformation and content diffusion through social media [6]. Furthermore, MDS appears in biological contexts such as interacting protein networks and biological network analysis [7].

This study employs the Whale Optimization Algorithm (WOA) to tackle the MDS problem, aiming to capitalize on WOA's benefits, such as its minimal parameter count and rapid convergence rate. The algorithm's effectiveness is assessed across diverse datasets and benchmarked against several state-of-the-art techniques. WOA, a metaheuristic, has demonstrated successful applications in addressing various optimization challenges, notably including its role in epileptic detection [8].

The remainder of this paper is organized as follows: Section 2 reviews related work, encompassing exact algorithms and heuristics applied to the MDS problem. Section 3 outlines the Whale Optimization Algorithm (WOA). The proposed WOA algorithm specifically for MDS is detailed in Section 4. Section 5 presents the experimental results, and Section 6 concludes the paper while discussing potential future research directions.

2. Related works

Exact (exponential-time) methods have been extensively utilized to explore the NP-hard MDS problem [1, 9]. Recent efforts to address the MDS problem in a

graph with n vertices have yielded exact algorithms that improve upon the naive $\mathcal{O}(2^n)$ approach, which simply examines all possible subsets of V. According to current knowledge, the most efficient exact algorithm for the MDS problem operates in $\mathcal{O}(1.4864^n)$ time and polynomial space. This algorithm was developed using the measure and conquer strategy by Iwata [10].

However, exact methods operating at exponential scales are practical only for small networks, severely limiting their applicability. Consequently, researchers have predominantly focused on stochastic computational heuristics and, more recently, metaheuristics.

This section offers a brief overview of heuristic-based techniques for minimum dominating set (MDS). These methods can be categorized in various ways, including whether they are inspired by natural phenomena or not [11]. Many metaheuristics draw inspiration from natural processes and are classified into two main categories: (I) evolutionary algorithms and (II) swarm intelligence. Evolutionary algorithms mimic biological evolution principles such as crossover and mutation. Swarm intelligence systems are inspired by the collective behaviors observed in ants, fireflies, whales, grasshoppers, and other natural organisms. In contrast, metaheuristics inspired by non-natural events include Simulated Annealing (SA) [12] from physics and Harmony Search (HS) [13] from music.

In this context, several heuristic approaches have been advanced in the literature to tackle the MDS problem, including those proposed by [14-17]. Following these efforts, Sanchis [18] conducted an empirical investigation into various heuristic techniques. Sanchis extensively studied numerous greedy methods for the MDS problem.

Ant Colony Optimization (ACO) [19] stands out as one of the prominent swarm intelligence algorithms. Ho, Singh, and Ewe introduced ACO-TS in [20], an enhanced version of Ant Colony Optimization that integrates a mechanism inspired by genetic algorithms known as tournament selection to encourage the generation of diverse solutions.

Genetic Algorithm (GA) is a traditional evolutionary algorithm [21]. The GA was initially used to address the MDS issue in 2010 [22]. Two years later, the authors of [23] proposed Simulated Annealing, one of physics most well-known single-solution based metaheuristics inspired by non-natural phenomena. The Simulated Annealing (SA) metaheuristic tackles the MDS problem by employing a Stochastic Local Search (SLS) algorithm to enhance a solution by searching for a better one in its vicinity.

Later, in 2022, Abed et al. [24] introduced a two-step hybrid local search algorithm that combines different local search techniques. Initially, they employed local search as an exploratory technique, focusing on generating promising solutions across different regions of the solution space. Subsequently, another local search algorithm was used as an exploitation strategy in the second step, targeting feasible neighborhoods to enhance solution quality.

3. Whale Optimization Algorithm

Whale Optimization Algorithm (WOA) is a population-based metaheuristic optimization algorithm inspired by the hunting and feeding behavior of humpback

whales. It was introduced in 2016 [25] and used to solve optimization problems in various fields, such as engineering, finance, and computer science.

WOA uses a combination of exploitation and exploration strategies to search for the optimal solution. "Bubble-net feeding" describes their technique of locating and pursuing their prey. As illustrated in Figure 1, the humpback whales descend and then begin to blow bubbles to enclose the fish, which are too terrified to swim through them. In the meantime, the whales swim upward to the surface through the bubble net, devouring a large number of fish in a single gulp. The behavior of bubble-net feeding is mathematically represented as follows:

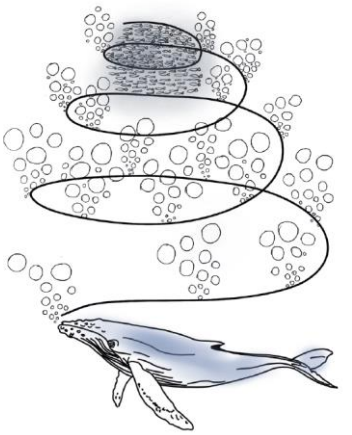

Figure 1
The bubble-net feeding behavior of a humpback whale.

