Received: February 14, 2022. Revised: March 10, 2022. 

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

# Modified Social Forces Algorithm: from Pedestrian Dynamic to Metaheuristic Optimization

**Purba Daru Kusuma[1]*** **Dimas Adiputra[2]**

[1]*Computer Engineering, Telkom University, Indonesia*
[2]*Electrical Engineering, Institut Teknologi Telkom Surabaya, Indonesia*
* Corresponding author's Email: purbodaru@telkomuniversity.ac.id

**Abstract:** This work proposes a new simple metaheuristic optimization method inspired by the social forces model used in pedestrian dynamics. The proposed model is a swarm-based model where a collective intelligence is shared among the agents, consisting of several persons or agents who walk over the search space to find the best solution. Each time a person finds a better solution, they share it with another person. The proposed model is then evaluated by implementing it to solve ten benchmark functions. The five are multimodal functions (Ackley, Rastrigin, Griewank, Bukin, and Michalewicz) while the other are unimodal (Sphere, Rosenbrock, Bohachevsky, Zakharov, and Booth). The performance is compared with five metaheuristic algorithms: particle swarm optimization, darts game optimizer, shell game optimization, marine predator algorithm, and Komodo mlipir algorithm. The simulation result shows that the proposed method is competitive enough to solve multimodal and unimodal functions. The performance is superior in solving Michalewicz and Zakharov functions but less competitive in solving the Bukin function. The results also imply that there is no single algorithm that is the best in solving all kinds of problems.

**Keywords:** Social forces, Metaheuristic, Multi-agent, Multimodal, Swarm intelligence.

## 1. Introduction

Avoiding the local optimal is one major issue in metaheuristic optimization. As an approximation-based method, metaheuristic optimization starts with randomized points within the search space. Then, during the iteration, the algorithm tries to find a better solution around its current position (local search). The algorithm can be trapped in the local optimal solution during this process when no better solution can be found [1]. This problem is the disadvantage of approximation-based optimization, such as the metaheuristic optimization, but is not the case in the exact method. Nevertheless, the metaheuristic is still popular due to its advantage in solving ample space or multi-dimensional problems where the exact approach is impossible due to the excessive resource consumption (time and computation) [2].

As shown by the previous research, every metaheuristic optimization method has its way of solving the local optimal problem. Genetic algorithm (GA) uses mutation process [3]. In simulated annealing (SA), the new solution can still be accepted, although it is worse than the current solution based on some probabilistic calculations [4]. In an artificial bee colony (ABC), scout bees find alternative food sources when the existing food sources are abandoned [5]. A worse solution is also accepted in tabu search (TS) [6]. In harmony search (HS), new alternatives can also be generated randomly within possible problem space rather than only from the harmony memory (current solutions) [7]. In invasive weed optimization (IWO), the generated seeds are typically distributed over the problem space related to their parent [8].

Nature has been inspiring many metaheuristic algorithms. Most of them were inspired by animal behavior, such as birds [9], ants [10], deer [11], monkey [12], bees [5], dolphins [13], marine predators [14], Komodo dragon [15], and so on. Some were inspired by the plants, such as weeds [8]. Many of them are swarm-based intelligence, where the

model consists of independent agents, and information is shared among them [16]. The examples are particle swarm optimization (PSO) [9], ant colony optimization (ACO) [10], artificial bee colony (ABC) [5], and so on. Besides nature, several metaheuristic algorithms are inspired by the game, such as shell game optimization (SGO) [17] and darts game optimizer (DGO) [18].

There are hundreds of metaheuristic algorithms, which the numbers are possible to keep increasing in the future. This large number of algorithms shows that studies in metaheuristic optimization are interesting. The numbers keep growing because no single metaheuristic method is the best for all types of problems [2]. Some algorithms are simple, while others are complicated. Moreover, these methods have been improved, modified, and combined. The methods have also been implemented in many aspects of life, such as the production process, scheduling, education, transportation, etc.

Although there are many new algorithms, the old school algorithms are still popular. Today, the GA is still widely used, for example, in intelligent controllers for induction motor [19], short-term load forecasting [20], internet of things (IoT) service selection [21], and many more. SA is also popular, especially in data mining tasks [22]. Besides these two algorithms, there are many examples of the old school algorithms in many recent optimization studies. Although these algorithms are beaten many times by many new algorithms, they are still popular due to their simplicity, making them easy to modify or combine. Therefore, the simplicity aspect should be concerned with developing a metaheuristic algorithm.

Therefore, this work proposes a new metaheuristic algorithm, which can avoid the local optimal trap. The algorithm is called the modified social forces algorithm (MSF). This algorithm is inspired by the social forces model, well-known for dynamic pedestrian modeling, especially crowd modeling [23]. In the basic form, the social forces model consists of a person that walks from his initial location to his destination point within a certain speed [23]. He may meet other persons while walking and avoid collision (interaction forces) [23].

