Received March 4, 2022, accepted March 20, 2022, date of publication April 8, 2022, date of current version April 26, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3165830

Parallel Hybrid Island Metaheuristic Algorithm

JIAWEI LI, (Member, IEEE), AND TAD GONSALVES
Department of Information and Communication Sciences, Faculty of Science and Technology, Sophia University, Tokyo 102-8554, Japan
Corresponding author: Tad Gonsalves (t-gonsal@sophia.ac.jp)

ABSTRACT This study introduces a novel Parallel Hybrid Island architecture which shows a parallel way to combine different meta-heuristic algorithms by using the island model as the base. The corresponding hybrid algorithm is called Parallel Hybrid Island Metaheuristic Algorithms (PHIMA). The hybrid parallel structure exploits the characteristics of the individual metaheuristic algorithms to boost robustness and diversity. Island Genetic Algorithm has been combined with Particle Swarm Optimization and Fireworks Algorithm to build three different PHIMA algorithms: PSO-GA (PHIMA-PGA), FWA-GA (PHIMA-FGA) and FWA-PSO-GA (PHIMA-FPGA). Further, another implementational variation known as ''co-evolution'' is applied to the sub-GA islands of PHIMA-FPGA to improve the performance on multi-modal high-dimensional problems. This variation is referred to as PHIMA-FPGA-Co. Each PHIMA Algorithm exhibits different advantages and characteristics, and the parallel hybridization using the island model is found to improve robustness and population diversity. The performances of the four new algorithms are compared with each other and that of the traditional Island GAs and all four proposed PHIMA algorithms show better result quality.

INDEX TERMS Meta-heuristic algorithms, hybrid algorithms, optimization, genetic algorithm, particle swarm algorithm, fireworks algorithm, co-evolution, island model.

I. INTRODUCTION

The Genetic Algorithm (GA), found in a wide range of successful applications such as helicopter autopilot [1], railway systems [2], big data [3] and many other areas of modern technology has gone through many changes over the years. For instance, Deb [4] tried to use decimal-chromosome GA to solve multi-objective problems and the proposed algorithm showed a fast convergence rate. Furthermore, researchers have paid a lot of attention to the operator improvement in GA, such as the adaptive probability of crossover and mutation [5], different selection approaches [6] and adaptation co-evolution for high-dimension problems [7].

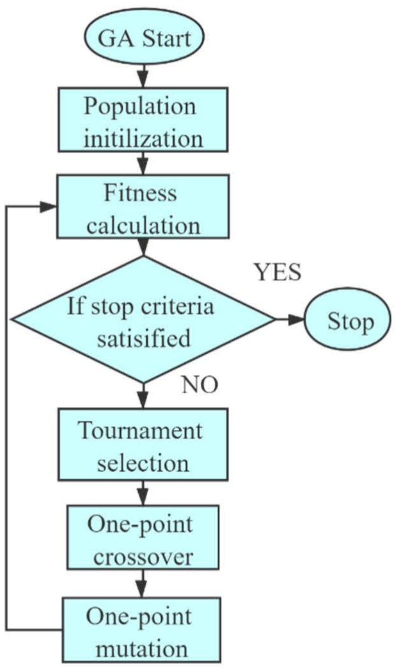
Particle Swarm Optimization (PSO), inspired by the swarm behavior of birds and insects is a popular research topic and has been successfully applied in many areas, such as antenna design [8], railway vehicle suspension systems [9] and parameter optimization for deep learning models [10].

The Fireworks Algorithm (FWA) [11], as its name implies, borrows ideas from the explosion of fireworks in the night sky. FWA has been successfully applied in areas such as power systems [12], image recognition [13], and financial systems [14].

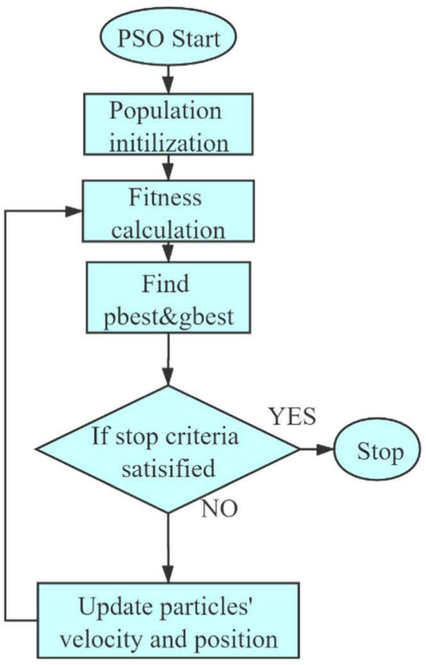
The associate editor coordinating the review of this manuscript and approving it for publication was Hisao Ishibuchi.

The GA, PSO, and FWA metaheuristic algorithms have received much attention in research and real-world applications. Although outstanding in their performance, these traditional metaheuristic algorithms suffer from the problem of premature convergence, especially in multi-modal and high-dimensional problems [15], [16]. Many studies have attempted to improve the performance of these algorithms by tweaking their internal structures. For example, in 2013, Melin *et al.* [17] used three fuzzy systems to adjust the PSO parameters and proved that this method can improve the optimization results. In 2017, Deng *et al.* [18] improved the PSO by applying alpha-stable distribution to update the particles and used dynamic inertia weight to improve the population search. In 2013, Zheng *et al.* [19] improved the simple FWA by adding new minimal explosion sparks, a new mapping strategy for out-space sparks and new operations for selection and Gaussian spark generation.

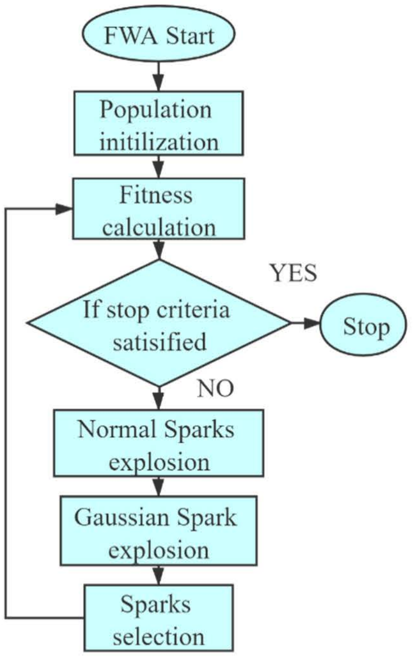
The memetic algorithm can also be regarded as an improvement over the meta-heuristic algorithm, because it combines the meta-heuristic algorithm and local search approach to achieve a good performance on both exploitation and exploration abilities. For example, in 2019, Gong *et al.* [20] proposed an Effective Memetic Algorithm (EMA) to solve the multi-objective job-shop scheduling problem (MOJSSP). Based on the simple Memetic Algorithm (MA), they applied a new hybrid crossover operator (HCO)



(a)



(b)



(c)

FIGURE 1. (a) GA flowchart (b) PSO flowchart (c) FWA flowchart.

and a new efficient variable neighborhood search approach (VNSA) to improve the memetic algorithm.

In spite of the detailed improvements, the existing meta-heuristic algorithms still suffer from several drawbacks such as slow convergences rate, trapping into local optima, having complex operators, long computational time, and so on [21]. In particular, they suffer from premature convergence in the case of multi-modal and high-dimensional problems. Some researchers have tried to modify their architecture, parameters and improve the partial operators of the algorithm to solve these weaknesses.

One such approach is to combine different algorithms to improve their performance. Manasrah and Ali [22] proposed a hybrid GA-PSO algorithm for cloud computing in 2018. The algorithm uses GA computations in the first half of the total iterations (n/2), followed by PSO computations for the remaining iterations. This hybrid algorithm was compared with GA, PSO, HSGA, WSGA, and MTCT. The experiments showed that the proposed algorithm can effectively reduce the total execution time compared to the other algorithms. In 2019, Şenel *et al.* [23] proposed a hybrid PSO-GWO algorithm. In PSO-GWO, the particles generated by PSO have a probability of being further optimized by the grey wolf optimizer (GWO) before the next PSO iterations. This hybrid algorithm shows better result quality and faster convergence speed compared to the simple PSO and GWO. In 2018, Tam *et al.* [24] proposed a hybrid GA-ACO-PSO algorithm for material identification. In this algorithm, the population is first optimized by the GA, and then further processed by ACO and PSO. The role of the ACO and PSO in this algorithm is to improve diversification and intensification, respectively. In 2020, Taher *et al.* [25] improved the GA by using an artificial bee colony (ABC). The new algorithm called GABC was tested on random number generation (RNG) and travelling salesman problem (TSP). The performance of GABC is better than that of the traditional GA.

However, most hybrid algorithm approaches hybridize different algorithms in series, which means the algorithms or parts of the algorithms are processed one after another. Only a few studies have attempted to combine algorithms in parallel.

One example is a novel PSO-GA-based hybrid algorithm proposed by Shi *et al.* [26]. The entire population of this algorithm is first divided into two equal parts: one sub-population corresponds to PSO, and the other corresponds to GA. Each sub-population processes independently during one single iteration step. The two sub-populations are then mixed at the end of each iteration step and moved to the next iteration step. This algorithm shows an effective performance in the experiments.

To address the issue of premature convergence in the case of multi-modal and high-dimensional problems, this paper introduces a novel method to hybridize algorithms in parallel using an island model. The island model of GA is one of the most famous applications of this idea [27]–[29]. Some researchers have also applied the island model to the PSO to improve its performance [30], [31].

The new hybrid approach proposed in this study is called Parallel Hybrid Island (PHI) architecture, the

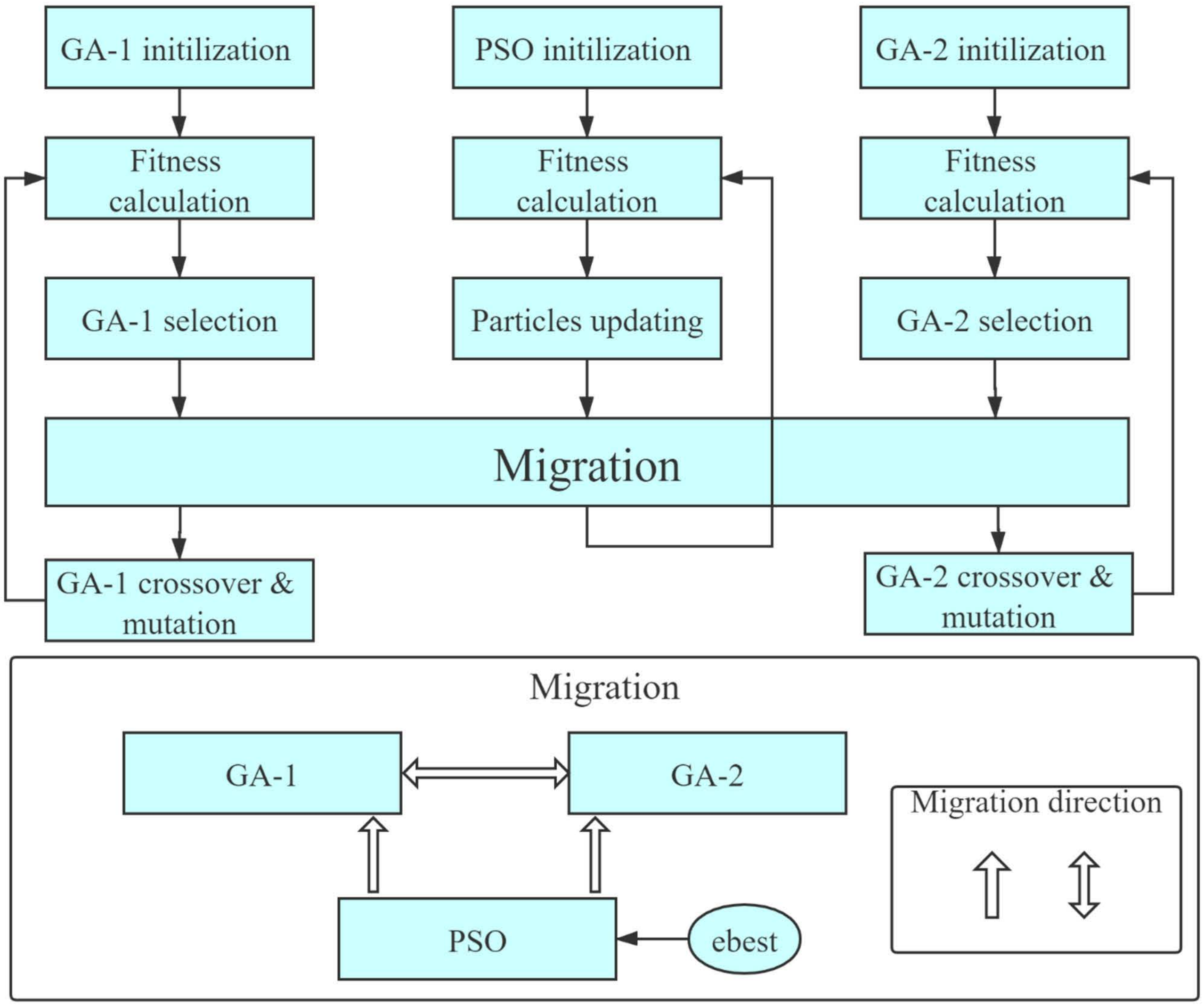


FIGURE 2. Flowchart of PHIMA-PGA and its migration.

corresponding hybrid metaheuristic algorithm is called Parallel Hybrid Island Metaheuristic Algorithm (PHIMA). In the PHI architecture, different algorithms are combined using the island model. Each algorithm corresponds to one or several islands, and processes information independently during each generation for most of the time. Sometimes, the islands share key information among themselves by using the migration operation.