3.1 *Encircling Prey*

Humpback whales are filter feeders that primarily consume small fish and krill, utilizing their positioning to locate, encircle, and pursue prey. Similarly, the Whale Optimization Algorithm (WOA) operates under the assumption that the current best candidate solution X^* represents the target prey or is very close to the optimum, given that the exact location of the optimal design in the search space is initially unknown. Therefore, the position of the prey is determined by the best whale $\overrightarrow{X^*}$ discovered up to iteration t. This mechanism is described by the following equations:

$$\vec{X}(t+1) = \overrightarrow{X^*}(t) - \vec{A}.\vec{D}, \tag{1}$$

$$\vec{D} = \left|\vec{C}.\overrightarrow{X^*}(t) - \vec{X}(t)\right|, \tag{2}$$

where $\vec{D}$ denotes the computed modified distance between the whale $\vec{X}(t)$ and the prey; $\vec{A}$ and $\vec{C}$ are coefficient vectors formulated mathematically by the following equations:

$$\vec{A} = 2a\vec{r} - a \tag{5}$$

$$\vec{C} = 2\vec{r} \tag{6}$$

Throughout the length of the iterations, a is reduced linearly from 2 to 0. In equations (5) and (6), $\vec{r}$ is an n-dimensional random vector.

3.2 *The bubble-net foraging (Exploitation)*

Humpback whale foraging generates individual bubbles in a circle, leading to what's known as bubble-net foraging, and two approaches are proposed to:

3.2.1 Shrinking encircling procedure

This effect is achieved by reducing the value of a in Equation (5) from 2 to 0 throughout the number of iterations. As a result, A is a random value in the range $[-a, a]$. The new position of a search whale may be placed anywhere between the starting location of the whale and the position of the current best whale by setting random values for A in $[-1,1]$.

3.2.2 Spiral motion updating

To emulate the helix-shaped movement of humpback whales, a spiral equation is built between the positions of whale and prey (best solution produced so far), in which l is a random number in the range $[-1,1]$ and the logarithmic spiral form is defined by the constant b.

$$\vec{X}(t+1) = \overrightarrow{D'}\exp(bl)\cos(2\pi l) + \overrightarrow{X^*}(t) \tag{7}$$

$$\overrightarrow{D'} = |\overrightarrow{X^*}(t) - \vec{X}(t)| \tag{8}$$

It should be noted that humpback whales swim around their prey in a shrinking circle and along a spiral-shaped path at the same time, assuming a 50% chance of selecting either the shrinking encircling procedure or the spiral motion to update their position. As a result, the mathematical model of this behavior is as follows:

$$\vec{X}(t+1) = \begin{cases} \text{Shrinking encircling (Equation (5))} & \text{if } p < 0.5 \\ \text{Spiral motion (Equation (7))} & \text{if } p \geq 0.5 \end{cases} \tag{9}$$

where p is a random value between $[0,1]$.

3.3 *Searching for prey (Exploration)*

Along with the bubble-net technique, humpback whales hunt for prey at random. Unlike the exploitation phase, we update a search agent's position in the exploration phase based on a randomly picked search agent $\overrightarrow{X_{rand}}$ rather than the best search agent found so far. When $|A| > 1$, a random search agent is chosen to update the position of the search agents, and the optimal solution is chosen when $|A| < 1$. Following is a description of the mathematical model:

$$\vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A} \cdot \vec{D}, \tag{10}$$

$$\vec{D} = |\vec{C} \cdot \overrightarrow{X_{rand}} - \vec{X}(t)|. \tag{11}$$

4. Lobw for Minimum Dominating Set Problem

In this section, we discuss how we intend to deal with the MDS problem. Our technique takes as input a problem instance $G = (V, E)$, where V and E represent its set of vertices and set of edges, respectively. An initial population P of N search agents (solutions) is used in the Lobw method. As a result, the size of the vector representing whale in P is equal to the number of nodes in the graph $G = (V, E)$. The $i - th$ node in G is a member of the dominant set if the $i - th$ element of the vector has a value of 1. Algorithm 1 describes the proposed approach.

4.1 *Local Search Technique*

When whales use a spiraling or shrinking motion to pursue their prey in a real-world setting, the prey continues to move. As a result, the prey would have relocated by the time the whales arrived at its location. In order to find their prey precisely, whales must therefore seek it out in the neighborhood. A local search technique has been used to simulate the local searching behavior of the whales in order to execute this scenario in the whale system, drawing inspiration from [26]. Two whales are chosen at random ($\overrightarrow{x_{r1}}, \overrightarrow{x_{r2}}$), and two other whales are created using:

$$\begin{aligned} \overrightarrow{w_i^1}(t+1) &= \overrightarrow{w_i}(t) + (\overrightarrow{x_{r1}} - \overrightarrow{x_{r2}}), \\ \overrightarrow{w_i^2}(t+1) &= \overrightarrow{w_i}(t) - (\overrightarrow{x_{r1}} - \overrightarrow{x_{r2}}), \end{aligned} \tag{12}$$

where $\overrightarrow{w_i^1}(t+1)$ and $\overrightarrow{w_i^2}(t+1)$ are the two neighbors of solution $\overrightarrow{w_i}(t)$. These two new whales are compared to the original whale $\overrightarrow{w_i}(t)$, and if either has a better fitness value, the freshly created whale with the best fitness replaces the original in the population. Algorithm 2 provides the pseudocode for this local search method.

