Dove swarm optimization algorithm

Mu-Chun Su¹, Jieh-Haur Chen²,³, Andina Mugi Utami³, Shih-Chieh Lin⁴, Hsi-Hsien Wei⁵

¹Department of Computer Science and Information Engineering, National Central University, Jhongli, Taoyuan 320317, Taiwan. Email: mchu@csie.ncu.edu.tw
²Department of Civil Engineering; Research Center of Smart Construction, National Central University, Jhongli, Taoyuan 320317, Taiwan. Email: jhchen@ncu.edu.tw
³Department of Civil Engineering, National Central University, Jhongli, Taoyuan 320317, Taiwan. Email: andina@ncu.edu.tw
⁴Department of Computer Science and Information Engineering, National Central University, Jhongli, Taiwan. E-mail: ya-kumolin@gmail.com
⁵Department of Building and Real Estate, Hong Kong Polytechnic University, Hong Kong. Email: bhwei@polyu.edu.hk

Corresponding author: Jieh-Haur Chen (e-mail: jhchen@ncu.edu.tw).

This work was partly supported by the Ministry of Science and Technology (MOST), Taiwan, under their promoting academic excellent of universities program, grant numbers MOST 110-2622-E-008-018-CC2, 110-2221-E-008-052-MY3, 109-2634-F-008-007, 108-2221-E-008-002-MY3, and 107-2221-E-008-084-MY2.

ABSTRACT Popular methods to deal with computation become strenuous due to the optimization demands that develop more complex nowadays. This research aims to propose a new optimal algorithm, Dove Swarm Optimization (DSO), that adopts the foraging behaviors of doves to have six benchmark functions expressing DSO performance. By considering competition for forage, DSO is designed to ensure the most satisfied dove as well as optimization, then compared with 15 popular optimization algorithms using random initial and lattice initial values. The results reveal that DSO performs the best in time efficiency and well in both convergences for these functions in a reasonable region from 1 to 3 seconds, and population diversity for the initialization method from less than 1 second to 9 seconds dependent on the population size. As a result, DSO is indeed a time-efficient and effective algorithm in solving optimization problems.

INDEX TERMS swarm intelligence, optimization algorithm, computational intelligence.

I. INTRODUCTION

Optimization is a method to generate the optimal solution from all of the possible solutions. Over the decades, the optimization methods have been evolved. The researcher have figured out optimization to discover the most suitable possible solutions by using various mechanisms. Optimization algorithms have become a useful tool in numerous research domains. In the engineering domains, it is prevalent to utilize metaheuristic algorithms such as: Genetic algorithm (GA). GA was developed based on the theory of Darwin. The swarm intelligence (SI) algorithm is typically defined as “nature-inspired algorithms that concern the collective, emerging behavior of multiple, interacting agents that follow some simple rules”. It is based on social behavior from swarms (e.g. troops) of the organism in nature. The SI advantages are to explore creatures’ working mechanism [1-5]. They are relevant because of their simplicity of inspiration, flexibility, derivative-free mechanism, and local optimum avoidance such as Particle Swarm Optimization (PSO), Ant Colony Optimizer (ACO), and Artificial Bee Colony (ABC) [9-15]. They may fit better in particular demands [16,17]. Moreover, although hardware techniques have developed rapidly to deal with computation, the optimization demands become more complex nowadays. It is always a strong desire to develop an SI algorithm designed for optimization problem more efficiently. By considering competition for forage, DSO is designed to ensure the most satisfied dove as well as optimization. As a result, the research objective is to develop the dove swarm optimization (DSO) algorithm using SI concepts in solving optimization problems.

II. OPTIMIZATION ALGORITHMS

The optimal algorithms can be categorized into two classes by mathematics. One is the derivative-based approach (e.g., steepest descent method and Newton’s method etc.), and the other is the derivative-free approach (e.g., GAs, simulated annealing, evolutionary programming, evolutionary strategies, and swarm intelligence such as ACO and PSO etc. [9-15]). The derivative-based algorithm is also a gradient-based algorithm that can determine the search
direction according to the derivative information of the objective function. The derivative-free algorithm is an iterative and random search algorithm that uses a set of operators or mechanisms to find the global optimal solution in the large solution space. The basic concepts of these two optimizations and popular algorithms are shown in the following sections.

