Boolean Binary Grey Wolf Optimizer

Rodrigo Cesar Lira
Computer Engineering Department
University of Pernambuco
Recife, Brazil
rcls@ecomp.poli.br

Mariana Macedo
Center for Collective Learning
University of Toulouse
Occitania, France
mmacedo@biocomplexlab.org

Hugo Valadares Siqueira
Department of Electrical Engineering
Federal University of Technology
Paraná, Brazil
hugosiqueira@utfpr.edu.br

Carmelo Bastos-Filho
Computer Engineering Department
University of Pernambuco
Recife, Brazil
carmelofilho@ieee.org

***Abstract*—Several binary swarm algorithms use the identical continuous proposal, adding a transfer function to mapping from continuous to binary space. It has been shown that binary operators are more appropriate and efficient for binary optimisation. Based on it, we proposed the Boolean Binary Grey Wolf Optimizer, a new version of Grey Wolf Optimizer to solve binary optimisation. Our proposal uniquely operates in the binary space using binary vector and boolean operators, precisely AND, OR, XOR, and NOT gates. We used OneMax, ZeroMax and 0-1 Knapsack to compare our proposal with bGWO1, bGWO2 and BPSO. In general, BBGWO outperformed the other swarm-based algorithms in different scenarios. Furthermore, analysing the swarm hamming distance and the unique candidate solutions through optimisation, we found that the proposal can overcome premature convergence, and most of the time, wolves are in different positions.**

***Index Terms*—binary optimisation, grey wolf optimizer, swarm intelligence**

I. Introduction

Many algorithms from the Swarm Intelligence field were proposed or adapted to perform binary optimisation, since it can be helpful for various combinatorial problems [1], [2]. However, there is still room to create or improve the algorithms to be more efficient and less computationally costly [3]–[5]. One common issue over most binary proposals is the direct translation of continuous vectors to binary ones [6], [7]. Even though these versions can converge the swarm to optimal solutions, the computational cost is higher than necessary since it commonly uses continuous operators. The movements in a continuous space do not always translate into changes in the binary space. Thus, developing new completely binary-based algorithms inspired by Swarm Intelligence might help researchers to solve more efficiently problems such as profit unit commitment [8], feature selection [9], [10], task scheduling [11] and mining high utility itemset [12].

One of the algorithms that have been shown to present a good performance in optimisation problems is the Grey wolf optimizer (GWO) [13]. Considering the promising results of boolean operators for binary problems, we propose in this paper a novel Boolean Binary Grey Wolf Optimizer (BBGWO) that simplifies the bGWO [14] by incorporating boolean logic gates instead of applying a transfer mechanism. Our proposal outperforms bGWO versions on three benchmarks: OneMax, ZeroMax, and 0-1 Knapsack problem.

Despite the similarities with Pazhaniraja et. al [12], our proposal does not depend on the type of problem. It is also not affected whether the task regards maximization or minimization. Moreover, other versions of GWO show that operators such as the ones inspired in quantum can improve the performance of bGWO by balancing better exploration-exploitation [15]. However, it adds complexity in comparison to boolean gates. Another recurrent issue from binary optimisation is the presence of similar individuals in the swarm that speeds up the suboptimal convergence [1] which we also consider in our proposal.

The paper is divided into the following sections: Section II explains the original version of GWO, Section III proposes the BBGWO algorithm. Section IV details the experimental setup. In Section V, we discuss the results. Finally, Section VI presents our conclusions and future works.

II. Grey Wolf Optimizer (GWO)

In 2014, Mirjalili *et al.* [13] proposed the Grey Wolf Optimizer (GWO), a continuous swarm-based algorithm inspired by the social hierarchy and group hunting of grey wolves. In the proposal, each wolf in the pack represents a candidate solution, and the three fittest wolves (*leaders*) influence the rest of the swarm to hunt in the best region found so far. The leaders are called alpha (α), beta (β), and delta (δ) in ascending order.

In GWO, the leaders initiate the encircling behaviour modelled in Equations 1 and 2, which attract the swarm near to them. $\mathbf{A}$ and $\mathbf{C}$ are the two vectors defined as $\mathbf{A} = 2\mathbf{a} \cdot \mathbf{r}_1 - \mathbf{a}$, and $\mathbf{C} = 2 \cdot \mathbf{r}_2$, where $\mathbf{r}$ is a random vector [13]. Equation 3 calculates the new wolf position, and the swarm keep converging as they improve their fitness. GWO pseudocode is summarized by Algorithm 1.

