Laboratoire d'Informatique Fondamentale, de Recherche Opérationnelle, de Combinatoire et d'Économétrie

Availaible on line at http://www.liforce.usthb.dz

bulletin-liforce@usthb.dz

Bulletin du Laboratoire 01 (2023) 01 - 20

Minimum Dominating Set using an Enhanced Binary Whale Algorithm

Belkacem ZOUILEKH & Sadek BOUROUBI

USTHB, Faculty of Mathematics,
P.B. 32 El-Alia, 16111, Bab Ezzouar, Algiers, Algeria.

bzouilekh@usthb.dz & sbouroubi@usthb.dz

Abstract : The Whale Optimization Algorithm (WOA) is a novel metaheuristic algorithm inspired by the social hunting behavior of humpback whales. This paper describes a binary version of the WOA applied to the Minimum Dominating Set (MDS) problem, which is a challenging combinatorial optimization problem that is known to be NP-hard. However, WOA has a drawback of premature convergence, which may result in the algorithm becoming trapped in local optima and failing to reach the global optimum. To overcome this limitation, the paper proposes an enhanced version of WOA called the Local search Optimization Binary Whale (Lobw) that incorporates a Local Search technique to improve exploitation ability and prevent the algorithm from being trapped in local optima. The Lobw algorithm is evaluated on several benchmark datasets, and the results demonstrate its promising performance in terms of solution quality and stability when 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.

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

1 Introduction

One the most crucial problems in graph theory is known as the Minimum Dominating Set. It is also an NP-hard problem that cannot be solved in polynomial time [1]. In an undirected graph, $G = (V, E)$, where V represents the set of vertices and E represents the set of edges, a subset S of V is considered a dominating set of G if each vertex $v \in V$ is either an element of S or is adjacent to an element of S. This subset is referred to as a dominating set of G, and it is said that S dominates G or that G is dominated by S. The minimum cardinality of a dominating set in G is represented by $\gamma(G)$, which is called domination number. If S dominates G such that $|S| = \gamma(G)$, then S is called the Minimum Dominating Set, or MDS for short. So, the problem of dominating set involves finding the value of $\gamma(G)$ and identifying a dominating set that has the minimum number of vertices. The mathematical model can be described as follows:

$$\min \; f(x) = \sum_{v_i \in V} x_i \tag{1.1}$$

$$\text{subject to:} \begin{cases} \sum\limits_{v_j \in N[v_i]} x_j \geq 1, \quad \forall v_i \in V \\ x_i \in \{0, 1\} \end{cases} \tag{1.2}$$

where x_i is a binary indicator that takes 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 ensure that every vertex is either in the minimum dominating set or has a neighbor in the set.

Minimum dominating sets, as well as their variations such as minimum connected dominating set and minimum weighted dominating set, have diverse applications in a wide range of fields. These include ad hoc wireless networks [2], sensor networks, and MANETs [3] for routing purposes, as well as determining main subject sentences for document summary through sentence network design [4]. Additionally, MDS can be used to model and study the positive impact in social networks [5], as well as control the spread of epidemics and diagnose and control epidemic spreadings in various areas of human society, including virus spread in computer systems, misinformation, and content diffusion through social media [6]. MDS can also be found in biological aspects such as interacting protein networks and biological network analysis [7].

This paper applies the Whale Optimization Algorithm (WOA) to solve the MDS problem. The aim is to leverage the advantages of WOA, such as its low number of parameters and high convergence rate. The proposed algorithm's performance is evaluated on various data sets and compared with some state-of-the-art techniques.

The rest of the paper is structured as follows: Section 2 presents related work, including exact algorithms and heuristics, that have been applied to solving the MDS problem. Section 3 describes the WOA algorithm. The proposed WOA algorithm is discussed in section 4. Section 5 presents the experimental results, and finally, section 6 concludes the paper and discusses some future research directions.