2.1 Derivative-Based Optimization

The derivative-based algorithm is a gradient-based algorithm that can determine the search direction according to the derivative information of its objective function(s). There are two most popular derivative-based optimal algorithms: the steepest descent method and Newton’s method. These two algorithms are the foundation of most gradient-based algorithms. Instrumental algorithms can be regarded as a form of compromise between steepest descent method and Newton’s methods. The derivative-based algorithm is usually applied to optimizing nonlinear system models, allowing such models to play a prominent role in the framework of soft computing. In fact, it is one of the major algorithms used for neural network learning. In addition, the least-squares method is another widely employed algorithm because the sum of squared errors is chosen as the object function to be minimized for empirical cases [18-23].

2.2 Derivative-Free Optimization

The derivative-free algorithm is an iterative and random search algorithm using a set of operators or mechanisms to find the global solution in the large solution space. Common characteristics shared by these methods are derivative freeness, intuitive guidelines, slowness, flexibility, randomness, analytic opacity, and iterative nature. There are popular derivative-free optimization methods: evolutionary computation (EC), simulated annealing (SA), and random search method. The EC methods are included in Genetic algorithms (GA), Evolutionary Programming (EP), Evolution Strategies (ES), and Genetic Programming (GP). Of those four in EC methods, majority have been done with GA that is typical derivative-free stochastic optimization methods based on the concepts of natural selection and evolutionary processes. As a general-purpose optimization tool, GA has been moving out of academia and finding significant applications in many other venues. Numerous researchers pay attention to both EC and SA due to their optimization capabilities for both continuous and discrete optimization problems. Both of them are motivated by so-called nature’s wisdom: EC is loosely based on the concepts of natural selection and evolution while SA originated in the annealing processes found in thermo-dynamics and metallurgy. Random search is primarily for continuous optimization problems and it is typically the simplest and most intuitive optimization scheme. Although the concept and implementation of random search is simpler than those of EC and SA, it cannot be inferred that EC and SA outperform for all problems all the time. In general, one would not expect any single technique to outperform all the others in a given application.

2.3 Swarm Intelligence

The swarm intelligence is also a popular derivative-free optimization method recently. It usually adopts a population of simple agents based on social-psychological to find the optimal solution. Many properties of social insect (or animals) collective behaviors have attracted a great amount of attention from researchers. Social insects have inspired us with a powerful concept to create decentralized systems of simple interacting, and often mobile, agents (e.g., ants, bees, birds). A rich source of mechanisms in social insect collective behaviors may serve as metaphors for designing the so-called swarm-intelligence-based systems [18-23].

Swarm intelligence is the emergent collective intelligence of groups of simple agents which communicate directly or indirectly with each other, and which collectively carry out distributed problem solving. There are swarm intelligence algorithms for optimization such as ACO, PSO, and glowworm swarm optimization (GSO) etc. The Study regarding bird flocking and fish schooling was already a research topic for social psychology in the 1930s. An influential simulation of bird flocking was proposed where the study assumed that flocking birds were attributed to the following three local forces: collision avoidance, velocity matching, and flocking centering. Additionally, a similar idea about bird flocking was also introduced at about the same time. A rich source of mechanisms in bird flocking and fish schooling may serve as metaphors for designing computational systems. Different computational systems are inspired by different subsets of the available metaphors [24].

III. DOVE SWARM OPTIMIZATION ALGORITHM

People easily observe that doves forage at plazas where there are crumbs around and each dove searches crumbs. Some doves may be satisfied but not for all. One may observe that unsatisfied doves fly forward to spots for more crumbs. Gradually, we can see that fed doves can occupy spots with most crumbs. The doves’ foraging behavior has motivated ones to propose a novel optimal algorithm. In this method, the optimization objective function is \( f(W) \). Each data pattern, \( W \), in a data set is regarded as a position with crumbs and the amount of crumbs in these place, \( W \) has \( f(W) \) crumbs. The best solution means where it is the place with the most crumbs. Figure 1 demonstrates the flowchart for the DSO algorithm.
Two efficient weight initialization schemes were proposed to initialize the weight vectors to accelerate the training process for constructing a topologically ordered feature map. Based on the initialization scheme, we propose an initialization method especially suitable for the algorithm. Let the smallest hyper-rectangle for the parameter space, which contains the valid values of all the parameters, be denoted as $[l_1,u_1], \ldots, [l_M,u_M]$ where $l_a$ and $u_a$ represent the low bound and the up bound of the a-dimension in the solution space. The basic idea of the proposed initialization method is to squeeze the n-dimensional hyper-rectangle into a 2-dimensional plane so that a two-dimensional net can effectively cover the solution space. For the clarity purpose, we use $i$ and $j$ to index the rectangular cells from 1 to $A \times B$. The detail steps are as follows:

