Partial Leader Optimizer

Purba Daru Kusuma [a,*], Faisal Candrasyah Hasibuan [a]

[a] *Computer Engineering, Telkom University, Buah Batu Street, Bandung, Indonesia*
Corresponding author: [*]purbodaru@telkomuniversity.ac.id

***Abstract*— A new swarm intelligence-based metaheuristic optimizer, namely Partial Leader Optimizer (PLO), is presented. PLO contains several autonomous agents that represent the solution. The best solution represents collective intelligence, i.e., the leader. PLO has distinct mechanics in finding the acceptable solution during the given iteration. Every agent moves to a specified target in every iteration. Two options can be chosen to determine the target. First, the target is calculated by pushing the virtual best solution away from the corresponding agent. Second, the target is randomly chosen within the solution space. This target selection is conducted stochastically based on the threshold that is set manually before the iteration. Then, several candidates are generated between the target and the agent's current location. The distance between adjacent candidates is the same. The agent moves to the best candidate and updates the best solution. Simulation is implemented to observe and analyze the PLO's performance. The well-known 23 benchmark functions are used as the optimization problems. In this simulation, PLO is benchmarked with marine predator algorithm (MPA), particle swarm optimization (PSO), average subtraction-based optimizer (ASBO), slime mold algorithm (SMA), and pelican optimization algorithm (POA). The result shows that PLO is competitive compared to these algorithms, especially in solving fixed-dimension multimodal functions. PLO is better than PSO, MPA, SMA, ASBO, and POA in optimizing 22, 19, 18, 9, and 20 functions out of 23, respectively.**

***Keywords*— Metaheuristic; swarm intelligence; quantitative optimization.**

Manuscript received 8 Aug. 2022; revised 7 Nov. 2022; accepted 26 Dec. 2022. Date of publication 31 Aug. 2023.

I. INTRODUCTION

Optimization is a broad subject that spreads widely in many areas, such as operations research, engineering, finance, computation, telecommunication, and many others. Its popularity comes from its primary characteristic of achieving optimal results within limited resources. The optimal result can be considered as minimizing cost or maximizing revenue/profit. The examples are as follows. Wu et al. [1] proposed an optimization model for inventory routing problems to minimize transportation costs, i.e., fuel consumption. Miyata and Nagano [2] proposed an optimization model for the distributed flow shop to minimize the make-span. This model is optimized using the variable neighborhood search (VNS) and iterated greedy algorithm [2]. Othman et al. [3] proposed a new method by hybridizing the water flow algorithm and VNS to solve the classic traveling salesman problem. Fathollahi-Fard et al. [4] proposed an optimization model in the home healthcare system to optimize the total cost and unemployment time and maintain continuity. Mokhtari et al. [5] proposed an optimization model to solve the university course timetabling problem in the postgraduate course. Shi [6] optimized the 5G network using Viterbi and Bayesian algorithms to meet the increasing network demand within the given bandwidth. Zhao et al. [7] proposed an optimized urban rail transit using a genetic algorithm for passengers' travel time and cost and operational cost.

Swarm intelligence is a popular method used extensively in many optimization studies. Swarm intelligence is part of a metaheuristic algorithm. As a metaheuristic algorithm, swarm intelligence uses a stochastic approach so that it tries to find an acceptable or high-quality solution without guaranteeing the optimal global solution [8]. In general, swarm intelligence consists of several autonomous and distributed agents that work independently to find the best solution without centralized coordination [9]. PSO and ant colony optimization (ACO) are popular swarm intelligence algorithms.

Many shortcoming metaheuristics algorithms were developed based on the swarm intelligence mechanism. In general, these shortcoming algorithms use a leader or several leaders as a reference to improve the solution. PSO becomes the early algorithm that adopts this mechanism by using the global and local best solutions as references. Marine predator

algorithm (MPA) and grey wolf optimizer (GWO) are popular algorithms adopting the leader concept. In MPA, the predator represents the local leader and interacts with its corresponding prey [10]. In GWO, the three best solutions represent the leaders so that the entire wolves move toward the resultant of these leaders [11].

Many shortcoming algorithms adopt the leader concepts in many ways. Several algorithms, such as Komodo Mlipir and Red Deer Algorithms (RDA), combine the leader concept and cross-over. In KMA, the male dragons represent the leaders [12]. The big male dragons crawl toward the better big male dragons and avoid the worse big male dragons [12]. The female dragons cross over with the highest quality male dragon [12]. The small male dragons follow the big male dragons [12]. In RDA, the male commanders represent the leaders, creating a group of harems [13]. Several other shortcoming algorithms are northern goshawk optimization (NGO) [9], golden search optimization algorithm (GSOA) [14], pelican optimization algorithm (POA) [15], hybrid leader-based optimization (HLBO) [16], three influential members-based optimizer (TIMBO) [17], tunicate swarm algorithm (TSA) [18], squirrel search optimizer (SSO) [19], butterfly optimization algorithm (BOA) [20], multi leader optimizer (MLO) [21], and so on. In POA, every iteration generates a randomized leader, and entire agents try to follow this leader [15]. In NGO, the leader is selected randomly from the population [NGO]. This concept is like MPA during the eddy formation [10]. In GSOA, the best solution becomes the leader, and the agent moves toward this leader based on sinusoid calculation [14]. In MLO, several best agents become the leaders, and the number of leaders is set manually [21]. This mechanism can be seen as a dynamic version of GWO.

