Cone Search: A Simple Metaheuristic Optimization Algorithm

Purba Daru Kusuma, *Member, IAENG*, Ratna Astuti Nugrahaeni, Ashri Dinimaharawati

***Abstract*—This paper demonstrates a novel simple metaheuristic algorithm, the cone search (CS). This name comes from the distinct strategy of CS in searching the sub-optimal solution. In the early iteration, its local search space is wide to facilitate exploration. The local search space is reduced linearly during the iteration so that the exploration changes to exploitation gradually. As a swarm intelligence, CS contains several autonomous agents and a collective intelligence called memory. This memory consists of several best solutions. In this work, CS is challenged to find the global optimal of 23 benchmark functions. In the simulation, CS is compared with four metaheuristic algorithms: particle swarm optimization (PSO), marine-predators algorithm (MPA), Komodo mlipir algorithm (KMA), and pelican optimization algorithm (POA). The result shows that CS performs well in solving these 23 functions. Moreover, it can find the global optimal of six fixed-dimension multimodal functions: Branin, six hump camel, Hartman 6, Shekel 5, Shekel 7, and Goldstein price. CS beats PSO, MPA, KMA, and POA in solving 23, 22, 21, 22 functions respectively.**

***Index Terms*—Metaheuristic, multi-agent, optimization, swarm intelligence.**

I. Introduction

METAHEURISTIC method is an approximate and stochastic based method that has been used extensively in many optimization problems, such as manufacturing, transportation, education, and so on. As an optimization method, its objective is to find the optimal solution within the solution space. In recent days, there are a lot of of metaheuristic algorithms. Every algorithm has its own strategy. Some algorithms are inherited or modified from the previous algorithms.

Ironically, there are two criticisms due to the massive development of new metaheuristic algorithms. The first critic is the use of metaphors, especially nature-based metaphors [1]. The use of metaphors might hide the true novel strategy of the algorithm. The second critic is beating competition among algorithms [2].

In the early era of metaheuristic algorithms, the development of algorithms was not so massive as today. Several well-known algorithms were developed during this period, such as evolutionary programming [3], genetic algorithm [4], simulated annealing [5], tabu search [6], particle swarm optimization [7], ant colony optimization [8], variable neighborhood search [9], and so on. There is a clear distinction among these algorithms although only a few algorithms were proposed in this period. From 2000 to 2010, the adoption of metaphors triggered many researchers to propose metaphors-based algorithms. There were many nature-inspired algorithms during this period, such as sheep-flocks heredity model [10], bacterial foraging optimization [11], invasive weed optimization [12], cat swarm optimization [13], firefly algorithm [14], paddy field algorithm [15], and so on. After 2010, there was a huge blow in the use of animals as metaphors of the algorithms. Several animal names were adopted as name of algorithms, such as wolf [16], penguins [17], grey wolf [18], chicken [19], buffalo [20], lion [21], dolphin [22], and so on. Besides animal and plant, several algorithms used other nature mechanics as metaphors, such as water [23], virus [24], cloud [25], gravity [26], and so on.

In the earlier era, many studies in metaheuristic algorithms focused on the objective. The main objectives of metaheuristic algorithm are to find the sub-optimal solution [27] and to escape from the local optimal trap [28]. This consideration is based on the nature of metaheuristic algorithm that adopts stochastic approach. As a stochastic-based algorithm, it does not ensure that the global optimal solution can be found [29]. This circumstance is different from the exact method where the global optimal solution is guaranteed to find [29]. Fortunately, the metaheuristic method is powerful and flexible enough to solve the large problem space and complex problems where the exact method is impossible to conduct because of the excessive computational consumption [29]. In metaheuristic algorithm, a better solution is found during the iteration.

The local optimal trap is a classic problem in metaheuristic studies. In general, improvement is still conducted in the metaheuristic process if the new solution is better than the current one. This improvement is usually called intensification or exploitation [30]. Local search is commonly used in this phase. The optimal solution is assumed to be found if a better solution cannot be found. Unfortunately, in many problems, namely multimodal or non-convex problems, the problem space contains many optimal solutions [31]. One solution is the global optimal solution while others are local optimal [31]. Diversification or exploration avoids this trap by finding other alternatives within the problem space [30].

As explained previously, many shortcoming algorithms

Manuscript received February 21, 2022; revised August 10, 2022. This work was supported and funded by Telkom University, Indonesia.

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

Ratna Astuti Nugrahaeni is a lecturer and researcher in computer engineering, Telkom University, Indonesia. (e-mail: ratnaan@telkomuniversity.ac.id).

Ashri Dinimaharawati is a lecturer and researcher in computer engineering, Telkom University, Indonesia. (e-mail: ashridini@telkomuniversity.ac.id).

focused on beating other previous algorithms rather than proposing new ways in finding the sub optimal solution and escaping from the local optimal. The examples are as follows. Suyanto et al. [32] compared his proposed algorithm, the komodo mlipir algorithm (KMA), with genetic algorithm (GA), success-history based parameter adaptation differential evolution (SHADE), equilibrium optimizer (EO), marine-predators algorithm (MPA), and slime mold algorithm (SMA). Dehghani et al. [33] proposed a dart game optimizer (DGO) and compared it with GA, particle swarm optimization (PSO), gravitational search algorithm (GSA), teaching learning-based optimization (TLBO), grey wolf optimizer (GOA), whale optimization algorithm (WOA), and MPA [33]. Dehghani et al. [34] proposed shell game optimization (SGO) and compared its performance with GA, PSO, GSA, TLBO, spotted hyena optimizer (SHO), and emperor penguin optimizer (EPO). From these three examples, it is shown that GA is a favorite algorithm to beat.

