Hybrid Marine Predator Algorithm and Hide Object Game Optimization

Purba Daru Kusuma, *Member IAENG*, Dimas Adiputra

***Abstract*—This work presents a hybrid metaheuristic, namely a hybrid marine predator algorithm – hide object game optimization (MPA-HOGO). MPA-HOGO is built by combining two shortcoming metaheuristics: marine predator algorithm (MPA) and hide object game optimization (HOGO). The distinct strategy in MPA-HOGO is as follows. MPA-HOGO utilizes an exploration-to-exploitation strategy gradually as the iteration goes on. It uses the global best solution evaluated in every iteration to ensure that the final solution comes from the best solution found during the iteration. It also eliminates the normalized score of every solution to reduce the computational process. This metaheuristic is then tested by solving 23 classic functions. MPA-HOGO is benchmarked with particle swarm optimization (PSO), Komodo mlipir algorithm (KMA), MPA, and HOGO. The result indicates that MPA-HOGO is superior among these metaheuristics. MPA-HOGO is also competitive in solving practical optimization problems. Through simulation, MPA-HOGO produces 13%, 7%, and 2% lower total costs than PSO, MPA, and HOGO respectively in solving inventory management problems in the vendor-managed inventory system.**

***Index Terms*—metaheuristic, marine predator algorithm, hide object game optimization, vendor managed inventory.**

I. Introduction

Optimization is a study that has been observed and applied extensively. This popularity comes from its objective of solving many problems for the best outcome with limited resources and several constraints. People always try to solve many problems and meet their objectives in many real-world problems. Contrary, resources that can be used are limited. An optimization problem can be found daily within individual to large organizations. Optimization is essential in many areas, such as production process [1], logistics [2], transportation [3], telecommunication [4], education [5], health care [6], and so on. These broad areas make optimization becomes a multi-disciplinary work.

Metaheuristic is a popular optimization method that are extensively studied and implemented in many areas. It utilizes approximate approach so that it can tackle the limitation in computational resources in solving complex optimization problems [7]. Contrary, the exact method needs excessive computational resources that make it impossible to be implemented to solve complex problems [7]. Metaheuristic does not ensure to find the global optimal solution. Metaheuristic only gives the best effort to find the acceptable or sub-optimal solution [7].

In recent days, there are a huge number of metaheuristics. In the early era of metaheuristics, the introduced algorithms were simple. They had distinct exploration and exploitation mechanisms, such as genetic algorithm (GA), PSO, simulated annealing, tabu search (TS), and so on. Due to their simplicity, these old-fashioned ones are still used and modified until now. On the other hand, many metaheuristics have been inspired by nature in the last decades. They were developed based on the mechanics of animals during mating [8], foraging [9], or the combination of these two behaviors [10]. The examples of these algorithms are red deer algorithm (RDA) [8], ant colony optimization (ACO) [11], grey wolf optimizer (GWO) [12], artificial bee colony (ABC) [13], cuckoo search (CS) [14], cat swarm optimization (CSO) [15], KMA [16], and so on.

Marine predator algorithm (MPA) and hide object game optimization (HOGO) are two examples of shortcoming metaheuristics. MPA and HOGO were firstly introduced in 2020. MPA imitates the behavior of marine predators, such as sharks, tuna, and so on, during hunting the prey [17]. It uses two random movements: Brownian motion and Levy Movement [17]. On the other hand, HOGO was inspired by the classic hide object game [18]. MPA is more popular than HOGO even though both algorithms were proven as competitive metaheuristic algorithms. MPA has been widely used and combined in many optimization studies, for example, in optimizing power systems [19], feature selection [20], electronic circuit modeling [21], and so on.

In the first introduction of MPA and HOGO, these algorithms outperformed several metaheuristics in solving 23 functions. The MPA outperformed gravitational search algorithm (GSA), GA, PSO, covariance matrix adaptation evolution strategy (CMA-ES), CS, Salp swarm algorithm (SSA), success-history based parameter adaptation differential evolution (SHADE), and ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood (LSHADE-cnEpSin) [17]. Meanwhile, HOGO outperformed GSA, GA, PSO, teaching learning-based optimization (TLBO), grasshopper optimization algorithm (GOA), GWO, spotted hyena optimizer (SHO), and emperor penguin optimizer (EPO) [18].

This fact shows that MPA and HOGO are superior to these previous algorithms. Their outstanding performance comes from their distinct strategy in conducting exploration and

Manuscript received April 17, 2022; revised October 27, 2022. This work was financially supported by Telkom University, Indonesia.

Purba Daru Kusuma is an assistant professor in computer engineering, Telkom University, Indonesia (e-mail: purboaru@telkomuniversity.ac.id).

Dimas Adiputra is a lecturer in electrical engineering, Institut Teknologi Telkom Surabayam Indonesia (e-mail: adimas@ittelkom-sby.ac.id)

exploitation. The critical success of MPA is in the exploration-to-exploitation strategy during the iteration [17]. Meanwhile, the critical success factor of HOGO is on focusing not only on the best solution only but also avoiding the agent to move toward worse and the worst solutions [18]. Based on this circumstance, creating a better new metaheuristic that combines these two critical success factors will be challenging.

This work proposes a new metaheuristic developed by hybridizing the basic form of MPA and HOGO. This proposed metaheuristic is called a hybrid marine predator algorithm and hide object game optimization (MPA-HOGO) so that it is easily recognized as the combination of MPA and HOGO. MPA-HOGO adapts and modifies the key success factors of MPA and HOGO. Moreover, it also simplifies these two algorithms.

In general, this work's contribution is creating a new metaheuristic algorithm by hybridizing the existing MPA and HOGO. Through this hybridization, the advantages of these two algorithms are combined and modified to improve the algorithms' performance. Below are the contributions of this work.

1) MPA-HOGO modifies the exploration-to-exploitation strategy in MPA, previously split into three distinct phases into gradual transition during the iteration.
2) MPA-HOGO eliminates the normalized fitness score and the probability as conducted in HOGO to reduce the computation process.
3) MPA-HOGO introduces the global best solution to make sure that the final solution comes from the best solution found during the iteration.
4) MPA-HOGO changes the normal distribution implemented in the three steps movement in HOGO into uniform distribution to make it simpler.
5) MPA-HOGO prevents the agent stays in the current solution for the next iteration event though its candidate is worse than the current solution.

