International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 6, December 2023, pp. 7016~7026
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i6.pp7016-7026

Best-worst northern goshawk optimizer: a new stochastic optimization method

Purba Daru Kusuma, Faisal Candrasyah Hasibuan
Department of Computer Engineering, Faculty of Electrical Engineering, Telkom University, Bandung, Indonesia

Article Info

Article history:

Received Apr 12, 2023
Revised Jul 14, 2023
Accepted Jul 17, 2023

Keywords:

Agent system
Local search
Metaheuristic
Northern goshawk optimization
Particle swarm optimization
Stochastic optimization
Swarm intelligence

ABSTRACT

This study introduces a new metaheuristic method: the best-worst northern goshawk optimizer (BW-NGO). This algorithm is an enhanced version of the northern goshawk optimizer (NGO). Every BW-NGO iteration consists of four phases. First, each agent advances toward the best agent and away from the worst agent. Second, each agent moves relatively to the agent selected at random. Third, each agent conducts a local search. Fourth, each agent traces the space at random. The first three phases are mandatory, while the fourth phase is optional. Simulation is performed to assess the performance of BW-NGO. In this simulation, BW-NGO is confronted with four algorithms: particle swarm optimization (PSO), pelican optimization algorithm (POA), golden search optimizer (GSO), and northern goshawk optimizer (NGO). The result exhibits that BW-NGO discovers an acceptable solution for the 23 benchmark functions. BW-NGO is better than PSO, POA, GSO, and NGO in consecutively optimizing 22, 20, 15, and 11 functions. BW-NGO can discover the global optimal solution for three functions.

Corresponding Author:

Purba Daru Kusuma
Department of Computer Engineering, Faculty of Electrical Engineering, Telkom University
Bandung, Indonesia
Email: purbodaru@telkomuniversity.ac.id

1. INTRODUCTION

Metaheuristic is a technique that is extensively used in studies regarding optimization. Its popularity mainly comes from its flexibility in tackling various optimization problems, from simple ones to very complicated ones. Metaheuristic becomes flexible because it adopts an approximate approach that does not trace all possible solutions [1]. This strategy makes metaheuristics adaptive enough to solve optimization problems using limited computational resources. However, this benefit is offset by the metaheuristic choosing a suboptimal answer as the acceptable option and cannot guarantee the global optimal solution [1].

Recently, many metaheuristics have been ready to use for any optimization problems. Many old algorithms are still popular. Genetic algorithm (GA) is still implemented in recent engineering studies, such as hydrogen liquefaction process [2], town logistics distribution systems [3], construction projects [4], spectrometry peak detection [5], plain text encryption [6], Parkinson disease prediction [7], and so on. GA is also utilized in optimization studies related to the financial sector, such as in-stock selection [8], credit rating assessment [9], and cryptocurrency price forecasting [10]. Artificial bee colony (ABC) is also used in a lot of optimization studies in the engineering sector, such as in power contract capacity optimization [11], parallel machine scheduling [12], tunnel deformation prediction [13], and hybrid flow-shop scheduling [14]. Besides these old algorithms, many later algorithms have received positive attention. Grey wolf optimizer (GWO) has been modified and implemented in several engineering studies, such as in flood evacuation planning [15],

abrasive jet machining [16], workflow scheduling [17], and so on. Marine predator algorithm (MPA) is also implemented in many optimization studies in engineering, such as in determining the optimal parameters of supercapacitor [18], car design impact problem [19], welded beam design problem [19], detecting the structural damage [20], optimizing the performance and cost of heating, cooling, and power system [21], diagnosing the rolling bearing fault in the rotating machinery [22], and so on.

A lot of new metaheuristics were built based on swarm intelligence. The examples of them are tunicate swarm algorithm (TSA) [23], golden search optimization (GSO) [24], election-based optimization algorithm (EBOA) [25], hybrid leader based optimization (HLBO) [26], northern goshawk optimization (NGO) [27], butterfly optimization algorithm (BOA) [28], Komodo mlipir algorithm (KMA) [29], mixed leader based optimizer (MLBO) [30], multileader optimizer (MLO) [31], pelican optimization algorithm (POA) [32], and so on. Several new algorithms were built by modifying the previous swarm-based metaheuristic, such as the chaotic slime mold algorithm (CSMA) [33], stochastic Komodo algorithm (SKA) [34], modified honey badger algorithm (MHGA) [35], hybrid pelican Komodo algorithm (HPKA) [36], and many others.

Numerous factors contribute to the extensive development of metaheuristics. First, four familiar mechanics can be taken, especially in the swarm intelligence-based algorithm. These mechanics are getting closer to the high-quality solution or farther from the low-quality solution, interacting with other solutions, searching around the current solution (neighborhood/local search), and searching randomly within the space. Many variations can be taken in every part. Moreover, many algorithms are developed by conducting several searches. In almost all algorithms, the random search is only done in the initialization phase.

Meanwhile, several algorithms, such as simulated annealing (SA) [37], tabu search (TS) [38], and variable neighborhood search (VNS) [39], are developed based on local search. Second, various random distributions or movements can be chosen due to the characteristic of the metaheuristic as a stochastic system. Almost all algorithms choose uniform distribution due to its simplicity. Meanwhile, several algorithms choose other distributions. The normal distribution can be found in the invasive weed optimizer (IWO) [40]. Levy flight can be found in cuckoo search (CS) [41]. The combination between levy flight and Brownian movement can be found in MPA [42].