The popularity of old-fashioned algorithm, such as GA, SA, and PSO, is still high although they were beaten many times by many shortcoming algorithms. Many studies in optimization still use these algorithms today. These algorithms were improved, modified, and combined with other algorithms many times and implemented in many areas. Many studies were conducted in improving the GA in its certain operators: encoding, selection, crossover, or mutation [35]. GA has many variants, such as binary GA [36] and non-dominated sorting GA (NSGA II) [37].

SA is the other example of an efficient and easy to implement algorithm [38]. Many studies were conducted to modify this algorithm. Several examples of its modified version are cloud-theory based simulated annealing [39], curious simulated annealing [38], fast simulated annealing [40], and sequential Monte Carlo simulated annealing [41].

PSO is the other old-fashioned algorithm that is widely used and modified until today. In its original form, its strategy is very simple. There are several agents or particles that fly within the solution space to find a better solution [42]. Its movement depends on its location and speed. Its next speed is determined by four aspects: current speed, local best solution, global best solution, and certain probabilistic value [42]. PSO has several variants, such as cooperative PSO [43], adaptive PSO [44], and multi-objective PSO [45].

Based on this explanation, studies to develop metaheuristic algorithm is still challenging. Meanwhile, hiding behind metaphors should be avoided. Moreover, proposing a simple algorithm may be better so that it can be implemented and improved widely rather than proposing a complex algorithm.

This work proposes a metaheuristic optimization model in which diversification and intensification are conducted based on the iteration. In the beginning, diversification is conducted. The diversification is reduced gradually as the iteration increases. On the other side, the intensification is conducted gradually. This mechanism is conducted by setting the observation range wide enough in the initialization phase. Then, this observation range is reduced linearly until the lowest observation range is reached.

CS is a swarm-based intelligence. The system contains several agents [46]. Each agent works autonomously. Meanwhile, there is a centralized collective intelligence that is shared among agents [46].

The remainder of this paper is organized as follows. The model of CS, which consists of conceptual model, mathematical model, and the algorithm is explained in section two. The simulation was conducted to evaluate the performance of CS, and the result is described in section three. Section four discusses a deeper analysis due to the simulation result and the findings. This work is then summarized and concluded in section five.

II. Proposed Model

A. Conceptual Model

Cone search is a metaheuristic algorithm that adopts swarm intelligence. It contains several agents whose objective is to find the best solution. Each agent finds the better solution by finding an alternative solution within its observation range. The observation range must be inside the search space or problem space.

The illustration of the agent's observation range and the search space is shown in Fig. 1. Fig. 1 illustrates a single dimension search space. In Fig. 1, the search space is wide enough. Contrary, the agent's observation range is narrow so that it cannot cover the whole search space. The agent's location is in the middle of the agent's observation range.

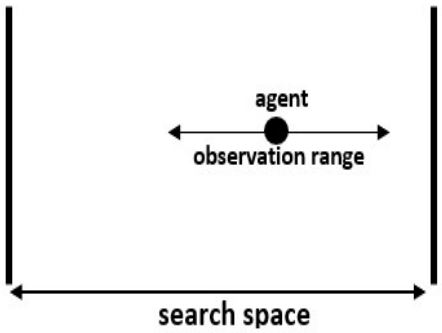

Fig. 1. Illustration of the observation range and the search space.

The observation range size is dynamic during the iteration. In the beginning, the observation range is wide. Then, it decreases linearly during the iteration. The observation range is narrow when the maximum iteration is reached. The motivation of this mechanism is as follows. In the beginning, the algorithm focuses on diversification. That is why the observation range should be wide enough to obtain any possible solutions. A wide observation range is also designed to avoid local optimal trap in the early iteration period.

Intensification is conducted gradually as the iteration goes. Narrower observation range is designed to limit the possible solution and achieve convergence The algorithm focuses on improving the solution by searching for possible alternatives near the current solution. This strategy becomes the main concept of CS and main distinction with other metaheuristic algorithms. The illustration of this dynamic observation range is shown in Fig. 2.

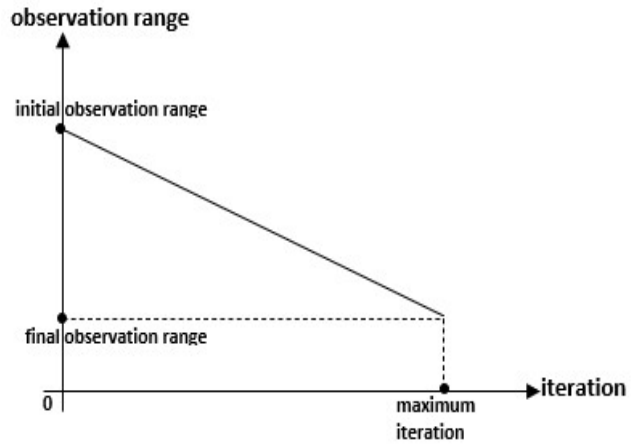

Fig. 2. Dynamic observation range.