2 Related works

Exact (exponential-time) approaches have been used extensively to study the MDS problem, which is NP-hard [1, 8]. In addressing the MDS problem in a graph with n vertices, numerous authors have recently suggested exact algorithms that surpass the naive $\mathcal{O}(2^n)$ which merely searches through all potential case subsets of V. Based on our current information, the best exact algorithm for the MDS problem performs in $\mathcal{O}(1.4864^n)$ time and polynomial space. This algorithm was constructed through the measure and conquer approach by Y. Iwata [9].

Unfortunately, the exact procedures that execute at an exponential scale are only viable for small networks, greatly restricting their usefulness. As a result, scientific researchers have focused mostly on stochastic computational heuristics and, more recently, metaheuristics.

This section provides a quick overview of heuristic-based minimum dominating set techniques. These methods can be classified in a variety of ways. They can be either nature-inspired or not [10]. Many metaheuristics are inspired by natural occurrences and are classified into two types: (I) evolutionary algorithms and (II) swarm intelligence. In nature, evolutionary algorithms employ the theory of biological evolution (e.g., crossover and mutation). The communal and foraging behavior of ants, fireflies, whales, grasshoppers, and many other organisms and species in nature inspired swarm intelligence systems. Metaheuristics inspired by non-natural events like Simulated Annealing (SA) [11] from physics and Harmony Search (HS) [12] from musics.

In this regard, various heuristic approaches have been developed in the state of the art to deal with the MDS problem, such as [13, 14, 15, 16]. After that L. A. Sanchis [17] conducted an experimental study on various heuristic techniques. He has exhaustively researched numerous greedy methods to the MDS problem.

Ant Colony Optimization (ACO) [18] is one of the most well-known swarm intelligence algorithms. C. K. Ho et al., in [19], introduced ACO-TS, an upgraded Ant Colony Optimization metaheuristic that incorporates a technique for promoting the construction of various solutions based on a genetic algorithm notion termed tournament selection.

Genetic Algorithm (GA) is a traditional evolutionary algorithm [20]. The GA was initially used to address the MDS issue in 2010 [21]. Two years later, the authors of [22] proposed Simulated Annealing, one of physics most well-known single-solution based metaheuristics inspired by non-natural phenomena. The SA metaheuristic addresses the MDS problem by using a Stochastic Local Search (SLS) algorithm to reinforce a solution by searching for a stronger one in its surroundings.

Later, in 2022, the authors [23] proposed a two-step hybrid local search algorithm that hybridized local search techniques, local search as an exploratory technique in the initial step, which focuses on developing promising solutions in various regions of the solution space. They suggest another local search algorithm as an exploitation strategy for the second step. It navigates viable neighborhood locations in order to improve 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 [24] 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:

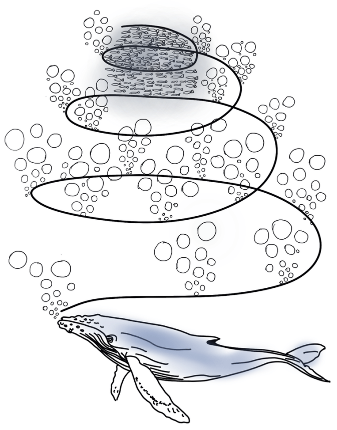


Figure 1: The bubble-net feeding behavior of a humpback whale.

3.1 Encircling Prey

Humpback whales are filter feeders and feed mainly on small fish and krill. They use their position to find, surround and chase prey. The WOA algorithm makes the assumption that the current best candidate solution X^* is the target prey or is very close to the optimum because the location of the optimal design in the search space is not previously known. The position of the prey is therefore determined by the best whale $\overrightarrow{X^*}$ that has been discovered up to that iteration t. Such a mechanism is described by the ensuing equations:

$$\overrightarrow{X}(t+1) = \overrightarrow{X^*}(t) - \overrightarrow{A}.\overrightarrow{D}, \tag{3.1}$$

$$\overrightarrow{D} = \left| \overrightarrow{C}.\overrightarrow{X^*}(t) - \overrightarrow{X}(t) \right|, \tag{3.2}$$

where $\overrightarrow{D}$ is the computed modified distance between the whale $\overrightarrow{X}(t)$ and the prey; $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors constructed mathematically using the following equations:

$$\overrightarrow{A} = 2a\overrightarrow{r} - a \tag{3.3}$$

$$\overrightarrow{C} = 2\overrightarrow{r} \tag{3.4}$$

Throughout the length of the iterations, a is reduced linearly from 2 to 0. In equations (3.3) and (3.4), $\overrightarrow{r}$ 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 (3.3) 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.