All agents will generally follow the leader in the swarm intelligence-based metaheuristic algorithms. This leader can be a single global leader, a single local leader, a combination between the global leader and local leader, or several global leaders. In some algorithms, random movement within the solution space is conducted only if the leader-guided movement fails to improve the agent's current solution. On the other hand, swarm intelligence-based algorithms where not all agents follow the leader are hard to find.

There are also various stochastic mechanisms when moving toward a leader. Many algorithms work uniformly and randomly. Some other algorithms choose a normal distribution. There are few algorithms to choose Levy Flight or Brownian motion. Most algorithms generate only one replacement candidate.

Based on this circumstance, this work presents a novel leader-based metaheuristic algorithm, a Partial Leader Optimizer (PLO). The concept is that not all agents refer to the leader, i.e., the best solution, to improve their current solution. Meanwhile, some agents refer to specific randomized solutions within the solution space. Moreover, this work proposes multiple solution candidates rather than single or static candidates for each agent, like in most metaheuristic algorithms.

The main contribution of this work is that in PLO, only several agents will conduct the leader-guided movement. Meanwhile, the other ones will conduct randomized movements. Due to the stochastic mechanism, this role will be shuffled in every iteration.

This paper is organized as follows. Section one presents the background; review of swarm intelligence, especially the shortcoming metaheuristic algorithms; the research objective; and the contribution of this work. Section two presents the proposed model, which consists of the concept, algorithm, and mathematical model of PLO and the simulation scenario. Section three presents the simulation result and discusses the in-depth analysis and findings regarding the result. Finally, section four summarizes the conclusion and future research potential.

II. Material and Method

A. Proposed Model

The model of PLO is presented in three parts: concept, algorithm, and mathematical model. The concept represents the mechanics of the exploration-exploitation strategy and its reasoning. The algorithm represents procedural formalization and is presented in pseudocode form. The mathematical model represents the more detailed formalization of each process conducted in the algorithm.

Swarm intelligence is the basis of the PLO algorithm. There are a specific number of autonomous agents in it. These agents stand in for the solution. Its main purpose is to find the most optimal solution within the solution space. Due to its autonomy, each agent moves within the solution space independently based on its perception of its environment. Each agent tries to find a better solution in every iteration. Besides, there is collective intelligence, namely, the best solution. The best solution is the solution whose fitness is the best among agents. This best solution becomes the final at the end of the iteration. As a metaheuristic algorithm, PLO is divided into two phases. Phase one is initialization. Phase two is iteration. In the initialization, all agents are randomized within the solution space. This distribution follows a uniform distribution. In the iteration, each agent tries to improve its solution quality.

This algorithm is called a partial leader optimizer because the best solution (leader) is not guaranteed to guide the improvement. In every iteration, an agent will determine its target. There are two options regarding this process. The first option calculates the target by pushing the virtual best solution away from the agent's solution. The idea is that the best solution is currently better than the agent's current solution. So, there is an opportunity to improve the best and the agent's current solution by pushing the best solution away from the agent's current solution. The target is randomized within the solution space in the second option. This option is considered because there is no guarantee that the first option will improve the solution. These options are chosen stochastically by each agent based on a certain threshold. So, in every iteration, some agents may choose the first option while others choose the second one.

After the target is chosen, the agent will try to find a better solution along its current location to the target. In this process, the agent will move in several steps to its target. The step size is equal. The optimal solution may be laid between the agent's current solution and the target.

Every time the agent moves to a new step, the agent will update its solution. If this new step is better than the agent's current solution, this new solution replaces the agent's old solution. This process is conducted until the last step. After this process ends, the best solution is updated by comparing the best and the agent's solutions. If the agent's new solution is better than the best solution, then this agent's new solution becomes the best solution.

This simple concept is then transformed into the algorithm. Before explained further, there are annotations used in this work. These annotations are as follows. Meanwhile, the algorithm of PLO is presented in algorithm 1.

b_l	lower boundary
b_u	upper boundary
c	candidate
C	set of candidates
i	candidate index
x	solution
x_{best}	the best solution
x_{target}	target
X	set of solutions
t	iteration
t_{max}	maximum iteration
r	threshold

algorithm 1: Partial Leader Optimizer

```
1   output: x_best
2   begin
3     for all X
4       initialize x using (1)
5       update x_best using (2)
6     end
7     for t = 1 to t_max
8       for all X
9         calculate x_target using (3)
10        for i = 1 to n(C)
11          calculate c_i using (4)
12          update x using (5)
13        end
14        update x_best using (2)
15      end
16    end
17  end
```

Below is the explanation of algorithm 1. Line 1 states that the best solution becomes the output of the algorithm. Line 3 to line 6 represents the initialization phase. Line 7 to line 16 represents the iteration phase. The initialization phase consists of two processes for all solutions: setting up the initial solution (line 4) and updating the best solution (line 5). The iteration phase contains two loops. The outer loop iterates from the first iteration to the maximum iteration.

Meanwhile, the inner loop iterates for all agents. In the iteration phase, there are three sequential processes conducted for every agent: determining the target (line 9), conducting multiple steps (line 10 to line 13), and updating the best solution (line 14). There are two processes conducted within the multiple-step movement. The first is determining the movement candidate (line 13), and the second is updating the current solution based on the generated candidate (line 14).