The remainder of this paper is constructed as follows. The basic form and characteristics of MPA and HOGO are reviewed in section two. Then, the formal model of MPA-HOGO is explained in section three. The test carried out to evaluate MPA-HOGO, and the related result are presented in section four. After that, the profound analysis regarding the simulation result is discussed in section five. Finally, the conclusion and future research potential are summarized in section six.

II. Related Works

MPA and HOGO are two shortcoming metaheuristics. Both metaheuristics were introduced in 2020. MPA is a metaheuristic that imitates the behavior of marine predators during hunting prey [17]. Meanwhile, HOGO is a metaheuristic adapted from the classic hide object game [18]. These two algorithms have distinct mechanisms for exploration and exploitation.

In MPA, there are two agents, the predators, and the prey. The relationship between the prey and predators is one-to-one so that each prey is exclusively related to a specific predator. Each prey conducts both exploration and exploitation [17]. Meanwhile, the predator represents the best solution so far. The prey does not interact with other prey. The predator also does not interact with other predators. After the iteration, the best predator becomes the final solution. There are two movements implemented in MPA: Levy movement and Brownian motion. Levy movement has been chosen due to its similarity to marine predators' behavior during hunting prey [22]. Both prey and predator locations represent the solution within the problem space.

In MPA, the iteration affects the selection of exploitation and exploration. Exploration means the prey moves toward their predator, while exploitation means the prey moves based on their predator's movement. The iteration is split into three phases with the equal size [17]: (1) exploration by utilizing the Brownian motion, (2) Levy movement implementation for exploration and Brownian implementation for exploitation, (3) implementation of the Levy movement for exploitation. Each time the prey moves to its new solution, the algorithm updates the predator's new solution. This prey's new solution replaces its predator's current solution only if its new solution is better than its current one. Otherwise, the predator is still in its current solution.

The last process in every iteration is implementing Eddy formation. The prey moves toward a new solution based on two options in this process. The prey moves toward the solution within its local search space in the first option. In the second option, the prey moves toward two randomly selected prey. In the beginning, its local search space is vast. Then, this local search space is gradually narrower as the iteration continues. This strategy also represents a shifting mechanism from exploration-dominant strategy to exploitation-dominant strategy.

HOGO implements a very different strategy rather than MPA. It consists of only a set of agents or solutions, where iteration does not affect the chosen strategy. In every iteration, both exploitation and exploration are conducted. Meanwhile, there is intensive interaction among agents.

In HOGO, the best and worst solutions are selected in every iteration [18]. Besides the actual fitness score, the algorithm also stores the normalized fitness score of every agent. This normalized fitness score mechanism also can be found in other algorithms, such as the football game-based optimizer (FBGO) [23]. A candidate for the following location is calculated based on three considerations. The first one is that the agent moves toward the best solution. On the contrary, the second one is that the agent moves away from the worst solution. The third one is that the agent moves toward or away from the randomly selected agents by comparing the agent's normalized score and the selected agent's normalized score.

These three considerations are then accumulated and combined with the agent's current location as the exploitation candidate. If this candidate is better than the agent's current candidate, then this candidate becomes the agent's following location. Otherwise, there are two options. The first option is that the agent will stay in its current location. The second option is that the agent will move to its new location that is chosen randomly within its local search space.

This concept is represented in six processes. The first process is finding the best and worst solutions among the population. The second process is normalizing all solutions. This normalized score is called a voice. The third process calculates the probability of every voice by dividing the

solution's voice by the maximum voice. A higher voice means a higher probability. The fourth process is generating the primary candidate by using the three steps. The fifth process generates alternative candidates by finding randomized solutions around the current solution. The sixth process determines the agent's next solution based on three possibilities: the primary candidate, alternative candidate, or current. If the primary candidate is better than the current solution, the main candidate becomes the next solution.

Meanwhile, suppose this primary candidate is not better than the current candidate, and the ratio between the current solution voice and the maximum voice is equal to or less than 0.5. In that case, the alternative candidate becomes the next solution. Otherwise, the agent remains on the current solution.

There are several notes due to these two algorithms. HOGO does not implement a global best solution evaluated in every iteration, as it is used in many metaheuristic algorithms, such as in PSO [24]. HOGO keeps only the current best solution calculated based on current solutions. In the next iteration, this best solution will be forgotten. It means that the last current best solution becomes the final solution, although this current best solution is not the best solution during the iteration. In other words, the current best solution will replace the previous best solution, although this current best solution is worse than the previous best solution. In this condition, HOGO may miss the actual best solution. The probability of missing the real best solution is still the same in the whole iteration. This condition is different from the simulated annealing (SA). In SA, a worse solution may be accepted to replace the current solution [25]. But its probability is reduced during the iteration [25]. It means the probability of a worse solution being accepted becomes more difficult as the iteration goes [25]. Although SA generally focuses on neighborhood search, i.e., exploitation, exploration is accessible in the early iteration.

The exploration-to-exploitation concept is applied in MPA by implementing two mechanisms. The first mechanism is by dividing the iteration into three fixed-length phases. The first phase focuses on exploration. Then, the second phase creates a balance between exploration and exploitation. Finally, the third phase focuses on exploitation. The second mechanism is by controlling the local problem space during the Eddy formation process. In it, the local search space becomes narrower as the iteration goes.

The exploration-to-exploitation strategy benefits in finding the area of the global optimal solution and avoiding the local optimal trap earlier. Then, it can focus on exploiting this possible narrow area. Moreover, this strategy also prevents the algorithm jumps to other new areas in the middle of the iteration and stops the exploitation process.

Unlike HOGO, in MPA, the best solution during the iteration becomes the final solution. It is because the final solution is selected from the best predator. Meanwhile, the prey's solution can replace the predator's solution only if this prey's solution is better than the predator's one.

On the other hand, HOGO has an advantage over MPA. First, HOGO finds the best solution among the entire population in every iteration. Contrary, in MPA, the best solution is evaluated locally by comparing the prey and the predator. The best solution among predators is found after the iteration process ends. Second, HOGO considers the best solution and the worst solution. HOGO tends to avoid the worst and selected solutions in the three-step process if their fitness is worse than the current solution.

Based on this review, it is possible to hybridize these two metaheuristics to benefit from their advantages in specific ways. The first one implements the exploration-to-exploitation strategy. The global best solution is solved in the second one. And the third one considers both the worst solution and the best solution.