**Step 2-1. Initialization of the cells on the four corners:** The weight vectors of the four neurons on the corners of the network are initialized as (1).

$$w_{A,1} = (l_1, l_2, \ldots, l_M)^T$$  
$$w_{A,B} = (u_1, u_2, \ldots, u_M)^T$$  
$$w_{1,B} = (l_1, l_2, \ldots, l_{M/2}, u_{M/2}, \ldots, u_M)^T$$  
$$w_{A,1} = (u_1, u_2, \ldots, u_{M/2}, l_{M/2}, \ldots, l_M)^T$$

**Step 2-2. Initialization of the cells on the four edges:** We initialize the cells' value on the four edges according to (2):

$$w_{i,j} = \frac{w_{i,j} - w_{i-1,j}}{B-1} + \frac{w_{i+1,j} - w_{i,j}}{B-1}, \quad \text{for } j = 2, \ldots, B-1$$  
$$w_{i,j} = \frac{w_{i,j} - w_{i,j-1}}{A-1} + \frac{w_{i,j+1} - w_{i,j}}{A-1}, \quad \text{for } i = 2, \ldots, A-1$$

**Step 2-3. Initialization of the remaining cells:** The weight vectors of the four neurons on the corners of the network are initialized.

We initialize the remaining neurons from top to bottom and from left to right. The pseudo-code description of the initialization method for the remaining neurons is given as follows:

```plaintext
Begin 
For j from 2 to B-1 
Begin 
For i from 2 to A-1 
Begin 
End 
End 
End 
End 
End 
End
```

**FIGURE 1. Flowchart of DSO**

**Step1:** Decide the number of doves and then deploy them on the solution space. Assume that the number of doves is pre-specified to be $N$. These doves can be randomly distributed on the space; however, we suggest deploying them uniformly on a rectangular region.

**Step2:** Set the number of epochs, $E=0$ and set the degree of satiety, $s^d_d$ for dove $d$, $d = 1, \ldots, N$. The initialization of the position vector $W_d \subset R^d$ of dove $d$ can be done in two ways. The simplest way is to randomly initialize $W_d$ around the solution space. The other way is to initialize lattice initialized method. The steps are shown as follows:
\[
\frac{w_j^{t+1} = w_j^t + \eta \beta_j^t (w_d^t - w_j^t)}{maxDistance: \max_{1 \leq j \leq N} \|w_j - w_d\|}
\]  

The parameters, \( \eta \), is the learning rate for updating the dove position vector, respectively. Detailed descriptions of the updating Equations (8)-(10) are given in next step.

**Step8:** Go to step3 and increase the number of epochs by one (i.e., \( e = e+1 \)) until the terminate condition is met. The terminate condition is as follows.

\[|f_{d_j^e} - T(e)| \leq \varepsilon \quad \text{or} \quad e \leq \text{the set maxepoch} \quad (11)\]

The dove swarm optimization algorithm has the order of complexity, \( O(\mathcal{N}N_a e) \) where \( N_a \) is the number of data points in the data set, \( N \) is the number of doves, and \( e \) is the number of epochs.

If the optimization is the minimum criterion that it's the best solution to find the minimum \( w_j^e \), then (5) and (6) can be respectively change to (12) and (13).

\[d_j^e = \arg \min \{f(w_j^e)\}, \text{ for } j = 1, ..., N \quad (12)\]

\[S_j^e = \begin{cases} \lambda S_j^{e-1} + e^{-f(w_j^e)-f(w_{d_j}^e)}, & \text{if } f(w_{d_j}^e) \neq 0 \\ \lambda S_j^{e-1} + 1, & \text{if } f(w_{d_j}^e) = 0 \end{cases}, \text{ for } j = 1, ..., N \quad (13)\]

For easier understanding, we interpret the updating rules given in Equations (8)-(10) as follows:

1. An individual is influenced by the success of the best individual in the flock and tries to imitate the behavior of the best individual. That is, doves move toward the dove with the highest degree of satiety to find more food. This social learning is simulated by updating the position vector \( w_j^e \) to be more like the position vector of the dove with the highest degree of satiety \( w_{d_j}^e \). (i.e., \( w_j^{e+1} = w_j^e + \eta \beta_j^e (w_{d_j}^e - w_j^e) \)).
2. When a dove is with a higher degree of satiety, it is prone to become conservative and would hesitate to change its present foraging policy. On the contrary, when a dove is with a lower degree of satiety it would probably have a strong desire to change its present foraging policy and be more willing to imitate the behavior of the best individual. This social influence is simulated by making the adjustment proportional to the value of the first term on the right-hand-side of (9). (i.e., \( \frac{s_j^e - s_j^f}{s_j^e} \)).
3. Basically, social impact gradually decays as it spreads out; therefore, the degree of impact is inversely proportional to the distance between the
IV. RESULT AND RULES OF AUDITING STANDARDS

For the experiments, assume that the population size is pre-set to 100 for all algorithms including GA, and PSO algorithms. The maximum generations are pre-set to 100, either. For all the simulations, the parameters for all algorithms remain the same. The settings are as follows: In the GA, each variable was represented by 16 bits and the crossover probability and the mutation probability are set to 0.75 and 0.05, respectively. In PSO, the maximum velocity is set to 10 and the inertial weight decreased linearly from 0.9 to 0.4. The parameters for the proposed DSO algorithm are set \( \lambda = 0.9 \), and \( \eta = 0.18 - 0.375 \).

It is necessary for new developed optimization algorithms to be evaluated using benchmark/test functions. There are numerous functions available where some are recommended including De Jong f4, Rastrigin, Giunta, Griewank, Ackley, and Rosenbrock functions [25]. This study adopts these six benchmark functions to evaluate DSO, and the results of DSO and linear mapping methods. They are expressed in mathematical way as follows [26-29]:

1. De Jong f4 function:
\[
f(x) = \sum_{i=1}^{50} i \cdot x_{i}^{4} \tag{14}
\]

2. Rastrigin function:
\[
f(x) = \sum_{i=1}^{50} (x_{i}^{2} - 10 \cos(2\pi x_{i}) + 10) \tag{15}
\]

3. Giunta function:
\[
f(x) = \sum_{i=1}^{30} \sin\left(\frac{16}{15} x_{i} - 1\right) + \sin\left(\frac{16}{15} x_{i} - 1\right) + 0.3 \tag{16}
\]

4. Griewank function:
\[
f(x) = \frac{1}{4000} \sum_{i=1}^{50} (x_{i} - 100)^{2} - \prod_{i=1}^{50} \cos\left(\frac{x_{i} - 100}{\sqrt{i}}\right) + 1 \tag{17}
\]

5. Ackley function:
\[
f(x) = -20 \cdot \exp\left(-0.2 \cdot \frac{1}{30} \sum_{i=1}^{30} x_{i}^{2}\right) - \exp\left(\frac{1}{30} \sum_{i=1}^{30} \cos(2\pi x_{i})\right) + 20 + e \tag{18}
\]

6. Rosenbrock function:
\[
f(x) = \sum_{i=1}^{30} [100(x_{i+1} - x_{i}^{2})^{2} + (x_{i} - 1)^{2}] \tag{19}
\]

The global minimum of Giunta is near 0.9, and all others is zero. For the comparison purpose, we also utilize the GA, the PSO algorithm, and some other 15 existing popular methods such as SOC [30,31], SOC 2[30,31], DSPO[32-34], FAPO 1, FAPO 2, and arPSO [32-34], SFS [35], MCS [35,36], GSA [35,37], ABC [35,38], DHS [39], CKH [40], MBO [41], IHS[40], and HS[40] to test these six functions. The results are in Tables 1-6.

| Algorithm | Population size | Minimum generations | Runs | Initial Range | Best solution | Time (sec.) |
|-----------|----------------|---------------------|------|---------------|---------------|-------------|
| DSO       | 100            | 100                 | 100  | (-32,520)     | 1.43E-04       | 0.00658     |
| PSO       | 100            | 100                 | 100  | (-30,20)      | 2.34E-05       | 0.01777     |
| GA        | 100            | 100                 | 100  | (-10,0)       | 1.16E-05       | 0.01777     |
| DSPO      | 80             | 10000               | 200  | (-20,20)      | 9.51E-04       | NA          |
| arPSO     | 25             | 12000               | 30   | (-1.28,1.28)  | 2.08E-16       | 0.01777     |
| SFS       | 100            | 1000                | 25   | (-3.2,3.2)    | 3.05E-04       | NA          |
| MCS       | 100            | 1500                | 25   | (-1.28,1.28)  | 1.55E-05       | NA          |
| GSA       | 100            | 1500                | 25   | (-1.28,1.28)  | 4.20E-05       | NA          |
| ABC       | 100            | 1500                | 25   | (-3.2,3.2)    | 6.32E-11       | 0.01777     |