The formalization of the initialization phase is presented in (1) and (2). Equation (1) states that the initial solution is randomized within the solution space. Uniform distribution is applied in this process. The solution space is limited by using the lower and upper boundaries. Equation (2) is used for the best solution updating process. If the new solution is better than the best solution, this solution will replace the current best solution to become the new one. Otherwise, the best solution remains the same.

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

$$x_{best}{}' = \begin{cases} x, f(x) < f(x_{best}) \\ x_{best}, else \end{cases} \tag{2}$$

Equation (3) states two possible ways or options to determine the target. The first option is pushing the best solution away from the related one. The movement step size is uniformly randomized between zero and the gap between the best and related solutions. The second option is to generate the target randomly within the solution space. The selection is conducted by generating a random number ranging from 0 to 1. The first option is chosen if this number is below the threshold. Otherwise, the second option is chosen.

$$x_{tar} = \begin{cases} x_{best} + U(0,1).(x_{best} - x), U(0,1) < r \\ U(b_l, b_u), else \end{cases} \tag{3}$$

Equation (4) states that the candidate solution is between the current and target solutions. The distance between the adjacent candidates is equal. The earlier candidates are closer to the current solution. Meanwhile, the later candidates are closer to the target.

$$c_i = x + \left(\frac{i}{n(C)}\right).(x_{tar} - x) \tag{4}$$

Equation (5) is used for the solution updating process. If the candidate is better than the solution, it will replace the current solution to become the new one. Otherwise, the solution remains the same.

$$x' = \begin{cases} c_i, f(c_i) < f(x) \\ x, else \end{cases} \tag{5}$$

Based on the explanation above, the algorithm complexity of PLO is shown as $O(t_{max}.n(X).n(C))$. It means the algorithm complexity is linear to the maximum iteration, the population size, and the number of candidates.

B. Simulation

PLO is then implemented into the simulation so that its performance can be evaluated. In this work, there are three simulations. The first simulation is used to analyze the performance of PLO in solving the 23 benchmark functions. This first simulation compares PLO with five other metaheuristic algorithms: PSO, MPA, SMA, ASBO, and POA. The second and third simulations are conducted to analyze the algorithm's sensitivity. The second simulation is conducted to analyze the sensitivity of the maximum iteration, and the last one is conducted to analyze the sensitivity of the threshold.

The 23 benchmark functions are selected based on two reasons. The first reason is that this algorithm represents various problems, from unimodal and multimodal problems, low dimension to big dimension problems, and narrow solution space to large solution space problems. These 23 functions are divided into three categories. These functions

are distributed into seven high-dimension unimodal functions (function 1 to function 7), six high-dimension multimodal functions (function 8 to function 13), and ten fixed-dimension multimodal functions (function 14 to function 23). A detailed description of these 23 benchmark functions is presented in Table 1. Second, these 23 functions have been used extensively in many studies proposing new metaheuristic algorithms.

TABLE I
BENCHMARK FUNCTIONS

F	Function	Solution space	Dim
1	Sphere	[-100, 100]	20
2	Schwefel 2.22	[-100, 100]	20
3	Schwefel 1.2	[-100, 100]	20
4	Schwefel 2.21	[-100, 100]	20
5	Rosenbrock	[-30, 30]	20
6	Step	[-100, 100]	20
7	Quartic	[-1.28, 1.28]	20
8	Schwefel	[-500, 500]	20
9	Ratsrigin	[-5.12, 5.12]	20
10	Ackley	[-32, 32]	20
11	Griewank	[-600, 600]	20
12	Penalized	[-50, 50]	20
13	Penalized 2	[-50, 50]	20
14	Shekel Foxholes	[-65, 65]	2
15	Kowalik	[-5, 5]	4
16	Six Hump Camel	[-5, 5]	2
17	Branin	[-5, 5]	2
18	Goldstein-Price	[-2, 2]	2
19	Hartman 3	[1, 3]	3
20	Hartman 6	[0, 1]	6
21	Shekel 5	[0, 10]	4
22	Shekel 7	[0, 10]	4
23	Shekel 10	[0, 10]	4

Some reasons for choosing the five algorithms for comparison are as follows. PSO is an early metaheuristic algorithm that uses swarm intelligence. PSO also becomes the foundation for the development of a lot of later metaheuristic algorithms. Moreover, because of its popularity, many optimization studies used, modified, or combined PSO. The example is as follows. Liu et al. [22] used PSO to improve the accuracy of the rock slope slip simulation. Gao et al. [23] combined PSO with Levy flight, also used in MPA, to solve high latency issues in the mobile cloud computing system. Habib et al. [24] used the modified PSO (MPSO) to reduce the sidelobe level (SSL) in beam pointing for uniform hexagonal array (UHA) antennas. Many studies used and modified MPA and SMA are the popular shortcoming metaheuristic algorithms. The example is as follows. Abdel-Basset [25] used MPA in proposing a task scheduling model in the fog computing environment to improve the quality of services. Liu et al. [26] combined SMA and ACO to solve the classic traveling salesman problem, arguing that SMA can cover the disadvantage of ACO in the easiness of falling into the optimal local trap. Dhawale et al. [27] improved the basic SMA by combining it with sinusoid chaotic behavior to become an algorithm, namely chaotic SMA (CSMA).