By combining PSO and/or FWA to the island GA, three novel PHIMA algorithms are proposed in this paper: PHIMA-PGA (PSO and GA), PHIMA-FGA (FWA and GA), PHIMA-FPGA (FWA, PSO and GA). By combining PSO and/or FWA into the island GA using PHI architecture, the population diversity of the algorithm is expected to be enhanced and therefore, the result quality can be improved. Moreover, a co-evolution structure is applied to the sub-GA islands of the PHIMA-FPGA, to enhance the performance of the algorithm on high-dimensional problems. The PHIMA-FPGA with the co-evolution strategy is the fourth PHIMA algorithm and is referred to as PHIMA-FPGA-Co in this study.

The multi-population feature of the PHI architecture can increase the diversification of the algorithm population and combine the characteristics of the different sub-algorithms. The experimental results show that in most cases, all four PHIMA algorithms have better performance in terms of the result quality compared to the traditional Island GA algorithm. Additionally, PHIMA-PGA and PHIMA-FPGA exhibit faster convergence speeds. Finally, the comparison between PHIMA-FPGA and PHIMA-FPGA-Co proves the co-evolution structure can improve the convergence speed on all the proposed benchmark functions.

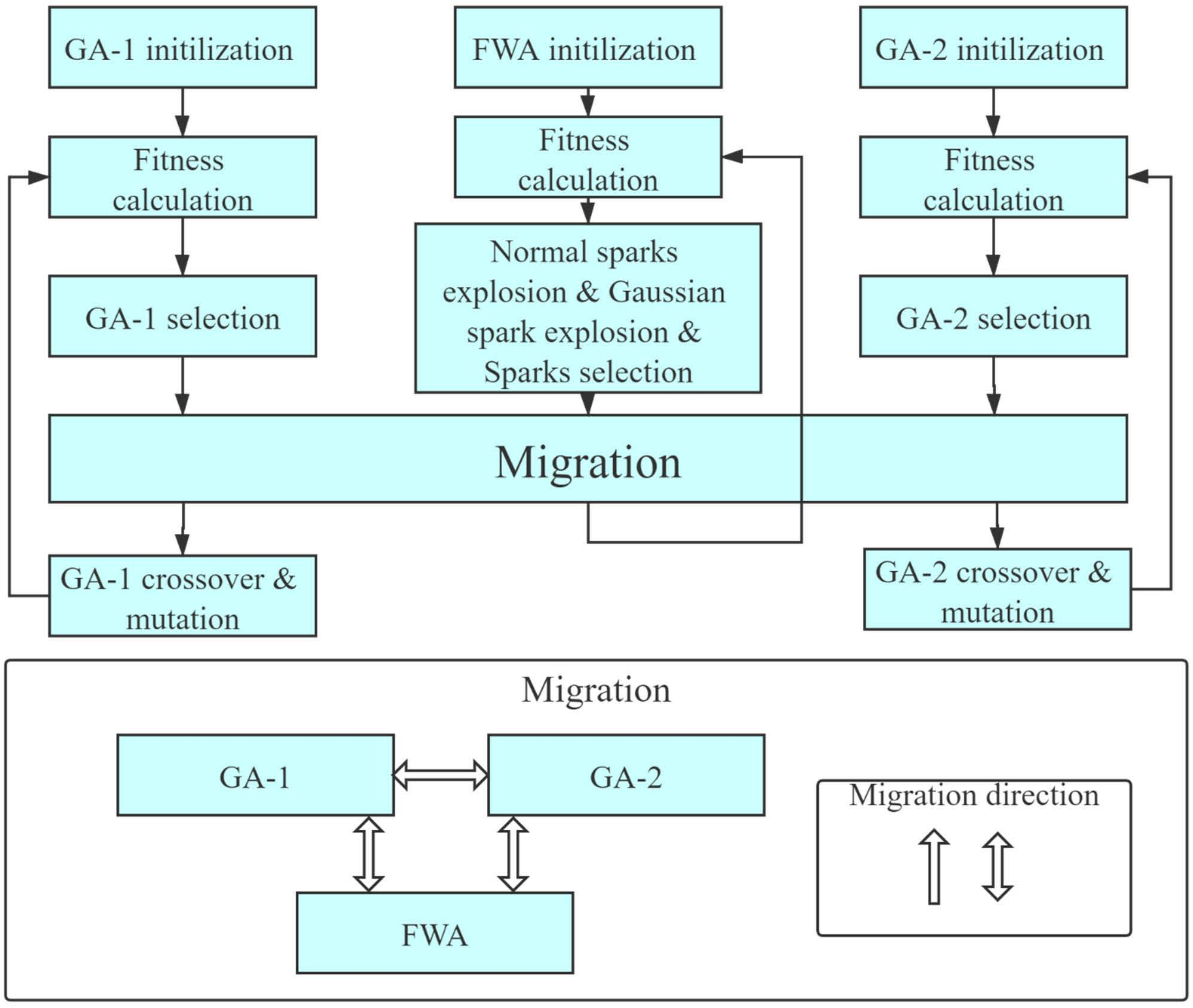


FIGURE 3. Flowchart of PHIMA-FGA and its migration.

This paper is organized as follows: Section II briefly describes the three meta-heuristic algorithms forming the fundamental building blocks of our new parallel hybrid algorithm. Section III describes in detail the hybrid island meta-heuristic structure. Section IV tabulates the results and discusses their significance. Section V concludes the study and indicates areas for further research.

II. METAHEURISTIC ALGORIHTMS

Metaheuristic algorithms are widely used in solving optimization problems. This section briefly describes three well-known algorithms: Genetic Algorithm, Particle Swarm Optimization, and Fireworks algorithm, which are used as fundamental building blocks of the hybrid metaheuristic described in the next section. The island model and the co-evolution strategy are also described at the end of the section.

A. GENETIC ALGORITHM

The Genetic Algorithm (GA) metaphor was inspired by the Darwinian theory of evolution. The algorithm mimics the principle of genetics in the natural world, and produces optimal solutions based on the idea of *the survival of the fittest*. It begins with a group of possible solutions called ‘‘individuals’’ or ‘‘chromosomes’’. Each chromosome contains several ‘‘genes’’, which define the features of the individual. The initial chromosomes are randomly generated and evaluated for their fitness which depends on the objective function. Individuals with higher fitness values have a higher opportunity for survival. The selection operation selects better fit individuals from the original population. These individuals are then

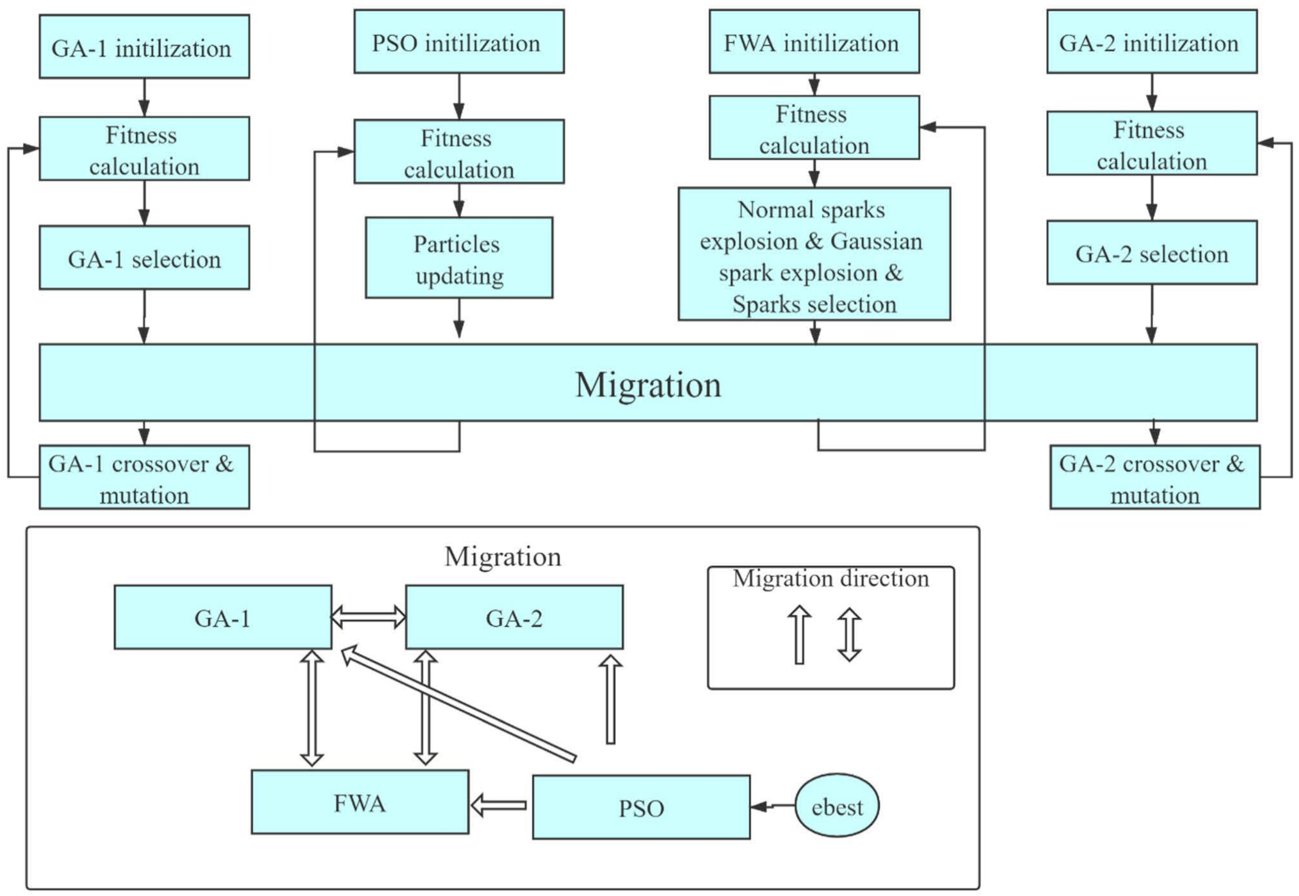


FIGURE 4. **Flowchart of PHIMA-FPG and its migration.**

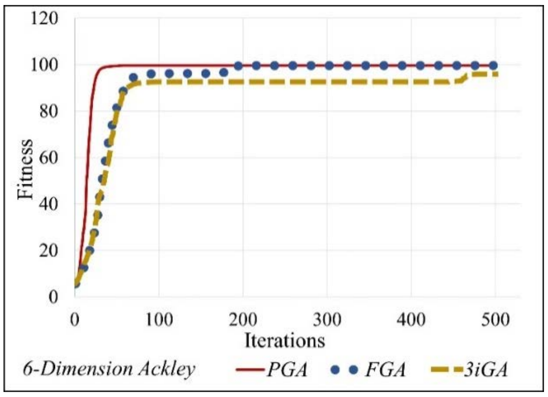


FIGURE 5. **Experiment 1 6-Dimension Ackley.**

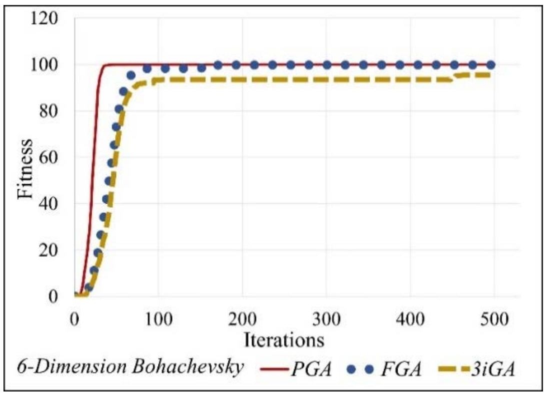


FIGURE 6. **Experiment 1 6-Dimension Bohachevsky_1.**

subjected to ‘‘crossover’’ with a ‘‘crossover probability (p_c)’’ and then ‘‘mutation’’ operations with a ‘‘mutation probability (p_m)’’. Subsequently, the offspring population is generated and moved to the next iteration step. The GA population is allowed to evolve through a predetermined number of iterations (Fig. 1). In the proposed algorithms, the tournament selection with size 2, single-point crossover and single-point mutation operators are used for the GA part.

B. PARTICLE SWARM OPTIMIZATION

The Particle Swarm Optimization (PSO) metaphor is inspired by how swarms of insects and flocks of birds search for food in the natural world. Each candidate solution is called a ‘‘particle’’ in PSO, which corresponds to a single bird in the flock or an insect in a swarm in the natural world. By sharing key information, the individuals collaborate to

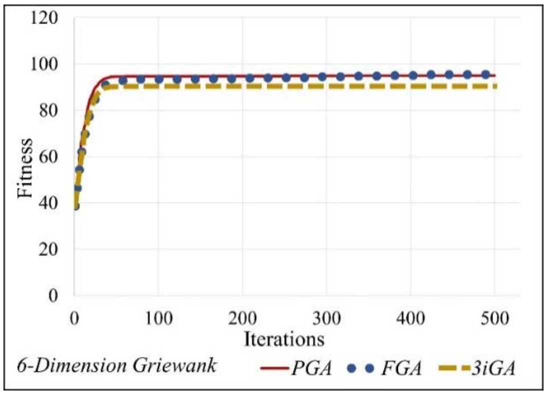


FIGURE 7. Experiment 1 6-Dimension Griewank.

