Received: August 10, 2020. Revised: September 3, 2020.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Self-adaptive Hybrid Bio-inspired Optimization Algorithm by Using Sigmoidal Function

Prasitchai Boonserm[1]* **Suchada Sitjongsataporn[2]**

[1]*The Electrical Engineering Graduate Program, Faculty of Engineering, Mahanakorn University of Technology, Bangkok, Thailand*
[2]*Department of Electronic Engineering, Mahanakorn Institute of Innovation, Faculty of Engineering, Mahanakorn University of Technology, Bangkok, Thailand*
* Corresponding author's Email: prasitchai.boonserm@gmail.com

Abstract: The article presents a new hybrid algorithm, which designs based on traditional bio-inspired optimization algorithms. The algorithm leverages the advantage of Particle Swarm Optimization (PSO), Differential Evolution (DE), and Artificial Bee Colony (ABC), replacing other algorithm weaknesses. A new algorithm we proposed is the Fast bio-inspired Optimization Algorithm (FOA). The DE uses multi-parent for trial vector calculation. It increases the diversity of the solution, while the sigmoidal function adds a self-adaptive characteristic to the proposed algorithm. The function replaces a weighting scheme of PSO. In sub-optimal avoidance, the FOA includes a scout bee behavior from ABC. It makes FOA providing the solution faster than traditional versions, while the solution quality is maintained at an acceptable level. According to a new design, an FOA can reduce the algorithm runtime up to 43.57%, 37.14%, 40.78%, and 31.30% compared to PSO, DE, ABC, and DEPSO, respectively. The DEPSO is the hybrid algorithm between DE and PSO. The best solution to FOA is better than the traditional version of the algorithms. The new algorithm design and the optimization speed improvement are the highlight contribution of this article.

Keywords: Fast bio-inspired optimization algorithm (FOA), Differential evolution particle swarm optimization (DEPSO), Sigmoidal function, Optimization algorithm, Bio-inspired algorithm.

1. Introduction

The data is a key element driving the economy through the application of algorithms. Algorithms play a role in extracting the data supporting the business needs. The optimization algorithm plays its part in the decision support system providing the optimal solution to support the business decision and technical resolution. However, the complexity of the problem is the factor influencing the algorithm run for a long time. Based on limited computational resources, the optimization algorithms in a decision-support system software suite cannot provide a solution to the business on time. Therefore, the development of an optimization algorithm, which can provide an acceptable quality of solution and well adaptive to the time constraints, is the highlight contribution of this research. The bio-inspired optimization algorithms had been using for problem-solving in many types of applications, that is why the development for a better algorithm performance is undoubtedly essential.

The applications of the conventional bio-inspired optimization algorithms and its hybrid versions can be found mostly in the financial business and telecommunication research. The famous algorithms used are the Genetic Algorithm (GA), Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). In the financial business, GA is used for searching the stock trading rules in the market, aiming to understand the optimal rules for profit in trading [1]. The GA identifies a subset of technical trading rules that related to the buying and selling signal of stock investment. The profit in trading is the fitness value calculating by Sharpe Ratio [2], which is well-known performance measurement in financial. The ratio is the average return earned over the risk-free rate per unit of total risk. The GA provides the solution with a higher profit in investment comparing to common trading strategies like Double CrossOver and Bay-and-Hold strategy. The ABC is an alternative optimization algorithm used for decision support. ABC is used for real estate portfolio optimization based on the risk preference coefficient [3]. The selection of a real estate portfolio for investment comes from the solution space, known as the food sources of the bee. The solution with a better value of risk preference coefficient has a higher chance of winning. The research team also compared the algorithm performance between ABC and GA. The ABC outperforms GA on both convergence speed and quality of the solution. Another example of financial application, M. Marinaki [4-5], proposed the application of the GA and ACO for the credit risk assessment problem. The algorithms classify the financial institutes into different levels according to their credit risk. The GA and ACO search for a potential subset of the features on the solution space. The integration of the nearest neighbor with those algorithms used for risk assessment and classification. On the ACO, S. Kumar [6] introduced a new approach to finding the pricing options on buying and selling the contract to get maximum benefit. The ant trail in ACO represents a set of pricing options. The optimal solution is the ant trail that has the highest pheromone concentration. The solution produced by ACO is better than Monte Carlo, the traditional technique. Also, the computational time of ACO is better than Monte Carlo. In the financial domain, the time-series problem is another challenging problem due to the nature of the problem is dynamic. The PSO is combined with KMV to work as the prediction model for the risk of blockchain in China financial market. The KMV is a credit risk model, which is known as the structural approach of pricing credit risk. The hybrid work of PSO with KMV can better identify the credit risk of listed companies in China's blockchain financial market than traditional KMV [7]. Moreover, the hybrid algorithm between ACO and Support Vector Regression (SVR) is researched and overcomes the challenge of time-series problems. The hybrid algorithm proposed by W. C. Hong [8] providing excellent solution quality on the exchange rate forecasting as well as the modified Particle Swarm Optimization (PSO), which proposed by W. H. Zhong [9]. The parameters influencing the exchange rate is selected and encoded into the ant trail for ACO and the position of a particle in PSO. An acceptable quality of the solution is the evidence showing the performance of ACO and PSO.

In telecommunication research, the ACO is the famous bio-inspired optimization algorithm using for solving the network problems. K. M. Sim and W. H. Sun [10] presented their idea to use ACO searching for the optimal network route. The potential routes were encoded into the ant trails. The trail with the highest pheromone is selected to be the best network route. On another routing problem, D. Zhao [11] proposed a new strategy on ACO to improve the way to increase the pheromone on the trails, called ant-weight strategy. The proposed ACO delivers a network route with a higher packet delivery ratio and lower communication costs. Not only the ACO but GA, PSO, and ABC have been used in network problem resolution. J. Lee [12] used GA to resolve the network link partitioning. A chromosome represents a set of partitions of the communication link. GA applies the genetic operations for solution searching to get a chromosome with the best fitness value, which is eventually the final solution of the link partitioning. In the case of the waveform design for wireless power transfer, DE [13] is the algorithm used in the waveform design optimization problem. The DE delivers the optimal solution outperforms Sequential Convex Programming (SCP), which is a well-known algorithm for waveform design. Moreover, Z. Dongming [14] presented a modified version of PSO by applying the quantum behavior into traditional PSO. It is a technique to generate diversity in a solution space. The algorithm is used for searching the optimal route between a source and a destination node in the mobile network. This modified version of PSO delivers higher accuracy and faster convergence speed than traditional GA and PSO. In space-time code selection [15], PSO is the optimizer for space-time code for multi-inputs single-output system. The PSO delivers the minimum bit-error-rate for a given throughput objective and delivers maximum throughput for a given bit-error rate. On ABC, C. Ozturk [16] used ABC to find the best way to deploy the mobile sensor network to get the maximum coverage network area. The result shows that ABC outperforms the PSO for the quality of the solution.