Moreover, he also tries to avoid collision with the rigid objects around him, such as walls (repulsion force) [23]. In the improved model, this person may also be attracted to other objects around him. This model is implemented into all persons in the system to be a multi-agent system.

Meanwhile, the proposed MSF model has a specific mechanism for local searching (intensification) and avoiding the optimal local trap (diversification). In this work, the social-forces model is modified into a metaheuristic optimization. The model still adopts the desired driving force and interaction force with modification in the formulation. A person may walk to his target at a certain speed or visit attractive objects around him. In this algorithm, the walking behavior represents finding a better solution. This proposed model also adopts swarm intelligence where some information (attractive thing) is shared among persons [16] so that other people may visit this attractive object.

The contributions of this work are as follows.

1. This work transforms the social forces model commonly used in crowd modeling into a metaheuristic algorithm, rather than using metaphors of nature (animals or plants).
2. This work proposes a new metaheuristic algorithm that is simple in mechanism and calculation but can avoid local optimal trap.

The remainder of this paper is organized as follows. Several well-known or shortcoming metaheuristic methods are discussed in section two. The proposed model is explained in section three. The simulation scenario for testing the proposed algorithm's performance and the simulation result is shown in section four. The more profound analysis of the outcome and findings is explained in section five. Finally, the conclusion and future work are summarized in section six.

## 2. Related works

In general, the metaheuristic algorithm consists of two activities: intensification and diversification. Intensification is a mechanism to improve the current solution by searching for new solutions near the current solution (neighborhood search or local search). On the other hand, diversification is a mechanism to find a new solution somewhere else within the search space. Diversification is conducted to avoid the local optimal problem. In some algorithms, these mechanisms are conducted in different steps, while in some others, these mechanisms are blended or unclearly separated. The followings are several popular or shortcoming metaheuristic algorithms, including the source of inspiration and their mechanism.

A genetic algorithm is an algorithm inspired by the evolution process [3]. Evolution theory says that the creature may adapt to the environment change by changing the traits gradually. As an algorithm, the better solutions are generated by the crossover process between the best current solutions (intensification). Meanwhile, mutation may occur with specific probabilistic calculations (diversification).

Simulated annealing adopts metal's annealing (controlled heating and cooling) process [4]. The heating process is done as fast as possible, but the cooling process is gradually done to avoid defects. A new solution may be better or worse than the current solution in every iteration. A better solution will be accepted immediately (intensification). Meanwhile, a worse solution can still be accepted with certain probabilistic calculations (diversification). Here, the probabilistic scale is the temperature. As the algorithm runs the calculation, the temperature gradually decreases to keep the search coverage minimum.

Tabu search is an example of a metaheuristic algorithm that works based on local or neighborhood search [6]. It has a list of solutions called memory. In every iteration, several candidates are generated near the current best solution. The best candidate is chosen among these candidates, which is the best among them, and it is not in the tabu list. If this candidate is better than the current best solution, it becomes the best solution. This solution is then pushed into the tabu list. The oldest solution is removed from the list if the tabu list is complete.

Particle swarm optimization is inspired by the flock of birds [9], a swarm intelligence consisting of several agents. In the beginning, each agent is generated randomly within the search space. The improvisation is conducted for every agent depending on its current position, local best, global best, and certain probabilistic value. Each time an agent moves to a new position, the local best and the global best are updated.

Ant colony optimization is inspired by ants when searching for food [10]. It is instrumental in solving combinatorial problems. This algorithm is multi-agent, graph-based, and swarms intelligence simultaneously. In the beginning, every ant searches for food randomly and returns by laying down a pheromone. Other ants that find the path will not search randomly but follow this path and update the pheromone. Over time, the pheromone evaporates.

Capuchin search algorithm is an algorithm that is inspired by the dynamic behavior of the capuchin monkeys in the forest [12]. To search the food, a Capuchin monkey jumps from tree to tree (diversification) or swings and climbs from branch to branch within a tree (intensification). In a way, the algorithm searches the solution locally first. Then, it jumps to other areas within the search space to check whether it is the optimal solution.

The Quantum dolphin swarm algorithm is inspired by a group of dolphins [13], which improves the dolphin swarm algorithm. Here, the hunting process consists of two stages: searching and predating. The critical aspects of the hunting process are echolocation, division of the dolphins, cooperation, and information exchange.

The red deer algorithm is inspired by the mating behavior of red deer [11]. The population is divided into male deer and hinds. A group of hinds becomes a harem. Some male deer become commanders during mating, and the others become stags. The mating of the commanders conducts diversification with some of their and other commanders' harem. The intensification is conducted during the fight between male commanders and stags. It also happened when he stags mate with their nearest hind.