Algorithm 1: Lobw algorithm
Require: A graph $G = (V, E)$; Local Search procedure
Ensure: Fittest whale in the population (MDS)
1: $Itern \Leftarrow$ maximum number of iterations
2: $N \Leftarrow$ number of whales
3: $t \Leftarrow 0$
4: Generate a population of N whales X_i, $i = 1,2,3, \ldots, N$
5: Compute the fitness function f_i of each whale $X_{\overrightarrow{w_i}}$ using Equation (1)
6: Sort the whales in decreasing fitness order
7: Choose the most fit whale X^* to be the prey
8: **while** $t \leq itern$ **do**
9: Reduce the value of a from 2 to 0
10: **for** $i \leftarrow 1$ **to** N **do**
11: Compute A and C using Equation (5) and (6) resp.
12: $p = random(0,1)$
13: **if** $p < 0.5$ **then**

if $A \geq 1$ **then**
// Exploration through shrinking encircling //
Pick a whale at random from the population
Update position based on Equation (10)
else
// Exploitation through shrinking encircling //
Update position based on Equation (3)
end if
else
// Spiral motion //
Update position based on Equation (7)
end if
if w_i doesn't change **then**
Choose two whales $candidate_1$ and $candidate_2$ at random from the population N other than whale $\overrightarrow{w_i}$
Perform the Local Search on whale $\overrightarrow{w_i}$
else
Continue
end if
end for
Rank whales from best to worst and find the fittest one
Update the global optimal solution X^*
end while
return X^*

Algorithm 2: Local Search
Require: Two random whales $\overrightarrow{x_{r1}}$ and $\overrightarrow{x_{r2}}$ and current solution $\overrightarrow{w_i}$
Ensure: Fittest whale
Generate two neighbors candidate solutions $candidate_1$ and $candidate_2$ using Equation (12)
f_1 = fitness $(candidate_1)$
f_2 = fitness $(candidate_2)$
Sort the two generated whales and take the fittest one as $candidate$ and its fitness f
if $f < fitness(\overrightarrow{w_i})$ **then** $\overrightarrow{w_i} = candidate$
else
return $\overrightarrow{w_i}$
end if
return $\overrightarrow{w_i}$

5. Experiment and results

The experimental results of the improved binary whale optimization approach (Lobw) are presented in this section. The first subsection lists the experimental datasets. The second subsection then provides an explanation of the performance measurements. The impact of the local search on the proposed

Lobw is discussed in the following subsection. The comparisons with relevant studies are included in this section's last subsection. Each graph from the two literature benchmarks was run ten times. Our Lobw algorithm was implemented using Python. All testing was conducted on a machine equipped with 16 gigabytes of RAM and a 2.6 GHz Intel Core i5 CPU.

5.1 *Datasets Description*

The performance of Lobw is evaluated using two benchmark datasets. This section provides a brief description of various datasets.

5.1.1 Random Geometric Graphs

Previously presented in [23] are 42 arbitrary geometric networks with up to 400 nodes. For the purpose of building the networks, the authors utilized the generating instructions from [20] and [27]. Each network is constructed by randomly placing n nodes (refer to the No. of nodes column in Table 1) within an $M \times M$ area, following a uniform distribution (see the Area column in Table 1). Graph instances were generated based on the range parameter indicated in the Range column of Table 1. It was ensured that each graph formed a connected structure following the guidelines in [27]. The Nb of instances column represents the number of graph instances per network.

Table 1
Random Geometric Graphs

Network id	Nb of nodes (n)	Range	Area (A)	Nb of instances
$N1_A^n$	400	210-240	3000×3000	4
$N2_A^n$	350	200-230	2500×2500	4
$N3_A^n$	300	180-220	2000×2000	5
$N4_A^n$	250	130-160	1500×1500	4
$N5_A^n$	200	100-160	1000×1000	7
$N6_A^n$	200	70-120	700×700	6
$N7_A^n$	100	80-120	600×600	5
$N8_A^n$	80	60-120	400×400	7

5.1.2 Known dominating set

The benchmark graphs from [23] consist of 21 graphs, each with either 400 or 800 vertices. The minimum dominating set for these graphs is determined. The authors utilized the method described in [20] to generate networks with known domination numbers and specified densities denoted as $\mathbf{G_{d,p}^n}$, where n is the number of nodes, d is the domination number, and p is the probability for edge generation. The detailed technique was initially introduced in [18]. This benchmark includes two types of graphs: one set with 400 vertices and another with 800 vertices, across various densities (0.1, 0.3, and 0.5) as shown in Table 2.

Table 2
Known dominance number benchmark

Network Id	Nb of nodes (n)	p	d	Nb of instances
$G_{d,0.1}^{400}$	400	0.1	8,11,14,18,23	5
$G_{d,0.3}^{400}$	400	0.3	3,5,8,11,14	5
$G_{d,0.5}^{400}$	400	0.5	3,5,8,11	4
$G_{d,0.1}^{800}$	800	0.1	11,14,22	3
$G_{d,0.3}^{800}$	800	0.3	3,5	2
$G_{d,0.5}^{800}$	800	0.5	3,6	2

5.2 *Performance Metrics*

The Lobw algorithm is examined using four different measures. The worst, best, mean fitness value, and standard deviation are the metrics. They are defined mathematically as follows:

$$\text{Best} = \min\{fitness_i,\ i = 1, \ldots, 10\},$$

$$\text{Worst} = \max fitness_i,\ i = 1, \ldots, 10\},$$

$$\text{Mean} = \frac{1}{10}\sum_{i=1}^{10} fitness_i,$$

$$\text{Standard deviation} = \sqrt{\frac{\sum_{i=1}^{10} |fitness_i - mean|^2}{10}},$$

$$\text{hits} = count(best) \text{ for 10 runs.}$$

Here, 10 refers to the number of runs conducted and count (best) is a variable used to keep track of the number of times the best solution appeared among the 10 runs.