III. Proposed Model

MPA-HOGO is mainly developed based on HOGO. As a HOGO model, this proposed metaheuristic still adopts several mechanisms in both exploitation and exploration. There are several agents that act autonomously to find the optimal solution. Each agent concerns on three parameters during exploitation: the best, the worst, and the randomly selected solution. These parameters are represented in three steps that will be accumulated together as candidates for the next movement. First, it represents the movement toward the best solution. The second step represents the movement away from the worst solution. The third step represents the movement related to the randomly selected solution. If the selected solution is better than the current solution, the agent moves toward this solution. Otherwise, the agent moves away from this selected solution.

The accumulation of these three steps is then combined with the current solution to produce the candidate for the next movement. Like in HOGO, this candidate replaces the current solution only if this candidate is better than the current solution. Otherwise, the agent will conduct exploration. The agent will find a new alternative solution around its current solution during the exploration. This area can be called a local search space. When this local search space is wide enough, the new solution can be far from the current solution. Otherwise, the new solution is only near the current solution when the local search space is narrow.

Like MPA, exploration-dominant strategy is taken in the early iteration. Then, this strategy moves gradually to exploitation-dominant strategy as the iteration continues. This concept is then enriched with the iteration-controlled mechanism implemented in MPA. This mechanism is conducted by setting the local search space wide enough at the iteration's beginning. Then, this local search space becomes narrower as the iteration continues. This mechanism is like in MPA. In the beginning, the algorithm focuses on finding the area of the optimal solution. After this area has been found, then in the later iteration, the algorithm focuses on exploiting this area to find the optimal solution more effectively and forgetting the possibility of searching for other solutions somewhere else within the problem space.

Moreover, this concept is also equipped with the existence of the global best. The global best is used to store the best solution found so far. The existence of the global best also becomes anticipation that only the best solution found during the iteration will become the final solution. The current best solution replaces the global best solution only if it is better than it.

MPA-HOGO simplifies the basic form of HOGO in several ways. First, it uses a uniform distribution rather than

a normal distribution to determine the three steps. Second, it no longer uses a normalized score to determine the direction of the third step and the exploration.

This concept is then interpreted into a mathematical model and the algorithm. There are some annotations that are used in the algorithm and mathematical model. Below are these annotations and their descriptions. The algorithm is shown in algorithm 1.

b_l	lower bound
b_u	upper bound
c_1	exploitation candidate
c_2	exploration candidate
d_1	step related to the best solution
d_2	step related to the worst solution
d_3	step related to the selected solution
D	dimension
f	fitness
n	population size
r_t	iteration ratio
r_e	exploration ratio
r_w	local space ratio
x	solution
x_{best}	the best solution
x_{worst}	the worst solution
$x_{globbest}$	global best solution
x_{sel}	selected solution
t	time or iteration
t_{max}	maximum iteration
U	uniform random

algorithm 1: MPA-HOGO main algorithm

```
output: x_globbest
begin
  for i=1 to n
    initialize(x_i)
  end
  for t=1 to t_max
    find (x_best)
    find (x_worst)
    update (x_globbest)
    for i=1 to n
      find (x_sel)
      calculate (c_1)
      calculate (c_2)
      determine next position (x_i)
    end
  end
end
```

All initial solutions are randomized within the search space. It is formalized by using (1), in which the initial solution follows a uniform distribution.

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

Iteration is conducted after initialization. The iteration runs until the maximum iteration is reached. Three processes are conducted at the beginning of every iteration: determining the best solution, determining the worst solution, updating the global best, and determining the iteration ratio. These processes are formalized by using (2) to (5).

$$x_{best} = x \in X, min\big(f(x)\big) \tag{2}$$

$$x_{worst} = x \in X, max\big(f(x)\big) \tag{3}$$

$$x_{globbest} = \begin{cases} x_{best}, f(x_{best}) < f\big(x_{globbest}\big) \\ x_{globbest}, else \end{cases} \tag{4}$$

$$r_t = 1 - \frac{t}{t_{max}} \tag{5}$$

After these variables are determined, the next process is determining the next solution for every agent. This process is conducted for all agents by iterating this process for entire population. There are two candidates used to determine the next solution: the exploitation candidate and the exploration candidate. Determination of the exploitation candidate is formalized by using (6) to (10). Meanwhile, determination of the exploration candidate is formalized by using (11) and (12).

$$x_{sel} = U(X) \tag{6}$$

$$d_1 = U(0,1).r_t.(x_{best} - x) \tag{7}$$

$$d_2 = U(0,1).r_t.(x - x_{worst}) \tag{8}$$

$$d_3 = \begin{cases} U(0,1).r_t.(x_{sel} - x), f(x_{sel}) < f(x) \\ U(0,1).r_t.(x - x_{sel}), else \end{cases} \tag{9}$$

$$c_1 = x + d_1 + d_2 + d_3 \tag{10}$$

Below is the explanation of (6) to (10). Equation (6) states that a solution is selected randomly within the set of solutions. The exploitation movement toward the best solution is stated in equation (7). Equation (8) states that the exploitation avoids the worst solution. Equation (9) states that exploitation moves toward the selected solution only if this selected solution is better than the current solution. Otherwise, the exploitation avoids this selected solution. Equation (10) states that the exploitation candidate is determined by accumulating the current solution with these three steps.

$$r_e = (1 - r_w) + U(0,1).r_t \tag{11}$$

$$c_2 = \begin{cases} x + r_e.(x - b_c), b_l \geq 0 \vee b_u \leq 0 \\ r_e.x, else \end{cases} \tag{12}$$

Below is the explanation of (11) and (12). Equation (11) determines the exploration ratio, which depends on the local space ratio and the iteration ratio. A higher local space ratio means more expansive local problem space for exploration. Then, the exploration candidate is determined by using (12).

After these two candidates are determined, the final process is determining the next solution. The exploitation candidate is chosen only if the exploitation candidate is better than the current solution. Otherwise, the exploration candidate becomes the next solution. This process is formalized by using (13).

$$x' = \begin{cases} c_1, f(c_1) < f(x) \\ c_2, else \end{cases} \quad (13)$$

IV. Simulation

Four tests are carried out to evaluate MPA-HOGO's performance. The first test is observing MPA-HOGO's performance in solving 23 classic functions. The second test is carried out to observe the convergence of MPA-HOGO. The third test is carried out to observe the relation between the local space ratio and the metaheuristic's performance. The fourth test is carried to observe MPA-HOGO's performance in solving the practical problem.

