Mathematical Modelling of Engineering Problems
Vol. 12, No. 3, March, 2025, pp. 1021-1030
Journal homepage: http://iieta.org/journals/mmep

Fast Slow Optimization: An Adaptive Stochastic Optimization that Employs Both Short and Long Step Size Searches

Purba Daru Kusuma

Department of Computer Engineering, Telkom University, Bandung 40258, Indonesia

Corresponding Author Email: purbodaru@telkomuniversity.ac.id

https://doi.org/10.18280/mmep.120329

Received: 2 January 2025
Revised: 15 February 2025
Accepted: 20 February 2025
Available online: 31 March 2025

Keywords:
optimization, metaheuristic, engineering problem, stochastic, swarm intelligence, adaptability

ABSTRACT

This paper introduces a new metaheuristic called fast slow optimization (FSO). It is enhanced with an adaptive technique. It behaves more exploitatively after facing improvement while becoming more explorative after facing stagnation. It contains two sequential stages in every iteration where the fast movement is conducted in the first stage and slow movement is conducted in the second stage. There are two options in every stage where the first option represents exploitative action, and the second option represents explorative action. In this paper, FSO is confronted with five other metaheuristics: hiking optimization (HO), golden search optimization (GSO), tailor optimization algorithm (TOA), osprey optimization algorithm (OOA), and orangutan optimization algorithm (OOA) to solve two use cases: 23 standard functions and four engineering design problems. The result shows supremacy of FSO in solving 23 functions as it is better than HO, GSO, TOA, OOA, and OOA in 23, 22, 15, 12, and 12 functions. Meanwhile, FSO is competitive in handling the four engineering design problems.

1. INTRODUCTION

Optimization problems are commonly found in engineering fields. Some optimization problems in power systems are economic load dispatch (ELD) problem [1], economic emission dispatch (EED) problem [2], unit commitment (UC) problem [3], power stability problem [4], and so on. Some optimization problems in manufacturing systems are flow shop scheduling problem [5], job shop scheduling problem [6], batch scheduling problem [7], and so on. Vehicle routing problem is a popular optimization problem with various derivatives, such as multi depot vehicle routing problem [8], capacitated vehicle routing problem [9], and pickup and delivery problem [10].

Objective functions are common in optimization problems. Some problems have a single objective while other problems have multiple objectives. ELD is an example of a single objective problem where its objective is minimizing the operational or fuel cost [1]. Meanwhile, EED is a multi-objective problem where its objective is minimizing the fuel cost and emission cost [2].

Constraint is a common aspect in optimization problems. Constraint limits the possible or available solutions that can be picked during the optimization process. Some constraints are equality constraints while some others are inequality constraints. The example of equality constraint in ELD problem is that the total power output should be equal to the power demand [1]. Meanwhile, the example of inequality constraint in capacitated vehicle routing problem is that the quantity of the load may not exceed the vehicle capacity [9], or the travel time of each vehicle may not exceed certain time [10]. In some cases, a penalty is applied when the constraint is violated.

There are a lot of new metaheuristics were introduced in recent years. Many of them are metaphor-inspired metaheuristics. On the other hand, a few of them are metaphor-free metaheuristics. Some of these metaphor-inspired metaheuristics are HO [11], TOA [12], orangutan optimization algorithm (OOA) [13], osprey optimization algorithm (OOA) [14], giant armadillo optimization (GAO) [15], fennec fox optimization (FFO) [16], crayfish optimization algorithm (COA) [17], prairie dog optimization (PDO) [18], Komodo mlipir algorithm (KMA) [19], marine predator algorithm (MPA) [20], swarm magnetic optimizer (SMO) [21], horse herd optimization (HHO) [22], and so on. Meanwhile, the example of metaphor-free metaheuristics is quad tournament optimization (QTO) [23], GSO [24], fully informed search algorithm (FISA) [25], subtraction average based optimization (SABO) [26], average subtraction-based optimization (ASBO) [27], and so on.

Despite the massive development of metaheuristics and the implementation of metaheuristics in vast and various engineering optimization problems, the challenges and unresolved problems remain exist. The existence of non-free-lunch (NFL) theory states that there is not any absolute supreme technique that can handle all problems superiorly. Meanwhile, the adaptive approach is less considered in many techniques as they focus on exploration and exploitation. Whereas an adaptive approach is important so that the technique can take more appropriate action when it faces

improvement and stagnation. Metaphors are also criticized in the development of metaheuristics. The existence of metaphors is often blamed as the hiding mechanism of mere or trivial novelty.

Based on these unresolved problems and challenges, this work is aimed at introducing a new technique called FSO. FSO is constructed based on swarm intelligence so that it contains a certain number of autonomous agents that are active in improving the solution autonomously in every iteration. FSO also employs an adaptive approach so that it behaves differently when it faces improvement and stagnation. The terms fast and slow means FSO conducts both fast movement (long step size) and slow movement (short step size).

Based on its explanation, below are the scientific contributions of FSO.

- This paper introduces a new metaphor-free called FSO.
- A new adaptive approach is employed in FSO.
- FSO employs both fast movement and slow movement in every iteration.
- The performance of FSO is assessed to challenge it to solve 23 standard functions representing the unconstrained problems and four engineering design problems representing the constrained problems.

2. RELATED WORKS

Exploration and exploitation are two standard actions in metaheuristic techniques. There are some definitions regarding exploitation and exploration. Exploitation can be defined as searching near the recent solution or location [14]. On the other hand, exploration means searching for a broader area within the solution space [14]. The objective of exploitation is to enhance the quality of the recent solution. On the other hand, the objective of exploration is avoiding from being entrapped in the recent solution.

Both actions and activities are important for any technique as they employ stochastic optimization. It means there is not any guarantee that the optimization process will end with finding the actual optimal solution. Moreover, it is not guaranteed that the current search will provide improvement. For this reason, it makes sense that searching near the current location may provide better opportunities for improvement rather than searching randomly within space without any clues. It also makes sense that moving toward the location or solution whose quality is better will provide better opportunity for improvement. This activity can be called exploitation. Based on the previous statement, a better opportunity for improvement is not equal to a guarantee for improvement. On the other hand, this activity may push the optimization process into the area known as the local optimal [14]. This circumstance makes exploration become rational as the optimization process tries to find other solutions within space that is far from the current solution or against the trend. Exploration becomes more important especially when the optimization process fails to improve for certain trials.

The genetic algorithm (GA) is an example of classic or old technique that employs both exploitation and exploration in a clear separation. The exploitation is conducted through crossover while exploration is conducted through mutation [28]. During crossover, some individuals exchange their solutions to create new solutions called offsprings. It means that the value of the offsprings is not far from their parents. Meanwhile, certain rules are applied to determine which individuals are selected as parents. In many cases, individuals whose quality is high have a better chance of being selected as parents. But it does not guarantee that high quality individuals will be selected as parents. Low quality individuals still have this chance although the probability is low to give diversity. Meanwhile, through mutation, a fraction of the solution is randomized within space as exploration.