Based on the application reviews, the bio-inspired algorithms had been using for decades. The majority use-cases of ACO are on combinatorial optimization problems resolution, for example, the problems on graphs and discrete structures like telecommunication network routing and constrains based route selection. The publications of ACO give an idea of modifying the algorithm to get a better solution. While the GA, PSO, and ABC are outstanding on the resolution to numerical optimization problems, for instance, the problems in

Received: August 10, 2020. Revised: September 3, 2020.

a continuous domain, searching for the optimal rules for profit in trading, the selection of a real estate portfolio based on the risk preference coefficient, and some numerical problems. The hybrid of the algorithm generates the diversity of solutions producing a better solution quality and computational time comparing with the conventional version.

Therefore, to improve the performance of an optimization algorithm, the development of a new hybrid optimization algorithm, which can provide an acceptable quality of solution and well adaptive to the time constraints, is invented to support the processing of a large amount of data in business reality. In other words, the time-consuming is an obstacle causing by the limitation of computational resources. The small size of the business faces difficulty in reaching an excellent performance of the optimization application. On the budget confinement, algorithm improvement is worth and beneficial to the business. Therefore, this article proposes a self-adaptive hybrid bio-inspired optimization algorithm by using a sigmoidal function, called Fast bio-inspired Optimization Algorithm (FOA), which can faster deliver the solution. In contrast, the quality of the solution is comparable with off-the-shelf algorithms.

For article organization, the article consists of six sections. Section 2 presents the background study and literature review, which describe the knowledge and characteristic of the conventional algorithms. Section 3 presents an analysis of the algorithm. The strengths and weaknesses of each conventional technique are presented and discussed. This section shows the origin of the FOA, which combines the advantage of conventional algorithms replacing the weakness of each other. Section 4 shows the proposed algorithm in detail. The description of the FOA, including the new features of the sigmoidal function for self-adaptive weight rebalancing to PSO, is described in this section. Section 5 discusses the algorithm performance and comparison. The performance of the proposed algorithms compared against the conventional algorithms and DEPSO, the hybrid algorithm of DE and PSO. Section 6 is the conclusion, which is the summary of the performance of the proposed algorithms. This section includes future research opportunities for further algorithm enhancement and development.

2. Background study and literature review

To design an optimization algorithm concerning the business constraints, the conventional version of algorithms and its hybrid versions are in the scope of the literature study. Reviewing the existing algorithms benefits the researcher and reader to understand the ground of algorithms, including strengths and weaknesses. The review of conventional algorithms and its hybrid version comprises GA, PSO, ACO, ABC, Differential Evolution (DE), which is the variance of GA and DEPSO that is the hybrid version of DE and PSO.

2.1 Genetic algorithm

The genetic algorithm (GA) is a bio-inspired optimization algorithm introduced by J. H. Holland [17]. A chromosome string represents a feasible solution. Meaning, a population of chromosomes substitute for all feasible solutions in the solution space. The best chromosome is the solution to the problem, reproducing from the genetic operations, which are reproduction, crossover, and mutation. The quality of offspring depends on the diversity of parent chromosomes, number of crossover-points, crossover-positions, mutation rate, and parent selection techniques, which are random selection and roulette wheel selection. While the fitness function is the measurement to the quality of the offspring, indicating how good of the solution. The termination of GA depends on either the reproduction cycle criterion or the error threshold or both.

Moreover, there are many variances in GA. Researchers adjust the genetic operations, which is not significantly changed the structure of GA—for example, the self-configuring GA with a modified uniform crossover operator [18]. The probability of applying genetic operators are changed on the fly in runtime, based on the population level, which classifier in the early state. More studies of GA regarding the theory, literature review, and application can be found in the survey paper of S. Mirjalili [19].

2.2 Particle swarm optimization

The particle swarm optimization (PSO) simulates the social behavior of animals—the algorithm presented by J. Kennedy and R. Eberhart [20] illustrating the behavior of the flocks of birds. Each bird called a particle representing a feasible solution in the solution space, while a flock of birds represents a swarm. The particle tries to adjust its position based on the information providing by the swarm and its experience. The information from swarm is the best position among particles, known as “global best, *gbest*.” In contrast, particle experience is the best position so far from an individual particle known as “particle best, *pbest*.” These pieces of information are used in position updates. A summation of the current particle position with a new velocity is a new position of a particle. Additionally, a new velocity is derived from a list of parameters; current velocity, current particle position, particle best, global best, inertia weight, coefficient values, and a random number. From the experiment, the coefficient value influences the speed and quality of the solution. The termination criterion of PSO is the same as GA. The best particle position is the best solution delivered by the PSO. For PSO enhancement, there are many variances, which mostly focuses on the PSO parameter configuration, identifying the number of particles, and topology change [21]. The enhancement applied for a particular application-specific. However, the theory of PSO maintains mostly the same.

2.3 Ant colony optimization

From the review of Ant colony optimization (ACO) [22], the ACO has inspiration from the foraging behavior of the ant, which was introduced by M. Dorigo [23]. An ant trail represents a solution in the solution space. Therefore, the number of ants denotes the number of candidate solutions. The ACO starts by allowing the ants to choose the path exploring for the food. The path selection is based on the pheromone intensity. However, the ants will randomly choose the path for the initial cycle. Otherwise, the path with higher intensity of pheromone has more chances to be selected. There are three operations in a running cycle, which consists of selecting the path, adding the pheromone to the trail, evaporating of pheromone. Once ant found the food, the pheromone will be added to the trail while they walk back to the nest. However, at the same time, the pheromone is evaporating by the pre-defined evaporation rate. The pheromone intensity denotes the fitness value of ACO. Therefore, the trail with the highest pheromone will be the best solution for ACO. The termination of ACO depends on either the algorithm running cycle or the quality of the solution or both.

2.4 Artificial bee colony

The artificial bee colony (ABC) is the algorithm mimicking the foraging behavior of the honeybee—the algorithm proposed by D. Karaboga [24]. In ABC, the position of food source represents a feasible solution in the solution space. The food sources are initialized randomly in the first step of the algorithm. On the optimization, there are three types of bees, which play different roles in foraging. The employed bees fly to the food source and responsible for food source evaluation. The amount of nectar represents the fitness value of the solution. Once the nectar evaluation is completed, the employed bees fly back to its hive for information sharing. The onlooker bees, who observe the shared information, select the food source based on the nectar level. The food source with a higher level of nectar has more chances to be selected than the lower one. When the employed bee finished information sharing, the bee will update the positions of food sources based on the neighborhood information. If a new food source has a higher amount of nectar, the bee will delete the old position of food source in her memory and remember a new one. After the onlooker bee complete the food source selection, they will update the position of food source as well. At the last step of each cycle, the scout bees fly out for an alternative food source searching. The existing food source will be abandoned if the fitness value is not improved for a defined period. The food source with the highest amount of nectar is the best solution for ABC. The termination of ABC depends on the running cycle criterion, the error threshold, the solution improvement limit threshold, or the combination of those criteria.