In the first test, MPA-HOGO is implemented to solve 23 classic functions. These functions are popular and widely used in many metaheuristic studies, such as in KMA [16], HOGO [18], MPA [17], RDA [8], EPO [26], FBGO [23], and so on. These functions are divided into three groups. The first group consists of seven multi-dimension unimodal functions. On the other hand, six multi-dimension multimodal functions are included in the second group. The third group consists of ten fixed dimension multimodal functions. The detailed description of these functions is shown in Table 1.

TABLE I
23 Benchmark Functions

No	Function	D	$[b_l, b_u]$	min(f)
1	Sphere	10	[-100, 100]	0
2	Schwefel 2.22	10	[-100, 100]	0
3	Schwefel 1.2	10	[-100, 100]	0
4	Schwefel 2.21	10	[-100, 100]	0
5	Rosenbrock	10	[-30, 30]	0
6	Step	10	[-100, 100]	0
7	Quartic	10	[-1.28, 1.28]	0
8	Schwefel	10	[-500, 500]	-4189.8
9	Ratsrigin	10	[-5.12, 5.12]	0
10	Ackley	10	[-32, 32]	0
11	Griewank	10	[-600, 600]	0
12	Penalized	10	[-50, 50]	0
13	Penalized 2	10	[-50, 50]	0
14	Shekel Foxholes	2	[-65, 65]	1
15	Kowalik	4	[-5, 5]	0.0003
16	Six Hump Camel	2	[-5, 5]	-1.0316
17	Branin	2	[-5, 5]	0.398
18	Goldstein-Price	2	[-2, 2]	3
19	Hartman 3	3	[1, 3]	-3.86
20	Hartman 6	6	[0, 1]	-3.32
21	Shekel 5	4	[0, 10]	-10.1532
22	Shekel 7	4	[0, 10]	-10.4028
23	Shekel 10	4	[0, 10]	-10.5363

Each function in the first group has only one optimal, the global optimal [27]. These functions do not have any local optimal, so the algorithm will not be trapped in the local optimal. The main objective is to find the global optimal as fast as possible. Meanwhile, these functions have ample problem space and high dimensions. This circumstance can make the algorithm fails to reach the global optimal within the given iteration. In these functions, 0 becomes the center of the problem space.

Each function in the second group has several optimal solutions as a multimodal function. There is only one global optimal, while the others are local optimal [27]. All these functions have high dimensions. Some functions have ample problem space, while others have narrow problem space. Like in the first group, 0 becomes the central of the problem space in this second group.

The third group consists of multimodal functions with fixed dimension problem space. Shekel Foxholes becomes the only function in this group with an ample problem space, while the other functions have narrow to moderate problem space. Unlike the first and the second groups, the dimension of these functions is low. From function 14 to function 18, 0 becomes the center of the problem space. On the other hand, in function 19 to function 23, the center of the problem space is not 0.

In this test, MPA-HOGO is compared with four metaheuristics: PSO, KMA, MPA, and HOGO. They are chosen because of several reasons. Moreover, these four algorithms have distinct mechanisms during exploration and exploitation.

PSO represents a well-known metaheuristic built based on swarm intelligence. Moreover, PSO is also the early swarm intelligence algorithm, and it has many derivatives. As a swarm intelligence, the system consists of several autonomous agents that work independently to find the global optimal [28]. Although they work autonomously, a collective intelligence, namely global best, is shared among these agents. The global best is the current solution with the best fitness value. Its popularity comes from its simple mechanism. This mechanism can be formalized using a single equation to determine the agent's following location. Several parameters determine the agent's next position: the agent's current location, the global best, and the local best [24]. In this simulation, all weights in PSO are set to 0.5.

KMA represents a very brand-new metaheuristic mimicking the behavior of the Komodo dragon, an ancient monitor lizard that lives on Komodo island, Indonesia, during foraging and mating. The system consists of a several number of Komodo divided into three groups: big males, females, and petite males [16]. The big males move toward other better big males and avoid getting closer to other worse big males (exploitation) [16]. Females conduct reproduction in two optional ways. The first way is by mating with the highest quality big male to create two offspring (exploitation) [16]. The better offspring replaces the female. The second way is by conducting parthenogenesis or asexual reproduction. In this way, a female moves toward a randomized location within the problem space (exploration). The petite males move toward the big males at a certain low speed (exploitation) [16]. KMA can be seen as a hybrid algorithm that combines swarm intelligence and evolutionary algorithm. This mechanism is also adopted in RDA. There is only one female in this simulation, while the number of big males and petite males is balanced.

MPA and HOGO are chosen because this proposed metaheuristic is built by hybridizing these two algorithms. This test is conducted to determine whether MPA-HOGO is better or worse than its basic form. In other words, due to this simulation, it can be evaluated in which functions MPA-HOGO is better or worse than its original form.

Typical parameters used in all these metaheuristics are a maximum iteration of 100, a population size of 20, and 30 runs implementation for every function. The result is shown in Table 2, in which the bold font represents the best result. In MPA-HOGO, the local space ratio is set 0.1.