This strategy is very different from many common algorithms. In general, iteration does not correlate with diversification or intensification. The example is as follows. In PSO, intensification and diversification are conducted simultaneously by combining the local best and global best in single formulae to determine the agent's movement [7]. In invasive weed optimization, intensification and diversification are conducted simultaneously in every iteration by spreading new weed within the problem space based on normal distribution [12]. In GA, intensification and diversification are conducted in every iteration. The cross-over represents the intensification, while mutation represents the diversification [4].

Simulated annealing (SA) is an algorithm in which iteration affects the diversification. In SA, diversification is easy to conduct in early iteration when the temperature is still high [5]. Diversification becomes more difficult as the iteration declines. In SA, diversification is conducted by accepting a worse solution by certain probabilistic calculations [5].

Each agent generates several candidates in every iteration. These candidates are uniformly distributed within the agent's current observation range. Then, the fitness score of these candidates is calculated based on the fitness function. The candidate whose fitness score is the best becomes the agent's best candidate. This best candidate is then sent to the memory for memory updating. This process is conducted for all agents in every iteration.

Memory is an entity whose role is to store a certain number of best solutions. It has a fixed capacity. In the beginning, this memory is empty. The memory will be updated every time an agent sends a solution to memory. If the number of solutions is below the memory capacity, a solution sent to it will be stored immediately. Otherwise, the memory will select its current worst solution. If this worst solution is worse than the incoming solution, this worst solution will be replaced. Otherwise, the incoming solution will be rejected.

The agent's next step is moving to a new location after it sends its best candidate to the memory. This process depends on the fitness score of its current location and its current best candidate. If its current best candidate is better than its current location, it becomes the agent's next location. Otherwise, the agent will pick one solution in the memory randomly as its next location. The motivation is as follows. If the agent's best candidate is better than the agent's current location, it is assumed that there is a possibility to exploit an area near the agent's current location. Otherwise, the agent should explore other locations that have a better opportunity.

B. Mathematical Model

This concept is transformed into a mathematical model. The main algorithm of the cone search is shown in algorithm 1. Table 1 shows the annotations used in this work.

TABLE I
ANNOTATIONS

Annotation	Description
i	agent index
j	candidate index
b_l	left border (lower bound)
b_r	right border (upper bound)
d_o	observation range
k_{init}	initial observation radius constant
k_{final}	final observation radius constant
r_o	observation radius
$r_{o\text{-}init}$	initial observation radius
$r_{o\text{-}final}$	final observation radius
Δr	observation-radius decrease rate
M	memory
x	location
f	fitness score
a	agent
M_c	memory capacity
$n(M)$	current memory size
D	dimension
c	candidate
c_{best}	Best candidate
t	time / iteration
t_{max}	maximum iteration
s_{final}	final solution
s_{in}	incoming solution
s_{worst}	the worst solution in the memory
P	problem space

algorithm 1: cone search

```
1   begin
2     for i = 1to n(A) do
3       set x(a_i)
4       calculate f(a_i)
5       update (M, a_i)
6     end for
7     for t = 1 to t_max do
8       for i = 1 to n(A) do
9         d_o(a_i) = define-observation-range (a_i)
10        for j = 1 to n(C) do
11          c_j = generate candidate (d_o(a_i))
12          c_best = sort-best (C)
13          update (M, c_best)
14          x(a_i) = find-next (a_i, c_best, M)
15        end for
16      end for
17      r_o = r_o - Δr
18    end for
19    s_final = sort-best (s in M)
20  end
```

The explanation of algorithm 1 is as follows. Lines 2 to 6

represent the initialization process, which consists of setting the agents' initial location, calculating their fitness, and sending them to the memory to be updated. In the initialization phase, the initial location of agents is uniformly distributed within the solution space. Lines 7 to 18 represent the iteration process that runs until the maximum iteration is reached. It consists of several processes: defining the observation range for selected agent (line 9), generating the candidates near the selected agent (line 11), selecting the best candidate (line 12), updating the memory (line 13), determining the agent's next location (line 14), and defining new observation radius (line 17). Line 19 becomes the finalization, where the best solution inside the memory becomes the final solution. The memory updating process is explained in more detail in algorithm 2.

algorithm 2: memory updating process

```
1   begin
2     if n(M) < Mc then
3       push (sin, M)
4     else
5       sworst = find-worst (M)
6       if f(sin) < f(sworst) then
7         replace (sworst, sin)
8       end if
9     end if
10  end
```

Below is the explanation of algorithm 2. Line 3 shows that the incoming solution will be pushed into the memory if the memory's current size is less than the memory capacity. Line 5 to 8 represents the mechanism if the memory's current size is not less than memory capacity. Line 5 represents the process of finding the worst solution in the memory. Line 7 represents replacing the worst solution with the incoming solution if the incoming solution is better than the current worst solution.