The massive development of metaheuristics is also correlated with the fact that no algorithm or method is suitable for tackling all and various problems. The example is as follows. HPKA is superior to GWO in solving the fixed dimensional multimodal functions but inferior to GWO in solving the high dimensional unimodal functions [36]. On the other hand, GWO is inferior to MPA, KMA, POA, and HPKA in solving the portfolio optimization problem in which the solution space is presented in an integer [36]. GSO generally outperforms the gravitational search algorithm (GSA) but is still inferior in solving step functions [24]. NGO is inferior to GSA in solving penalized two, although NGO generally is superior among other benchmark algorithms [27]. SKA can discover the best result only in 4 out of 10 functions compared to other benchmark algorithms [34]. GA can outperform BOA in solving two functions, although, in general, BOA outperforms GA [28]. This circumstance occurs because a combined method or random distribution may be effective for some problems but less effective for others.

Moreover, many metaheuristics that hybridize previous algorithms still have a weakness. On the other hand, optimizing many new challenges, such as electric vehicles, cloud computing, and others, is not easy. It makes the study in proposing a new metaheuristic still promising.

Regarding this problem and opportunity, this study proposes a new metaheuristic that modifies the primary form of NGO. There are several reasons for choosing NGO as a baseline. First, NGO is a new algorithm, and this algorithm's exploration, implementation, and modification are still limited. Second, NGO is unique because it does not adopt the mechanism to follow the highest quality agent, although NGO is a swarm intelligence-based metaheuristic.

The modification is done by accommodating the four standard searches in the metaheuristics, especially those built based on swarm intelligence. In every iteration, all agents do these four searches. One search is optional, while the other three are mandatory. The best-worst northern goshawk optimizer (BW-NGO) is the name of the proposed method. The best-worst term comes from the action that accommodates movement toward the best agent and away from the worst agent. As the improvement version of NGO, below are the contributions of this work: i) BW-NGO is an enhanced version of NGO as the new metaheuristic and ii) BW-NGO implements four common searches in every iteration.

The remainder of this paper is constructed as follows. Section 2 explores and reviews the previous studies regarding metaheuristics, especially algorithms developed based on swarm intelligence (SI). Section 3 presents the model of BW-NGO that consists of the core concept, algorithm, and mathematical model. Section 4 presents the simulation carried out in this work to evaluate the performance of BW-NGO, it is result, and the in-depth analysis of the result and findings. Section 5 summarizes this work and future opportunities.

2. RELATED WORKS

Based on the common perspective, two strategies are carried out by any metaheuristic. The first strategy is exploration, and the second strategy is exploitation. This perspective is common and can be used in the single solution-based algorithm, such as VNS, TS, and SA, or the population-based algorithm, such as ABC, GA, and PSO. The exploitation can be interpreted as any effort to discover a better solution around or near the current one. Exploration can be interpreted as any effort to discover a better solution anywhere within space.

As the SI-based metaheuristic is extensively studied, there is a new perspective that enriches the previous perspective. Four common strategies are found in many new metaheuristics. The first strategy is getting closer to the highest quality solution and going away from the lowest quality solution. The second strategy involves engaging with one or more chosen solutions from the population. Discovering a superior solution close to the current one is the third strategy. The fourth strategy is to do a random search inside the search area. This perspective can be used as an alternative because there is an ambiguous distinction between exploration and exploitation in many recent swarm intelligence-based metaheuristics. Below is a deeper review of these four strategies.

The first strategy is getting closer to the highest quality solution and going away from the worst solution. In general, getting closer to the highest quality solution is more popular than moving away from the worst solution. In some algorithms, the highest quality solution is called a leader. Each algorithm builds its method regarding this first strategy. In PSO, there are two types of highest quality solutions: global and local best [43]. In PSO, the existence of the lowest quality solution is not considered. Every agent tries to move toward the global and local best with a certain proportion [43]. In GSO, every agent also moves toward the global and local best, but the proportion is determined stochastically and enriched with sinusoid movement [24]. In GSO, replacing the worst solution with the randomly selected solution can be seen as an effort to avoid the worst solution [24]. In MPA, the predator represents the local best, and there is an interaction between every predator and its prey [42]. In KMA, the movement toward the best solution can be found in many parts: the movement of giant male dragons toward the better big male dragons and the small male dragons toward the giant male dragons [29]. In GWO, all agents move toward the results of the best three agents [44]. These three agents are obtained by a sorting process carried out in every iteration [44]. MLO also adopts multiple global leaders. The similarity is that these leaders are also obtained by sorting in every iteration [31]. The difference is that in MLO, the number of leaders is determined by the user [31]. In shell game optimizer (SGO), the best solution among the population in every iteration becomes one factor considered in the first strategy besides two randomly selected solutions [45]. Meanwhile, the worst solution is avoided by putting the worst solution in the lowest probability of being chosen [45].

The second strategy involves interacting with the corresponding and randomly selected agents. This strategy is less popular than the first strategy. This method is employed during the Eddy formation phase of MPA when there is a chance that the prey would travel toward the difference between two randomly picked prey [42]. This action is a part of the exploration. In NGO, the first guided movement is the movement of every agent relative to a randomly selected agent [27]. In this movement, the corresponding agent moves toward the randomly selected agent if this selected agent is better than the corresponding agent [27]. Otherwise, the corresponding agent moves away from the selected agent [27]. In HLBO, the randomly selected agent is constructed together with the best agent to become the target [26]. In HLBO, the proportion of the best agent and the randomly selected agent is determined based on the normalized quality of these agents. The combination of the best and randomly selected agents can also be found in MLBO. The difference is that the proportion of the best agent increases as the iteration continues, while the proportion of the randomly selected agent decreases [30]. In SGO, two randomly selected agents are combined with the best agent [45].

The third strategy is the local search or neighborhood search. In general, this strategy is exploitation. In old metaheuristics such as SA and TS, this strategy becomes the backbone as the modification of the current solution is carried out in every iteration. In IWO, spreading new seeds around the current seed with normal distribution also can be seen as the interpretation of this third strategy [40]. The new seeds will be concentrated near the current seed, although there is the possibility that the new seed is far from the current seed. Levy flight can be perceived as a local search when the flight distance is short. The guided movement can also be seen as a local search when the corresponding agent is near the target. In MPA, the second option of the eddy formation can also be seen as a local search with certain ambiguity [42]. In the early iteration, the local space is broad and declines as the iteration goes on [42]. Other algorithms, such as NGO [27], MLO [31], and HLBO [26], also carry out a similar strategy in their second phase. The distinction between these algorithms and MPA is that their method only accepts the new solution generated by the local search if it is superior to the existing solution.