Marine predator algorithm (MPA) is an algorithm inspired by the behavior of the predators and preys in the ocean [14]. Here, the predators represent the elitists or the best solution. The system consists of two sets: predators and prey. The behavior of the predators is affected by the prey. On the other side, the behavior of the prey is also affected by the predator. This algorithm combines the Brownian and Levy movements. It is an example of an algorithm that the iteration affects the mechanism. The iteration is divided into three phases. In the first phase, all individuals conduct the Brownian movement. Then, in the second phase, half of the population conduct Brownian movement while the rest conduct Levy movement. In the third phase, all individuals conduct the Levy movement.

Komodo mlipir algorithm (KMA) is an algorithm that is inspired by the behavior of the Komodo dragon during mating and foraging [15]. It is also the hybridization between evolutionary algorithms and swarms intelligence. The population is divided into big males, females, and small males. Big males conduct intensification (attraction) with certain diversification (distraction) by interacting with other big males. Females conduct intensification or diversification by mating with the highest quality big male or conducting asexual reproduction. Small males conduct intensification by following big males with a certain speed called mlipir. This algorithm is an example of several ways to conduct intensification and diversification in one algorithm.

Shell game optimization is an algorithm inspired by the mechanics of the traditional game, namely the shell game [17]. There are three possible states for every agent in every iteration: a combination of intensification and diversification. The state is chosen based on some probabilistic calculation from the agent's fitness score relative to the entire population's fitness score. First, the agent follows the current best solution (intensification) in the first state. Then, in the second state, the agent follows the best solution and

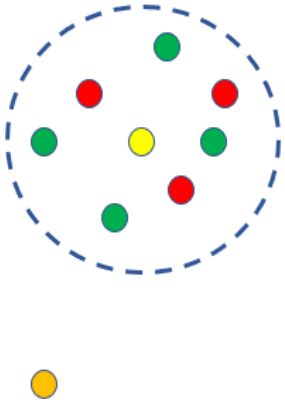

Figure. 1 Proposed model illustration

one randomly picked solution (intensification and diversification). The agent follows two randomly picked solutions (diversification).

Darts game optimizer (DGO) is a metaheuristic algorithm inspired by the mechanics of the darts game [18]. In general, this game is an intensification-oriented algorithm because all agents are designed to follow the current best solution at a certain speed. This speed is determined based on the accumulation of three discrete scores based on the darts score matrix. These scores are determined based on the probabilistic circumstance determined by comparing the agent's fitness score relative to the fitness score of the entire population. This algorithm is also an example of minimum adjustment so that the disparity between the best and the worst scenario is minimum.

Several studies developed a hybrid heuristic method. The hybrid method is implemented to combine the benefit of several algorithms. The other reason is to tackle the weakness of an algorithm. Many old-fashioned algorithms, such as GA, TS, PSO, and SA, are commonly used in this method. The reason is that these algorithms are simple and easy to modify. The example of studies conducting a hybrid heuristic is as follows.

Umam, Mustafid, and Suryono [24] combined GA and TS in solving the flow shop scheduling problem. Their motivation is that GA effectively produces good and fast solutions in complex problem spaces. But GA is not so good in the intensification. On the other side, TS is superior in conducting a local search. Tian and Liu [25] combined GA and PSO in solving the job shop scheduling problem. Their motivation is that PSO can cover the weakness in GA, where the convergence may occur too fast.

This explanation shows that every metaheuristic algorithm has its mechanics. Some algorithms separate the intensification and diversification, while the others combine both in a single formula. Meanwhile, several algorithms are developed based on intensification but accept diversification by certain probabilistic calculations. Although their mechanism is various, all metaheuristic algorithms have the same objective: finding a near-optimal solution or acceptable solution while avoiding local optimal.

The proposed algorithm is adopted from the pedestrian dynamic or crowd movement model called social forces model in this work. It is a swarm-based intelligence that consists of several agents, where every agent moves to their destination. An agent may be attracted to an interesting object near it when walking (intensification). After an agent arrives at its current destination, it determines a new destination. This new destination may be the global best (convergence) or within the search space (diversification).

## 3. Model

The proposed model consists of several agents. Each agent walks to its target or destination with a certain walking speed or step size. During walking, several more attractive or less attractive objects may be met within the observation range. The less attractive objects will not be considered. Meanwhile, if there are any attractive objects, the agent will first go to the most attractive object. If there is no attractive object, it will go to its target. After it reaches its current target, it will determine its new target. The illustration is shown in Fig. 1. In Fig. 1, the green dots represent the more attractive objects, the red dots represent the less attractive objects, the blue-lined circle represents the observation range, and the orange circle represents the target.

As a swarm intelligence, the agents interact with each other. Every time an agent finds a better, more attractive object, it will share with the other as collective intelligence. If this more attractive object is better than the current collective best-attractive object, it will replace it as a new collective best-attractive object. When the agent determines the new target, it has the option to choose between the collective best-attractive object or any other location as the new target.