5.3 *Effect of the Local Search*

To showcase how the local search of the proposed algorithm can affect its performance, we conducted a test using the suggested method named Lobw, and a version without local search named Bwoa. Both versions were compared based on the results obtained from using the same initial population of 100 randomly generated whales, while employing the truly random function for the WOA. The outcomes of ten separate runs for each instance are presented in tables N1 to N8.

Lobw produced smaller dominant sets for 7 graph examples (indicated by 0 in the hits column of Bwoa). These improvements were achieved exclusively for networks containing at least 300 nodes, with the majority occurring in network N1. In 35 instances, both versions identified the best solutions of equivalent quality. Lobw provided a significantly higher number of hits for 34 of the 35 graph cases. So far, Bwoa has produced more hits than Lobw on only one instance ($N2_{2500}^{350}$ in range 200). The Tables 3, 4, 5, 6, 7, 8, 9, and 10 were simulated and visually presented in Figure 2 to illustrate the difference in best

solution hits between the Bwoa and Lobw versions. According to this figure, Lobw could outperform Bwoa in terms of reaching the best solution. Lobw showed enhanced solution qualities that were statistically verified throughout all ten runs (see the average column avg.). The local search approach, which enables the whales to improve their exploitation by conducting searches around the most advantageous areas where they are located, is what can be used to explain why Lobw is superior to Bwoa.

Table 3
The effect of the local search in the Lobw algorithm on $N1_{3000}^{400}$

Network	Instance Range	Bwoa				Lobw			
		Best	Avg	Std	Hits	Best	Avg	Std	Hits
$N1_{3000}^{400}$	210	69	69.9	1.20	0	67	67.4	0.52	6
	220	64	65.8	1.14	0	62	64.1	1.37	1
	230	61	62.4	1.07	0	57	59.5	1.27	1
	240	55	56.1	1.45	0	54	55.2	0.63	1

Table 4
The effect of the local search in the Lobw algorithm on $N2_{2500}^{350}$

Network	Instance Range	Bwoa				Lobw			
		Best	Avg	Std	Hits	Best	Avg	Std	Hits
$N2_{2500}^{350}$	200	54	55.1	1	3	54	54.9	0.32	1
	210	48	50.6	1.71	0	47	48.5	0.71	1
	220	43	44.7	0.95	1	43	44.2	0.63	1
	230	42	43.2	0.63	1	42	42.7	0.48	3

Table 5
The effect of the local search in the Lobw algorithm on $N3_{2000}^{300}$

Network	Instance Range	Bwoa				Lobw			
		Best	Avg	Std	Hits	Best	Avg	Std	Hits
$N3_{2000}^{300}$	180	43	43.8	0.63	0	42	43.1	0.74	2
	190	39	40.8	1.03	1	39	39.9	0.57	2
	200	36	36.7	0.82	0	35	35.6	0.70	5
	210	34	34.7	0.67	4	34	34.2	0.42	8
	220	31	32.2	0.63	1	31	31,5	0.53	5

Table 6
The effect of the local search in the Lobw algorithm on $N4_{1500}^{250}$

Network	Instance Range	Bwoa				Lobw			
		Best	Avg	Std	Hits	Best	Avg	Std	Hits
$N4_{1500}^{250}$	130	47	47.1	0.32	9	47	47	0	10
	140	40	41.4	0.70	1	40	40.2	0.42	8
	150	37	37.9	0.57	2	37	37.3	0.48	7
	160	33	34.1	0.74	2	33	33.9	0.32	1

Table 7
The effect of the local search in the Lobw algorithm on $N5_{1000}^{200}$

Network	Instance Range	Bwoa				Lobw			
		Best	Avg	Std	Hits	Best	Avg	Std	Hits
$N5_{1000}^{200}$	100	35	36.4	0.84	1	35	35.1	0.32	9
	110	30	31.6	0.84	1	30	30.6	0.70	5
	120	23	23.9	0.57	2	23	23.7	0.67	4
	130	23	23.9	0.32	1	23	23.4	0.52	6
	140	20	21.1	0.57	1	20	20.8	0.42	2
	150	17	17.7	0.48	3	17	17.4	0.52	6
	160	17	17	0	10	17	17	0	10

Table 8
The effect of the local search in the Lobw algorithm on $N6_{700}^{200}$

Network	Instance Range	Bwoa				Lobw			
		Best	Avg	Std	Hits	Best	Avg	Std	Hits
$N6_{700}^{200}$	70	32	32	0	10	32	32	0	10
	80	26	26.6	0.70	5	26	26.3	0.48	7
	90	22	23.2	0.63	1	22	22.7	0.48	3
	100	18	19.1	0.74	2	18	18.4	0.52	6
	110	16	16.9	0.74	3	16	16	0	10
	120	14	14.1	0.32	9	14	14	0	10