The fourth strategy is the random search around space. This random search usually follows a uniform distribution, so the probability of any solution within the solution space becoming the initial solution is equal. In general, all metaheuristics deploy this strategy in the initialization phase. Nevertheless, only a few algorithms implement this strategy in the iteration phase. In ABC, the scout bee discovers a new food source within the solution space after the onlooker bee fails to discover a better solution around the current food source within some trials [46]. In harmony search (HS), the random search is also carried out if the generated random number is higher than the harmony memory consideration rate (HMCR) [47].

Based on this explanation, there is an opportunity to develop a new metaheuristic that accommodates the four common strategies in every iteration. It is because this kind of algorithm is still challenging to discover. Many old algorithms are developed based on local search. Most of all, a swarm intelligence-based metaheuristic focuses on the strategy related to the highest quality agent. Some new algorithms adopt local search as complementary to the first strategy. Only a few algorithms carry out a random search in the iteration phase.

3. METHOD

The central concept of BW-NGO is accommodating four joint movements in the SI. This accommodation is transformed into four phases that the agents carry out. The first phase entails moving towards the best agent and away from the worst agent. The second phase consists of agent mobility relative to the system. The local search phase is the third phase. The random search within space constitutes the fourth phase. The first three phases are required for all agents in each iteration.

Meanwhile, the fourth phase is optional. It is carried out only if these three phases do not produce any improvement. These phases are conducted sequentially. Below is a more detailed explanation of every phase. Exploration helps avoid the local optimal entrapment, while exploitation helps improve the current solution, especially when the current solution is in the region where the global optimal solution exists.

The first phase is carried out by moving toward the best agent and avoiding the worst agent. These best and worst agents are determined from the previous iteration. These two movements are carried out in a single movement. In the early iteration, the proportion of moving away from the worst agent is high, while the proportion of moving toward the best agent is low. This proportion shifts gradually as the iteration goes on. The proportion of the movement toward the best agent increases.

Meanwhile, the proportion of the movement away from the worst agent decreases. This strategy is developed based on several reasons. In the beginning, optimization is designed to avoid low-quality solutions. Meanwhile, as the iteration goes, the optimization will focus on the improvement toward the best solution.

The second phase creates interaction between the corresponding agent and the other agent in the system. This other agent is selected randomly. Then, the quality of both agents is compared. If the selected agent is better than the corresponding agent, then this corresponding agent moves toward the selected agent. Contrary, if the selected agent is not better than the corresponding agent, then this corresponding agent avoids the selected agent.

The third phase is the local search or neighborhood search. The new solution is generated randomly around the corresponding agent within its observation range. In the early iteration, the observation range was wide to accommodate the exploration. Then, this observation range decreases gradually as the iteration goes on. It is seen as the strategy shifting toward exploitation.

The fourth phase is the random search. It is carried out by generating a new solution within space. This phase is designed to focus on exploration. This step is only implemented if there is no improvement after implementing the preceding three phases. It can be interpreted that the acceptable solution has been found or the agent is trapped on the local optimal.

There is a candidate generated in every phase for every agent. This candidate becomes the replacement for the corresponding agent only if this candidate has better quality than the corresponding agent. Otherwise, the agent remains in its current state or solution.

This concept is then transformed into the algorithm. The algorithm of BW-NGO is presented in algorithm 1. Meanwhile, (1) to (15) are the formalization of processes in the algorithm. Several annotations used in this formal model are as follows.

a : Agent
A : Set of agents
a_{best} : The best agent
a_{worst} : The worst agent
a_{gbest} : The global best agent
b_l : Lower boundary
b_u : Upper boundary
c_1 : First candidate

c_2 : Second candidate
c_3 : Third candidate
c_4 : Fourth candidate
d_1 : First direction
d_2 : Second direction
f : Fitness/objective function
U : Uniform random

Below is the explanation of algorithm 1. The optimization process is split into two steps: the initialization and the iteration. The random search is done in the initialization to discover the agent's initial solution. The iteration has two sub-steps. Discovering the best agent, the worst agent, and the worldwide best agent is the first sub-step. The second sub-step consists of all agents' movement. The global best agent is the agent whose quality is the best to the current the iteration. This global best agent becomes the final solution.

Algorithm 1. Best-worst northern goshawk optimizer

```
output: a_gbest
begin
  for all A
    initialize a using (1)
  end for
  t=1
  while t≤t_max
    discover a_best using (2)
    discover a_worst using (3)
    update a_gbest using (4)
    for all A
      conduct the first phase using (5) to (8)
      conduct the second phase using (9) to (11)
      conduct the third phase using (12) and (13)
      if improvement fails, then
        conduct the fourth phase using (14) and (15)
      end if
  end while
end
```

$$a = U(b_l, b_u) \tag{1}$$

$$a_{best,t} = a \in A \wedge \min\big(f(a_t)\big) \tag{2}$$

$$a_{worst,t} = a \in A \wedge \max\big(f(a_t)\big) \tag{3}$$

$$a'_{gbest} = \begin{cases} a_{best}, f(a_{best}) < f\big(a_{gbest}\big) \\ a_{gbest}, else \end{cases} \tag{4}$$

$$d_1 = U(0,1).\left(\frac{t}{t_{max}}\right).(a_{best} - U(1,2).a) \tag{5}$$

$$d_2 = U(0,1).\left(1 - \frac{t}{t_{max}}\right).(a - U(1,2).a_{worst}) \tag{6}$$