| Algorithm | Population size | Minimum generations | Runs | Initial Range | Best solution | Time (sec.) |
|-----------|----------------|---------------------|------|---------------|---------------|-------------|
| DSO       | 100            | 100                 | 10   | (-30,20)      | 1.03E-01       | 0.04677     |
| PSO       | 100            | 100                 | 10   | (-10,10)      | 4.33E-01       | 0.01777     |
| GA        | 100            | 100                 | 10   | (-10,10)      | 4.17E-01       | NA          |
| arPSO     | 25             | 12000               | 30   | (-1.28,1.28)  | 2.14E-01       | 0.01777     |
| FAPSO 1   | 80             | 1500                | 50   | (-1.28,1.28)  | 3.14E-01       | NA          |
| FAPSO 2   | 80             | 1500                | 50   | (-1.28,1.28)  | 1.09E-01       | NA          |
| SOC 1     | 400            | 800                 | 50   | (-1.28,1.28)  | 2.14E-01       | 0.01777     |
| SOC 2     | 400            | 800                 | 50   | (-1.28,1.28)  | 0.45E-05       | NA          |
| DSS       | 10             | 50,000              | 50   | (-32,32)      | 6.56E-01       | 0.01777     |
| CKH       | 50             | 50                  | 100  | (-32,32)      | 1.69E-01       | NA          |

| Algorithm | Population size | Minimum generations | Runs | Initial Range | Best solution | Time (sec.) |
|-----------|----------------|---------------------|------|---------------|---------------|-------------|
| DSO       | 100            | 100                 | 10   | (-10,10)      | 9.67E-01       | 0.05208     |
| PSO       | 100            | 100                 | 10   | (-10,10)      | 9.69E-01       | 0.07812     |
| GA        | 100            | 100                 | 10   | (-10,10)      | 2.46E-01       | 0.34063     |
| DSPO      | 80             | 10000               | 200  | (-10,10)      | 1.37E-01       | NA          |

| Algorithm | Population size | Minimum generations | Runs | Initial Range | Best solution | Time (sec.) |
|-----------|----------------|---------------------|------|---------------|---------------|-------------|
| DSO       | 100            | 100                 | 10   | (-0.00,006)   | 6.28E-02       | 0.00594     |
| PSO       | 100            | 100                 | 10   | (-0.00,006)   | 2.46E-02       | 0.14067     |
| GA        | 100            | 100                 | 10   | (-0.00,006)   | 1.35E-01       | NA          |
| arPSO     | 25             | 12000               | 30   | (-0.00,006)   | 1.58E-01       | NA          |
| MBO       | 51             | 1000                | 50   | (-32,32)      | 1.20E+01       | NA          |

| Algorithm | Population size | Minimum generations | Runs | Initial Range | Best solution | Time (sec.) |
|-----------|----------------|---------------------|------|---------------|---------------|-------------|
| DSO       | 100            | 100                 | 10   | (-32,32)      | 7.97E-01       | 0.09140     |
| PSO       | 100            | 100                 | 10   | (-32,32)      | 4.82E+02       | 0.04778     |
| GA        | 100            | 100                 | 10   | (-32,32)      | 4.97E+00       | 0.234        |
| arPSO     | 25             | 12000               | 30   | (-32,32)      | 1.84E+03       | 0.88964     |
| MBO       | 51             | 1000                | 50   | (-32,32)      | 1.20E+01       | NA          |
In Tables 1-6, we tabulate the comparison of the simulation results of the proposed DSO algorithm with others. The mean column and the standard deviation column represent the mean and the standard deviation of the best solutions of 10 runs. The comparison of algorithms on the computational time as shown in the last column. All these five optimal algorithms were run on an Intel(R) Core(TM) i7 2.93GHz computer with 4 GB RAM under Microsoft Windows 7 operating system. For the overall results compared in the tables, DSO outruns GA and PSO. Although the standard deviation of DSO’s results is slightly larger than that of PSO’s due to initial input values, we employ the lattice initial method in order to resolve it shown in Table 7. There is a comparison for the convergence performance among functions including De Jone f4, Rastrigin, Giunta, Griewank, Ackley, and Rosenbrock. The convergence for these functions fall into a reasonable region from 1 to 3 seconds. The results yielded from DSO outperform that from PSO. DSO is an efficient and effective algorithm in solving optimization problems. Table 8 demonstrates comparative experiments of population diversity for the initialization method. The results all fall in a reasonable range from less than 1 second to 3 seconds dependent on the population size.