On the other hand, Altay [28] developed another chaotic mechanism for essential SMA by applying ten chaotic maps, such as the Chebyshev map, circle map, Gauss map, and others. Al-qanees et al. [29] used MPA to improve the adaptive neuro-fuzzy inference system (ANFIS). This system is implemented in the COVID-19 confirmed cases forecasting system [29]. Finally, ASBO and POA represented the shortcoming of metaheuristic algorithms. These algorithms are firstly introduced in 2022. Although these two algorithms show outstanding performance, studies that use these algorithms are still hard to find.

This first simulation is conducted based on several settings. The maximum iteration and population size are 100 and 20, respectively. In PLO, the number of candidates is ten, and the threshold is 0.5. In PSO, all weights are 0.1. In MPA, the FAD is 0.5. There is not any adjusted parameter in ASBO and POA.

III. RESULT AND DISCUSSION

This section presents the simulation result and the in-depth analysis of the result and findings. Table 2 to Table 4 presents the result of the first simulation. Table 2 presents the average fitness score, while Table 3 presents the standard deviation. Then, Table 4 presents the number of functions where PLO is better than other algorithms. In Table 4, the data is divided based on the groups. Table 5 shows the performance of PLO within the various maximum iteration. Table 6 shows the performance of PLO within the various threshold.

Table 2 shows that PLO performs well. It can find an acceptable solution for the 23 functions. Moreover, PLO can find the optimal global solution for seven functions: Schwefel 2.22, Shekel Foxholes, Six Hump Camel, Branin, Goldstein-Price, Shekel 7, and Shekel 10. Among these seven functions, one function is in the first group, while the six functions are in the third group. PLO also outperforms all sparing algorithms in solving eight functions: Shekel Foxholes, Kowalik, Six Hump Camel, Branin, Hartman 6, Shekel 5, Shekel 7, and Shekel 10.

Table 3 shows that the performance variance of PLO varies depending on the problem it tries to solve. PLO performs zero variance in solving two functions: Goldstein-Price and Schwefel 2.22. Meanwhile, PLO performs very low variance in optimizing Shekel Foxholes, Branin, Six Hump Camel, Shekel 7, and Shekel 10.

Table 4 shows that PLO is competitive compared with the five algorithms. PSO becomes the most straightforward algorithm to beat. On the other hand, ASBO becomes the most challenging algorithm to beat. PLO is very superior in solving the functions in the third group. Meanwhile, PLO is inferior in the first and second groups, especially compared to ASBO. PLO outperforms PSO, MPA, SMA, ASBO, and POA in solving 22, 19, 18, 9, and 20 functions.

In simulation two, the maximum iteration consists of three values. The first value is 25, the second is 50, and the third is 75. These three values are less than 100, as set in the first simulation. Table 5 shows that, in general, the convergence of the optimization process is achieved in low maximum iteration. There are 12 functions where the convergence is achieved when the maximum iteration is set at 25.

TABLE II
SIMULATION RESULT ON 23 BENCHMARK FUNCTIONS (AVERAGE FITNESS SCORE)

F	PSO	MPA	ASBO	SMA	POA	PLO	Better Than
1	$2.773x10^{3}$	$3.064x10^{2}$	$5.342x10^{-23}$	$1.283x10^{3}$	$1.589x10^{4}$	1.952	PSO, MPA, SMA, POA
2	0	0	0	0	0	0	-
3	$8.197x10^{3}$	$9.527x10^{2}$	$1.064x10^{-3}$	$6.542x10^{3}$	$2.132x10^{4}$	$1.733x10^{2}$	PSO, MPA, SMA, POA
4	$2.348x10^{1}$	1.400	$2.998x10^{-9}$	$1.770x10^{1}$	$5.416x10^{1}$	8.531	PSO, SMA, POA
5	$1.095x10^{6}$	$6.654x10^{1}$	$1.854x10^{1}$	$5.714x10^{5}$	$2.323x10^{7}$	$1.513x10^{2}$	PSO, SMA, POA
6	$2.094x10^{3}$	$2.885x10^{2}$	$6.335x10^{-2}$	$7.670x10^{2}$	$1.501x10^{4}$	1.099	PSO, MPA, SMA, POA
7	$3.035x10^{-1}$	$6.838x10^{-2}$	$9.218x10^{-3}$	$2.371x10^{1}$	7.427	$2.598x10^{-2}$	PSO, MPA, SMA, POA
8	$-2.231x10^{3}$	$-2.679x10^{3}$	$-3.385x10^{3}$	$-5.732x10^{3}$	$-2.896x10^{3}$	$-5.181x10^{3}$	PSO, MPA, ASBO, POA
9	$1.457x10^{2}$	$7.455x10^{1}$	3.560	$1.578x10^{1}$	$1.914x10^{2}$	$3.037x10^{1}$	PSO, MPA
10	$1.108x10^{1}$	5.870	1.677	7.025	$1.863x10^{1}$	4.823	PSO, MPA, SMA, POA
11	$2.480x10^{1}$	3.977	$8.692x10^{-2}$	8.787	$1.500x10^{2}$	$4.757x10^{-1}$	PSO, MPA, SMA, POA
12	$3.744x10^{4}$	5.580	$2.586x10^{-3}$	$4.613x10^{3}$	$2.214x10^{7}$	3.817	PSO, MPA, SMA, POA
13	$5.220x10^{5}$	$2.741x10^{2}$	5.148	$5.454x10^{5}$	$7.594x10^{7}$	$1.221x10^{1}$	PSO, MPA, SMA, POA
14	4.991	5.319	1.103	1.310	1.785	$9.980x10^{-1}$	PSO, MPA, ASBO, SMA, POA
15	$1.989x10^{-2}$	$4.429x10^{-3}$	$7.526x10^{-2}$	$9.362x10^{-2}$	$2.997x10^{-3}$	$4.700x10^{-4}$	PSO, MPA, ASBO, SMA, POA
16	-1.030	-1.022	$-7.618x10^{-2}$	$-3.286x10^{-2}$	-1.029	-1.032	PSO, MPA, ASBO, SMA, POA
17	1.504	$5.816x10^{-1}$	$6.438x10^{-1}$	$6.312x10^{-1}$	$4.009x10^{-1}$	$3.981x10^{-1}$	PSO, MPA, ASBO, SMA, POA
18	$1.097x10^{1}$	4.668	3.000	3.000	3.046	3.000	PSO, MPA, POA
19	$-1.278x10^{-2}$	-3.827	$-4.954x10^{-2}$	$-4.954x10^{-2}$	$-4.954x10^{-2}$	$-4.778x10^{-2}$	PSO
20	-2.464	-1.957	-1.223	-1.596	-2.957	-3.308	PSO, MPA, ASBO, SMA, POA
21	-4.419	-1.849	-9.137	-7.074	-3.085	-9.850	PSO, MPA, ASBO, SMA, POA
22	-4.109	-1.798	-8.214	-6.513	-3.400	$-1.040x10^{1}$	PSO, MPA, ASBO, SMA, POA
23	-5.048	-1.856	-8.994	-8.012	-3.838	$-1.054x10^{1}$	PSO, MPA, ASBO, SMA, POA