2.5 Differential evolution

The differential evolution (DE) is the variance of GA, which was presented by R. Storn and K. Price [25]. DE was designed to resolve the vector-based optimization problem. As it is the GA's variance, therefore, the problem encoding is then the same as GA but deviates in terms of reproduction operations. In offspring reproduction, the delta vector is calculated from the difference of selected parent chromosomes. The weighted delta is applied to a third parent, which is the offspring. Hence, the quality of the offspring depends on the delta vector that describes the differences in the solution quality. The crossover and mutation operations are the same as GA. The greedy selection is the parent selection technique for DE. The fitness of offspring and parents are compared, the win will proceed for the next generation. The termination of DE is the same as GA.

2.6 Differential evolution particle swarm optimization

The differential evolution particle swarm optimization (DEPSO) is a hybrid optimization algorithm, which combined the DE and PSO. The operation of DE and PSO are turn-based [26]. The solution is encoded into the form of a vector representing the particle in PSO and chromosome in DE. The starting point of the algorithm depends on the use of DEPSO. Both DE and PSO can be the first algorithm of the hybrid version. In DEPSO [27], which is the latest valid version of DEPSO at the moment. The transition from PSO to DE, each particle is cloned and mutates per the preconfigurable parameter, which is ruled by multiplicative lognormal random numbers. For reproduction, the selection method is a stochastic tournament. The reproduction applies DE operation. The best offspring from the pair of parents is selected and proceed to the PSO cycle. From the literature review, DEPSO is better than DE and PSO in terms of the quality of the solution but computational time [28]. Opposition-based learning (OBL) had been applied to improve the performance of DEPSO [29]. However, the number of parameters that add to the algorithm making DEPSO hard on performance tuning. The vector with the best fitness value is the final solution of DEPSO. The termination of DEPSO is the same as DE and PSO [26].

3. Analysis of algorithms

The analysis of the bio-inspired optimization algorithms is presented in Table 1. It highlights the strengths and weaknesses of a particular conventional algorithm and DEPSO. According to the review, GA is the most popular algorithm used in many business applications due to its flexibility to all types of problems. The high complexity of the problem and the big size of the solution space directly impact the convergence speed of GA. Moreover, the improper setting of the termination criterion can degrade the solution quality. Based on these constraints, a near real-time solution cannot be achieved without fine-tuning by the expertise.

Table 1. The strengths and weaknesses of existing algorithms

Algorithms	**Strengths**	**Weaknesses**
Genetic Algorithm (GA) [17]	• Mutation operation introduces a chance to allow algorithms spring out from the local minima based on the current offspring. • The roulette wheel is the selection techniques providing the probabilistic selection to parent chromosome representing the character of natural selection.	• Limited solution diversity as the offsprings are bounded by the parent chromosomes when applying the crossover operation. • Take a long time in convergence. • Improper setting of the termination criterion can degrade the solution quality.
Differential Evolution (DE) [25]	• Suitable for solution exploitation. • Reserve property of the population diversity. • Quick in convergence.	• It is unstable in convergence.
Particle Swarm Optimization (PSO) [20]	• The algorithm has the solution memory features *pbest* and *gbest*, which is useful for the reproduction of the next generation of particles. • Quick in convergence.	• The solution diversity impacts convergence. • Untimely convergence. • Easily trap at local minima. • The algorithm needs expertise in parameter tuning. Misconfiguring PSO parameters leads to poor quality of the solution.
Ant Colony Optimization (ACO) [23]	• Good at optimization on the graph-based problems. • The efficient technique for solution update relies on the pheromone release rate, evaporation rate, and pheromone intensity of the trails or solution.	• Observed the conversion overhead for numerical problem encoding.
Artificial Bee Colony (ABC) [24]	• Scout bee helps the algorithm escape from local minima by randomizing a new food source. • The bee dance is the probabilistic selection for a candidate solution.	• Take a long time in convergence • Observed the event of trapping at local minima.
Differential Evolution Particle Swarm Optimization (DEPSO) [27]	• Improved quality of solution compared to conventional algorithms	• Parameter configuration needs expertise to advise.

Received: August 10, 2020. Revised: September 3, 2020.

DE denoted as the variance of GA, which is good at solution exploitation and fast solution delivery. The design of DE overcomes the weakness of GA in terms of speed of convergence. The delta vectors make DE long-jump in solution space, accelerating the speed of solution exploitation at the early stage. However, the quality of the solution delivering by DE fluctuates due to the side effect of the long jump. In order to improve the algorithm for fast optimization, DE has more potentiality of improvement than GA.

In the PSO, the balance of applying *pbest* and *gbest* influences solution quality. The different type of problem needs expertise in parameter tuning or a proper pre-defined parameter setting. Misconfiguring PSO parameters leads to poor quality of solution due to local optimal and slow in convergence. However, the design of PSO has supported both exploration and exploitation but not sub-optimal avoidance. The highlight of PSO is the parameter configuration that can balance exploration and exploitation. The coefficient value weights the delta vectors calculating from *pbest* and *gbest*, influencing the exploration behavior of the algorithm—fixed coefficient values direct-variation to the speed of convergence. Therefore, A proper parameter setting to PSO will improve both quality of the solution and the speed of convergence.

ACO is an optimization algorithm that is suitable for combinatorial problem-solving, for example, network route finding, constraint-based communication design. Refer to the characteristic of ACO, the problem in graph form is easily encoded into ant trails, which represent the feasible solution of ACO. The numerical problem is rarely found on ACO due to the difficulty of solution encoding and transformation. In brief, the ACO is suitable for a specific type of application, especially on graph-related problems. It generates a high cost of solution transformation for the problems in the numerical domain.

On ABC, the performance of the algorithm is based on the initial factors of the feasible solution. The ABC delivers a stable quality of the solution compared to the same running cycle of itself. A random bit of solution is twisted in the solution update process leading to slow convergence and also possibly trapping at local optimal; however, it introduces the fine-tuning of solution quality. Nevertheless, ABC has an outstanding feature called the scout bee, which searches for an alternative solution when the current solution is not updated for some time. The scout bee makes ABC avoiding the sub-optimal, which increases the potentiality for the optimal solution. On the algorithm comparison, the ABC outperforms GA on both convergence speed and quality of the solution [3].

DEPSO is a hybrid creation of a bio-inspired algorithm. The solution exploitation from DE enhances PSO by improving the particle position with the DE delta vector. By applying the vector, the convergence speed is improved dramatically. Moreover, the DE delta vector also introduces particle diversity leading to a variety of solutions. The elite chromosomes turn to particle members replacing the particles with low quality. It yields up the mean of particle quality for the entire swarm [27]. In PSO turn, the balancing of exploitation and exploration is maintained aligning with the parameter setup. PSO applies for solution fine-tuning, delivering a better solution in terms of precision. After the PSO cycle finished, DE takes its turn to convert the group of particles to a population of chromosomes and apply DE operations. The algorithm runs as a cooperative co-evolutionary, which supports each other to improve the solution delivery. There is evidence showing the DEPSO is better than DE and PSO in terms of the quality of the solution [28]. However, the parameter setup for initializing the algorithm needs the expert's advice due to a combination of algorithms. The number of configuration parameters is the summation of DE and PSO.