TABLE II
BENCHMARK SIMULATION RESULT

Function	Fitness Value					Better Than
	PSO	KMA	MPA	HOGO	MPA-HOGO	
1	3.614×10^{2}	6.001×10^{2}	8.615×10^{-1}	8.983×10^{-4}	**3.952×10^{-25}**	PSO, KMA, MPA, HOGO
2	7.700×10^{-9}	1.618×10^{-2}	**0**	**0**	**0**	PSO, KMA
3	2.008×10^{3}	1.775×10^{3}	5.469	4.803×10^{1}	**3.273×10^{-23}**	PSO, KMA, MPA, HOGO
4	1.155×10^{1}	1.473×10^{1}	7.529×10^{-1}	2.402×10^{-1}	**0**	PSO, KMA, MPA, HOGO
5	5.289×10^{4}	5.387×10^{4}	1.685×10^{1}	3.841×10^{1}	**9.000**	PSO, KMA, MPA, HOGO
6	1.946×10^{2}	4.404×10^{2}	2.894	**1.168×10^{-2}**	2.51×10^{-1}	PSO, KMA, MPA
7	1.021×10^{-1}	1.703×10^{-1}	7.061×10^{-3}	1.163×10^{-1}	**3.245×10^{-4}**	PSO, KMA, MPA, HOGO
8	-3.243×10^{3}	**-3.255×10^{3}**	-1.828×10^{3}	-2.751×10^{3}	-2.247×10^{3}	MPA
9	3.697×10^{1}	4.220×10^{1}	**1.149**	3.826×10^{1}	1.109×10^{1}	PSO, KMA, HOGO
10	8.193	9.210	6.039×10^{-1}	1.995×10^{-2}	**4.441×10^{-16}**	PSO, KMA, MPA, HOGO
11	4.005	7.119	4.165×10^{-1}	2.859×10^{-1}	**0**	PSO, KMA, MPA, HOGO
12	1.608×10^{1}	1.732×10^{1}	1.150	3.062×10^{-2}	**7.964×10^{-6}**	PSO, KMA, MPA, HOGO
13	1.577×10^{3}	1.179×10^{4}	3.206	**8.661×10^{-2}**	2.419	PSO, KMA, MPA
14	9.357	9.297	5.754	6.558	**4.818**	PSO, KMA, MPA, HOGO
15	6.041×10^{-3}	1.523×10^{-2}	**3.597×10^{-3}**	5.815×10^{-3}	3.858×10^{-2}	-
16	**-1.032**	-1.022	-1.024	-1.031	-1.000	-
17	3.981×10^{-1}	6.758×10^{-1}	9.957×10^{-1}	**4.071×10^{-1}**	4.309×10^{-1}	KMA, MPA
18	3.871	3.199	4.982	3.027	**3.000**	PSO, KMA, MPA, HOGO
19	-3.097	-5.062×10^{-1}	-3.719	-4.954×10^{-2}	**-3.843**	PSO, KMA, MPA, HOGO
20	-3.220	-2.841	-2.045	**-3.231**	-1.457	-
21	-4.743	**-6.139**	-1.612	-4.853	-4.778	PSO, MPA
22	-5.416	**-6.980**	-1.899	-6.124	-4.858	MPA
23	-5.000	**-6.186**	-1.924	-6.076	-3.273	MPA

The result indicates that MPA-HOGO generally achieves the main objectives of metaheuristics: finding the sub-optimal solution and avoiding the local optimal entrapment. This achievement occurs in all functions. Moreover, MPA-HOGO can find the global optimal in certain functions: Schwefel 2.22, Schwefel 2.21, Griewank, and Goldstein-Price.

This proposed metaheuristic is also proven competitive among other algorithms (PSO, KMA, MPA, and HOGO). It outperforms all these algorithms in solving 11 functions. It is better than both PSO and KMA in solving 16 functions. It is better than MPA in solving 18 functions. It is also better than HOGO in solving 12 functions. Unfortunately, its performance is less competitive in solving three functions: Kowalik, Six Hump Camel, and Hartman 6.

The second test is conducted to observe the convergence aspect of MPA-HOGO. Like in the first simulation, MPA-HOGO is tested to solve 23 benchmark functions in this second one. But, in this simulation, there are three maximum iterations for every function: 60, 120, and 180. The first value represents the maximum iteration far lower than in the first simulation. The second value represents the maximum slightly higher iteration than in the first simulation. The third value represents the maximum iteration that is far higher than the first simulation. The result is shown in Table 3.

The result indicates that the convergence condition depends on the function to solve. In seventeen functions, convergence is achieved in the low maximum iteration. These functions include Schwefel 2.22, Rosenbrock, Step, Rastrigin, Griewank, Penalized, Penalized 2, Shekel Foxholes, Kowalik, Six Hump Camel, Branin, Goldstein-Price, Hartman 3, Hartman 6, Shekel 5, Shekel 7, and Shekel 10. In two functions, convergence occurs in the moderate maximum iteration. These functions include Schwefel 2.21 and Ackley. In other algorithms, the convergence has not been achieved yet in the high maximum iteration, but the acceptable solution has been achieved.

TABLE III
CONVERGENCE RESULT

Function	$t_{max} = 60$	$t_{max} = 120$	$t_{max} = 180$
1	1.743×10^{-13}	1.946×10^{-30}	5.694×10^{-48}
2	0	0	0
3	3.901×10^{-12}	1.245×10^{-28}	1.809×10^{-45}
4	2.598×10^{-7}	0	0
5	9.000	9.000	9.000
6	2.772×10^{-1}	2.750×10^{-1}	2.882×10^{-1}
7	6.645×10^{-4}	1.935×10^{-4}	9.689×10^{-5}
8	-2.046×10^{3}	-2.268×10^{3}	-2.288×10^{3}
9	1.258×10^{1}	1.116×10^{1}	1.064×10^{1}
10	2.635×10^{-1}	4.441×10^{-16}	4.441×10^{-16}
11	5.826×10^{-14}	0	0
12	7.964×10^{-6}	7.964×10^{-6}	7.964×10^{-6}
13	2.585	2.312	2.153
14	4.613	3.545	3.979
15	3.589×10^{-2}	3.346×10^{-2}	4.067×10^{-4}
16	-1.000	-1.000	-1.002
17	4.441×10^{-1}	4.133×10^{-1}	4.141×10^{-1}
18	3.000	3.000	3.000
19	-3.818	-3.861	-3.866
20	-1.336	-1.204	-1.396
21	-4.859	-5.384	-5.255
22	-3.018	-5.191	-6.014
23	-3.638	-3.865	-4.093

In the third simulation, the local space ratio is set 0.1, 0.5, and 0.9. The first value represents the small local space ratio while the third value represents the big local space ratio. The result is shown in Table 4. Meanwhile, the best performance is presented in bold font.

Table 4 shows that the response of MPA-HOGO regarding the local space ratio is various. There are 13, 4, and 3 functions that the best result is obtained when the local space ratio is high, moderate, and low respectively. These 13 functions are distributed in all categories. There is one function that the best result is obtained when the local space ratio is moderate or high. Meanwhile, there are two functions that the best result is obtained in all local space ratio.