Algorithm 1: GWO Pseudocode

Initialise the **a**, **A**, and **C**
Initialise the N wolves randomly
Find the α, β, δ solutions based on fitness
while stop criterion is not reached **do**
  for all $wolf$ **do**
    Compute leaders' influence

$$\mathbf{D}_{l=[\alpha,\beta,\delta]} = |\mathbf{C}_l \cdot \mathbf{x}_l - \mathbf{x}| \quad (1)$$

$$\mathbf{x'}_{l=[\alpha,\beta,\delta]} = \mathbf{x}_l - \mathbf{A}_l \cdot \mathbf{D}_l, \quad (2)$$

    Update wolf position

$$\mathbf{x}(i+1) = \frac{\mathbf{x'}_\alpha + \mathbf{x'}_\beta + \mathbf{x'}_\delta}{3}. \quad (3)$$

  end for
  Update **a**, **A** and **C**
  Evaluate the current position of individual wolves
  Update α, β, δ
end while
Return the best solution

III. Our proposal: Boolean Binary Grey Wolf Optimizer (BBGWO)

We develop a Boolean Binary Grey Wolf Optimizer (BBGWO) that uniquely operates in the binary space. Therefore, our agents are binary vectors, and the interactions among them are also performed in the binary domain using OR, AND, XOR and NOT boolean operations.

Our proposal has a hyperparameter called modification rate (MR), representing a rate of positions modified in each wolf per iteration. We select D dimensions through this rate and update the wolf position based on swarm behaviour. If the swarm fitness changes, the leaders will attract the agents as GWO [13]. Otherwise, we will flip some dimensions, a mechanism similar to the mutation on Genetic Algorithm [16]. The movement equation allows the algorithm to explore and exploit the search space when the fitness is still changing. Furthermore, we added a mechanism ($\bar{x}^t_{i,d}$) to create diversity, considering which GWO tends to converge quickly [17]. Therefore, the movement is calculated using Equation 4:

$$x^{t+1}_{i,d} = \begin{cases} L^t_{\alpha,d} \otimes (L^t_{\beta,d} \otimes L^t_{\delta,d}), \text{ if } sum(\text{fit}^t) \neq sum(\text{fit}^{t-1}), \\ \bar{x}^t_{i,d}, \text{ otherwise.} \end{cases} \quad (4)$$

The symbol $\otimes$ represents the "XOR" operator, $\bar{x}^t_{i,d}$ is the position $x^t_{i,d}$ flipped (0 to 1, or 1 to 0, as a NOT operator), and $sum(fit^t)$ is the sum of fitness for the whole swarm at iteration *t*. For each leader, we can compute the leaders' influence ($L^t_{\{\alpha,\beta,\delta\}}$) using Equation 5:

$$L^t_{l,d} = M(x^t_{l,d}, x^t_{i,d}, r_d), \quad (5)$$

where $r_d = rand\{0,1\}$, l is α, β or δ, and $M(\cdot)$ is the 3-input *Majority Gate* [18] is defined using well-known gates as follows:

$$M(a,b,c) = (a \odot b) \oplus (b \odot c) \oplus (a \odot c). \quad (6)$$

The symbols *a*, *b* and *c* are bits, $\oplus$ is the "OR" operator, and $\odot$ is the "AND" operator.

In Figure 1, we illustrate how the Equation 5 works in BBGWO on the creation of $L^t_{\alpha,d}$. Considering MR = 50%, we randomly select the indexes: 0, 2, and 5 to perform the modification. In (a), we see that $x_{\alpha,0}$ and $x_{9,0}$ have different bits, 1 and 0, respectively. There, r takes an important role, selecting one of them randomly. In (b), the bits from the selected dimension are equal. Therefore, the random bit (r = 1) does not influence the gate output. Lastly, in (c), we see different bits in both candidate solutions, and the random bit (r = 0) tends the output to the bit from x_9. Using the behaviour seen in (a) and (c), we try to mimic the encircling behaviour, and (b) shows a mechanism to maintain the candidate solution in the same leader's region.