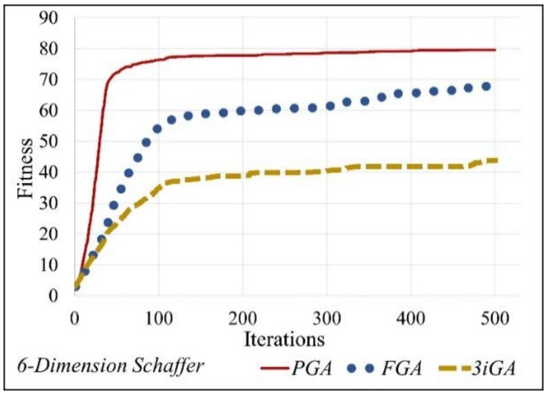


FIGURE 8. Experiment 1 6-Dimension Schaffer.

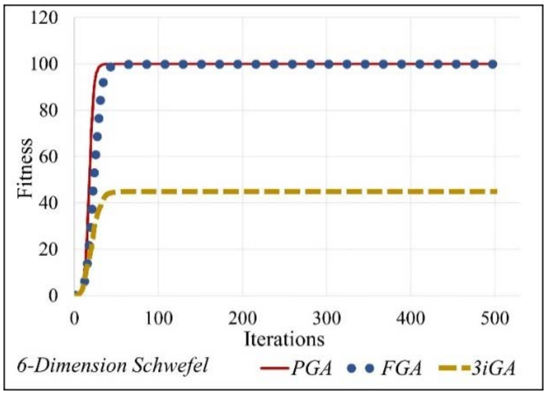


FIGURE 9. Experiment 1 6-Dimension Schwefel.

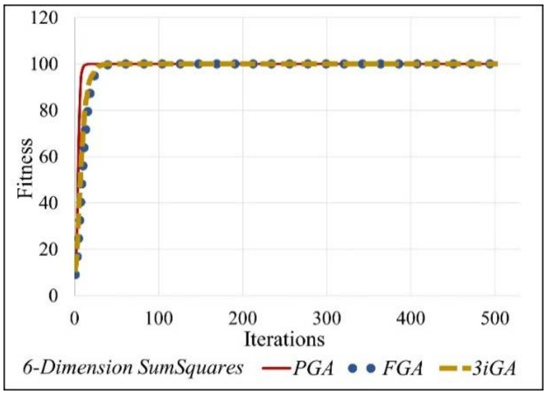


FIGURE 10. Experiment 1 6-Dimension SumSquares.

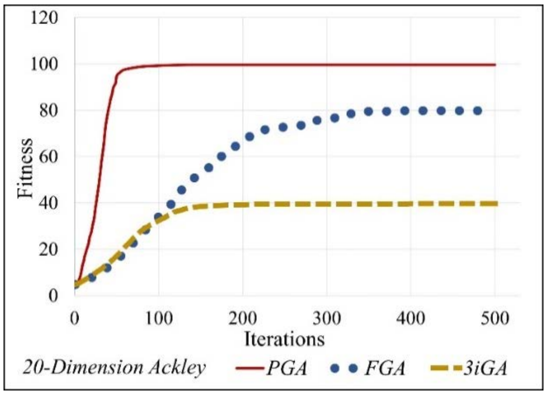


FIGURE 11. Experiment 1 20-Dimension Ackley.

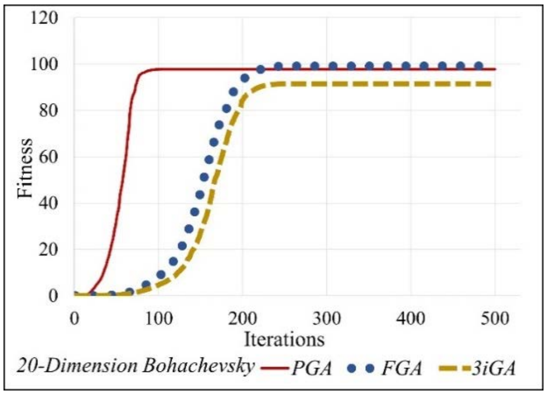


FIGURE 12. Experiment 1 20-Dimension Bohachevsky_1.

find the "food", which corresponds to the global solution of an optimization problem.

In PSO, each particle has two features, velocity and position. During each iteration, each particle updates its velocity and direction based on four parameters: its previous position, its previous velocity, its own historic optimal position, and the current global optimal position. The particle velocity and position updating equations are given in (1) and (2) [32].

$$v_i = w * v_{i-1} + c_1 * rand() * \left(pbest_i - x_i\right) + c_2 * rand() * (gbest - x_i) \tag{1}$$

$$x_i = x_{i-1} + v_i \tag{2}$$

where, *pbest* is the historic optimal position for each particle, *gbest* is the current global optimal position, v is the particle velocity, w is the inertia weight, x is the particle position. c_1 and c_2 are learning factors that are constants normally between 1 and 2, and $rand()$ is a uniform random number between 0 and 1. To avoid over-large or over-small movements, based on (2), particle velocity bounds are imposed as

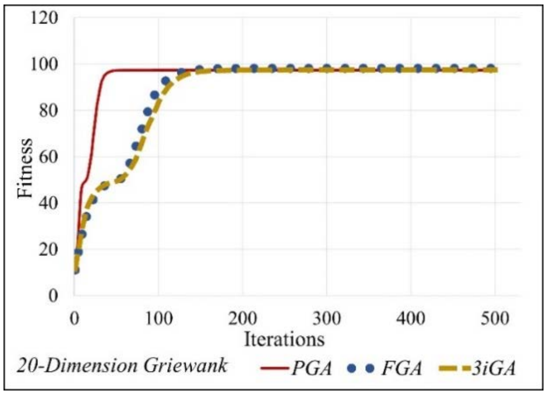


FIGURE 13. Experiment 1 20-Dimension Griewank.

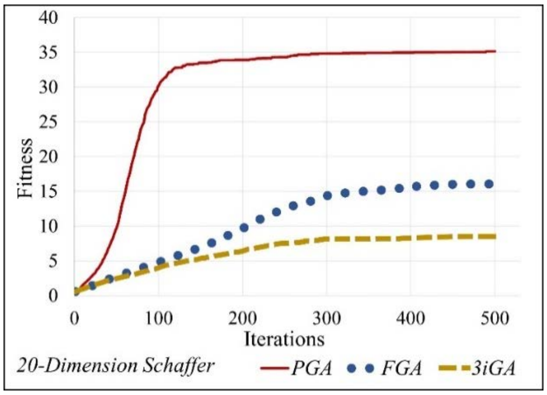


FIGURE 14. Experiment 1 20-Dimension Schaffer.

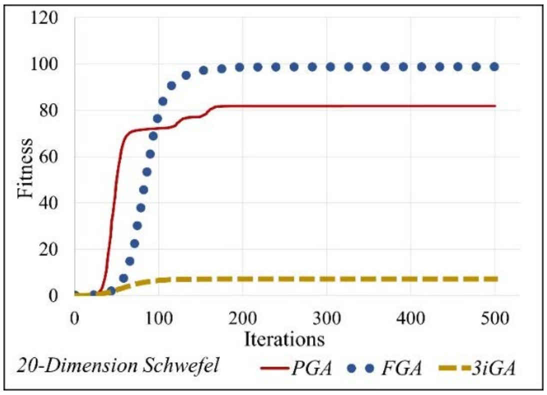


FIGURE 15. Experiment 1 20-Dimension Schwefel.

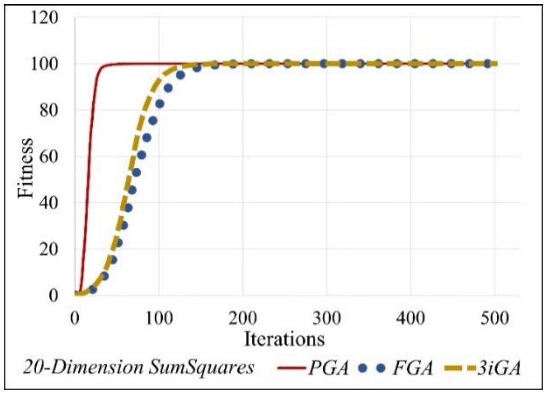


FIGURE 16. Experiment 1 20-Dimension SumSquares.

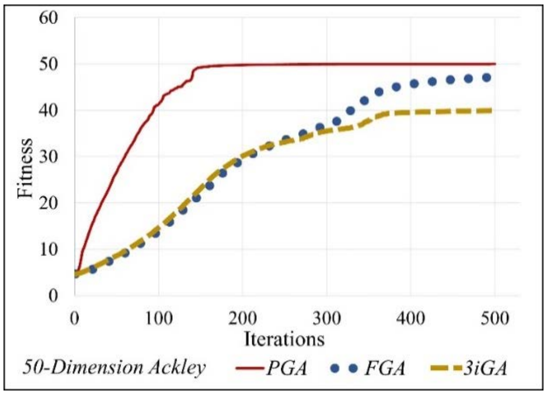


FIGURE 17. Experiment 1 50-Dimension Ackley.

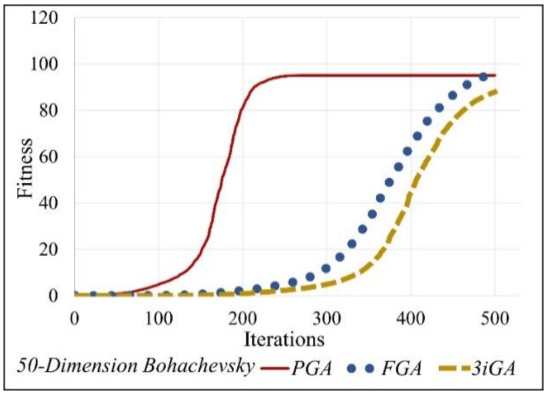


FIGURE 18. Experiment 1 50-Dimension Bohachevsky_1.

shown in (3), where Thr_{pso_min} and Thr_{pso_max} are the lower and upper bound respectively. The flowchart of the PSO used in this paper is shows in Fig.1.

$$v_{i,} = \begin{cases} v_{i,}, Thr_{pso\_max} \geq |v_{i,}| \geq Thr_{pso\_min} \\ \pm Thr_{pso\_min}, |v_{i,}| < Thr_{pso\_min} \\ \pm Thr_{pso\_max}, |v_{i,}| > Thr_{pso\_max} \end{cases} \quad (3)$$

C. FIREWORKS ALGORITHM

The Fireworks algorithm (FWA) proposed by Tan in 2010 [11] is inspired by the explosion of fireworks in the night sky. The first step in the FWA is to generate a group of random individuals. Each individual called a ''spark'' is evaluated and given a fitness value. During the calculation, each spark will further generate several sparks for the next iteration step depending on its fitness value. The spark with a higher fitness generates more sparks with a shorter explosion radius, whereas a lower-fitness spark generates fewer sparks with a longer explosion radius. The explosion radius A_i and

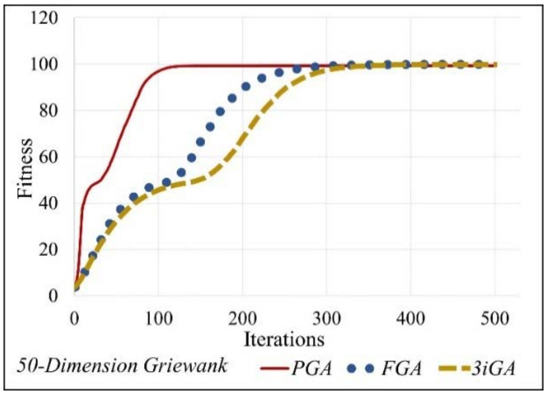


FIGURE 19. Experiment 1 50-Dimension Griewank.

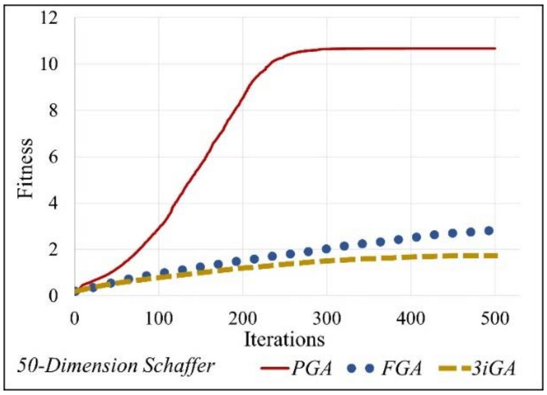


FIGURE 20. Experiment 1 50-Dimension Schaffer.