TABLE III
SIMULATION RESULT ON 23 BENCHMARK FUNCTIONS (STANDARD DEVIATION)

F	PSO	MPA	ASBO	SMA	POA	PLO
1	$8.032x10^{2}$	$1.785x10^{2}$	$1.118x10^{-22}$	$6.718x10^{2}$	$2.511x10^{3}$	2.734
2	0	0	0	0	0	0
3	$1.954x10^{3}$	$5.431x10^{2}$	$2.816x10^{-3}$	$3.465x10^{3}$	$6.512x10^{3}$	$1.590x10^{2}$
4	5.632	1.208	$2.298x10^{-9}$	$1.058x10^{1}$	5.669	2.588
5	$1.191x10^{6}$	$9.673x10^{1}$	$2.230x10^{-2}$	$6.293x10^{5}$	$1.158x10^{7}$	$1.048x10^{2}$
6	$1.018x10^{3}$	$1.393x10^{2}$	$3.280x10^{-2}$	$4.863x10^{2}$	$3.565x10^{3}$	1.748
7	$1.767x10^{-1}$	$4.041x10^{-2}$	$4.229x10^{-3}$	$2.580x10^{1}$	3.653	$1.295x10^{-2}$
8	$-3.927x10^{2}$	$3.205x10^{2}$	$3.013x10^{2}$	$2.844x10^{2}$	$4.203x10^{2}$	$8.645x10^{2}$
9	$1.510x10^{1}$	$2.519x10^{1}$	1.529	5.024	$2.112x10^{1}$	9.787
10	$9.293x10^{-1}$	1.255	$4.256x10^{-1}$	1.969	$5.316x10^{-1}$	1.166
11	8.509	1.309	$6.292x10^{-2}$	6.457	$3.190x10^{1}$	$3.511x10^{-1}$
12	$9.085x10^{4}$	2.208	$4.439x10^{-3}$	$1.329x10^{4}$	$1.218x10^{7}$	3.652
13	$8.018x10^{5}$	$5.539x10^{2}$	1.126	$9.580x10^{5}$	$2.453x10^{7}$	$1.214x10^{1}$
14	3.560	3.059	$3.134x10^{-1}$	1.141	1.085	$3.448x10^{-16}$
15	$2.321x10^{-2}$	$3.630x10^{-3}$	$3.328x10^{-2}$	$3.665x10^{-2}$	$2.121x10^{-3}$	$4.073x10^{-4}$
16	$4.588x10^{-3}$	$9.428x10^{-3}$	$1.631x10^{-1}$	$8.057x10^{-2}$	$1.721x10^{-3}$	$2.293x10^{-16}$
17	2.993	$1.587x10^{-1}$	$1.844x10^{-1}$	$4.364x10^{-2}$	$2.523x10^{-3}$	$5.722x10^{-17}$
18	$2.046x10^{1}$	1.244	0	0	$5.324x10^{-2}$	0
19	$-3.973x10^{-2}$	$1.253x10^{-1}$	$1.433x10^{-17}$	$1.436x10^{-17}$	$1.436x10^{-17}$	$7.667x10^{-3}$
20	$3.780x10^{-1}$	$4.149x10^{-1}$	$4.026x10^{-1}$	$4.806x10^{-1}$	$1.676x10^{-1}$	$3.959x10^{-2}$
21	2.580	$6.200x10^{-1}$	2.104	2.752	1.117	1.236
22	3.543	$5.121x10^{-1}$	2.696	2.648	1.386	$8.383x10^{-6}$
23	3.126	$5.614x10^{-1}$	2.529	3.199	1.529	$3.669x10^{-15}$