$$c_1 = a + d_1 + d_2 \tag{7}$$

$$a' = \begin{cases} c_1, f(c_1) < f(a) \\ a, else \end{cases} \tag{8}$$

$$a_s = U(A) \tag{9}$$

$$c_2 = \begin{cases} a + U(0,1).(a_s - U(1,2).a), f(a_s) < f(a) \\ a_s + U(0,1).(a - U(1,2).a_s), else \end{cases} \tag{10}$$

$$a' = \begin{cases} c_2, f(c_2) < f(a) \\ a, else \end{cases} \tag{11}$$

$$c_3 = a + \left(1 - \frac{t}{t_{max}}\right).(2.U(0,1) - 1).a \quad (12)$$

$$a' = \begin{cases} c_3, f(c_3) < f(a) \\ a, else \end{cases} \quad (13)$$

$$c_4 = U(b_l, b_u) \quad (14)$$

$$a' = \begin{cases} c_4, f(c_4) < f(a) \\ a, else \end{cases} \quad (15)$$

Below is the explanation of (1) to (15). In (1) states that the agent's initial solution is generated randomly within the space. In (2) states that the best agent is the agent whose solution is the best among all agents in the current iteration. In (3) states that the worst agent is the agent whose solution is the worst among all agents in the current iteration. In (4) states that the best agent becomes the global best agent only if it is better than the current global best agent. In (5) states that the proportion of the movement toward the best agent is linearly proportional to the iteration. In (6) states that the proportion of the movement away from the worst agent is reversely proportional to the iteration. In (7) states that the first candidate is obtained by combing the agent's current solution with movements generated from (5) and (6). In (8) states that this first candidate replaces the agent's current solution only if this first candidate is better than the current one. In (9) states that the agent is selected randomly from the population. In (10) states that the second candidate is obtained by the movement of the corresponding agent relative to the selected agent. The movement toward the selected agent is carried out if the selected agent is better than the corresponding agent. The opposite direction is taken if the selected agent is not better than the corresponding agent. In (11) states that the second candidate replaces the agent's current solution only if this second candidate is better than the agent's current solution. In (12) formalizes the local search and generates the third candidate. In (12) also exhibits that the local space declines gradually as the iteration continues. In (13) states that the third candidate replaces the agent's current solution only if this third candidate is better than the agent's current solution. In (14) states that the fourth candidate is generated randomly within the space. In (15) states that the fourth candidate only replaces the agent's current solution if it is better than the current one.

The BW-NGO complexity can be expressed as $O(2t_{max}.n(A))$. The following explains this presentation: BW-complexity NGOs are proportional to the maximum number of iterations or population size. As exhibited in the iteration, the iteration from the first iteration to the maximum iteration becomes the outer loop. In contrast, the iteration from the first agent to the last agent constitutes the inner loop. Every outer loop contains two inner loops in succession. The initial inner loop determines which agents are the greatest, worst, and global best. Tracing the quality of all agents is required to complete this process. To accommodate the modification, the second inner loop is executed.

4. RESULTS AND DISCUSSION

4.1. Simulation

Simulation is carried out to assess the performance of BW-NGO. This work uses BW-NGO to overcome the acceptable solution of the 23 benchmark functions. These 23 functions have been used extensively and have become common in many studies proposing a new metaheuristic. The popularity of these functions comes from several reasons. First, these functions consist of 7 unimodal functions and 16 multimodal functions. Second, these functions consist of 13 high dimensional functions and ten fixed dimensional functions. In the high dimensional functions, the dimension varies from 1 to thousands. In the fixed dimensional functions, the dimension is fixed, and it is usually tiny. These functions also represent problems with various spaces, from narrow to very wide. Generally, these functions can be categorized into three clusters: high dimension unimodal, high dimension multimodal, and fixed dimension multimodal functions. Table 1 provides a thorough summary of the functions. The high-dimensional functions F1 through F7 are unimodal. F8 through F13 are high-dimensional functions with several modes. Finally, F14 to F23 are multimodal functions with fixed dimensions.

This simulation benchmarked BW-NGO with four algorithms: PSO, POA, GSO, and NGO. PSO represents the early metaheuristic developed based on swarm intelligence. It is necessary to measure how far BW-NGO is better than its origin. NGO is chosen to measure the improvement of BW-NGO with the NGO as its basic form. POA and GSO are chosen because these two algorithms are new SI-based metaheuristics. The result is exhibited in Table 2.

Table 2 exhibits that, in general, BW-NGO can discover the acceptable solution for the 23 functions. Moreover, it can be the global optimal solution for three functions: Six Hump Camel, Branin, and Goldstein Price. All these three functions are fixed-dimension multimodal functions. BW-NGO is also very competitive compared to the four algorithms. The data regarding the number of functions where BW-NGO beats the benchmark algorithms are exhibited in Table 3.

Table 1. Benchmark functions

No	Function	Dim	Space	Target
F1	Sphere	50	[-100, 100]	0
F2	Schwefel 2.22	50	[-100, 100]	0
F3	Schwefel 1.2	50	[-100, 100]	0
F4	Schwefel 2.21	50	[-100, 100]	0
F5	Rosenbrock	50	[-30, 30]	0
F6	Step	50	[-100, 100]	0
F7	Quartic	50	[-1.28, 1.28]	0
F8	Schwefel	50	[-500, 500]	-418.9×*dim*
F9	Ratsrigin	50	[-5.12, 5.12]	0
F10	Ackley	50	[-32, 32]	0
F11	Griewank	50	[-600, 600]	0
F12	Penalized	50	[-50, 50]	0
F13	Penalized 2	50	[-50, 50]	0
F14	Shekel Foxholes	2	[-65, 65]	1
F15	Kowalik	4	[-5, 5]	0.0003
F16	Six Hump Camel	2	[-5, 5]	-1.0316
F17	Branin	2	[-5, 5]	0.398
F18	Goldstein-Price	2	[-2, 2]	3
F19	Hartman 3	3	[1, 3]	-3.86
F20	Hartman 6	6	[0, 1]	-3.32
F21	Shekel 5	4	[0, 10]	-10.1532
F22	Shekel 7	4	[0, 10]	-10.4028
F23	Shekel 10	4	[0, 10]	-10.5363