The annotations used in the mathematical model are as follows. The modified social forces algorithm is formalized in algorithm 1. The algorithm consists of two rounds: initialization and iteration. The agent's initial position, target, and global best are determined during the initialization. These are determined randomly within the search space. This process is formalized by using Eq. (1) and Eq. (2). The initial global best calculation is standardized by using Eq. (3).

$$x_i(0) = U(\mathbb{R}) \quad (1)$$

| | |
|---|---|
| $x$ | current position |
| $x_{ta}$ | target |
| $x_{at}$ | attractive object |
| $x_{atb}$ | the best attractive objects |
| $x_{lo}$ | local best |
| $x_{go}$ | global best |
| $x_{wa}$ | walking position |
| $x_{sh}$ | shifting position |
| $\Delta x$ | step size |
| $f$ | fitness function |
| $i$ | agent index |
| $n$ | number of agents |
| $m$ | number of attractive objects |
| $p$ | probabilistic ratio |
| $r$ | observation range |
| $\mathbb{R}$ | search space |
| $t$ | iteration |
| $t_{max}$ | maximum iteration |

algorithm 1: modified social forces algorithm

```
//initialization
for i = 1 to n do
  determine x_i
  determine x_ta,i
  x_lo,i = x_i
end for
calculate x_go
//iteration
for t = 1 to t_max do
  for i = 1 to n do
    for j = 1 to m do
      determine x_at,i
    end for
    select x_atb,i
    update x_i
    update x_lo
    update x_go
    if x_i = x_ta,i then
      update x_ta,i
    end if
  end for
end for
```

$$x_{ta,i}(0) = U(\mathbb{R}) \tag{2}$$

$$x_{go}(0) = x_{lo}(0), x_{lo} \in X_{lo} \wedge \min\left(f\left(x_{lo}(0)\right)\right) \tag{3}$$

The iteration runs until the maximum iteration is reached. Several processes are conducted sequentially in every iteration. These processes are determining the attractive objects for every agent, selecting the most attractive object, updating the agent's next position, updating the local best and global best, and finally updating the agent's next target if it must be updated. These processes are formalized using Eq. (4) to Eq. (11).

$$x_{at,i,j} = U(\mathbb{R}) \wedge \left|x_{at,i,j} - x_i\right| \leq r \tag{4}$$

$$x_{atb,i} = x_{at,i} \in X_{at,i} \wedge \min\left(f\left(x_{at,i}\right)\right) \tag{5}$$

$$x_i' = \begin{cases} x_{atb,i}, f\left(x_{atb,i}\right) < f(x_i) \\ x_{wa}, else \end{cases} \tag{6}$$

$$x_{wa,i} = \begin{cases} x_{ta,i}, \left|x_i - x_{ta,i}\right| \leq \Delta x_i \\ x_{sh,i}, else \end{cases} \tag{7}$$

$$x_{sh,i} = \begin{cases} x_i + \Delta x_i, x_i < x_{ta,i} \\ x_i - \Delta x_i, x_i > x_{ta,i} \end{cases} \tag{8}$$

$$x_{lo,i}' = \begin{cases} x_{lo,i}, f(x_i) \geq f\left(x_{lo,i}\right) \\ x_i, else \end{cases} \tag{9}$$

$$x_{go}' = \begin{cases} x_{lo,i}, f\left(x_{lo,i}\right) < f\left(x_{go}\right) \\ x_{go}, else \end{cases} \tag{10}$$

$$x_{ta,i}' = \begin{cases} x_{ta,i}, x_i \neq x_{ta,i} \\ x_{go}, x_i = x_{ta,i} \wedge U(0,1) \leq p \\ U(\mathbb{R}), x_i = x_{ta,i} \wedge U(0,1) > p \end{cases} \tag{11}$$

Eq. (4) states that the attractive objects are generated randomly within the agent's observation range. The most attractive object is the highest fitness score, as calculated using Eq. (5). After that, the most attractive object becomes the agent's next position if its fitness score is better than the current, as declared by Eq. (6). Otherwise, the agent chooses to walk. Eq. (7) states that the agent arrives at the target if its distance is less than its step size. Otherwise, an agent chooses to shift. Eq. (8) states that the agent shifts to the new position depending on its current and step size. Eq. (9) declares that the agent's local best will be replaced with the agent's current position if the agent's current position's fitness score is better than the agent's local best's fitness score. Eq. (10) states that the agent local best will replace the global best if its fitness score is better than the global best's. Eq. (11) is used to determine the agent's next target. If the agent has arrived at its current target, the next target can be chosen from the global best or any location within the search space.