TABLE IV
HEAD-TO-HEAD COMPARISON

Algorithm	Number of Functions that PLO beats			
	1st Group	2nd Group	3rd Group	Total
PSO	6	6	10	22
MPA	4	6	9	19
ASBO	0	1	8	9
SMA	6	4	8	18
POA	6	5	9	20

TABLE V
RELATION BETWEEN MAXIMUM ITERATION AND THE PERFORMANCE

F	Average Fitness Score		
	t_{max} = 25	t_{max} = 50	t_{max} = 75
1	$4.866x10^{1}$	9.479	1.541
2	0	0	0
3	$5.603x10^{2}$	$3.259x10^{2}$	$2.522x10^{2}$
4	9.990	9.298	8.887
5	$1.878x10^{3}$	$2.790x10^{2}$	$3.331x10^{2}$
6	$3.628x10^{1}$	6.966	2.606
7	$6.026x10^{-2}$	$5.603x10^{-2}$	$3.193x10^{-2}$
8	$-5.402x10^{3}$	$-5.676x10^{3}$	$-5.661x10^{3}$
9	$4.158x10^{1}$	$3.030x10^{1}$	$3.214x10^{1}$
10	5.265	5.101	5.250
11	1.420	$8.983x10^{-1}$	$5.718x10^{-1}$
12	4.394	4.560	3.310

F	Average Fitness Score		
	t_{max} = 25	t_{max} = 50	t_{max} = 75
13	2.781×10^{1}	2.564×10^{1}	1.533×10^{1}
14	1.224	9.980×10^{-1}	9.980×10^{-1}
15	2.042×10^{-3}	6.639×10^{-4}	6.414×10^{-4}
16	-1.032	-1.032	-1.032
17	3.981×10^{-1}	3.981×10^{-1}	3.981×10^{-1}
18	3.000	3.000	3.000
19	-4.650×10^{-2}	-4.954×10^{-2}	-4.636×10^{-2}
20	-3.322	-3.279	-3.279
21	-8.226	-9.231	-8.434
22	-7.778	-9.466	-9.217
23	-7.761	-9.894	-1.029×10^{1}

TABLE VI
RELATION BETWEEN THRESHOLD AND THE PERFORMANCE

F	Average Fitness Score		
	r= 0.25	r= 0.5	r = 0.75
1	**5.714**	1.952	4.370
2	**0**	**0**	**0**
3	1.846×10^{2}	**1.733×10^{2}**	2.283×10^{2}
4	**7.232**	8.531	9.521
5	3.421×10^{2}	**1.513×10^{2}**	3.196×10^{2}
6	4.938	**1.099**	3.766
7	**1.659×10^{-2}**	2.598×10^{-2}	4.577×10^{-2}
8	**-5.526×10^{3}**	-5.181×10^{3}	-5.460×10^{3}
9	**2.639×10^{1}**	3.037×10^{1}	3.625×10^{1}
10	**4.300**	4.823	5.732
11	**4.600×10^{-1}**	4.757×10^{-1}	5.467×10^{-1}
12	**1.500**	3.817	3.581
13	**9.489**	1.221×10^{1}	1.692×10^{1}
14	**9.980×10^{-1}**	**9.980×10^{-1}**	1.043
15	**4.209×10^{-4}**	4.700×10^{-4}	6.603×10^{-4}
16	**-1.032**	**-1.032**	**-1.032**
17	**3.981×10^{-1}**	**3.981×10^{-1}**	**3.981×10^{-1}**
18	**3.000**	**3.000**	**3.000**
19	-4.846×10^{-2}	-4.778×10^{-2}	**-4.954×10^{-1}**
20	-3.291	**-3.308**	-3.279
21	-9.814	-9.850	-8.321
22	-1.039×10^{1}	**-1.040×10^{1}**	-8.830
23	-1.051×10^{1}	**-1.054×10^{1}**	-9.600

In the third simulation, there are three values for the threshold. The first value is 0.25, the second is 0.5, and the third is 0.75. The best result is written in bold font in Table 6. Table 6 shows the algorithm's performance in response to different threshold values varies. The best outcome is found for six functions at all threshold levels. When the threshold is low, ten functions produce their best results. There are six functions where a moderate threshold yields the best results. When the threshold is high, only one function produces the best results. Meanwhile, there is one function where the best result is obtained when the threshold is low or moderate.

Although the result has shown that the performance of PLO is acceptable, it is not wise to conclude that PLO is superior to the defeated algorithms, such as PSO, MPA, SMA, and POA. As explained in the no-free-lunch theory, there is not any perfect algorithm. On the other hand, the algorithm's performance lays on the problem it tries to tackle. Besides, many metaheuristic algorithms depend on their adjusted parameters to control their performance. For example, the dominant exploitation strategy may be better at solving unimodal problems, while the dominant exploration strategy may be better at solving multimodal problems. As indicated in Table 6, a low threshold may be suitable for some problems, while a moderate threshold may be suitable for others.

The result in Table 6 also shows that diversifying agent roles is essential. Too many agents that conduct leader-guided movement ends with a less satisfying result. On the other hand, balancing leader-guided and randomized movements ends with better results.