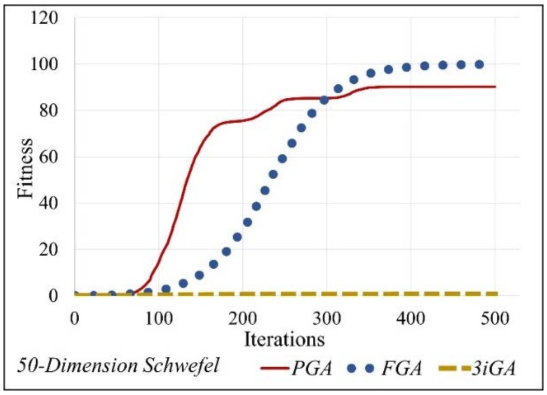


FIGURE 21. Experiment 1 50-Dimension Schwefel.

the number of sparks S_i are computed using (4) and (5) [11].

$$A_i = \hat{A} * \frac{f(x_i) - y_{min} + \varepsilon}{\sum_{i=1}^{N} (f(x_i) - y_{min}) + \varepsilon} \tag{4}$$

$$S_i = M * \frac{y_{max} - f(x_i) + \varepsilon}{\sum_{i=1}^{N} (y_{max} - f(x_i)) + \varepsilon} \tag{5}$$

where, $\hat{A}$ and M are the parameters to control the explosion radius and the total number of sparks respectively. y_{max} and y_{min} correspond to the worst individual and best individuals, respectively, in the case of minimization problems.

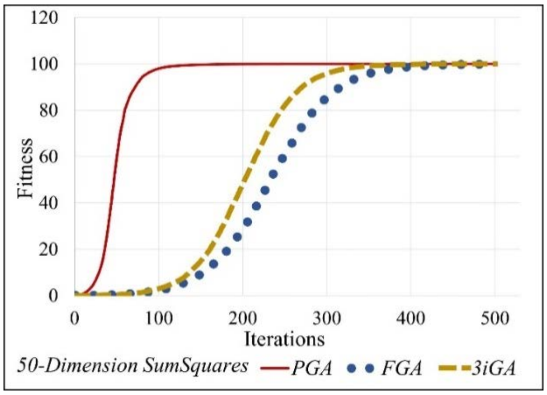


FIGURE 22. Experiment 1 50-Dimension SumSquares.

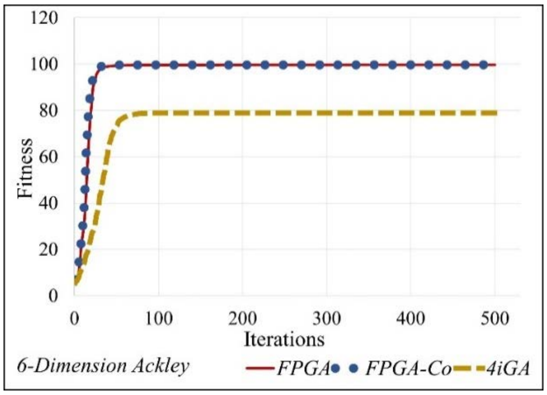


FIGURE 23. Experiment 2 6-Dimension Ackley.

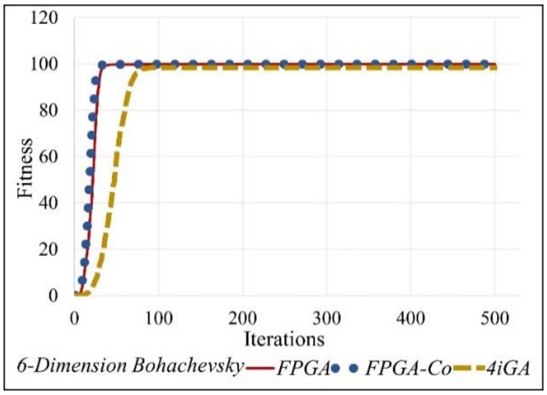


FIGURE 24. Experiment 2 6-Dimension Bohachevsky_1.

In the real calculation of FWA, a spark sometimes produces excessive sparks which will slow down the calculation time, or the value of S_i can be lower than 1 and therefore the information of this spark will be lost. Therefore, S_i bounds are imposed as shown in (6) [19], where a and b are both constants and $a<b<1$. Moreover, similar to equation (3) in the Section II.B, to avoid over-small search precision, a lower precision threshold Thr_{fwa} is imposed on the explosion radius

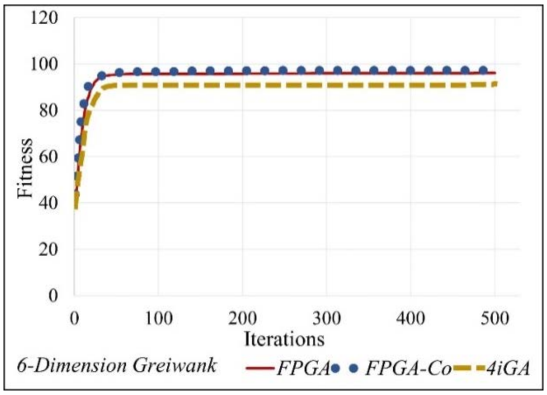


FIGURE 25. Experiment 2 6-Dimension Griewank.

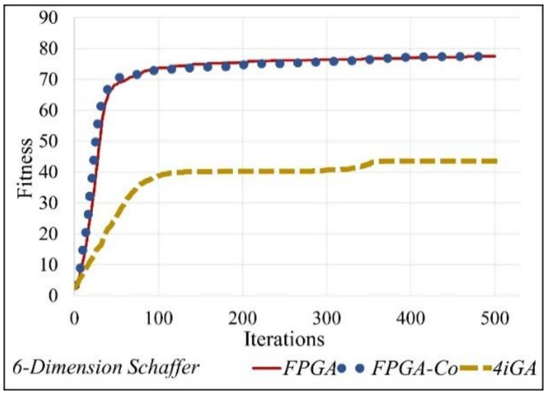


FIGURE 26. Experiment 2 6-Dimension Schaffer.

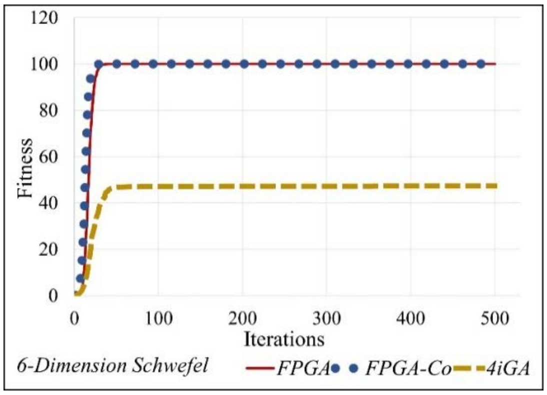


FIGURE 27. Experiment 2 6-Dimension Schwefel.

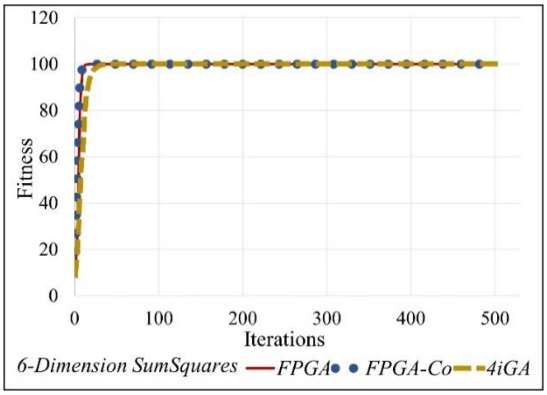


FIGURE 28. Experiment 2 6-Dimension SumSquares.

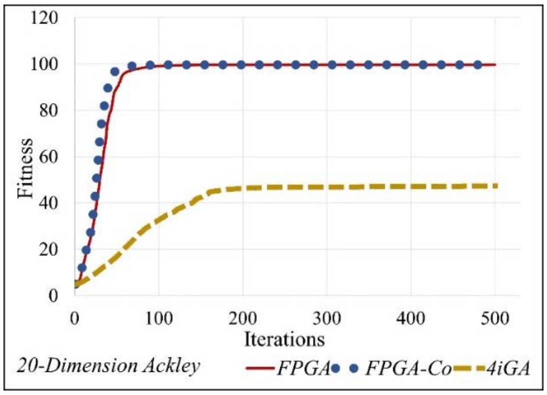


FIGURE 29. Experiment 2 20-Dimension Ackley.

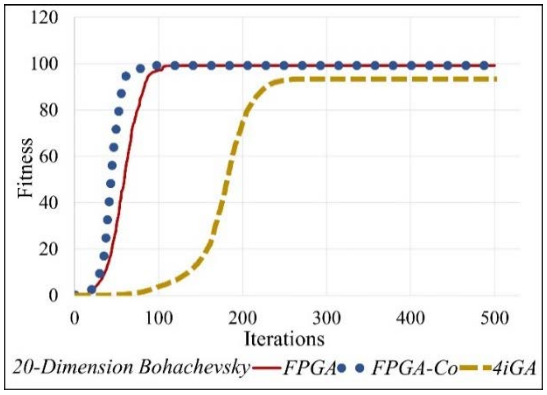


FIGURE 30. Experiment 2 20-Dimension Bohachevsky_1.

A_i as shown in (7) [19].

$$S_i = \begin{cases} round\ (a * M)\ , & S_i < aM \\ round\ (b * M)\ , & S_i > bM \\ round\ (S_i)\ , & other \end{cases} \tag{6}$$

$$A_i = \begin{cases} \dfrac{f\ (x_i) - y_{min} + \varepsilon}{\sum_{i=1}^{N} (f\ (x_i) - y_{min}) + \varepsilon} * \hat{A}, & A_i \geq Thr_{fwa} * \hat{A} \\ Thr_{fwa} * \hat{A}, & A_i < Thr_{fwa} * \hat{A} \end{cases} \tag{7}$$

To increase the diversity of the population, mutation operation is added. After generating normal sparks, a random spark will be selected. Then certain dimensions of the selected spark are subjected to a Gaussian mutation operation to generate a mutation spark or Gaussian spark. The generation of a Gaussian spark follows (8).

$$x_{ik} = x_{ik} * e \tag{8}$$

where, e is the Gaussian distribution with both mean and variance values equal to 1.

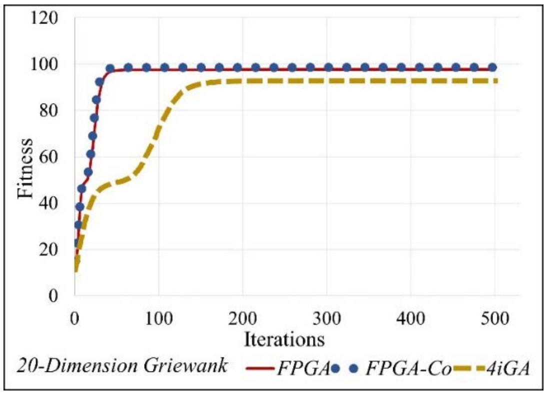


FIGURE 31. Experiment 2 20-Dimension Griewank.

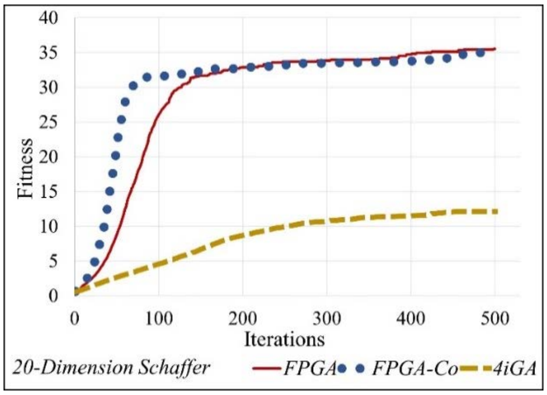


FIGURE 32. Experiment 2 20-Dimension Schaffer.

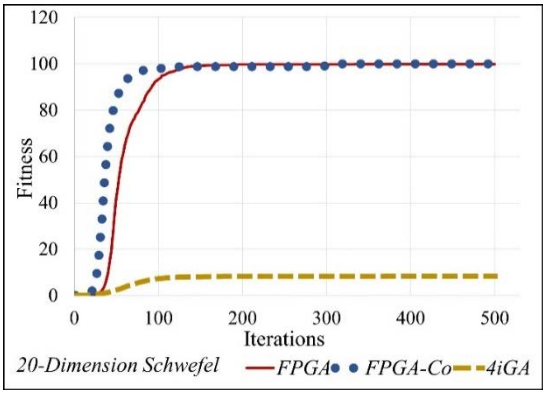


FIGURE 33. Experiment 2 20-Dimension Schwefel.

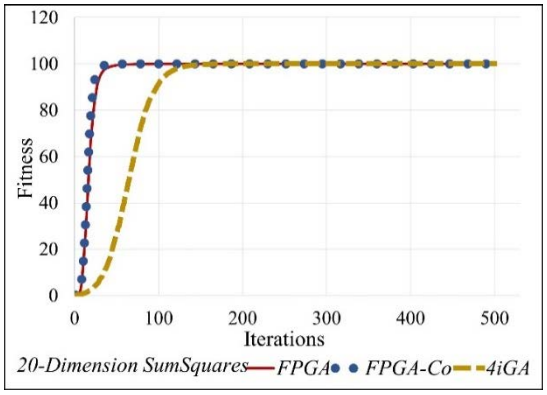


FIGURE 34. Experiment 2 20-Dimension SumSquares.