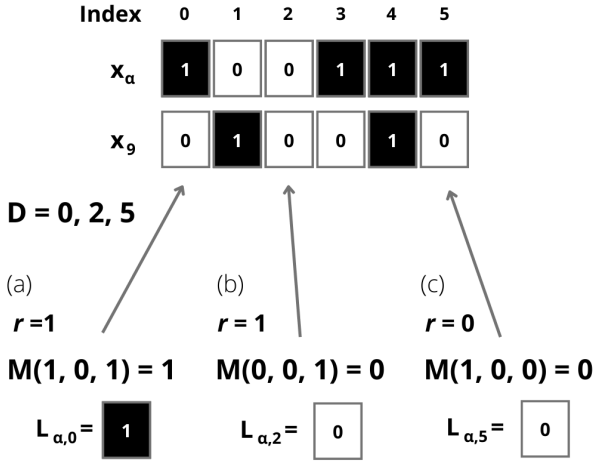

Fig. 1. Using Majority Gate as described in Equation 5. In (a) and (c) we see the gate output when the dimension has different bits in each solution, and (b) shows the results when they are equals.

The proposal pseudocode is described in Algorithm 2. BBGWO starts initializing the wolves (*i.e.*, binary vectors) in the search space and computing the three fittest wolves in the swarm (α, β, and δ). Next, until the stop criterion is reached, the algorithm updates the D selected dimensions of each wolf using Equation 4.

IV. Experimental Setup

We analyse the performance of BBGWO on three problems: OneMax, ZeroMax, and 0-1 Knapsack. OneMax and ZeroMax are two binary problems in that the goal is to maximise the number of ones, and zeros, respectively, in the candidate solution. 0-1 Knapsack is a problem in that the goal is to

Algorithm 2: BBGWO Pseudocode

```
Initialize randomly N wolves
Find α, β, δ leaders based on the fitness
while stop criterion is not reached do
  for all wolf do
    Select D dimensions
    Update each D dimension using Equation 4
  end for
  Update wolves' fitness
  Update α, β, δ leaders
end while
Return the best solution
```

maximise the number of items in a bag (knapsack) considering its weight limitation [19].

Our proposal was compared with bGWO1, bGWO2, and BPSO. bGWO1 and bGWO2 were introduced by Emary et. al [14] to solve binary problems based on the GWO. BPSO is a binary version of the well-known Particle Swarm Optimization [20]. We used BPSO from PySwarms [21], and we developed bGWO1, and bGWO2 using Python™ programming language. We executed all the simulations in an Intel Core i7 computer with 16 GB of RAM running the Ubuntu Linux OS 22.04 64 bits operating system.

We analyse three scenarios using 10, 50, and 100 dimensions for each problem. In our results, we use the pattern *function(dimension)* to indicate the simulation scenario. Thus, OneMax(100) means a simulation scenario using the OneMax problem with 100 dimensions.

For each scenario, we perform 30 simulations of each algorithm with 1,000 iterations and 30 agents. As a parametric analysis, we evaluate the impact of different MR values in BBGWO. We analyse the following MR values: 5%, 10%, 20%, 25%, 30%, 35%, 40%, 45%, 50%, 55% and 60%. For the BPSO algorithm, we adopt c_1 = c_2 = 2, w = 0.9, and Global topology [20].

V. Results

A. Impact of MR on the BBGWO performance

Previous works demonstrated that the number of dimensions flipped per iteration affects the performance of swarm-based algorithms [19]. In this way, we use MR to control how fast positions can be changed per iteration. Figure 2 shows the fitness values reached by the BBGWO for the OneMax, ZeroMax and 0-1 Knapsack problems using 10, 50 and 100 dimensions.

When we consider only ten dimensions for the problems, we see that the swarm converges similar regardless of the values of MR. As the 0-1 Knapsack problem is more complex than OneMax and ZeroMax, we observe that with 50 dimensions, the performance decreases as we increase the number of flipped dimensions because it harms the exploitation. In this way, the slow and careful convergence shows a high rate of efficiency using a similar number of iterations. For the case with 100 dimensions, we observed that 1,000 iterations are insufficient for the swarm convergence using small values of MR. Moreover, the performance decreases as we increase the value of MR for the 0-1 Knapsack problem. Thus, we propose that using 10% for MR is potentially more adequate for different problems. In the future, we would like to transform this parameter into an adaptive operator, so the swarm identifies the necessary balance between exploration and exploitation.

B. Individuals' diversity and convergence

In binary swarm-based algorithms, two limitations can be often identified: many individuals with similar positions and premature convergence. We demonstrate that the BBGWO can overcome both limitations multiple times over iterations due to Equation 4. We choose the 0-1 Knapsack problem with 100 dimensions (the most complex problem) to discuss the BBGWO dynamic behaviour.

First, in Figure 3, we show that the majority of the time, the swarm is not in the same position. We also identify multiple points in time that the swarm changes from entire diversity to half of the swarm being in unique positions. When the swarm converges near the optimal solution, the diversity reaches lower values than at previous points in time.