Simulated annealing (SA) is an example of an old technique that focuses on exploitation. SA in a single entity metaheuristic that conducts neighborhood search. As is known in neighborhood search, this search is slow movement search to maintain accuracy. The consequence of this strategy is that high maximum iteration is needed especially when the solution space is vast. Exploration is accommodated by employing a conditional acceptance rule. A better solution is accepted immediately. On the other hand, a worse solution still can be accepted. But this opportunity decreases as the iteration goes on.

KMA is an example of a new technique that is dominated by exploitation. As swarm-based metaheuristic, KMA consists of a population of agents. In every iteration, this swarm is split into three groups: high quality agents, moderate quality agents, and low-quality agents [19]. For the first group, an agent tends to move toward other agents (exploitation) but still can avoid them (exploration) so that it is called exploitation dominance. For the second group, there is a 50 percent probability that an agent performs crossover with the highest quality agent (exploitation) and 50 percent probability to perform full random search (exploration). It means that there is a balance between exploitation and exploration for the second group. For the third group, all agents move toward the resultant of high-quality agents. It means that full exploitation is employed in the third group.

Zebra optimization algorithm (ZOA) is an example of a new technique that performs balance exploration-exploitation. The directed search is conducted in the first stage [29]. There are two options that will be selected stochastically with equal opportunity. The first option is the movement toward the best agent as it represents exploitation. The second option is the movement relative to a randomly chosen agent where the direction is determined based on the relative quality between the agent and the randomly chosen agent. This second option represents exploration. The neighborhood search is conducted in the second stage [29]. But this neighborhood search cannot be called fully exploitative. In the early iteration, the local space is wide so that it can be seen as exploration. Then the local space size narrower as iteration goes on. It means that in the late iteration, it can be seen as exploitation.

Coati optimization algorithm (COA) is also an example of a new technique that performs a balance exploration-exploitation. It also consists of two stages in every iteration. Unlike ZOA, that employs stochastic decision-making mechanism in the first stage, COA employs this balancing strategy through swarm split. Half of the swarm performs the movement toward the best agent [30], and it represents exploitation. On the other hand, half of the swarm performs the movement relative to a target [30] that is generated within the space [30], and it represents exploration.

One big problem in metaphor-inspired techniques is that they often hide their actual methods and promote the imitation of the behavior of their metaphors as a novel approach. For example, the splitting of population into three groups in KMA

[19] (big males, females, and small males) is splitting the population into three groups based on the quality of the individuals. Meanwhile, the motion toward the best agent has different terms in COA and ZOA. In COA, this motion is called the coati hunts the iguana that is still on the tree [30]. In ZOA, this motion is called the zebras follows the pioneer zebra during foraging [29]. It means that both iguana on the tree and pioneer zebra refer to the same entity which is the best agent. Meanwhile, the local search with reduced space during iteration has multiple names. In COA, it is called escaping from the predators [30]. In ZOA, it is called zebra escapes from the lion [29]. In OOA, it is called bringing the fish to the appropriate location [14].

Meanwhile, the metaphor-free technique faces difficulty in giving appropriate and easy listening terms for its mechanism. For example, GSO is constructed based on the directed search toward the local best and the global best [24]. This search is like the MPA [20]. But the step size calculation is different. While MPA employs levy movement and Brownian motion [20], GSO employs sinusoid distribution [24]. In MPA, the mechanism is called predator catching the prey. This mechanism is also found in COA [17]. In COA, the mechanism is called cave which is the middle between the local best and the global best. In this circumstance, the metaphor-inspired techniques seem better but on the other hand, the metaphor-free techniques provide fair and clear explanation.

Based on this explanation, there are a lot of mechanisms to accommodate both exploitation and exploration. Some metaheuristics may be exploration dominant while some others are exploitation dominant. Some techniques may try to balance exploitation and exploration. These mechanisms are developed based on static setting, for example swarm split, iteration, or relative quality within the swarm. Unfortunately, many of these mechanisms do not consider the improvement status of the process. In other words, the metaheuristics that determine their action, exploration or exploitation, based on the improvement status is rare to find. This circumstance becomes the motivation of this work to construct a new technique that employs adaptive behavior where exploration or exploitation is determined based on the improvement status.

3. MODEL

FSO is constructed based on the combination of fast motion and slow motion. The fast motion is interpreted as the agent moves with long step size while the slow motion is interpreted as the agent moves with short step size. Fast motion is designed to move toward better solutions as fast as possible. On the other hand, slow motion is designed to take accurate motion.

FSO is also designed as an adaptive technique. In other words, an adaptive approach is employed to FSO so that it takes different actions when it faces improvement or stagnation. Improvement means the agent can find a better solution in the previous movement. On the other hand, stagnation means the agent fails to find a better solution in the previous movement. When it faces improvement, agents tend to be more exploitative. On the other hand, agents tend to be more explorative when it faces stagnation.

Each agent performs two stages in every iteration. The first stage is for fast motion while the second stage is for slow motion. Meanwhile, there are two options that can be chosen in every stage. The first option is the exploitative movement while the second option is the exploration movement. The action in the first stage is taken based on the improvement status in the second stage of the previous iteration. Meanwhile, the action in the second stage is taken based on the improvement status in the first stage of the same iteration.

There are two options in the first stage. The first option is the combination of the motion toward the best agent and the motion toward a randomly chosen better agent. This first option represents the exploitative movement of the first stage. The second option is the combination of the motion toward two randomly chosen better agents. This second option represents the exploration of the first stage.

There are two options in the second stage. The first option is motion toward the best agent. This first option represents the exploitative movement of the second stage. The second option is motion relative to a randomly chosen agent. This second option represents the explorative movement of the second stage.

A stringent acceptance approach is applied in FSO. This approach is applied in both stages. It means that the candidate in every stage can replace the current value of the agent only if this candidate offers improvement.

The formalization of FSO is provided in both algorithm and mathematical formulation. The algorithm is provided in pseudocode as it is presented in algorithm 1 to algorithm 4. Meanwhile, the mathematical formulation of FSO is presented in Eq. (1) to Eq. (13). The notations that are used in this paper are provided in nomenclature.

There are four algorithms presented in this paper. Algorithm 1 provides the general process of FSO. Algorithm 2 provides the process during initialization. Algorithm 3 provides the process during the first stage. Algorithm 4 provides the process during the second stage.

FSO consists of two phases as shown in algorithm 1. The first phase is the initialization as presented in algorithm 1 from line 2 to line 4. The second phase is the iteration as presented in algorithm 1 from line 5 to line 10. The best agent becomes the final solution as presented in line 11.