TABLE IV
LOCAL SPACE RATIO SIMULATION RESULT

Function	$r_w = 0.1$	$r_w = 0.5$	$r_w = 0.9$
1	$4.045x10^{2}$	$7.242x10^{-25}$	**$1.962x10^{-103}$**
2	**0**	**0**	**0**
3	$1.589x10^{3}$	$3.161x10^{-23}$	**$2.635x10^{-98}$**
4	$1.953x10^{1}$	0	**$1.793x10^{-51}$**
5	$1.644x10^{5}$	9.000	**8.953**
6	$2.313x10^{2}$	**$2.484x10^{-1}$**	$3.456x10^{-1}$
7	3.587	$3.603x10^{-4}$	**$1.044x10^{-75}$**
8	**$-2.227x10^{3}$**	$-1.846x10^{3}$	$-1.554x10^{3}$
9	$3.158x10^{1}$	$1.014x10^{1}$	**2.905**
10	$1.481x10^{1}$	**$4.441x10^{-16}$**	**$4.441x10^{-16}$**
11	5.451	$1.461x10^{-1}$	**$3.689x10^{-2}$**
12	$2.304x10^{5}$	$4.439x10^{-2}$	**$7.965x10^{-6}$**
13	$4.380x10^{6}$	6.722	**3.854**
14	1.778	4.640	6.567
15	**$8.572x10^{-2}$**	$9.290x10^{-2}$	$1.064x10^{-1}$
16	**$-8.699x10^{-2}$**	-1.004	$-9.916x10^{-1}$
17	$6.382x10^{-1}$	**$4.364x10^{-1}$**	$5.937x10^{-1}$
18	**3.000**	**3.000**	**3.000**
19	$-4.954x10^{-2}$	-3.849	**-3.894**
20	-1.194	**-1.393**	-1.375
21	$-8.593x10^{-1}$	-5.104	**-5.287**
22	$-9.357x10^{-1}$	-4.535	**-5.350**
23	-1.023	-3.377	**-4.214**

The fourth test is carried out to evaluate MPA-HOGO's performance in solving the real-world optimization problem. In this simulation, the case is a supply chain problem in the vendor-managed inventory (VMI) system. VMI is a supply chain management system where the vendor manages the clients' stock [29]. It is different from the standard vendor-client system. In the common system, clients send the purchase order to their vendor. Then, the vendor must fulfill these orders in exact items and quantity within the due date. The vendors do not have access to their clients' inventory.

A client does not need to send a purchase order to make its vendors send products to fill the client's inventory. On the other hand, in VMI, the vendor has full access to monitor and manage its clients' inventory [30]. Through VMI, the vendor has better information about its clients' real needs to serve them better.

In this test, there is a vendor and 40 clients. Eight of them are big clients, while the others are small clients. The vendor manages only a single product. The vendor must keep its big clients' inventory within 1,000 to 2,000 units. Meanwhile, the vendor must keep its small clients' inventory within 100 to 200 units. The unit cost of a big client is 40,000 rupiah per product. On the contrary, the unit cost of a small client is 50,000 rupiah per product. The objective is to minimize the total cost of maintaining these clients. This optimization problem can be seen as a high dimension problem due to there are 40 clients that must be managed.

In this test, MPA-HOGO is also compared with algorithms as in the first simulation: PSO, KMA, MPA, and HOGO, as shown in Table 5. The population size is 20. The maximum iteration is 100.

TABLE V
REAL WORLD PROBLEM SIMULATION RESULT

Algorithm	Total Cost (rupiah)
PSO	107,474,838
KMA	91,004,832
MPA	101,681,290
HOGO	95,501,899
MPA-HOGO	93,557,187

Table 5 indicates that MPA-HOGO is competitive enough in solving this VMI optimization problem. Its performance outperforms the PSO, MPA, and HOGO. Its total cost is the lowest one among other algorithms. Its total cost is 13% lower than PSO, 7% lower than MPA, and 2% lower than HOGO. On the other hand, its performance is worse than KMA. Its total cost is 3% higher than KMA.

V. Discussion

The test result indicates that MPA-HOGO is a good metaheuristic. It can solve a problem and find an acceptable or near-optimal solution within the given iteration. MPA-HOGO is also competitive enough compared with the other algorithms (PSO, KMA, MPA, and HOGO) to solve the theoretical mathematic functions as indicated through the 23 functions. Moreover, this hybridized algorithm is better than the MPA and HOGO's original forms. The convergence test also shows that MPA-HOGO can find an acceptable solution, mainly in the low iteration. Meanwhile, there are several notes due to this achievement.

MPA-HOGO's superiority occurs mainly in solving the unimodal functions, as indicated in the first group. Its superiority occurs in solving five out of six given functions. It can find the global optimal or near-optimal solution fast enough even though the problem space is very large, and the dimension is high. The extremely high precision result obtained in solving Sphere and Schwefel 1.2 proves this superiority. The exploration-dominant mechanism in the earlier iteration makes the algorithm jump faster to the area close to the optimal solution. Meanwhile, the exploitation-dominant mechanism in the later iteration supports the system to focus on the optimal solution neighborhood. Moreover, the narrow local problem space in the later iteration also improves its precision.

MPA-HOGO is superior in solving the high dimension multimodal functions, as indicated in the second group. Its superiority occurs in solving four out of seven given functions. This result indicates that MPA-HOGO is robust enough in solving multimodal problems, whether the problem space is narrow or large. MPA-HOGO can tackle the local optimal problem by solving the multimodal functions, as it is common in metaheuristic studies. Exploration-dominant and vast local problem space in the earlier iteration make the system can find the area near the true optimal solution fast. Meanwhile, the exploitation-dominant with narrow local problem space avoids the agents in the system jumping to other areas where local optimal lies within the problem space when it fails to find a better solution in the later iteration.

Both the first and second groups have a similarity. The central point of their problem space is 0. The local problem space in the later iteration is also close to 0. This circumstance gives two benefits. The first benefit is that the precision of the solution will be very high. The result in solving Sphere, Schwefel 1.2, Quartic, Ackley, and Penalized proves this argument. Its precision is far higher than other metaheuristics. The second benefit is that jumping to other areas within the search space can be avoided.

MPA-HOGO is less competitive in solving functions in the third group. Its superiority occurs only in solving three out of ten given functions. Meanwhile, this proposed algorithm fails to beat any algorithms in solving Kowalik, Six Hump Camel,

and Hartman 6. This proposed metaheuristic is also not competitive in solving Shekel 5, Shekel 7, and Shekel 10. There is similarity in Six Hump Camel, Hartman 6, Shekel 5, Shekel 7, and Shekel 10. The optimal solution for all these functions is not at 0. Fortunately, the performance gap between MPA-HOGO and other algorithms is not far.