Second, we can measure the hamming distance among wolves as a proxy to estimate how near the wolves are to each other. Distance equal to zero means that all the swarm is at the same position, and as the distance increases, the swarm would be potentially farther from each other.

Figure 4 shows a higher decrease in the hamming distance on the first 200 iterations as the swarm is initially uniformly distributed in the search space and slowly converges. On iteration 220, we observe that the swarm increases the distance between each other and converges again. The same behaviour happens at iterations 283, 411, 450, 544, 600 and 891, which are also associated with a highly diverse set of positions in the swarm in Figure 3.

Thus, we can argue that BBGWO (i) can explore and exploit over iterations, (ii) can overcome premature convergence and (iii) most of the time, wolves are in different positions.

C. Comparing BBGWO to other algorithms

Lastly, we present the performance of BBGWO compared to the bGWO1, bGWO2 and BPSO algorithms in OneMax, ZeroMax, and 0-1 Knapsack in different scenarios. The results found in the simulations show that as we increase the number of dimensions, the performance across swarm-based algorithms displays a unique behaviour.

The results from OneMax with 10, 50 and 100 dimensions are shown in Figure 5 (A-C), respectively. Using 10 dimensions, we see that all algorithms have similar results. In higher dimensions, we see that bGWO1 is slightly better than BBGWO. Additionally, bGWO2 found the worst result in the problem.

Using ZeroMax, we found a similar behaviour, but bGWO2 changed place with bGWO1, as shown in Figure 6. Again,

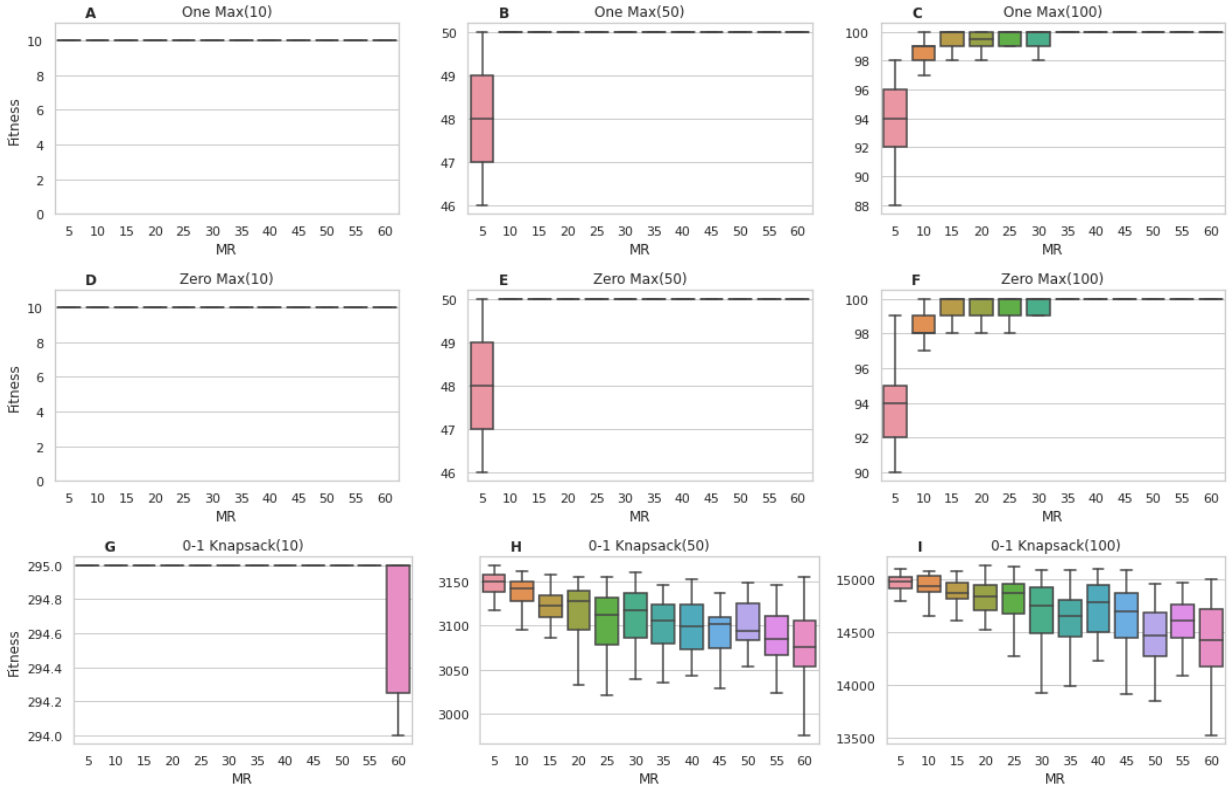

Fig. 2. Box plot of fitness values found by BBGWO varying *MR* on the 30 simulations of each problem using 10, 50 and 100 dimensions.