The process during initialization of each agent is formalized using algorithm 2. It consists of three processes. The first process is the construction of the initial value of an agent using Eq. (1) where this initial solution or location is distributed uniformly within space. Then. The updating of the best agent is conducted by comparing the best agent with the related agent as formulated using Eq. (2). The third process is setting the initial status of the agent to 1 using Eq. (3).

$$a_{i,j} = b_{lo,j} + r_1(b_{hi,j} - b_{lo,j}) \tag{1}$$

$$a'_{best} = \begin{cases} a_i, o(a_i) < o(a_{best}) \\ a_{best}, else \end{cases} \tag{2}$$

$$s_i = 1 \tag{3}$$

The process in the first stage is formalized using algorithm 3. It consists of six processes. The first process is the construction of the set consisting of all better agents plus the best agent using Eq. (4). The second process is selecting a randomly chosen better agent using Eq. (5). The third process is generating the first candidate using Eq. (10). The fourth process is updating the status using Eq. (12). The fifth process is updating the agent based on stringent acceptance using Eq. (13). The sixth process is updating the best agent using Eq. (2).

$$A_{bet,i} = \{\forall a_k \in A, o(a_k) < o(a_i)\} \cup a_{best} \quad (4)$$

$$a_{s1,i} = r_p(A_{bet}) \quad (5)$$

$$a_{s2,i} = r_p(A) \quad (6)$$

$$m_{1,i,j} = a_{best,j} - r_2 a_{i,j} \quad (7)$$

$$m_{2,i,j} = a_{s1,i,j} - r_2 a_{i,j} \quad (8)$$

$$m_{3,i,j} = \begin{cases} a_{s2,i,j} - r_2 a_{i,j}, o(a_{s2,i}) < o(a_i) \\ a_{i,j} - a_{s2,i,j}, else \end{cases} \quad (9)$$

$$c_{1,i,j} = \begin{cases} a_{i,j} + r_1 m_{1,i,j} + r_1 m_{2,i,j}, s_i = 1 \\ a_{i,j} + r_1 m_{2,i,j} + r_1 m'_{2,i,j}, else \end{cases} \quad (10)$$

$$c_{2,i,j} = \begin{cases} a_{i,j} + r_1 m_{1,i,j}, s_i = 1 \\ a_{i,j} + r_1 m_{3,i,j}, else \end{cases} \quad (11)$$

$$s_i = \begin{cases} 1, o(c_i) < o(s_i) \\ 0, else \end{cases} \quad (12)$$

$$a'_i = \begin{cases} c_i, o(c_i) < o(a_i) \\ a_i, else \end{cases} \quad (13)$$

The process in the second stage is formalized using algorithm 4. It consists of five processes. The first process is selecting a randomly chosen agent using Eq. (6). The second process is generating the second candidate using Eq. (11). The third process is updating the status using Eq. (12). The fourth process is updating the agent based on stringent acceptance using Eq. (13). The fifth process is updating the best agent using Eq. (2).

Based on this explanation, the balancing between exploitation and exploration in FSO is conducted in several mechanisms. The first stage represents exploration as it conducts fast motion while the second stage represents exploitation as it conducted slow motion. The step size of the first stage tends to double than the second stage. Meanwhile, the first option in each stage represents exploitation while the second option represents exploration based on the target or reference. In this context, the difference between exploitation and exploration at the stage level is based on the step size. On the other hand, the difference between exploitation and exploration at the option level is based on the target. Balancing in the stage level is designed to handle the trade-off between speed and accuracy. On the other hand, balancing in the option level is designed to handle the trade-off between depth and broad regarding the circumstance, whether improvement or stagnation. This approach makes FSO different from other techniques.

Algorithm 1: General process of FSO

```
start
for i=1 to n
initialize a_i (see algorithm 2)
end for
for t=1 to t_max
for i=1 to n
perform first stage (see algorithm 3)
perform second stage (see algorithm 4)
end for
end for
return a_best
stop
```

Algorithm 2: Initialization of agent

```
start
set initial value of a_i using Eq. (1)
update a_best using Eq. (2)
set s_i using Eq. (3)
stop
```

Algorithm 3: First stage

```
start
construct A_bet using Eq. (4)
select a_s1 using Eq. (5)
generate c_1 using Eq. (10)
update s_i using Eq. (12)
update a_i using Eq. (13)
update a_best using Eq. (2)
stop
```

Algorithm 4: Second stage

```
start
select a_s1 using Eq. (6)
generate c_1 using Eq. (11)
update s_i using Eq. (12)
update a_i using Eq. (13)
update a_best using Eq. (2)
stop
```

4. EXPERIMENT AND RESULT

This section presents an experiment that is conducted to assess the performance of FSO. There are two use cases in this paper. The first use case is the set of 23 functions. The first use case represents an unconstrained problem. The second use case is a set of four engineering design problems. This second use case represents the constrained problem.

Five new metaheuristics are chosen as benchmarks. These five techniques include HO [11], GSO [24], TOA [12], OOA [14], and OOA [13]. The first two techniques represent metaheuristics that employ stringent acceptance rule. On the other hand, the last three techniques represent metaheuristics that do not employ stringent acceptance rule. In both cases, the setting for all metaheuristics including FSO and its benchmarks is the same. The swarm size is set to 5. The maximum iteration is set to 20.

In this paper, all benchmarks are new techniques. Meanwhile, some popular classic techniques such as GA, PSO, grey wolf optimization (GWO), or SA are excluded from the benchmarks. There are two reasons due to this decision. First, many recent studies that introduce new techniques assessed the performance of their proposed technique with the new existing techniques to ensure the advancement. Second, in many studies, these old techniques have been beaten many times with significant gaps.