## 4. Simulation and result

This proposed model is evaluated for its performance using ten benchmark functions. These

Table 1. Benchmark functions

| No | Function | Model | Search Space | Step Size | Obs. Range |
|---|---|---|---|---|---|
| 1 | Ackley | $-20 \cdot exp\left(-0.2 \cdot \sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos 2\pi x_i\right) + 20 + exp(1)$ | [-32, 32] | 2 | 1 |
| 2 | Rastrigin | $10d + \sum_{i=1}^{D}\left(x_i^2 - 10cos(2\pi x_i)\right)$ | [-5.12, 5.12] | 0.2 | 0.1 |
| 3 | Griewank | $\frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} cos\left(\frac{x_i}{\sqrt{i}}\right)+1$ | [-600, 600] | 40 | 20 |
| 4 | Bukin | $100\sqrt{\lvert x_2 - 0.01x_1^2\rvert} + 0.01\lvert x_1 + 10\rvert$ | $x_1 \in [-15, -5]$ $x_2 \in [-3,3]$ | 0.2 | 0.1 |
| 5 | Michalewicz | $-\sum_{i=1}^{D}\left((\sin x_i) \cdot \left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m}\right), m = 10$ | [0, π] | 0.2 | 0.1 |
| 6 | Sphere | $\sum_{i=1}^{D} x_i^2$ | [-5.12, 5.12] | 0.2 | 0.1 |
| 7 | Rosenbrock | $\sum_{i=1}^{D-1}(100(x_{i+1} + x_i^2)^2 + (x_i - 1)^2)$ | [-5, 10] | 0.2 | 0.1 |
| 8 | Bohachevsky | $x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$ | [-100, 100] | 10 | 5 |
| 9 | Zakharov | $\sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^4$ | [-5, 10] | 0.2 | 0.1 |
| 10 | Booth | $(x_1 + 2x_2 + 7)^2 + (2x_1 + x_2 + 5)^2$ | [-10, 10] | 0.2 | 0.1 |

functions are divided into multimodal functions and unimodal functions. A multimodal function has many optimal solutions, while the unimodal function only has one optimal global solution [26]. There are five multimodal functions to be the benchmark: Ackley, Rastrigin, Griewank, Bukin, and Michalewicz. The other benchmarks are five unimodal functions: Sphere, Rosenbrock, Bohachevsky, Zakharov, and Booth. They are chosen based on a study proposing a nature-inspired metaheuristic algorithm, namely emperor-penguins colony [27]. Griewank and Bohachevsky represent functions with ample search space [27]. The target of all functions is 0, except Michalewicz, which is -4.6876, with the number of decision variables (*D*) being 5. A more detailed description of these functions is shown in Table 1.

This proposed model is compared with other benchmark methods: PSO, DGO, SGO, MPA, and KMA. The work chooses PSO because it represents a well-known metaheuristic algorithm that adopts swarm intelligence. On the other hand, the other four algorithms are chosen to represent the shortcoming metaphors-inspired algorithm. For instance, the DGO and SGO represent game-inspired algorithms, while MPA and KMA represent the animal-inspired algorithms. PSO represents the metaheuristic algorithm whose mechanism is straightforward. DGO and SGO represent metaheuristic algorithms whose mechanism is not simple, but the calculation is straightforward. On the other hand, MPA and KMA represent metaheuristic algorithms with complicated mechanisms.

Table 2. Simulation result

| Function | Benchmark Score | | | | | Proposed Model | Better than |
|---|---|---|---|---|---|---|---|
| | PSO | DGO | SGO | MPA | KMA | | |
| Ackley | 1.2539 | 5.7696 | 7.5826 | **0.5845** | 5.8316 | 1.5658 | DGO, SGO, KMA |
| Rastrigin | 8.2983 | 8.2560 | 13.7112 | **0.5092** | 9.4842 | 1.4434 | PSO, DGO, SGO, KMA |
| Griewank | 0.3487 | 1.8907 | 3.3059 | **0.2416** | 1.3936 | 0.3759 | DGO, SGO, KMA |
| Bukin | 0.0295 | 0.9549 | **0.0236** | 0.9866 | 0.0142 | 1.6614 | - |
| Michalewicz | -3.6446 | -3.6344 | -3.3566 | -2.2806 | -3.3984 | **-4.5642** | PSO, DGO, SGO, MPA, KMA |
| Sphere | **0.0002** | 0.3017 | 0.6304 | 0.0007 | 0.1970 | 0.0009 | DGO, SGO, KMA |
| Rosenbrock | 50.3277 | 851.3852 | 689.6760 | **4.0196** | 28.9723 | 4.1941 | PSO, DGO, SGO, KMA |
| Bohachevsky | **0** | 7.0106 | 4.3290 | 0.0852 | 0.4220 | 0.1398 | DGO, SGO, KMA |
| Zakharov | 0.7140 | 25.2796 | 20.1854 | 0.0082 | 7.0872 | **0.0071** | PSO, DGO, SGO, MPA, KMA |
| Booth | **6.9638e-28** | 0.7539 | 0.2577 | 0.2630 | 0.1423 | 6.6268e-05 | DGO, SGO, MPA, KMA |

All algorithms (the proposed and benchmark algorithms) are set with several common parameters. The population size is 20. The maximum iteration is 100.