$$\overrightarrow{X}(t+1) = \overrightarrow{D'}\exp(bl)\cos(2\pi l) + \overrightarrow{X^*}(t) \tag{3.5}$$

$$\overrightarrow{D'} = |\overrightarrow{X^*}(t) - \overrightarrow{X}(t)| \tag{3.6}$$

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:

$$\overrightarrow{X}(t+1) = \begin{cases} \text{Shrinking encircling (Equation (3.3))} & \text{if } p < 0.5 \\ \text{Spiral motion (Equation (3.5))} & \text{if } p \geq 0.5 \end{cases} \tag{3.7}$$

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:

$$\overrightarrow{X}(t+1) = \overrightarrow{X_{rand}} - \overrightarrow{A} \cdot \overrightarrow{D}, \tag{3.8}$$

$$\overrightarrow{D} = |\overrightarrow{C} \cdot \overrightarrow{X_{rand}} - \overrightarrow{X}(t)|. \tag{3.9}$$

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 [25]. Two whales are chosen at random $(\overrightarrow{x_{r1}}, \overrightarrow{x_{r2}})$, and two other whales are created using:

$$\begin{aligned} \overrightarrow{w_i^1}(t+1) &= \overrightarrow{w_i}(t) + (\overrightarrow{x_{r1}} - \overrightarrow{x_{r2}}), \\ \overrightarrow{w_i^2}(t+1) &= \overrightarrow{w_i}(t) - (\overrightarrow{x_{r1}} - \overrightarrow{x_{r2}}), \end{aligned} \tag{4.1}$$

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 $X_{wi}$ using Equation (1.1)
 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 (3.3) and (3.4) resp.
12:         $p = random(0, 1)$
13:         if $p < 0.5$ then
14:             if $A \geq 1$ then
15:                 // Exploration through shrinking encircling //
16:                 Pick a whale at random from the population
17:                 Update position based on Equation (3.8)
18:             else
19:                 // Exploitation through shrinking encircling //
20:                 Update position based on Equation (3.1)
21:         else
22:             // Spiral motion //
23:             Update position based on Equation (3.5)
24:         if $w_i$ doesn't change then
25:             Choose two whales $candidate_1$ and $candidate_2$ at random from the
26:             population $N$ other than whale $w_i$
27:             Perform the Local Search on whale $w_i$
28:         else
29:             Continue
30:     Rank whales from best to worst and find the fittest one
31:     Update the global optimal solution $X^*$
32: 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
1: Generate two neighbors candidate solutions $candidate_1$ and $candidate_2$ using Equation (4.1)
2: $f_1$ = fitness($candidate_1$)
3: $f_2$ = fitness($candidate_2$)
4: Sort the two generated whales and take the fittest one as $candidate$ and its fitness $f$
5: if $f < fitness(w_i)$ then $\overrightarrow{w_i} = candidate$
6: else
7:     return $\overrightarrow{w_i}$
8: 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. Also, all testing was done on a machine 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 [22] are 42 arbitrary geometric networks with up to 400 nodes. For the purpose of building the networks, the authors utilized the generating instructions from [19] and [26]. Every network is made by randomly placing n nodes (see column No. of nodes in Table 1) in an $M \times M$ area according to a uniform distribution (see column Area in Table 1). Based on the range parameter shown in column Range in Table 1, a number of graph instances were created for each network. They made sure that the graph constituted a connected graph by adhering to the directions in [26]. In addition, the column Nb of instances is 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

Graphs from [22] consisting of 21 graphs with 400 and 800 vertices each. The minimum dominating set in this benchmark set is determined. The method described in [19] was used by the authors to create networks with known dominance numbers and given densities denoted $\mathbf{G^n_{d,p}}$, where n is the number of nodes, d is the dominance number, and p is the probability for edge generation. The detailed technique is Initially described in [17]. Two types of graphs with 400 and 800 vertices, various densities (0.1, 0.3, and 0.5), were employed in this benchmark (see Table 2).