Several variables should be calculated before the initialization process. These variables are initial observation radius, final observation radius, and observation radius decrease rate. This calculation is formalized by using (1) to (3).

$$r_{oinit} = k_{init} \cdot \left| b_r(P) - b_l(P) \right| \tag{1}$$

$$r_{of\,inal} = k_{f\,inal} \cdot \left| b_r(P) - b_l(P) \right| \tag{2}$$

$$\Delta r = \frac{r_{oinit} - r_{of\,inal}}{t_{max}} \tag{3}$$

The observation range is a space within the solution space where an agent can generate several candidates. The agent's current location becomes the agent's observation ranges central. In general, the observation range width is twice as the observation radius. But the observation range must be within the problem space. If it surpasses the problem space, the observation range will be cut. The determination of the agent's observation range is formalized by using (4) for the left border and (5) for the right border.

$$b_l\left(d_o(a)\right) = \begin{cases} x(a) - r_o, b_l(P) \le x(a) - r_o \\ b_l(P), else \end{cases} \tag{4}$$

$$b_r\left(d_o(a)\right) = \begin{cases} x(a) + r_o, b_r(P) \ge x(a) + r_o \\ b_r(P), else \end{cases} \tag{5}$$

Several candidates are generated in every iteration for every agent. The motivation to generate more than one candidate is to search faster. This mechanism is like the tabu search [6] or cat swarm optimization [13]. This mechanism is formalized by using (6) and (7). Equation (6) states that the candidates are uniformly distributed within the agent's observation range. Equation (7) states that the candidate whose fitness score is the best becomes the best candidate.

$$c(a) = U\left(d_o(a)\right) \tag{6}$$

$$c_{best}(a) = c \in C(a) \wedge min(f(c)) \tag{7}$$

Determining the agent's next location becomes the last process in every iteration. As explained previously, the agent will move to its best candidate if it is better. Otherwise, the agent will pick a solution randomly from memory. This process is formalized by using (8).

$$a' = \begin{cases} c_{best}(a), f\left(c_{best}(a)\right) < f(a) \\ U(M), else \end{cases} \tag{8}$$

The complexity of CS is presented by using big O notation as $O(n(A).t_{max}(n(C) + n(M)))$. The explanation is as follows. Four variables affect the looping process. They are the maximum iteration, the number of agents, the number of candidates, and memory size. In general, the algorithm runs from the first iteration until the maximum iteration. All agents work in every iteration. Every agent in every iteration conducts two processes. The first process is generating a certain number of candidates. The second process is finding the worst solution within the memory size during the memory updating process.

III. Simulation and Result

Evaluation of CS is conducted by implementing this algorithm to find the global optimal of the well-known 23 functions. These functions have been used in a lot of studies that proposed new metaheuristic algorithm, such as in the first appearance of DGO [33], SGO [34], KMA [32], and GWO [18]. These functions can be classified into three groups: high dimension unimodal functions, high dimension multimodal functions, and fixed dimension multimodal functions. The detail description of these functions is shown in Table 2. The first group consists of function 1 to function 7. The second group consists of function 8 to function 13. The third group consists of function 14 to function 23.

TABLE II
DESCRIPTION OF 23 FUNCTIONS

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

In the first simulation, CS is benchmarked with four metaheuristic algorithms: PSO, MPA, KMA, and pelican optimization algorithm (POA). These algorithms are chosen based on several reasons. PSO represents the old-fashioned algorithm. PSO has been implemented in many optimization studies. Moreover, PSO is the earlier version of the swarm intelligence. MPA represents the shortcoming algorithms that has been used in many optimization studies. KMA and POA represent the brand-new algorithms that implement different strategy. KMA is a hybrid algorithm that combines the swarm movement (foraging) and evolution-based improvement (mating). POA is a swarm-based algorithm, but its global target is randomized in every iteration.

The parameter setting of these algorithms is as follows. The population size is 20, and the maximum iteration is 100. In CS, the number of candidates is 10 and the memory capacity is 10. In PSO, the weights are 0.1. In MPA, the fishing aggregate devices (FAD) is 0.2. In KMA, the big male proportion is 0.4, there is only one female, and the mlipir rate is 0.5.

Parameter setting in the first simulation is as follows. The initial observation radius constant is 0.1, and the final observation radius constant is 0.001. There are 30 runs for every benchmark function. The result is shown in Table 3. The last column in Table 3 describes the sparing algorithms that are beaten by the proposed algorithm in the related function.

Table 3 shows that CS performs well in solving the 23 benchmark functions. It can find the acceptable solutions in all three groups. It means that CS can tackle challenge in both unimodal functions and multimodal functions. CS is fast enough in finding the optimal solution of the unimodal functions. Meanwhile, CS can escape from the local optimal entrapment in solving the multimodal functions. Moreover, CS can find the global optimal solution in solving six fixed dimension multimodal functions: six hump camel, Branin, Goldstein price, Hartman 6, Shekel 5, and Shekel 7. These six functions are the fixed dimension multimodal functions.

CS is superior compared with the sparing algorithms. CS outperforms the four algorithms in solving 21 functions. Specifically, CS outperforms PSO, MPA, KMA, and POA in solving 23, 22, 21, and 22 functions respectively. Meanwhile, the proposed algorithm is less competitive in solving Hartman 3.

The second simulation is conducted to observe the performance of CS related to the initial observation radius constant. In this simulation, there are three values of the initial observation radius constant: 0.5, 0.1, and 0.02. The result is shown in Table 4.