Table 2. Simulation result

Function	PSO	POA	GSO	NGO	BW-NGO	Better Than
F1	1.508×10^{4}	6.491×10^{4}	1.546×10^{4}	1.196×10^{-12}	1.145×10^{-20}	PSO, POA, GSO, NGO
F2	0	0	1.810×10^{60}	0	1.046×10^{-152}	GSO
F3	5.941×10^{4}	1.499×10^{5}	2.878×10^{4}	3.182×10^{1}	9.900	PSO, POA, GSO, NGO
F4	3.522×10^{1}	7.151×10^{1}	3.500×10^{1}	1.669×10^{-5}	1.873×10^{-8}	PSO, POA, GSO, NGO
F5	9.213×10^{6}	1.513×10^{8}	1.075×10^{7}	4.881×10^{1}	4.872×10^{1}	PSO, POA, GSO, NGO
F6	1.398×10^{4}	6.244×10^{4}	1.403×10^{4}	8.532	7.428	PSO, POA, GSO, NGO
F7	5.548	1.258×10^{2}	5.518	4.700×10^{-3}	1.457×10^{-2}	PSO, POA, GSO
F8	-3.741×10^{3}	-4.450×10^{3}	-6.392×10^{3}	-6.053×10^{3}	-7.773×10^{3}	PSO, POA, GSO, NGO
F9	4.032×10^{2}	6.035×10^{2}	3.214×10^{2}	5.556×10^{-11}	2.293×10^{-4}	PSO, POA, GSO
F10	1.463×10^{1}	1.980×10^{1}	1.998×10^{1}	1.996×10^{-7}	1.769×10^{-11}	PSO, POA, GSO, NGO
F11	1.393×10^{2}	5.936×10^{2}	1.360×10^{2}	2.350×10^{-8}	7.668×10^{-12}	PSO, POA, GSO, NGO
F12	1.603×10^{6}	2.687×10^{8}	2.223×10^{6}	5.784×10^{-1}	4.698×0^{-1}	PSO, POA, GSO, NGO
F13	1.379×10^{7}	6.724×10^{8}	1.774×10^{7}	2.990	2.875	PSO, POA, GSO, NGO
F14	7.311	1.811	5.051	1.047	4.720	PSO, GSO
F15	1.715×10^{-2}	2.822×10^{-3}	2.171×10^{-3}	5.323×10^{-4}	4.525×10^{-4}	PSO, POA, GSO, NGO
F16	-1.023	-1.029	-1.032	-1.032	-1.031	PSO, POA
F17	5.356×10^{-1}	4.026×10^{-1}	3.981×10^{-1}	3.981×10^{-1}	3.981×10^{-1}	PSO, POA
F18	7.786	3.061	3.000	3.000	3.000	PSO, POA
F19	-2.123×10^{-1}	-4.954×10^{-2}	-4.954×10^{-2}	-4.954×10^{-2}	-4.954×10^{-2}	PSO
F20	-2.686	-2.914	-3.272	-3.283	-3.199	PSO, POA
F21	-4.743	-3.517	-9.086	-9.183	-5.525	PSO, POA
F22	-4.542	-4.236	-8.465	-9.776	-5.547	PSO, POA
F23	-3.706	-3.454	-8.904	-9.440	-6.160	PSO, POA

Table 3. Cluster based comparison result

Cluster	Number of Functions Beaten by BW-NGO			
	PSO	POA	GSO	NGO
1	6	6	7	5
2	6	6	6	5
3	10	8	2	1
Total	22	20	15	11

Table 3 exhibits that the improvement produced by BW-NGO is significant. BW-NGO is superior to PSO and POA by beating these algorithms in 22 and 20 functions, respectively. BW-NGO is also very competitive by creating better performance than GSO in 15 functions. Meanwhile, the draw between BW-NGO and GSO occurs in three functions: Branin, Goldstein Price, and Hartman 3. It means that GSO is better than BW-NGO in only 5 functions. BW-NGO is better than NGO in 11 functions and draws in 3 functions. It means that NGO is still better than BW-NGO in 9 functions.

The second simulation is carried out to evaluate the performance of BW-NGO with the population size variation. In this simulation, the population size is set at 5, 10, and 15. These values represent a low population size. The result is exhibited in Table 4.

Table 4. Relation between population size and performance

Function	Average Fitness Score		
	$n(A)$=5	$n(A)$=10	$n(A)$=15
F1	8.689×10^{-22}	8.975×10^{-21}	9.152×10^{-21}
F2	0	1.270×10^{-215}	2.891×10^{-170}
F3	5.925	4.201	7.580
F4	3.159×10^{-8}	2.438×10^{-8}	2.753×10^{-8}
F5	4.888×10^{1}	4.877×10^{1}	4.874×10^{1}
F6	9.346	8.282	7.678
F7	1.714×10^{-2}	1.221×10^{-2}	1.371×10^{-2}
F8	-8.262×10^{3}	-7.907×10^{3}	-7.909×10^{3}
F9	8.748×10^{-2}	1.149	2.097×10^{-7}
F10	3.769×10^{-12}	1.174×10^{-11}	1.766×10^{-11}
F11	4.917×10^{-11}	2.019×10^{-13}	4.596×10^{-14}
F12	7.390×10^{-1}	5.790×10^{-1}	5.183×10^{-1}
F13	2.981	2.939	2.906
F14	5.637	4.660	4.667
F15	1.608×10^{-3}	1.553×10^{-3}	5.166×10^{-4}
F16	-1.030	-1.032	-1.032
F17	3.981×10^{-1}	3.981×10^{-1}	3.981×10^{-1}
F18	1.088×10^{1}	3.000	3.000
F19	-4.954×10^{-2}	-4.954×10^{-2}	-4.954×10^{-2}
F20	-3.014	-3.167	-3.191
F21	-5.531	-5.914	-6.369
F22	-5.702	-5.255	-5.576
F23	-4.581	-5.554	-6.011

4.2. Discussion

In this section, an in-depth analysis of the simulation result is carried out. This analysis is conducted based on the head-to-head comparison between BW-NGO and the benchmark algorithm. This analysis consists of the result and the strategy difference between BW-NGO and the benchmark algorithm.