The study of algorithms helps the researcher and reader to understand the ground of algorithms, strengths, and weaknesses. In summary, DE is faster than GA in terms of the convergence speed, while the DEPSO can provide delicate quality of solution than DE and PSO. About sub-optimal avoidance, ABC uses a low-cost technique comparing to other algorithms in the same class. At the same time, the ACO is good at resolving the combinatorial domain. To develop the Fast bio-inspired Optimization Algorithm (FOA), this work leverage a core process of DE and PSO algorithm by uses the strengths of DE, PSO, and ABC to improve the weakness of each other. The consideration of the speed of convergence and solution quality are the concerns in algorithm development. The DE and PSO are selected to be the main algorithm. Scout bee feature of ABC is added into the design for suboptimal avoidance. The sigmoidal function is introduced to generate self-adaptive for rebalancing the exploration and exploitation behavior of PSO. The sigmoidal functions also use for solution refinement and automate parameter configuration without expertise. The detail of the proposed algorithm is presented in section 4.

Received: August 10, 2020. Revised: September 3, 2020.

Algorithm 1: The Fast bio-inspired Optimization Algorithm (FOA)

```
   Data: ${j, k, n, t, x, population \in \mathbb{Z}}$, ${i : i = 1 \dots |population|}$, ${random : random = 1 \dots |population|}$
   Result: $gbest\_vector$
1  $vector\_format \leftarrow$ encode($problem$)
2  $vector_i \leftarrow$ initial_solution($vector\_format, constraints, boundary$), $\forall vector \in population, turn \leftarrow PSO$
3  while termination criteria is not satisfied do
4      foreach i do
5          $vector_i.fitness \leftarrow$ fitness_function($vector_i$)
6          $vector_i.pbest\_position \leftarrow vector_i.fitness > vector_i.pbest\_fitness \,?\, vector_i.position : vector_i.pbest\_position$
7          $gbest\_vector \leftarrow vector_i.fitness > gbest\_vector.fitness \,?\, vector_i : gbest\_vector$
8      end
9      if gbest_vector is not change for sometime then
10         $scout\_bee\_vector \leftarrow$ initial_solution($vector\_format, constraints, boundary$)
11         $scout\_bee\_vector.fitness \leftarrow$ fitness_function($scout\_bee\_vector$)
12         $gbest\_vector \leftarrow scout\_bee\_vector.fitness > gbest\_vector.fitness \,?\, scout\_bee\_vector : gbest\_vector$
13         $vector_{random} \leftarrow scout\_bee\_vector$
14         reset($t$)
15     end
16     if termination criteria is satisfied then
17         return($gbest\_vector$)
18     else
19         if turn is PSO then
20             foreach i do
21                 $vector_i.velocity \leftarrow$ cal_velocity($vector_i.velocity, gbest\_vector.position, vector_i.pbest\_position, t$)
22                 $vector_i.position \leftarrow$ update_position($vector_i.position, vector_i.velocity$)
23             end
24             $turn \leftarrow DE$
25         else
26             foreach i do
27                 for $j \leftarrow 1 : 2x$ do
28                     $parent\_vector_j \leftarrow$ selection_function($population$)
29                 end
30                 for $k \leftarrow 1 : x$ do
31                     $delta\_vector_k \leftarrow parent\_vector_{2k-1}.position - parent\_vector_{2k}.position$
32                 end
33                 $trial\_vector \leftarrow$ average($delta\_vector$)
34                 $offspring\_vector.position \leftarrow vector_i.position + trial\_vector$
35                 $offspring\_vector.fitness \leftarrow$ fitness_function($offspring\_vector$)
36                 $vector_i \leftarrow offspring\_vector.fitness > vector_i.fitness \,?\, offspring\_vector : vector_i$
37             end
38             $turn \leftarrow PSO$
39         end
40     end
41 end
42 return($gbest\_vector$)
```

Figure. 1 The fast bio-inspired optimization (FOA) algorithm

4. Proposed algorithm

In the design of the Fast bio-inspired Optimization Algorithm (FOA), Fig. 1, we modify the parameter configuration in different ways comparing to the traditional algorithm. The sigmoidal function is added to PSO to tuning the exploitation and exploration automatically, and the number of parent chromosomes for the trial vector, which is used for the delta vector calculation, is fixed to eight chromosomes following the algorithm design. The scout bee process of ABC is introduced for sub-optimal avoidance. This section describes the algorithm in detail by line. Therefore, the statement "At line x" refers to the explanation of the algorithm at line x and the statement "At line x – y" describes the meaning of the algorithm as a whole block starting from line x to line y.

4.1 Algorithm explanation

From Fig. 1, the proposed algorithm starts with the problem encoding at line 1. The variable, namely *vector_format*, represents a vector structure of the feasible solution. The encoding technique of the proposed algorithm is similar to other conventional algorithms. So, there is no specific encoding discussion for this algorithm. Once the vector structure is set up, the algorithm starts initializing the feasible solutions in the form of *vector_format*, considering the constraints and boundaries of the solution space. All initial vectors are the member of the group, namely *population*. The number of initial vectors depends on the parameter setup. The first turn of the algorithm is set to PSO at line 2. Once the initial vectors are completely created, the algorithm can be stopped if the termination criteria are satisfied, and the *gbest_vector* representing the best solution will return the solution for decoding at line 3. The termination criteria can be the maximum number of cycles, acceptable error distance, or both. In cases of the criteria are not satisfied, the algorithm continues searching for an optimal solution. At line 4 – 8, The fitness value of all vectors is calculated by a fitness function. For particular *vector_i*, the particle best vector, known as $vector_i$. *pbest_position*, is determined by comparing the current fitness value with the fitness value of existing $vector_i$. *pbest_fitness*. The algorithm will replace the $vector_i$. *pbest_position* with a current position if a new fitness value is higher than the fitness value of the particle best vector, $vector_i$. *pbest_fitness*. The vector with the best fitness value among the population will be recorded as the global best vector, *gbest_vector*. At line 16, if the termination criterion is an error distance or the *gbest_vector* from the first generation satisfies the termination criterion, the algorithm will stop and return *gbest_vector* for solution decoding. Otherwise, the PSO will take a turn in optimization. At line 19 – 24, on every cycle, *t*, the algorithm updates the fitness value of a particular $vector_i$ based on its new position by using functions, cal_velocity () and update_position (). The function cal_velocity () returns a new velocity of particular $vector_i$, which is calculated by Eq. (1). The function update_position () uses the output of function cal_velocity () to update the position of the vector following Eq. (5).