Table 2: Known dominance number benchmark

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

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

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

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

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

Hits $=$ *count*(*best*) 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 created smaller dominant sets for 7 graph examples (see 0 in the hits column of Bwoa); all of these improvements were obtained for networks with at least 300 nodes, with the majority of them for network N1. For 35 instances, both versions found best solutions of equal 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 $N1_{3000}^{400}$

	Instance	BWOA				LOBW			
Network	Range	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 $N2_{2500}^{350}$

	Instance	BWOA				LOBW			
Network	Range	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 $N3_{2000}^{300}$

	Instance	BWOA				LOBW			
Network	Range	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 $N4_{1500}^{250}$

	Instance	BWOA				LOBW			
Network	Range	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 $N5^{200}_{1000}$

	Instance	BWOA				LOBW			
Network	Range	Best	Avg	Std	Hits	Best	Avg	Std	Hits
	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
$N5^{200}_{1000}$	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 $N6^{200}_{700}$

	Instance	BWOA				LOBW			
Network	Range	Best	Avg	Std	Hits	Best	Avg	Std	Hits
	70	32	32	0	10	32	32	0	10
	80	26	26.6	0.70	5	26	26.3	0.48	7
$N6^{200}_{700}$	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 $N7^{100}_{600}$

	Instance	BWOA				LOBW			
Network	Range	Best	Avg	Std	Hits	Best	Avg	Std	Hits
	80	18	18.2	0.42	8	18	18	0	10
	90	15	15	0	10	15	15	0	10
$N7^{100}_{600}$	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 $N8^{80}_{400}$

	Instance	BWOA				LOBW			
Network	Range	Best	Avg	Std	Hits	Best	Avg	Std	Hits
	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
$N8^{80}_{400}$	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