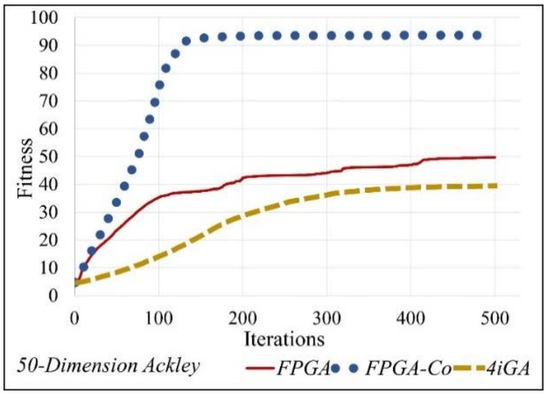


FIGURE 35. Experiment 2 50-Dimension Ackley.

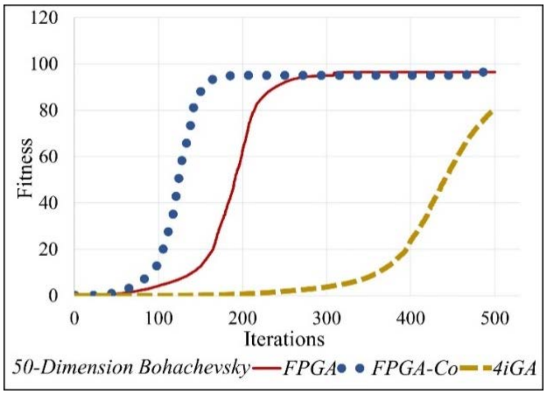


FIGURE 36. Experiment 2 50-Dimension Bohachevsky_1.

After the normal sparks and Gaussian spark explosions, the best spark in the candidate population is directly passed to the next iteration. The remaining sparks are selected by the selection operator. Roulette Wheel Selection was used in the original FWA paper [11]. The selection probability of each spark is calculated using (9) and (10).

$$p(x_i) = \frac{R(x_i)}{\sum R(x_j)} \tag{9}$$

$$R(x_i) = \sum d\left(x_i - x_j\right) = \sum \left\|x_i - x_j\right\| \tag{10}$$

where, $R(x_i)$ is the total distance from the spark x_i to all other sparks. The equation indicates that the sparks that are relatively far from other sparks will have a relatively high probability of being selected. This enhances the diversity of the population. However, the distance calculation is very time-consuming, and the computation time would be very large when the population size increases, such as for some complex or high-dimension problems.

Therefore, instead of Roulette Wheel selection, Elitism-Random Selection is used in this study. As the name implies,

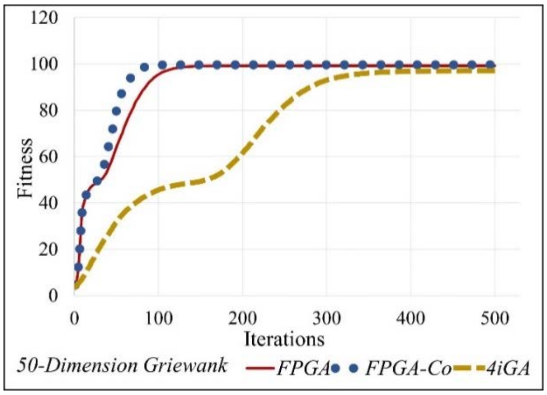


FIGURE 37. Experiment 2 50-Dimension Griewank.

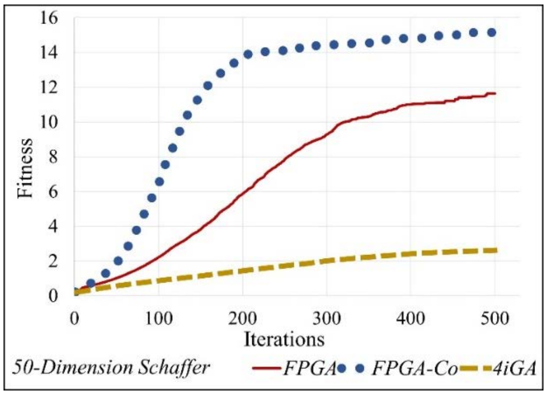


FIGURE 38. Experiment 2 50-Dimension Schaffer.

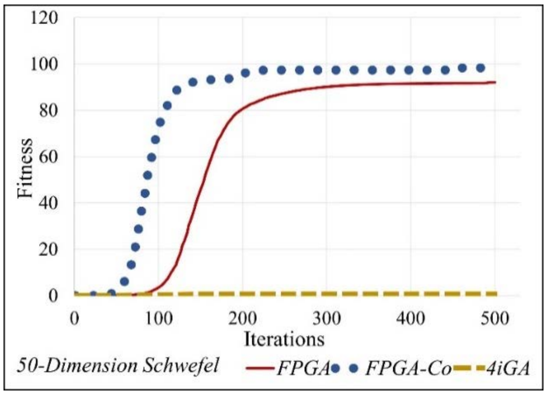


FIGURE 39. Experiment 2 50-Dimension Schwefel.

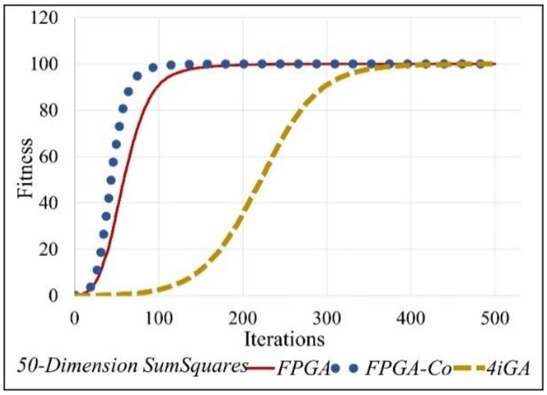


FIGURE 40. Experiment 2 50-Dimension SumSquares.

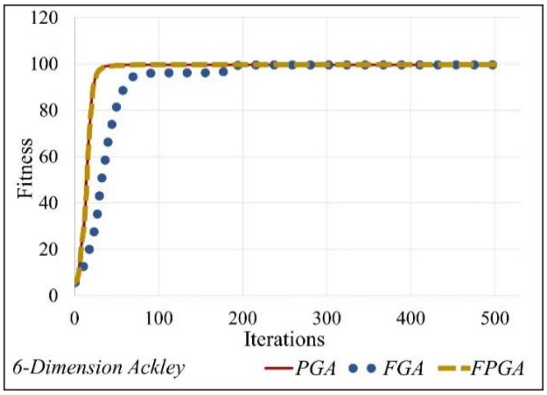


FIGURE 41. Experiment 3 6-Dimension Ackley.

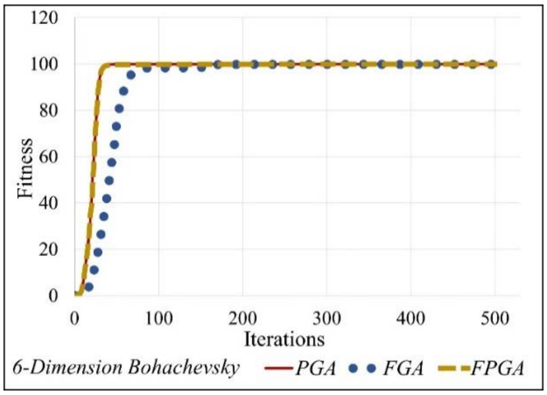


FIGURE 42. Experiment 3 6-Dimension Bohachevsky_1.

in the Elitism-Random selection, the best solution is directly passed to the next iteration, and the remaining sparks are selected randomly. Elitism-Random Selection is a time-saving approach compared to the Roulette Wheel Selection [19]. The flowchart of the FWA which used in this paper is shown in Fig.1.

D. ISLAND MODEL

In the island GA, the whole population is divided into several sub-population groups called islands. Each island processes an entire GA computation independently and periodically shares fitter individuals between each other for every certain number of iterations. The sharing of individuals is called migration and the number of generations is called migration interval ($M_{interval}$). In the proposed model, during the migration, the top certain percentage (migration ratio, M_{ratio}) of the fitter individuals is selected from each island, and a copy of them replaces the lesser fit individuals in the other islands. Additionally, an extra elite individual from each island is transferred to the other islands during migration.

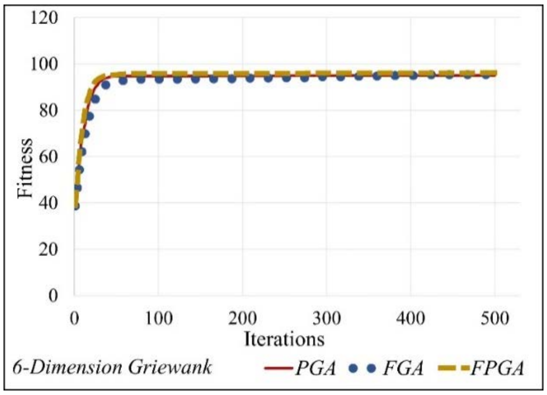


FIGURE 43. Experiment 3 6-Dimension Griewank.

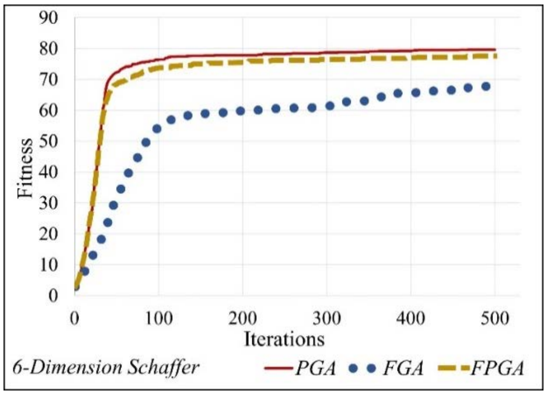


FIGURE 44. Experiment 3 6-Dimension Schaffer.

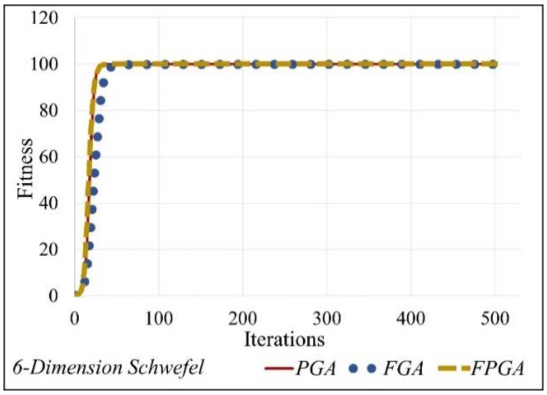


FIGURE 45. Experiment 3 6-Dimension Schwefel.

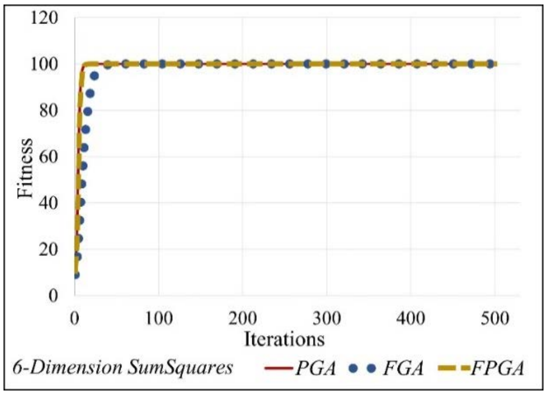


FIGURE 46. Experiment 3 6-Dimension SumSquares.

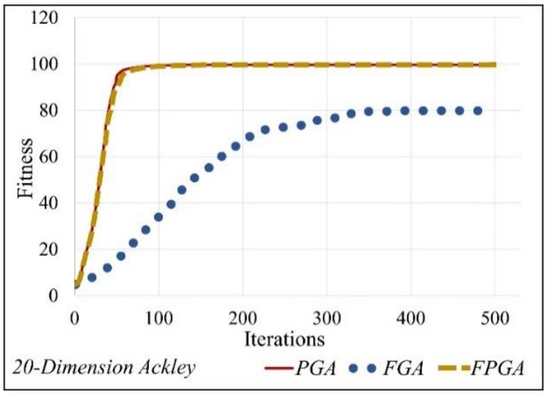


FIGURE 47. Experiment 3 20-Dimension Ackley.

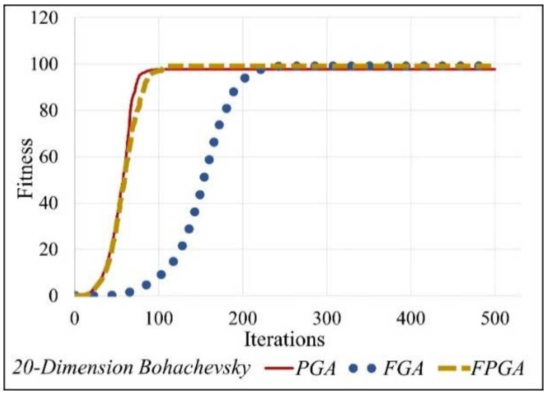


FIGURE 48. Experiment 3 20-Dimension Bohachevsky_1.

For convenience, the copy of good individuals and the extra elite doing the migration is referred to as migration group in this study.

It has been demonstrated by many researchers that the island GA improves the convergence rate and result quality compared to the classic GA [33], [34]. The independence feature of the island model enables the population to reach a wide search region with a larger opportunity, thus increasing the global search ability. Meanwhile, the migration operation allows the key information to be shared between each island so that the algorithm efficiency is maintained or even increased. The independence feature makes it easy and feasible to hybridize different algorithms using the one island model system.