Once all vectors already updated, the next algorithm cycle is set to DE and go to line 3 – 15. At line 25 – 38, the algorithm runs the modified DE optimization. The mutation process is modified to reduce the overall runtime of the algorithm. The number of delta vector, *x*, uses for calculating the trial vector. A pair of parents are selected by function selection_function () to generate a delta vector. In the meantime, an average of delta vectors represents a trial vector for a particular $vector_i$. The number of delta vector influence the improvement of the convergence direction of the offspring. The addition operation reproduces the offspring, *offspring_vector*, by adding the trial vector, *trial_vector* to $vector_i$. Once the offspring are reproduced, the fitness value of the offspring is calculated by function fitness_function (). The offspring replaces $vector_i$, if and only if the fitness value of the offspring is better than $vector_i$. Once all vectors already updated, the next algorithm cycle is set back to PSO and go to line 3 – 15. At line 9 – 15, if the best solution is not updated for sometime coupled with the termination criteria are not satisfied, the algorithm will create a new vector, namely *scout_bee_vector*, in the form of *vector_format* under the constraints and boundaries of the solution space, to replace a vector which randomly selects from the population. The running cycle, *t*, is reset. If the fitness value of a new vector is better than the fitness value of the best solution, *scount_bee_vector* will replace the global best vector, *gbest_vector*. The algorithm runs until the termination criteria are satisfied.

4.2 Mathematical formulas explanation

In the FOA, five mathematical formulas; Eq. (1) to Eq. (5) are used in the algorithm explanation. The Eq. (1) is the function cal_velocity () at line 21. $vector_i$. *velocity (t+1)* denotes a new velocity of a particular $vector_i$ where the inertia weight, *w*, is the constant for $vector_i$. *velocity(t)*. It is the weight to preserve the current velocity of $vector_i$. The $vector_i$. *velocity (t)*, and $vector_i$. *position(t)* denote the velocity and position of the $vector_i$ at time *t*, respectively. $vector_i$ *pbest_position(t)* is the position of the particle best vector of the $vector_i$ that has learned so far. *gbest_position* denotes the global best vector at time *t*. The random number, $r_1(t)$ and $r_2(t)$, are the uniform distribution, $r_1, r_2 \sim U$ (0,1). The present of r_1, r_2, is to preserve the natural behavior of swarm.

$c_1(t)$ denotes the constant used in weighting the vector direction, which is the difference between $vector_i$ and particle best vector at time *t* while $c_2(t)$ is the constant used in weighting the vector direction, which is the difference between $vector_i$ and global best vector at time *t*.

In the proposed algorithm, $c_1(t)$ and $c_2(t)$ constant are replaced by sigmoidal functions shown in Eq. (2) and Eq. (3), respectively. The calculation of Eq. (2) and Eq. (3) use the value *u* from Eq. (4).

$$vector_i.\ velocity(t+1) = w \times vector_i.velocity(t) + c_1(t)r_1(t)[vector_i.pbest\_position(t) - vector_i.\ position(t)] + c_2(t)r_2(t)[gbest\_vector(t) - vector_i.\ position(t)], \quad (1)$$

$$c_1(t) = \frac{w_{start} - w_{end}}{1 + e^{u(t-sm)}} + w_{end}, \quad (2)$$

$$c_2(t) = \frac{w_{start} - w_{end}}{1 + e^{-u(t-sm)}} + w_{end}, \quad (3)$$

$$u = 10^{\log(m)-2}, \quad (4)$$

$$vector_i.\ position\ (t+1) = vector_i.pbest\_position(t) + vector_i.\ velocity(t+1). \quad (5)$$

At the running cycle *t*, $c_1(t)$ denotes the decreasing sigmoidal function, which is Eq. (2). $c_2(t)$ denotes the increasing sigmoidal function, which is Eq. (3). The value of both functions is in the range of defined weight, between w_{start} and w_{end}. The w_{start} and w_{end} can be any value depending on the application. The default value of w_{start} is 1, and the default value of w_{end} is zero.

The parameter *s* denotes the sigmoidal constant, which is in the range between 0.25 and 0.75, while the parameter *m* denotes the expected maximum number of cycles in running the algorithm.

In every cycle *t*, the value of $c_1(t)$ and $c_2(t)$ changes automatically. The value of $c_1(t)$ will start from the upper bound, decreasing to the lower bound and vice versa for $c_2(t)$. In other words, the algorithm in the PSO starts the optimization relies on the local information and finishes with the global information.

The sigmoidal functions provide more accuracy of solution convergence to the global optimal than a traditional PSO. Once the new velocity of $vector_i$ is calculated, the $vector_i.\ position$ will be updated by Eq. (5).

For Eq. (5), the $vector_i.\ position(t+1)$ denotes a new position of a particular $vector_i$ where $vector_i.\ pbest\_position(t)$ and $vector_i.\ velocity(t+1)$ are the same definition as described in Eq. (1).

5. Algorithm performance comparison

To validate the Fast bio-inspired Optimization Algorithm (FOA), we run the algorithm solving numerical optimization problems. Various benchmark functions consisting of Sphere, Griewank, Rastrigin, Rosenbrock, and Ackley. The minimum value of all benchmark functions is zero. The performances of FOA are compared with the Particle Swarm Optimization (PSO), Differential Evolution (DE), Artificial Bee Colony (ABC), and the hybrid algorithm, Differential Evolution Particle Swarm Optimization (DEPSO). The experimental result of FOA was compared by using the configuration similar to the experiments, as presented by J. Kennedy and R. Eberhart [20] for PSO, R. Storn and K. Price [25] for DE, D. Karaboga [24] for ABC and V. Miranda and R. Alves [27] for DEPSO. The number of initial solution in the experiment was set to 100. The number of solutions in our algorithm was equivalent to that setup in all experiments. The number of running cycles bounded with the maximum cycle number. It is to avoid the overrun of algorithms. The maximum cycle number used in FOA is equal to that applied in PSO, DE, ABC, and DEPSO. The 100 experiments were performed for each benchmark function. For the parameter configuration of FOA, the wstart is 1, and the wend is zero. Sigmoidal constant s is 0.50.

Table 2 presents the results of the experiments in numerical optimization problems. The number of maximum cycles is 1000 cycles, and the solution dimensions for each benchmark function is 30 dimensions. The column, namely "Best," "Worst," "Mean," and "Std," represents the best, the worst, the average, and the standard deviation of the objective values produced by particular benchmark functions. The lower objective value represents a better solution. The "Best Cycle" and "Worst Cycle" column shows the reached cycle that particular algorithms provide the best and the worst objective values, respectively. The "Success Rate" column displays the percentage of experiment round that the experiment is reached. 100% of success means the algorithm is reached the termination criteria in all experiments of a particular benchmark function. The experiment with the value of solution greater or equal to 1.00 is considered as an unsuccessful experiment.

The results in Table 2, in boldface, shows the outstanding value comparing between algorithms. The best objective values of FOA are better than the best objective values of other algorithms in three benchmark functions, which are the Sphere function, the Griewank function, and the Ackley function. A poor objective value on Rastrigin and Rosenbrock functions can be observed on FOA due to the Rastrigin function produces many local minima while the global minimum of the Rosenbrock is in the narrow valley. The proposed algorithm might get stuck in the local minima and discover only the valley.