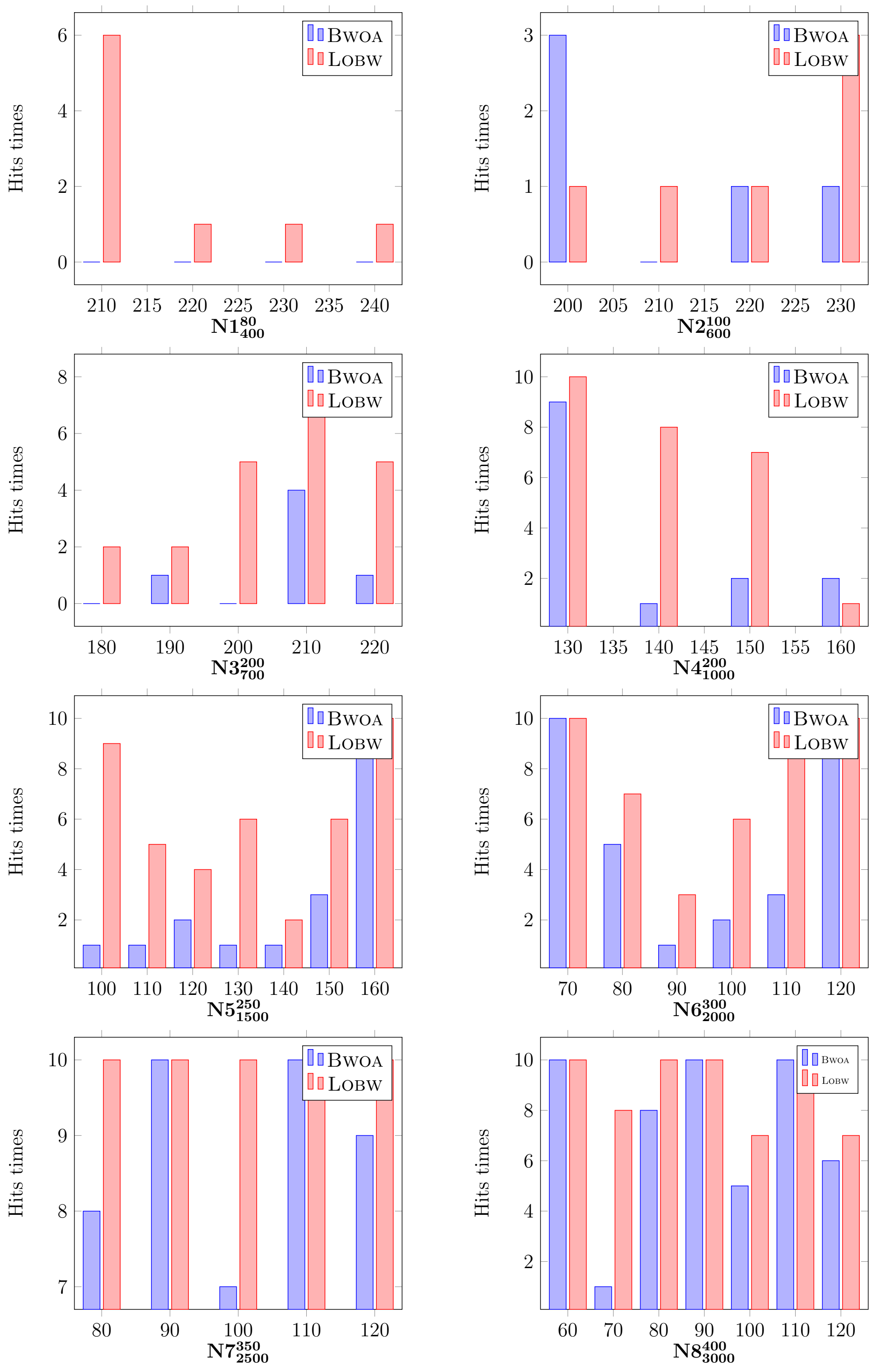


Figure 2: Hits analysis on various graphs for Lobw and Bwoa.

5.4 Numerical results to the relevant works

In this section the results of Lobw are evaluated with the state-of-the-art algorithms in the literature. The datasets used in this comparison were taken from [22]. Many works have constructed random geometric graphs 5.1.1 based on Bettstetter's work [26]. The comparison with these works, however, will be inaccurate because the data may have slight alterations due to the random distribution procedure used to generate it. To fix this problem, all of the methods in the comparison used the same dataset. The Hga [21] and Samds [22] values were obtained from [22]. However, this does not affect the comparison because their stopping criterion was 100 iterations each run for 20 runs, whereas Lobw evaluated using 100 iterations per run for the 10 runs.

In Table 11, the data are statistically reported. Each graph's statistics are shown in the form of min., avg., std., and worst., which indicate the minimum, average, standard deviation, and worst values, respectively.

Lobw outperforms all state-of-the-art solutions in terms of offering the best solution. Additionally, Lobw can improve on the best-known solution in 32 of 42 graph instances and match the best-known solution in the remaining ten. In larger graphs, the differences between Lobw and the other algorithms become more pronounced. For example, in instances with 300, 350, and 400 vertices, Lobw produced much better solutions in all graph instances. In contrast, for large graphs, Samds generated the most effective minimum dominating sets, while Hga performed better for relatively smaller graphs. However, Samds outperformed Hga in terms of convergence rate, enabling faster identification of the best solution. Based on these results, we will consider only Samds for the comparison with Lobw. Notice that each table row in the comparison highlights the best output of 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 [22]. 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.

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 [27]. The package used can be downloaded from bit.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 [28]. As we are comparing many hypotheses, we must modify the Wilcoxon's test using Holm's method. On the horizontal axis of these diagram plots, each algorithm under consideration is ranked. The algorithm with the best performance is given a rank that is close to 1, followed by the second with a rank that is higher, and so on. Strong horizontal bars link the related algorithms, which we cannot distinguish from the Holm-adjusted Wilcoxon's test, which is viewed as statistically equal.