Table 9
The effect of the local search in the Lobw algorithm on $N7_{600}^{100}$

Network	Instance Range	Bwoa				Lobw			
		Best	Avg	Std	Hits	Best	Avg	Std	Hits
$N7_{600}^{100}$	80	18	18.2	0.42	8	18	18	0	10
	90	15	15	0	10	15	15	0	10
	10	13	13.3	0.48	7	13	13	0	10
	110	11	11	0	10	11	11	0	10
	120	9	9.1	0.32	9	9	9	0	10

Table 10
The effect of the local search in the Lobw algorithm on $N8_{400}^{80}$

Network	Instance Range	Bwoa				Lobw			
		Best	Avg	Std	Hits	Best	Avg	Std	Hits
$N8_{400}^{80}$	60	15	15	0	10	15	15	0	10
	70	12	12.9	0.32	1	12	12.2	0.42	8
	80	9	9.2	0.42	8	9	9	0	10
	90	8	8	0	10	8	8	0	10
	100	7	7.5	0.53	5	7	7.3	0.48	7
	110	6	6	0	10	6	6	0	10
	120	5	5.4	0.52	6	5	5.3	0.48	7

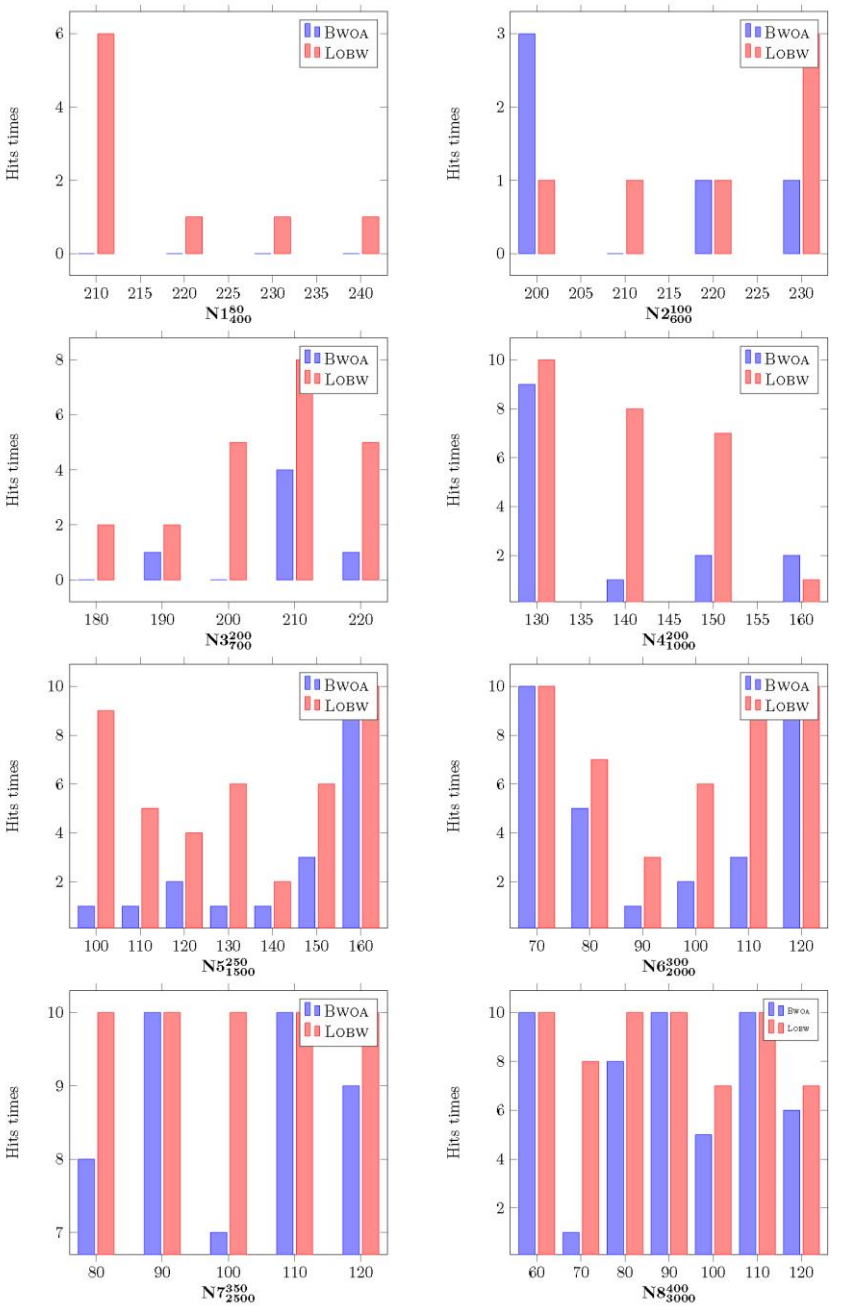

Figure 2
Hits analysis on various graphs for Lobw and Bwoa.

5.4 *Numerical results to the relevant works*

In this section, the performance of Lobw is evaluated against state-of-the-art algorithms from the literature. The datasets used for comparison were sourced from [23]. Many studies have constructed random geometric graphs (see Section 5.1.1) based on Bettstetter's work [27]. However, direct comparison with these works may be inaccurate due to potential variations introduced by the random distribution procedure used in data generation. To mitigate this issue, all methods in the comparison were evaluated on the same dataset.

The Hga [22] and Samds [23] values were extracted from [23]. Nonetheless, this does not impact the comparison significantly, as their evaluation utilized a stopping criterion of 100 iterations per run over 20 runs, whereas Lobw was evaluated using 100 iterations per run across 10 runs.