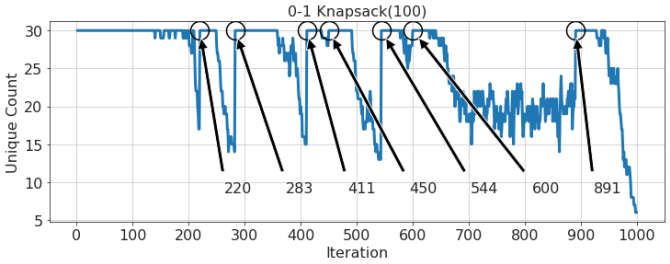

Fig. 3. Number of unique positions in the swarm over the iterations for the 0-1 Knapsack problem using 100 dimensions.

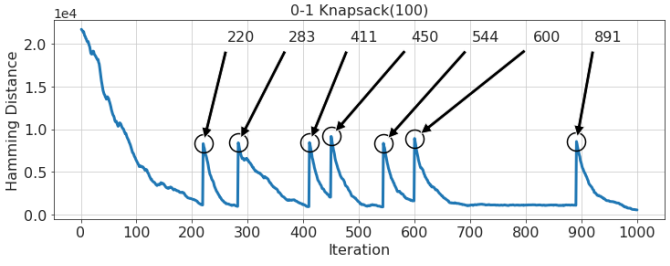

Fig. 4. Hamming distance between wolves over the iterations for the 0-1 Knapsack problem using 100 dimensions.

using 100 dimensions, we see BBGWO reaching good results, but bGWO2 is slightly better than the proposal.

OneMax and ZeroMax are opposite problems. We observe that for the OneMax, bGWO1 performed better than bGWO2, and for the ZeroMax problem, the bGWO2 performed better than bGWO1. We argue that this may be due to a biased operator to one direction (0 or 1) in each algorithm. In contrast, BBGWO showed near-optimal results in both problems. Figure 7 shows the results from the most challenging problem used in our simulations. Using 0-1 Knapsack, we see BBGWO outperforming the other algorithms in 50 and 100 dimensions.

Figures 5, 6 and 7 show how leaders' information quickly guides the BBGWO, bGWO1 and bGWO2 to the best region in the search space. In the initial iterations, we consistently see them showing better results than BPSO which means that BPSO has a slower convergence compared to them. For some scenarios, BPSO can overcome bGWO1 and bGWO2 due to its slower convergence.

Using a confidence rate of 95%, we apply the Wilcoxon test to compare the efficiency across algorithms, shown in Tables I and II. '–' indicates that there is no statistical diference between the solutions, '▲' indicates BBGWO achieved better results than the other algorithm and '▼' represents that our proposal reached worse results than algorithm compared. In general, BBGWO has a better performance than the other algorithms for 10 and 50 dimensions for all the problems. For the case of 100 dimensions, we observe in Table III that BBGWO also outperformed the other algorithms in general. However, in the case of OneMax and ZeroMax problems, bGWO1 and bGWO2 are slightly better than our proposal.