Traditional algorithms, such as the variable neighborhood search (VNS), tabu search (TS), genetic algorithm (GA), and PSO, are still employed and developed in various optimization research even if many shortcoming methods have outperformed them. The example is as follows. Rejer and Jankowski [30] improved the basic GA with an aggressive mutation method, a fast GA with aggressive mutation (FGAAM), to decrease the time feature finding time for feature selection. Sajadi and Ahmadi [31] combined GA with vibration damping optimization (VDO) to optimize the inventory management of perishable products. Krityakierne et al. [32] used TS to optimize home healthcare routing and scheduling. There are several reasons for this circumstance. First, these old-fashioned algorithms are battle-proven algorithms implemented in many optimization studies. Second, their mechanics are uncomplicated, so they can be combined with other algorithms to improve their performance and tackle their weaknesses, especially in avoiding the local trap. Third, many metaheuristic algorithms lay on the iteration and the population size. Improving the algorithm's performance by increasing the maximum iteration or population size is easy.

On the other hand, the intention of developing a new algorithm is still high. This circumstance also comes from several reasons. First, as stated in the no-free-lunch theory, no perfect method or algorithm is superior to solving all problems. On the contrary, there are many problems with specific circumstances (objective and constraint). Moreover, to date, many problems have become more complex than in the previous time. Second, there are a lot of mathematical solutions that can be used and have not been explored to construct many new algorithms.

IV. CONCLUSION

This work has proposed a new metaheuristic method, a partial leader optimizer (PLO). Based on the explanation, this algorithm is shown as a simple algorithm. Meanwhile, the simulation results show that PLO is competitive compared to PSO, MPA, ASBO, SMA, and POA. PLO is better than PSO, MPA, SMA, ASBO, and POA in finding the optimal solution of 22, 19, 18, 9, and 20 functions, respectively. The simulation result also shows that PLO can achieve an acceptable solution in the low iteration. Moreover, the low or moderate threshold is generally better than the high threshold in solving the benchmark functions.

Various approaches can continue this work. First, more studies to implement PLO in many real-world applications are needed to create a more comprehensive evaluation of the performance of PLO. Second, more studies to combine PLO with other algorithms, either the old-fashioned algorithms or the shortcoming ones, are also challenging.

ACKNOWLEDGMENT

Telkom University, Indonesia, financially supported this work.

REFERENCES

[1] W. Wu, W. Zhou, Y. Lin, Y. Xie, and W. Jin, "A hybrid metaheuristic algorithm for location inventory routing problem with time windows and fuel consumption," *Expert Syst Appl*, vol. 166, p. 114034, Mar. 2021, doi: 10.1016/j.eswa.2020.114034.

[2] H. H. Miyata and M. S. Nagano, "An iterated greedy algorithm for distributed blocking flow shop with setup times and maintenance operations to minimize makespan," *Comput Ind Eng*, vol. 171, p. 108366, Sep. 2022, doi: 10.1016/j.cie.2022.108366.

[3] M. R. Othman, Z. Ali Othman, A. I. Srour, and N. S. Sani, "A Hybrid Water Flow-Like Algorithm and Variable Neighbourhood Search for Traveling Salesman Problem," *Int J Adv Sci Eng Inf Technol*, vol. 9, no. 5, p. 1505, Oct. 2019, doi: 10.18517/ijaseit.9.5.7957.

[4] A. M. Fathollahi-Fard, A. Ahmadi, and B. Karimi, "Multi-Objective Optimization of Home Healthcare with Working-Time Balancing and Care Continuity," *Sustainability*, vol. 13, no. 22, p. 12431, Nov. 2021, doi: 10.3390/su132212431.

[5] M. Mokhtari, M. Vaziri Sarashk, M. Asadpour, N. Saeidi, and O. Boyer, "Developing a Model for the University Course Timetabling Problem: A Case Study," *Complexity*, vol. 2021, pp. 1–12, Dec. 2021, doi: 10.1155/2021/9940866.

[6] X. Shi, "A Method of Optimizing Network Topology Structure Combining Viterbi Algorithm and Bayesian Algorithm," *Wirel Commun Mob Comput*, vol. 2021, pp. 1–12, May 2021, doi: 10.1155/2021/5513349.

[7] J. Zhao, M. Ye, Z. Yang, Z. Xing, and Z. Zhang, "Operation optimizing for minimizing passenger travel time cost and operating cost with time-dependent demand and skip-stop patterns: Nonlinear integer programming model with linear constraints," *Transp Res Interdiscip Perspect*, vol. 9, p. 100309, Mar. 2021, doi: 10.1016/j.trip.2021.100309.

[8] J. Swan *et al.*, "Metaheuristics 'In the Large,'" *Eur J Oper Res*, vol. 297, no. 2, pp. 393–406, Mar. 2022, doi: 10.1016/j.ejor.2021.05.042.

[9] M. Dehghani, S. Hubalovsky, and P. Trojovsky, "Northern Goshawk Optimization: A New Swarm-Based Algorithm for Solving Optimization Problems," *IEEE Access*, vol. 9, pp. 162059–162080, 2021, doi: 10.1109/ACCESS.2021.3133286.

[10] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine Predators Algorithm: A nature-inspired metaheuristic," *Expert Syst Appl*, vol. 152, p. 113377, Aug. 2020, doi: 10.1016/j.eswa.2020.113377.

[11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[12] S. Suyanto, A. A. Ariyanto, and A. F. Ariyanto, "Komodo Mlipir Algorithm," *Appl Soft Comput*, vol. 114, p. 108043, Jan. 2022, doi: 10.1016/j.asoc.2021.108043.