The strength for the proposed algorithm relative to well-established algorithms described in tables 1-6 mainly lie on time required to yield the results, which is the most efficient. This is a significant advantage compared to other algorithms if dealing with complex demands. Problem demands gradually become complicated nowadays due to multi-aspect considerations in reality. Time-efficiency usually implies better capability in dealing with higher dimensional problems. Nevertheless, achieving time efficiency may cost a slight gap to the optimization. Tables 1-6 demonstrate this phenomenon where DSO still yields one of the best solutions. Although it is typically a trade-off between optimization and time-efficiency, the proposed algorithm performs well for both.

### Table 7: The Results of the Lattice Initial Methods

| Algorithm | Population size | Maximum generations | Runs | Initial Range | Best solution | Time(s) |
|-----------|-----------------|---------------------|------|---------------|---------------|---------|
| DSO       | 100             | 100                 | 10   | (-30,30)      | 8.876-01-5.13E-02 | 0.035   |
| GA        | 100             | 100                 | 10   | (-30,30)      | 2.99E-01-2.77E-00 | 0.043   |
| DSPSO     | 60              | 10000               | 200  | NA            | 4.77E+01      | NA      |
| acPSO     | 25              | 12000               | 30   | NA            | 1.84E+01-1.53E+01 | NA  |
| FAPSO 1   | 80              | 26000               | 50   | (-15,30)      | 2.63E+02      | NA      |
| FAPSO 2   | 80              | 26000               | 50   | (-15,30)      | 2.63E+02      | NA      |
| SOC 1     | 400             | 800                 | 30   | (-30,30)      | 1.9E+01-2.7E+00 | 0.00   |
| SOC 2     | 400             | 800                 | 30   | (-30,30)      | 6.6E+02-7.72E+01 | NA  |
| MBO       | 51              | 10900               | 30   | (-2,2)        | 3.90E+07      | NA      |
| DRS       | 10              | 50,000              | 50   | (-32,32)      | 8.6E+06-7.6E+06 | NA      |
| IRS       | 10              | 50,000              | 50   | (-32,32)      | 8.1E+06-8.3E+06 | NA      |
| HS        | 10              | 50,000              | 50   | (-32,32)      | 1.0E+07-6.7E+07 | NA      |

### Table 8: Comparative Experiment of Population Diversity for Initialization

| Population size | Avg. Time | Best Solution | Avg. Time | Best Solution |
|-----------------|-----------|---------------|-----------|---------------|
| 10x10           | 0.011     | 1.17E-30      | 0.0139    | 8.15E-26      |
| 25x25           | 0.0406    | 0             | 0.0407    | 4.42E-34      |
| 50x50           | 0.1313    | 4.35E-39      | 0.1313    | 1.01E-34      |
| 100x100         | 0.5172    | 4.54E-41      | 0.5172    | 8.32E-35      |
| 400x400         | 8.3862    | 4.73E-44      | 8.5123    | 4.87E-38      |

### V. CONCLUSION

In this study, we proposed a novel optimal algorithm, DSO, based on the swarm intelligence concept by observing foraging behaviors at doves. We evaluate the proposed algorithm by six standard benchmarks and compare with 15 existing algorithms using random initial value and lattice initial value. In random initial, the result shows that DSO significantly outruns GA in finding solutions and computing time, and so it does to PSO and others slightly. In lattice initial condition, DSO performs much better than PSO and others in both finding solutions and computing time. As a result, DSO is an efficient and effective algorithm in solving optimization problems. The proposed algorithm may have varieties for future work. It can be applied to most optimization problems in science and social science fields such as commination, qualifying, quantifying, costing, scheduling, resource allocating, predicting, etc. Its accuracy and efficiency benefits industries in both academic and practical aspects.

### ACKNOWLEDGMENT

This work was partly supported by the Ministry of Science and Technology (MOST), Taiwan, under their promoting academic excellent of universities program, grant numbers MOST 107-2222-E-008-108-CC2, 100-2221-E-008-052-MY3, 109-2634-F-008-007, 108-2221-E-008-002-MY3, and 107-2221-E-008-084-MY2.