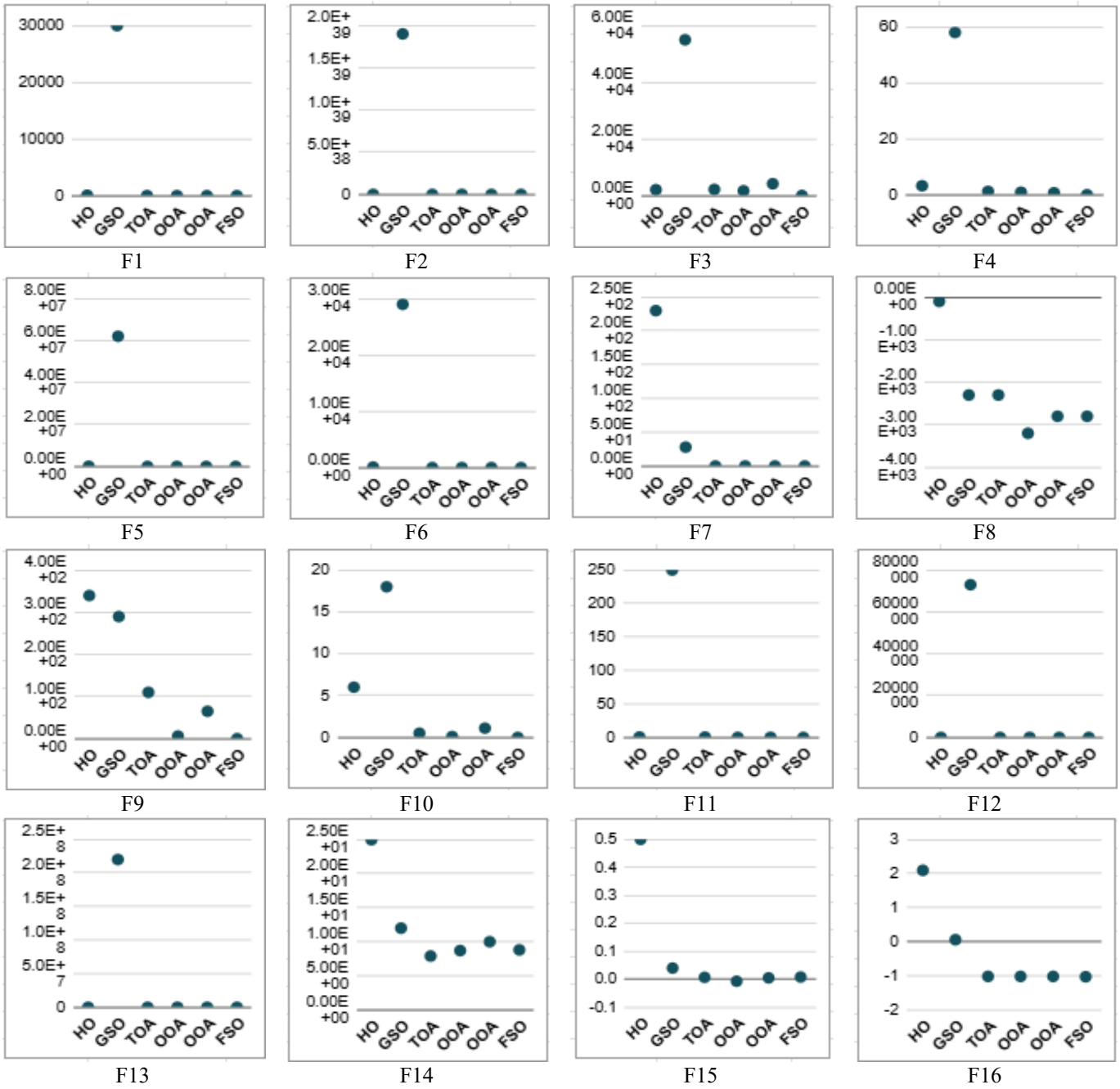
The set of 23 functions is chosen due to its coverage and its popularity. It consists of 13 high dimension functions where the dimension of the problem can be only one or expanded to thousands. Among these high dimension functions, the first seven functions are unimodal where each function contains only one optimal solution, and the last six functions are multimodal where each function contains multiple optimal solutions. The first seven functions are designed to assess exploitation capability while the last six functions are designed to assess the exploration capability. This set also contains ten fixed dimension functions. All these ten functions are

multimodal but more ambiguous based on their terrain and steep slopes or holes. These functions are designed to assess the balancing capability between exploitation and exploration. The detailed description of this set of 23 functions can be found in references [24, 29].

The result of the assessment of the first use case is presented in Tables 1-4. Table 1 to Table 3 present the raw result that contains the average score, standard deviation, and the mean rank. Moreover, Table 4 provides the summary result representing the supremacy of FSO compared to its benchmarks. This summary is clustered based on the group of functions. The illustration of the average score is also presented in Figure 1.

Table 1 shows the supremacy of FSO compared to its benchmarks in solving high dimension unimodal functions. FSO performs the best of all seven functions (F1 to F7). There are four techniques that achieve the best result in solving F2 which are TOA [12], OOA [14], OOA [13], and FSO. All these techniques employ stringent acceptance. GSO becomes the worst technique while HO becomes the second worst technique. GSO [24] performs the worst result in six functions (F1 to F6) while HO [11] performs the second worst in these six functions. The disparity between the best result and the worst result is wide in all seven functions.

Table 2 shows that FSO is still superior in solving high dimension multimodal functions. FSO provides the best result in solving five functions (F9 to F13). FSO provides the third-best result in solving F8. GSO provides the worst result in four functions (F10 to F13). Meanwhile, HO provides the worst result in solving two functions (F8 and F9). The disparity between the best technique and worst technique is wide in five functions (F9 to F13). On the other hand, this disparity is narrow or not significant in solving F8.