The first comparison is the comparison between BW-NGO and PSO. Overall, BW-NGO outperforms PSO in almost all functions. PSO outperforms BW-NGO only in solving Schwefel 2.22. This superiority can be seen as a significant gap implemented in PSO and BW-NGO. In PSO, all agents move toward the global and local best agents with a certain proportion or weight. PSO carries out random movement only in the initialization phase. It differs from BW-NGO, which carries out all strategies in every iteration.

The second comparison is the comparison between BW-NGO and POA. Like PSO, BW-NGO also outperforms POA in almost all functions. BW-NGO outperforms POA on all functions in the second cluster. Meanwhile, POA outperforms BW-NGO on 1 and 2 functions in the first and third clusters. Different from PSO, POA focuses on random search and local search. POA conducts a full random search in the initialization phase.

Meanwhile, POA conducts partial random movement in the first phase of every iteration. It is called partial because after the random target is generated, the agent can move toward or away relative to this target based on the quality. The local search is carried out in the second phase.

The third comparison is the comparison between BW-NGO and GSO. Table 3 exhibits that BW-NGO is superior to GSO in the first and second clusters but inferior in the third cluster. It means that BW-NGO is superior to GSO in solving high-dimension problems. On the other hand, GSO is superior to BW-NGO in solving low-dimension problems. From the strategy perspective, GSO implements three of four movements in the SI. First, GSO implements a random search in the initialization. Second, GSO implements movement toward the global and local best with each iteration. In addition, GSO avoids the worst solution by replacing the worst agent in each iteration with a randomly picked agent. Thirdly, GSO implements local search with each iteration by combining this with the initial movement.

The fourth comparison is the comparison between BW-NGO and NGO. Table 3 exhibits that BW-NGO outperforms NGO in solving 11 functions. Meanwhile, Table 2 exhibits that BW-NGO is drawn to NGOs in solving four functions in the third cluster: Six Hump Camel, Branin, Goldstein-Price, and Hartman 3. It means that NGO is still superior to BW-NGO in solving eight functions: two functions in the first cluster, one in the second cluster, and five in the third cluster. It means that, in general, BW-NGO is superior to NGOs in solving big-dimension problems but less superior in solving low-dimension problems.

5. CONCLUSION

This work has demonstrated that the proposed algorithm, the BW-NGO, is a competitive metaheuristic. It can discover an acceptable solution in solving all functions. Moreover, it can discover the global optimal solution in solving three functions: Six Hump Camel, Branin, and Goldstein Price. BW-NGO is also competitive compared to the benchmark algorithms. BW-NGO is better than PSO, POA, GSO, and NGO in optimizing 22, 20, 15, and 11 functions. Overall, BW-NGO is superior in solving big-dimension problems. The simulation result also exhibits that the BW-NGO can provide an acceptable solution with a low population size. Several opportunities can be created based on this work. This work has proven that the modification of NGOs is still open. Future studies can also be conducted by comparing BW-NGO and NGO in solving various real-world optimization problems, from numerical to combinatorial problems.

ACKNOWLEDGEMENTS

The authors received financial support for publication of this article from Telkom University, Indonesia.

REFERENCES

[1] H. R. Moshtaghi, A. T. Eshlaghy, and M. R. Motadel, "A comprehensive review on meta-heuristic algorithms and their classification with novel approach," *Journal of Applied Research on Industrial Engineering*, vol. 8, no. 1, pp. 63–89, 2021.

[2] J. Zhu, G. Wang, Y. Li, Z. Duo, and C. Sun, "Optimization of hydrogen liquefaction process based on parallel genetic algorithm," *International Journal of Hydrogen Energy*, vol. 47, no. 63, pp. 27038–27048, Jul. 2022, doi: 10.1016/j.ijhydene.2022.06.062.

[3] H. Cui, J. Qiu, J. Cao, M. Guo, X. Chen, and S. Gorbachev, "Route optimization in township logistics distribution considering customer satisfaction based on adaptive genetic algorithm," *Mathematics and Computers in Simulation*, vol. 204, pp. 28–42, Feb. 2023, doi: 10.1016/j.matcom.2022.05.020.

[4] C. Liu, F. Zhang, H. Zhang, Z. Shi, and H. Zhu, "Optimization of assembly sequence of building components based on simulated annealing genetic algorithm," *Alexandria Engineering Journal*, vol. 62, pp. 257–268, Jan. 2023, doi: 10.1016/j.aej.2022.07.025.

[5] J. Zhou *et al.*, "Combination of continuous wavelet transform and genetic algorithm-based Otsu for efficient mass spectrometry peak detection," *Biochemical and Biophysical Research Communications*, vol. 624, pp. 75–80, Oct. 2022, doi: 10.1016/j.bbrc.2022.07.083.

[6] R. B. Abduljabbar, O. K. Hamid, and N. J. Alhyani, "Features of genetic algorithm for plain text encryption," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 1, pp. 434–441, Feb. 2021, doi: 10.11591/ijece.v11i1.pp434-441.