### REFERENCES

[1] S. Mirjalili and S. M. Hashim, “A new hybrid psooga algorithm for function optimization,” in: IEEE international conference on computer and information application, 2010, pp. 374–377.

[2] N. Bharanidharan and H. Rajaguru, “Performance enhancement of swarm intelligence techniques in dementia classification using dragonfly-based hybrid algorithms,” International Journal of Imaging Systems and Technology, vol. 30, no. 1, 2019, pp. 57-74.

[3] J. Tang, G. Liu, Q. T. Pan, “A review on representative swarm intelligence algorithms for solving optimization problems:
applications and trends,” IEEE-CCAA Journal of Automatica Sinica, vol. 8, no. 10, 2021, pp. 1627-1643.

[4] Y.K. Zhou, B. Rao, W. Wang, “UAV swarm intelligence: recent advances and future trends,” IEEE Access, vol. 8, 2020, pp. 183856-183878.

[5] A.J. AL-Mousawi, ”Wireless communication networks and swarm intelligence,” Wireless Networks, vol. 27, no. 3, 2021, 1755-1782.

[6] J. Kennedy, “Particle swarm optimization,” Encyclopedia of machine learn-ing, Springer, US, 2010, pp. 760–766.

[7] M. B. Marco Dorigo, “Ant colony optimization,” Encyclopedia of machine learning, Springer, US, 2010, pp. 36–39.

[8] D. Karaboga, B. Basturk, “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm,” Journal of global optimization, vol. 39, no. 2007, pp. 65-74.

[9] N.A. Ab Aziz, Z. Ibrahim, M. Mubin, S.W. Nawawi, M.S. Mohamad, “Improving particle swarm optimization via adaptive switching asynchronous - synchronous update,” Applied Soft Computing, vol. 72, 2018, pp. 298-311.

[10] W.N. Chen, D.Z. Tan, ”Set-based discrete particle swarm optimization and its applications: a survey,” Frontiers of Computer Science, vol. 12, no. 2, 2018, pp. 203-216.

[11] H.T. Rauf, U. Shoaib, M.I. Lali, M. Alhaisoni, M.N. Irfan, A.M. Khan, ”Particle swarm optimization with probability sequence for global optimization,” IEEE Access, vol. 8, 2020, pp. 110535-110549.

[12] A. Kumar, S. Bawa, “Generalized ant colony optimizer: swarm-based meta-heuristic algorithm for cloud services execution,” Journal of King Saud University – Computer Science and Engineering, vol. 32, no. 1, 2020, pp. 1614-1623.

[13] D. Santhakumar and S. Logeswari, “Hybrid ant lion mutated ant colony optimizer technique for Leukemia prediction using microarray gene data,” Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 2, 2021, pp. 2965-2973.

[14] F. Boudardara and B. Gorkemli, “Solving artificial ant problem using two artificial bee colony programming versions,” Applied Intelligence, vol. 50. No. 11, 2020, pp. 3695-3717.

[15] M. Maeda and S. Tsuda, “Reduction of artificial bee colony algorithm for global optimization,” Neurocomputing, vol. 148, 2015, 70-74.

[16] S. Mirjalili, “Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm,” Knowledge-Based Systems, vol. 89, 2015, pp. 228–249.

[17] S. Mirjalili, and A. Lewis, “The whale optimization algorithm,” Advances in Engineering Software, vol. 95, 2016, pp. 51-67.

[18] R. Mallipeddi, S. Mallipeddi, and P. N. Suganthan, “Ensemble strategies with adaptive evolutionary programming,” Information Sciences, vol. 180, no. 9, 2010, pp. 1571-1581.

[19] P.P. Zhang and Q. Wang, “Perturbation analysis and condition numbers of mixed least squares-scaled total least squares problem,” Numerical Algorithms, Early Access, 2021.

[20] S.T. Barratt and S.P. Boyd, “Least squares auto-tuning,” Engineering Optimization, vol. 53, no. 5, 2020, pp. 789-810.

[21] Y. Li, M.S. Wei, F.X. Zhang, J.L. Zhao, “On accurate error estimates for the quaternion least squares and weighted least squares problems,” International Journal of Computer Mathematics, vol. 97, no. 8, 2020, pp. 1662-1677.

[22] G. Even-Tzur, “Reliability criteria in sequential least-squares adjustment,” Journal of Surveying Engineering, vol. 147. No 3, 2021, pp. 04021007.