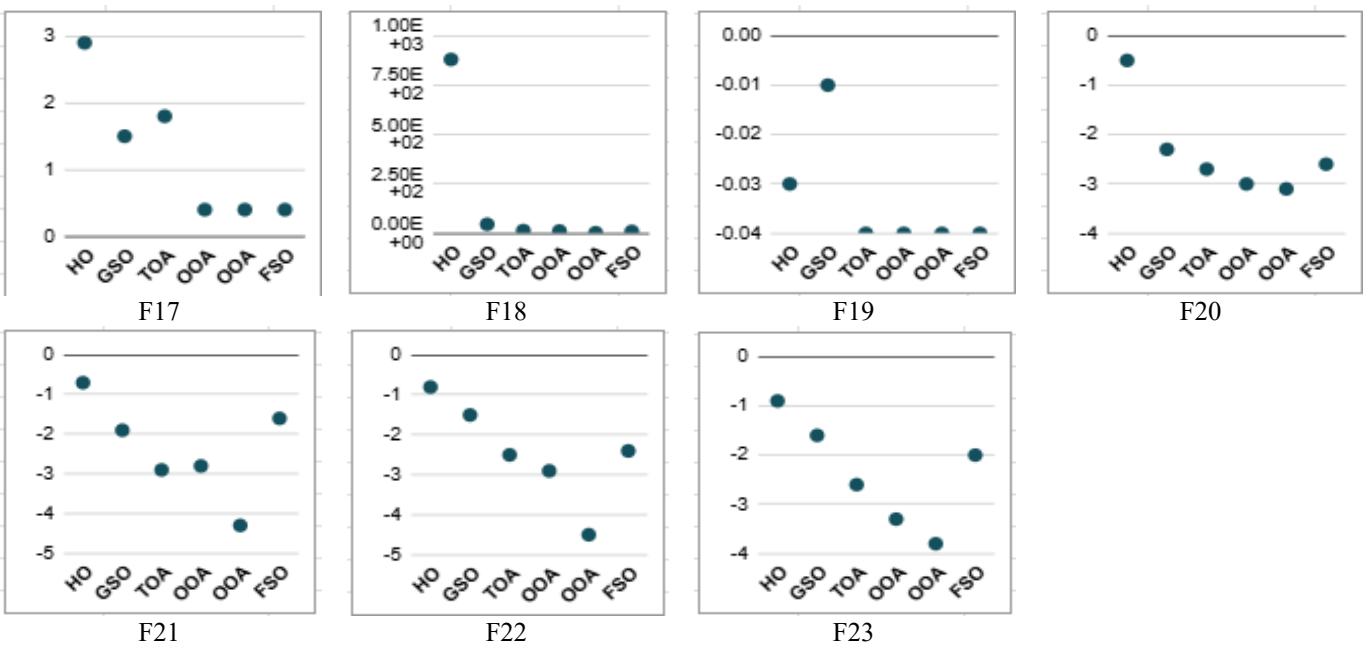


Figure 1. Visualization of average score on handling 23 functions

Table 1. Result on handling the seven high dimension unimodal functions

F	Parameter	HO [11]	GSO [24]	TOA [12]	OOA [14]	OOA [13]	FSO
	mean	6.7067×10^{1}	3.0956×10^{4}	2.2045	0.2264	0.3089	0.0000
1	std. deviation	5.0243×10^{1}	7.4281×10^{3}	1.3525	0.2239	0.2672	0.0000
	mean rank	5	6	4	2	3	1
	mean	3.1214×10^{1}	1.9480×10^{39}	0.0000	0.0000	0.0000	0.0000
2	std. deviation	1.3941×10^{2}	5.3946×10^{39}	0.0000	0.0000	0.0000	0.0000
	mean rank	5	6	1	1	1	1
	mean	2.1581×10^{3}	5.5785×10^{4}	2.2421×10^{3}	1.7030×10^{3}	4.2801×10^{3}	0.0317
3	std. deviation	3.1886×10^{3}	1.9445×10^{4}	2.5574×10^{3}	1.8712×10^{3}	4.6606×10^{3}	0.0701
	mean rank	3	6	4	2	5	1
	mean	3.2024	5.8114×10^{1}	1.2317	0.9497	0.7425	0.0003
4	std. deviation	1.1170	7.9856	0.5935	0.4730	0.4874	0.0001
	mean rank	5	6	4	3	2	1
	mean	7.1969×10^{4}	6.2250×10^{7}	5.7483×10^{1}	3.2537×10^{1}	3.3391×10^{1}	2.8958×10^{1}
5	std. deviation	7.1311×10^{4}	2.9263×10^{7}	2.1245×10^{1}	4.8084	3.8406	0.0219
	mean rank	5	6	4	2	3	1
	mean	1.0333×10^{2}	2.9838×10^{4}	9.3620	6.1358	6.4017	6.4269
6	std. deviation	6.5143×10^{1}	7.0270×10^{3}	3.4739	0.5216	0.6620	0.4435
	mean rank	5	6	4	1	2	3
	mean	2.3896×10^{2}	2.8586×10^{1}	0.0382	0.0341	0.0497	0.0139
7	std. deviation	1.2022×10^{2}	1.3990×10^{1}	0.0177	0.0191	0.0387	0.0090
	mean rank	6	5	3	2	4	1

Table 2. Result on handling the six high dimension multimodal functions

F	Parameter	HO [11]	GSO [24]	TOA [12]	OOA [14]	OOA [13]	FSO
	mean	-9.7116×10^{1}	-2.3663×10^{3}	-2.3485×10^{3}	-3.2548×10^{3}	-2.8741×10^{3}	-2.4280×10^{3}
8	std. deviation	1.9157×10^{1}	6.2863×10^{2}	6.2800×10^{2}	5.0664×10^{2}	3.1792×10^{2}	5.0390×10^{2}
	mean rank	6	4	5	1	2	3
	std. deviation	3.4396×10^{2}	2.9051×10^{2}	1.1579×10^{2}	6.2866	6.5429×10^{1}	3.2295×10^{1}
9	range	3.8061×10^{1}	4.2971×10^{1}	6.4522×10^{1}	9.8475	8.4568×10^{1}	6.2440×10^{1}
	mean rank	6	5	4	1	3	2
	mean	6.0209	1.8791×10^{1}	0.5285	0.1124	1.1740	0.0000
10	std. deviation	0.9102	0.8264	0.2726	0.0419	4.2713	0.0000
	mean rank	5	6	3	2	4	1
	mean	0.6340	2.5806×10^{2}	0.5397	0.1375	0.1657	0.0265
11	std. deviation	0.2743	8.1853×10^{1}	0.2416	0.1880	0.1943	0.0524
	mean rank	5	6	4	2	3	1
	mean	6.6508	7.3570×10^{7}	1.3137	1.0278	0.9762	1.1564
12	std. deviation	2.6183	4.8881×10^{7}	0.2403	0.2589	0.2699	0.2484
	mean rank	5	6	4	2	1	3
	mean	4.5890	2.2624×10^{8}	3.8484	3.3123	3.4053	3.1358
13	std. deviation	2.7921	1.2377×10^{8}	0.3737	0.2001	0.2233	0.1360
	mean rank	5	6	4	2	3	1

Table 3. Result on handling the ten fixed dimension multimodal functions

F	Parameter	HO [11]	GSO [24]	TOA [12]	OOA [14]	OOA [13]	FSO
	mean	2.5435×10^{1}	1.2547×10^{1}	7.9703	8.7213	1.0218×10^{1}	1.0583×10^{1}
14	std. deviation	4.7701×10^{1}	7.5718	3.8744	4.4183	3.9105	3.6507
	mean rank	6	5	1	2	3	4
	std. deviation	0.5175	0.0437	0.0074	0.0077	0.0052	0.0154
15	range	0.8961	0.0307	0.0105	0.0157	0.0058	0.0311
	mean rank	6	5	2	3	1	4
	mean	2.1009	0.0667	-1.0263	-1.0281	-1.0239	-1.0164
16	std. deviation	3.4114	3.8215	0.0132	0.0054	0.0076	0.0185
	mean rank	6	5	2	1	3	4
	mean	2.9546	1.5031	1.8591	0.4003	0.4052	1.8957
17	std. deviation	2.6251	3.2795	3.7882	0.0029	0.0087	3.2231
	mean rank	6	3	4	1	2	5
	mean	8.8197×10^{2}	4.6057×10^{1}	1.4926×10^{1}	1.2785×10^{1}	3.2036	2.5756×10^{1}
18	std. deviation	9.5148×10^{2}	1.5648×10^{2}	2.0776×10^{1}	2.3371×10^{1}	0.2914	2.7426×10^{1}
	mean rank	6	5	3	2	1	4
	mean	-0.0356	-0.0116	-0.0495	-0.0495	-0.0495	-0.0495
19	std. deviation	0.0178	0.0142	0.0000	0.0000	0.0000	0.0000
	mean rank	6	5	1	1	1	1
	std. deviation	-0.5893	-2.3039	-2.7243	-3.0198	-3.1278	-2.7834
20	range	0.6830	0.6419	0.3777	0.1773	0.0863	1.1504
	mean rank	6	5	4	2	1	3
	mean	-0.7009	-1.9127	-2.9500	-2.8517	-4.3461	-2.7834
21	std. deviation	0.4990	1.5459	1.5347	1.3631	1.0897	1.1504
	mean rank	6	5	2	3	1	4
	mean	-0.8014	-1.5237	-2.5544	-2.9293	-4.5219	-3.0006
22	std. deviation	0.3902	0.7377	1.2049	1.2045	1.6423	1.4554
	mean rank	6	5	4	3	1	2
	mean	-0.9625	-1.6093	-2.6118	-3.3261	-3.8362	-3.0161
23	std. deviation	0.5681	0.7129	1.4392	2.0399	1.4924	1.3360
	mean rank	6	5	4	2	1	3