[13] A. M. Fathollahi-Fard, M. Hajiaghaei-Keshteli, and R. Tavakkoli-Moghaddam, "Red deer algorithm (RDA): a new nature-inspired meta-heuristic," *Soft comput*, vol. 24, no. 19, pp. 14637–14665, Oct. 2020, doi: 10.1007/s00500-020-04812-z.

[14] M. Noroozi, H. Mohammadi, E. Efatinasab, A. Lashgari, M. Eslami, and B. Khan, "Golden Search Optimization Algorithm," *IEEE Access*, vol. 10, pp. 37515–37532, 2022, doi: 10.1109/ACCESS.2022.3162853.

[15] P. Trojovský and M. Dehghani, "Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications," *Sensors*, vol. 22, no. 3, p. 855, Jan. 2022, doi: 10.3390/s22030855.

[16] M. Dehghani and P. Trojovský, "Hybrid leader based optimization: a new stochastic optimization algorithm for solving optimization applications," *Sci Rep*, vol. 12, no. 1, p. 5549, Dec. 2022, doi: 10.1038/s41598-022-09514-0.

[17] F. Zeidabadi, M. Dehghani, and O. Malik, "TIMBO: Three Influential Members Based Optimizer," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 5, pp. 121–128, Oct. 2021, doi: 10.22266/ijies2021.1031.12.

[18] S. Kaur, L. K. Awasthi, A. L. Sangal, and G. Dhiman, "Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization," *Eng Appl Artif Intell*, vol. 90, p. 103541, Apr. 2020, doi: 10.1016/j.engappai.2020.103541.

[19] M. Suman, V. Sakthivel, and P. Sathya, "Squirrel Search Optimizer: Nature Inspired Metaheuristic Strategy for Solving Disparate Economic Dispatch Problems," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 5, pp. 111–121, Oct. 2020, doi: 10.22266/ijies2020.1031.11.

[20] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," *Soft comput*, vol. 23, no. 3, pp. 715–734, Feb. 2019, doi: 10.1007/s00500-018-3102-4.

[21] M. Dehghani *et al.*, "MLO: Multi Leader Optimizer," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 6, pp. 364–373, Dec. 2020, doi: 10.22266/ijies2020.1231.32.

[22] B. Liu, Z. Wang, and X. Zhong, "Particle Swarm Optimization Algorithm in Numerical Simulation of Saturated Rock Slope Slip," *Math Probl Eng*, vol. 2021, pp. 1–11, Mar. 2021, doi: 10.1155/2021/6682659.

[23] T. Gao, Q. Tang, J. Li, Y. Zhang, Y. Li, and J. Zhang, "A Particle Swarm Optimization With Lévy Flight for Service Caching and Task Offloading in Edge-Cloud Computing," *IEEE Access*, vol. 10, pp. 76636–76647, 2022, doi: 10.1109/ACCESS.2022.3192846.

[24] H. Mohammed Hussein, K. Katzis, L. P. Mfupe, and E. T. Bekele, "Performance Optimization of High-Altitude Platform Wireless Communication Network Exploiting TVWS Spectrums Based on Modified PSO," *IEEE Open Journal of Vehicular Technology*, vol. 3, pp. 356–366, 2022, doi: 10.1109/OJVT.2022.3191762.

[25] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-Aware Marine Predators Algorithm for Task Scheduling in IoT-Based Fog Computing Applications," *IEEE Trans Industr Inform*, vol. 17, no. 7, pp. 5068–5076, Jul. 2021, doi: 10.1109/TII.2020.3001067.

[26] M. Liu *et al.*, "A Slime Mold-Ant Colony Fusion Algorithm for Solving Traveling Salesman Problem," *IEEE Access*, vol. 8, pp. 202508–202521, 2020, doi: 10.1109/ACCESS.2020.3035584.

[27] D. Dhawale, V. K. Kamboj, and P. Anand, "An effective solution to numerical and multi-disciplinary design optimization problems using chaotic slime mold algorithm," *Eng Comput*, May 2021, doi: 10.1007/s00366-021-01409-4.

[28] O. Altay, "Chaotic slime mould optimization algorithm for global optimization," *Artif Intell Rev*, vol. 55, no. 5, pp. 3979–4040, Jun. 2022, doi: 10.1007/s10462-021-10100-5.

[29] M. A. A. Al-qaness, A. A. Ewees, H. Fan, L. Abualigah, and M. Abd Elaziz, "Marine Predators Algorithm for Forecasting Confirmed Cases of COVID-19 in Italy, USA, Iran and Korea," *Int J Environ Res Public Health*, vol. 17, no. 10, p. 3520, May 2020, doi: 10.3390/ijerph17103520.

[30] I. Rejer and J. Jankowski, "fGAAM: A fast and resizable genetic algorithm with aggressive mutation for feature selection," *Pattern Analysis and Applications*, vol. 25, no. 2, pp. 253–269, May 2022, doi: 10.1007/s10044-021-01000-z.

[31] S. J. Sajadi and A. Ahmadi, "An integrated optimization model and metaheuristics for assortment planning, shelf space allocation, and inventory management of perishable products: A real application," *PLoS One*, vol. 17, no. 3, p. e0264186, Mar. 2022, doi: 10.1371/journal.pone.0264186.

[32] T. Krityakierne, O. Limphattharachai, and W. Laesanklang, "Nurse-patient relationship for multi-period home health care routing and scheduling problem," *PLoS One*, vol. 17, no. 5, p. e0268517, May 2022, doi: 10.1371/journal.pone.0268517.