E. CO-EVOLUTION

For high-dimensional problems, the traditional GA faces the problem of a slow convergence rate. The Cooperative Coevolutionary Algorithm can efficiently lower the dimensions of the problem to improve the performance on large-scale

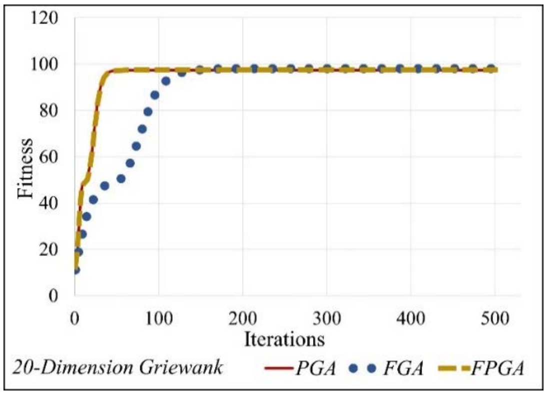


FIGURE 49. Experiment 3 20-Dimension Griewank.

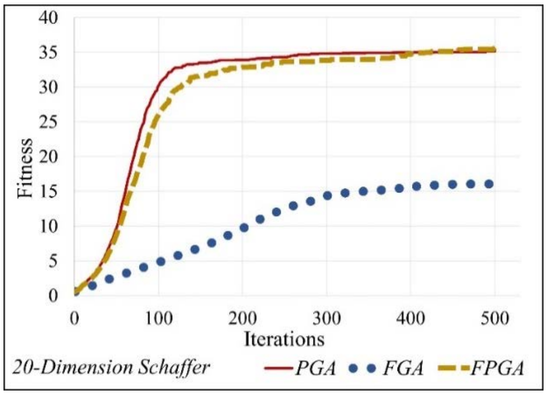


FIGURE 50. Experiment 3 20-Dimension Schaffer.

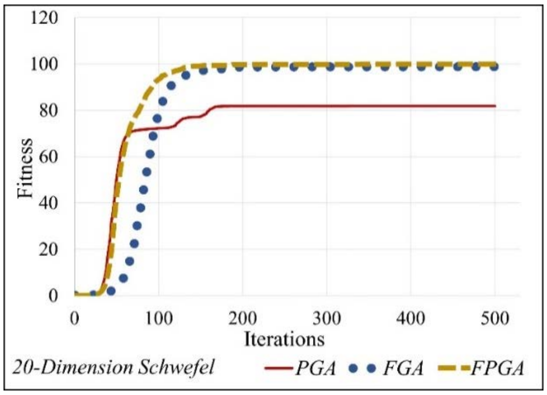


FIGURE 51. Experiment 3 20-Dimension Schwefel.

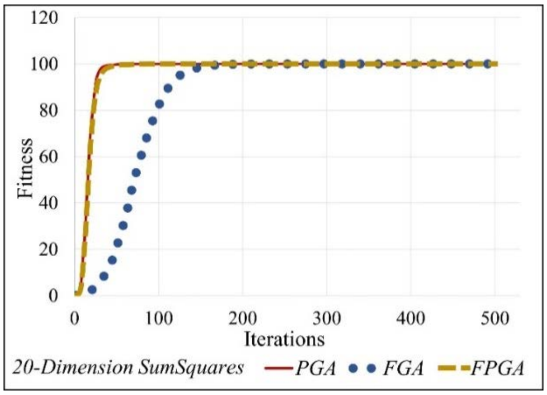


FIGURE 52. Experiment 3 20-Dimension SumSquares.

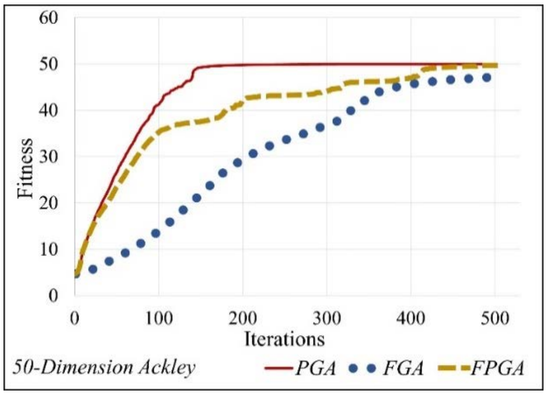


FIGURE 53. Experiment 3 50-Dimension Ackley.

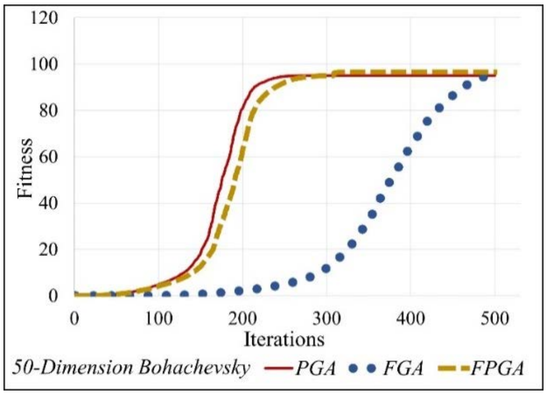


FIGURE 54. Experiment 3 50-Dimension Bohachevsky_1.

problems [7]. It consists in dividing the high-dimensional problem into several subproblems with relatively low dimensions. All subgroups evolve independently and solve the problem cooperatively. In the proposed model, the entire GA population ($GA^{combined}$) is equally divided into two sub-GAs, $subGA^1$ and $subGA^2$, based on the dimensions. Each sub-GA performs selection, crossover, and mutation independently. During the fitness calculation operation, an elite individual ($elite^{total}$) is selected. This $elite^{total}$ refers to the current global best solution for the entire population. The individuals from the first-half-dimension population ($subGA^1$) are combined with the last-half-dimension part of elite individuals ($elite^2$) to perform the fitness evaluation. Similarly, the individuals from the last-half-dimension population ($subGA^2$) are combined with the first-half-dimension part of elite individuals ($elite^1$) during the fitness evaluation. The new elite individuals are updated when all individuals from both the sub-GA populations are evaluated. Then each sub-GA population performs selection, crossover and mutation independently and continues to the next iteration. In Experiment 2, the

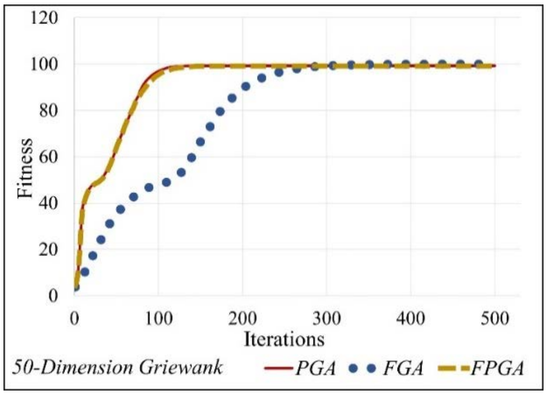


FIGURE 55. Experiment 3 50-Dimension Griewank.

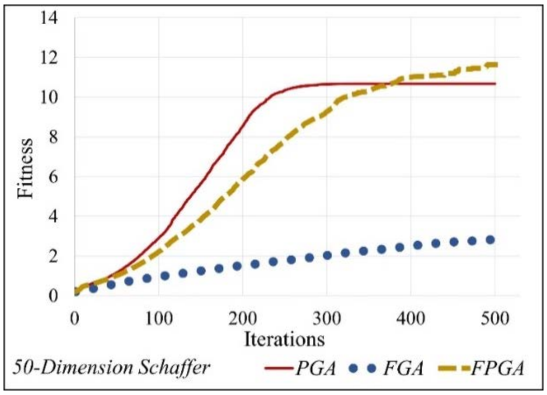


FIGURE 56. Experiment 3 50-Dimension Schaffer.

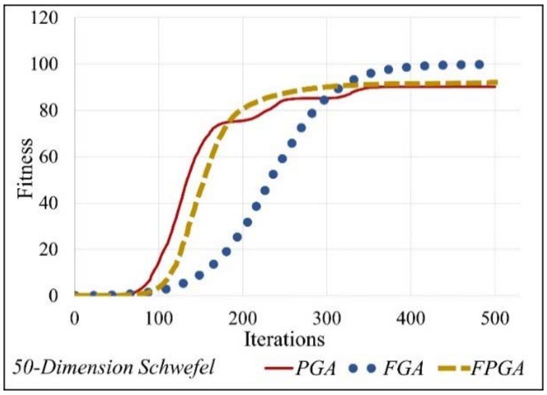


FIGURE 57. Experiment 3 50-Dimension Schwefel.

performance of the models with and without the Cooperative Coevolutionary strategy is compared. The experimental results show that the Cooperative Coevolutionary algorithm provides improvements in both convergence speed and result quality. The pseudocode code for a Cooperative Coevolutionary Algorithm with two sub-GAs is shown below:

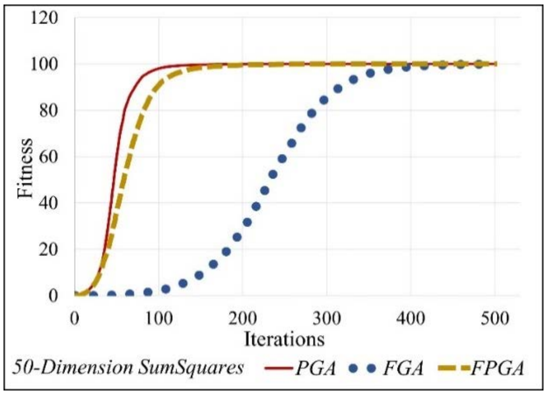


FIGURE 58. Experiment 3 50-Dimension SumSquares.

```
(1) Start
(2) Initialization of $subGA^1$ population group
(3) Initialization of $subGA^2$ population group
(4) Random combination of $subGA^1$ and $subGA^2$ to generate $GA^{combined}$ population group
(5) Fitness evaluation of $GA^{combined}$
(6) Find the current best individual $elite^{total}$ from $GA^{combined}$
(7) Divide $elite^{total}$ into $elite^1$ and $elite^2$
(8) While Stop condition is not met:
(9) Fitness evaluation of population $subGA_i^1$ by using $elite^2$
(10) Fitness evaluation of population $subGA_i^2$ using $elite^1$
(11) Find one current best individual $elite_{i+1}^{total}$ from $subGA_i^1$ and $subGA_i^2$
(12) Divide $elite_{i+1}^{total}$ into $elite_{i+1}^1$ and $elite_{i+1}^2$
(13) If the stop condition is satisfied:
(14) Output the result, Go to (23)
(15) Else:
(16) $subGA_i^1$ selection, generate $subGA_{i+1}^1$
(17) $subGA_i^2$ selection, generate $subGA_{i+1}^2$
(18) Crossover of $subGA_{i+1}^1$
(19) Crossover of $subGA_{i+1}^2$
(20) Mutation of $subGA_{i+1}^1$
(21) Mutation of $subGA_{i+1}^2$
(22) Go to (8)
(23) End
```

III. HYBRID-ISLAND MERAHEURISTIC STRUCTURE

In 2003, Osaba *et al.* [35] introduced a novel approach for the island GA model which contains several sub-GA islands with different crossover functions and dynamic crossover probabilities. From their experimental results, they proposed that ''the multi-crossover feature enhanced diversification, allowing a broader exploration'', because different crossover operations have different ways to explore the search space of the optimization problems. Similarly, different algorithms also have different ways to explore the search space.

Based on this idea, we propose a new hybrid architecture called the Parallel Hybrid Island (PHI) architecture. This approach uses the island model to combine different algorithms. The corresponding hybrid algorithms are called Parallel Hybrid Island Metaheuristic Algorithm (PHIMA). In PHIMA, each sub island corresponds to an algorithm that computes independently and then shares the key information during the migration operation. This parallel hybrid architecture can combine characteristics from different algorithms. The hybridization of populations from the different algorithms can improve population diversity because different algorithms have different evolution trails.

In this study, island GAs has been combined with PSO or/and FWA to form four PHIMAs: PHIMA-PGA (PSO and GA), PHIMA-FGA (FWA and GA), PHIMA-FPGA (FWA, PSO and GA) and PHIMA-FPGA-Co (PHIMA-FPGA with co-evolution).

In all four proposed PHIMAs, the number of GA islands is always larger than that of the other algorithms, making it the main body of the hybrid algorithm. The reason is that GA has a comprehensive and stable performance compared to PSO and FWA. Through experimentation, it is found that the proposed PHIMA algorithms with two sub-GA islands show better robustness compared to that with only one sub-GA island. PSO and FWA form the additional components of the new hybrid algorithm. Compared to GA, PSO has the advantages of faster convergence speed [36] and better local search ability [37]. The spark explosion of the FWA can further improve the diversity of the PHIMAs. Compared to the PSO and GA, FWA has a more stochastic way to update the search agents, because the new spark generation is random. Therefore, FWA is expected to further improve the diversity of the PHIMAs. By replacing the PSO and FWA, the PHIMA is expected to obtain different characteristics.