In Table 11, the data are presented statistically. Each graph's statistics are displayed as min., avg., std., and worst., representing the minimum, average, standard deviation, and worst values, respectively.

Lobw demonstrates superior performance compared to all state-of-the-art solutions by consistently providing the best solutions. Specifically, Lobw improved upon the best-known solution in 32 out of 42 graph instances and matched it in the remaining ten instances. In larger graphs, the advantages of Lobw over other algorithms are more pronounced. For instance, in instances with 300, 350, and 400 vertices, Lobw significantly outperformed other methods across all instances.

In contrast, Samds achieved the most effective minimum dominating sets for large graphs, while Hga performed better for relatively smaller graphs. Notably, Samds exhibited superior convergence rates, enabling quicker identification of the best solution. Based on these findings, we will focus the comparison solely on Samds against Lobw. Each row in the comparison tables highlights the best result achieved by the top-performing algorithm in bold type.

For the second benchmark set, see Table 12, where the performance of Lobw is compared to that of Samds, the best-known method in this dataset. The benchmark graph instances and data are drawn from the same paper [23]. Lobw and Samds results are given for each graph in terms of the average solution of the best solution found in 10 runs (column Average) and the reaching times of the best solution found over 10 runs (column Optimal Reached).

Both algorithms are able to find the dominating set, so the comparison will be based on how many times each algorithm finds the optimal solution in 10 runs. Furthermore, Lobw is able to find the optimal solution in 10 runs for all graph instances in this benchmark set. Samds can only prove optimality 10 times in 11 out of 21 cases (3 graphs with 800 vertices and 7 graphs with 400 vertices) and can match the results of Lobw in only 11 further cases.

In terms of statistical differences between the methods investigated, only the first benchmark set of random goemetric graphs makes sense, as the second benchmark set results of Samds and Lobw are too evident. Proceed to the statistical section to see if there are any statistical differences between subsets of instances from Table 1.

Table 11
A comparison of the performance of various algorithms on the first benchmark sets

Instance		Best		Mean		Std.		Worst
Network	Range	Samds	Lobw	Samds	Lobw	Samds	Lobw	Lobw
$N1_{3000}^{400}$	210	75	**67**	80.15	67.4	3.10	0.52	68
	220	73	**62**	79.25	64.1	3.18	1.37	66
	230	71	**57**	74.10	59.5	2.22	1.27	61
	240	63	**54**	68.80	55.2	2.98	0.63	56
$N2_{2500}^{350}$	200	61	**54**	66.35	54.9	2.13	0.32	55
	210	58	**47**	61.85	48.5	2.18	0.71	49
	220	49	**43**	55.05	44.2	2.31	0.63	45
	230	48	**42**	54.05	42.7	2.42	0.48	43
$N3_{2000}^{300}$	180	47	**42**	52.35	43.1	2.41	0.74	44
	190	46	**39**	50.20	39.9	2.24	0.57	41
	200	40	**35**	45.25	35.6	2.55	0.7	37
	210	39	**34**	43.60	34.2	1.70	0.42	35
	220	36	**31**	40.65	31.5	1.90	0.53	32
$N4_{1500}^{250}$	130	51	**47**	56.05	47	2.68	0	**47**
	140	46	**40**	48.65	40.2	1.35	0.42	41
	150	41	**37**	44.75	37.3	1.71	0.48	38
	160	37	**33**	40.90	33.9	1.92	0.32	34
$N5_{1000}^{200}$	100	38	**35**	41.35	35.1	1.87	0.32	36
	110	33	**30**	37.20	30.6	2.44	0.7	32
	120	26	**23**	30.10	23.7	1.45	0.67	25
	130	25	**23**	28.15	23.4	1.42	0.52	24
	140	22	**20**	25.70	20.8	1.84	0.42	21
	150	20	**17**	23.55	17.4	1.43	0.52	18
	160	19	**17**	21.30	17	1.30	0	**17**
$N6_{700}^{200}$	70	35	**32**	39.95	32	2.09	0	**32**
	80	29	**26**	33.50	26.3	1.73	0.48	27
	90	25	**22**	29.25	22.7	1.94	0.48	23
	100	20	**18**	24.20	18.4	1.70	0.52	19
	110	18	**16**	21.40	16	1.93	0	**16**
	120	15	**14**	19.55	14	1.43	0	**14**
$N7_{600}^{100}$	80	**18**	**18**	20.55	18	1.23	0	18
	90	**15**	**15**	17.65	15	1.73	0	15
	10	**13**	**13**	14.95	13	1.05	0	13
	110	**11**	**11**	12.85	11	1.23	0	11

	120	**9**	**9**	12	9	1.49	0	9
	60	**15**	**15**	16.35	15	0.67	0	15
	70	**12**	**12**	14.40	12.2	1.14	0.42	13
	80	10	**9**	12	9	1.03	0	**9**
$N8_{400}^{80}$	90	**8**	**8**	9.60	8	1.27	0	8
	100	**7**	**7**	9.10	7.3	0.97	0.48	8
	110	**6**	**6**	7.55	6	0.69	0	6
	120	**5**	**5**	7	5.3	1.08	0.48	6