The third simulation is conducted to observe the relation between the final observation radius constant and the performance of CS. There are three values of this constant: 0.01, 0.001, and 0.0001. The result is shown in Table 5.

TABLE III
SIMULATION RESULT

Function	Average Fitness Score					Better Than
	PSO	MPA	KMA	POA	CS	
1	6.703×10^{1}	1.459×10^{1}	5.492×10^{1}	2.867×10^{2}	7.119×10^{-2}	PSO, MPA, KMA, POA
2	6.913×10^{1}	2.873×10^{-3}	4.973×10^{-2}	3.055×10^{-1}	2.398×10^{-6}	PSO, MPA, KMA, POA
3	2.699×10^{2}	2.102×10^{1}	2.752×10^{2}	4.786×10^{2}	9.775×10^{-2}	PSO, MPA, KMA, POA
4	5.017	2.701×10^{-1}	5.438	1.054×10^{1}	1.211×10^{-1}	PSO, MPA, KMA, POA
5	8.221×10^{3}	4.814	1.434×10^{3}	7.406×10^{3}	3.957	PSO, MPA, KMA, POA
6	4.541×10^{1}	7.540	2.484×10^{1}	5.932×10^{1}	1.851×10^{-2}	PSO, MPA, KMA, POA
7	4.546×10^{-3}	8.686×10^{-3}	1.601×10^{-1}	5.457×10^{-2}	3.620×10^{-4}	PSO, MPA, KMA, POA
8	-1.137×10^{3}	-1.264×10^{3}	-1.857×10^{3}	-1.390×10^{3}	-1.685×10^{3}	PSO, MPA, POA
9	1.716×10^{1}	5.555	6.494	1.766×10^{1}	2.437	PSO, MPA, KMA, POA
10	4.441	3.106	4.756	9.113	1.594×10^{-1}	PSO, MPA, KMA, POA
11	1.304	9.063×10^{-1}	1.171	2.555	2.199×10^{-1}	PSO, MPA, KMA, POA
12	5.795	1.332	2.016	1.262×10^{1}	1.468×10^{-3}	PSO, MPA, KMA, POA
13	1.571×10^{1}	6.390	3.682×10^{2}	6.441×10^{2}	3.232×10^{-2}	PSO, MPA, KMA, POA
14	5.292	4.592	7.395	2.098	1.894	PSO, MPA, KMA, POA
15	3.069×10^{-2}	4.673×10^{-3}	1.403×10^{-2}	2.205×10^{-3}	7.513×10^{-4}	PSO, MPA, KMA, POA
16	-1.029	-1.027	-1.015	-1.029	-1.032	PSO, MPA, KMA, POA
17	8.897×10^{-1}	6.130×10^{-1}	4.010×10^{-1}	4.024×10^{-1}	3.981×10^{-1}	PSO, MPA, KMA, POA
18	6.800	4.029	3.029	3.063	3.000	PSO, MPA, KMA, POA
19	-5.931×10^{-3}	-3.802	-4.295×10^{-1}	-4.954×10^{-2}	-4.931×10^{-2}	PSO
20	-2.442	-2.027	-2.807	-2.916	-3.322	PSO, MPA, KMA, POA
21	-4.192	-1.886	-8.395	-4.204	-1.015×10^{1}	PSO, MPA, KMA, POA
22	-3.948	-1.809	-7.409	-3.967	-1.040×10^{1}	PSO, MPA, KMA, POA
23	-4.023	-2.381	-7.007	-3.260	-1.053×10^{1}	PSO, MPA, KMA, POA

TABLE IV
RELATION BETWEEN INITIAL OBSERVATION RADIUS CONSTANTS AND FITNESS SCORE

Function	Average Fitness Score		
	$r_{o\text{-}init} = 0.5$	$r_{o\text{-}init} = 0.1$	$r_{o\text{-}init} = 0.02$
1	4.521×10^{-1}	4.001×10^{-2}	1.078×10^{-2}
2	5.757×10^{-5}	2.049×10^{-8}	8.239×10^{-3}
3	7.851×10^{-1}	5.329×10^{-2}	1.106×10^{-2}
4	5.201×10^{-1}	1.244×10^{-1}	6.409×10^{-2}
5	4.299	5.377	3.722
6	1.686×10^{-1}	1.229×10^{-2}	4.297×10^{-3}
7	2.044×10^{-3}	2.806×10^{-4}	1.149×10^{-2}
8	-2.046×10^{3}	-1.783×10^{3}	-1.681×10^{3}
9	2.288	2.588	1.421×10^{1}
10	9.807×10^{-1}	1.683×10^{-1}	6.791×10^{-2}
11	4.042×10^{-1}	2.153×10^{-1}	7.011×10^{-2}
12	3.356×10^{-2}	1.712×10^{-3}	3.833
13	2.371×10^{-1}	3.284×10^{-2}	4.562×10^{-3}
14	9.980×10^{-1}	1.726	1.184
15	8.101×10^{-4}	7.820×10^{-4}	1.118×10^{-3}
16	-1.032	-1.032	-1.032
17	3.981×10^{-1}	3.981×10^{-1}	3.981×10^{-1}
18	3.004	3.000	4.5883
19	-4.862×10^{-2}	-4.930×10^{-2}	-4.267×10^{-2}
20	-3.272	-3.322	-3.322
21	-1.011×10^{1}	-1.015×10^{1}	-1.015×10^{1}
22	-1.036×10^{1}	-1.040×10^{1}	-1.040×10^{1}
23	-1.049×10^{1}	-1.053×10^{1}	-1.024×10^{1}