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.

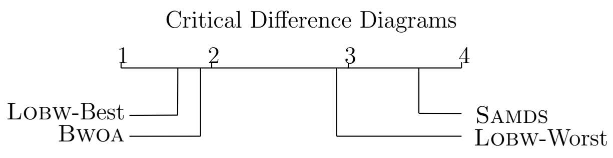
Figure 3b displays the critical difference diagram of LOBW, BWOA, and SAMDS in terms of the average solution of the best solution found in 10 runs. LOBW clearly outperforms SAMDS and BWOA, which is obvious and demonstrates the effectiveness of the local search in improving the exploitation process. As 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.

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

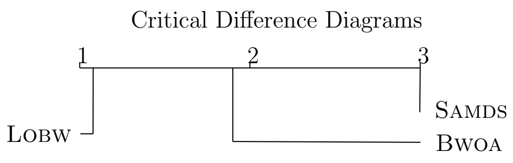
	Instance	Best		Mean		Std.		Worst
Network	Range	Samds	Lobw	Samds	Lobw	Samds	Lobw	Lobw
$N1^{400}_{3000}$	210	75	**67**	80.15	67.4	3.10	0.52	68
	220	73	**62**	79.25	64.1	3.18	1.37	66
	230	71	**57**	74.10	59.5	2.22	1.27	61
	240	63	**54**	68.80	55.2	2.98	0.63	56
$N2^{350}_{2500}$	200	61	**54**	66.35	54.9	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
$N3^{300}_{2000}$	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
$N4^{250}_{1500}$	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
$N5^{200}_{1000}$	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**
$N6^{200}_{700}$	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**
$N7^{100}_{600}$	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
$N8^{80}_{400}$	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**
	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

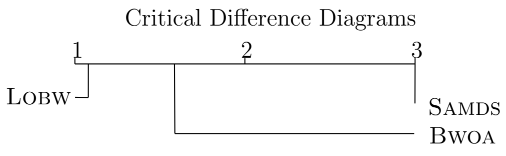
		Averge		Optimal Reached	
Network	Optimal	SAMDS	LOBW	SAMDS	LOBW
	8	8	8	10	10
	11	11.1	11	9	10
$N^{400}_{0.1,d}$	14	14.2	14	9	10
	18	18	18	10	10
	23	23	23	10	10
	3	3.8	3	8	10
	5	5.2	5	8	10
$N^{400}_{0.3,d}$	8	9	8	8	10
	11	11	11	10	10
	14	14	14	10	10
	3	3.1	3	9	10
$N^{400}_{0.5,d}$	5	5.3	5	9	10
	8	8	8	10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



(a) Best solutions ranking of Bwoa, Samds and Lobw (with worst solution)



(b) Average ranking



(c) Standard deviation (Std)

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

6 Conclusion

Currently, it's fairly usual to solve optimization problems by simulating the cognitive behavior of animals and insects. The most thoroughly examined metaheuristic algorithm in the field of swarm intelligence, a new, straightforward, and reliable optimization strategy, is the Whale Optimization Algorithm, which is based on the social hunting behavior of humpback whales. In the current work, an improved whale optimization technique known as Lobw has been designed for tackling the minimum dominating set, one of the key graph theory problems. The suggested algorithm's exploration and exploitation capabilities are balanced by the employment of a local search mechanism by Lobw, which also aids WOA in preventing premature convergence. Additionally, the suggested method was tested using two benchmark datasets from the literature. As well, our results show that the proposed Lobw method is a reliable and efficient way to solve the problem of the minimum dominating set when compared to other state-of-the-art approaches. In this study, we examine our suggested algorithm using two benchmark test datasets. We will look into how well Lobw performs in practical applications. The whale optimization approach will also be extended for use with other combinatorial optimization issues as part of our other future work paths.

References

[1] T. W. Haynes, S. Hedetniemi, P. Slater, Fundamentals of domination in graphs, CRC press, 2013.

[2] F. Dai, J. Wu, An extended localized algorithm for connected dominating set formation in ad hoc wireless networks, IEEE transactions on parallel and distributed systems 15 (10) (2004) 908-920.