Again, on the function of Sphere, Griewank, and Ackley, the worst objective values, as well as the average objective value that produces by the FOA is much smaller than the best objective value delivering by PSO, DE, ABC, and DEPSO. In other words, the

Table 2 The comparison of algorithm performance

(a) The objective values of the proposed algorithm – fast bio-inspired optimization algorithm (FOA),

Benchmark Functions	Best Cycle	Worst Cycle	Best	Worst	Mean	Std	%success
Sphere	**785**	**198**	**$5.21x10^{-32}$**	**$6.48x10^{-12}$**	**$4.32x10^{-14}$**	$2.84x10^{-12}$	72%
Griewank	**542**	**127**	**$2.42x10^{-24}$**	**$1.87x10^{-12}$**	**$8.53x10^{-15}$**	$8.56x10^{-13}$	**96%**
Rastrigin	**126**	**36**	$3.11x10^{-2}$	$3.98x10^{-2}$	$3.91x10^{-2}$	$3.75x10^{-3}$	**100%**
Rosenbrock	**215**	**185**	$1.26x10^{-2}$	$2.99x10^{-1}$	$1.87x10^{-1}$	$1.09x10^{-1}$	**100%**
Ackley	**512**	**122**	**$2.61x10^{-14}$**	**$8.90x10^{-9}$**	**$8.95x10^{-11}$**	$4.18x10^{-9}$	**100%**

(b) The objective values of the particle swarm optimization (PSO)

Benchmark Functions	Best Cycle	Worst Cycle	Best	Worst	Mean	Std	%success
Sphere	1000	763	$4.32x10^{-9}$	$3.42x10^{-7}$	$8.27x10^{-8}$	$1.54x10^{-7}$	61%
Griewank	692	492	$3.53x10^{-10}$	$8.27x10^{-7}$	$7.43x10^{-9}$	$3.74x10^{-7}$	81%
Rastrigin	629	526	$9.48x10^{-10}$	$2.02x10^{-7}$	$6.28x10^{-9}$	$8.91x10^{-8}$	**100%**
Rosenbrock	542	443	$2.92x10^{-3}$	**$6.09x10^{-2}$**	**$6.75x10^{-3}$**	$2.41x10^{-2}$	95%
Ackley	1000	629	$3.25x10^{-6}$	$2.37x10^{-5}$	$6.62x10^{-6}$	$8.45x10^{-6}$	**100%**

(c) The objective values of the differential evolution (DE)

Benchmark Functions	Best Cycle	Worst Cycle	Best	Worst	Mean	Std	%success
Sphere	1000	729	$1.09x10^{-8}$	$8.73x10^{-8}$	$5.63x10^{-8}$	$3.14x10^{-8}$	**77%**
Griewank	599	450	$7.82x10^{-10}$	$9.82x10^{-8}$	$4.59x10^{-9}$	$4.54x10^{-8}$	74%
Rastrigin	698	128	$5.82x10^{-4}$	$9.83x10^{-4}$	$9.03x10^{-4}$	$1.63x10^{-4}$	**100%**
Rosenbrock	349	254	$9.02x10^{-2}$	$2.41x10^{-1}$	$1.36x10^{-1}$	$5.80x10^{-2}$	97%
Ackley	822	402	$1.54x10^{-6}$	$9.84x10^{-5}$	$4.69x10^{-6}$	$4.29x10^{-5}$	**100%**

(d) The objective values of the artificial bee colony (ABC)

Benchmark Functions	Best Cycle	Worst Cycle	Best	Worst	Mean	Std	%success
Sphere	972	825	$1.75x10^{-11}$	$1.32x10^{-9}$	$3.77x10^{-10}$	$4.12x10^{-10}$	73%
Griewank	620	578	$6.27x10^{-11}$	$5.66x10^{-8}$	$4.22x10^{-9}$	$2.87x10^{-8}$	82%
Rastrigin	657	423	**$2.41x10^{-11}$**	**$1.43x10^{-7}$**	**$4.07x10^{-9}$**	$6.31x10^{-8}$	**100%**
Rosenbrock	432	230	**$7.93x10^{-4}$**	$7.11x10^{-2}$	**$6.75x10^{-3}$**	$3.02x10^{-2}$	**100%**
Ackley	1000	527	$2.17x10^{-6}$	$1.05x10^{-5}$	$5.36x10^{-6}$	$3.89x10^{-6}$	**100%**

(e) The objective values of the differential evolution particle swarm optimization (DEPSO)

Benchmark Functions	Best Cycle	Worst Cycle	Best	Worst	Mean	Std	%success
Sphere	895	449	$6.84x10^{-24}$	$1.39x10^{-12}$	$2.96x10^{-13}$	$5.63x10^{-13}$	71%
Griewank	625	297	$8.12x10^{-14}$	$2.30x10^{-12}$	$3.60x10^{-13}$	$9.32x10^{-13}$	88%
Rastrigin	452	238	$3.44x10^{-2}$	$3.97x10^{-2}$	$3.45x10^{-2}$	$2.38x10^{-3}$	**100%**
Rosenbrock	514	283	$3.43x10^{-2}$	$2.23x10^{-1}$	$5.69x10^{-2}$	$7.95x10^{-2}$	98%
Ackley	687	311	$6.47x10^{-10}$	$3.10x10^{-9}$	$8.78x10^{-10}$	$1.05x10^{-9}$	**100%**

solution of the FOA in the worst-case scenario is better than the best solution to other algorithms.

In the case of the success rate, the FOA can deliver an optimal solution with 100% on Rastrigin function, Rosenbrock function, and Ackley function. For the Griewank function and Ackley function, the FOA outperforms other algorithms on the best, the worst, and the average of objective values while the success rate of FOA is higher or equal to other algorithms. The experimental results show that the success rate of the FOA is better than other algorithms except those running on the Sphere function. The ABC is the algorithm that fits the problem like Rastrigin function and Rosenbrock

Received: August 10, 2020. Revised: September 3, 2020.