Table 12
Numerical results of Lobw and Samds on the second benchmark sets

Network	Optimal	Averge		Optimal Reached	
		Samds	Lobw	Samds	Lobw
$N_{0.1,d}^{400}$	8	8	8	10	10
	11	11.1	11	9	10
	14	14.2	14	9	10
	18	18	18	10	10
	23	23	23	10	10
$N_{0.3,d}^{400}$	3	3.8	3	8	10
	5	5.2	5	8	10
	8	9	8	8	10
	11	11	11	10	10
	14	14	14	10	10
$N_{0.5,d}^{400}$	3	3.1	3	9	10
	5	5.3	5	9	10
	8	8	8	10	10
	11	11	11	10	10
$N_{0.1,d}^{800}$	11	11.3	11	7	10
	14	14	14	10	10
	22	22	22	10	10
$N_{0.3,d}^{800}$	3	7.3	3	6	10
	5	5	5	10	10
$N_{0.5,d}^{800}$	3	3.4	3	8	10
	6	6	6	10	10

5.5 *Statistical Analysis*

To identify statistical differences (if any) pertaining to Lobw and Samds from Table 11 and Bwoa from Tables 3, 4, 5, 6, 7, 8, 9, 10. We rate algorithms using Critical Difference (CD) diagrams, first presented in [28]. The package used can be downloaded from bit.ly/3KYCDoFbit.ly/3KYCDoF. They are a useful resource for assessing different algorithms. Using the non-parametric

Friedman test in the first stage, which examines the algorithms separately for each data set to establish the average ranks of algorithms and determines whether there are any notable differences, before drawing any diagrams. In the second stage, we undertake a post-hoc study after the Friedman test rejects the null hypothesis that every algorithm performs identically. Where each pair of algorithms is compared using a Wilcoxon signed-rank test to see if there is a significant difference [29]. Given the comparison of multiple hypotheses, Wilcoxon's test is adjusted using Holm's method. In the diagram plots, algorithms under evaluation are ranked along the horizontal axis. The algorithm demonstrating the best performance receives a rank closer to 1, followed by subsequent ranks for others. Strong horizontal bars connect algorithms that, according to the Holm-adjusted Wilcoxon's test, are statistically indistinguishable and considered equivalent.

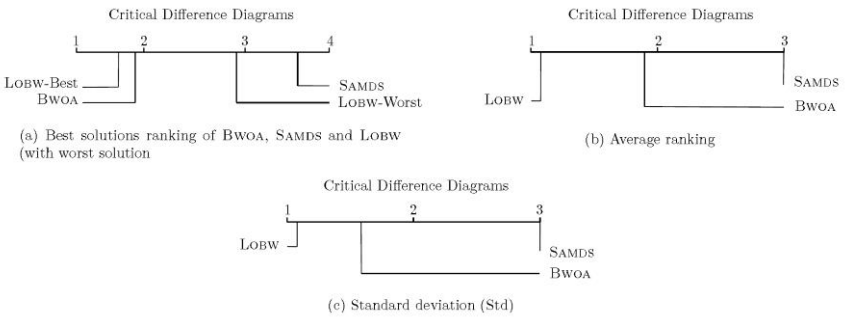

Figure 3
Best solutions ranking of Bwoa, Samds, and Lobw

In terms of the best solution, the mean rankings of the critical difference plots in Figure 3a demonstrate that Lobw outperforms the other algorithms statistically. The statistical closeness of Bwoa's performance to our technique while being ranked second confirms the efficacy of the WOA algorithm in solving the minimum dominating set problem. The Samds algorithm ultimately proves to be the weakest in the investigation; even the poorest solutions found by Lobw in 10 runs yield superior outcomes to the Samds algorithm's best.

Figure 3b displays the critical difference diagram of Lobw, Bwoa, and Samds in terms of the average solution of the best solution found in 10 runs. Lobw clearly outperforms Samds and Bwoa, which is obvious and demonstrates the effectiveness of the local search in improving the exploitation process. As a result, we can say that, in terms of stability, Lobw performs better than cutting-edge methods. Additionally, Figure 3c, which displays a comparison diagram of the standard deviations obtained using the three approaches, supports the stability of the Lobw algorithm. We can see this in Table 11, where the Lobw std. column produced 10 zeros out of 42 cases, and the remaining cases are too close to zero. Yet, Bwoa performs better than Samds when standard deviation values are high, which demonstrates that Samds has a propensity to converge to a local optimum.

6. Conclusion

Currently, it is common practice to solve optimization problems by simulating the cognitive behavior of animals and insects. One of the most extensively studied metaheuristic algorithms in the field of swarm intelligence is the Whale Optimization Algorithm (WOA), which is inspired by the social hunting behavior of humpback whales. In this study, an enhanced version of WOA known as Lobw has been developed specifically to address the minimum dominating set problem, a fundamental problem in graph theory. Lobw balances exploration and exploitation capabilities by incorporating a local search mechanism, which helps WOA in avoiding premature convergence. The proposed method was evaluated using two benchmark datasets from the literature. Our results demonstrate that Lobw is a reliable and efficient approach for solving the minimum dominating set problem compared to other state-of-the-art methods. Furthermore, this study investigates the performance of Lobw using these benchmark datasets and examines its practical applicability. Our future research directions include extending the whale optimization approach to tackle other combinatorial optimization problems.