[3] J. Blum, M. Ding, A. Thaeler, X. Cheng, Connected dominating set in sensor networks and manets, in: Handbook of combinatorial optimization, Springer, 2004, pp. 329-369.

[4] C. Shen, T. Li, Multi-document summarization via the minimum dominating set, in: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), 2010, pp. 984-992.

[5] M. M. Daliri Khomami, A. Rezvanian, N. Bagherpour, M. R. Meybodi, Minimum positive influence dominating set and its application in influence maximization: a learning automata approach, Applied Intelligence 48 (3) (2018) 570-593.

[6] D. Zhao, G. Xiao, Z. Wang, L. Wang, L. Xu, Minimum dominating set of multiplex networks: definition, application, and identification, IEEE Transactions on Systems, Man, and Cybernetics: Systems 51 (12) (2020) 7823-7837.

[7] S. Wuchty, Controllability in protein interaction networks, Proceedings of the National Academy of Sciences 111 (19) (2014) 7156-7160.

[8] M. R. Garey, D. S. Johnson, Computers and intractability, Vol. 174, freeman San Francisco, 1979.

[9] Y. Iwata, A faster algorithm for dominating set analyzed by the potential method, in: International Symposium on Parameterized and Exact Computation, Springer, 2011, pp. 41-54.

[10] E.-G. Talbi, Metaheuristics: from design to implementation, John Wiley & Sons, 2009.

[11] S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi, Optimization by simulated annealing, science 220 (4598) (1983) 671-680.

[12] Z. W. Geem, J. H. Kim, G. V. Loganathan, A new heuristic optimization algorithm: harmony search, simulation 76 (2) (2001) 60-68.

[13] A. K. Parekh, Analysis of a greedy heuristic for finding small dominating sets in graphs, Information processing letters 39 (5) (1991) 237-240.

[14] P.-J. Wan, K. M. Alzoubi, O. Frieder, A simple heuristic for minimum connected dominating set in graphs, International Journal of Foundations of Computer Science 14 (02) (2003) 323-333.

[15] Y. Alkhalifah, R. L. Wainwright, A genetic algorithm applied to graph problems involving subsets of vertices, in: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), Vol. 1, IEEE, 2004, pp. 303-308.

[16] R. Misra, C. Mandal, Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks, IEEE Transactions on parallel and distributed systems 21 (3) (2009) 292-302.

[17] L. A. Sanchis, Experimental analysis of heuristic algorithms for the dominating set problem, Algorithmica 33 (1) (2002) 3-18.

[18] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE computational intelligence magazine 1 (4) (2006) 28-39.

[19] C. K. Ho, Y. P. Singh, H. T. Ewe, An enhanced ant colony optimization metaheuristic for the minimum dominating set problem, Applied Artificial Intelligence 20 (10) (2006) 881-903.

[20] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT press, 1992.

[21] A.-R. Hedar, R. Ismail, Hybrid genetic algorithm for minimum dominating set problem, in: International conference on computational science and its applications, Springer, 2010, pp. 457-467.

[22] A.-R. Hedar, R. Ismail, Simulated annealing with stochastic local search for minimum dominating set problem, International Journal of Machine Learning and Cybernetics, Springer, 3 (2) (2012) 97-109.

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

[24] S. Mirjalili, A. Lewis, The whale optimization algorithm, Advances in engineering software 95 (2016) 51-67.

[25] B. Chen, W. Zeng, Y. Lin, D. Zhang, A new local search-based multiobjective optimization algorithm, IEEE transactions on evolutionary computation 19 (1) (2014) 50-73.

[26] C. Bettstetter, On the minimum node degree and connectivity of a wireless multihop network, in: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, 2002, pp. 80-91.

[27] J. Dem¡sar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine learning research 7 (1) (2006) 1-30.

[28] A. Benavoli, G. Corani, F. Mangili, Should we really use post-hoc tests based on mean-ranks?, The Journal of Machine Learning Research 17 (1) (2016) 152-161.