Table 4. Supremacy of FSO among its benchmarks

Cluster	HO [11]	GSO [24]	TOA [12]	OOA [14]	OOA [13]
1	7	7	6	6	6
2	6	6	6	5	5
3	10	9	3	2	2
Total	23	22	15	15	15

Table 5. Number of constraints and dimension size

Design Problem	Number of Constraints	Dimension Size
pressure vessel	4	4
speed reducer	11	7
welded beam	7	4
spring	4	3

Table 3 shows that FSO is still competitive but not superior in solving ten fixed dimension multimodal functions. FSO is twice on the first rank (F16 and F19), once on the second rank (F18), twice on the third rank (F14 and F17), four times on the fourth rank (F14, F20, F22, and F23), and once on the fifth rank (F21). HO becomes the worst technique as it is on the sixth rank in all ten functions while GSO becomes the second worst technique as it is on the fifth rank also in eight functions. The disparity between the best and the worst techniques is narrow in eight functions (F14, F16, F17, F19 to F23) and moderate in two functions (F15 and F18). This result shows the tight competition among techniques in this group of functions.

Table 4 shows the supremacy summary of FSO compared to its benchmarks. FSO is absolute superior to HO as it is better than HO in all 23 functions. Meanwhile, FSO is superior to GSO in 22 functions. It means that FSO performs better than HO and GSO in all three group of functions, both unimodal and multimodal ones. Meanwhile, FSO is superior to TOA [12], OOA [14], and OOA [13] in solving high dimension functions. FSO is better than TOA [12], OOA [14], and OOA [13] in 12, 11, and 11 high dimension functions. Unfortunately, FSO is inferior to TOA, OOA, and OOA in solving these fixed dimension functions although the disparity is narrow.

The second use case is a set of four engineering design problems. These problems include pressure vessels, speed reducers, welded beam, and spring design problems. These four problems represent constrained optimization problems so that the solution can be put independently in space as the solution in certain dimensions depends on the solution in other dimensions. The number of constraints and dimensions of each design problem is provided in Table 5. Meanwhile, the detailed description of each problem including the objective function, space, and constraints can be found in many studies [29, 30].

The quadratic penalty function is provided to handle the constraint. The penalty is applied to the violation of the constraints. The value of the penalty increases quadratically to the linear increase of the constraint violation. It means the penalty will reduce the quality of the solution whenever the solution is created with constraint violation as collateral.

Table 6 shows the result of solving the pressure vessel design problem. FSO becomes the second-best technique while OOA [13] becomes the best technique. HO becomes the worst technique. The disparity between the best and worst techniques is wide. The disparity between FSO and OOA [13] is narrow while the disparity between FSO and HO is wide.

Table 6. Result in solving pressure vessel design problem

Metaheuristic	Mean	Rank
HO [11]	8.2960×10^{12}	6
GSO [24]	7.0617×10^{5}	5
TOA [12]	2.5932×10^{3}	4
OOA [14]	4.2352×10^{2}	3
OOA [13]	9.6887	1
FSO	9.7770	2

Table 7. Result in solving speed reducer design problem

Metaheuristic	Mean	Rank
HO [11]	3.5955×10^{3}	3
GSO [24]	3.7416×10^{3}	6
TOA [12]	3.5573×10^{3}	2
OOA [14]	3.6135×10^{3}	5
OOA [13]	3.5410×10^{3}	1
FSO	3.6086×10^{3}	4

Table 8. Result in solving welded beam design problem

Metaheuristic	Mean	Rank
HO [11]	3.4528×10^{11}	6
GSO [24]	9.9702×10^{9}	5
TOA [12]	1.9500×10^{9}	3
OOA [14]	2.8511×10^{9}	4
OOA [13]	1.2966×10^{8}	1
FSO	5.6165×10^{8}	2

Table 9. Result in solving tension/compression spring design problem

Metaheuristic	Mean	Rank
HO [11]	7.3642×10^{2}	6
GSO [24]	6.7858×10^{1}	5
TOA [12]	8.4084	3
OOA [14]	3.9753	1
OOA [13]	6.2541	2
FSO	1.4016×10^{1}	4

Table 7 shows the result of solving the speed reducer design problem. FSO becomes the fourth best while OOA [13] becomes the best technique. GSO becomes the worst technique. The disparity between OOA and GSO is narrow. It means the distance between FSO and OOA is narrow. Meanwhile the disparity between FSO and GSO is also narrow. This result also shows that the competition among techniques in solving the speed reducer design problem is tight.

Table 8 shows the result of solving the welded beam design problem. FSO becomes the second best after OOA [13] as the best. Meanwhile, HO becomes the worst technique. The disparity between OOA and HO is wide. The disparity between FSO and OOA is wide. Last, the disparity between FSO and HO is also wide. It means that FSO is in the middle between the best and worst techniques.

Table 9 shows the result of solving the spring design problem. FSO becomes the fourth best after OOA [14] as the best. HO becomes the worst technique. The disparity between OOA and HO is wide. The disparity among four metaheuristics (TOA [12], OOA [14], OOA [13], and FSO) is narrow.

5. DISCUSSION

Overall, the result shows that FSO is acceptable and competitive as a new metaheuristic. This competitiveness can be found in both cases, the set of 23 standard functions as the unconstrained problem and the set of 4 engineering design problems as the constrained problem. FSO can find the quasi-optimal solution to all problems.

The result in solving the 23 standard functions shows the accepted capability of FSO. FSO has superior exploitation capability due to its supremacy in the high dimension unimodal functions. FSO also has superior exploration capability due to its supremacy in the high dimension multimodal functions. Meanwhile, the capability of FSO in balancing the exploitation and exploration is competitive compared to TOA, OOA, and OOA; and superior to HO and GSO.