[7] N. Benayad, Z. Soumaya, B. D. Taoufiq, and A. Abdelkrim, "Features selection by genetic algorithm optimization with k-nearest neighbour and learning ensemble to predict Parkinson disease," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 2, pp. 1982–1989, Apr. 2022, doi: 10.11591/ijece.v12i2.pp1982-1989.

[8] M. Ozcalici and M. Bumin, "Optimizing filter rule parameters with genetic algorithm and stock selection with artificial neural networks for an improved trading: The case of Borsa Istanbul," *Expert Systems with Applications*, vol. 208, Dec. 2022, doi: 10.1016/j.eswa.2022.118120.

[9] R. Estran, A. Souchaud, and D. Abitbol, "Using a genetic algorithm to optimize an expert credit rating model," *Expert Systems with Applications*, vol. 203, Oct. 2022, doi: 10.1016/j.eswa.2022.117506.

[10] S. G. Quek, G. Selvachandran, J. H. Tan, H. Y. A. Thiang, N. T. Tuan, and L. H. Son, "A new hybrid model of fuzzy time series and genetic algorithm based machine learning algorithm: A case study of forecasting prices of nine types of major cryptocurrencies," *Big Data Research*, vol. 28, May 2022, doi: 10.1016/j.bdr.2022.100315.

[11] C.-C. Lo and C.-S. Chiang, "Application of the artificial bee colony algorithm to power contract capacity optimization," *Microprocessors and Microsystems*, vol. 93, Sep. 2022, doi: 10.1016/j.micpro.2022.104621.

[12] D. Lei and H. Yang, "Scheduling unrelated parallel machines with preventive maintenance and setup time: Multi-sub-colony artificial bee colony," *Applied Soft Computing*, vol. 125, Aug. 2022, doi: 10.1016/j.asoc.2022.109154.

[13] T. Feng, C. Wang, J. Zhang, B. Wang, and Y.-F. Jin, "An improved artificial bee colony-random forest (IABC-RF) model for predicting the tunnel deformation due to an adjacent foundation pit excavation," *Underground Space*, vol. 7, no. 4, pp. 514–527, Aug. 2022, doi: 10.1016/j.undsp.2021.11.004.

[14] X.-R. Tao, Q.-K. Pan, and L. Gao, "An efficient self-adaptive artificial bee colony algorithm for the distributed resource-constrained hybrid flowshop problem," *Computers and Industrial Engineering*, vol. 169, Jul. 2022, doi: 10.1016/j.cie.2022.108200.

[15] S. Khalilpourazari and S. H. R. Pasandideh, "Designing emergency flood evacuation plans using robust optimization and artificial intelligence," *Journal of Combinatorial Optimization*, vol. 41, no. 3, pp. 640–677, Apr. 2021, doi: 10.1007/s10878-021-00699-0.

[16] K. Kalita, S. Pal, S. Haldar, and S. Chakraborty, "A hybrid TOPSIS-PR-GWO approach for multi-objective process parameter