TABLE V
RELATION BETWEEN FINAL OBSERVATION RADIUS CONSTANTS AND FITNESS SCORE

Function	Average Fitness Score		
	$r_{o\text{-}final} = 0.01$	$r_{o\text{-}final} = 0.01$	$r_{o\text{-}final} = 0.0001$
1	7.133×10^{-1}	3.876×10^{-2}	2.043×10^{-2}
2	1.689×10^{-5}	3.286×10^{-9}	3.088×10^{-15}
3	5.732×10^{-1}	4.787×10^{-2}	2.441×10^{-2}
4	6.266×10^{-1}	1.389×10^{-1}	8.388×10^{-2}
5	6.642	3.953	3.802
6	2.101×10^{-1}	1.473×10^{-2}	7.193×10^{-3}
7	2.207×10^{-4}	4.243×10^{-4}	3.051×10^{-4}
8	-1.731×10^{3}	-1.823×10^{3}	-1.764×10^{3}
9	3.069	2.808	2.308
10	1.110	1.693×10^{-1}	9.274×10^{-2}
11	3.099×10^{-1}	2.275×10^{-1}	1.345×10^{-1}
12	1.809×10^{-2}	1.347×10^{-3}	1.011×10^{-3}
13	2.950×10^{-1}	3.033×10^{-2}	1.624×10^{-2}
14	2.421	1.984	2.288
15	8.022×10^{-4}	7.392×10^{-4}	7.088×10^{-4}
16	-1.032	-1.032	-1.032
17	3.981×10^{-1}	3.981×10^{-1}	3.981×10^{-1}
18	3.003	3.000	3.000
19	-4.871×10^{-2}	-4.932×10^{-2}	-4.938×10^{-2}
20	-3.322	-3.322	-3.322
21	-1.009×10^{1}	-1.015×10^{1}	-1.015×10^{1}
22	-1.033×10^{1}	-1.040×10^{1}	-1.040×10^{1}
23	-1.049×10^{1}	-1.053×10^{1}	-1.053×10^{1}

Table 4 shows that in general, the initial observation radius constant does not affect the proposed algorithm's performance significantly. This circumstance occurs mostly in the multimodal functions. Meanwhile, the response related the change in the initial optimization radius constant is different among functions. Among these 23 functions, the most optimal solution can be found when the initial problem space constant radius is narrow, moderate, or wide.

Table 5 shows that in general, wider final observation radius constant is counter-productive to the algorithm's performance. In many functions, the best performance is achieved when the final problem space radius is narrower. Meanwhile, in some functions, the change in the final observation radius constant can change the proposed algorithm's performance significantly. But, in many functions, the change is not significant.

IV. DISCUSSION

This section discusses deeper analysis due to the result and findings. This analysis is conducted in four considerations. The first is the ability of CS to meet the general objectives of the metaheuristic algorithm: finding a near-optimal solution and avoiding local optimal trap. The second is the performance comparison between CS and benchmark algorithms. The third is the evaluation of the parameters of CS in the context of the performance. The fourth is the algorithm complexity.

The simulation result shows that CS has met the two objectives of the metaheuristic algorithm. These objectives are achieved in all benchmark functions. Moreover, CS can find the global optimal solution in solving six fixed-dimension multimodal functions.

CS is also superior, compared with all sparing algorithms: PSO, MPA, KMA, and POA. This superiority occurs in 21 functions. Meanwhile, CS is less competitive in solving Hartman 3 functions. The simulation result strengthens the statement that no algorithm best solves all problems as it was stated as no free lunch theorem. The no free lunch theorem states that the algorithm's effectiveness also depends on the of problems [47].

The result shows that CS is superior in solving problems without considering the problem space size. It can solve problems with very narrow problem space, such as Quartic or Rastrigin. Besides, CS is still superior in solving problems with very large problem space, such as Schwefel and Griewank.

The observation radius plays an important role in improving the algorithm. The final observation radius must be set as small as possible to conduct the intensification properly, especially when the potential solution space has been found. On the other side, the initial observation radius must be set small enough to avoid wasting iteration in finding the broad solution. But this value cannot be too low that the true optimal solution becomes difficult to find.

There is challenge in making CS popular and widely used in the real-world optimization studies. On the other hand, the old-fashioned algorithms are still popular although they have been beaten by the later algorithms. For example, genetic algorithm is still used in many optimization studies, such as for optimizing the long short-term memory [48], wood-types classification [49], and stock trend forecasting [50]. Ant colony optimization is also still popular to solve many combinatorial optimization problems, such as in the pickup and delivery problem [51], tourism route planning [52], and assignment problem in cloud computing [53]. This circumstance comes from the simplicity and flexibility of these old-fashioned algorithms. On the other hand, the performance of optimization can be improved by increasing the maximum iteration or population size. These methods are easier than implementing more complex algorithm as conducted in many shortcoming algorithms. This circumstance also becomes challenge to propose more powerful algorithm while keeping it simple and flexible.