A. PHIMA-PGA

PHIMA-PGA is a 3-hybrid-island algorithm composed of two sub-GA islands and one sub-PSO island. Each island performs the optimization process independently and migration occurs every two iterations ($M_{interval} = 2$). The M_{ratio} is equal to 2% for every island. Fig. 2 shows the flowchart of PHIMA-PGA and the migration process. As illustrated in Fig. 2, the migration process occurs from GA1 to GA2, GA2 to GA1 and PSO to GA1 and GA2. The sub-PSO island only donates the copy of the migration population to the sub-GA islands. Moreover, during the migration, the sub-PSO island collects the information of the current global best solution, *ebest*, of the entire system of PHIMA-PGA system for updating the particle positions in the next iterations. The iteration equations of PSO ((1) and (2)) are accordingly modified into (11) and (12).

$$\begin{aligned} v_i = w * v_{i-1} + c_1 * rand() * \left(pbest_i - x_i\right) \\ + c_2 * rand() * (gbest - x_i) \\ + c_3 * rand() * (ebest - x_i) \end{aligned} \tag{11}$$

$$x_i = x_{i-1} + v_i \tag{12}$$

B. PHIMA-FGA

PHIMA-FGA is a 3-hybrid-island algorithm composed of two sub-GA islands and one sub-FWA island. Like PHIMA-PGA, each island performs the optimization process independently and migration occurs for every two iterations ($M_{interval} = 2$). The M_{ratio} is equal to 2% for every island. Migration occurs between every island, which means that every island donates the migration population to all other islands. Roulette Wheel Selection is replaced with the Elitism-Random selection. Fig. 3 shows the flowcharts of PHIMA-FGA and its migration process.

C. PHIMA-FPGA

PHIMA-FPGA is a 4-hybrid-island algorithm composed of two sub-GA islands, one sub-PSO island and one sub-FWA island. Migration occurs for every 2 iterations ($M_{interval} = 2$). The M_{ratio} equals 1.2% for every island. Fig. 4 shows the flowcharts of PHIMA-FPGA and its migration process. During migration, two sub-GAs and sub-FWA islands donate migration populations to each other. The sub-PSO island only donates the migration population to the other three islands and collects the information of *ebest* from the overall system to update the position of individuals by using (11) and (12). In PHIMA-FGA, too, Elitism-Random Selection is used instead of Roulette Wheel Selection.

D. PHIMA-FPGA-CO

The co-evolution system is used on two sub-GA islands of the PHIMA-FPGA to build the PHIMA-FPGA-Co algorithm. For each sub-GA island, the entire population is equally divided into two smaller groups based on their dimensions. The addition of co-evolution is expected to improve the performance of the algorithm for high-dimensional problems. PHIMA-FPGA and PHIMA-FPGA-Co share the same parameters in this study.

IV. EXPERIMENTS

In the experiments, six objective functions [38] with dimensions of 6, 20 and 50, have been used as shown in Table 1. For every algorithm, each objective function is tested 20 times to obtain an average result for robustness.

Instead of error, the fitness has been used to indicate the result quality. The fitness can be computed from the error using (13). Using the fitness can decrease the weight of extremely bad results and therefore reduce the noise during the result analysis.

$$fitness = \frac{100}{1 + error} \tag{13}$$

A. EXPERIMENT 1

In this paper, PHI architecture is expected to improve the performance of island GAs by introducing PSO and/or FWA to the island GAs. In Experiment 1, the performance of PHIMA-PGA and PHIMA-FGA is compared to that of the 3-island GA (3iGA) to prove this idea. In all three algorithms, the total population size is equal to the dimensions $D \times 72$. The total population size is equally divided among the three sub islands, making the population size of each island $D \times 24$. The algorithms terminate on completing 500 iterations. The rest of the parameters are listed in Table 2. These parameter

TABLE 1. Objective functions.

Ackley	$f(x)=20-20\,exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\right)+e-exp\left(\frac{1}{N}\sum_{i=1}^{N}cos(2\pi x_i)\right)$ $x_i\in[-32.768, 32.768]$ $x_i=0.$ $\forall i\in\{1\ldots N\}, f(x)=0.$
SumSquare	$f(x)=\sum_{i=1}^{N}ix_i^2$ $x_i\in[-5.12, 5.12]$ $x_i=0.$ $\forall i\in\{1\ldots N\}, f(x)=0.$
Schwefel	$f(x)=418.9828872724339\cdot N-\sum_{i=1}^{N}x_i sin\left(\sqrt{\lvert x_i\rvert}\right)$ $x_i\in[100, 500]$ $x_i=420.96874636.$ $\forall i\in\{1\ldots N\}, f(x)=0.$
Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{N}x_i^2-\prod_{i=1}^{N}cos\left(\frac{x_i}{\sqrt{i}}\right)+1$ $x_i\in[-100, 100]$ $x_i=0.$ $\forall i\in\{1\ldots N\}, f(x)=0.$
Bohacke-vsky_1	$f(x)=\sum_{i=1}^{N-1}(x_i^2+2x_{i+1}^2-0.3\,cos(3\pi x_i)-0.4\,cos(4\pi x_i)+0.7)$ $x_i\in[-100, 100]$ $x_i=0.$ $\forall i\in\{1\ldots N\}, f(x)=0.$
Schaffer	$f(x)=\sum_{i=1}^{N-1}(x_i^2+x_{i+1}^2)^{0.25}\cdot\left[sin^2\left(50\cdot(x_i^2+x_{i+1}^2)^{0.10}\right)+10\right]$ $x_i\in[-100, 100]$ $xi=0.$ $\forall i\in\{1\ldots N\}, f(x)=0.$

TABLE 2. Experiment's parameters.

Algorithm	Parameter	Value
GA	L_x	15/dimension
GA	p_c	70%
GA	p_m	10%
PSO	w	0.5
PSO	c_1	1.5
PSO	c_2	1
PSO	c_3	1
PSO	Thr_{pso_max}	$\frac{range_{up}-range_{down}}{20}$
PSO	Thr_{pso_min}	0.01
FWA	$\hat{A}$	100
FWA	M	10
FWA	a	0.1
FWA	b	0.9
FWA	Thr_{fwa}	0.01

TABLE 3. Experiment 1 fitness.

	6 Dimension		
	PGA	FGA	3iGA
Ackley	99.6	99.66	95.99
SumSquares	100	100	100
Schwefel	100	99.9	45.01
Greiwank	95.03	95.52	90.31
Bohachevsky_1	99.96	99.84	95.4
Schaffer	79.6	67.88	43.8
	20 Dimensions		
	PGA	FGA	3iGA
Ackley	99.6	79.85	39.62
SumSquares	100	100	100
Schwefel	81.81	98.84	7.14
Greiwank	97.32	98.09	97.42
Bohachevsky_1	97.72	99.19	91.48
Schaffer	35.14	16.05	8.54
	50 Dimensions		
	PGA	FGA	3iGA
Ackley	49.99	47.21	39.92
SumSquares	100	99.9	99.99
Schwefel	90.19	99.9	0.59
Greiwank	99.27	99.93	99.86
Bohachevsky_1	95.04	95.95	87.9
Schaffer	10.67	2.85	1.73

settings are also used in Experiments 2 and 3. The results of Experiment 1 are shown in Fig. 5 to Fig. 22. The final fitness values for each algorithm after completing 500 iterations are listed in Table 3.

The convergence speed and result quality of PHIMA-PGA and PHIMA-FGA are better than 3iGA in most cases, except for the 50-dimension Griewank (Fig. 19), 20-dimention SumSquares (Fig. 16) and 50-dimension SumSquares (Fig. 22). The result quality of PHIMA-PGA on 50-dimension Griewank is slightly inferior to that of 3iGA, the convergence speed of PHIMA-FGA on 20-dimension and 50-dimension SumSquares is slower than that of 3iGA. In addition, the result quality of PHIMA-FGA on 50-dimension SumSquares is worse than that of 3iGA. The SumSquares is a single-pole, smooth and symmetric function. Its simple terrain can be used to test the convergence speed of the algorithms. The slow convergence speed of PHIMA-FGA

TABLE 4. Experiment 2 fitness.

	6 Dimensions		
	FPGA	FPGA-Co	4iGA
Ackley	99.7	99.66	78.92
SumSquares	100	100	100
Schwefel	100	100	47.3
Greiwank	96.13	97.23	91.36
Bohachevsky_1	99.94	99.93	98.32
Schaffer	77.51	77.59	43.55
	20 Dimensions		
	FPGA	FPGA-Co	4iGA
Ackley	99.6	99.6	47.38
SumSquares	100	100	100
Schwefel	99.83	99.94	8.24
Greiwank	97.63	98.55	92.65
Bohachevsky_1	99.19	99.18	93.38
Schaffer	35.52	34.98	12.12
	50 Dimensions		
	FPGA	FPGA-Co	4iGA
Ackley	49.66	93.61	39.39
SumSquares	100	100	99.96
Schwefel	92.03	98.29	0.77
Greiwank	99.22	99.64	97.06
Bohachevsky_1	96.46	96.46	80.71
Schaffer	11.63	15.17	2.6

TABLE 5. Experiment 3 fitness.

	6 Dimensions		
	PGA	FGA	FPGA
Ackley	99.6	99.66	99.7
SumSquares	100	100	100
Schwefel	100	99.9	100
Greiwank	95.03	95.52	96.13
Bohachevsky_1	99.96	99.84	99.94
Schaffer	79.6	67.88	77.51
	20 Dimensions		
	PGA	FGA	FPGA
Ackley	99.6	79.85	99.6
SumSquares	100	100	100
Schwefel	81.81	98.84	99.83
Greiwank	97.32	98.09	97.63
Bohachevsky_1	97.72	99.19	99.19
Schaffer	35.14	16.05	35.52
	50 Dimensions		
	PGA	FGA	FPGA
Ackley	49.99	47.21	49.66
SumSquares	100	99.9	100
Schwefel	90.19	99.9	92.03
Greiwank	99.27	99.93	99.22
Bohachevsky_1	95.04	95.95	96.46
Schaffer	10.67	2.85	11.63

on the SumSquares function indicates that the sub-FWA island will have a negative effect on the convergence speed.

Moreover, PHIMA-PGA shows a better convergence speed than PHIMA-FGA for all functions in all dimensions. Moreover, in most of the proposed cases, the result quality of PHIMA-PGA is better than that of PHIMA-FGA, except 6-dimension Ackley (Fig. 5) and Griewank (Fig. 7), 20-dimension Bohachevsky_1 (Fig. 12), Griewank (Fig. 13) and Schwefel (Fig. 15), 50-dimension Bohachevsky_1 (Fig. 18), Griewank (Fig. 19) and Schwefel (Fig. 21). Griewank and Schwefel are functions that have large number of local optimal solutions. Bohachevsky_1 function has a large number of vibrations caused by the term "*0.3cos* ($3\pi x_i$)*-0.4cos(*$4\pi x_i$*)*". This type of oscillations can be considered as local optima where the algorithm easily gets stuck.

B. EXPERIMENT 2

For the same purpose as Experiment 1, in Experiment 2, PHIMA-FPGA and PHIMA-FPGA-Co are compared with 4-island GA (4iGA). In all three algorithms, the total population size is equal to D×72, and the total population size is equally divided among the four sub-islands. Therefore, the population size for each island is equal to D×18. The algorithms stop at the 500th iteration. The rest of the parameters are listed in Table 2. The results of Experiment 2 are shown in Fig. 23 to Fig. 40. The fitness results are presented in Table 4.

Both PHIMA-FPGA and PHIMA-FPGA-Co show an apparently better performance compared to the 4-island GA in terms of convergence speed and result quality for all the proposed cases. Compared to that of PHIMA-FPGA, PHIMA-FPGA-Co shows better performance in terms of convergence speed for all the proposed 6-dimension and 20-dimension functions and a significantly improved performance on convergence rate for 50-dimension functions. Moreover, the result quality of PHIMA-FPGA-Co is slightly better than that of PHIMA-FPGA in most cases, except 6-dimension Ackley (Fig. 23), 6-dimension Bohachevsky_1 (Fig. 24), 20-dimension Bohachevsky_1 (Fig. 30), 20-dimension Schaffer (Fig. 32) and 50-dimenson Bohachevsky_1 (Fig. 36). Moreover, the result quality has significantly improved for 50-dimension Ackley (Fig. 35).

C. EXPERIMENT 3

In Experiment 3, the results of PHIMA-FPGA from Experiment 2 are compared with those of PHIMA-PGA and PHIMA-FGA from Experiment 1. The comparison results of Experiment 3 are shown in Fig. 41 to Fig. 58. The fitness results are presented in Table 5.

In terms of 6-dimension functions, except for the 6-dimension Schaffer (Fig. 44), the graphs of PHIMA-FPGA and PHIMA-PGA nearly coincide. For 20-dimension functions, the convergence speed of PHIMA-FPGA is slightly slower than that of PHIMA-PGA, but its result quality is better than that of PHIMA-PGA for all the proposed 20-dimension functions. For 50-dimension functions, the convergence speed of PHIMA-FPGA is faster than that of PHIMA-PGA, and shows better result quality on Bohachevsky_1 (Fig. 54), Schaffer (Fig. 56) and Schwefel (Fig. 57) functions. However, its result quality on 50-dimension Ackley (Fig. 53), Griewank (Fig. 55) and SumSquares (Fig. 58) is slightly inferior to that of PHIMA-PGA.