optimization," *Process Integration and Optimization for Sustainability*, vol. 6, no. 4, pp. 1011–1026, Dec. 2022, doi: 10.1007/s41660-022-00256-0.
[17] N. Arora and R. Kumar, "HPSOGWO: A hybrid algorithm for scientific workflow scheduling in cloud computing," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 10, 2020, doi: 10.14569/IJACSA.2020.0111078.
[18] D. Yousri, A. Fathy, and H. Rezk, "A new comprehensive learning marine predator algorithm for extracting the optimal parameters of supercapacitor model," *Journal of Energy Storage*, vol. 42, Oct. 2021, doi: 10.1016/j.est.2021.103035.
[19] K. Zhong, G. Zhou, W. Deng, Y. Zhou, and Q. Luo, "MOMPA: Multi-objective marine predator algorithm," *Computer Methods in Applied Mechanics and Engineering*, vol. 385, Nov. 2021, doi: 10.1016/j.cma.2021.114029.
[20] L. V. Ho *et al.*, "A hybrid computational intelligence approach for structural damage detection using marine predator algorithm and feedforward neural networks," *Computers and Structures*, vol. 252, Aug. 2021, doi: 10.1016/j.compstruc.2021.106568.
[21] X. Sun, G. Wang, L. Xu, H. Yuan, and N. Yousefi, "Optimal performance of a combined heat-power system with a proton exchange membrane fuel cell using a developed marine predators algorithm," *Journal of Cleaner Production*, vol. 284, Feb. 2021, doi: 10.1016/j.jclepro.2020.124776.
[22] X. Chen, X. Qi, Z. Wang, C. Cui, B. Wu, and Y. Yang, "Fault diagnosis of rolling bearing using marine predators algorithm-based support vector machine and topology learning and out-of-sample embedding," *Measurement*, vol. 176, May 2021, doi: 10.1016/j.measurement.2021.109116.
[23] S. Kaur, L. K. Awasthi, A. L. Sangal, and G. Dhiman, "Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization," *Engineering Applications of Artificial Intelligence*, vol. 90, Apr. 2020, doi: 10.1016/j.engappai.2020.103541.
[24] M. Noroozi, H. Mohammadi, E. Efatinasab, A. Lashgari, M. Eslami, and B. Khan, "Golden search optimization algorithm," *IEEE Access*, vol. 10, pp. 37515–37532, 2022, doi: 10.1109/ACCESS.2022.3162853.
[25] P. Trojovský and M. Dehghani, "A new optimization algorithm based on mimicking the voting process for leader selection," *PeerJ Computer Science*, vol. 8, May 2022, doi: 10.7717/peerj-cs.976.
[26] M. Dehghani and P. Trojovský, "Hybrid leader based optimization: a new stochastic optimization algorithm for solving optimization applications," *Scientific Reports*, vol. 12, no. 1, Apr. 2022, doi: 10.1038/s41598-022-09514-0.
[27] M. Dehghani, S. Hubalovsky, and P. Trojovsky, "Northern goshawk optimization: a new swarm-based algorithm for solving optimization problems," *IEEE Access*, vol. 9, pp. 162059–162080, 2021, doi: 10.1109/ACCESS.2021.3133286.
[28] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," *Soft Computing*, vol. 23, no. 3, pp. 715–734, Feb. 2019, doi: 10.1007/s00500-018-3102-4.
[29] S. Suyanto, A. A. Ariyanto, and A. F. Ariyanto, "Komodo mlipir algorithm," *Applied Soft Computing*, vol. 114, Jan. 2022, doi: 10.1016/j.asoc.2021.108043.
[30] F. Zeidabadi, S. Doumari, M. Dehghani, and O. Malik, "MLBO: Mixed leader based optimizer for solving optimization problems," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 4, pp. 472–479, Aug. 2021, doi: 10.22266/ijies2021.0831.41.
[31] M. Dehghani *et al.*, "MLO: Multi leader optimizer," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 6, pp. 364–373, Dec. 2020, doi: 10.22266/ijies2020.1231.32.
[32] P. Trojovský and M. Dehghani, "Pelican optimization algorithm: a novel nature-inspired algorithm for engineering applications," *Sensors*, vol. 22, no. 3, Jan. 2022, doi: 10.3390/s22030855.
[33] O. Altay, "Chaotic slime mould optimization algorithm for global optimization," *Artificial Intelligence Review*, vol. 55, no. 5, pp. 3979–4040, Jun. 2022, doi: 10.1007/s10462-021-10100-5.
[34] P. D. Kusuma and M. Kallista, "Stochastic komodo algorithm," *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 4, Aug. 2022, doi: 10.22266/ijies2022.0831.15.
[35] S. A. Yasear and H. M. Ghanimi, "A modified honey badger algorithm for solving optimal power flow optimization problem," *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 4, Aug. 2022, doi: 10.22266/ijies2022.0831.14.
[36] P. D. Kusuma and A. Dinimaharawati, "Hybrid pelican Komodo algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 6, 2022, doi: 10.14569/IJACSA.2022.0130607.
[37] A. Kuznetsov, L. Wieclaw, N. Poluyanenko, L. Hamera, S. Kandiy, and Y. Lohachova, "Optimization of a simulated annealing algorithm for s-boxes generating," *Sensors*, vol. 22, no. 16, Aug. 2022, doi: 10.3390/s22166073.
[38] M. A. Noman, M. Alatefi, A. M. Al-Ahmari, and T. Ali, "Tabu search algorithm based on lower bound and exact algorithm solutions for minimizing the makespan in non-identical parallel machines scheduling," *Mathematical Problems in Engineering*, pp. 1–9, Dec. 2021, doi: 10.1155/2021/1856734.
[39] P. Lou, Y. Chen, and S. Gao, "Adaptive variable neighborhood search-based supply network reconfiguration for robustness enhancement," *Complexity*, pp. 1–21, Dec. 2020, doi: 10.1155/2020/1292938.
[40] F. Zhao, S. Du, H. Lu, W. Ma, and H. Song, "A hybrid self-adaptive invasive weed algorithm with differential evolution," *Connection Science*, vol. 33, no. 4, pp. 929–953, Oct. 2021, doi: 10.1080/09540091.2021.1917517.
[41] L. Zhang, Y. Yu, Y. Luo, and S. Zhang, "Improved cuckoo search algorithm and its application to permutation flow shop scheduling problem," *Journal of Algorithms and Computational Technology*, vol. 14, Jan. 2020, doi: 10.1177/1748302620962403.
[42] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine Predators algorithm: A nature-inspired metaheuristic," *Expert Systems with Applications*, vol. 152, p. 113377, Aug. 2020, doi: 10.1016/j.eswa.2020.113377.
[43] D. Freitas, L. G. Lopes, and F. Morgado-Dias, "Particle swarm optimisation: a historical review up to the current developments," *Entropy*, vol. 22, no. 3, Mar. 2020, doi: 10.3390/e22030362.
[44] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.
[45] M. Dehghani, Z. Montazeri, O. Malik, H. Givi, and J. Guerrero, "Shell game optimization: a novel game-based algorithm," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 3, pp. 246–255, Jun. 2020, doi: 10.22266/ijies2020.0630.23.
[46] A. Kumar, D. Kumar, and S. K. Jarial, "A review on artificial bee colony algorithms and their applications to data clustering," *Cybernetics and Information Technologies*, vol. 17, no. 3, pp. 3–28, Sep. 2017, doi: 10.1515/cait-2017-0027.
[47] M. Dubey, V. Kumar, M. Kaur, and T.-P. Dao, "A systematic review on harmony search algorithm: theory, literature, and applications," *Mathematical Problems in Engineering*, pp. 1–22, Apr. 2021, doi: 10.1155/2021/5594267.

BIOGRAPHIES OF AUTHORS

Purba Daru Kusuma received his bachelor, and master's degrees in electrical engineering from Bandung Institute of Technology, Indonesia, in 2002 and 2009 respectively. He received his doctoral degree in computer science from Gadjah Mada University, Indonesia, in 2017. Currently, he is an assistant professor in Telkom University, Indonesia. His research area is artificial intelligence, machine learning, and operational research. He can be contacted by email: purbodaru@telkomuniversity.ac.id.

Faisal Candrasyah Hasibuan is a Computer Engineering lecturer at Telkom University, Bandung, West Java, Indonesia. Status as an alma mater, he completed a bachelor's program at the place where he works now. He was granted a master's degree in electrical engineering with a Computer Engineering specialization at Bandung Institute of Technology (ITB). The field of research interests are but are not limited to the embedded system, internet of things, and intelligent control system. He can be contacted by email: faicanhasfcb@telkomuniversity.ac.id.