The specific setting for these algorithms is as follows. In PSO, the current speed, cognitive, and social weights are set at 0.5. In SGO, the threshold between the first and second states is set to be 0.05. Meanwhile, the threshold between the second and third states is set at 0.025. In MPA, the fish aggregating device (FAD) is set at 0.2. In KMA, the big male proportion is 0.2, the number of females is 1, and the radius of parthenogenesis is 0.5. In the proposed algorithm, the probabilistic ratio is set at 0.5, and the number of attracting objects is set at 20.

The number of decisions for Bukin, Bohachevksy, and Booth functions is 2, while for other functions, 5. In this simulation, the step size and observation range are set depending on the benchmark functions, which can be found in Table 1. A more extensive search space means a more extended step size and a broader observation range. The simulation runs 30 times for every function to get the result shown in Table 2. In Table 2, the best score is written in bold font.

Table 2 shows that the proposed algorithm is proven to achieve the metaheuristic algorithm's objective. It can find a near-optimal solution in all benchmark functions, both multimodal and unimodal. The result also shows that the algorithm can avoid the local optimal trap in the five benchmark functions.

Table 2 also shows that the proposed algorithm is competitive enough compared with the other five algorithms. The proposed algorithm outperforms all benchmark algorithms in solving Michalewicz and Zakharov functions. In general, this proposed algorithm is very competitive compared with DGO, SGO, and KMA. It is better than at least three algorithms in solving nine functions. Unfortunately, this proposed algorithm is not competitive in solving the Bukin function.

The MPA becomes the algorithm that is most difficult to beat. Its performance is the best in solving four functions. But the proposed algorithm is still better than MPA in solving three functions: Michalewicz, Zakharov, and Booth.

The second simulation is conducted to observe the relation between the probabilistic ratio and the proposed algorithm. There are three values of the probabilistic ratio (0.1; 0.5; 0.9). The result is shown in Table 3. The first value represents very low probabilistic ratio. On the other side, the third value represents a very high probabilistic ratio. The best score for every function is in bold font.

Table 3 shows that, in general, a higher probabilistic ratio can improve the performance of the proposed algorithm. The best fitness score is achieved in all multimodal functions when the probabilistic ratio is very high. On the other side, a higher probabilistic ratio can improve the performance in solving Rosenbrock, Bohachevsky, and Zakharov functions. Meanwhile, in the other two unimodal functions, a higher probabilistic ratio fails to improve the performance. In solving sphere function, its score tends to fluctuate.

The third simulation is conducted to evaluate the relation between the number of attracting objects and the performance of the proposed algorithm. In this simulation, there are three values of attracting objects (5, 10, 30). In this work, the probabilistic ratio is set at 0.5. The result is shown in Table 4.

Table 3. Relation between probabilistic ratio and average fitness score

| Function | Average Fitness Score | | |
|---|---|---|---|
| | $p$ = 0.1 | $p$ = 0.5 | $p$ = 0.9 |
| Ackley | 3.0213 | 1.5749 | **0.9878** |
| Rastrigin | 3.3066 | 1.6603 | **1.2988** |
| Griewank | 0.4604 | 0.3473 | **0.2543** |
| Bukin | 1.3554 | 1.7357 | **1.3041** |
| Michalewicz | -4.5132 | -4.5749 | **-4.5900** |
| Sphere | **0.0009** | 0.0011 | 0.0010 |
| Rosenbrock | 4.1050 | 4.1007 | **4.0156** |
| Bohachevsky | 0.2778 | 0.1247 | **0.0539** |
| Zakharov | 0.0197 | 0.0082 | **0.0061** |
| Booth | **6.199e-5** | 6.656e-5 | 7.573e-5 |

Table 4. Relation between number of attracting objects and average fitness score

| Function | Average Fitness Score | | |
|---|---|---|---|
| | $m$ = 5 | $m$ = 10 | $m$ = 30 |
| Ackley | 4.8765 | 2.2471 | **1.3572** |
| Rastrigin | 4.8102 | 2.8249 | **1.1204** |
| Griewank | 0.5051 | 0.4222 | **0.3099** |
| Bukin | 10.0043 | 2.1702 | **1.2798** |
| Michalewicz | -4.3143 | -4.4809 | **-4.5964** |
| Sphere | 0.0018 | 0.0014 | **0.0009** |
| Rosenbrock | 75.2478 | 14.6083 | **3.9484** |
| Bohachevsky | 0.3875 | 0.2359 | **0.1272** |
| Zakharov | 1.0516 | 0.0213 | **0.0052** |
| Booth | 0.0005 | 0.0001 | **4.3155e-05** |

Table 4 shows that, in general, the increase of the number of attracting objects improves the fitness score, mainly when the number of attracting objects ranges from 5 to 30. But its significance may be different among functions. The improvement is significant for solving Bukin, Rosenbrock, Zakharov, and Booth functions within this range. Meanwhile, its improvement is moderate for solving Ackley, Rastrigin, and Sphere functions. In the end, its improvement is not significant for solving Griewank and Michalewicz functions.