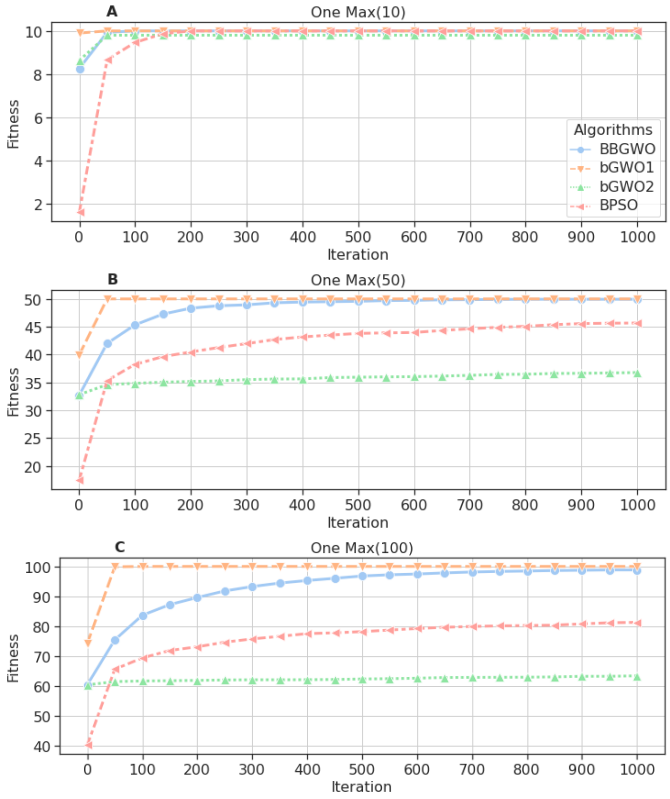

Fig. 5. Mean fitness found by BBGWO, bGWO1, bGWO2, and BPSO on the 30 simulations of OneMax problem using 10 (**A**), 50 (**B**) and 100 (**C**) dimensions.

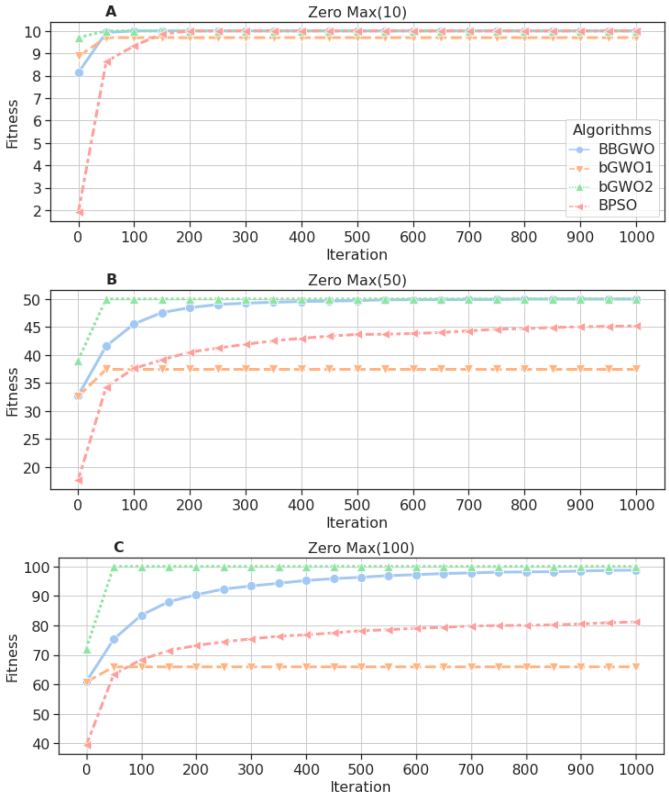

Fig. 6. Mean fitness found by BBGWO, bGWO1, bGWO2, and BPSO on the 30 simulations of ZeroMax problem using 10 (**A**), 50 (**B**) and 100 (**C**) dimensions.

TABLE I
RESULTS OF FITNESS VALUES AND WILCOXON TEST WITH A CONFIDENCE LEVEL OF 95% COMPARING THE BBGWO WITH THE OTHER ALGORITHMS USING 10 DIMENSIONS.

10 Dimensions					
Problem		BBGWO	bGWO1	bGWO2	BPSO
OneMax	Fitness	10.00	10.00	9.80	10.00
	STD	0.00	0.00	0.40	0.00
	Wilcoxon	–	–	▲	–
ZeroMax	Fitness	10.00	9.70	10.00	10.00
	STD	0.00	0.53	0.00	0.00
	Wilcoxon	-	▲	–	–
Knapsack	Fitness	295.00	292.30	287.20	293.33
	STD	0.00	8.08	14.45	1.99
	Wilcoxon	-	▲	▲	▲

TABLE II
RESULTS OF FITNESS VALUES AND WILCOXON TEST WITH A CONFIDENCE LEVEL OF 95% COMPARING THE BBGWO WITH THE OTHER ALGORITHMS USING 50 DIMENSIONS.

50 Dimensions					
Problem		BBGWO	bGWO1	bGWO2	BPSO
OneMax	Fitness	49.93	50.00	36.77	45.67
	STD	0.25	0.00	1.31	0.60
	Wilcoxon	-	–	▲	▲
ZeroMax	Fitness	49.97	37.43	50.00	45.20
	STD	0.18	1.50	0.00	0.60
	Wilcoxon	-	▲	–	▲
Knapsack	Fitness	3143.57	2937.87	2894.70	2965.47
	STD	12.45	75.01	85.67	17.83
	Wilcoxon	-	▲	▲	▲

TABLE III
RESULTS OF FITNESS VALUES AND WILCOXON TEST WITH A CONFIDENCE LEVEL OF 95% COMPARING THE BBGWO WITH THE OTHER ALGORITHMS USING 100 DIMENSIONS.