The result of the third test indicates that too wide exploration space is counterproductive. Exploration is necessary in escaping from the local optimal entrapment. But the convergence is difficult to achieve when the exploration space is too wide. Moreover, narrow local search space makes the algorithm focuses on the narrow region once the region where the global optimal solution exists.

The result of the fourth test shows that MPA-HOGO is not superior only in solving the theoretical mathematical problem but also the real-world optimization problem. On the other hand, KMA, which is inferior to MPA, HOGO, and MPA-HOGO, shows its superiority in solving real-world problems. This circumstance strengthens the non-free-lunch theory that says that the performance of the metaheuristic depends on the problem to solve [31]. The different problems may produce different performances.

In the end, all methods must be tested to solve real-world problems to prove their performance. Many real-world optimization problems use integer representation, such as the number of products, vehicles, etc. It is very different from theoretical mathematical problems using floating point numbers, where higher precision may produce a significant gap in the result in the floating-point number. The difference also occurs in the objective function. In theoretical mathematic problems, their objective is complicated. Contrary, in the operations research problems, their objective is primarily simple, such as minimizing cost [32], make-span [33], penalties [34], travel distance [35], number of vehicles [36], and so on. These objectives are linear and calculated by accumulating the values of all considered parameters.

The next challenge is making this MPA-HOGO widely used. This work has shown that MPA-HOGO is competitive in optimizing the theoretical problems. Meanwhile, there are many shortcoming metaheuristics such as the northern goshawk optimizer (NGO) [37] or golden search optimizer (GSO) [38] that are also superior to the old-fashioned algorithms. On the other hand, many old-fashioned algorithms are still popular today. The examples are as follows. Yang et al. [39] combined harmony search and firefly algorithm to optimize six continuous functions. Wang et al. [40] modified the genetic algorithm to optimize the edge cloud cooperative computing system. Artificial bee colony algorithm is also used in many optimizations, such as in uncertain shortest path problem [41], indoor optimization problem [42], wireless sensor network [43], and so on.

VI. Conclusion

This work has demonstrated that MPA-HOGO is a good and competitive metaheuristic. It can find the near-optimal solution or acceptable solution within the given iteration. It achieves convergence fast. MPA-HOGO is proven in solving theoretical unimodal and multimodal functions, whether the search space is narrow or wide. Its performance is superior in solving 11 benchmark functions. Through simulation, MPA-HOGO is better than its origin, both MPA and HOGO. It is better than MPA in solving 18 functions, and HOGO solving 12 functions. MPA-HOGO is also competitive in solving a practical problem. In this study, MPA-HOGO produces a lower total cost than PSO, MPA, and HOGO in solving inventory management problems in the VMI system.

This work also demonstrated that studies in hybridizing existing metaheuristics are as important as developing new ones. In the future, there is potential to improve this proposed algorithm by combining it with other algorithms. Moreover, implementing this algorithm to optimize many real-world problems like the combinatorial problem is also enjoyable.

References

[1] M. Takano and M. Nagano, "Solving the permutation flow shop problem with blocking and setup time constraints," *International Journal of Industrial Engineering Computations,* vol. 11, pp. 469–480, 2020.

[2] W. Yu, C. Ha, and S. Park, "A hybrid genetic algorithm for integrated truck scheduling and product routing on the cross-docking system with multiple receiving and shipping docks," *Mathematical Problems in Engineering*, ID: 2026834, pp. 1-17, 2021.

[3] Y. Xie, Y. Kong, H. Xiang, Y. Hou, and D. Han, "A metaheuristic with learning mechanism for solving the multi-school heterogeneous school bus routing problem," *IAENG International Journal of Computer Science*, vol. 48, no. 4, pp. 884-892, 2021.

[4] H. M. Ali, J. Liu, and W. Ejaz, "Planning capacity for 5G and beyond wireless networks by discrete fireworks algorithm with ensemble of local search methods," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, ID: 185, pp. 1-24, 2020.

[5] I. Balan, "A new genetic approach for course timetabling problem," *Journal of Applied Computer Science & Mathematics*, vol. 15, no. 1, pp. 9-14, 2021.

[6] Z. A. Abdalkareem, A. Amir, M. A. Al-Betar, P. Ekhan, and A. I. Hammouri, "Healthcare scheduling in optimization context: a review," *Health and Technology*, vol. 11, pp. 445-469, 2021.

[7] 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.

[8] A. M. Fathollahi-Fard, M. Hajjaghaei-Keshteli, and R. Tavakkoli-Moghaddam, "Red deer algorithm (RDA): a new nature-inspired metaheuristic," *Soft Computing*, vol. 24, pp. 14637-14665, 2020.

[9] M. Braik, A. Sheta, H. Al-Hiary, "A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm," *Neural Computing and Applications*, vol. 33, pp. 2515-2547, 2021.

[10] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," *Soft Computing*, vol. 23, no. 3, pp. 715-734, 2019.

[11] S. Liang, T. Jiao, W. Du, and S. Qu, "An improved ant colony optimization algorithm based on context for tourism route planning," *PLOS One*, vol. 16, no. 9, ID: e0257317, pp. 1-16, 2021.

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

[13] K. Hussain, M. N. M. Salleh, S. Cheng, Y. Shi, and R. Naseem, "Artificial bee colony algorithm: a component-wise analysis using diversity measurement," *Journal of King Saud University – Computer and Information Sciences*, vol. 32, no. 7, pp. 794-808, 2020.

[14] K. Sharma, S. Singh, and R. Doriya, "Optimized cuckoo search algorithm using tournament selection function for robot path planning," *International Journal of Advanced Robotic System*, vol. 18, no. 3, pp. 1-11, 2021.

[15] A. M Ahmed, T. A. Rashid, and S. A. M. Saeed, "Cat swarm optimization algorithm: a survey and performance evaluation," *Computational Intelligence and Neuroscience*, vol. 2020, ID: 4854895, pp. 1-20, 2020.

[16] Suyanto, A. A. Ariyanto, and A. F. Ariyanto, "Komodo mlipir algorithm," *Applied Soft Computing*, vol. 114, ID: 108043, 2022.

[17] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. M. Gandomi, "Marine predators algorithm: a nature-inspired metaheuristic," *Expert Systems with Applications*, vol. 152, ID: 113377, 2020.

[18] M. Dehghani, Z. Montazeri, S. Saremi, A. Dehghani, O. P. Malik, K. Al-Haddad, and J. M. Guerrero, "HOGO: hide objects game optimization," *International Journal of Intelligent Engineering & Systems*, vol. 13, no. 4, pp. 216-225, 2020.