Table 3 The comparison of algorithm performance with fixed maximum cycle number

(a) The objective values of the FOA

Benchmark Functions	Best	Worst	Mean	Std
Sphere	**1.04x10^{-50}**	**3.28x10^{-31}**	**3.27x10^{-42}**	5.41x10^{-31}
Griewank	**1.94x10^{-58}**	**2.41x10^{-14}**	**3.75x10^{-16}**	6.84x10^{-14}
Rastrigin	**3.83x10^{-27}**	1.09x10^{-2}	4.11x10^{-9}	2.54x10^{-3}
Rosenbrock	**4.93x10^{-4}**	1.50x10^{-2}	1.29x10^{-1}	6.20x10^{-2}
Ackley	**1.07x10^{-14}**	**8.17x10^{-9}**	**4.39x10^{-11}**	8.45x10^{-9}

(b) The objective values of the PSO

Best	Worst	Mean	Std
4.32x10^{-9}	1.03x10^{-7}	2.72x10^{-8}	1.64x10^{-8}
5.42x10^{-11}	2.02x10^{-7}	8.47x10^{-9}	4.92x10^{-8}
6.44x10^{-12}	1.90x10^{-9}	3.24x10^{-10}	6.81x10^{-10}
1.98x10^{-3}	5.91x10^{-2}	2.29x10^{-2}	2.14x10^{-2}
3.45x10^{-8}	6.73x10^{-7}	9.83x10^{-7}	2.54x10^{-7}

(c) The objective values of the DE

Benchmark Functions	Best	Worst	Mean	Std
Sphere	1.09x10^{-8}	4.00x10^{-8}	2.99x10^{-8}	8.12x10^{-9}
Griewank	5.24x10^{-10}	4.72x10^{-9}	3.22x10^{-9}	1.41x10^{-9}
Rastrigin	3.22x10^{-9}	4.63x10^{-9}	3.65x10^{-9}	5.42x10^{-10}
Rosenbrock	4.63x10^{-3}	**1.44x10^{-2}**	**7.09x10^{-3}**	2.85x10^{-3}
Ackley	2.64x10^{-9}	1.93x10^{-7}	4.69x10^{-8}	5.55x10^{-8}

(d) The objective values of the ABC

Best	Worst	Mean	Std
2.82x10^{-13}	1.54x10^{-11}	6.85x10^{-12}	5.14x10^{-12}
7.39x10^{-12}	8.17x10^{-9}	2.17x10^{-10}	3.69x10^{-9}
2.92x10^{-14}	**7.17x10^{-10}**	**2.81x10^{-11}**	2.45x10^{-10}
7.18x10^{-4}	6.09x10^{-2}	8.47x10^{-3}	2.80x10^{-2}
2.92x10^{-7}	4.93x10^{-6}	6.28x10^{-6}	2.52x10^{-6}

(e) The objective values of the Traditional DEPSO

Benchmark Functions	Best	Worst	Mean	Std
Sphere	5.79x10^{-31}	1.43x10^{-30}	6.01x10^{-31}	3.79x10^{-31}
Griewank	5.62x10^{-14}	1.03x10^{-13}	8.10x10^{-14}	1.78x10^{-14}
Rastrigin	3.83x10^{-4}	2.11x10^{-3}	1.44x10^{-3}	6.59x10^{-4}
Rosenbrock	9.28x10^{-2}	1.91x10^{-1}	1.22x10^{-1}	3.86x10^{-2}
Ackley	5.39x10^{-10}	1.32x10^{-8}	3.97x10^{-9}	5.01x10^{-9}

function, while the DE can deliver an optimal solution with the highest success rate on the Sphere function. About the running cycle of the algorithm, the FOA outperforms all algorithms on both “Best Cycle” and the “Worst Cycle.”

For the Sphere function, the FOA can deliver the best solution faster than PSO, DE, ABC, and DEPSO, 21.50%, 21.50%, 19.23%, and 12.29%, respectively. The delivery time of the worst solution for FOA is better than other algorithms for 71.37% on average.

For the Griewank function, the solution delivered by the FOA is faster than PSO, DE, ABC, and DEPSO for 21.68%, 9.52%, 12.58%, and 13.28%, respectively, for the best solution. The number of termination cycle of the worst solution is also smaller than those algorithms up to 72.04% on average. The performance of the FOA regarding the time to solution delivery is maintained.

The FOA is a 100% success rate on delivering the solution on the Rastrigin function with a significant improvement in the speed to best solution delivery comparing to PSO, DE, ABC, and DEPSO, 79.67%, 81.95%, 80.82%, and 72.12%, respectively. Not only the runtime to deliver the best solution but the worst solution as well. The time of the worst solution of the Rastrigin function producing by the FOA is 89.05% on average.

For the Rosenbrock function, the FOA can deliver the best solution faster than PSO, DE, ABC, and DEPSO, 60.33%, 38.40%, 50.23%, and 58.17% respectively. The delivery time of the worst solution for FOA is better than other algorithms for 38.84% on average.

Lastly, on the Ackley function, the FOA also deliver the solution with fewer running cycle comparing to other algorithms. The FOA is faster than PSO, DE, ABC, and DEPSO for 48.80%, 37.71%, 48.80%, and 25.47%, respectively, while the number of running cycles to the worst solution is smaller than other algorithms 73.89% on average.

Conclusively, in the running cycle perspective, the FOA can provide the solution significantly faster than other algorithms on both the best solution and the worst solution. Based on the results shown in Table 2, the best objective values are obtained from the higher number of cycles. We then consider the cycle that the solution is reached.

According to Table 3, the maximum cycle number in the experiment is set to a static value at 1000. For the experiment configuration, the parameter setup is the same as the previous experiment except for the termination criterion. The number of feasible solutions was set to 100. The termination criterion is considered only the maximum cycle number. If the solution is not improved for some time, the algorithm still runs until reaching the defined maximum cycle number. From Table 3, results show that the maximum cycle number impacts the quality of the solution. The FOA delivers a better solution than other algorithms, which are in boldface. The best objective values delivered by the FOA are improved and better than the best objective values from PSO, DE, ABC, and DEPSO in all benchmark functions. However, the worst and the average of the solution on the Rastrigin function and Rosenbrock function are weaker than DE and ABC. The DE can produce a better quality of the average and the worst solution for the Rosenbrock function while the ABC is better for Rastrigin. In conclusion, the FOA with a higher running cycle can deliver better objective values.

In summary, we can conclude from the experiments that the FOA can be used as an alternative optimization algorithm for the numerical optimization problem. The algorithm has the potential to outperform existing bio-inspired optimization algorithms; DE, PSO, ABC, and DEPSO, in terms of time to solution delivery and solution quality. The FOA can save time as it delivers the optimal solution with smaller algorithm runtime comparing to traditional algorithms. In other words, the strength of the FOA is fast in solution delivery, while the solution quality is maintained. Moreover, the FOA can deliver a better quality of solution comparing to other algorithms, at the same maximum cycle number.

6. Conclusion

The article presents a hybrid algorithm based on traditional bio-inspired optimization algorithms, comprising Particle Swarm Optimization (PSO), Differential Evolution (DE), and Artificial Bee Colony (ABC). Our proposed algorithm, namely, the Fast bio-inspired Optimization Algorithm (FOA), which is the combination of the modified DE, PSO, ABC, and sigmoidal function. The sigmoidal functions replace the weighting scheme of PSO. The new weighting of PSO rebalances the solution exploration and exploitation, while the modified DE increases the solution diversity. Additionally, the feature of sub-optimal avoidance was added by introducing the scout bee behavior of ABC. The FOA can reduce the algorithm runtime up to 43.57%, 37.14%, 40.78%, and 31.30% compared to PSO, DE, ABC, and DEPSO, respectively.

The essential contribution of FOA is the speed of solution resolution comparing to DE, PSO, ABC, and DEPSO algorithms. In contrast, the solution quality is maintained at an acceptable level.