100 Dimensions					
Problem		BBGWO	bGWO1	bGWO2	BPSO
OneMax	Fitness	98.83	100.00	63.43	81.30
	STD	1.13	0.00	2.09	1.29
	Wilcoxon	-	▼	▲	▲
ZeroMax	Fitness	98.70	65.97	100.00	81.17
	STD	0.94	2.36	0.00	1.10
	Wilcoxon	-	▲	▼	▲
Knapsack	Fitness	14983.67	14107.80	10368.27	13266.50
	STD	103.85	279.81	334.36	119.67
	Wilcoxon	-	▲	▲	▲

VI. CONCLUSIONS

We proposed the Boolean Binary Grey Wolf Optimizer (BBGWO) that incorporates only binary operators using logic gates. BBGWO has high accuracy and a simple implementation. This proposal has a hyperparameter representing each wolf's rate of positions modified per iteration. Through a parametric analysis, we found that $MR = 10\%$ is potentially more adequate for the tested scenarios.

In two opposite problems, OneMax and ZeroMax, BBGWO showed promising results, demonstrating that it is not biased in one direction. BBGWO outperformed bGWO1, bGWO2, and BPSO on OneMax, ZeroMax, and 0-1 Knapsack problems. BBGWO displays better performance than the others as we in-

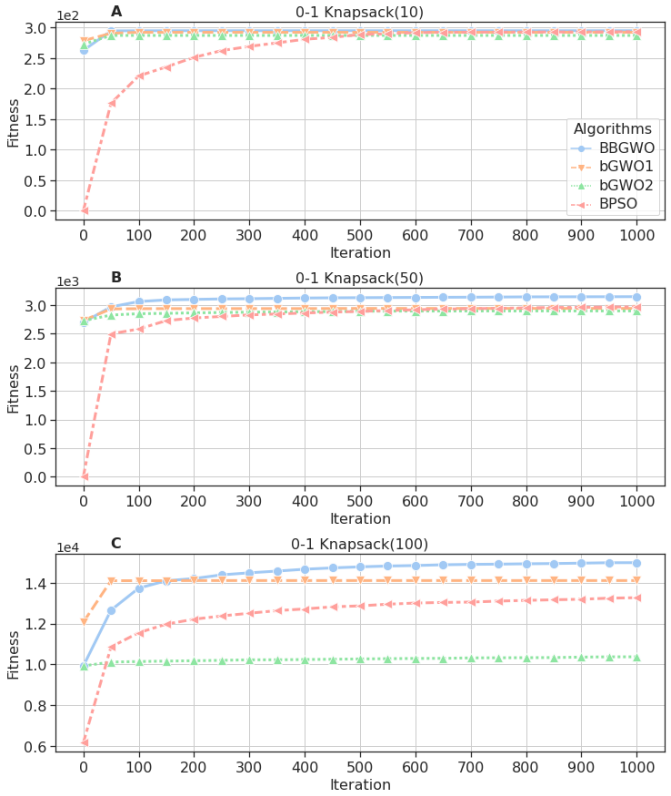

Fig. 7. Mean fitness found by BBGWO, bGWO1, bGWO2, and BPSO on the 30 simulations of 0-1 Knapsack problem using 10 (**A**), 50 (**B**), and 100 (**C**) dimensions.

crease the complexity of the problem. Thus, BBGWO showed a balanced convergence that allows the swarm to explore and exploit quick but carefully.

In addition, we found that the proposal can overcome premature convergence by evaluating the swarm hamming distance and the unique candidate solutions. Using our most complex scenario, most of the time, wolves are in different positions.

For future works, we aim to compare BBGWO with other evolutionary and swarm-based algorithms and apply it to other problems like Feature Selection. We also aim to simplify the parametrization of BBGWO by using an adaptive swarm behaviour. Better binary optimization algorithms are vital for solving real-world problems, as they can help with resource allocation, hyperparametrisation, dimension reduction or feature selection in different fields.

References

[1] M. Macedo, H. Siqueira, E. Figueiredo, C. Santana, R. C. Lira, A. Gokhale, and C. Bastos-Filho, "Overview on binary optimization using swarm-inspired algorithms," *IEEE Access*, vol. 9, pp. 149 814–149 858, 2021.

[2] P. Santos, M. Macedo, E. Figueiredo, C. J. Santana, F. Soares, H. Siqueira, A. Maciel, A. Gokhale, and C. J. Bastos-Filho, "Application of pso-based clustering algorithms on educational databases," in *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, 2017, pp. 1–6.