The superiority of FSO compared to its benchmarks in handling high dimension functions comes from its nature that employs directed search twice in full step size in every iteration. GSO and HO employ the directed search once as it they are single search techniques. OOA [14] and OOA [13] employ directed search once in their first stage. Meanwhile, TOA employs directed search twice, but the second search is conducted with very slow motion in the later iteration.

On the other hand, the less competitiveness of FSO compared to TOA [12], OOA [14], and OOA [13] in handling the fixed dimension functions comes from the step size during the iteration. In general, FSO performs full length step size in both searches, where the step size of the first stage is double than the second stage. On the other hand, TOA [12], OOA [14], and OOA [13] employes the declining step size during their second stage. It is shown that highly accurate motion is needed to overcome the fixed dimension functions. Fortunately, the disparity between FSO and these three techniques in overcoming the fixed dimension functions is narrow.

The result in solving the engineering design problems shows the accepted capability of FSO in handling constrained problems. FSO is still competitive although not superior in these four problems while the supremacy in these problems belongs to OOA. Fortunately, the disparity between FSO and the best technique is narrow in three design problems out of four design problems.

The result also reveals the no-free-lunch (NFL) theory. Despite its supremacy in handling high dimension functions, FSO is just competitive in handling the fixed dimension functions compared to TOA, OOA, and OOA. But it keeps its superiority compared to HO and GSO. OOA may be the best technique in handling the fixed dimension functions but only in seven functions.

In general, the stringent acceptance approach is better than the loose acceptance approach. As mentioned previously, the stringent acceptance approach protects the agents from the worse circumstances although the consequence is that they may be stuck in a certain solution for a certain time. But this circumstance is proven better than falling into the worse solution.

Still stated in the NFL theory, the performance of any techniques is affected not only by the construction of the technique but also the nature of the problems. As shown in the result, the high dimension functions are better handled with the multiple directed searches. On the other hand, the fixed dimension functions are better handled with techniques that are enhanced with the declining step size to very slow motion. The constrained problems make the disparity among techniques narrow as shown in many economic dispatch

studies [2]. The nature of the problems also affects the disparity among techniques. As the result shows, the disparity among techniques in the high dimension functions is generally wide while in the fixed dimension functions is generally narrow.

Despite its positive result, there are limitations in this paper. The first limitation comes from the nature of the FSO. There are a lot of approaches that can be used to construct a new technique whether reference, step size, swarm split, and the decision mechanism. Unfortunately, it is impossible to accommodate all these options into a single technique. On the other hand, each option always has strength and weakness. This limitation makes the development of a new technique is still challenging. Moreover, the optimization problems also evolve due to dynamism in the engineering sector. Second, FSO is assessed using standard unconstrained and constrained problems in this paper. On the other hand, there are also many more standard functions such as CEC series and it is impossible to test a technique using all cases in a single paper. This limitation encourages the assessment of FSO using more various practical problems in future studies.

The implementation of FSO to solve broader practical problems is also challenging. In the end, all techniques should be tested to solve the practical problems rather than only the standard problems. First, in general, the practical problems are constrained problems. For example, in the economic dispatch problem, the constraint is the matching between the demand and total power output. In the capacitated vehicle routing problem, the total capacity of the fleet limits the load or customers that can be handled. All these constraints arise because in the practical or real-world problems, optimization always faces resources that are limited. The next challenge is handling combinatorial problems, such as scheduling, allocation, or timetable problems. In these problems, the quantitative approach should be transformed into permutation.

6. CONCLUSIONS

This paper has presented the FSO as a new and adaptive metaheuristic. The presentation includes the concept, formalization, and assessment. The result shows that the performance of FSO is acceptable as it can find the quasi-optimal solution for the use cases. Compared to the benchmarks, FSO is also competitive in handling both the set of standard functions and the four classic engineering design problems. FSO is better than HO, GSO, TOA, OOA, and OOA in handling 23, 22, 15, 15, and 15 functions out of 23 functions where, FSO is superior in handling the high dimension functions. Meanwhile, FSO is still dominant in the fixed dimension functions when it is compared to HO and GSO.

Future studies can be performed in several ways due to this work. First, developing an adaptive approach to be embedded in any static metaheuristic techniques. The exploration of new adaptive approaches is still interesting and challenging. Second, implementing FSO to solve various practical optimization problems is also challenging, whether it is implemented in its basic form or the improved one. Third, implementing FSO as a metaheuristic to support the machine learning work to improve the quality of the net architecture is also important.

ACKNOWLEDGMENT

The author thanks Telkom University for providing the funding for the publication of this paper.

REFERENCES

[1] Zein, H., Raharjo, J., Mardiyanto, I.R. (2022). A method for completing economic load dispatch using the technique of narrowing down area. IEEE Access, 10: 30822-30831. https://doi.org/10.1109/ACCESS.2022.3158928

[2] Puspitasari, K.M.D., Raharjo, J., Sastrosubroto, A.S., Rahmat, B. (2022). Generator scheduling optimization involving emission to determine emission reduction costs. International Journal of Engineering, 35(8): 1468-1478. https://doi.org/10.5829/ije.2022.35.08b.02

[3] Abbas, A.I., Anwer, A. (2022). The optimal solution for unit commitment problem using binary hybrid grey wolf optimizer. International Journal of Electrical and Computer Engineering, 12(1): 122-130. https://doi.org/10.11591/ijece.v12i1.pp122-130

[4] Robandi, I., Syafarudin, Hasanah, R.N., Guntur, H.L., Lystianingrum, V., Djalal, M.R., Prakasa, M.A., Himawari, W. (2024). Stability improvement of Sulbagsel electricity system integrated wind power plant using SVC-PSS3C based on improved mayfly algorithm. Results in Engineering, 24: 103407. https://doi.org/10.1016/j.rineng.2024.103407

[5] Li, Y.Z., Gao, K., Meng, L.L., Jing, X.L., Zhang, B. (2023). Heuristics and metaheuristics to minimize makespan for flowshop with peak power consumption constraints. International Journal of Industrial Engineering Computations, 14(2): 221-238. https://doi.org/10.5267/j.ijiec.2023.2.004

[6] Yuan, S., Zhu, X., Cai, W., Gao, J., Zhang, R. (2025). Mathematical modeling and hybrid evolutionary algorithm to schedule flexible job shop with discrete operation sequence flexibility. Computers and Operations Research, 176: 106952. https://doi.org/10.1016/j.cor.2024.106952

[7] Chen, Y., Zhao, X., Mumtaz, J., Guangyuan, C., Wang, C. (2025). Batch processing machine scheduling problems using a self-adaptive approach based on dynamic programming. Computers and Operations Research, 176: 106933. https://doi.org/10.1016/j.cor.2024.106933

[8] Wei, X., Niu, C., Zhao, L., Wang, Y. (2025). Combination of ant colony and student psychology based optimization for the multi-depot electric vehicle routing problem with time windows. Cluster Computing, 28(2): 99. https://doi.org/10.1007/s10586-024-04821-9