References

[1] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, Fundamentals of Domination in Graphs. Boca Raton, FL, USA: CRC Press (2013).

[2] F. Dai and J. Wu, "An extended localized algorithm for connected dominating set formation in ad hoc wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 10, pp. 908–920 (Oct. 2004).

[3] J. Blum, M. Ding, A. Thaeler, and X. Cheng, "Connected dominating set in sensor networks and MANETs," in Handbook of Combinatorial Optimization, D.-Z. Du and P. Pardalos, Eds. Boston, MA, USA: Springer, pp. 329–369 (2004).

[4] C. Shen and T. Li, "Multi-document summarization via the minimum dominating set," in Proc. 23rd Int. Conf. Computational Linguistics (COLING), Beijing, China, pp. 984–992 (2010).

[5] M. M. Daliri Khomami, A. Rezvanian, N. Bagherpour, and M. R. Meybodi, "Minimum positive influence dominating set and its application in influence maximization: A learning automata approach," Applied Intelligence, vol. 48, no. 3, pp. 570–593 (Mar. 2018).

[6] D. Zhao, G. Xiao, Z. Wang, L. Wang, and L. Xu, "Minimum dominating set of multiplex networks: Definition, application, and identification," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 12, pp. 7823–7837 (Dec. 2021).

[7] S. Wuchty, "Controllability in protein interaction networks," Proceedings of the National Academy of Sciences, vol. 111, no. 19, pp. 7156–7160 (May 2014).

[8] E. H. Houssein, A. Hamad, A. E. Hassanien, and A. A. Fahmy, "Epileptic detection based on whale optimization enhanced support vector machine," *Journal of Information and Optimization Sciences*, vol. 40, no. 3, pp. 699–723 (2019), doi: 10.1080/02522667.2018.1453671.

[9] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, vol. 174. San Francisco, CA, USA: W. H. Freeman (1979).

[10] Y. Iwata, "A faster algorithm for dominating set analyzed by the potential method," in Proc. Int. Symp. Parameterized and Exact Computation, Zurich, Switzerland: Springer, pp. 41–54 (2011).

[11] E.-G. Talbi, Metaheuristics: From Design to Implementation. Hoboken, NJ, USA: John Wiley & Sons (2009).

[12] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science,* vol. 220, no. 4598, pp. 671–680 (May 1983).

[13] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," Simulation, vol. 76, no. 2, pp. 60–68 (Feb. 2001).

[14] A. K. Parekh, "Analysis of a greedy heuristic for finding small dominating sets in graphs," Information Processing Letters, vol. 39, no. 5, pp. 237–240 (Mar. 1991).

[15] P.-J. Wan, K. M. Alzoubi, and O. Frieder, "A simple heuristic for minimum connected dominating set in graphs," *International Journal of Foundations of Computer Science*, vol. 14, no. 2, pp. 323–333 (Apr. 2003).

[16] Y. Alkhalifah and R. L. Wainwright, "A genetic algorithm applied to graph problems involving subsets of vertices," in Proc. 2004 Congress on Evolutionary Computation (CEC'04), vol. 1, Portland, OR, USA: IEEE, pp. 303–308 (2004).

[17] R. Misra and C. Mandal, "Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 3, pp. 292–302 (Mar. 2009).

[18] L. A. Sanchis, "Experimental analysis of heuristic algorithms for the dominating set problem," *Algorithmica*, vol. 33, no. 1, pp. 3–18 (Jan. 2002).

[19] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," *IEEE Computational Intelligence Magazine,* vol. 1, no. 4, pp. 28–39 (Nov. 2006).

[20] C. K. Ho, Y. P. Singh, and H. T. Ewe, "An enhanced ant colony optimization metaheuristic for the minimum dominating set problem," *Applied Artificial Intelligence,* vol. 20, no. 10, pp. 881–903 (Oct. 2006).

[21] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Cambridge, MA, USA: MIT Press (1992).

[22] A.-R. Hedar and R. Ismail, "Hybrid genetic algorithm for minimum dominating set problem," in Proc. Int. Conf. Computational Science and Its Applications (ICCSA), Fukuoka, Japan: Springer, pp. 457–467 (2010).

[23] A.-R. Hedar and R. Ismail, "Simulated annealing with stochastic local search for minimum dominating set problem," *International Journal of Machine Learning and Cybernetics,* vol. 3, no. 2, pp. 97–109 (Apr. 2012).

[24] S. A. Abed, H. M. Rais, J. Watada, and N. R. Sabar, "A hybrid local search algorithm for minimum dominating set problems," *Engineering Applications of Artificial Intelligence,* vol. 114, p. 105053 (Aug. 2022).

[25] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67 (May 2016).

[26] B. Chen, W. Zeng, Y. Lin, and D. Zhang, "A new local search-based multiobjective optimization algorithm," *IEEE Transactions on Evolutionary Computation,* vol. 19, no. 1, pp. 50–73 (Feb. 2015).

[27] C. Bettstetter, "On the minimum node degree and connectivity of a wireless multihop network," in Proc. 3rd ACM Int. Symp. Mobile Ad Hoc Networking & Computing (MobiHoc), Lausanne, Switzerland, pp. 80–91 (2002).

[28] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30 (Jan. 2006).

[29] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks?," *Journal of Machine Learning Research,* vol. 17, no. 1, pp. 152–161 (Jan. 2016).

Received May, 2023