V. CONCLUSION

This work has demonstrated that CS has met the two main objectives of metaheuristic algorithms: finding the near-

optimal solution and avoiding the local trap. It can solve both unimodal and multimodal functions. Moreover, CS is proven to find the global optimal solution in solving six functions: Six hump camel, Branin, Goldstein price, Hartman 6, Shekel 5, and Shekel 7. Its performance is also superior, compared with the benchmark algorithms. CS outperforms PSO, MPA, KMA, and POA in solving 23, 22, 21, and 22 functions respectively.

CS may rise many future research potentials. It should be implemented and tested in many optimization problems, such as operations research, signal processing, and so on. Hybridizing CS with other methods is also interesting. The implementation of CS in optimizing discrete or combinatorial problems is challenging.

REFERENCES

[1] J. Swan, S. Adriaensen, A.E.I. Brownlee, K. Hammond, C.G. Johnson, A. Kheiri, F. Krawiec, J.J. Merelo, L.L. Minku, E. Ozcan, G.L. Pappa, P. Garcia-Sanchez, K. Sorensen, S. Vob, M. Wagner, and D.R. White, "Metaheuristics in the large," *European Journal of Operational Research*, vol. 297, pp. 393-406, 2022.

[2] S. Adriaensen, T. Brys, and A. Nowe, "Fair-share ILS: a simple state-of-the-art iterated local search hyperheuristic," *Proceeding of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014.

[3] D. B. Fogel and L. J. Fogel, "An introduction to evolutionary programming," Lecture Notes in Computer Science, vol. 1063, 1996.

[4] J. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Application to Biology*, MIT Press, 1975.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[6] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers and Operations Research*, vol. 13, no. 5, pp. 533-549, 1986.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceeding of International Conference on Neural Network*, 1995.

[8] M. Dorigo, V. Maniezzo, and A. Coloni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, part B (Cybernetics)*, vol. 26, no. 1, pp. 29-41, 1996.

[9] N. Mladenovic and P. Hansen, "Variable neighborhood search," *Computers and Operations Research*, vol. 24, no. 11, pp. 1097-1100, 1997.

[10] H. Kim and B. Ahn, "A new evolutionary algorithm based on sheep flocks heredity model," *Proceeding od IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 514-517, 2001.

[11] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52-67, 2002.

[12] A. R. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecological Informatic*, vol. 1, no. 4, pp. 355-366, 2006.

[13] S. C. Chu, P. W. Tsai, and J. S. Pan, "Cat swarm optimization," *Proceeding of Pacific Rim International Conference on Artificial Intelligence*, pp. 854-858, 2006.

[14] X. S. Yang, "Firefly algorithms for multimodal optimization," *Proceeding of International Symposium on Stochastic Algorithms*, pp. 169-178, 2009.

[15] U. Premaratne, J. Samarabandu, and T. Sidhu, "A new biologically inspired optimization algorithm," *Proceeding of the 2009 International Conference on Industrial and Information Systems (ICIIS)*, pp. 279-284, 2009.

[16] R. Tang, S. Fong, X. S. Yang, and S. Deb, "Wolf search algorithm with ephemeral memory," *Proceeding of 7th International Conference on Digital Information Management (ICDIM 2012)*, pp. 165-172, 2012.

[17] S. Harifi, M. Khalilian, J. Mohammadzadeh, and S. Ebrahimnejad, "Emperor penguins colony: a new metaheuristic algorithm for optimization," *Evolutionary Intelligence*, 2019.

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

[19] X. Meng, Y. Liu, X. Gao, and H. Zhang, "A new bio-inspired algorithm: chicken swarm optimization" *Proceeding of International Conference in Swarm Intelligence*, pp. 86-94, 2014.

[20] J. B. Odili, M. N. M. Kahar, and S. Anwar, "African buffalo optimization: a swarm-intelligence technique," *Procedia Computer Science*, vol. 76, 443-448, 2015.

[21] M. Yazdani and F. Jolai, "Lion optimization algorithm: a nature-inspired metaheuristic algorithm," *Journal of Computational Design and Engineering*, vol. 3, no. 1, pp. 24-36, 2016.

[22] W. Qiao and Z. Yang, "Solving large-scale function optimization problem by using a new metaheuristic algorithm based on quantum dolphin swarm algorithm," *IEEE Access*, vol. 7, no. 1, pp. 138972-138989, 2019.

[23] Y. J. Zheng, "Water wave optimization: a new nature-inspired metaheuristic," *Computers and Operations Research*, vol. 55, no. 1, pp. 1-11, 2015.

[24] Y. C. Liang and J. R. Cuevas-Juarez, "A novel meta-heuristic algorithm for continuous optimization problems: virus optimization algorithm," *Engineering Optimization*, vol. 48, no. 1, pp. 73-93, 2016.

[25] G. W. Yan and Z. J. Hao, "A novel optimization algorithm based on atmosphere clouds model," *International Journal of Computational Intelligence and Applications*, vol. 12, no. 1, ID: 1350002, 2013.

[26] E. Rashedi, H. Nezamabadi-Pour, and S. Saryadi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232-2248, 2009.