[3] C. García-Martinez, F. J. Rodriguez, and M. Lozano, "Analysing the significance of no free lunch theorems on the set of real-world binary problems," in *2011 11th International Conference on Intelligent Systems Design and Applications*, 2011, pp. 344–349.

[4] J. T. Belotti, D. S. Castanho, L. N. Araujo, L. V. da Silva, T. A. Alves, Y. S. Tadano, S. L. Stevan Jr, F. C. Corrêa, and H. V. Siqueira, "Air pollution epidemiology: A simplified generalized linear model approach optimized by bio-inspired metaheuristics," *Environmental Research*, vol. 191, p. 110106, 2020.

[5] E. D. Puchta, R. Lucas, F. R. Ferreira, H. V. Siqueira, and M. S. Kaster, "Gaussian adaptive pid control optimized via genetic algorithm applied to a step-down dc-dc converter," in *2016 12th IEEE International Conference on Industry Applications (INDUSCON)*. IEEE, 2016, pp. 1–6.

[6] S. Taghian, M. H. Nadimi-Shahraki, and H. Zamani, "Comparative analysis of transfer function-based binary metaheuristic algorithms for feature selection," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 2018, pp. 1–6.

[7] H. Siqueira, M. Macedo, Y. d. S. Tadano, T. A. Alves, S. L. Stevan, D. S. Oliveira, M. H. Marinho, P. S. Neto, J. F. de Oliveira, I. Luna *et al.*, "Selection of temporal lags for predicting riverflow series from hydroelectric plants using variable selection methods," *Energies*, vol. 13, no. 16, p. 4236, 2020.

[8] S. Reddy, L. K. Panwar, B. K. Panigrahi, R. Kumar, and A. Alsumaiti, "Binary grey wolf optimizer models for profit based unit commitment of price-taking genco in electricity market," *Swarm and evolutionary computation*, vol. 44, pp. 957–971, 2019.

[9] Q. M. Alzubi, M. Anbar, Z. N. M. Alqattan, M. A. Al-Betar, and R. Abdullah, "Intrusion detection system based on a modified binary grey wolf optimisation," *Neural Comput. Appl.*, vol. 32, no. 10, p. 6125–6137, may 2020. [Online]. Available: https://doi.org/10.1007/s00521-019-04103-1

[10] H. Siqueira, C. Santana, M. Macedo, E. Figueiredo, A. Gokhale, and C. Bastos-Filho, "Simplified binary cat swarm optimization," *Integrated Computer-Aided Engineering*, vol. 28, no. 1, pp. 35–50, 2021.

[11] K. Jiang, H. Ni, R. Han, and X. Wang, "An improved multi-objective grey wolf optimizer for dependent task scheduling in edge computing," *International Journal of Innovative Computing Information and Control*, vol. 15, no. 6, pp. 2289–2304, 2019.

[12] N. Pazhaniraja, S. Sountharrajan, and B. Sathis Kumar, "High utility itemset mining: a boolean operators-based modified grey wolf optimization algorithm," *Soft Computing*, vol. 24, no. 21, pp. 16 691–16 704, 2020.

[13] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[14] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," *Neurocomputing*, vol. 172, pp. 371–381, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231215010504

[15] K. Srikanth, L. K. Panwar, B. K. Panigrahi, E. Herrera-Viedma, A. K. Sangaiah, and G.-G. Wang, "Meta-heuristic framework: Quantum inspired binary grey wolf optimizer for unit commitment problem," *Computers & Electrical Engineering*, vol. 70, pp. 243–260, 2018.

[16] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[17] R. C. Lira, M. Macedo, H. V. Siqueira, R. Menezes, and C. Bastos-Filho, "Modelling the social interactions in grey wolf optimizer," in *2021 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 2021, pp. 1–6.

[18] M. Houshmand, S. H. Khayat, and R. Rezaei, "Genetic algorithm based logic optimization for multi-output majority gate-based nano-electronic circuits," in *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 1, 2009, pp. 584–588.

[19] C. J. Santana, M. Macedo, H. Siqueira, A. Gokhale, and C. J. Bastos-Filho, "A novel binary artificial bee colony algorithm," *Future Generation Computer Systems*, vol. 98, pp. 180–196, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18325652

[20] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, 1997, pp. 4104–4108 vol.5.

[21] L. J. V. Miranda, "PySwarms, a research-toolkit for Particle Swarm Optimization in Python," *Journal of Open Source Software*, vol. 3, 2018. [Online]. Available: https://doi.org/10.21105/joss.00433