[23] E. Ghaderpour and S.D. Pagiatakis, ”LSWAVE: a MATLAB software for the least-squares wavelet and cross-wavelet analyses,” GPS solutions, vol. 23, no. 2, 2019, pp. 50.

[24] M. C. Su, S. Y. Su, and Y. X. Zhao, “A Swarm-Inspired Projection Algorithm,” Pattern Recognition, vol. 42, no. 11, 2009, pp. 2764-2786.

[25] K. Krishnanand, and D. Ghose, “Glowsworm swarm optimisation: a new method for optimising multi-modal functions,” International Journal of Com-putational Intelligence Studies, vol. 1, no. 1, 2009, pp. 93-119.

[26] Y.Q. Zhou, G. Zhou, Y.J. Wang, G.W. Zhao, “A glowworm swarm optimization algorithm based tribes,” Applied Mathematics and Information Sciences, vol. 7, no. 2, 2013, pp. 537-541.
Mu-Chun Su received the B. S. degree in electronics engineering from National Chiao Tung University, Taiwan, in 1986, and the M. S. and Ph.D. degrees in electrical engineering from University of Maryland, College Park, in 1990 and 1993, respectively. He was the IEEE Franklin V. Taylor Award recipient in 1991. He has authored more than 100 journal and refereed conference papers. He is currently a professor of computer science and information engineering at National Central University, Taiwan. He is a senior member of the IEEE Computational Intelligence Society and Systems, Man, and Cybernetics Society. He is also an IET Fellow. He serves as an associate editor for many journals. His current research interests include computational intelligence, neural networks, fuzzy systems, swarm intelligence, affective computing, human-computer interaction, robotics, pattern recognition, biomedical signal processing, image processing, and rehabilitation technology.

Jieh-Haur Chen received the B.S. degree in civil engineering from National Central University (NCU), Taiwan, in 1995, the M.S. degree in project management from Northwestern University, Evanston, IL, USA in 1999, and Ph.D. degree in civil engineering from University of Wisconsin-Madison, WI, USA in 2003. Since 2004, he has been with the Civil Engineering Department of NCU. He is the author of more than 150 articles. His research interests include computational intelligence, managerial finance in engineering, smart city applications, and property management. He is an editor of major engineering-related journals and senior member of National Science Council committee board in the civil engineering area, Taiwan.

Dr. Chen has received the permanent Distinguished Professor title from NCU since 2013. His awards and honors include Extraordinary Military Service Award in 1997, College Outstanding Service Award in 2009, NCU Outstanding Researcher Award of the Year in 2011 and 2012, Special Contribution in Technical Transferring Award in 2015 and 2016, and NCU Excellence in Service Award in 2017.

Andina Mugi Utami received the B.Eng. Degree in civil engineering from The State Polytechnic of Malang (POLINEMA), Indonesia, in 2017, the M.Sc. degree in construction management from National Central University (NCU), Taiwan in 2020, and currently as a Ph.D. student in construction engineering management from National Central University (NCU), Taiwan. Her research interest includes diversion channel design, optimization blasting productivity, construction project investment, safety law comparison, and the optimization algorithm. Since 2017, She has been with PT. Yodya Karya as Dam Engineer in Mega Dam Project (Margatiga Dam - Indonesia), 2017-2018. Her award include Outstanding Student Achievement in The State Polytechnic of Malang (POLINEMA), Indonesia, 2017.

Shih-Chieh Lin received the B.S. degree in electronics engineering from National central university(NCU), Taiwan in 2005, and received the M.S. and Ph.D. degrees in computer science and information engineering from National Central University (NCU), Taiwan in 2007 and 2013. Shih-Chieh has 7 years in the testing and vendor for the semiconductor industry. He has served in software department in KYEC Inc. and D&E software department in ASML. He has expertise in image processing, software layout, firmware development, and application software development. He also has research interests in Neural Network, Artificial Intelligence, and robots.

Hsi-Hsien Wei received the B.S. degree in civil engineering from Tamkang University, Taiwan, and Ph.D. degree in Civil and Environmental Engineering from University of Maryland, College Park. He was a senior research fellow in the Department of Structural Engineering at Ben-Gurion University in Israel from 2014 to 2015. He has been an assistant professor in the Department of Building and Real Estate, the Hong Kong Polytechnic University since 2016. His research interests include digital construction & construction technology, resilience and sustainability assessment of the built environment, and interdependent infrastructure systems.