Future work and the improvement opportunities to FOA are the experiments on the combinatorial optimization problems and a real implementation of business applications in a big data domain.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

The author's contributions are provided as follows: Conceptualization, methodology, software, formal analysis, investigation, resources, data curation, writing—original draft preparation, visualization, Prasitchai Boonserm;

Validation, writing—review and editing, Prasitchai Boonserm and Suchada Sitjongsataporn;

Supervision, Suchada Sitjongsataporn.

References

[1] F. A. Badawy, H. Y. Abdelazim, and M. G. Darwish, "Genetic Algorithms for Predicting the Egyptian Stock Market", In: *Proc. of 2005 International Conf. On Information and Communication Technology*, Cairo, Egypt, pp. 109-122, 2005.

[2] W. F. Sharpe, "The Sharpe Ratio", *Journal of Portfolio Management*, Vol. 21, No.1, pp. 49-58, 1994.

[3] L. Hong-mei, W. Zhuo-fu, and L. Hui-min, "Artificial bee colony algorithm for real estate portfolio optimization based on risk preference coefficient", In: *Proc. of 2010 International Conf. On Management Science and Engineering*, Melbourne, Australia, pp. 1682-1687, 2010.

[4] M. Marinaki, Y. Marinakis, and C. Zopounidis, "Application of a genetic algorithm for the credit risk assessment problem", *Foundations of Computing and Decision Sciences*, Vol. 32, No. 2, pp. 139-152, 2007.

[5] Y. Marinakis, M. Marinaki, and C. Zopounidis, "Application of Ant Colony Optimization to Credit Risk Assessment", *New Mathematics and Natural Computation*, Vol. 4, No. 1, pp. 107-122, 2008.

[6] S. Kumar, G. Chadha, R. K. Thulasiram, and P. Thulasiraman, "Ant Colony Optimization to price exotic options", In: *Proc. of 2009 IEEE Congress On Evolutionary Computation*, Trondheim, Norway, pp. 2366-2373, 2009.

[7] Y. Song, F. Zhang, and C. Liu, "The risk of block chain financial market based on particle swarm optimization", *Journal of Computational and Applied Mathematics*, Vol. 370, 2020.

[8] W. C. Hong, Y. F. Chen, P. W. Chen, and Y. H. Yeh, "Continuous ant colony optimization algorithms in a support vector regression based financial forecasting model", In: *Proc. of Third International Conf. On Natural Computation*, Haikou, China, pp. 548-552, 2007.

[9] W. H. Zhong, J. ZHang, and W. N. Chen, "A novel discrete particle swarm optimization to solve traveling salesman problem", In: *Proc. of 2007 IEEE Congress On Evolutionary Computation*, Singapore, pp. 3283-3287, 2007.

[10] K. M. Sim and W. H. Sun, "Multiple ant-colony optimization for network routing", In: *Proc. of First International Symposium On Cyber Worlds*, Tokyo, Japan, pp. 277-281, 2002.

[11] D. Zhao, L. Luo, and K. Zhang, "An improved ant colony optimization for the communication network routing problem", *Mathematical and Computer Modelling*, Vol. 52, No. 11–12, pp. 1976-1981, 2010.

[12] J. Lee, Y. Choi, B. Zhang, and C. S. Kim, "Using a genetic algorithm for communication link partitioning", In: *Proc. of 1997 IEEE International Conf. On Evolutionary Computation*, Indianapolis, IN, USA, pp. 581-584, 1997.

[13] P. Doanis, A. D. Boursianis, J. Huillery, A. Bréard, Y. Duric, and S. K. Goudos, "Differential Evolution in Waveform Design for Wireless Power Transfer", *Telecom*, Vol. 1, No. 2, pp. 96-113, 2020.

[14] Z. Dongming, X. Kewen, W. Baozhu, and G. Jinyong, "An Approach to Mobile IP Routing Based on QPSO Algorithm", In: *Proc. of 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, Wuhan, China, pp. 667-671, 2008.

[15] D. Mavares, M. Oropeza, and R. Velásquez, "Space-time code selection via particle swarm optimization", *Annals of Telecommunications*, Vol. 75, pp. 59-66, 2020.

[16] C. Ozturk, D. Karaboga, and B. Gorkemli, "Article Probabilistic Dynamic Deployment of Wireless Sensor Networks by Artificial Bee Colony Algorithm", *Sensors (Basel)*, Vol. 11, No.6, pp. 6056-6065, 2011.

[17] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press, Cambridge, MA, USA, 1992.

[18] E. Semenkin and M. Semenkina, "Self-configuring Genetic Algorithm with Modified Uniform Crossover Operator", In: *Proc. of International Conf. in Swam Intelligence*, Shenzhen, China, pp. 414-412, 2012.

[19] S. Mirjalili, J. S. Dong, A. S. Sadiq, and H. Faris, *Genetic Algorithm: Theory, Literature Review, and Application in Image Reconstruction, Nature-Inspired Optimizers*, Springer, Cham, Switzerland, 2020.

[20] J. Kennedy and R. Eberhart, "Particle swarm optimization", In: *Proc. of International Conf. On Neural Networks*, Perth, WA, Australia, pp. 1942-1948, 1995.

[21] S. Sengupta, S. Basak, and R. A. II. Peters, "Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives", *Machine Learning and Knowledge Extraction*, Vol. 1, No. 1, pp. 157-191, 2019.

[22] M. Dorigo and T. Stützle, *Handbook of Metaheuristics: Ant Colony Optimization: Overview and Recent Advances*, Springer, Cham, Switzerland, 2019.

[23] M. Dorigo, *Optimization, Learning and Natural Algorithms*, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[24] D. Karaboga, *An idea based on Honey Bee Swarm for Numerical Optimization Technical Report*, Erciyes University, Kayseri, Turkey, 2005.

[25] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", *Journal of Global Optimization*, Vol. 11, No. 4, pp. 341-359, 1997.

[26] P. Boonserm and S. Sitjongsataporn, "A robust and efficient algorithm for numerical optimization problem: DEPSO-Scout: A new hybrid algorithm based on DEPSO and ABC", In: *Proc. of 2017 International Electrical Engineering Congress*, Pattaya, Thailand, pp. 1-4, 2017.

[27] V. Miranda and R. Alves, "Differential Evolutionary Particle Swarm Optimization (DEEPSO): A Successful Hybrid", In: *Proc. of 2013 BRICS Congress On Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, Ipojuca, Brazil, pp. 368-374, 2013.

[28] B. Luitel and G. K. Venayagamoorthy, "Differential evolution particle swarm optimization for digital filter design", In: *Proc. of 2008 IEEE Congress On Evolutionary Computation*, Hong Kong, China, pp. 3954-3961, 2008.

[29] L. Han and X. He, "A Novel Opposition-Based Particle Swarm Optimization for Noisy

Received: August 10, 2020. Revised: September 3, 2020.

Problems", In: *Proc. of Thrid International Conf. On Natural Computation*, Haikou, China, pp. 624-629, 2007.