[9] Hafedh, A.G., Hasan, H.M. (2025). Hybrid strategies for CVRP initial solution: Leveraging weighted score insertion with grid search and multiple insertion with iterative tournament. International Journal of Intelligent Engineering & Systems, 18(1): 733-753. https://doi.org/10.22266/ijies2025.0229.52

[10] Yang, J., Li, Y. (2025). A multicommodity pickup and delivery problem with time windows and handling time in the omni-channel last-mile delivery. International Transactions in Operational Research, 32(3): 1524-1565. https://doi.org/10.1111/itor.13362

[11] Oladejo, S.O., Ekwe, S.O., Mirjalili, S. (2024). The hiking optimization algorithm: A novel human-based metaheuristic approach. Knowledge-Based Systems,

296: 111880. https://doi.org/10.1016/j.knosys.2024.111880
[12] Hamadneh, T., Batiha, B., Alsayyed, O., Montazeri, Z., Ashtiani, H.J., Jafarpour, M., Dehghani, M., Eguchi, K. (2025). On the application of tailor optimization algorithm for solving real-World optimization application. International Journal of Intelligent Engineering and Systems, 18(1): 1-12. https://doi.org/10.22266/ijies2025.0229.01
[13] Hamadneh, T., Batiha, B., Gharib, G.M., Montazeri, Z., Werner, F., Dhiman, G., Dehghani, M., Jawad, R.K., Aram, E., Ibraheem, I.K., Eguchi, K. (2025). Orangutan optimization algorithm: an innovative bio-Inspired metaheuristic approach for solving engineering optimization problems. International Journal of Intelligent Engineering and Systems, 18(1): 47-58. https://doi.org/10.22266/ijies2025.0229.07
[14] Dehghani, M., Trojovský, P. (2023). Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems. Frontiers in Mechanical Engineering, 8: 1126450. https://doi.org/10.3389/fmech.2022.1126450
[15] Alsayyed, O., Hamadneh, T., Al-Tarawneh, H., Alqudah, M., Gochhait, S., Leonova, I., Malik, O.P., Dehghani, M. (2023). Giant armadillo optimization: A new bio-inspired metaheuristic algorithm for solving optimization problems. Biomimetics, 8(8): 619. https://doi.org/10.3390/biomimetics8080619
[16] Trojovska, E., Dehghani, M., Trojovský, P. (2022). Fennec fox optimization: A new nature-inspired optimization algorithm. IEEE Access, 10: 84417-84443. https://doi.org/10.1109/ACCESS.2022.3197745
[17] Jia, H., Rao, H., Wen, C., Mirjalili, S. (2023). Crayfish optimization algorithm. Artificial Intelligence Review, 56(Suppl 2): 1919-1979. https://doi.org/10.1007/s10462-023-10567-4
[18] Ezugwu, A.E., Agushaka, J.O., Abualigah, L., Mirjalili, S., Gandomi, A.H. (2022). Prairie dog optimization algorithm. Neural Computing and Applications, 34(22): 20017-20065. https://doi.org/10.1007/s00521-022-07530-9
[19] Suyanto, S., Ariyanto, A.A., Ariyanto, A.F. (2022). Komodo Mlipir Algorithm. Applied Soft Computing, 114: 108043. https://doi.org/10.1016/j.asoc.2021.108043
[20] Faramarzi, A., Heidarinejad, M., Mirjalili, S., Gandomi, A.H. (2020). Marine Predators Algorithm: A nature-inspired metaheuristic. Expert Systems with Applications, 152: 113377. https://doi.org/10.1016/j.eswa.2020.113377
[21] Kusuma, P.D., Hasibuan, F.C. (2023). Swarm magnetic optimizer: A new optimizer that adopts magnetic behaviour. International Journal of Intelligent Engineering and Systems, 16(4): 264-275. https://doi.org/10.22266/ijies2023.0831.22
[22] Hasanah, R.N, Robandi, I., Syafaruddin, Guntur, H.L., Lystianingrum, V., Djalal, M.R., Prakasa, M.A., Himawari, W. (2024). A novel horse herd optimization algorithm for optimal economic dispatch in sulbagsel electricity system. International Journal of Intelligent Engineering and Systems. 17(6): 1059-1069. https://doi.org/10.22266/ijies2024.1231.78
[23] Kusuma, P.D., Kallista, M. (2023). Quad tournament optimizer: A novel metaheuristic based on tournament among four strategies. International Journal of Intelligent Engineering and Systems, 16(2): 268-278. https://doi.org/10.22266/ijies2023.0430.22
[24] Noroozi, M., Mohammadi, H., Efatinasab, E., Lashgari, A., Eslami, M., Khan, B. (2022). Golden search optimization algorithm. IEEE Access, 10: 37515-37532. https://doi.org/10.1109/ACCESS.2022.3162853
[25] Ghasemi, M., Rahimnejad, A., Akbari, E., Rao, R.V., Trojovský, P., Trojovská, E., Gadsden, S.A. (2023). A new metaphor-Less simple algorithm based on rao algorithms: A Fully Informed Search Algorithm (FISA). PeerJ Computer Science, 9: e1431. https://doi.org/10.7717/peerj-cs.1431
[26] Trojovský, P., Dehghani, M. (2023). Subtraction-Average-Based optimizer: A new swarm-inspired metaheuristic algorithm for solving optimization problems. Biomimetics, 8(2): 149. https://doi.org/10.3390/biomimetics8020149
[27] Dehghani, M., Hubálovský, Š., Trojovský, P. (2022). A new optimization algorithm based on average and subtraction of the best and worst members of the population for solving various optimization problems. PeerJ Computer Science, 8: e910. https://doi.org/10.7717/peerj-cs.910
[28] Katoch, S., Chauhan, S.S., Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. Multimedia Tools and Applications, 80: 8091-8126. https://doi.org/10.1007/s11042-020-10139-6
[29] Trojovska, E., Dehghani, M., Trojovsky, P. (2022). Zebra optimization algorithm: A new bio-Inspired optimization algorithm for solving optimization algorithm. IEEE Access, 10: 49445-49473. https://doi.org/10.1109/ACCESS.2022.3172789
[30] Dehghani, M., Montazeri, Z., Trojovská, E., Trojovský, P. (2023). Coati optimization algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. Knowledge-Based Systems, 259: 110011. https://doi.org/10.1016/j.knosys.2022.110011

NOMENCLATURE

a	agent
A	swarm or population
a_{best}	the best agent
A_{bet}	set of the better agents
a_s	selected agent
b_{lo}	lower border
b_{hi}	higher border
c	candidate
i	index of agent
o	objective function
m	movement
n	number of agents
s	status of agent
r_1	uniform random between 0 to 1
r_2	uniform random whether 1 or 2
r_p	uniform random of population
t	iteration
t_{max}	maximum iteration