[27] S. K. Pal, C. S. Rai, and A. P. Singh, "Comparative study of firefly algorithm and particle swarm optimization for noisy optimization problems", *International Journal of Intelligent Systems and Applications*, vol. 4, no. 10, pp. 50-57, 2017.

[28] P. Verma and R. P. Parouha, "An advanced hybrid meta-heuristic algorithm for solving small- and large-scale engineering design optimization problems," *Journal of Electrical Systems and Information Technology*, vol. 8, 2021.

[29] 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.

[30] M. A. A. Rahman, B. Ismail, K. Naidu, and M. K. Rahmat, "Review on population-based metaheuristic search techniques for optimal power flow," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 15, no. 1, pp. 373-381, 2019.

[31] 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.

[32] Suyanto, A.A. Ariyanto, and A.F. Ariyanto, "Komodo mlipir algorithm," *Applied Soft Computing*, vol. 114, ID: 108403, 2022.

[33] M. Deghani, Z. Montazeri, H. Givi, J. M. Guerrero, and G. Dhiman, "Darts game optimizer: a new optimization technique based on darts game," *International Journal of Intelligent Engineering & Systems*, vol. 13, no. 5, pp. 286-294, 2020.

[34] M. Deghani, Z. Montazeri, O. M. Malik, H. Givi, and J. M. Guerero, "Shell game optimization: a novel game-based algorithm," *International Journal of Intelligent Engineering & Systems*, vol. 13, no. 3, pp. 246-255, 2020.

[35] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, pp. 8091-8126, 2021.

[36] A. W. Payne and R. C. Glen, "Molecular recoqnition using binary genetic system," Journal of Molecular Graphics, vol. 11, no. 2, pp. 74-91, 1993.

[37] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[38] T. Guilmeau, E. Chouzenoux, and V. Elvira, "Simulated annealing: a review and a new scheme," *IEEE Statistical Signal Processing Workshop*, 2021.

[39] E. Torabzadeh and M. Zandieh, "Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop," *Advances in Engineering Software*, vol. 41, no. 10-11, pp. 1238-1243, 2010.

[40] S. Rubenthaler, T. Ryden, and M. Wiktorsson, "Fast simulated annealing in R^d with an application to maximum likelihood estimation in state-space models," *Stochastic Process and Their Applications*, vol. 119, no. 6, pp. 1912-1931, 2009.

[41] E. Zhou and X. Chen, "Sequential Monte Carlo simulated annealing," *Journal of Global Optimization*, vol. 55, no. 1, pp. 101-124, 2013.

[42] D. Freitas, L. G. Lopes, and F. Morgado-Dias, "Particle swarm optimization: a historical review up to the current developments," *Entropy*, vol. 22, ID: 362, pp. 1-36, 2020.

[43] F. Van den-Bergh and A. P. Engelbrecht, "Cooperative learning in neural networks using particle swarm optimizers," *South African Computer Journal*, vol. 26, pp. 84-90, 2000.

[44] Z. Zhan, J. yang, Y. Li, and H. S. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on System, Man, and Cybernetics*, vol. 39, pp. 1362-1381, 2009.
[45] X. Hu and R. C. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," *Proceeding of the Congress on Evolutionary Computation (CEC)*, pp. 1677-1681, 2002.
[46] Y. Qawqzeh, M. T. Alharbi, A. Jaradat, and 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.
[47] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67-82, 1997.
[48] A. S. Girsang and D. Tanjung, "Fast genetic algorithm for long short-term memory optimization," *Engineering Letters*, vol. 30, no. 2, pp. 528-536, 2022.
[49] S. Santosa, R. A. Pramunendar, D. P. Prabowo, and Y. P. Santosa, "Wood types classification using back-propagation neural network based on genetic algorithm with gray level co-occurence matrix for feature extraction," *IAENG International Journal of Computer Science*, vol. 46, no. 2, pp. 149-155, 2019.
[50] R. Abraham, M. E. Samad, A. M. Bakhach, H. El-Chaarani, A. Sardouk, S. E. Nemar, and D. Jaber, "Forecasting a stock trend using genetic algorithm and random forest," *Journal of Risk and Financial Management*, vol. 15, ID: 188, pp. 1-18, 2022.
[51] P. N. K. Phuc and N. L. P. Thao, "Ant colony optimization for multiple pickup and multiple delivery vehicle routing problem with time window and heterogeneous fleets," *Logistics*, vol. 5, ID: 28, pp. 1-13, 2021.
[52] 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.
[53] G. B. H Bindu, K. Ramani, and C. S. Bindu, "Optimized resource scheduling using the meta heuristic algorithm in cloud computing," *IAENG International Journal of Computer Science*, vol. 47, no. 3, pp. 360-366, 2020.

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.

Ratna Astuti Nugrahaeni is a lecturer and researcher in computer engineering in Telkom University, Indonesia. She received her bachelor's program in computer system from Telkom University, Indonesia. She received her master's degree in electrical engineering from Bandung Institute of Technology, Indonesia. Her research interests are in artificial intelligence and machine learning.

Ashri Dinimaharawati is a lecturer and researcher in computer engineering in Telkom University, Indonesia. She received her bachelor's program from Education University of Indonesia, Indonesia. She received her master's degree in electrical engineering from Bandung Institute of Technology, Indonesia. Her research interests are in artificial intelligence, game development, and software engineering.