[19] M. Z. Islam, M. L. Othman, N. I. A. Wahab, V. Veerasamy, S. R. Opu, A. Inbamani, and V. Annamalai, "Marine predators algorithm for solving single-objective optimal power flow," PLOS One, vol. 16, no. 8, pp. 1-27, 2021.
[20] D. S. A. Elminaam, A. Nabil, A. Ibraheem, and E. H. Houssein, "An efficient marine predators algorithm for feature selection," *IEEE Access*, vol. 9, no. 60136-60153, 2021.
[21] 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, 2021.
[22] E. Humphries, N. Queiroz, J. R. M. Dyer, N. G. Pade, M. K. Musyl, K. M. Schaefer, D. W. Fuller, J. M. Brunnschweiler, T. K. Doyle, J. D. R. Houghton, G. C. Hays, C. S. Jones, L. R. Noble, V. J. Wearmouth, E. J. Southall, and D. W. Sims, "Environmental context explains Levy and Brownian movement patterns of marine predators," *Nature*, vol. 465, pp. 1066-1069, 2010.
[23] M. Dehghani, M. Mardaneh, J. M. Guerrero, O. P. Malik, and V. Kumar, "Football game based optimization: an application to solve energy commitment problem," *International Journal of Intelligent Engineering & Systems*, vol. 13, no. 5, pp. 514-523, 2020.
[24] D. Freitas, L. G. Lopes, and F. Morgado-Dias, "Particle swarm optimization: a historical review up to the current development," *Entropy*, vol. 22, ID: 362, pp. 1-36, 2020.
[25] T. Guilmeau, E. Chouzenoux, and V. Elvira, "Simulated annealing: a review and a new scheme," *IEEE Statistical Signal Processing Workshop (SSP)*, Rio de Janeiro, Brazil, 2021.
[26] S. Harifi, M. Khalilian, J. Mohammadzadeh, and S. Ebrahimnejad, "Emperor penguins colony: a new metaheuristic algorithm for optimization," *Evolutionary Intelligence*, vol. 12, pp. 211-226, 2019.
[27] K. Hussain, M. N. M. Salleh, S. Cheng, and R. Naseem, "Common benchmark functions for metaheuristic evaluation: a review," *International Journal on Informatics Visualization*, vol. 1, no. 4, pp. 218-223, 2017.
[28] Y. Qawqzeh, M.T. Alharbi, A. Jaradat, K.N.A. Sattar, "A review of swarm intelligence algorithms deployment for scheduling and optimization in cloud computing environments," *PeerJ Computer Science*, pp. 1-17, 2021.
[29] P. D. Kusuma and M. Kallista, "Collaborative vendor managed inventory model by using multi agent system and continuous review (r, Q) replenishment policy," *Journal of Applied Engineering Science*, vol. 20, no. 1, pp. 254-263, 2022.
[30] F. Casino, T. K. Dasaklis, and C. Patsakis, "Enhanced vendor-managed inventory through blockchain," *Proc. of 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, Piraeus, Greece, 2019.
[31] W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67-82, 1997.
[32] D. Gupta and H. Singh, "A heuristic approach to n x m flow shop scheduling problem in which processing times are associated with their respective probabilities with no-idle constraint," *ISRN Operations Research*, ID: 948541, pp. 1-9, 2013.
[33] B. Naderi and R. Ruiz, "A scatter search algorithm for the distributed permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 239, no. 2, pp. 323-334, 2014.
[34] K. Geng, C. Ye, L. Cao, and L. Liu, "Multi-objective reentrant hybrid flowshop scheduling with machines turning on and off control strategy using improved multi-verse optimizer algorithm," *Mathematical Problems in Engineering*, ID: 2573873, pp. 1-19, 2019.
[35] H. Hernandez-Perez and J. J. Salazar-Gonzalez, "A branch-and-cut algorithm for the split-demand one-commodity pickup-and-delivery travelling salesman problem," *European Journal of Operational Research*, vol. 297, no. 2, pp. 467-483, 2022.
[36] Y. Wang, Q. Li, X. Guan, J. Fan, Y. Liu, and H. Wang, "Collaboration and resource sharing in the multidepot multiperiod vehicle routing problem with pickups and deliveries," *Sustainability*, vol. 12, ID: 5966, pp. 1-32, 2020.
[37] 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.
[38] 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.
[39] J. Yang, X. Zhang, S. Fu, and Y. Hu, "A hybrid harmony search algorithm based on firefly algorithm and Boltzmann machine," *IAENG International Journal of Computer Science*, vol. 49, no. 1, pp. 241-251, 2022.
[40] B. Wang, B. Lv, and Y. Song, "A hybrid genetic algorithm with integer coding for task offloading in edge-cloud cooperative computing," *IAENG International Journal of Computer Science*, vol. 49, no. 2, pp. 503-510, 2022.
[41] A. Ebrahimnejad, M. Enayattabr, H. Motameni, and H. Garg, "Modified artificial bee colony algorithm for solving mixed interval-valued fuzzy shortest path problem," *Complex & Intelligent Systems*, vol. 7, pp. 1527-1545, 2021.
[42] X. Zong, A. Liu, C. Wang, Z. Ye, and J. Du, "Indoor evacuation model based on visual-guidance artificial bee colony algorithm," *Building Simulation*, vol. 15, pp. 645-658, 2022.
[43] R. S. Raghav, U. Prabu, M. Rajeswari, D. Saravanan, and K. Thirugnanasambandam, "Cuddle death algorithm using ABC for detecting unhealthy nodes in wireless sensor networks," *Evolutionary Intelligence*, vol. 15, pp. 1605-1617, 2022.

Purba Daru Kusuma is an assistant professor in computer engineering in Telkom University, Indonesia. He received his bachelor and master's degrees in electrical engineering from Bandung Institute of Technology, Indonesia. He received his doctoral degree in computer science from Gadjah Mada University, Indonesia. His research interests are in artificial intelligence, machine learning, and operational research. He currently becomes a member of IAENG.

Dimas Adiputra is a lecturer and researcher in electrical engineering in Institut Teknologi Telkom Surabaya, Indonesia. He received her bachelor's program in electrical engineering from President University, Indonesia. He received his master's and doctoral degree in electrical engineering from Universiti Teknologi Malaysia, Malaysia. His research interests are in control system, internet of things, and healthcare device.