## 5. Discussion

There are several findings due to the simulation result. The first finding is that the proposed algorithm is proven as a good metaheuristic algorithm. The algorithm is good in solving problems whose problem space is narrow or large. It can find a near-optimal solution in both unimodal and multimodal functions. Moreover, the problem space does not affect its performance.

The second finding is that the result strengthens the no-free-lunch theory, which says that no algorithm is the best for all problems. The algorithm's performance depends on the problem it tries to solve [28]. The proposed algorithm is very competitive in solving Michalewicz and Zakharov functions. On the other hand, it is less competitive in solving the Bukin function.

The third finding is that a higher probabilistic ratio is preferred, especially in solving multimodal problems. Table 3 shows that its performance is the best when the probabilistic ratio is set high. On the other hand, in solving a unimodal problem, the response to the probabilistic ratio varies depending on the problem's characteristics. Theoretically, a higher probabilistic ratio makes every agent prefers to exploit the global best solution after it reaches its destination. More effort is conducted around the global best solution rather than exploring search space randomly. Diversification is still conducted when the agent moves from its position to its current destination. The current worse solution is still accepted during this process if the agent has not reached its destination yet. This circumstance is like simulated annealing, but accepting the worse solution is not based on probabilistic calculation.

The fourth finding is that, in general, the number of attracting objects relates to increasing the proposed algorithm's performance. However, the response is various depending on the characteristic of the problems. The improvement is significant in solving unimodal problems but less significant in solving multimodal problems.

The complexity of this proposed algorithm can be presented as $O(t_{max}nm)$. The explanation is as follows. This algorithm contains three loops. First, the iteration process until the maximum iteration is reached. The second loop is the action of every agent in every iteration. The third loop is generating particular attractive objects for every agent in every iteration. The complexity is proportional to each of these three variables when the other ones are set constant.

There are notes related to this work. In general, all metaheuristic algorithms consist of several adjusted parameters. These parameters can be adjusted to improve the performance of the algorithm. The adjusted parameters in the proposed algorithm and the benchmark algorithms are fixed for all ten benchmark functions. It means that the constellation between the proposed algorithm and the benchmark algorithms may be different with different settings. Based on this circumstance, it is not wise to judge that an algorithm is better than other algorithms in solving all problems.

Meanwhile, there are several easy methods to improve the performance of metaheuristic algorithms. First, the performance can be improved by increasing the maximum iteration. As an iterative method, there is the possibility that the future solution is better than

the current solution except if it is trapped in the local optimal. Moreover, the performance can also be improved for the population-based algorithm by increasing the population size. These two factors also become reasons why simple old-school algorithms are still popular, although the shortcomings often beat them.

## 6. Conclusion

This work has demonstrated that the modified social forces algorithm is proven as a competitive metaheuristic optimization method. Its performance is good enough in solving both multimodal and unimodal problems. It means this method is proven to avoid the optimal local trap. It is competitive compared with shell game optimization, darts game optimizer, and Komodo mlipir algorithm. The algorithm performance is superior in solving Michalewicz and Zakharov functions but less competitive in solving the Bukin function.

There are several research potentials related to this work. First, this algorithm still needs improvement and modification in solving many other functions. Second, this algorithm still needs modification to solve a combinatorial optimization problem. Third, implementing this algorithm into real-world cases is also challenging. Moreover, making this model becomes widely used is another challenge.

## Conflicts of interest

The authors declare no conflict of interest.

## Author contributions

Conceptualization: Kusuma; methodology: Kusuma, software: Kusuma; validation: Kusuma and Adiputra; formal analysis: Kusuma and Adiputra; investigation: Kusuma and Adiputra; writing-original paper draft: Kusuma; writing-review and editing: Adiputra; funding acquisition: Kusuma.

## Acknowledgments

This work is funded and supported by Telkom University, Indonesia.

## References

[1] H. R. Moshtaghi, A. T. Eshlaghy, and M. R. Motadel, "A Comprehensive Review on Meta-Heuristic Algorithms and Their Classification with Novel Approach", *Journal of Applied Research on Industrial Engineering*, Vol. 8, No. 1, pp. 63-89, 2021.

[2] H. Stegherr, M. Heider, and J. Hahner, "Classifying Metaheuristics: Towards a Unified Multi-Level Classification System", *Natural Computing*, pp. 1-17, 2020.

[3] S. Katoch, S. S. Chauhan, and V. Kumar, "A Review on Genetic Algorithm: Past, Present, and Future", *Multimedia Tools and Application*, Vol. 80, pp. 8091-8126, 2021.