Compared with PHIMA-PGA and PHIMA-FPGA, PHIMA-FGA has the slowest convergence speed for all the proposed cases. However, it has the best result quality on 20-dimension Griewank (Fig. 49), 50-dimension Griewank (Fig. 55) and 50-dimension Schwefel (Fig. 57).

D. RESULT AND ANALYSIS

All the novel hybrid algorithms show better result quality compared to the traditional island GA. Judging from the results, it can be concluded that using the proposed PHI architecture to hybrid island GAs with PSO and/or FWA can significantly improve the algorithm performance compared to the classic island GA. Also, the character of the PHIMA varies with the changes of algorithms combinations. The PHIMA-PGA has a significant advantage in terms of convergence speed, while PHIMA-FPGA has the best comprehensive performance on the result quality. Moreover, one merit for the PHIMA is, because each island in the island model is relatively independent, it is easy to apply improvement approach to some of the islands without influence the others. The comparison between PHIMA-FPGA and PHIMA-FPGA-Co in the experiment 2 proves the co-evolution structure can improve the convergence speed on all the proposed benchmark functions. It is also found to improve the result quality for most of them.

V. CONCLUSION

To address the issue of premature convergence in multimodal and high-dimensional problems, this study proposed a novel hybridization approach that uses the island model to combine different meta-heuristic optimization algorithms in parallel. The proposed approach is called Parallel Hybrid Island (PHI) architecture and Parallel Hybrid Island Metaheuristic Algorithms (PHIMA). In the experiments, four newly developed PHIMA algorithms, namely, PHIMA-PGA, PHIMA-FGA, PHIMA-FPGA, and PHIMA-FPGA-Co are tested on six benchmark functions: Ackley, Sumsquare, Schwefel, Griewank, Bohachevsky_1 and Schaffer in 6, 20 and 50 dimensions. The performances of these four algorithms are compared with one another and with that of the traditional island GA. The experiment result shows that the proposed PHI architecture is an effective improvement approach compared to the traditional island GA.

PHI architecture owes its success to the judicious combination of the salient characteristics of different algorithms that contributes to the population diversity. Its parallel search mechanism simultaneously handles the balance between exploration and exploitation. The PHI architecture proposed a new concept for hybridizing different metaheuristic algorithms. The independent computation among the islands makes it easy to hybridize different metaheuristic algorithms (and their variants) or add different structures, such as co-evolution and memetic algorithm. One of the greatest advantages of this novel parallel hybrid method is that it puts no limit on the number of algorithms that can be hybridized.

The study reported the results based on the values of the hyperparameters obtained through a brief pilot study. Optimization of the hyper-parameters [39]–[41] such as M_{ratio}, $M_{interval}$, and so on, although has the potential of further improving the performance of the proposed parallel hybrid algorithm, is itself not the main objective of this study.

As a sequel to this study, three further improvements are expected. First, in this study, the co-evolution structure is applied only to the sub-GA islands. The co-evolution structure can also be experimented on sub-PSO and sub-FWA islands. Second, it is possible to find a better island structure to combine these three algorithms. Finally, through experimentation, it is found that the new hybrid algorithms show different characteristics depending on the individual algorithms used in the parallel hybrid structure. Therefore, it is expected to further extend this idea to combine a variety of algorithms, especially the state-of-the-art algorithms of the moment, to develop a more robust PHIMA that can handle real-life higher dimensional problems.

REFERENCES

[1] N. Aldawoodi, *An Approach to Designing an Unmanned Helicopter Autopilot Using Genetic Algorithms and Simulated Annealing*. Tampa, FL, USA: Univ. of South Florida, 2008, p. 99.

[2] I. Persson and S. D. Iwnick, "Optimisation of railway wheel profiles using a genetic algorithm," *Vehicle Syst. Dyn.*, vol. 41, pp. 517–526, Jan. 2004.

[3] N. Hans, S. Mahajan, and S. Omkar, "Big data clustering using genetic algorithm on Hadoop mapreduce," *Int. J. Sci. Technol. Res.*, vol. 4, pp. 58–62, Apr. 2015.

[4] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Hoboken, NJ, USA: Wiley, 2001.

[5] C. W. Ho, K. H. Lee, and K. S. Leung, "A genetic algorithm based on mutation and crossover with adaptive probabilities," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 1, Jul. 1999, pp. 768–775.

[6] N. M. Razali and J. Geraghty, "Genetic algorithm performance with different selection strategies in solving TSP," in *Proc. World Congr. Eng.*, vol. 2, no. 1. Hong Kong: International Association of Engineers, Jul. 2011, pp. 1–6.

[7] J. Morrison and F. Oppacher, "A general model of co-evolution for genetic algorithms," in *Artificial Neural Nets and Genetic Algorithms*. Vienna, Austria: Springer, 1999.

[8] J. Robinson, S. Sinton, and Y. Rahmat-Samii, "Particle swarm, genetic algorithm, and their hybrids: Optimization of a profiled corrugated horn antenna," in *Proc. IEEE Int. Symp. Antennas Propag.*, vol. 1, San Antonio, TX, USA, Jun. 2002, pp. 314–317.

[9] H. Selamat and S. D. A. Bilong, "Optimal controller design for a railway vehicle suspension system using particle swarm optimization," in *Proc. 9th Asian Control Conf. (ASCC)*, Jun. 2013, pp. 1–5.

[10] B. Qolomany, M. Maabreh, A. Al-Fuqaha, A. Gupta, and D. Benhaddou, "Parameters optimization of deep learning models using particle swarm optimization," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2017, pp. 1285–1290.

[11] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *Proc. Int. Conf. Swarm Intell.* Berlin, Germany: Springer, Jun. 2010, pp. 355–364.

[12] A. M. Imran and M. Kowsalya, "A new power system reconfiguration scheme for power loss minimization and voltage profile enhancement using fireworks algorithm," *Int. J. Elect. Power Energy Syst.*, vol. 62, pp. 312–322, Nov. 2014.

[13] M. Tuba, N. Bacanin, and A. Alihodzic, "Multilevel image thresholding by fireworks algorithm," in *Proc. 25th Int. Conf. Radioelektronika (RADIOELEKTRONIKA)*, Apr. 2015, pp. 326–330.

[14] N. Bacanin and M. Tuba, "Fireworks algorithm applied to constrained portfolio optimization problem," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 1242–1249.
[15] H. M. Pandey, A. Chaudhary, and D. Mehrotra, "A comparative review of approaches to prevent premature convergence in GA," *Appl. Soft Comput.*, vol. 24, pp. 1047–1077, Nov. 2014.
[16] S. Malik and S. Wadhwa, "Preventing premature convergence in genetic algorithm using DGCA and elitist technique," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 4, no. 6, p. 410, 2014.
[17] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, and M. Valdez, "Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic," *Expert Syst. Appl.*, vol. 40, no. 8, pp. 3196–3206, Jun. 2013.
[18] W. Deng, H. Zhao, X. Yang, J. Xiong, M. Sun, and B. Li, "Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment," *Appl. Soft Comput.*, vol. 59, pp. 288–302, Oct. 2017.
[19] S. Zheng, A. Janecek, and Y. Tan, "Enhanced fireworks algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 2069–2077.
[20] G. Gong, Q. Deng, R. Chiong, X. Gong, and H. Huang, "An effective memetic algorithm for multi-objective job-shop scheduling," *Knowl.-Based Syst.*, vol. 182, Oct. 2019, Art. no. 104840.
[21] Z. Beheshti and S. M. H. Shamsuddin, "A review of population-based meta-heuristic algorithms," *Int. J. Adv. Soft Comput. Appl.*, vol. 5, no. 1, pp. 1–35, 2013.
[22] A. M. Manasrah and H. B. Ali, "Workflow scheduling using hybrid GA-PSO algorithm in cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–16, Jan. 2018.
[23] F. A. Şenel, F. Gökçe, A. S. Yüksel, and T. Yiğit, "A novel hybrid PSO–GWO algorithm for optimization problems," *Eng. Comput.*, vol. 35, no. 4, pp. 1359–1373, Oct. 2019.
[24] J. H. Tam, Z. C. Ong, Z. Ismail, B. C. Ang, S. Y. Khoo, and W. L. Li, "Inverse identification of elastic properties of composite materials using hybrid GA-ACO-PSO algorithm," *Inverse Problems Sci. Eng.*, vol. 26, no. 10, pp. 1432–1463, Oct. 2018.
[25] A. A. K. Taher and S. M. Kadhim, "Improvement of genetic algorithm using artificial bee colony," *Bull. Electr. Eng. Informat.*, vol. 9, no. 5, pp. 2125–2133, Oct. 2020.
[26] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and L. M. Wang, "An improved GA and a novel PSO-GA-based hybrid algorithm," *Inf. Process. Lett.*, vol. 93, no. 5, pp. 255–261, Mar. 2005.
[27] A. A. Gozali, B. Kurniawan, W. Weng, and S. Fujimura, "Solving university course timetabling problem using localized island model genetic algorithm with dual dynamic migration policy," *IEEJ Trans. Electr. Electron. Eng.*, vol. 15, no. 3, pp. 389–400, Mar. 2020.
[28] J. M. Palomo-Romero, L. Salas-Morera, and L. García-Hernández, "An island model genetic algorithm for unequal area facility layout problems," *Expert Syst. Appl.*, vol. 68, pp. 151–162, Feb. 2017.
[29] X. Sun, L.-F. Lai, P. Chou, L.-R. Chen, and C.-C. Wu, "On GPU implementation of the island model genetic algorithm for solving the unequal area facility layout problem," *Appl. Sci.*, vol. 8, no. 9, p. 1604, Sep. 2018.
[30] S.-H. Park, H.-T. Kim, and K.-T. Kim, "Stepped-frequency ISAR motion compensation using particle swarm optimization with an island model," *Prog. Electromagn. Res.*, vol. 85, pp. 25–37, 2008.
[31] H. Abadlia, N. Smairi, and K. Ghedira, "Particle swarm optimization based on island models," in *Proc. Genet. Evol. Comput. Conf. Companion*, Jul. 2017, pp. 49–50.
[32] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.
[33] Q. Meng, J. Wu, J. Ellis, and P. J. Kennedy, "Dynamic island model based on spectral clustering in genetic algorithm," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 1724–1731.
[34] C.-C. Li, C.-H. Lin, and J.-C. Liu, "Parallel genetic algorithms on the graphics processing units using island model and simulated annealing," *Adv. Mech. Eng.*, vol. 9, no. 7, 2017, Art. no. 1687814017707413.
[35] E. Osaba, E. Onieva, R. Carballedo, F. Diaz, A. Perallos, and X. Zhang, "A multi-crossover and adaptive island based population algorithm for solving routing problems," *J. Zhejiang Univ. Sci. C*, vol. 14, no. 11, pp. 815–821, Nov. 2013.
[36] F. D. Wihartiko, H. Wijayanti, and F. Virgantari, "Performance comparison of genetic algorithms and particle swarm optimization for model integer programming bus timetabling problem," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 332, Mar. 2018, Art. no. 012020.
[37] V. Kachitvichyanukul, "Comparison of three evolutionary algorithms: GA, PSO, and DE," *Ind. Eng. Manage. Syst.*, vol. 11, no. 3, pp. 215–223, Sep. 2012.
[38] (Nov. 14, 2019). *Benchmarks—DEAP 1.3.0 Documentation*. [Online]. Available: https://deap.readthedocs.io/en/master/api/benchmarks.html
[39] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 201–216, Apr. 2020.
[40] C. Ansotegui, M. Sellmann, and K. Tierney, "A gender-based genetic algorithm for the automatic configuration of algorithms," in *Principles and Practice of Constraint Programming—CP 2009*, I. P. Gent, Ed. Berlin, Germany: Springer, 2009, pp. 142–157.
[41] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*, C. A. C. Coello, Ed. Berlin, Germany: Springer, 2011, pp. 507–523.

JIAWEI LI (Member, IEEE) received the B.S. degree in electronic and electrical engineering and the M.S. degree in electrical power systems from the University of Birmingham, Birmingham, U.K., in 2015 and 2016, respectively, and the second M.S. degree in green science and engineering from Sophia University, Tokyo, Japan, where he is currently pursuing the Ph.D. degree in green science and engineering.

From 2020 to 2021, he was a Research Assistant at the Faculty of Science and Technology, Sophia University. His research interests include the optimization algorithms improvement and application and neuro-evolution of artificial neural networks.

TAD GONSALVES received the B.S. degree in theoretical physics, the M.S. degree in astrophysics, and the Ph.D. degree in information systems from Sophia University, Tokyo, Japan.

He is a Full Professor with the Department of Information and Communication Sciences, Faculty of Science and Technology, Sophia University. He has published 100 papers in international conferences and journals. He is the author of the book *Artificial Introduction: A Non-Technical Introduction* (Sophia University Press, Tokyo, Japan, 2017) and coauthor of *Artificial Intelligence for Business Optimization: Research and Applications* (CRC press, London, 2021). His research interests include bio-inspired optimization techniques and the application of deep learning techniques to diverse problems, such as autonomous driving, drones, digital art and music, and computational linguistics. His research laboratory (https://www.gonken.tokyo/) in Tokyo specializes in applications of deep learning and multi-GPU computing.

• • •