[4] T. Guilmeau, E. Chouzenoux, and V. Elvira, "Simulated Annealing: A Review and a New Scheme", In: *Proc. of 2021 IEEE Statistical Signal Processing Workshop (SSP)*, Rio de Janeiro, Brazil, pp. 101-105, 2021.

[5] 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.

[6] J. Bi, Z. Wu, L. Wang, D. Xie, and X. Zhao, "A Tabu Search-Based Algorithm for Airport Gate Assignment: A Case Study in Kunming, China", *Journal of Advanced Transportation*, Vol. 2020, ID: 8835201, pp. 1-13, 2020.

[7] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A New Heuristic Optimization: Harmony Search", *Simulation*, Vol. 76, No. 2, pp. 60-68, 2001.

[8] A. R. Mehrabian and C. Lucas, "A Novel Numerical Optimization Algorithm Inspired from Weed Colonization", *Ecological Informatics*, Vol. 1, No. 4, pp. 355-366, 2006.

[9] D. Freitas, L. G. Lopes, and F. M. Dias, "Particle Swarm Optimisation: A Historical Review Up to the Current Developments", *Entropy*, Vol. 22, No. 3, ID: 362, pp. 1-36, 2020.

[10] 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.

[11] A. M. F. Fard, M. H. Keshteli, and R. T. Moghaddam, "Red Deer Algorithm (RDA): A New Nature-Inspired Meta-Heuristic", *Soft Computing*, Vol.19, No. 1, pp.1-29, 2020.

[12] M. Braik, A. Sheta, and H. A. Hiary, "A Novel Meta-Heuristic Search Algorithm for Solving Optimization Problems: Capuchin Search Algorithm", *Neural Computing and Applications*, Vol. 33, pp. 2515-2547, 2020.

[13] 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.

[14] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine Predators Algorithm: A Nature-Inspired Metaheuristic", *Experts Systems with Applications*, Vol. 152, ID: 113377, pp. 1-48, 2020.

[15] Suyanto, A. A. Ariyanto, and A. F. Ariyanto, "Komodo Mlipir Algorithm", *Applied Soft Computing*, Vol. 114, ID: 108043, pp. 1-17, 2022.

[16] 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*, Vol. 7, pp. 1-17, 2021.

[17] M. Dehghani, Z. Montazeri, O. P. Malik, H. Givi, and J. M. Guerrero, "Shell Game Optimization: A Novel Game-Based Algorithm", *International Journal of Intelligent Engineering & Systems*, Vol. 13, No. 3, pp. 246-255, 2020, doi: 10.22266/ijies2020.0630.23.

[18] M. Dehghani, 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, doi: 10.22266/ijies2020.1031.26.

[19] I. M. Mehedi, N. Saad, M. A. Magzoub, M. A. Saggaf, and A. H. Milyani, "Simulation Analysis and Experimental Evaluation of Improved Field-Oriented Controlled Induction Motors Incorporating Intelligent Controllers", *IEEE Access*, Vol. 10, pp. 18380-18394, 2022.

[20] J. Son, J. Cha, H. Kim, Y. M. Wi, "Day-ahead Short-Term Load Forecasting for Holidays Based on Modification of Similar Days' Load Profiles", *IEEE Access*, Vol. 10, pp. 17864-17880, 2022.

[21] K. Khadir, N. Guermouche, T. Monteil, and A. Guittoum, "A Genetic Algorithm based Approach for Fluctuating QoS Aware of IoT Services", *IEEE Access*, Vol. 10, pp. 17946-17965, 2022.

[22] M. Xu and C. Li, "Data Mining Method of Enterprise Human Resource Management Based on Simulated Annealing Algorithm", *Security and Communication Networks*, Vol. 2021, ID: 6342970, pp. 1-9, 2021.

[23] D. Helbing and P. Molnar, "Social Forces Model for Pedestrian Dynamics", *Physical Review E*, Vol. 51, No. 5, pp. 4282-4286, 1995.

[24] M. S. Umam, Mustafid, and Suryono, "A Hybrid Genetic Algorithm and Tabu Search for Minimizing Makespan in Flow Shop Scheduling Problem", *Journal of King Saud University – Computer and Information Sciences*, pp. 1-9, 2021, in press.

[25] X. Tian and X. Liu, "Improved Hybrid Heuristic Algorithm Inspired by Tissue-Like Membrane System to Solve Job Shop Scheduling Problem", *Processes*, Vol. 9, ID: 219, pp. 1-18, 2021.

[26] 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.

[27] S. Harifi, M. Khalilian, J. Mohammadzadeh, and S. Ebrahimnejad, "Emperor Penguins Colony: A New Metaheuristic Algorithm for Optimization", *Evolutionary Intelligence*, Vol. 12, No. 2, pp. 211-226, 2019.

[28] 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.